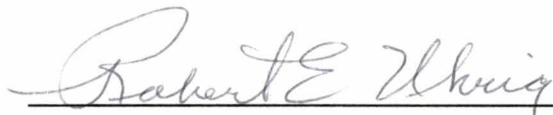
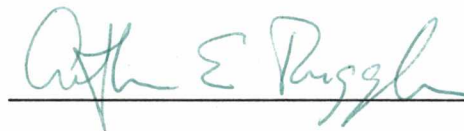


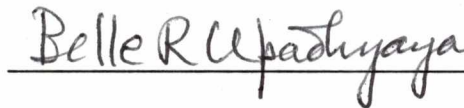
To the Graduate Council:

I am submitting herewith a thesis written by Vaclav Hojny entitled "Emulation of Probabilistic-Safety-Assessment Code in a Nuclear Power Plant Risk Monitor by Artificial Neural Networks". I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Nuclear Engineering.


Robert E. Uhrig, Major Professor

We have read this thesis
and recommend its acceptance:





Accepted for the Council:


Associate Vice Chancellor
and Dean of the Graduate School

**EMULATION OF
PROBABILISTIC-SAFETY-ASSESSMENT CODE
IN A NUCLEAR POWER PLANT RISK MONITOR
BY ARTIFICIAL NEURAL NETWORKS**

A Thesis

Presented for the

Master of Science

Degree

The University of Tennessee at Knoxville

Vaclav Hojny

August 1995

Acknowledgment

It is my great pleasure to thank those who helped me throughout the course of my study and made this work possible.

First of all, I would like to thank my major professor, Dr. Robert E. Uhrig for his supervision of my work and all his advice and support. I would also like to thank the other members of my committee, Dr. Belle R. Upadhyaya and Dr. Arthur E. Ruggles for their suggestions and for serving on the committee. My thanks also goes to Dr. J. Wesley Hines for his consultations and comments, and to Dr. Bojan Tomic of the International Atomic Energy Agency in Vienna, Austria, for keeping me informed about the latest development in the field of PSA and risk monitors.

The computer code Tree-Master used for evaluation of the test PSA model fault tree was kindly provided to the Nuclear Engineering Department of the University of Tennessee at Knoxville by its author Dr. Antonin Wild. This help is greatly appreciated.

This research was partly sponsored by the Tennessee Valley Authority under research contract No. TV-93732V. This support is gratefully acknowledged.

Abstract

This work represents a feasibility study of utilizing artificial neural networks for quick evaluation of Probabilistic Safety Assessment (PSA) models in the risk monitors of nuclear power plants.

The study provides a brief overview of methodology and applications of the PSA, and a detailed description of the tasks and structure of the risk monitors and their individual parts. It identifies a possible role of the artificial neural networks in the risk monitors and gives an overview of the paradigms suitable for this application. The approach used and the results obtained when using these paradigms for evaluation of a small test PSA model and their expected performance for the full-scope PSA models are discussed in detail.

The results obtained demonstrate that though the artificial neural networks would be probably capable of performing the desired evaluation of the PSA models, their training would require pre-computation of such a vast volume of data that it makes their proposed utilization impractical. The study concludes that the investigated application of the artificial neural networks in the risk monitor is not feasible in the present state of knowledge.

Table of Contents

1. INTRODUCTION	1
2. PROBABILISTIC SAFETY ASSESSMENT	3
2.1. PSA Methodology	3
2.1.1. Identification of Possible Accident Scenarios	3
2.1.2. Assessment of Frequencies of the Accident Sequences	6
2.1.3. Assessment of Consequences of the Accident Sequences	9
2.1.4. Final Assessment of the Risk	10
2.2. Applications of the PSA	11
3. RISK MONITOR	13
3.1. Configuration Control	13
3.2. State-of-the-Art of the Risk Monitor	17
3.3. Structure of the Risk Monitor	20
3.3.1. Knowledge Base of the Risk Monitor	22
3.3.1.1. Specificities of Risk-Monitor PSA Models	23
3.3.1.2. Necessary Modifications of the Base-Line PSA Models	27
3.3.1.3. Other Information Stored in the Knowledge Base	30
3.3.2. Input Interface of the Risk Monitor	32
3.3.2.1. Manual Update of the Risk Monitor	33
3.3.2.2. Automatic Update of the Risk Monitor	36
3.3.3. Inference Engine of the Risk Monitor	37
3.3.3.1. Adjustment of the PSA Models	38
3.3.3.2. Evaluation of the Adjusted PSA Models	42
3.3.3.3. Derivation of Risk Monitor Recommendations	43
3.3.4. Output Interface of the Risk Monitor	46
3.4. Verification and Validation of the Risk Monitor	50
3.5. Possible Role of Artificial Neural Networks in Risk Monitor	54
4. ARTIFICIAL NEURAL NETWORKS	60
4.1. General Description	60
4.2. Paradigms Suitable for Utilization in Risk Monitor	64
4.2.1. Backpropagation Network	65
4.2.2. Functional Link Network	68
4.2.3. Modular Neural Network	69
4.2.4. Radial Basis Function Network	75
4.2.5. Generalized Regression Neural Network	78

5. QUANTIFICATION OF PSA MODELS WITH ARTIFICIAL NEURAL NETWORKS	83
5.1. Test PSA Model	83
5.2. Problem Implementation	86
5.2.1. Risk Parameters of Interest	89
5.2.1.1. Core-Damage Frequency Calculation	89
5.2.1.2. Risk Control Importances Calculation	90
5.2.1.3. Risk Reduction Importances Calculation	93
5.2.2. Gathering the Data for ANN Training	95
5.2.3. Preprocessing the Data for ANN Training	100
5.2.3.1. Preprocessing of the Input Vectors	100
5.2.3.2. Preprocessing of the Output Vectors	102
5.3. Application of Selected ANN Paradigms	104
5.3.1. Backpropagation Network	105
5.3.1.1. BPN Design	106
5.3.1.2. Data Preprocessing for the BPN	109
5.3.1.3. BPN Training	110
5.3.1.4. Verification and Validation of the Trained BPN	126
5.3.1.5. BPN Knowledge Analysis	139
5.3.2. Functional Link Network	151
5.3.3. Modular Neural Network	152
5.3.3.1. MNN Design	153
5.3.3.2. Data Preprocessing for the MNN	155
5.3.3.3. MNN Training	156
5.3.3.4. Verification and Validation of the Trained MNN	158
5.3.4. Radial Basis Function Network	161
5.3.4.1. RBFN Design	164
5.3.4.2. Data Preprocessing for the RBFN	165
5.3.4.3. RBFN Training	166
5.3.4.4. Verification and Validation of the Trained RBFN	169
5.3.5. Generalized Regression Neural Network	175
5.3.5.1. GRNN Design	175
5.3.5.2. Data Preprocessing for the GRNN	177
5.3.5.3. GRNN Training	178
5.3.5.4. Verification and Validation of the Trained GRNN	179
5.4. Evaluation of ANNs Performance on the Test PSA Model	185
5.4.1. Comparison of the Individual ANN Paradigms	185
5.4.2. Reasons for the Difficulties Encountered by Individual ANN Paradigms	193
5.4.3. General Reason for the Observed ANN Difficulties	198
5.5. Expected ANNs Performance on a Full-Scope PSA Model	201

6. CONCLUSIONS AND RECOMMENDATIONS	203
REFERENCES	206
Vita	215

List of Tables

Table 1 - Basic events of the test PSA model	88
Table 2 - Configurations included in the training set .	113
Table 3 - Configurations included in the testing set .	119
Table 4 - Training configurations with an error of the computed core-damage frequency exceeding 5% .	131
Table 5 - Testing configurations with an error of the computed core-damage frequency exceeding 10%	134
Table 6 - Weights of the hidden units	141
Table 7 - Weights of the output unit	143

List of Figures

Figure 1 - Steps of the Probabilistic Safety Assessment	4
Figure 2 - Structure of the risk monitor	21
Figure 3 - Evaluation of the PSA model	43
Figure 4 - Topology of the backpropagation network . .	66
Figure 5 - Processing unit of the backpropagation network	67
Figure 6 - Topology of the modular neural network . . .	70
Figure 7 - Output unit of the MNN gating network . . .	72
Figure 8 - Pattern unit of the radial basis function network	76
Figure 9 - Topology of the generalized regression neural network	79
Figure 10 - System event tree for the test problem . .	84
Figure 11 - Test PSA model	87
Figure 12 - Learning curves of the BPN	121
Figure 13 - Absolute errors of the BPN actual output values on the training set	128
Figure 14 - Percentile errors of the core-damage frequency values computed from the BPN outputs for the training configurations . .	130
Figure 15 - Absolute errors of the BPN actual output values on the testing set	132
Figure 16 - Percentile errors of the core-damage frequency values computed from the BPN outputs for the testing configurations . .	133
Figure 17 - RAWs of individual basic events as computed by the BPN for the configuration with only the event number one guaranteed	137

Figure 18 - RAWs of individual basic events as computed by the BPN for the configuration with the events number one and number four guaranteed	138
Figure 19 - Activations of hidden units for the nominal configuration	145
Figure 20 - Activations of hidden units for the configuration with all basic events guaranteed	145
Figure 21 - Activations of the hidden units for the configurations yielding core-damage frequencies of the order of magnitude of 10^{-6}	147
Figure 22 - Activations of the hidden units for the configurations yielding core-damage frequencies of the order of magnitude of 10^{-5}	148
Figure 23 - Activations of the hidden units for the configurations yielding core-damage frequencies of the order of magnitude of 10^{-4}	149
Figure 24 - Percentile errors of the core-damage frequency values computed from the MNN outputs for the training configurations . .	159
Figure 25 - Percentile errors of the core-damage frequency values computed from the MNN outputs for the testing configurations . .	160
Figure 26 - RAWs of individual basic events as computed by the MNN for the configuration with only the event number one guaranteed	162
Figure 27 - RAWs of individual basic events as computed by the MNN for the configuration with the events number one and number four guaranteed	163
Figure 28 - Percentile errors of the core-damage frequency values computed from the RBFN outputs for the training configurations . .	170

Figure 29 - Percentile errors of the core-damage frequency values computed from the RBFN outputs for the testing configurations . .	171
Figure 30 - RAWs of individual basic events as computed by the RBFN for the configuration with only the event number one guaranteed . . .	173
Figure 31 - RAWs of individual basic events as computed by the RBFN for the configuration with the events number one and number four guaranteed	174
Figure 32 - Percentile errors of the core-damage frequency values computed from the GRNN outputs for the training configurations . .	180
Figure 33 - Percentile errors of the core-damage frequency values computed from the GRNN outputs for the testing configurations . .	181
Figure 34 - RAWs of individual basic events as computed by the GRNN for the configuration with only the event number one guaranteed	183
Figure 35 - RAWs of individual basic events as computed by the GRNN for the configuration with the events number one and number four guaranteed	184
Figure 36 - Overall percentile errors of the core-damage frequency values computed by the selected ANN paradigms for different sets of the model configurations	187
Figure 37 - RAWs of individual basic events as computed by different ANNs for the configuration with only the event number one guaranteed	191
Figure 38 - RAWs of individual basic events as computed by different ANNs for the configuration with the events number one and number four guaranteed	192

List of Abbreviations

ANN	- artificial neural network
BPN	- backpropagation network
ECCS	- emergency core cooling system
ESAS	- engineered safeguards actuating system
FLN	- functional link network
GRNN	- generalized regression neural network
MNN	- modular neural network
NPP	- nuclear power plant
PSA	- Probabilistic Safety Assessment
RAW	- risk achievement worth
RMS	- root-mean-squares error
RRW	- risk reduction worth
RBFN	- radial basis function network

1. INTRODUCTION

The Probabilistic Safety Assessment (PSA) has already reached status of a generally accepted methodology and PSA studies have been completed or are being carried out for practically all nuclear power plants (NPPs) in the world. Role of the PSA in the licensing of NPPs is gradually increasing; in the USA, e.g., the Individual Plant Examinations, which include PSA as their integral part, are requested by the U.S. Nuclear Regulatory Commission for each operating NPP [1]. Recently a strong potential of the PSA to provide help in making NPPs safer and more reliable has been also recognized. This has resulted in a current effort to make better use of that large amount of information accumulated by the PSA studies and to transfer it from the research and development organizations and departments to everyday use in NPPs. An ultimate goal seems to be turning of the rather static current PSAs into a dynamic tool usable for operational decisions in the NPPs, called a risk monitor.

The objective of the work presented is to contribute to this effort. The study represents a feasibility study of utilizing artificial neural networks (ANNs) in the risk monitor for emulation of calculations of risk parameters using a special PSA code. The ANNs, if successful in this application, will conduct these calculations much faster than

the original PSA code enabling thus a nearly on-line monitoring of the contemporary risk.

The study consists of six chapters; this introduction being the first one. The second chapter provides a necessary background for the study, giving a basic overview of methodology and applications of the PSA. The third chapter describes in detail structure and features of the risk monitor and identifies a possible role of the ANNs in it. The ANNs are briefly overviewed in the fourth chapter with special emphasis being on the ANN paradigms which can be used for function approximation applications. The fifth chapter, representing a core of the study, describes in detail investigations conducted to explore performance of the ANNs in emulation of the PSA code calculations and feasibility of their utilization in the risk monitor and the obtained results. The conclusions of this feasibility study and the recommendations for a possible future work are given in the sixth chapter.

2. PROBABILISTIC SAFETY ASSESSMENT

The Probabilistic Safety Assessment* (PSA) is a complex methodology used for quantitative assessment of the risk arising from operation of complicated technological plants, e.g., NPPs or chemical plants [2,3].

2.1. PSA Methodology

The amount of risk associated with an accident depends, in general, on the frequency and on the consequences of this accident. To assess the risk, the PSA needs to comprise identification and delineation of combinations of events which - if they occur - lead to a severe accident, assessment of frequencies of these event combinations, and assessment of their consequences - see Figure 1.

2.1.1. Identification of Possible Accident Scenarios

The first step in the identification of possible accident scenarios is determination of accident initiating events, i.e.

* There is an alternative name used for this methodology in literature: Probabilistic Risk Assessment (PRA). Both these names have exactly the same meaning. Nevertheless, the name PSA is gradually prevailing because the adjective "Safety" is considered to be more suitable for achieving public acceptance of NPPs. The term PSA is used on principle in this study.

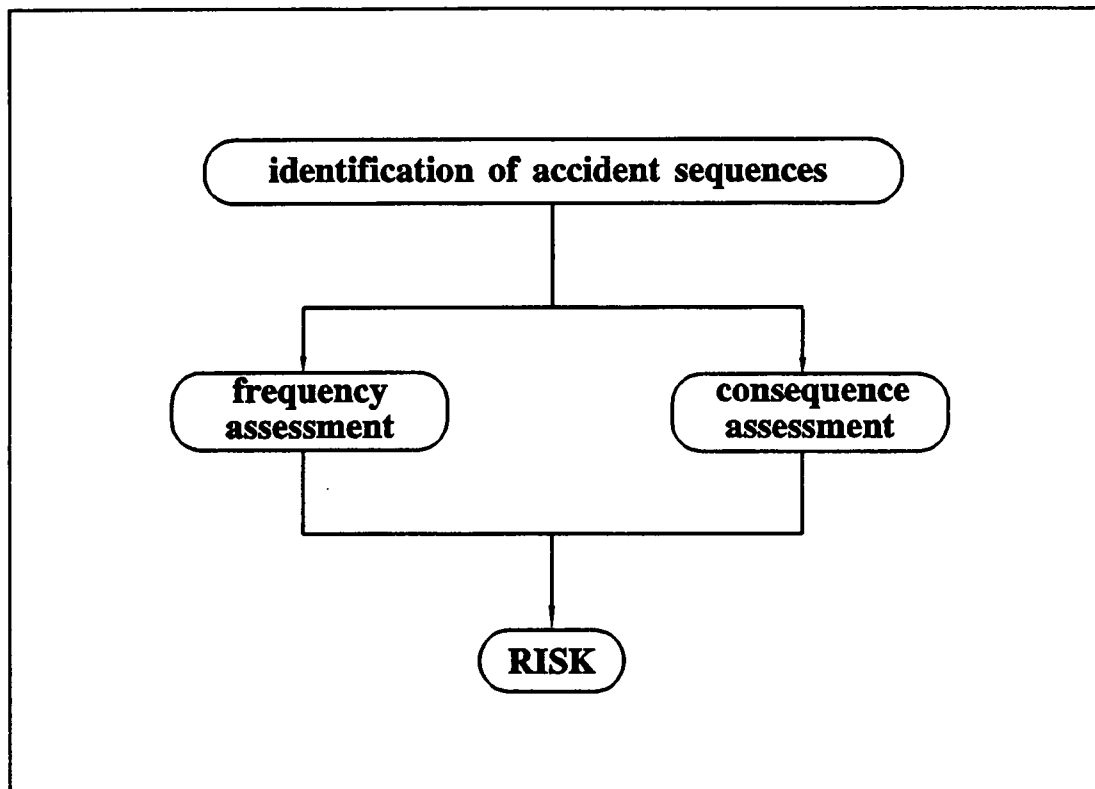


Figure 1 - Steps of the Probabilistic Safety Assessment

the events which may be at the start of a sequence of events leading to an accident. Normally, only the initiating events which may result in severe reactor core damage are considered in the PSAs. A general principle in this stage is to consider all possible initiating events regardless of anticipated frequency of their occurrences.

The possible accident scenarios, called accident sequences, represent all possible courses of subsequent events originating from each of the initiating events. The event

tree method is typically employed to identify these accident sequences; this method is based on development of so-called event trees for each initiating event. The event tree is a special flow diagram which graphically describes link-ups and dependencies among individual events in the accident sequence. The building of the event trees is usually carried out in two steps: first, the simpler function event trees are prepared, and only on the basis of them the detailed system event trees are developed.

Hence, the second step in the identification of possible accident scenarios is construction of the function event trees for all the initiating events identified. The events considered in the function event tree represent success or failure to carry out individual safety functions, i.e. the functions which have to be performed by NPP systems in order to mitigate the accident and prevent the reactor core from severe damage. The next step is expansion of the function event trees into the system event trees. The events in the system event trees directly represent success or failure of specific reactions of concrete NPP systems to particular initiating events. The system event trees thus outline individual accident sequences, each consisting of an initiating event and a succession of events characterizing success/failure of required reactions of specific NPP systems to this initiating event.

A list of possible initiating events, when initially put together, can contain up to a hundred of these events. Along the process of identification of possible accident sequences it is, however, usually established that sequences of events following some initiating events are very similar or even identical. This makes it possible to group the initiating events into groups represented by typical initiating events. The final list of initiating events to be included in PSA model is thus typically much shorter, and there are rarely more than 20 events in it.

2.1.2. Assessment of Frequencies of the Accident Sequences

To assess the frequency of a certain accident sequence, it is necessary to estimate frequency of its initiating event and probabilities of the other events in the sequence, i.e. probabilities of failures of individual NPP systems to perform certain required actions. The initiating event frequencies are estimated from accumulated operational data using various statistical techniques, typically employing a Bayesian approach. When a given initiating event is only a representative of a whole group of original initiating event (cf. section 2.1.1), its estimated frequency has to summarize the frequencies of all the initiating events in the group. In addition to it, some events considered as independent initiating events may sometimes also develop as a result of

other initiating events - to avoid overestimation of their frequencies it is necessary to exercise great caution and make sure that the root causes of all initiating events either directly listed or included in groups are well understood.

The probabilities of failures of individual NPP systems to perform their required tasks are estimated using the fault tree method. The fault tree method is based on development of the so-called fault trees for each event represented in the event trees. The fault tree is a special flow diagram which graphically describes interrelations of various basic undesirable events on the so-called top event of the fault tree. The top event is identical to the event represented in the event trees. The fault-tree basic events represent relatively simple events whose probabilities can be directly estimated; these events include not only various failure modes of individual NPP components but also possible human errors and external influences. The fault tree method thus allows the calculation of the probability of the top event from the known probabilities of the individual basic events. The basic event probabilities are again estimated from accumulated operational experience using various statistical techniques.

Each of the two graphical methods mentioned above (i.e. the event tree method and the fault tree method) is, in fact, capable of modeling on its own the whole accident sequence

from the initiating event to basic events. In general, the event tree method is more suitable for analysis of more general events, such as failures of the NPP as a whole, and the fault tree method is better for analysis of possible causes of failures of smaller compact systems. In any case, it is up to the designer of each PSA to choose the level of analysis where the events to be probabilistically quantified are simple enough to be analyzed with the fault tree method. Nevertheless, there are two general approaches used in the PSAs: the linked fault tree method, and the linked event tree method*. The prevailing linked fault tree method draws the division along the front-line systems, i.e. the systems which directly perform the safety functions. Thus, the events in the system event trees normally specify success/failure of the front-line systems only, while possible failures of the support systems, which are necessary for operation of the front-line systems, are treated as branches of the fault trees analyzing possible failures of these front-line systems. In turn, the rarely used linked event tree method extends the division to a somewhat lower level, where some of the events in the system event trees specify also success/failure of the support systems. Both these methods are clearly methodically adequate and should yield the same results. Nevertheless, the linked fault tree method is generally less demanding during

* There are also other terms used for the same thing in literature: the small event tree/large fault tree approach, and large event tree/small fault tree approach.

the actual development of the PSA model but requires more developed PSA code for its evaluation. The linked event tree method, in turn, is more demanding on the model development (as the complicated inter-dependencies between the NPP systems has to be treated in the less transparent event trees), but this manual pre-evaluation of the system inter-dependencies allows the use of a simpler PSA code for the model evaluation.

2.1.3. Assessment of Consequences of the Accident Sequences

In principal, there are three different measures used for assessment of the consequences of individual accident sequences in NPPs; application of these different measures results in three different types of PSA called "levels". The basic PSA of level 1 employs a binary criterion distinguishing individual sequences as leading to or not leading to a core damage (the core damage is typically defined by a certain limiting temperature exceeded in a certain percentage of the fuel rods). Decision whether the core damage actually occurs for a particular accident sequence is usually made using an expert judgement based on the results of thermal-hydraulic calculations.

The more advanced level-2 PSA assesses the consequence of accident sequences according to the so-called source term which defines the volume and spectrum of radioactive materials

released from the NPP during the given sequence of events. To determine the source term, it is necessary to carry out much more detailed thermal-hydraulic analyses and an analysis of spreading and depositing of various materials throughout the NPP. Modeling these processes is very complicated and inevitable simplifications that have to be accepted typically lead to a rather high level of uncertainty in the results.

A still more advanced level-3 PSA carries the analyses even further and assesses the consequence of accident sequences directly by their effect on the health of the general public, using such measures as the radiation dose obtained by an individual from the general population or the number of fatalities due to the particular accident sequence. Once again, modeling the spreading of radioactive material through the environment is a very complex task and thus some additional uncertainties are introduced into the final assessment of the risk.

2.1.4. Final Assessment of the Risk

The risk arising from the possibility of occurrence of a certain accident scenario is characterized by a product of the frequency and consequence of this scenario. The total risk associated with the NPP operation is then equal to a sum of the risks associated with the individual accident scenarios.

The evaluation of the PSA models is conducted with the help of special computer codes. Evaluation of both event trees and fault trees is today typically carried out using a single complex PSA code. When the code is informed about the expected core damage for the individual accident scenarios, it directly produces the resulting risk parameter of the level-1 PSA: the core damage frequency. The extensions to the higher level PSAs are carried out separately using specialized codes computing the source term for the level-2 PSA, and the actual effects of NPP operation on health of the general public for the level-3 PSA.

2.2. Applications of the PSA

For a long time, the PSAs have been used practically only for evaluation of design safety of currently operating or commissioned NPPs. Thus, the results of the PSA studies have been utilized for identification of the most dangerous accident scenarios which should be more thoroughly addressed in operator training, systems and components which should be backfitted in order to achieve a significant improvement of NPP safety, and systems and components which should be well maintained in order to keep present level of the NPP safety [4-6]. From time to time, utilities operating NPPs have also

used the PSA to support their applications to the regulatory authorities for temporary or permanent reliefs from Technical Specifications requirements [7].

Recently, however, a potential of the PSA for some other utilizations have been recognized [8,9]. Of these new options, a rather widely exploited application has been the utilization of PSA models for a risk-based optimization of existing deterministic NPP Technical Specifications [10,11]. Besides that, the PSA models have also been used for a risk-based optimization of maintenance strategies [12].

Today there has been a clear trend by authorities, utilities and NPPs worldwide to make use of the large amount of information accumulated in the PSA studies and to transfer it from research and development organizations and departments to everyday use in the NPPs themselves. The qualitative information offered by PSA is readily exploited in a number of applications [13-16], and the current effort concentrates on utilization of the quantitative information provided by the PSA, as well. The ultimate objective then seems to be the development of a risk monitor for configuration control of NPPs in a nearly-on-line regime [17].

3. RISK MONITOR

The risk monitor* is generally a rather simple expert system designed for support of operational decisions in NPPs. It provides guidance to NPP operators and maintenance personnel in their reactions to changes in NPP configuration.

3.1. Configuration Control

In general, operational safety of NPPs is maintained by monitoring and control of various NPP safety-important parameters, and by detection of existing failures of NPP components and organization of their timely repairs. A minimal requested NPP safety level is regulated by the plant-specific Technical Specifications which is an important document authorized by regulatory authorities of particular country. Individual requirements of the Technical Specifications currently in use around the world have been set deterministically with margins ensuring that the NPP safety-related systems will be capable of fulfilling their

* Because this application is so new, there is still several names in use for the same thing. These alternative names are, e.g. the "risk meter" and also the "living PSA". The term "living PSA" has, however, typically other meaning: updating of the base-line PSA model whenever any change of the design or operating procedures of the particular NPP is adopted. As the term risk monitor seems to be gradually prevailing, it is used in this study.

tasks even under very adverse circumstances. These margins have been typically determined using engineering judgement while assuring necessary conservatism.

Unfortunately, requirements of the Technical Specifications determined this way give little credit to the real influence of addressed features on the NPP safety. This undermines effectiveness of the Technical Specifications as their requirements are rather inconsistent, i.e. sometimes unnecessarily restrictive and sometimes too benevolent. To improve the situation, much effort has been recently put into reviewing these requirements using some reliability considerations allowing to take into account the risk arising from a specific situation (cf. [18]). Yet, any really consistent improvement of the Technical Specifications needs to be clearly based on the plant-specific PSA. The basic level-1 PSA is normally quite sufficient to provide a foundation for the risk-based improvement of the Technical Specifications. Nevertheless, the PSA level to be actually used depends on the criterion employed when judging acceptability of the additional risk associated with unavailability of individual components of the NPP. Some preference should be generally given to the level-1 PSA as it does not involve large uncertainties comparing to the higher level PSAs. But the best choice seems to be the PSA of the so-called "level 1+", i.e. a normal level-1 PSA supplemented

with an analysis of possible failures of NPP containment and its support systems.

The risk-based assessment and improvement of the Technical Specifications are aimed at that part of this document which deals with the so-called configuration control of the NPP. The configuration control addresses the problem of great number of different possible combinations of simultaneous outages of various components in different trains of the safety-related systems that may occur in the NPP. The configuration of the NPP is determined by a set of NPP components/systems momentarily unavailable for accident mitigation. Because the NPP instantaneous risk may be dramatically increased for some configurations, it is desirable to know which configurations are associated with this highly increased risk and forbid operation of the NPP reactor at power when they happen to develop. The term configuration control thus denominates a procedure for evaluation of the risk associated with individual configurations [19].

Yet, this effort to improve quality of the current Technical Specifications brings some difficulties too. The Technical Specifications is a written document which, as it becomes more elaborated and complete, becomes also more complex and difficult to follow. This problem is especially

relevant for the configurations control as the great number of possible configurations and complicated interrelations among individual NPP systems makes it virtually impossible to codify the configuration control in a practically usable written guidebook. In fact, the apparent if-then structure of the regulations governing the configuration control strongly suggest that a computerized expert system should be used for this task. And this expert system is what is called a risk monitor.

Potential users of the risk monitors can be found both in utility industry and regulatory authorities. In NPPs, the risk monitor can be used for the nearly on-line configuration control and for support of operational decisions - i.e. decisions whether the NPP reactor can still safely continue its operation at nominal power or if it should be rather shut down when a given configuration develops. The NPP and utility managers can use the risk monitor for proper scheduling of maintenance activities in the NPP assuring that NPP safety is never compromised severely. Besides that, the NPP and utility managers can use the risk monitor for identification of possible actions that will allow an increase of NPP safety and an avoidance the necessity of shutting down the reactor, and also for support of their applications to the regulatory authorities for temporary or permanent reliefs from Technical Specifications requirements. The regulatory authorities can

then use the risk monitor for a quick evaluation of acceptability of the proposed reliefs.

3.2. State-of-the-Art of the Risk Monitor

Evaluation and quantification of PSA models (usually called summarily only "quantification") is not generally an easy task and typically takes hours or even days of computer time, even when a quite powerful computer is employed. While quantifying a normal PSA model, there is generally no special concern about the time necessary for this task. By contrast, the risk monitor, if it is to be really useful, has to be capable of a prompt response to a user request for evaluation of a specified configuration. Thus, it is clear that some additional measures have to be taken to develop a practical risk monitor.

Obvious measures to be considered when trying to solve this problem are simplification of the PSA models and/or speeding up of the computer codes used for their quantification. Unfortunately, until recently the existing algorithms used for PSA quantification were inherently too slow to allow solving the problem successfully (and, in fact, also the computer hardware available has not been fully up to the job). Thus, the only two completed risk monitors known from a still recent

history have suffered from this situation - while struggling to obtain response in a reasonable time they have had to accept so severe simplifications of their internal PSA models that their practical use has been very limited. These two pilot projects have been the US risk monitor PRISIM ([20] - developed by the JBF Associates, Knoxville, TN for the US NRC), and the British risk monitor ESSM ([21] - developed by the Nuclear Electric for the Heysham 2 NPP). The situation even initiated research aimed at development of risk monitors not based on the PSA. There was a project conducted in Europe, called FORMENTOR, which led to proposal of a risk monitor based on non-probabilistic qualitative models utilizing the goal trees and the success trees (these are, in substance, complements of the event trees and the fault trees, respectively). The resulting risk monitor is capable of providing limited guidance to plant operators, and its development is worth considering for the plants which do not have their PSA model already developed [22].

Recently, however, there was a significant progress in computational speed of algorithms employed for the quantification of PSA models, and even greater progress in computer hardware [23]. This progress has made successful development of risk monitors finally plausible and initiated new research all around the world [17] - the most advanced projects are currently in progress in the USA, Germany, and

the United Kingdom. The US project closest to the practical application is the risk monitor being developed by the Haliburton NUS for the San Onofre NPP (operated by the Southern California Edison) and planned for full use in 1994 [24]. Besides that, there has been a pilot study of the risk monitor (based on PSA model of the Surry NPP as published in [6]) conducted by the SAIC for the US Nuclear Regulatory Commission [25,26]. In Germany, there have been two different projects going on: a general risk monitor being developed in the GRS with its completion scheduled for 1994 [27,17], and the risk-monitor methodology called SAIS prepared by the TÜV Norddeutschland and being applied to different NPPs [28,29]. The British project has been carried out by the Nuclear Electric for the Heysham 2 NPP [30]; the project represents a further development of the ESSM risk monitor mentioned above. There is also a Japanese project called LIPSAS carried out by the Power Reactor and Nuclear Fuel Development Corporation; it seems to be a typical risk monitor, but there is little information available about it yet [31]. The developers of these advanced projects usually claim that their risk monitors can provide a response in a matter of minutes. In any case, these projects still seem to be marked by an apparent struggle with the response time of the risk monitors, and the fast evaluation of their embedded PSA models is typically made possible at the expenses of simplification of these models.

Another measure making the speeding up of the risk monitors possible is removal of the most time-consuming operation done by risk monitors - i.e. the full evaluation and quantification of PSA event/fault trees by a PSA code - from the actual computation done after the user's request. One way of accomplishing this is through reduction of the PSA models so that their basic events directly represent operational states of NPP safety-related systems. This reduction would allow to store the models directly in a form of their resulting accident sequences and to quantify them very quickly by a simple substitution of availabilities of individual systems into these pre-computed solutions [32]. It is, however, clear that such a huge reduction of the PSA models would cause great difficulties with adjustments of the models input values representing availabilities of whole NPP systems. A rather complicated evaluation would be clearly necessary for determination of their values which would likely result in an overall slowing down of the risk monitor again.

3.3. Structure of the Risk Monitor

The expert systems usually employ predominantly symbolic computing and deal with ill-structured problems involving ambiguous, contradictory, or imprecise data [33,34]. In this sense, the risk monitor is not a typical expert system as its

problem is structured fairly well. The risk monitor is, in fact, a coupled system relying to a great extent on numerical computing and utilizing only a rather simple deductive reasoning for interpretation of the numerical results [35]. Nevertheless, like any expert system, the risk monitor can be divided into several co-operating parts which perform different tasks. These parts are namely knowledge base, input interface, inference engine, and output interface - see Figure 2.

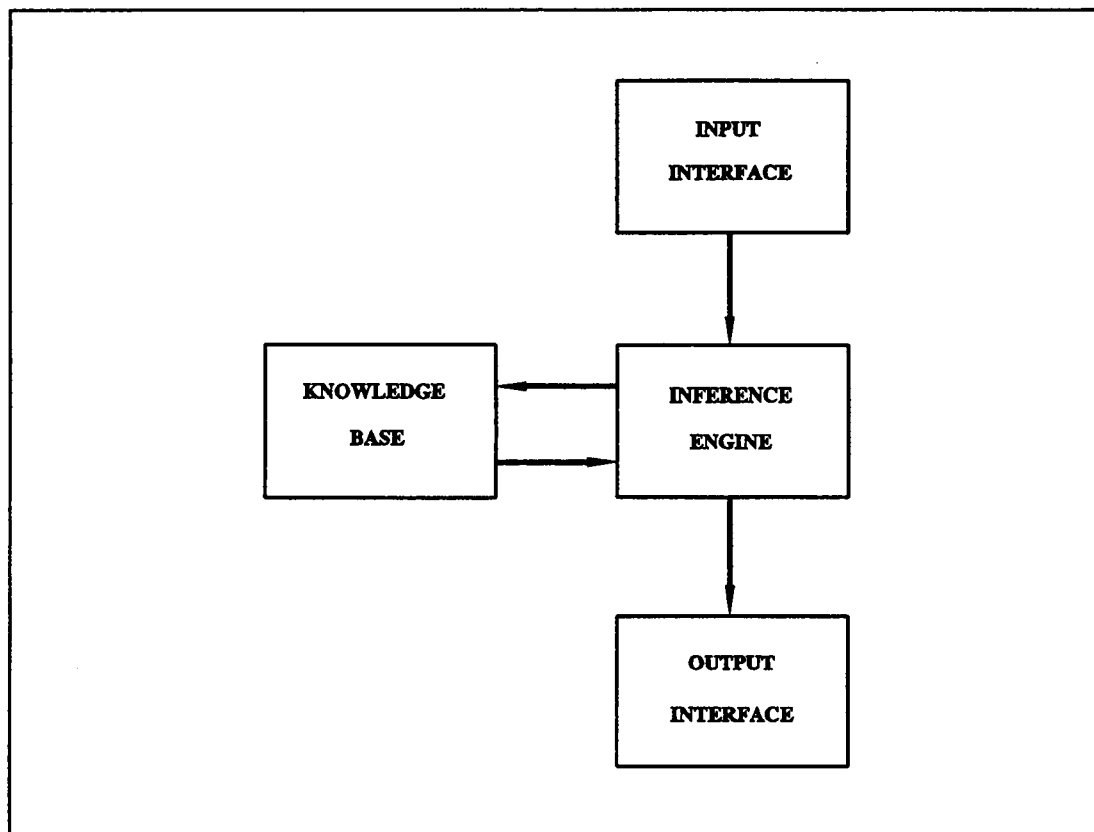


Figure 2 - Structure of the risk monitor

The knowledge base stores information essential for operation of the risk monitor, namely the PSA models and regulatory requirements concerning NPP operations. The input interface helps the user to communicate to the risk monitor in a user-friendly way which NPP components/systems are unavailable under the current configuration. The inference engine carries out an adjustment of the stored PSA models into the current NPP configuration, quantification of the adjusted PSA models and computation of NPP current risk characteristics, and decision about an optimal reaction to the present situation. The output interface converts results produced by the inference engine into meaningful recommendations and presents them to the user in a user-friendly way offering maximum possible support for his operational decisions.

3.3.1. Knowledge Base of the Risk Monitor

The knowledge base is a true core of the risk monitor which stores information essential for its operation. First of all, it has to accommodate whole plant-specific PSA models, i.e. namely the event tree/fault tree logic and probabilities of random occurrence of the models basic events. The PSA models should be, if possible, provided not only for the nominal-power-level operation regime, but also for hot-shutdown regime, and for one or more low- or transient-

power-level operation regimes of the particular NPP. In addition, lists of accident causes and/or precursors for all initiating events and all operation regimes should be stored in the knowledge base. Besides the PSA models, the knowledge base has to contain all relevant requirements of the Technical Specifications and/or any other regulatory requirements concerning NPP operations (e.g., safety goals).

3.3.1.1. Specificities of Risk-Monitor PSA Models

The PSA models used in the risk monitor are generally quite similar to ordinary base-line PSA models developed for the particular NPP. Nevertheless, there are some modifications of the original base-line PSA models which have to be carried out before the models are ready to be used in the risk monitor. These modifications have to be done in the PSA models for all NPP operation regimes, i.e. in the basic PSA model for the nominal-power-level operation regime, and in the PSA models for the hot-shutdown regime and low- or transient-power-level operation regime/regimes.

The purpose of the principal modifications of the base-line PSA models, in general, is to change the nature of risk characteristics calculated when evaluating the model. The original base-line PSA models provide typically average risk characteristics of the NPP over certain time period, usually

a year. Yet, the risk monitor reasoning has to be based on the current level of risk of the NPP operated with certain given configuration of its safety-related systems. Thus, the PSA models needed in the risk monitor should generally provide instantaneous risk characteristics characterizing current pointwise risk of the NPP. In practice then, the main difference of the base-line and risk-monitor PSA models consists in the way the maintenance of NPP components/systems is treated.

The basic events of fault trees and event trees can represent several different types of events and can be related to different effects. These events characterize not only various failures of NPP components/systems, but also the influences of human staff actions (e.g., errors of operators or maintenance personnel), external events, and maintenance actions on NPP operation. Specifically, the events characterizing maintenance actions in the base-line PSA models represent average unavailabilities of particular components/systems due to the maintenance over the whole time period evaluated by the models. Yet, in the risk-monitor PSA models, these events must never appear since these models are supposed to reflect the current NPP configuration. Instead, the fact that there is a maintenance action in progress in the NPP should be indicated by corresponding changes of the models. In addition, there are, actually, two different types of the

maintenance: preventive and corrective, which have to be modeled in different ways. The preventive maintenance is a test of the availability of the given component/system, and it is usually possible to return the tested component/system into operation relatively easily when there is an actual demand for it. By contrast, the corrective maintenance is a repair of the component/system failed during its operation or found failed during its test, and there is clearly no way of using this component/system when it is needed. The way of modeling of the preventive maintenance action depends strongly on the component/system involved. Some components/systems may be virtually unaffected by the preventive maintenance which therefore does not need to be modeled at all, other ones are more influenced but their maintenance can be still modeled by a simple addition of another event representing failure of their recovery, and still other ones require so complicated recovery actions that small fault trees have to be added to model their maintenance properly. The corrective maintenance, in turn, is typically modeled rather simply by setting probabilities of the relevant basic events of the models to unity.

The other types of basic events, which do not directly characterize the influence of the maintenance actions, can stay, basically, the same in both base-line and risk-monitor models. Nevertheless, the basic events characterizing

failures of individual NPP components/systems deal with several different types of events. The components/systems of the NPP are operated in different operational regimes, i.e. in any particular moment they can be either currently operating or on standby. While the operating components/systems are under operation stress only, the standby components/systems are subject of two different kinds of stress: the standby stress caused by temporary inactivity of the component/system, and the demand stress caused by an actual demand for the component/system to change its operational regime from standby to operating. Failure probability of the currently operating component/system due to the operation stress is typically quantified by an average probability of its failure over the time period evaluated by the models (or over the time of its actual operation cycle when it is shorter). Similarly, the failure probability of the standby component/system due to the standby stress is usually quantified by an average probability of failure over the time period between its two consecutive tests. Only the failure probability of the standby component/system due to the demand stress is normally expressed by its actual value, called availability on demand. Thus, even when the current NPP configuration is accurately modelled through corresponding changes in the PSA models structure, the resulting risk characteristics of the NPP obtained from the models still represent, to a great extent, only an average risk instead of the current instantaneous risk

of the NPP. Of course, it would be possible to replace the averaged probabilities mentioned above by the true time-dependent values closely following the actual situation, but it would represent enormous burden which is not, fortunately, really necessary. What we actually expect to gain from the risk monitor is just a possibility of comparisons of individual NPP configurations to the base-line configuration and among themselves. This can be successfully accomplished without using the truly time-dependent quantification of the model basic events*. Thus, we can keep the quantifications of failures of NPP components/systems, human influences, and external influences, as they normally appear in the base-line PSA models, unchanged in the risk-monitor PSA models. Nevertheless, the risk characteristics obtained from the resulting PSA models are thus only pseudo-instantaneous ones and need to be always recognized as such.

3.3.1.2. Necessary Modifications of the Base-Line PSA Models

Based on the reasoning presented in the previous section, the following modifications have to be performed in order to convert the original base-line PSA models to the PSA models

* After all, uncertainties inevitably present in the values of these probabilities, which are obtained from not always representative data available, would prevent us from obtaining truly reliable time-dependent instantaneous risk characteristics anyway.

suitable for utilization in the knowledge base of a risk monitor:

1. Removal of those basic events of the original base-line models which represent average unavailabilities of components/systems due to the preventive or corrective maintenance.
2. Provision of a possibility to accurately model the influence of the preventive maintenance (test) being in progress on those components/systems which are allowed by the Technical Specifications to be tested when NPP reactor is operated at power. This influence can normally be modeled by a simple addition of another basic event with OR-logic relation to the events representing other failure modes of these components/systems. The added basic event characterizes the probability of failure to recover these components/systems on demand from the maintenance. Nevertheless, depending on the complexity of the recovery actions needed, it may sometimes be necessary to add a little fault tree with several basic events instead of only the single event. The probability of this added basic event (and especially the whole fault-tree model of the recovery actions if used) is usually not readily available and so some additional analyses are needed [25].

3. Provision of a possibility to indicate that a certain component/system in the models is not available for accident mitigation due to its failure or due to the corrective maintenance (repair) being in progress on it. The objective of this modification is to allow easy setting of the overall probability of failure (availability) of the given NPP component/system to unity.
4. Addition of models of those NPP parts or recovery actions which have been neglected in the original base-line models because their normal unavailability have been found negligible in comparison with other events. These models may become important when the particular parts or actions are actually unavailable due to the maintenance or failure.
5. Addition of models of possible configurations of the NPP systems which have been omitted from the original base-line PSA models because they are prohibited by the existing Technical Specifications. This modification is very important because the support provided by the risk monitor should be especially helpful for these configurations, presumably marked with a high level of risk.
6. Provision of a possibility to fully model interrelations between individual units of the NPP. It is quite common that some NPP support systems (e.g. service water system)

are shared by several units of a multi-unit NPP, but the base-line PSA models normally address only the failures in a single unit. Yet, the risk-monitor PSA models need to model the real situation of the unit of concern and so they have to be able to also consider current operational status of the other units in all cases in which they may actually influence the risk of this unit.

7. In addition to the necessary modifications listed above, it is useful to consider still another one. This modification is namely tightening up of the success criterion of an NPP accident mitigation by considering, as intolerable, those accident sequences which require rather heroic recovery actions, or which cause serious damage or contamination of the NPP equipment (such as, e.g., "feed & bleed" cooling of the reactor core in the pressurized water reactors) even if they do not, actually, lead to the core damage. This modification takes into account the interests of the utilities (i.e. the risk-monitor users) which are naturally concerned not only with NPP operational safety but also with the repair costs of possible accidents.

3.3.1.3. Other Information Stored in the Knowledge Base

In addition to the logic and basic-event probabilities of the PSA models, a list of accident causes and/or precursors

should also be stored in the risk monitor knowledge base for all initiating events and all operation regimes. Moreover, this list should be made more informative by addition of percentile contributions of individual accident causes/precursors to the total frequency of individual initiating events for different operation regimes. This information can be used as a basis for identification of the most dangerous accident initiators for any particular configuration of the NPP.

Besides the PSA models, the knowledge base of risk monitor has to contain information about all regulatory requirements relevant to the particular NPP. This includes, first of all, all requirements of the plant specific Technical Specifications which regulate operation of the NPP when some of its components/systems are unavailable due to the maintenance or failure - i.e. namely the allowed outage times for individual NPP components and system trains, and list of NPP configurations for which further operation of the reactor on full power is explicitly prohibited. This information is obviously quite essential, as the risk monitor has to be informed if the current NPP configuration violates the Technical Specifications or not, and how long, if at all, the NPP can be operated in the nominal-power-level operation regime while this configuration exists.

In addition to the Technical Specifications requirements, there may also be other regulatory requirements which concern the particular NPP; they all need to be kept in the risk monitor knowledge base too. One of them may be, for example, a criterion for assessing the acceptable additional risk arising from the current NPP configuration, i.e. criterion determining if the current configuration creates a situation so dangerous that an immediate action has to be taken to reduce the risk (nevertheless, a criterion like this is still only under consideration around the world - for discussion see, e.g., [36]).

3.3.2. Input Interface of the Risk Monitor

The task of the risk monitor input interface is to provide for a user friendly indication of the NPP components/systems which are currently unavailable. As development of the input interface has never been really affected by the struggle with the computational speed, it is now probably the only part of the risk monitor which seems to be already fairly well developed. Both pilot risk monitors mentioned - PRISIM [20], and ESSM [21] - have used the same approach based on direct indication of unavailable components/systems in chart diagrams of the NPP. This approach seems to be an optimal way to manually update the risk monitor. Nevertheless, every effort should be put into a development of a more advanced risk

monitor which would be an integral part of NPP diagnostic system and so updated automatically by NPP monitoring system.

3.3.2.1. Manual Update of the Risk Monitor

The look of control screens with the chart diagrams for the manual update of the risk monitor needs to be developed with a close cooperation with the NPP personnel which will be assigned to update the risk monitor, i.e operators and/or maintenance personnel. In order to avoid unnecessary burden and likely mistakes, the chart diagrams used by the risk monitor input interface should be as close as possible to the diagrams used by the NPP computer system employed for monitoring of NPP status in the control room, and/or the diagrams should be also the same or at least similar to those used during training of operators (and/or maintenance personnel). This also means that even the input interface has to be strictly plant-specific.

The update of the risk monitor (i.e. indication of the NPP parts currently unavailable) should be, generally, based on a hierarchy of menu-driven control screens operated by mouse buttons. The initial screen of the risk monitor should offer a menu with a list of the safety-related systems of particular NPP - a chart probably would not be too helpful on this level. The screens with the chart diagrams of individual systems

should provide for indication of the unavailable component/system simply by pointing the mouse cursor to it. Unavailability of the component/system should be then also indicated directly in the chart diagram by a corresponding change of its color.

When there is the maintenance in progress on some NPP systems, whole parts of these systems can be affected due to a possible necessity to introduce some special measures (as, e.g., disconnection of a control signal or electric power supply, or change in position of certain valves). Though even the most complex maintenance action which may be conducted on the system does not mean, in fact, anything more than that certain parts of the system are unavailable or only conditionally available, it would be obviously much easier to handle them as one package. These actions typically concern some parts of individual trains or the trains as a whole of the affected system, and practically never cross borders among the trains. As the procedures for these actions are typically strictly regulated in the NPP Technical Specifications (or at least maintenance manuals) and their number is quite limited, it is possible to define states of the affected systems simply by indicating the type of maintenance action being in progress on them. Thus, the control screen of the particular system should also offer a menu with a list of these planned maintenance actions (together with the list of system trains

for indication which train is affected) and allow for entering the action as a whole. Naturally, entering such an action should also result in the corresponding color indication on the screen. This modification should indicate not only the action as a whole but also the status of all affected components, preferably with other color than the one used when these components are unavailable by themselves only (due to an unplanned maintenance or failure). This option may be of great importance because the planned maintenance actions go sometimes together with rather complicated changes in operation of the affected system (e.g., the other train/trains of normally fully standby system may be raised to a higher level of availability by starting some of its motors) and direct indication of the action as a whole may substantially simplify updating of the risk monitor.

Besides the basic information about the occurrence of particular maintenance action or other event causing the unavailability of some NPP components/systems, also information about the actual time point when the particular event started should be supplied to the risk monitor (when the event is not entered automatically). This information can be crucial for keeping record of the time windows remaining for maintenance of the individual affected components/systems before expiration of their allowed outage times specified in the Technical Specifications.

3.3.2.2. Automatic Update of the Risk Monitor

Modern NPPs are typically equipped with a sophisticated monitoring system connected to the NPP computer. Thus even the current monitoring system typically possesses a large amount of information about the status of the components/systems. Therefore, it should not, basically, be too difficult to extend the NPP monitoring system in such a way that it would be able to update the NPP risk monitor automatically.

Besides the addition of monitoring of the components which are not monitored in the existing system, another main extension of the current NPP monitoring system required for this application would consist in providing for monitoring the maintenance actions being in progress in the NPP. When the system is being brought to a maintenance regime, it is typically done by a very specific action consisting in a specific change of status of a certain component or a set of components. Hence, appearance of this specific pattern could specifically indicate the action being in progress and can thus be a source of necessary information for the risk monitor. Contrary to the manually updated risk monitor, there would be obviously no problem with keeping the automatically updated risk monitor informed about the precise time when the particular event started. Unfortunately, this advanced

implementation of the risk monitor to the NPP monitoring system is not likely to be a matter of the near future because the risk monitors will have to first prove their practical usefulness and reliability.

3.3.3. Inference Engine of the Risk Monitor

In any expert system, the inference engine is its only dynamic part. Using the information stored in the knowledge base and entered via the input interface, it performs all reasoning necessary for production of expert system decisions. Inference engines of typical expert systems use mainly symbolic computing processes which include identification of the rules which apply to a given situation and execution of these rules. The risk monitor, however, is not a typical expert system and so it uses also a rather atypical inference engine.

The risk monitor can be generally characterized as a coupled system utilizing shallow coupling of symbolic and numerical computing processes. It relies to a great extent on the numerical computing and uses the symbolic computing only for simple control of the numerical computing and interpretation of the its results (the coupling is considered shallow because the symbolic computing is done with only little knowledge about the processes involved in the parts

utilizing numerical computing [35])). The numerical computing is used in the risk monitor for quantification of the PSA models and computation of importance measures of individual components or systems. Use of the symbolic computing in the risk monitor is limited to a simple forward chaining process performing deductive reasoning used for interpretation of the results of the PSA models quantification with respect to the existing regulatory requirements, and identification of the components and systems for which importance measures should be computed.

The tasks of the risk monitor inference engine include namely adjustment of the PSA models to a given NPP configuration as specified via the input interface, evaluation of the adjusted PSA models and calculation of various risk characteristics for this configuration, and derivation of corresponding recommendations concerning further operation of the NPP.

3.3.3.1. Adjustment of the PSA Models

Any change in configuration of the NPP, i.e. any change in availability of its components/systems, requires adequate adjustment of its PSA model (or models if the models for different operational regimes are available). This adjustment almost always requires the introduction of some changes of

probabilities of certain basic events of the models, but may also require the introduction of some changes in the models structure (the way the adjustment is done is naturally very dependent on the change itself). The inference engine is informed about the current NPP configuration through the risk monitor input interface. The information transmitted to the inference engine, in general, comprises a list of the components/systems whose operational states have been directly affected by certain event (this list does not include the components/systems operational states of which have changed only secondarily as a result of the primary changes of the directly affected components/systems), and indication of the type of these events (i.e. if they involve the preventive maintenance, corrective maintenance, or failure). Task of the inference engine is an interpretation of this information and introduction of corresponding changes to the PSA models; a very simple symbolic computing procedure is used to achieve this.

If the given event represents preventive maintenance, this change consists in a certain reduction of the affected component/system availability. In this case, the model adjustment is accomplished by actuation of those parts of the models which represents the actions necessary for recovery on demand of the affected component/system from the maintenance regime (cf. section 3.3.1.2). In practice it means - in

dependence on complexity of these recovery actions - either simple change of probability value of the single basic event representing this recovery from zero to certain given value, or connection of the fault tree branch modeling the more complicated recovery actions.

If the event represents the corrective maintenance or failure, the change to be introduced consists in setting the overall probability of failure (availability) of the affected component/system in all models to unity. This could be seemingly easily accomplished by considering the basic event representing one of the failure modes causing failure of this component/system as guaranteed (i.e. setting the value of its probability to one), and considering all the basic events representing its other failure modes (including common cause failures), if there are any, as impossible (i.e. setting their probability values to zero, or disconnecting them from the models). Nevertheless, when doing this adjustment in practice, it is necessary to exercise great caution. First of all, it is unfortunately quite usual that the basic events representing individual failure modes of the same component appear in the parts of the event tree/fault tree models which are completely apart of one another. But in spite of that, it is crucial that the overall probability of all failure modes affecting the particular component/system is set to unity. Therefore, all these events, but the one considered

guaranteed, have to be considered impossible and their probability set to zero. Besides that, there is one more problem to deal with: in order to speed up the computation, all the PSA codes generally use a rather simplified algorithm for evaluation of the fault trees which overestimates resulting probabilities of the fault trees top events; this overestimation increases with an increase of the probabilities of the fault trees basic events*. Unfortunately, the risk monitor requires high speed of computation and deals with high probabilities of the basic events (indeed, even the probabilities equal to one are not rare). Hence, the specific features of the risk monitors create a highly unfavorable conditions for application of this simplified algorithm. In order to compensate the imperfection of the algorithm, all the events related to the guaranteed event solely by an OR-logic relation (i.e. all the events representing inputs to the same logical-OR gate as the guaranteed event, and also all the events representing inputs to all following logical-OR gates up to the first logical-AND gate - "following" being meant here in the sense of probability signal propagation through

* The probability of the top event is computed via minimal cut sets of the particular fault tree. Specifically, it is calculated using the so-called "inclusion/exclusion formula" which consists of a series of terms with decreasing magnitudes and alternating signs [37]. With addition of individual terms to the formula, the resulting top event probability is iteratively getting closer to the correct value. Unfortunately, computation of each additional term always requires more computational time than the computation of the previous one. Therefore, the PSA codes rarely compute more than two terms and typically only the first one is computed. This approximation of the whole formula by its first term only is called the "rare-events approximation", because it is sufficiently correct only for the basic events with probabilities not higher than 0.1.

the fault tree) have to be considered impossible. Specifically it means, that the whole branch of the fault tree comprising these events has to be disconnected, or that probabilities of all these events have to be set to zero.

3.3.3.2. Evaluation of the Adjusted PSA Models

Next task of the inference engine is evaluation of the adjusted PSA models. The evaluation of each model consists in loading the current basic-events probabilities to the model and quantification of the model, and yield current risk characteristics of the NPP - see Figure 3. The NPP risk characteristics obtained are the core-damage frequencies for individual initiating events in the model. Only the numerical computing is employed to accomplish this task while using special algorithms.

Because quantification of the PSA models is generally quite computer-time intensive, it actually determines the response time of the whole risk monitor. Therefore, only the fastest algorithms are suitable for utilization in the risk monitors; currently, the RELTREE algorithm of the Relcon AB, Sweden is probably the best suited for the job [23]. Nevertheless, even when a really fast algorithm is used, the time required for completion of this task still creates a problem (more about it can be found in section 3.5 of this study).

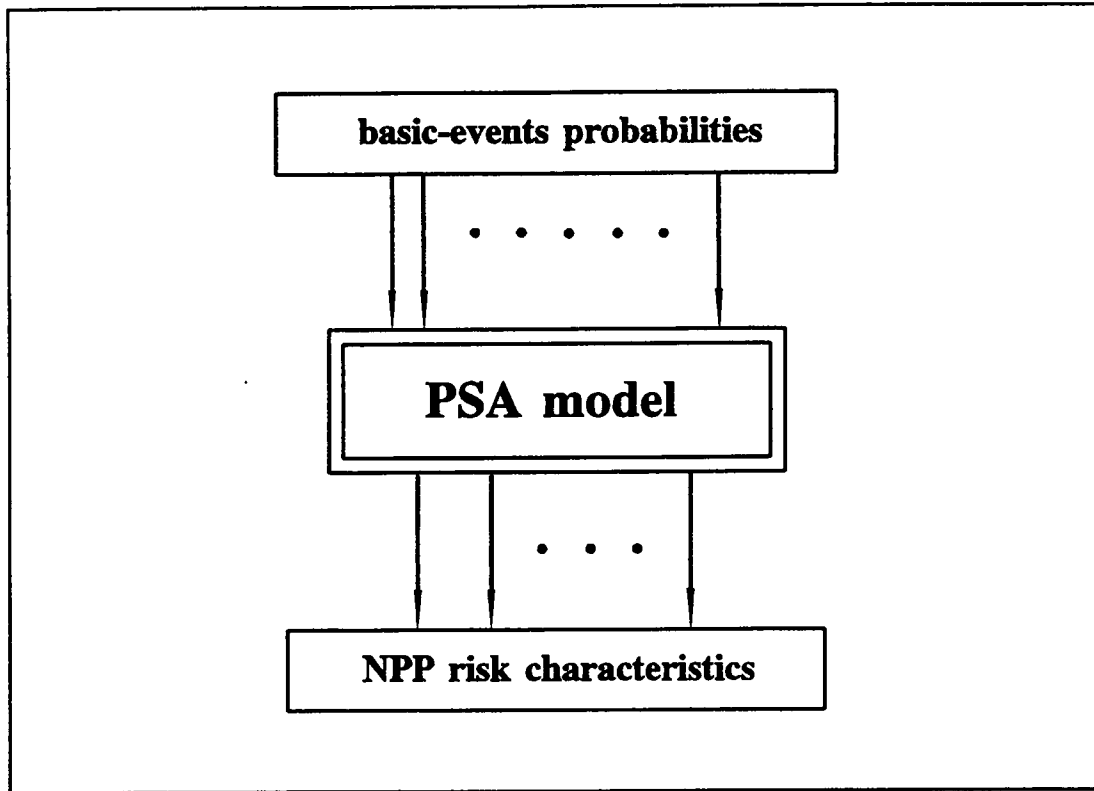


Figure 3 - Evaluation of the PSA model

3.3.3.3. Derivation of Risk Monitor Recommendations

The final task of the risk monitor inference engine is derivation of recommendations concerning further operation of the NPP. This comprises interpretation of the results obtained when quantifying the PSA models adjusted to the current NPP configuration, and control of computation of importance measures. To accomplish this task, the symbolic computing is employed while using a simple forward chaining based on deductive reasoning.

The first step in the effort to derive these recommendations is a comparison of the list of components/systems unavailable under the current configuration with requirements of the Technical Specifications stored in the knowledge base. Besides that, in case that there has been issued a criterion for assessment of acceptability of additional risk associated with the NPP configuration, the resulting core-damage frequency of the NPP obtained from quantification of the PSA model for nominal-power-level operation regime has to be compared with this criterion. Naturally, if there is any other regulatory requirement applicable, it needs to be checked against the current configuration as well. The results of these comparisons determine further process of derivation of the recommendations.

When the current configuration is found to be neither forbidden by the Technical Specifications, nor causing the increase of plant risk exceeding the level prescribed by the criterion for assessment of the acceptable additional risk, nor violating any other existing regulatory requirements, the risk monitor recommendations need to concentrate on identification of possible additional events which could cause violation of the regulatory requirements and/or severe accident, and on prioritization of the maintenance actions aimed on bringing the currently unavailable components/systems

back into the operational status. First of all, using the results of quantification of the PSA model for the nominal-power-level operation regime, the inference engine has to identify those accident causes/precursors which are the most dangerous under the current configuration. Then, the inference engine has to initiate computation of selected importance measures of the PSA model for the nominal-power-level operation regime, i.e. namely the risk control importances of individual components and systems (i.e. their importances for maintaining the current level of the risk), and the risk reduction importances of the currently unavailable components/systems (i.e. their importances for possible reduction of the current risk). Finally, providing that it is informed about the exact time when individual unavailable components/systems have gone out of their operational status, the risk monitor has to compute the time remaining to the expiration of their allowed outage times. Using this information, it then has to follow the time flow and, in case that the allowed outage time of any of the unavailable items expires, automatically re-start the process of checking the current configuration against the regulatory requirements, while considering this additional development.

When, in turn, the current configuration happens to violate any of the obligatory regulatory requirements for the NPP operation in the nominal-power-level operation regime, the

risk monitor recommendations need to concentrate on comparison of the risk associated with the current configuration for the individual operation regimes of the NPP, identification of recovery actions which could in the shortest time and most effectively reduce the current risk level, and identification of possible additional events which would cause the greatest increase of the risk level and/or even a severe accident. First of all, the inference engine has to initiate quantification of the PSA models for hot-shutdown and transient-power-level operation regimes. Next, it has to initiate computation of selected importance measures of the PSA models, i.e. the risk reduction importances of the currently unavailable components/systems for the nominal-power-level operation regime, and the risk control importances of individual components and systems for all operation regimes of the NPP. Finally, using the results of quantification of the PSA models for different operation regimes, the inference engine has to identify those accident causes/precursors which are the most dangerous under the current configuration and for individual NPP operation regimes.

3.3.4. Output Interface of the Risk Monitor

The task of the risk monitor output interface is to communicate the recommendations derived by the inference engine to the risk monitor user in a way providing a user-

friendly support for its operational decisions. In general, the recommendations have to be ranked according to their importance for the decision. This allows the risk monitor to present the user initially only with the most important summarized results while offering him a possibility to obtain more detailed information through user-friendly menus.

Naturally, the initial output screen of the risk monitor has to inform the user if the current configuration violates any regulatory requirements, and what the current overall level of the NPP risk is. This initial information should be presented in both a textual and a graphical way, using a color code preferably similar to the one typically utilized in the accident management procedures (i.e. red color for a high risk and yellow color for an increased risk). Obviously, the answer to this basic question determines the structure and contents of the following output screens. In any case, this basic information should be displayed (in a conveniently condensed form, of course) at all times and on all the screens presented by the risk monitor.

In case that the current NPP configuration appears to be within the limits prescribed by all valid regulatory requirements, the next base screen, accessible simply by clicking any key without necessity of actual selection from any menu, should provide the allowed outage times for the NPP

components/systems unavailable in the present configuration. If the risk monitor is informed about the time when these unavailable components/systems have gone out of their operational status, it should instead provide the actual time windows available for bringing these items back to the operational status before their allowed outage times expire. This information is obviously very important for the user, as it can help in prioritization of the maintenance and recovery actions. Besides that, the base screen should also offer a menu of available additional information; this menu would allow an easy access of desired information from the list of available items. The different items of detailed information should be displayed on separate screens, presenting namely an ordered list of the risk reduction importances of the currently unavailable components/systems, an ordered list of the currently most dangerous accident causes/precursors, and ordered lists of the current risk control importances of individual NPP components and individual NPP systems.

If the current NPP configuration violates any regulatory requirement, the next base screen (again accessible simply by clicking any key without necessity of actual selection from any menu) should provide a description of actions required by the regulatory authorities in the current situation including the time limit prescribed for their completion. Generally, when a certain configuration violates the Technical

Specifications or other obligatory regulatory requirements, a typical action prescribed by the regulatory authorities is a shutdown of the reactor. Nevertheless, neither the Technical Specifications nor any other regulatory requirements normally ask for an immediate reactor scram but usually specify a certain time period available for realization of this operation (length of this time period depends on severity of the situation). Thus, when the risk monitor response is sufficiently fast, a portion of the available time window can be used for exploring the available options. Indeed, the risk monitor recommendations can even be used to find the way to bring the NPP back to an acceptable configuration during this time period, thus allowing the avoidance of the reactor shutdown altogether.

In order to be able to exploit this possibility in full, the base screen should again offer a menu of available additional information allowing an easy access to a set of well-arranged separate screens providing all necessary information in detail. The first item to be offered this way should obviously be an exact quotation of the relevant parts of the regulatory documentation which would explain why and how the current configuration violates the regulatory requirements. Other important information offered should be a comparison of the current overall core-damage frequencies for individual operation regimes; this comparison might reveal

that the prescribed hot-shutdown operational regime or the transient low-power-level regimes are even more risky than the continuous operation in the nominal-power-level regime for some configurations [38]. Besides that, other screens offered should present ordered lists of the currently most dangerous accident causes/precursors in different operation regimes, and ordered lists of the risk reduction importances of the currently unavailable components/systems in different operation regimes. All these screens would provide the user with the information crucial for a truly informed decision if the reactor has to be really shut down or if there is a possibility or even advisability to avoid it, and what recovery actions have the highest potential to quickly improve safety of the NPP in the current situation.

3.4. Verification and Validation of the Risk Monitor

It appears to be quite common that newly developed expert systems are put into practical use without a proper check of their correctness, usually called verification and validation [39]. There is, in fact, a lot of confusion in distinguishing the meanings of these two terms. Nevertheless, the verification is normally referred to as checking if the product works as expected by its developer (i.e. if the product is built right), and the validation is referred to as

checking if the product performs as required by its user (i.e. if the right product is built) [40]. Thus, the verification typically means checking of correctness of individual parts of the product, while the validation means checking of usefulness of the product in some practical situations with known or predictable solutions.

The risk monitor, specifically, can be characterized as a coupled system with prevalence of the numerical computing above the symbolic computing (cf. section 3.3). Therefore, the methods suitable for its verification and validation are to be found more likely in the references dealing with conventional computer software [41], than in those dealing with true expert systems [40,42]. In any case, the success in building a really dependable risk monitor requires that its development starts with preparation of a good quality assurance program allowing thorough parallel verification of the process.

The risk monitor is actually a rather simple product, but its utilization in the very sensitive area of nuclear safety makes its thorough verification and validation really important. There are several features of the risk monitor which need verification. First of these features are the modifications introduced to the PSA models (cf. section 3.3.1.2). These modifications should be, first of all,

verified by an independent review, preferably performed by a team consisting of developers of the original PSA models and personnel familiar with the NPP of concern. Besides that, there are also ways to check these modifications randomly using some unique features of PSA models. In every case, to be able to use these features, the modified PSA models to be used in the risk monitor have to be quantified with a recognized, verified PSA code. The models can be tested namely for the effect of adding more and more basic events of the models considered guaranteed - the resulting core-damage frequency of any initiating event should never decrease when another guaranteed event is added to the previous configuration, and it should also never exceed the frequency of the particular initiating event.

Other features of the risk monitor specifically requiring verification are the algorithm used for the PSA models quantification together with the input interface and corresponding part of the inference engine performing adjustment of the models. To accomplish this verification successfully, it is reasonable to develop an auxiliary code, so called "driver", which controls execution of the relevant parts of the risk monitor. The driver enables use of the input interface as planned for the completed risk monitor; it produces the resulting core-damage frequencies for individual initiating events and operational regimes and allows

confirmation of their match with the corresponding values obtained while using a recognized, verified PSA code. The same approach can be used when checking another feature of the risk monitor, i.e. computation of various risk importances for individual basic events and groups of basic events. Also, the same approach can easily be used for verification of the stored allowed outage times against the Technical Specifications, and verification of proper shortening of the available time windows with the time.

In contrast with many other expert systems, there is no shortage of suitable cases with known required results which can be used for validation of the risk monitor. These cases are provided in abundance by the Technical Specifications valid for the NPP of concern. Of course, the information provided by the risk monitor has to be in full agreement with the Technical Specifications for the configurations explicitly covered in this fundamental document.

It is more usual than not, that certain details of design and operational procedures of, and even regulatory requirements for, any real NPP typically change during its lifetime. Therefore, the verification and validation of the risk monitor is a never-ending process which has to keep up with all these changes and maintain the risk monitor up to date. All relevant changes have to be introduced to the risk

monitor PSA models and/or other items of information stored in the risk monitor knowledge base. These changes have to be again verified and validated in the same way as the first version of the risk monitor.

In addition to all that mentioned above, there is one more thing that could be, generally, considered part of the verification and validation process of any software product, and especially the products serving for support of human decisions. It is a documentation of the merits of the product for the users. Though the possible merits of the risk monitor seem to be obvious, it still could be useful to demonstrate this intuitive feeling in a formal way - empirical methods which can be used for this task are described in [43].

3.5. Possible Role of Artificial Neural Networks in Risk Monitor

As it was already mentioned above (see section 3.2), the struggle to obtain a response from the risk monitor in a reasonable time is what practically determines the risk monitor design. The most time consuming operation of those performed by the risk monitor is specifically the quantification of the PSA models. Even the fastest algorithms available are not appear fast enough to meet the needs of the risk monitor. Therefore, some special modifications of the

risk monitor PSA models (in addition to those described in section 3.3.1.2) are typically introduced by the risk monitor builders with the aim to facilitate the burden of the algorithm. These additional modifications typically include namely [24,25]:

1. Conversion of event trees to fault trees and creation of a single master-logic fault tree for each of the models. Unfortunately, this modification brings the loss of possibility to calculate risk parameters separately for each initiating event (without this information it is practically impossible to identify the most dangerous accident causes/precursors for individual configurations).
2. Reorganization of the models, i.e. reduction of the number of gates, and moving repeated events to higher levels of the master-logic fault trees. This modification, however, substantially changes the originally transparent structure of the fault trees and thus creates great problems for actualization and verification of the models.
3. Simplification of the models, i.e. creation of super-components and neglectation of less influential basic events. Unfortunately, the neglectation of some events is a rather dangerous operation as virtually any event may become quite influential for certain configurations of the NPP (the

results of the risk monitor can be thus distorted in an optimistic way).

All the mentioned flaws, which are inevitably or potentially introduced to the risk monitor PSA models, can easily result in an unreliable performance of the risk monitor and its rejection by the users, and especially by the regulatory authorities responsible for nuclear safety. Therefore, extreme caution has to be exercised when carrying out these additional modifications, and there is clearly no guarantee of any success under such conditions.

Yet, there seems to be another approach capable of speeding up the operation of the risk monitor without the questionable PSA models modifications just described. It consists in utilization of the artificial neural networks (ANNs) which are well known for their very fast computation. The idea of employment of the ANNs in risk monitors is based on an assumption that a pre-trained ANN would be able to emulate successfully computations normally done with the PSA code - verification of this assumption is a main goal of this study. When successful, this approach would make it possible to remove the most time consuming operation done by risk monitors (i.e. full evaluation and quantification of the PSA models by a classical PSA algorithm) from the actual computation done by the risk monitor only after the request for its advice from

the user. Trial quantifications of the individual PSA models for different NPP configurations would be performed in advance using any commercial classical PSA code available, and the results of these computations would be used for supervised training of the ANNs. During the training, the ANNs would be, presumably, taught the structure and interrelations of the PSA models, allowing them to produce the same results as the original PSA code in a much shorter time. The ANNs would thus only replace the numerical computing originally performed by the PSA-code algorithm*.

The expected advantages of the risk monitor utilizing the ANNs for the PSA models quantification as compared to the one utilizing the classical PSA-code algorithm include:

- * impressively short response time,
- * more information about the NPP status as separated core-damage frequencies for individual initiating events remain available,
- * complete avoidance of the necessity to carry out the modifications of the PSA models needed for speeding up their quantification described above,

* This also means that the resulting risk monitor would not pass for a true hybrid system which generally integrates the symbolic computing of the expert systems with the neural computing of the ANNs. The role of the ANNs in the hybrid system is to perform generalized classification tasks which cannot be done using the symbolic computing because it requires well specified inputs and a deep understanding of their interrelations [44].

- * no problem with inclusion of the PSA models for different regimes into the risk monitor (quantification of several different models in a reasonable time would be very difficult with a classical PSA code), and
- * no necessity for the risk monitor user to purchase a rather expensive algorithm for quantification of the PSA models.

The first attempt to use the ANNs in the PSA field was made in the Department of Nuclear Engineering at the University of Tennessee in Knoxville in 1990 [45] when the ANN was trained to emulate the results of the pilot risk monitor PRISIM mentioned above. The ANN successfully learned to calculate failure probabilities of individual systems when some of their components were considered unavailable and it also proved to be able to reproduce any information used in its training. Nevertheless, the ANN completely failed to generalize the internal structure of the PRISIM PSA model and was not able to produce correct results for the configurations not used in its training. This failure and also the limited capabilities of the PRISIM itself effectively discouraged any further work along these lines at that time.

This study represents another attempt to employ the ANNs in risk monitors while profiting from recent progress of the ANNs, risk monitors, and computer hardware. The next fourth

chapter provides a brief introduction to the field of the ANNs with a special emphasis on the ANN paradigms suitable for the function approximation, and the following fifth chapter describes the approach used and the results obtained when investigating the feasibility of utilizing ANNs in the risk monitor.

4. ARTIFICIAL NEURAL NETWORKS

The field of artificial neural networks (ANNs) originated from an effort to emulate capabilities of biological nervous systems and particularly the mammalian brain. Researchers have been trying to simulate functions of individual biological neurons and whole networks of these neurons using mathematical models. Nevertheless, capabilities of some of the developed ANNs have appeared to be quite interesting also from the viewpoint of possible applications to data processing. With the advance of computer technology, it has become relatively easy to experiment with different ANN structures on digital computers. As a result, research in the ANN field has, to a great extent, become independent from its biological origin, and many researchers are now simply trying to develop new ANN paradigms useful for practical applications without any ambitions to emulate biological neural networks.

4.1. General Description

In general, the ANN is a special computing system using extensively parallel processing of data in a number of

interconnected processing units". The processing units are typically organized in several layers which include an input layer**, where input data are presented to the ANN, and one or more processing layers; the last of them serving as an output layer where the ANN reports its response to the presented input. Different ANN paradigms differ in the way in which the signals presented to the input layer propagate through the ANN, i.e. in the number of their processing layers, in the operations performed by the processing units in these layers, and in the topology of connections of the individual processing units. In every case, however, the ANN performs a mapping of individual input signals presented to its input layer to the corresponding output signals formed at its output layer. The input and output signals are represented by the

* The relatively short history of the ANN field shows itself in a still not fully settled terminology. Besides the rather neutral term processing unit several other, usually more biologically inspired terms are used as well in literature - e.g. nodes, neurodes, neurons, artificial neurons, and processing elements. Also, instead of the term artificial neural network some other terms may be encountered as, e.g., neural net, artificial layered network, connectionist model, or connectionist network. The terms processing unit and artificial neural network are used on principle in this study. This terminology has also made possible the introduction of a certain short cut in descriptions of different ANN paradigms - when talking about the processing units in individual ANN layers, the general adjective "processing" in the term processing unit is replaced with the specific one identical to the adjective in the name of the particular layer (e.g. the term "input unit" is used for a processing unit in the input layer).

** In literature, the input layer is sometimes not counted as a full-value layer of the ANNs. This practice is based on the fact that the processing units in this layer do not process the incoming signals and simply pass them to the next layer. Nevertheless, the input layer has its task in a distribution of the signals to the processing units of the next processing layer which is hardly less important than the task of processing units of some processing layers in some ANNs which only sum up the incoming signals. Also, the input layer is typically shown in the diagrams of ANN topologies and its exclusion from counting thus creates unnecessary confusion. Therefore, in this study the input layer is counted as a full-value ANN layer.

input and output vectors; individual elements of these vectors characterize the signal values sent to and coming out of individual processing units of the input and output layer, respectively.

Application of the ANN always consists of two stages: ANN training, and actual data processing with the ANN. The ANN is typically a non-algorithmic system, i.e. no effort is normally necessary to find a functional relation between individual input and output vectors. Instead, the ANN learns by data samples presented to it during the training stage; it adjusts weights in connections between its individual processing units and/or some other trainable parameters according to these samples. Different ANN paradigms use quite different training algorithms. Nevertheless, there are essentially just two basic general types of training: the supervised training, where the training data samples comprise both the input vectors and the required output vectors corresponding to them, and the unsupervised training, where the training data samples contain only the input vectors*. In any case, after completion of the training, the ANN is ready to operate in the data-processing stage when it simply accepts the input vectors and produces the corresponding output vectors. In spite of relative newness of the ANN field, there is already quite

* In fact, there is one more type of training, the reinforced training, which is somewhat in between of these two basic types. It is, however, used rather rarely and practically only in control applications.

extensive literature available (a good general overview can be found, e.g., in [46-49]).

The ANNs can be used for many different tasks - possible applications include, e.g., pattern recognition, function approximation, signal filtering, data compression, adaptive control, and optimization. In comparison with other data processing methods, the ANNs offer some specific advantages which make them especially useful for certain tasks. These advantages include [49]:

- * adaptive learning and self-organization, i.e. the ability to learn how to do the task by samples without necessity for the user to develop and implement potentially complicated algorithms,
- * generalization and robustness, i.e. the ability to produce a fairly correct response when presented with the input vectors which are noisy, distorted, or have not been used during the training,
- * fault tolerance, i.e. the ability to produce a fairly correct response when values of some of the ANN parameters are distorted (owing to distribution of knowledge in the ANNs), and
- * high speed, i.e. the ability to produce the response very quickly in the data-processing stage (especially when implemented on a special microchip).

4.2. Paradigms Suitable for Utilization in Risk Monitor

The planned task of ANNs in the risk monitor consists in the quantification of PSA models. This application is a case of the function approximation*, where the approximated functions represent the individual PSA models. The independent (input) variables of these functions are probabilities of selected basic events of the models, and the dependent (response) variables of the functions are individual risk characteristics of the NPP. When an ANN is to be used for function approximation, its training has to be supervised. In addition, it has to be capable of a good generalization, i.e. it has to be able to interpolate correctly the function response variables between the output vectors actually shown to it during training (obviously, it is not practically realizable to train the ANN with all possible output vectors). There are several different ANN paradigms which can be used for the function approximation [49,50]: namely the backpropagation network, the functional link network, the modular neural network, the radial basis function network, and the generalized regression neural network.

* There are several different names used for this application in literature - mapping, function approximation, and regression. By far the most frequently used is the name "mapping". Nevertheless, this name can be quite misleading as it fits to a general description of ANN functioning (and in this meaning is also used throughout this study). The names "function approximation" and "regression" have effectively the same meaning; the name function approximation is used on principle in this study.

4.2.1. Backpropagation Network

The backpropagation network* (BPN) [46-49,51,52] is a typical ANN consisting of three or more layers" - the layer/layers between the input and output layers are called hidden layer/layers. It is normally a fully-connected and feedforward ANN, i.e. each its processing unit sends its output signal to all of the processing units in only the next layer. A topology of the three-layer BPN with three output units is shown in Figure 4. All connections between the BPN processing units are provided with weights allowing modification of the signals coming into the target units. The hidden and output units sum the weighted signals received from the processing units of the previous layer and filter the result through an activation function which is typically a

* The backpropagation network is often called a "multilayer perceptron" in literature; this name is, in fact, a short for the full correct name "multilayer perceptron with backpropagation of error". Thus, as the main feature of this ANN paradigm is just this backpropagation of error, the name "backpropagation network" is somewhat more telling and is usually preferred in more practically oriented studies. The name "multilayer perceptron" alone actually better suits another ANN paradigm which comprises several layers of the processing units identical to those in the BPN, but uses no backpropagation of error. The perceptron is normally a two-layer ANN consisting only of the input layer and a single processing layer. Nevertheless, it is possible to use more processing layers while training only the weights in connections between the processing units of the two last layers; this way a true multilayer perceptron is created which even has some practical utility (see, e.g., [48]). In any case, the name backpropagation network is used solely in this study.

** In literature, the perceptron is sometimes also considered to be a special case of the BPN with two layers only. In this study, however, the BPN and the perceptron are considered to be distinct ANNs due to the fundamental difference of their training algorithms, i.e. due to the fact that there is no backpropagation of error in the perceptron training (cf. also the footnote above).

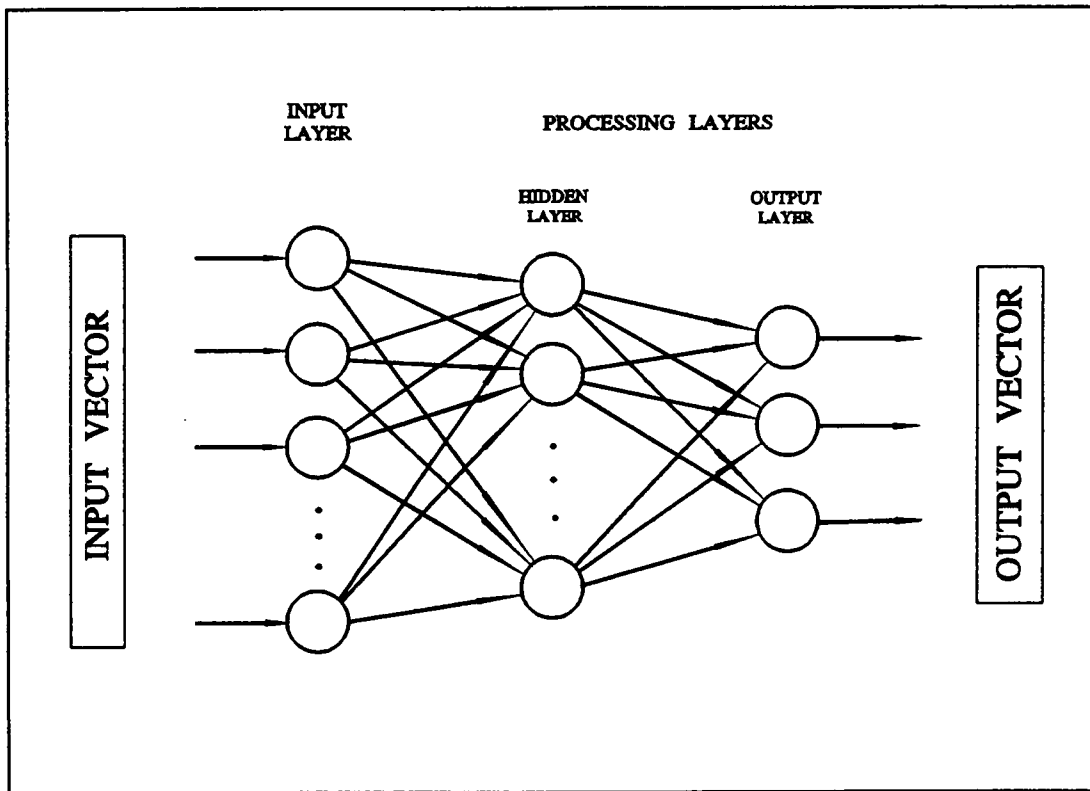


Figure 4 - Topology of the backpropagation network

nonlinear sigmoid function compressing the signal into the range from zero to one (see Figure 5). Each of the BPN input and hidden layers is usually augmented with an additional "processing" unit called a bias unit which has its output value fixed to one (it speeds up BPN training by effectively adjusting the threshold values of the activation functions of the processing units in the following layer).

The BPN utilizes supervised training based on the minimization of a sum of squared errors of BPN output values

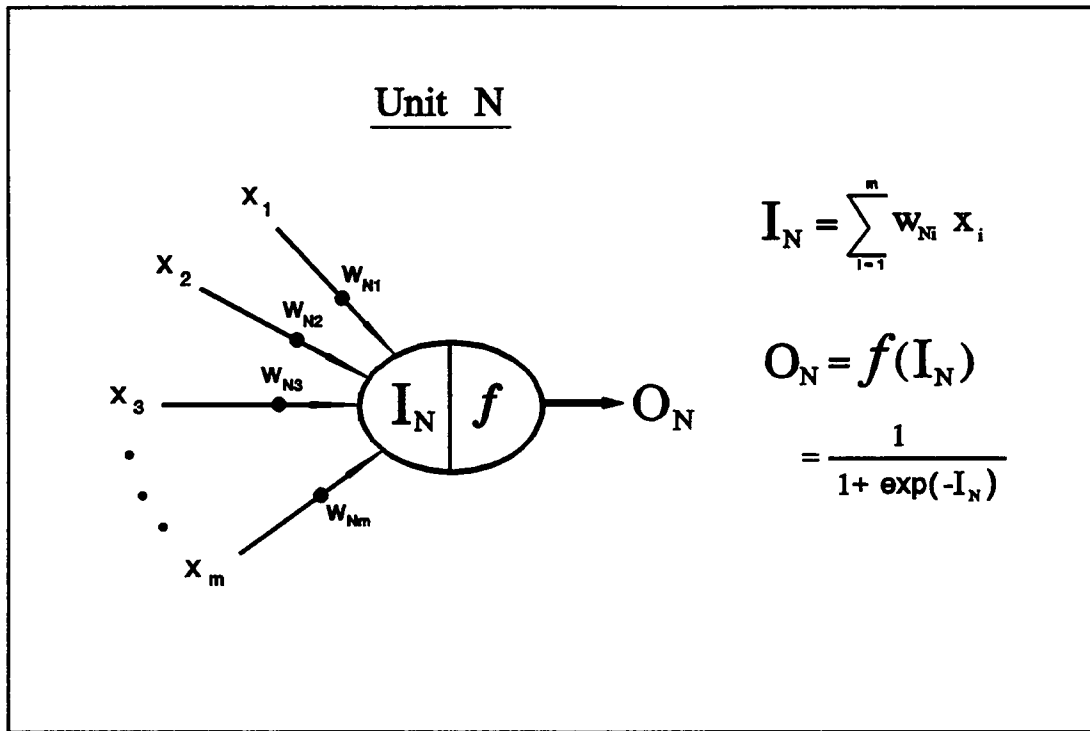


Figure 5 - Processing unit of the backpropagation network

over the set of training input-output vector pairs. This sum is a function of the weights in connections of the processing units. Before the training starts, the weights are randomized to small arbitrary (positive and negative) values. During the training, the BPN is presented with individual input vectors from the set of training samples and responds with certain output vectors. These output vectors are compared with the target output vectors also given in the training set, and the weights are updated in a way ensuring a decrease of the difference between the actual and target output vectors (typically, the steepest descent optimization algorithm is

used). All the individual input-output vector pairs from the training set are presented to the BPN in this way repeatedly until the error decreases to an acceptable level.

4.2.2. Functional Link Network

The functional link network (FLN) [53,54] is, in fact, just an extension of the BPN. It retains the structure of the BPN and differs only by a rather special and very complex pre-processing of the input data presented to it. This pre-processing consists in an expansion of the original input vectors given in the set of training samples by adding some new synthetic inputs. These new inputs are generated by applying certain nonlinear transformations to the original input vectors. Thus, no new information is actually provided to the FLN but representation of the available information is enhanced. The objective of this input vector enhancement is a reduction of nonlinearity of the evaluated problem. This allows a decrease in the number of processing units in the hidden layer, or even a complete elimination of the hidden layer.

There are a number of transformations which may be employed for the input vector enhancement, and it is usually not easy to find those which are optimal for the problem of concern. Naturally, when there is any reason to assume that certain

transformations could be useful, these should be applied first. Nevertheless, this information is not usually available, and experimentation is necessary to identify the suitable transformations. When the function to be approximated is likely to involve combining of the individual inputs, the transformations applied should include the products of the inputs (unfortunately, even for a moderate number of the original inputs, this approach leads to an explosion of the number of new inputs). On the other hand, when the function to be approximated is likely to involve some complex mathematics, it may appear useful to expand the original inputs with some mathematical series (e.g. polynomial or trigonometric). In more general cases, it may even become necessary to employ a certain combination of both approaches mentioned.

4.2.3. Modular Neural Network

The modular neural network (MNN) [52,55,56] is another rather special extension of the BPN. It is comprised of several independently operating expert networks competing to produce the correct response to individual input vectors, and a gating network mediating this competition. Basically, the MNN consists of the input layer, a processing "super-layer" comprising the expert networks and the gating network, and the output layer. The MNN input units are fully connected to the

input units of both the expert networks and the gating network, but there are no weights in these connections. The expert networks output units are fully connected to the MNN output units with connections having weights whose values correspond to the values produced in the output units of the gating network. The MNN output units perform a summation of the weighted incoming signals without applying any activation function to the result. A topology of the MNN with two input units, three expert networks, and a single output unit is shown in Figure 6.

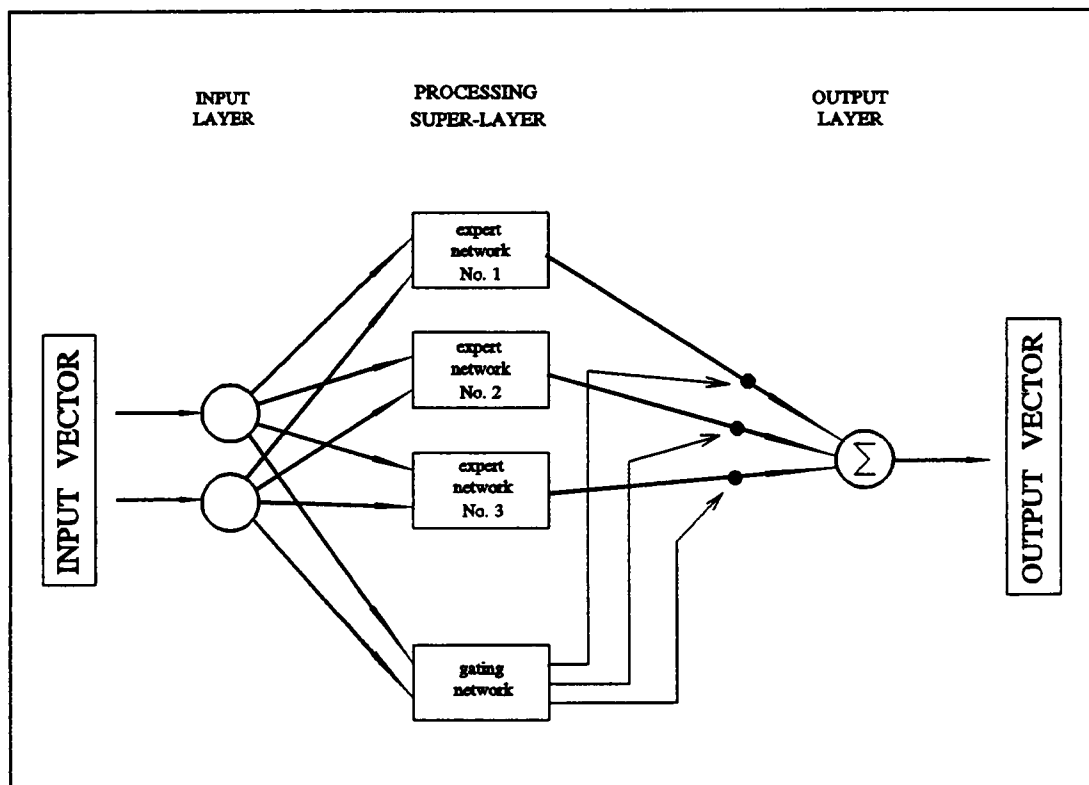


Figure 6 - Topology of the modular neural network

For the function approximation task the expert networks are typically simple perceptrons (i.e., in fact, degenerated BPNs with two layers only) with their output units performing only a simple summation without using any activation function. However, the expert networks can also be the classical three-layer BPNs (or even another MNNs, thus creating a complex hierarchical structure), and their output units may use nonlinear activation functions. The structure of the expert networks has to be chosen in such a way, that the task of concern cannot be solved by a single expert network, because otherwise the MNN during training would gradually degenerate into the ANN with the structure of this expert network (i.e. all the input vectors would be processed solely by one of the expert networks). In any case, all the expert networks in the MNN have to have the same structure with the same number of layers and identical processing units in them.

The gating network is a fully-connected feedforward ANN, having typically only two layers. There is one output unit in the gating network for each expert network in the whole MNN (hence, the weights in all connections between the output units of each expert network and the output units of the whole MNN are identical). Similarly to the BPN processing units, the gating network output units sum the weighted signals received from the input units and filter the result through an activation function. This activation function, however, is

the so-called "softmax" function which essentially normalizes activations of output units and amplifies differences between them - see Figure 7.

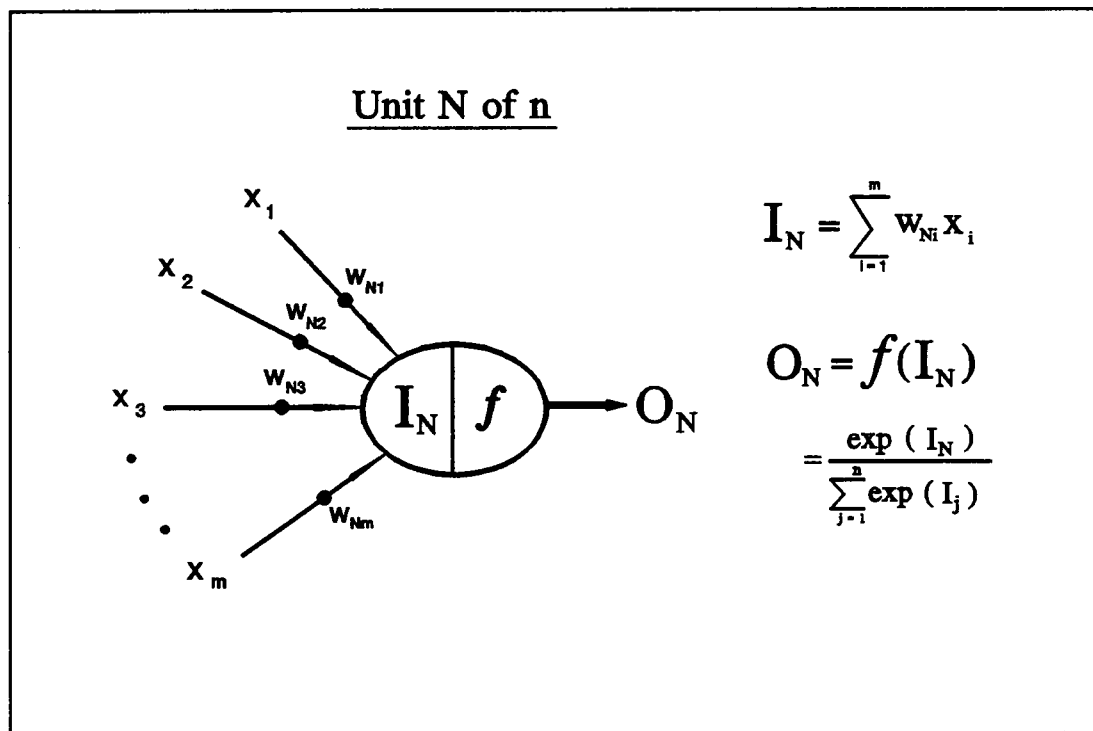


Figure 7 - Output unit of the MNN gating network

Operation of the MNN consists in division of a complex task to be solved into several simpler sub-tasks, finding of separated solutions for these sub-tasks, and combination of these sub-solutions into the desired solution of the original complex task (this approach is sometimes referred to as a "principle of divide and conquer"). To achieve this, the MNN

utilizes a special combination of supervised and unsupervised training based on maximization of the likelihood function which represents a product of the probabilities of generating the correct output vectors for individual input-output vector pairs. These probabilities are typically modeled using a mixture model (i.e. a linear combination) of multivariate Gaussian distributions, which characterize conditional probabilities of producing the correct output vectors by the individual expert networks for a given input vector from the training set. To maximize the likelihood function, its unknown parameters have to be optimized, i.e. properly adjusted during training. In the MNN, these parameters represent the weights in connections of the expert networks processing units, and the weights in connections of the expert networks output units and the output units of the whole MNN. All these weights are trained simultaneously starting from small arbitrary values. During the training, the MNN is presented with individual input vectors from the set of training samples to which individual expert networks and gating network respond with certain output vectors.

The basic training of the expert networks is supervised and utilizes a procedure very similar to that used by the BPN. The resulting update values of the weights for each expert network are, however, modified by a probability that the particular expert network is allowed to produce the particular

MNN output vector. The value of this probability is determined as a product of the distance between the output vector produced by the given expert network and the target output vector (typically Euclidean), and the value produced by the gating network output corresponding to this expert network. Values of these products, determined for individual expert networks, are then processed through the softmax function with the aim of amplifying the outcome of the competition of the expert networks. The resulting values of the products effectively represent posterior probabilities that the particular expert networks are allowed to produce the particular MNN output vector, while the values produced by the corresponding gating network outputs represent prior probabilities of it. The training of the gating network is essentially unsupervised and aimed at minimization of the differences between these prior and posterior probabilities. The training procedure for the gating network is, in fact, the same as used by the BPN; the elements of the target output vector of the gating network are determined by the posterior probabilities of the corresponding expert networks (these are, to a great extent, independent on the MNN target output vector). All the input-output vector pairs in the training set are in the described way repeatedly presented to the MNN until the MNN error decreases to an acceptable level, or until the MNN reaches a steady state when the values of its weights stop changing.

4.2.4. Radial Basis Function Network

The radial basis function network* (RBFN) [52,57-59] always consists of three layers: the input layer, the pattern (or hidden) layer, and the output layer (the topology of the RBFN is thus identical to the BPN as shown in Figure 4). It is a fully-connected and feedforward ANN with all connections between its processing units provided with weights. The individual pattern units compute their activation using a radial basis function; typically the Gaussian kernel function is used as shown in Figure 8. The activations of pattern units essentially characterize the distances of centers of radial basis functions of the pattern units from a given input vector. The radial basis functions thus produce localized, bounded, and radially symmetric activations, i.e. activations rapidly decreasing with the distance from the functions centers (in contrast, the BPN sigmoidal activation functions produce global and unbounded activations). Use of the radial basis activation functions requires a careful choice of the number of the pattern units to be used for a specific application, especially when a good generalization is needed; the areas of significant activation have to cover all the

* In literature the name "radial basis function network" has sometimes more general meaning and refers to all ANNs using radially symmetric and bounded activation functions (as, e.g., the generalized regression neural network, the probabilistic neural network, or even the counterpropagation network). In this study, however, the name RBFN always refers to the ANN with structure described in section 4.2.4.

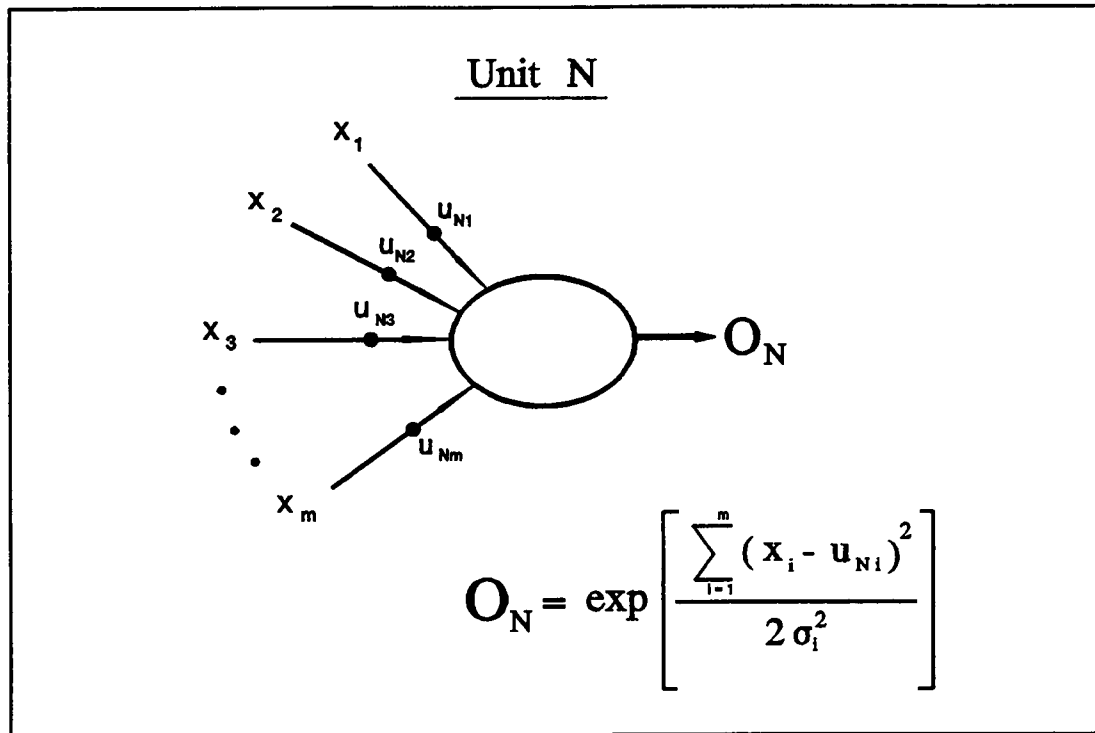


Figure 8 - Pattern unit of the radial basis function network

input space while overlapping in just the right way. For function approximation applications, this means that the samples included in the training set have to evenly represent all possible input vectors. The output units of the RBFN simply sum the weighted activations of individual pattern units without using any activation function. To speed up the training, the pattern layer is augmented with a bias unit which has its activation value fixed to one.

The training of the RBFN differs substantially from the training used for the BPN. It consists of two separate

stages. During the first stage, parameters of the radial basis functions (i.e. their centers and widths) for individual pattern units are set using the unsupervised training. The centers of radial basis functions of the individual pattern units define input vectors causing maximal activation of these units. Location of these centers is the first step of the training and is conducted with the help of some clustering algorithm (typically, the k-means algorithm is used). The clustering algorithms usually operate iteratively, and the clustering process is finished when locations of the centers for individual pattern units stabilize. The resulting values of individual elements of the center vectors are then directly used as values of the weights in connections between the input units and the corresponding pattern units. The widths of radial basis functions of the individual pattern units (denoted σ in Figure 8) determine the radii of the areas of the input space around the centers where activations of these units are significant. Their determination is the next step of the training and is performed using some of the "nearest-neighbor" heuristic.

In the second training stage, the weights in connections between the pattern units and the output units are determined using the supervised training based (as when training the BPN) on minimization of a sum of squared errors of RBFN output values over the set of training input-output vector pairs.

Before the training starts, these weights are randomized to small arbitrary values. At that stage, the weights in connections between the input units and the pattern units and the parameters of the radial basis functions of the pattern units are already set as determined in the first training stage and are not subject to any further changes. During this training, the RBFN is presented with individual input vectors from the set of training samples and responds with certain output vectors. These output vectors are compared with the target output vectors also given in the training set, and the individual weights are updated in a way ensuring a decrease of the difference between the actual and target output vectors (typically, the steepest descent optimization algorithm is used). The individual input-output training pairs are presented to the RBFN in this way repeatedly until the error decreases to an acceptable level.

4.2.5. Generalized Regression Neural Network

The generalized regression neural network (GRNN) [52,59-61] is a special extension of the RBFN. It is a feedforward ANN consisting of four layers: the input layer, the pattern layer, the summation layer, and the output layer - see Figure 9. While the processing units in the first three layers are fully-connected, each output unit is connected only to some processing units in the summation layer. The function of the

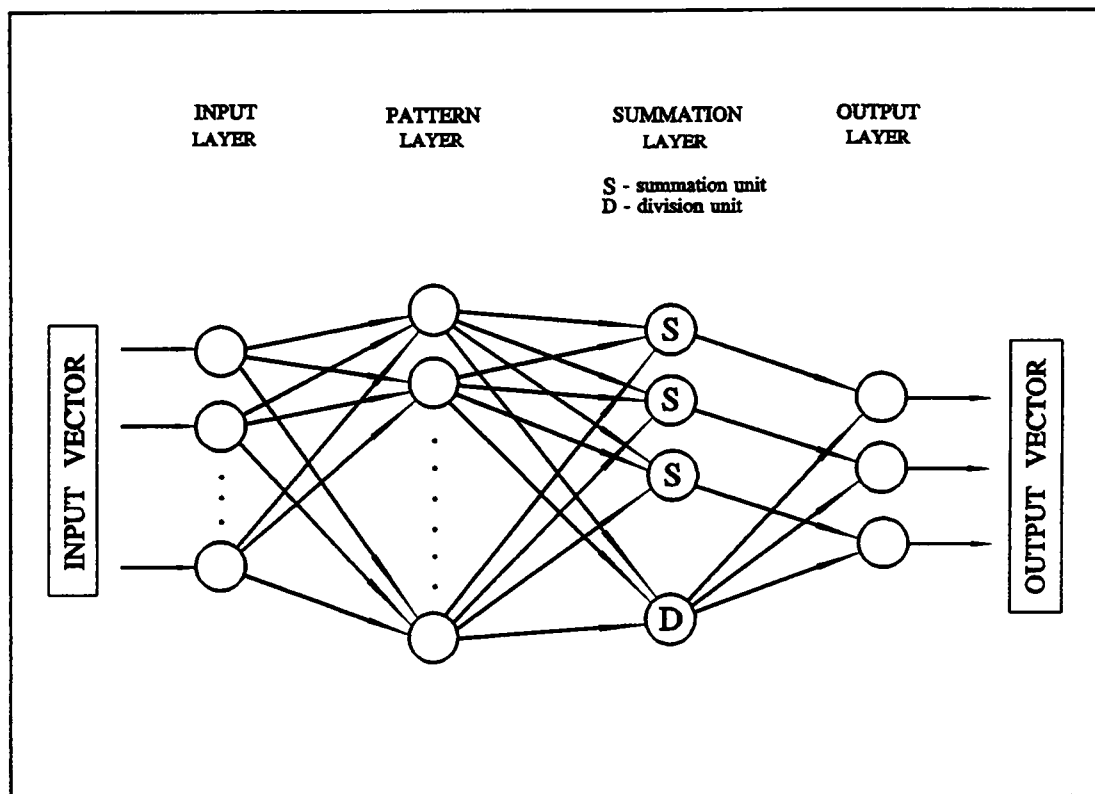


Figure 9 - Topology of the generalized regression neural network

input and pattern layers of the GRNN is exactly the same as it is in the RBFN (see section 4.2.4). The summation layer comprises two different types of processing units: the summation units and a single division unit. The number of the summation units is always the same as the number of the GRNN output units; their function is essentially the same as the function of the output units in the RBFN. The division unit only sums the weighted activations of the pattern units without using any activation function. Each of the GRNN output units is connected only to its corresponding summation

unit and to the division unit; there are no weights in these connections. The function of the output units consists in a simple division of the signal coming from the summation unit by the signal coming from the division unit. The summation and output layers together basically perform a normalization of the output vector, thus making the GRNN much less sensitive to the proper choice of the number of pattern units than the RBFN. The overlapping of radial basis functions of individual pattern units is not a problem for the GRNN; in fact, it turns out to be an important parameter allowing the user to influence generalization capabilities of the GRNN. In general, larger values of the width of radial basis functions of the pattern units result in a smoother interpolation of the values of elements of the output vectors among the values corresponding to the centers of radial basis functions of the individual pattern units.

The training of the GRNN is rather different from the training used for the RBFN. It is completed after presentation of each input-output vector pair from the training set to the GRNN input layer only once, i.e. both the centers of the radial basis functions of the pattern units and the weights in connections of the pattern units and the processing units in the summation layer are assigned simultaneously. The training of the pattern units is unsupervised, as in the case of the RBFN, but employs a

special clustering algorithm which makes it even unnecessary to define the number of pattern units in advance. Instead, it is the radius of the clusters what needs to be specified before the training starts. The first input vector in the training set becomes the center of the radial basis function of the first pattern unit. The next input vector is then compared with this center of the first pattern unit and is either assigned to the same pattern unit (cluster) if its distance from this center is less than the pre-specified radius, or becomes the center of the radial basis function of the next pattern unit otherwise. In the same manner, all the other input vectors are compared one-by-one with all the pattern units already set, and the whole pattern layer is thus gradually built. During this training, the determined values of individual elements of the center vectors are directly assigned to the weights in connections between the input units and the corresponding pattern units. Owing to much lower sensitivity of the GRNN to the overlapping of the radial basis functions of the pattern units, the widths of radial basis functions of the individual pattern units need not to be set according to the resulting structure of the pattern layer. Instead, their setting typically becomes the subject of experimentation as their values determine generalization properties of the GRNN.

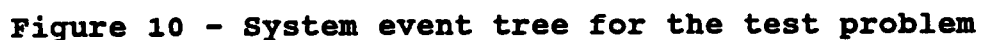
Simultaneously with building the pattern layer, the values of the weights in connections between the processing units in the pattern layer and the summation layer are also set using the supervised training. The weights in connections between each pattern unit and the individual summation units are directly assigned with values identical to the elements of the output vector corresponding in the training set to the input vector which formed the center of the radial basis function of that particular pattern unit. In case that some additional input vectors in the training set are assigned to the same pattern unit, values of the elements of their corresponding output vectors are simply added to the previous values of these weights. In the same time, the weight in the connection of each pattern unit and the division unit, which is originally set to zero, is increased by one for each input vector from the training set which is assigned to this pattern unit.

5. QUANTIFICATION OF PSA MODELS WITH ARTIFICIAL NEURAL NETWORKS

The reason for an effort to utilize the ANNs for the quantification of PSA models in risk monitors consists in an intention to exploit the exceptional speed of their computation. This chapter presents in detail the approach used and the results obtained when investigating performance of the ANNs in this application. To explore the ANNs performance, an example problem consisting in a very simple test PSA model was prepared first.

5.1. Test PSA Model

The test PSA model characterizes mitigation of the initiating event "small loss of coolant accident" in a pressurized water reactor. The front-line systems taking part in mitigation of this accident are typically the reactor scram system, the high pressure and the low pressure subsystem of the emergency core cooling system (ECCS), and the system of water accumulators. To prevent damage to the reactor core, a successful action of the reactor scram system is needed, first of all. Besides that, a successful operation of either the high pressure ECCS, or the water accumulators and the low pressure ECCS together, is also necessary. This situation is modeled by the system event tree shown in Figure 10.



84

include the most typical features of the PSA models (this was also reason for the particular choice of the initiating event because success criteria of the systems involved in its mitigation well represent all the criteria typically appearing in the PSA models). The fault trees thus included a typical 200% level of redundancy with its usual doubling for actuating signals, support systems shared by different front-line systems, and also a voting gate which is sometimes used for modeling of the operating scheme of especially crucial systems involving a complex redundancy. In contrast, the common cause failures were not considered in the test problem fault trees, because their influence was effectively the same as the influence of the shared support systems. The only PSA code available in the Nuclear Engineering Department of the University of Tennessee at Knoxville and able to carry out reliable quantifications of the PSA model, is the code Tree-Master [62]. Unfortunately, this code has been designed for evaluation of fault trees only, and is not equipped for handling event trees. Therefore, to be able to quantify the test PSA model using this code, the event tree shown in Figure 10 was converted into a fault tree. In addition, the sequence denoted five in the event tree in Figure 10, which represented an isolated failure of the reactor scram system, was omitted in the converted fault tree because it had virtually no influence on the core-damage probability (owing to its typically very low probability and especially high level of

redundancy). The resulting combined (master-logic) fault tree comprising both the converted event tree and the fault trees of individual systems is shown in Figure 11. This tree thus represents the whole test PSA model used for exploration of performance of the ANNs in this study. A brief description of meaning of the model basic events is given in Table 1 (ESAS is short for the engineered safeguards actuating system). Probabilities of these basic events for nominal configuration of the NPP (i.e. the configuration, when all safety features represented by the basic events of the PSA model are considered available), also given in Table 1, were quantified with estimated values more or less typical for NPP equipment. All the given values represent probabilities of failure on demand averaged over testing intervals of the individual systems/components.

5.2. Problem Implementation

To be able to derive a well-founded decision about further safe operation of the NPP, the risk monitor inference engine has to be supplied with the current risk parameters of the NPP of concern. The problem to be implemented is determination of these risk parameters by using the ANNs.

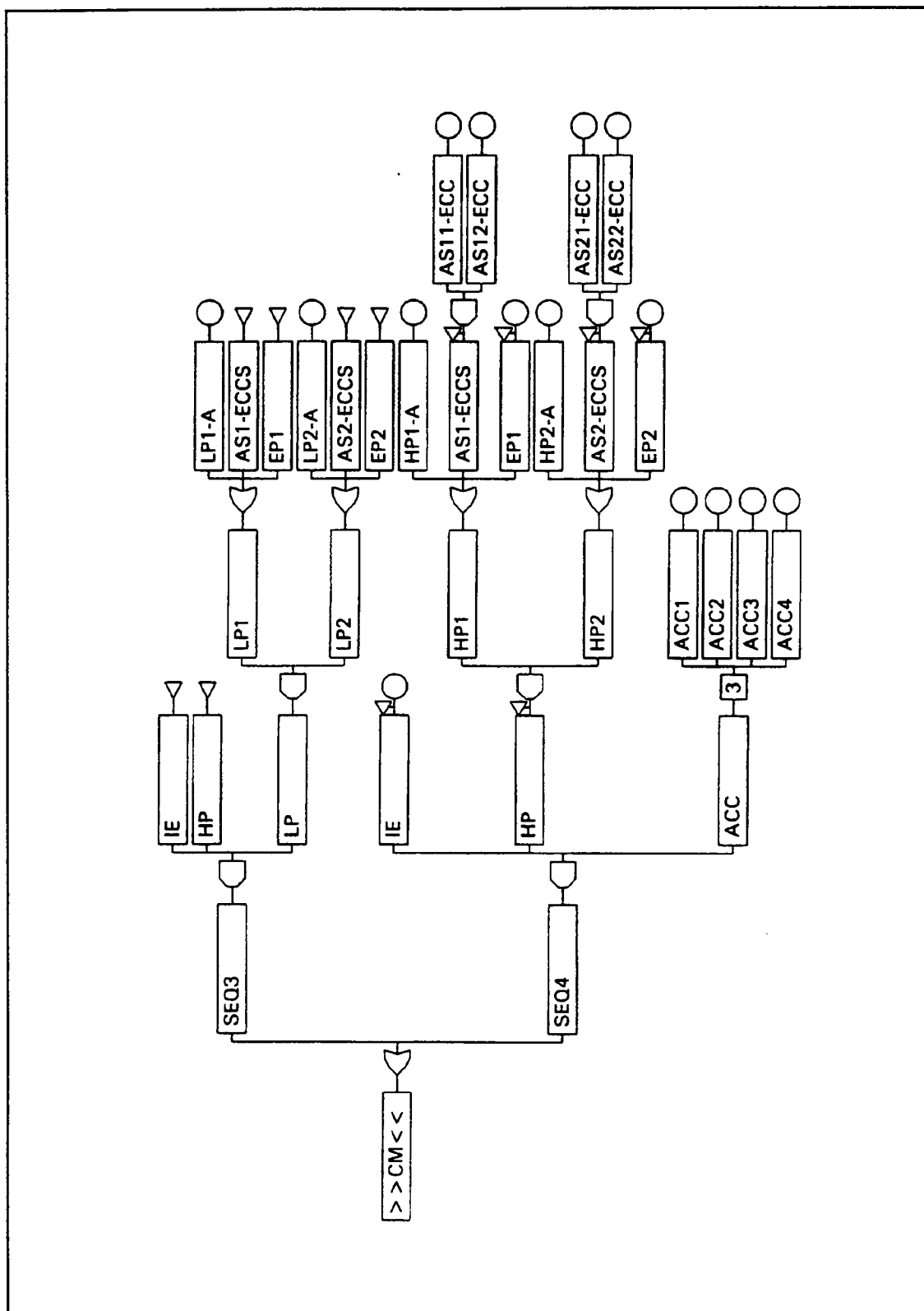


Figure 11 - Test PSA model

Table 1 - Basic events of the test PSA model

No	Name	Description	Probability
-	IE	probability/frequency of the initiating event	1×10^{-4}
1	LP1-A	actual failure of the train 1 of the low pressure	1×10^{-2}
2	LP2-A	actual failure of the train 2 of the low pressure ECCS	1×10^{-2}
3	HP1-A	actual failure of the train 1 of the high pressure ECCS	1×10^{-2}
4	HP2-A	actual failure of the train 2 of the high pressure ECCS	1×10^{-2}
5	EP1	no train-1 dieselgenerator power supply for the ECCS	1×10^{-1}
6	EP2	no train-2 dieselgenerator power supply for the ECCS	1×10^{-1}
7	AS11-ECC	no ECCS actuation signal from the set 1 of the ESAS train 1	1×10^{-3}
8	AS12-ECC	no ECCS actuation signal from the set 2 of the ESAS train 1	1×10^{-3}
9	AS21-ECC	no ECCS actuation signal from the set 1 of the ESAS train 2	1×10^{-3}
10	AS22-ECC	no ECCS actuation signal from the set 2 of the ESAS train 2	1×10^{-3}
11	ACC1	general failure of water accumulator No. 1	1×10^{-3}
12	ACC2	general failure of water accumulator No. 2	1×10^{-3}
13	ACC3	general failure of water accumulator No. 3	1×10^{-3}
14	ACC4	general failure of water accumulator No. 4	1×10^{-3}

5.2.1. Risk Parameters of Interest

There are several risk parameters of the NPP that the risk monitor needs to know before it can derive informed recommendations concerning further operation of the NPP. As discussed in detail in section 3.3.3.3 of this study, these parameters include the core-damage frequency of the NPP, the risk control importances of selected basic events and groups of basic events of the PSA model, and the risk reduction importances of selected basic events of the PSA model.

5.2.1.1. Core-Damage Frequency Calculation

The principal risk parameter to be calculated is the NPP core-damage frequency for the current configuration of the NPP. The PSA model normally consists of a certain number of event trees, one for each initiating event considered (cf. section 2.1.1). These event trees characterize scenarios of mitigation of individual, mutually independent initiating events (simultaneous occurrences of two or more independent initiating events are not considered due to their extremely low frequency); the test PSA model represents one such event tree. When quantifying these event trees (and the fault trees modeling their individual events), one obtains the core-damage frequencies for the particular initiating events. The core-damage frequency of the NPP is then determined as a

summation of the core-damage frequencies obtained for the individual initiating events.

5.2.1.2. Risk Control Importances Calculation

Other risk parameters required to be calculated are the risk control importances of selected basic events and groups of basic events of the PSA model (these groups typically represent whole systems or system trains).

In general, the PSA model characterizes influence of certain basic events on the NPP risk while these basic events represent various features provided for mitigation of possible accidents. The risk control importance of a given feature (or a group of features) quantifies the importance of this feature for maintaining the achieved level of the NPP risk. Its value generally characterizes an increase in the NPP core-damage frequency when the feature (or the group of features) is removed, i.e. is not available for the accident mitigation.

There are several different importance measures used for quantification of the risk control importances [63]. The importance measures typically used in the PSA are the Birnbaum importance [63], and the risk achievement worth (RAW) [64]. Of these measures, the RAW seems to be very suitable for computation using ANNs. The RAW of a certain event x can be

computed using the following formula [64]:

$$RAW(x) = \frac{R1(x)}{CR} \quad (1)$$

where $R1(x)$ is the NPP core-damage frequency when the probability of the basic event (or a group of these events) x is set to unity, and

CR is the core-damage frequency of the NPP with its current configuration (i.e. the configuration for which the importances are computed).

It is theoretically possible to train a single ANN to produce values of the RAWs of individual basic events of the PSA model as elements of its output vector. Nevertheless, as the RAWs need to be computed, in general, for all basic events of the PSA model, this ANN would have a very high number of output units which would be, actually, the same as the number of input units. In addition to that, should this ANN be required to provide the RAWs for certain groups of the basic events, it would be even necessary to add another output unit for each of these groups (thus ending with the number of output units of this ANN even exceeding the number of its input units). The training of such an ANN would inevitably be very difficult and its success quite unlikely. Other possible way would be to train a separate ANN for computation of each

of the required RAWs but it would result in a necessity to train an enormous number of ANNs. Fortunately, there is a better way of calculating this importance measure by using equation (1) and the ANN trained to produce the core-damage frequency values for different configurations.

The variable $R1(x)$ in equation (1) simply represents the core-damage frequency of the NPP configuration which differs from the current configuration only by one additional guaranteed event which represents the basic event x , i.e. it differs only by having the probability of the event x set to unity instead to its nominal value. Hence, to obtain the RAWs for all basic events of the PSA model (or any group of these events), it is only necessary to carry out repeated computations with the same ANN while appropriately changing the input vector. The desired RAWs are then easily computed using equation (1)*. Although this procedure is certainly more time consuming than a single pass through a specially trained ANN (if it could be developed), the extraordinary speed of the ANN computations should still enable the risk monitor to produce its response in a reasonable time.

* Of course, this operation does not need to be carried out for the events which are considered guaranteed already under the current configuration because their RAWs are obviously equal to one.

5.2.1.3. Risk Reduction Importances Calculation

The last required risk parameters are the risk reduction importances of selected basic events. As was already mentioned, the PSA model characterizes the influence of various features, provided for mitigation of possible accidents, on the NPP risk. The risk reduction importance of a given feature quantifies the importance of this feature for a possible further reduction of the current risk level. Its value generally characterizes a decrease in the NPP core-damage frequency when the feature is optimized, i.e. its realization is guaranteed.

Again, there are several different importance measures used for quantification of the risk reduction importances [63]. The importance measures typically used in the PSA are the Vesely-Fussel importance* [63], and the risk reduction worth (RRW) [64]. Of these measures, the RRW seems to be suitable for computation using the ANNs. The RRW of a certain event x can be computed using the following formula [64]:

$$RRW(x) = \frac{CR}{R0(x)} \quad (2)$$

* This importance measure is sometimes also called the Fussel-Vesely importance.

where CR is the core-damage frequency of the NPP with its current configuration (i.e. the configuration for which the importances are computed), and $RO(x)$ is the NPP core-damage frequency when the probability of the basic event x is set to zero.

As is the case for the RAW, a direct computation of the RRWs for all basic events of the PSA model with a single ANN or many separate ANNs would cause great difficulties. It is therefore necessary to resort to the approach proposed for computation of the RAWs while using equation (2). The variable $RO(x)$ in equation (2) represents the core-damage frequency of the NPP configuration which differs from the current configuration only by exclusion of the event x , i.e. it differs only by having the probability of the event x set to zero. Therefore, to obtain the RRWs for all basic events of the PSA model, it is only necessary to carry out repeated computations with the same ANN while appropriately changing the input vector. The desired RRWs are then computed using equation (2).

Unfortunately, the ANN trained to compute the core-damage frequencies for different NPP configurations cannot be directly used for these calculations. This ANN is trained to respond to the input vectors with their individual elements never acquiring lower values than the corresponding nominal-

configuration probabilities of the individual basic events of the PSA model. Hence, the input vectors with null elements are clearly out of the range for which this ANN is trained and it is almost certain that its responses to such vectors would be wrong. Therefore, the ANN assigned with computation of the RRWs needs to be trained with an extended set of the training samples. This extended set has to include the input-output vector pairs representing the configurations with one of the basic events of the PSA model considered impossible. Although it might be possible to use a single ANN trained to deal with this extended spectrum of possible configurations for all risk-monitor computations, it would probably be rather unwise because the resulting ANN would be likely marked with relatively high uncertainties in its output vectors. Thus, it is probably better to train one ANN with the basic training set and assign it with computation of the core-damage frequencies and RAWs, and to train another separate ANN with the extended training set and assign it with computation of the RRWs. This way, only the resulting RRWs will include the inevitable higher level of uncertainty.

5.2.2. Gathering the Data for ANN Training

The task of the ANN in the risk monitor consists in the quantification of the PSA models; the models thus represent unknown functions to be approximated by the trained ANN. The

independent (input) variables of these functions are the probabilities of basic events of the PSA models for the specific NPP configuration, the only dependent (response) variable of these functions is the core-damage frequency of the NPP for the specific configuration (cf. Figure 3). Nevertheless, the probabilities of some basic events of the PSA models remain the same for all possible configurations (it concerns, e.g., initiating events, some human errors, or external events). In addition, there are some basic events whose probabilities depend so closely on the probabilities of some other basic events that they do not count as the truly independent variables of the PSA models (it concerns, e.g., common cause failures; this notion of "close dependence" is, however, valid only in the context of the specific changes of the probabilities which are to be expected in the risk monitors). These basic events thus, in fact, do not represent the true input variables in the approximated functions. In the specific case of the test PSA model, almost all of its basic events characterize the failures of the NPP components/systems whose probabilities differ for the individual NPP configurations. The only exception is the initiating event (named IE in Figure 11), whose probability/frequency is, of course, constant. After omission of this event there are 14 input variables left in the test PSA model.

During the training, the ANN needs to be taught to map individual input vectors, whose elements represent the input variables of the given function (PSA model), into the output vectors, whose single element characterizes the corresponding response variable. The training thus has to be supervised, i.e. both the input and output vectors have to be shown to the ANN. Therefore, the training data set to be used for the ANN training has to contain sample pairs consisting of the input vectors and the corresponding output vectors. In contrast with typical ANN applications, there is no problem of the strictly limited data available for training in the case of ANNs emulating the PSA code. To obtain any input-output vector pair needed, a trial evaluation of the PSA model for the particular configuration is simply performed using any PSA code available. On the other hand, the number of these trial evaluations which can be realized in practice is obviously not unlimited (especially considering the relatively slow computation of the PSA codes). Nevertheless, it is also possible to make use of the relation between the RAWs and the core-damage frequency of the evaluated configuration as it is expressed by equation (1). As the modern PSA codes readily compute the RAW values, it is possible to determine the core-damage frequencies for many configurations by using this relation, thus avoiding the necessity to actually evaluate

them with the code*. The trial evaluations for the test PSA model were carried out using the code Tree-Master [62].

The configurations to be evaluated have to be carefully selected so that the output vectors obtained cover the whole output space, and also provide a good representation of the input space. Particularly, the boundary configurations, and the configurations with similar input vectors which yield significantly different output vectors need to be included [65,66]. Obviously, the lower boundary of the possible output vectors (i.e. of the possible core-damage frequencies) is set by the nominal configuration when all NPP components/systems are available, and the upper boundary is set by the configuration when all NPP components/systems, that can possibly fail, are really unavailable. Other configurations necessary to be included in the training set are naturally all configurations with always just one out of all possibly failed components/systems considered guaranteed. Of course, if only these basic input-output vector pairs were used for training, the ANN would very unlikely be capable of producing the correct output vectors for the configurations with more than one basic event considered guaranteed. Therefore, some

* Of course, this technique can be used only for the configurations involving component/system failures, or the maintenance actions with no possibility to recover the feature currently maintained. In other cases (i.e. for the maintenance where there is a possibility to recover the maintained feature and actually use it for the accident mitigation), the probability of the corresponding basic event is not equal to one and equation (1) cannot be used.

additional sample vector pairs, representing these intermediate configurations, have to be added to the training set. These additional samples need to be chosen using the method of trial and error (which is, actually, rather common in the ANN field), i.e. by gradual addition of the samples intended to help the ANN in learning the correct mapping for the configurations which it had failed to evaluate correctly when trained with the previous training set.

In the case of the test PSA model, only two possible probability values were used for the individual basic events with variable probabilities: the nominal probabilities as given in Table 1 when the components/systems represented by these events were considered available, and the probabilities equal to unity when they were considered unavailable (i.e. the recovery actions were not modeled as it would be necessary for any real model - cf. section 3.3.1.1). The core-damage frequencies (i.e. probabilities of the fault tree top event) obtained for the lower and upper boundary configurations were 1×10^{-6} and 1×10^{-4} , respectively (logically, the upper boundary value was equal to the probability of the initiating event itself as all the features assigned to mitigate the accident were considered unavailable for this configuration).

5.2.3. Preprocessing the Data for ANN Training

In general, the way the input and output vectors from the training set are presented to the ANN during its training has very significant influence on the speed and even on the very success of the training process. Therefore, a proper preprocessing of the data can relieve the ANN of a lot of unnecessary burden [54,65-68].

5.2.3.1. Preprocessing of the Input Vectors

The ANNs processing continuous-valued data, as is the case for the application of interest, generally assign a great importance to the magnitudes of values of their inputs [54,67]. Therefore, it is usually useful to equalize the importance of individual elements of the input vectors by scaling them to the same range of values. Nevertheless, in the application of concern, all the elements of the input vectors represent the same quantities, probabilities. Any possible differences in magnitudes of values of individual elements thus likely have a real significance in production of the model response value. Hence, this scaling is not only unnecessary in this application, but it would be probably even counterproductive.

In general, the calculation of the core-damage frequencies for individual initiating events of the PSA model requires evaluation of the fault trees which consists in the determination of the minimal cut sets of these fault trees. The minimal cut sets of individual fault trees are then combined together and form the minimal cut sets for the initiating events. Each minimal cut set includes the initiating event itself, and one or, much often, several basic events; the frequency of the minimal cut set is defined as a product of the initiating event frequency and the probabilities of these basic events. The core-damage frequency for the initiating event is then roughly equal to a summation of the frequencies of its individual minimal cut sets (in fact, this summation is only an approximation - cf. the footnote in section 3.3.3.1). The main operation performed with the basic event probabilities is thus their multiplication. As the basic event probabilities also form the inputs of the ANN, this multiplication is the operation which has to be performed by the ANN. But, the multiplication is actually an operation which is quite difficult to carry out with ANNs, because the ANNs (and humans as well) usually depend on an associative learning which is rather unsuitable for the multiplication [69]. In contrast, many ANN paradigms (e.g., the BPN) are inherently suitable for performing summations of its inputs. Thus, it is clear that for these ANN paradigms it is very useful to preprocess elements of

their input vectors with logarithmic transformation which effectively converts the operation carried out by the ANNs from the multiplication to the summation [54,67]. To achieve this conversion, the transformation performed with the output values has to be, naturally, exactly the same as the one used for preprocessing of the input vectors. The decadic logarithmic transformation is optimal for this task owing to its advantage in a relative transparency of the transformed values which retains some clear relation to the usual perception of numbers in the decadic system. The values of elements of the input vectors, which are actually used in the ANN training, thus are not directly the probabilities of individual basic events of the PSA model, but the decadic logarithms of these probabilities instead.

5.2.3.2. Preprocessing of the Output Vectors

In general, the output vectors again require equalization of importances of their elements as discussed for the input vectors in the previous section. Nevertheless, as the output vectors in the investigated application have only one element, there is no reason to be concerned with this issue.

The single element of the output vector represents the core-melt frequency of the NPP (for the particular case of the test PSA model this frequency corresponds to the probability

of the top event of the master-logic fault tree). Possible values of this frequency typically spread over several orders of magnitude. It would be rather unwise to try to train the ANN to produce such outputs directly, because the outputs provided by the ANNs inevitably include a certain error and the magnitude of this error tends to be typically uniform throughout the whole possible range. As a result, the percentile error of the output is much higher for the values closer to the lower limit of the range of possible outputs. For the application of interest however, it is desirable to achieve the uniform percentile error throughout the whole range of possible outputs. Therefore, it is necessary to train the ANN to produce a certain transformed parameter whose final inverse transformation yields the outputs with the required uniform percentile error. The transformation providing the required property is the logarithmic transformation. Again, specifically the decadic logarithmic transformation is optimal for this task for the same reasons as the ones leading to its preference for preprocessing of the ANN inputs (see section 5.2.3.1). In any case, when the decadic logarithmic transformation is used for preprocessing of the input vectors, it has to be used for preprocessing of the output vectors as well.

The values of the output vector's only element, which are actually used in the ANN training, thus are not directly the

core-damage frequencies, but the decadic logarithms of these frequencies instead. The expected uniform absolute error of the values produced by the trained ANN is then automatically transformed to the required uniform percentile error while performing the inverse transformation of the obtained values to the core-damage frequencies (i.e. while taking an inverse logarithm of the ANN output values).

Besides the preprocessing already discussed, it may prove to be necessary or at least useful to perform some additional scaling or transformations of values of the output vector element. The individual ANN paradigms produce different ranges of their output values and so the scaling of their training data set has to be customized individually for each paradigm in order to obtain the target output values corresponding to the natural ranges of the individual ANNs. Any additional preprocessing, if used, is described in the following sections specifically addressing the individual paradigms.

5.3. Application of Selected ANN Paradigms

There are several different ANN paradigms which can possibly be used for the quantification of PSA models: the backpropagation network (BPN), the functional link network

(FLN), the modular neural network (MNN), the radial basis function network (RBFN), and the generalized regression neural network (GRNN). A general description of these ANNs is given in chapter 4 of this study.

Of the listed ANN paradigms, by far the best understood and the most successful in practical applications is the BPN. Also in this study, the section dealing with application of the BPN to the problem of interest is more detailed and thorough than the sections addressing the other paradigms. The reason for this special treatment of the BPN is (besides the advantage of the abundance of guidance provided in literature) that the experience gained when experimenting with the BPN can be at least partially utilized when applying the other paradigms, which are still generally less well understood and much more difficult to apply in practice.

The commercial software package NeuralWorks Professional II/PLUS of the NeuralWare, Inc. (Version 386/387 1987-1993) [70] was used for implementation and performance exploration of all the ANN paradigms considered.

5.3.1. Backpropagation Network

The general description of the BPN structure and operation is given in section 4.2.1 of this study. This section

describes the approach used and the results obtained when applying the BPN to the problem of concern in general, and specifically to the test problem.

5.3.1.1. BPN Design

There are many decisions to make when a BPN for a certain application is to be designed [65]. Nevertheless, there is usually no need for a majority of them as they allow for use of such rarely sensible options as non-uniform connections between the layers, different parameters of activation functions of the processing units for individual processing layers or the individual units in one layer, or even some rather exotic activation functions of the processing units. It is generally wise to leave these options as the last recourse when the BPN utilizing the ordinary approaches have definitely failed to solve the problem. Fortunately, as it will be demonstrated on the test PSA model, there is no need for these rather desperate actions for the problem of interest. Thus, the normal fully-connected feedforward BPN using usual sigmoid activation functions in all processing units was used. To speed up the training, the bias units were added to both the input and hidden layers of the BPN. The basic slope of the sigmoid activation functions of 0.25 was used in all processing units, as the experiments done with using some other slope values did not proved to have any

significant influence on the BPN training for the problem of concern. Besides that, the BPN using no activation function in its output unit was also tested but it was marked with great difficulties during training and a rather poor performance of the resulting trained BPN.

An important decision, which is not always simple, is determination of the optimal number of hidden layers of the BPN. First of all, the function approximation task practically excludes a possibility of solving the problem without using any hidden layer at all (which is quite sufficient for the classification tasks with linearly-separable classes). A basic rule of thumb for determination of the number of hidden layers for ordinary problems is to use only one hidden layer as long as it is not quite clear that the BPN is not able to solve the problem and no preprocessing of training data is able to help. The reason for this rule is that the error backpropagation algorithm, as successful and effective it seems to be, gives only a very rough approximation of actual errors produced by the individual hidden units and so the addition of extra layers leads inevitably to the introduction of additional randomness into the BPN training [71]. Nevertheless, theoretical studies cited in reference [65] suggest that two hidden layers should be generally used for the function approximation applications. Fortunately, the BPN using a single hidden layer has appeared

to be able to handle the test problem fairly well. Some trials to train a four-layer BPN were carried out anyway but this BPN appeared to have great difficulties with convergence and always ended in a highly non-optimal local minimum.

The last decision to be made is choosing of the number of hidden units of the BPN (the number of units in the input and output layers is, normally, determined by the problem itself). This choice is very much problem specific and there are almost no rules to apply. The Kolmogorov Theorem does give a theoretical maximum of the number of units in a three-layer BPN (namely equal to $(2N+1)$ where N is number of the BPN inputs; for the test problem, $N = 14$ which yields 29 hidden units), but this information has a little use in practice [72]. To determine the optimum number of hidden units, many different BPNs were trained for the test problem, while trying 20, 10, 8, 7, 6, 5, 4, and 3 hidden units. These experiments showed that the BPN in the problem of concern was, in fact, relatively insensitive to the number of hidden units, provided that not less than five units was used. If the number of hidden units used were less than five, the BPN had serious problems with convergence, and if it did converge, it ended in a highly non-optimal local minimum. In general, the utilization of less hidden units forces the BPN into better generalization as it makes it more difficult for the BPN to memorize the samples included in the training set. Of course,

the lower number of the hidden units also ensures a higher speed of processing the data with the trained BPN. Therefore, the optimal number of BPN hidden units for the test problem is probably five.

5.3.1.2. Data Preprocessing for the BPN

Processing of input signals in the BPN essentially consists in summation of their values (cf. section 4.2.1). Therefore, as discussed in detail in section 5.2.3.1, the elements of the input vectors and also the only element of the output vectors need to be preprocessed using the decadic logarithmic transformation.

The boundary configurations of the modeled NPP (i.e. the nominal configuration when all the NPP components/systems are available, and the configuration when all NPP components/systems, that can possibly fail, are really unavailable) define a possible range of the resulting core-damage frequencies (cf. section 5.2.2). For the test PSA model, these boundary configurations give the range from 1×10^{-6} to 1×10^{-4} , which after application of the decadic logarithmic transformation yield the range from -6 to -4. Yet, the sigmoid activation function used in the BPN output unit can directly produce the outputs only in the range from zero to one, and in practice it is better to limit the range of the

target outputs even further into the interval from 0.1 to 0.9 only*.

Thus, the values of elements of the output vectors actually used in the BPN training, with the original range of 1×10^{-6} thru 1×10^4 , were first transformed with the decadic logarithmic transformation to the range of -6 thru -4, and then re-scaled to the more suitable range 0.1 thru 0.9 by using an additional linear transformation [54].

5.3.1.3. BPN Training

Identification of the procedure allowing successful training of the BPN in practical applications is a difficult task typically requiring a lot of experimentation. There is no universal procedure to apply in the training. Instead, there are many different techniques that can be utilized and

* Use of the output values reduced to the range of 0.1 thru 0.9 helps the BPN to learn faster. The reason is, that due to the nature of the sigmoid function, to achieve even a small change of the output value of the processing unit close to zero or one, a large change of the signals coming to this unit is necessary. Nevertheless, the fact that the true boundary values of outputs are actually well known (because no configuration can possibly result in the core-damage frequency out of the range limited by the boundary configurations) seems to suggest the scaling of the output vector values directly to the range of zero thru one. It would give an advantage that the BPN itself would not be inherently capable to produce unrealistic core-damage frequencies. Unfortunately, the core-damage frequency values very close to the boundary frequencies are quite common and the BPN trained with the output range from zero to one would thus have real difficulty learning the problem. Therefore, it is much better to use the "BPN-friendly" range of 0.1 to 0.9. The squashing of the values produced by the trained BPN to the correct range can be still easily achieved by using a simple if-then rule.

identification of those which are the most useful in a particular application can save a lot of time and effort and frequently even decide between the success or failure of the training.

As was already mentioned in section 5.2.2, the BPN training for the problem of concern started with the initial training set comprising the input-output vector pairs characterizing the boundary configurations (i.e. the nominal configuration, and the configuration with all basic events considered guaranteed) and the individual configurations with always just one out of all basic events considered guaranteed. In case of the training PSA model, its 14 basic events with variable probabilities gave 14 different configurations with only one of the basic events considered guaranteed. Together with the boundary configurations, the initial training set for the test model thus comprised 16 basic input-output vector pairs. As expected, the BPN trained with this basic vector pairs was unable to produce correct outputs for many configurations with more than one basic event considered guaranteed. Therefore, various additional input-output vectors pairs were gradually added to the training set in an effort to achieve better overall BPN performance. Selection of these additional samples was done with the aim of improving BPN performance for the configurations for which the BPN produced completely incorrect outputs when trained with the previous training

sets. For the test problem, the final training set, which yielded somewhat satisfactory results for majority of the possible configurations, comprised 119 input-output vector pairs. The configurations included in this training set, together with their correct core-damage frequencies, are given in Table 2 (the events considered guaranteed in the individual configurations are identified by their serial numbers assigned to them in Table 1).

An important challenge for the BPN in the problem of interest appears to be a recognition of saturation of the increase in the core-damage frequency when certain key model basic events are considered guaranteed. The general reason for this saturation is that many of the basic events in the PSA models represent, in fact, alternative causes of failure of the same NPP systems. Thus, all the NPP systems provided for mitigation of the individual initiating events can be effectively unavailable due to failures of certain NPP key components without any necessity for really all the basic events in the PSA model of this NPP being considered guaranteed. The trained BPN, however, tends to increase its estimate of the core-damage frequency always when another basic event of the model is considered guaranteed in addition to those considered guaranteed in the previous configuration. As the result, the trained BPN typically produces unrealistically high estimates of the core-damage frequency

Table 2 - Configurations included in the training set

No.	Events Guaranteed	Frequency
1		1.0×10^{-6}
2	7	1.0×10^{-6}
3	1	1.1×10^{-6}
4	5	1.0×10^{-5}
5	3	1.1×10^{-6}
6	11	1.0×10^{-6}
7	5, 9	1.0×10^{-5}
8	4, 5	1.1×10^{-5}
9	5, 11	1.0×10^{-5}
10	3, 9	1.1×10^{-6}
11	5, 9, 11	1.0×10^{-5}
12	4, 5, 11	1.1×10^{-5}
13	4, 5, 9, 11	1.1×10^{-5}
14	4, 5, 9	1.1×10^{-5}
15	11, 12	1.0×10^{-6}
16	7, 8	1.0×10^{-5}
17	1, 2	1.2×10^{-6}
18	3, 4	1.2×10^{-6}
19	11, 12, 13	1.2×10^{-6}
20	1, 2, 3	1.1×10^{-5}
21	1, 3, 4	1.1×10^{-5}
22	3, 4, 11, 12	1.4×10^{-6}
23	3, 4, 11, 12, 13	1.0×10^{-4}
24	8	1.0×10^{-6}
25	9	1.0×10^{-6}
26	10	1.0×10^{-6}
27	2	1.1×10^{-6}
28	4	1.1×10^{-6}
29	12	1.0×10^{-6}
30	13	1.0×10^{-6}
31	14	1.0×10^{-6}
32	6	1.0×10^{-5}
33	6, 7	1.0×10^{-5}
34	6, 8	1.0×10^{-5}

Table 2 continued

No.	Events Guaranteed	Frequency
35	3, 6	1.1×10^{-5}
36	5, 12	1.0×10^{-5}
37	5, 13	1.0×10^{-5}
38	5, 14	1.0×10^{-5}
39	6, 11	1.0×10^{-5}
40	6, 12	1.0×10^{-5}
41	6, 13	1.0×10^{-5}
42	6, 14	1.0×10^{-5}
43	3, 10	1.1×10^{-6}
44	4, 7	1.1×10^{-6}
45	4, 8	1.1×10^{-6}
46	5, 9, 12	1.0×10^{-5}
47	5, 9, 13	1.0×10^{-5}
48	5, 9, 14	1.0×10^{-5}
49	5, 10, 11	1.0×10^{-5}
50	5, 10, 12	1.0×10^{-5}
51	5, 10, 13	1.0×10^{-5}
52	5, 10, 14	1.0×10^{-5}
53	6, 7, 11	1.0×10^{-5}
54	6, 7, 12	1.0×10^{-5}
55	6, 7, 13	1.0×10^{-5}
56	6, 7, 14	1.0×10^{-5}
57	6, 8, 11	1.0×10^{-5}
58	6, 8, 12	1.0×10^{-5}
59	6, 8, 13	1.0×10^{-5}
60	6, 8, 14	1.0×10^{-5}
61	4, 5, 12	1.1×10^{-5}
62	4, 5, 13	1.1×10^{-5}
63	4, 5, 14	1.1×10^{-5}
64	3, 6, 11	1.1×10^{-5}
65	3, 6, 12	1.1×10^{-5}
66	3, 6, 13	1.1×10^{-5}
67	3, 6, 14	1.1×10^{-5}
68	4, 5, 9, 12	1.1×10^{-5}

Table 2 continued

No.	Events Guaranteed	Frequency
69	4, 5, 9, 13	1.1×10^{-5}
70	4, 5, 9, 14	1.1×10^{-5}
71	4, 5, 10, 11	1.1×10^{-5}
72	4, 5, 10, 12	1.1×10^{-5}
73	4, 5, 10, 13	1.1×10^{-5}
74	4, 5, 10, 14	1.1×10^{-5}
75	3, 6, 7, 11	1.1×10^{-5}
76	3, 6, 7, 12	1.1×10^{-5}
77	3, 6, 7, 13	1.1×10^{-5}
78	3, 6, 7, 14	1.1×10^{-5}
79	3, 6, 8, 11	1.1×10^{-5}
80	3, 6, 8, 12	1.1×10^{-5}
81	3, 6, 8, 13	1.1×10^{-5}
82	3, 6, 8, 14	1.1×10^{-5}
83	4, 5, 10	1.1×10^{-5}
84	3, 6, 7	1.1×10^{-5}
85	3, 6, 8	1.0×10^{-5}
86	11, 13	1.0×10^{-6}
87	11, 14	1.0×10^{-6}
88	12, 13	1.0×10^{-6}
89	12, 14	1.0×10^{-6}
90	13, 14	1.0×10^{-6}
91	9, 10	1.0×10^{-5}
92	11, 13, 14	1.2×10^{-6}
93	12, 13, 14	1.2×10^{-6}
94	1, 2, 4	1.1×10^{-5}
95	2, 3, 4	1.1×10^{-5}
96	3, 4, 11, 13	1.4×10^{-6}
97	3, 4, 11, 14	1.4×10^{-6}
98	3, 4, 12, 13	1.4×10^{-6}
99	3, 4, 12, 14	1.4×10^{-6}
100	3, 4, 13, 14	1.4×10^{-6}
101	3, 4, 11, 13, 14	1.0×10^{-4}
102	3, 4, 11, 12, 14	1.0×10^{-4}

Table 2 continued

No.	Events Guaranteed	Frequency
103	3, 4, 12, 13, 14	1.0×10^{-4}
104	5, 6	1.0×10^{-4}
105	5, 6, 11	1.0×10^{-4}
106	5, 6, 12	1.0×10^{-4}
107	5, 6, 13	1.0×10^{-4}
108	5, 6, 14	1.0×10^{-4}
109	5, 6, 11, 12	1.0×10^{-4}
110	5, 6, 11, 13	1.0×10^{-4}
111	5, 6, 11, 14	1.0×10^{-4}
112	5, 6, 11, 12, 13, 14	1.0×10^{-4}
113	7, 8, 9, 10	1.0×10^{-4}
114	7, 8, 9, 10, 11	1.0×10^{-4}
115	7, 8, 9, 10, 12	1.0×10^{-4}
116	7, 8, 9, 10, 13	1.0×10^{-4}
117	7, 8, 9, 10, 14	1.0×10^{-4}
118	7, 8, 9, 10, 11, 12, 13, 14	1.0×10^{-4}
119	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14	1.0×10^{-4}

for the configurations with many basic event considered guaranteed.

Nevertheless, a detailed exploration conducted while using the test PSA model showed that, for the problem of interest, the BPN was rather sensitive to overtraining, and when its training was interrupted in time, a much better overall performance of the BPN could be obtained. To identify the point when the overtraining effect starts, the cross-validation method was used [54]. This method is normally based on splitting the available input-output vector pairs into two separate sets: the training set, and the testing set. Only the training set is then used for the BPN training, and this training is occasionally interrupted and the performance of the current BPN is evaluated using the testing set. While the overall error of the BPN on the training set is typically steadily decreasing (provided that suitable training parameters are used), the error on the testing set ceases to decrease after certain number of training epochs and often even starts to increase. The point where this change of trend of the BPN error on the testing set occurs indicates the point when the BPN is already fully trained and any further training is useless or even damaging to its performance. In the problem of concern, however, there is an advantage of directly knowing what is one of the effects of the overtraining: it is the overestimation of the core-damage frequencies for the

configurations with certain combinations of the basic events considered guaranteed. This knowledge prompts the preparation of a special testing set addressing specifically this effect, i.e. the set comprising some selected input-output vector pairs representing the NPP configurations with many basic events considered guaranteed.

The testing set prepared in this way for the test PSA model comprised 16 configurations listed in Table 3 (the events considered guaranteed in the individual configurations are identified by their serial numbers assigned to them in Table 1). All the configurations included in this set represent the situations when all systems provided for mitigation of the considered initiating event are effectively disabled by multiple failures. In addition, almost all these configurations include far more basic events considered guaranteed than it is really necessary for the complete failure of the initiating event mitigation. For example, while the configuration number one in Table 3 represents a real minimal cut set for the initiating event (i.e., all the basic events considered guaranteed for this configuration must really happen if the mitigation of the initiating event is to be totally disabled; if any of these events is removed, the resulting configuration does not lead to the core damage), the configurations number four, eight, ten, and twelve include some excessive guaranteed events which actually do not make

Table 3 - Configurations included in the testing set

No	Events Guaranteed	Frequency
1	1, 2, 3, 4	1.0×10^{-4}
2	1, 2, 5, 6	1.0×10^{-4}
3	3, 4, 5, 6	1.0×10^{-4}
4	1, 2, 3, 4, 5, 6	1.0×10^{-4}
5	1, 2, 7, 8, 9, 10	1.0×10^{-4}
6	3, 4, 7, 8, 9, 10	1.0×10^{-4}
7	5, 6, 7, 8, 9, 10	1.0×10^{-4}
8	1, 2, 3, 4, 7, 8, 9, 10	1.0×10^{-4}
9	3, 4, 11, 12, 13, 14	1.0×10^{-4}
10	1, 2, 3, 4, 11, 12, 13, 14	1.0×10^{-4}
11	7, 8, 9, 10, 11, 12, 13, 14	1.0×10^{-4}
12	1, 2, 3, 4, 7, 8, 9, 10, 11, 12, 13, 14	1.0×10^{-4}
13	3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14	1.0×10^{-4}
14	3, 4, 7, 8, 9, 10, 11, 12, 13, 14	1.0×10^{-4}
15	1, 2, 5, 6, 11, 12, 13, 14	1.0×10^{-4}
16	1, 2, 7, 8, 9, 10, 11, 12, 13, 14	1.0×10^{-4}

any difference in the outcome of the accident. Naturally, with all the mitigating features unavailable the core-damage frequencies for these configurations are directly equal to the frequency of the initiating event itself, as also shown in Table 3.

The effect of overtraining is illustrated in Figure 12 by learning curves of the best performing BPN trained for the test problem (RMS error stands for the root-mean-squares error). In this case, the overtraining actually did not cause the typical increase of the BPN output error on the testing set, but manifested itself only by a virtual halt in the decrease of this error after about 250 thousand training epochs (i.e. the presentations of all vector pairs in the training set)*. This absence of the increase of the output error on the testing set was, however, observed only for the trained BPNs achieving the best final performance. For the trained BPNs showing somewhat less successful final performance, this increase did occur. Thus, the course of the learning curve for the testing set could also serve as an indicator of the BPN convergence to the global minimum.

* Unfortunately, the scale which had to be used in Figure 12 did not allow to show a continued but very small decrease of the error on the training set, and a slight oscillation of the error on the testing set around a certain "steady" value in the final period of the training. Also, the training was actually allowed to continue for about a million further training epochs but no further changes in the courses of the learning curves were observed.

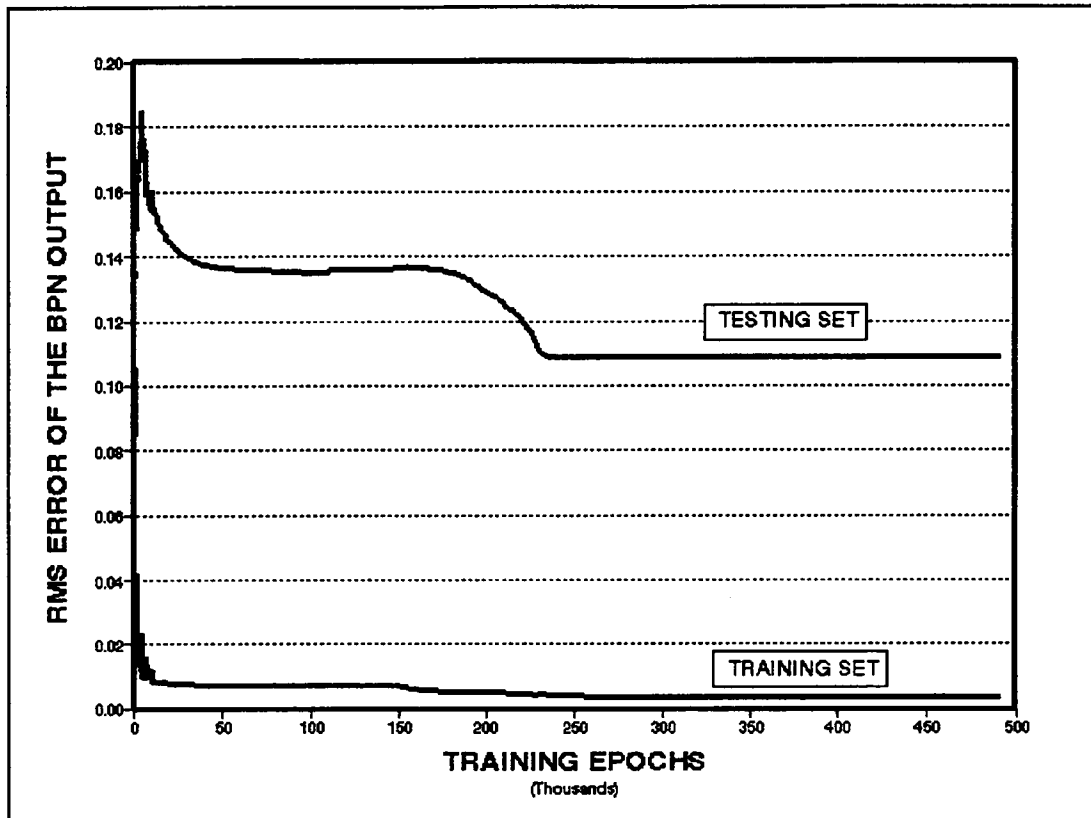


Figure 12 - Learning curves of the BPN

Experiments with different training techniques and parameters on the test problem had shown that, for the problem of interest, the surface of the BPN output error on the training set is very complex. It seems to include several "nearly-global" minima, rather deep and narrow, which (together with the global minimum, of course) all represent BPNs with a relatively good performance. The rest of the error surface is then probably dotted with many local minima which, although not really shallow, still represent the BPNs with a rather poor overall performance due to their

insufficient generalization. A successful convergence of the BPN to the global minimum (or at least one of the "nearly-global" minima) thus requires using a batch update of the BPN weights (i.e. their update with the combined updating term obtained when presenting the BPN with all vector pairs in the training set one-by-one), the fixed-step method of updating (i.e. the updating term is determined only according to the current error of the BPN outputs, without any regard to the update applied in the previous training epoch), and a relatively low value of the learning rate (i.e. the coefficient regulating magnitude of the updating term) of 0.2 or lower. For the test problem, this approach ensured a rather steady convergence of the training, but required training for a relatively large number of the training epochs, reaching to hundreds of thousands.

However, when using these training techniques and parameters from the very beginning of training, the abundance of the local minima on the error surface practically always causes the BPN to converge into a local minimum. Therefore, it is necessary to start the training using some very different techniques and parameters allowing the BPN to search the error surface and roughly locate the global or "nearly-global" minimum. This initial training of the randomized BPN requires using an on-line update of the BPN weights (i.e. their continuous update with updating term obtained from

presentation of any single vector pair from the training set to the BPN), the momentum method of updating (i.e. the updating term reflects the combined effect of both the current error of the BPN outputs and the update applied in the previous training epoch), and a higher value of the learning rate of about 0.5. This approach encourages rather violent changes of the BPN weight values (resembling to some extent a start of the simulated annealing process) and effectively performs a rough search of the error surface. Nevertheless, when this approach is applied to the full training set, the high number of vector pairs in the set often results in reaching huge values of some BPN weights which are far away from their desired final state (and sometimes even causes a crash of the training program). To avoid this, it is better to use for the initial training a different training set containing fewer input-output vector pairs thus limiting the search into a more reasonable area. The training set used for initial training of the BPN for the test problem comprised only ten vector pairs which actually represented simply the first ten pairs of the full training set used for the regular training as given in Table 2.

In order to determine the position of the global or "nearly-global" minimum on the error surface, the performance of the current BPN has to be tested frequently with the testing set comprised of as many input-output vector pairs as

possible. The initial testing set used for the test problem included both the full training set and the testing set prepared to be used in the actual BPN training (given in Tables 2 and 3). For the test problem, the BPN reached a good position for the full training usually within less than 50 training epochs. Unfortunately, no technique was found capable of distinguishing, during the initial pre-training or early on during the full training, if the BPN was converging into the truly global or only a "nearly-global" minimum.

There is one more problem to be addressed. The full training set consists of many input-output vector pairs representing various NPP configurations between the two boundary configurations, i.e. the nominal configuration with all basic event probabilities set to their nominal values given in Table 1, and the configuration with all basic events considered guaranteed with their probabilities set to unity. Thus, when the limiting configurations themselves are presented to the BPN only once in each training epoch, their influence is overcome by the great prevalence of these intermediate configurations. As a result, the BPN incorrectly extrapolates the significance of changes in its inputs and widens the spread of possible values of its output. The boundary configurations are thus typically evaluated incorrectly. However, since especially the lower boundary configuration is very important, it is desirable to train the

BPN to provide correct result for it. In order to ensure this, it is necessary to reinforce training of this configuration by its repeated occurrence in the training set. For the test problem this reinforcement was accomplished simply by the addition of two identical additional vector pairs representing this configuration at the end of the training set. The same problem exists also for the upper boundary configuration, but since its influence is much less significant no effort was made to correct it for the test problem.

The best performing BPN trained for this problem had five hidden units and was trained using the training procedure described above. Its training required 14 epochs of the initial pre-training with the reduced training set, and 251,000 epochs of the regular training with the full training set.

There was no attempt made at this stage of the research to train the BPN with the training set extended with the configurations considering impossibility of individual basic events of the test PSA model; such a BPN would be necessary for computation of the RRWs. It can be expected that the same problems (and likely even more severe) would be encountered in this training. Of course, in case that the BPN would be successfully trained to handle the test problem for the

configurations with the guaranteed events only, ability of the BPN to handle the extended variety of possible inputs would, also, have to be proved.

5.3.1.4. Verification and Validation of the Trained BPN

In general, the verification represents the checking if a software product works as expected by its developer, and the validation represents the checking if the product performs as required by its user (a detailed discussion is given in section 3.4 of this study). Specifically for the BPN (and any other ANN paradigm as well), it means that the verification is the checking of the trained BPN performance when processing the input vectors used for its training, and the validation is the checking of the trained BPN performance when processing the input vectors which have not been used for the training. The verification and validation are an important part of the training process as they allow identification of the configurations which the BPN so far failed to learn. Thus, they serve as a feedback of the training process helping to identify the input-output vector pairs which should be added to the training set. Actually, the whole procedure for successful training of the BPN for the test problem, described in the previous section 5.3.1.3, was developed as a result of an iterative process aimed at elimination of incorrect mappings identified by the verification and validation of the

trained BPN, and finding of the procedure ensuring the best training results. Also, the cross-validation method used in the training is based on an occasional validation of the BPN current status.

To be able to verify and validate the BPN performance, it is necessary to set some criteria of acceptability of the results obtained. These criteria are, naturally, very much application-specific and are usually set by the potential user of the application. For utilization of the BPN in the risk monitor, a main criterion could, for example, define a limit of the percentile error of the calculated core-damage frequency (10% error seems to be a reasonable value of this limit). Additional criteria could require a correct computation and ranking of the risk control and risk reduction importances of the model basic events (and groups of these events), at least on a certain number of top positions. Besides that, practical considerations also require that the BPN be able to achieve an acceptable performance while requiring a training set containing a reasonable number of the input-output vector pairs.

The success of BPN training for the test problem were, first of all, verified with the whole training set used for the training. Resulting absolute errors of the actual outputs of the best performing trained BPN (i.e. of the outputs as

produced initially before their final re-scaling and inverse transformation) for the input vectors in the training set are shown in Figure 13. However, it is rather difficult to make any judgment about the BPN performance on the basis of these direct output values.

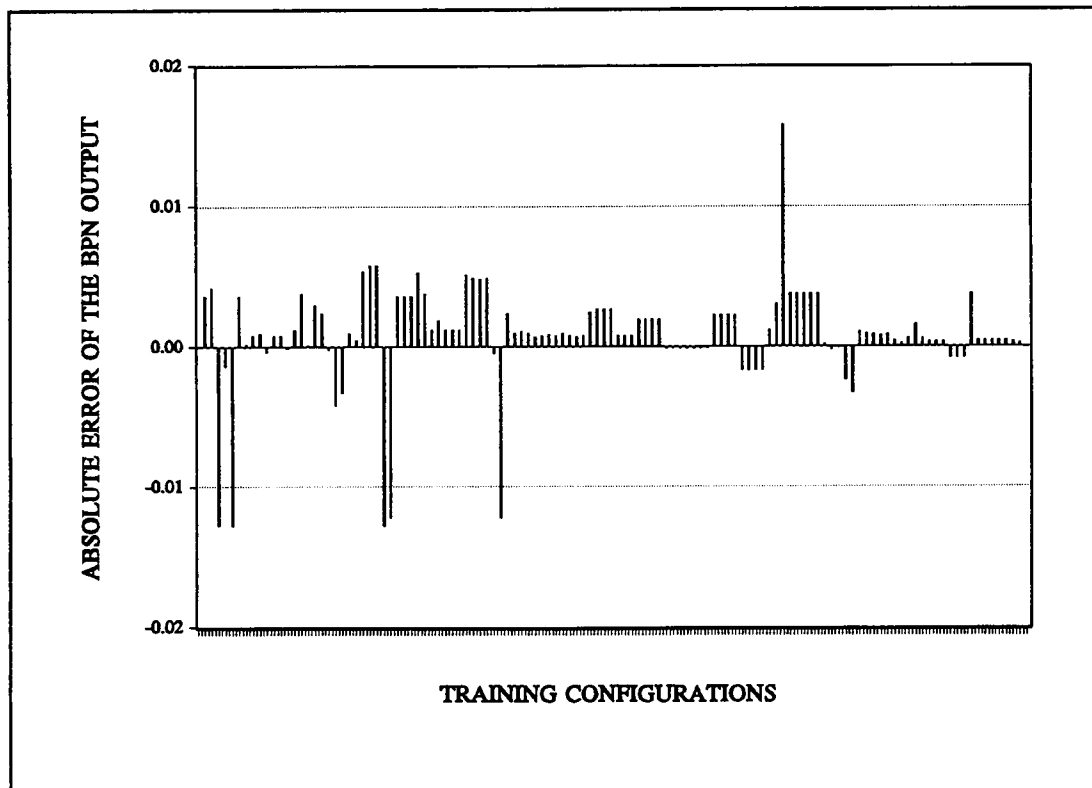


Figure 13 - Absolute errors of the BPN actual output values on the training set

A clear picture of the BPN performance is obtained only when the BPN actual outputs are re-scaled and transformed back to the values of the core-damage frequency. The percentile

errors of the resulting core-damage frequency values (i.e. the relative errors of their computed values in percents of the desired target values) computed by the best performing BPN for the configurations included in the training set are shown in the top graph in Figure 14. The bottom graph in Figure 14 then shows the percentile errors of the core-damage frequency after applying a procedure reducing the error. The reduction of the error is based on the knowledge that the resulting frequencies have to be strictly within the range limited by the frequencies of the boundary configurations (for the test problem, these limiting frequencies were 1×10^{-6} and 1×10^{-4} - cf. section 5.3.1.2). The error reduction procedure thus simply consists in replacement of the actually computed frequency values falling out of the correct range by the corresponding limiting frequency.

As it can be seen in Figure 14, errors of the computed frequencies for all configurations in the training set were well within the 10% tolerance. In fact, even the error value of 5% was exceeded only for the six configurations listed in Table 4. Interestingly, four of these configurations represent the configurations with only a single guaranteed event which indicates that the training of these configurations should be probably reinforced by using their multiple appearance in the training set. Nevertheless, absolute differences between the target and actually computed frequency values for all listed

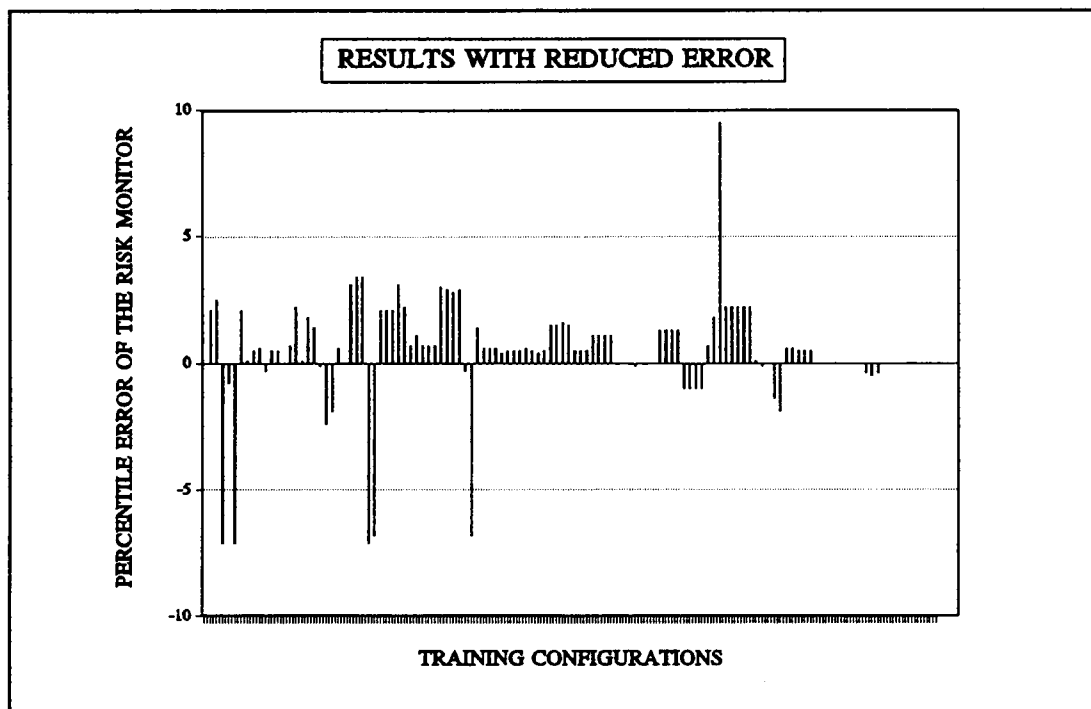
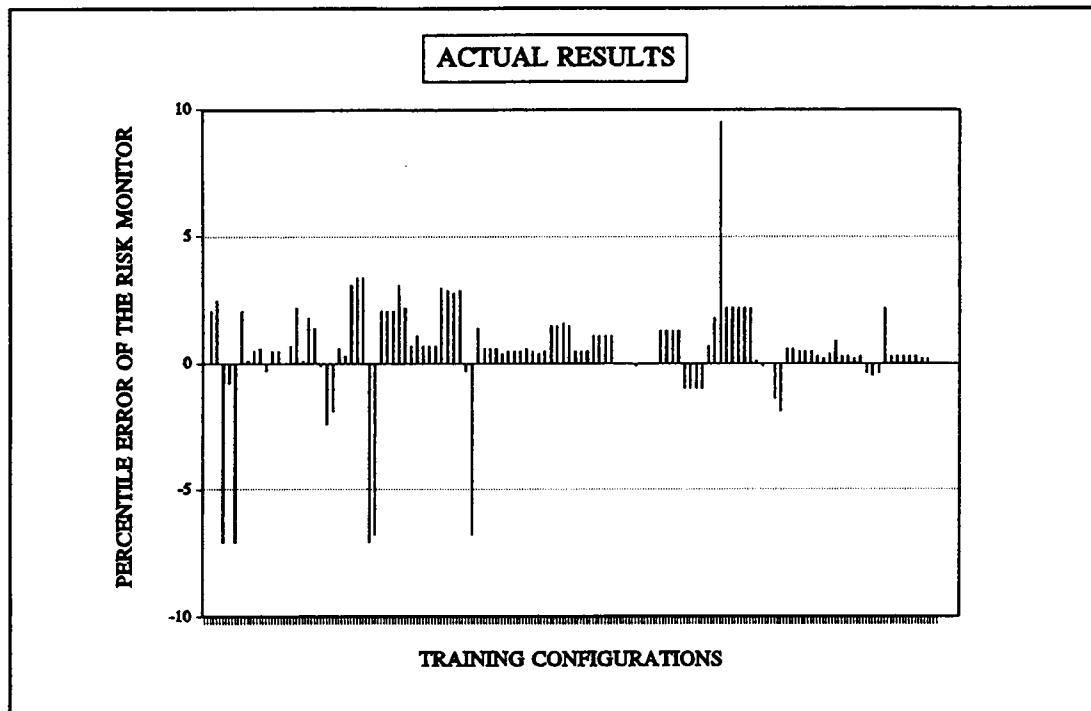


Figure 14 - Percentile errors of the core-damage frequency values computed from the BPN outputs for the training configurations

Table 4 - Training configurations with an error of the computed core-damage frequency exceeding 5%

Conf. No.	Events Guaranteed	Target Value	Computed Value	Error [%]
3	1	1.10×10^{-6}	1.02×10^{-6}	- 7.1
5	3	1.10×10^{-6}	1.02×10^{-6}	- 7.1
27	2	1.10×10^{-6}	1.02×10^{-6}	- 7.1
28	4	1.10×10^{-6}	1.02×10^{-6}	- 6.8
44	4, 7	1.10×10^{-6}	1.02×10^{-6}	- 6.8
85	3, 6, 8	1.00×10^{-5}	1.10×10^{-5}	+ 9.5

configurations is very low and clearly negligible with regards to the uncertainties normally present in the PSA. Thus it can be concluded that the BPN succeeded to learn the training set.

The success of BPN training can be, however, really judged only when the trained BPN is validated while processing some input vectors never used for its training. For the test problem, the best performing BPN was first validated with the testing set already utilized during the training. Resulting absolute errors of its actual outputs for the input vectors in the testing set are shown in Figure 15 and the percentile errors of the resulting core-damage frequency values computed for the testing configurations are shown in the top graph in

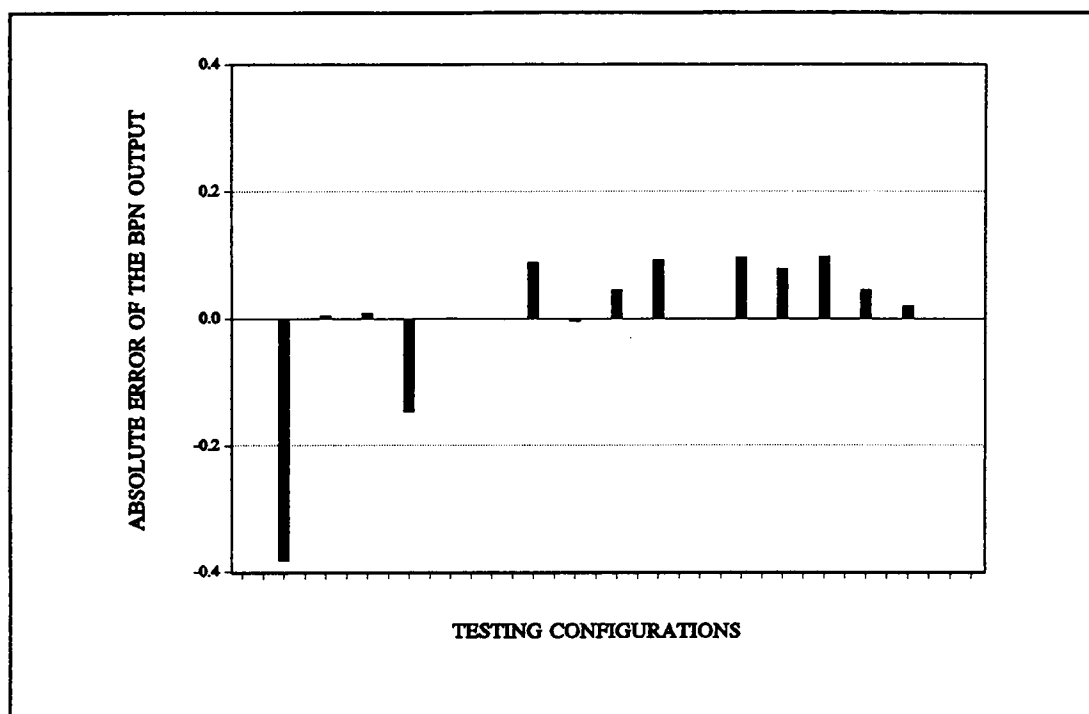


Figure 15 - Absolute errors of the BPN actual output values on the testing set

in Figure 16. The bottom graph in Figure 16 then shows the percentile errors of the computed core-damage frequency after applying the error reduction procedure.

As it is clear from Figure 16, the results for the configurations in the testing set were, not surprisingly, much poorer. Of the 16 configurations included in this set, the error of the computed frequencies exceeded the limit of 10% for as many as ten configurations (their list is given in Table 5). Considering the nature and purpose of these configurations (i.e. to serve only for the checking if the BPN

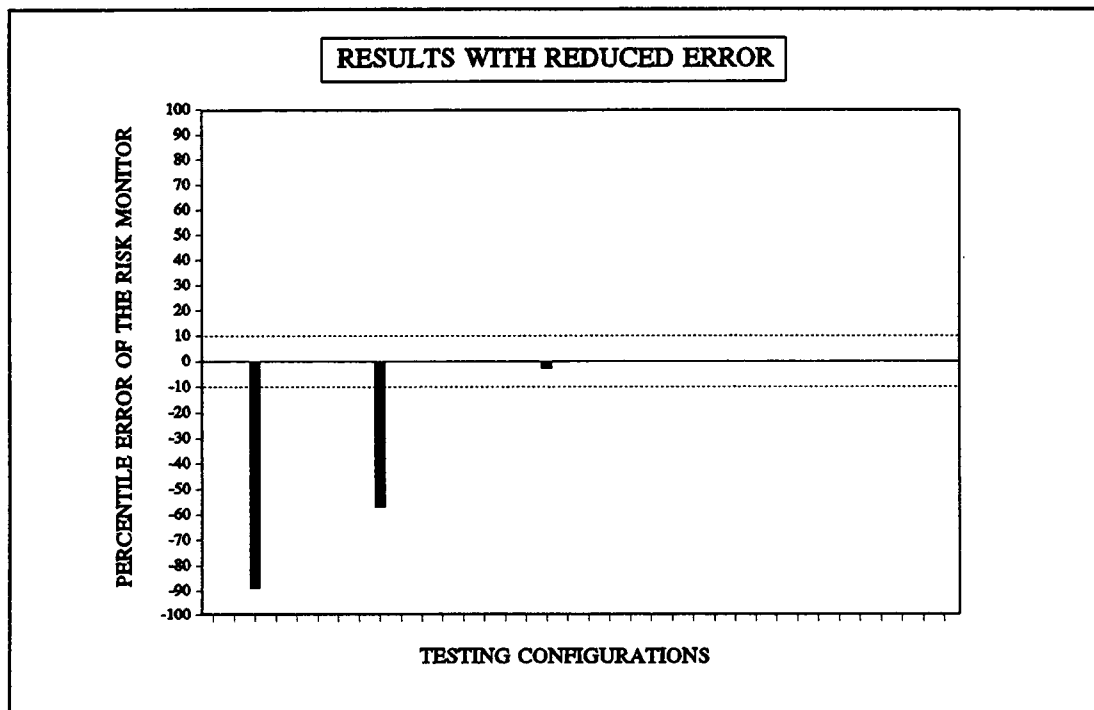
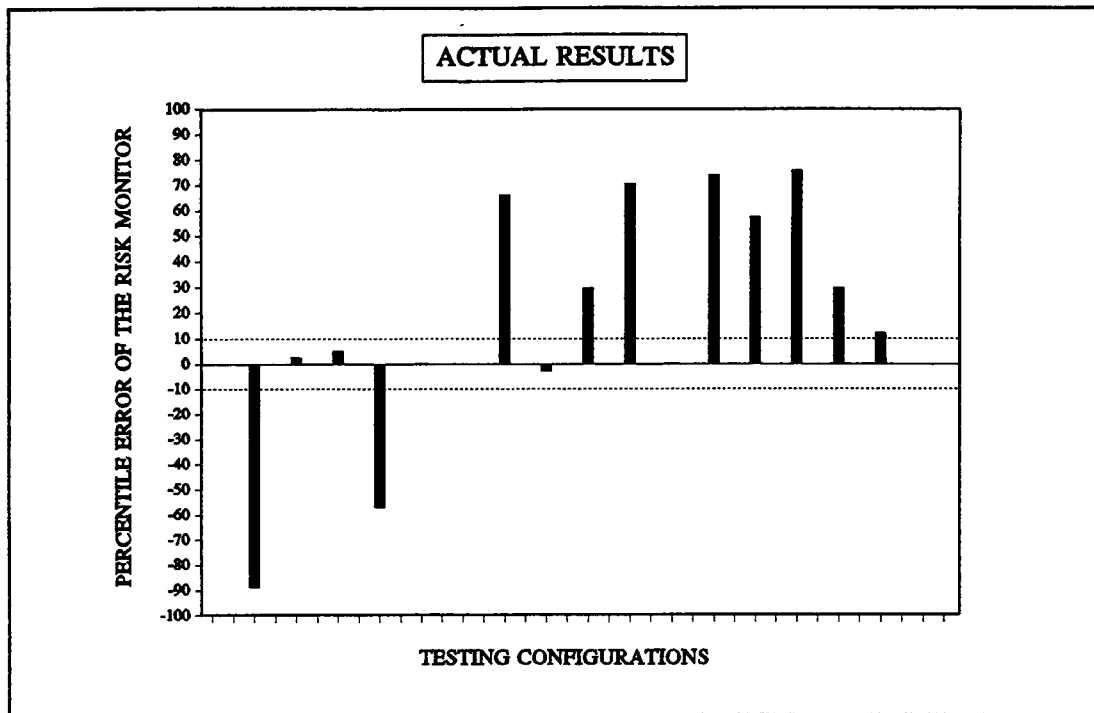


Figure 16 - Percentile errors of the core-damage frequency values computed from the BPN outputs for the testing configurations

Table 5 - Testing configurations with an error of the computed core-damage frequency exceeding 10%

C. No.	Events Guaranteed	Target Value	Computed Value	Error [%]
1	1, 2, 3, 4	1.00×10^{-4}	1.11×10^{-5}	-88.9
4	1, 2, 3, 4, 5, 6	1.10×10^{-4}	4.29×10^{-5}	-57.1
7	5, 6, 7, 8, 9, 10	1.00×10^{-4}	1.66×10^{-4}	+66.2
9	3, 4, 11, 12, 13, 14	1.00×10^{-4}	1.30×10^{-4}	+29.6
10	1, 2, 3, 4, 11, 12, 13, 14	1.00×10^{-4}	1.70×10^{-4}	+29.6
12	1, 2, 3, 4, 7, 8, 9, 10, 11, 12, 13, 14	1.00×10^{-4}	1.74×10^{-4}	+74.0
13	3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14	1.00×10^{-4}	1.57×10^{-4}	+57.3
14	3, 4, 7, 8, 9, 10, 11, 12, 13, 14	1.00×10^{-4}	1.76×10^{-4}	+75.7
15	1, 2, 5, 6, 11, 12, 13, 14	1.00×10^{-4}	1.30×10^{-4}	+29.8
16	1, 2, 7, 8, 9, 10, 11, 12, 13, 14	1.00×10^{-4}	1.12×10^{-4}	+12.3

learned to recognize saturation of its output value increase when the key basic events of the model are considered guaranteed), even a relatively high positive error in the computed frequency values can be tolerated. These configurations are virtually impossible to develop in practice (as the risk would reach unacceptable values much sooner), and the errors can be easily canceled out as shown in the bottom graph in Figure 16. In contrast, the two configurations whose core-damage frequencies were underestimated (one and four) represent real flaws in the performance of the trained BPN. The absolute errors for these configurations reached very high values (an order of magnitude for the configuration number one), too high even compared to the uncertainties inherent to the PSA. When training the BPN for utilization in a real risk monitor, these configurations would have to be transferred to the testing set so that the BPN could learn their correct mapping.

Obviously, the validation based only on the results for input vectors from the testing set cannot be considered sufficient for a well-founded judgement of the BPN performance, as these input vectors were used for cross-validation of the training process. In fact, the BPN considered "best performing" was identified precisely on the basis of its performance on this testing set. Hence, it is desirable to validate the trained BPN with some other input-

output vector pairs which are completely new to it. Specifically for the problem of concern, there is yet another option: validating performance of the BPN on computing the RAWs. The RAWs are quite crucial for the risk monitor operation and their values are sensitive to even small errors in the estimates of the core-damage frequencies. The best performing BPN trained for the test problem was validated on the computation of the RAWs of individual basic events for two configurations: the configuration with only the event number one guaranteed, and the configuration with the events number one and number four guaranteed (the events are identified by their serial numbers assigned to them in Table 1). The results of the BPN computation are shown in the top graphs in Figure 17 and Figure 18, respectively. The bottom graphs in Figures 17 and 18 represent the corresponding correct RAW values. From the dramatic differences of the corresponding graphs in both Figures, it is quite clear that the BPN performance in this task was rather poor. Especially for the configuration with the event number one guaranteed (see Figure 17), the BPN completely failed to recognize significance of the event number three, and highly underestimated significance of the event number five. For the configuration with the events number one and four guaranteed (see Figure 18), the BPN did recognize the significant events but highly overestimated significance of the event number five. These discrepancies identify the configurations which the BPN did not learn

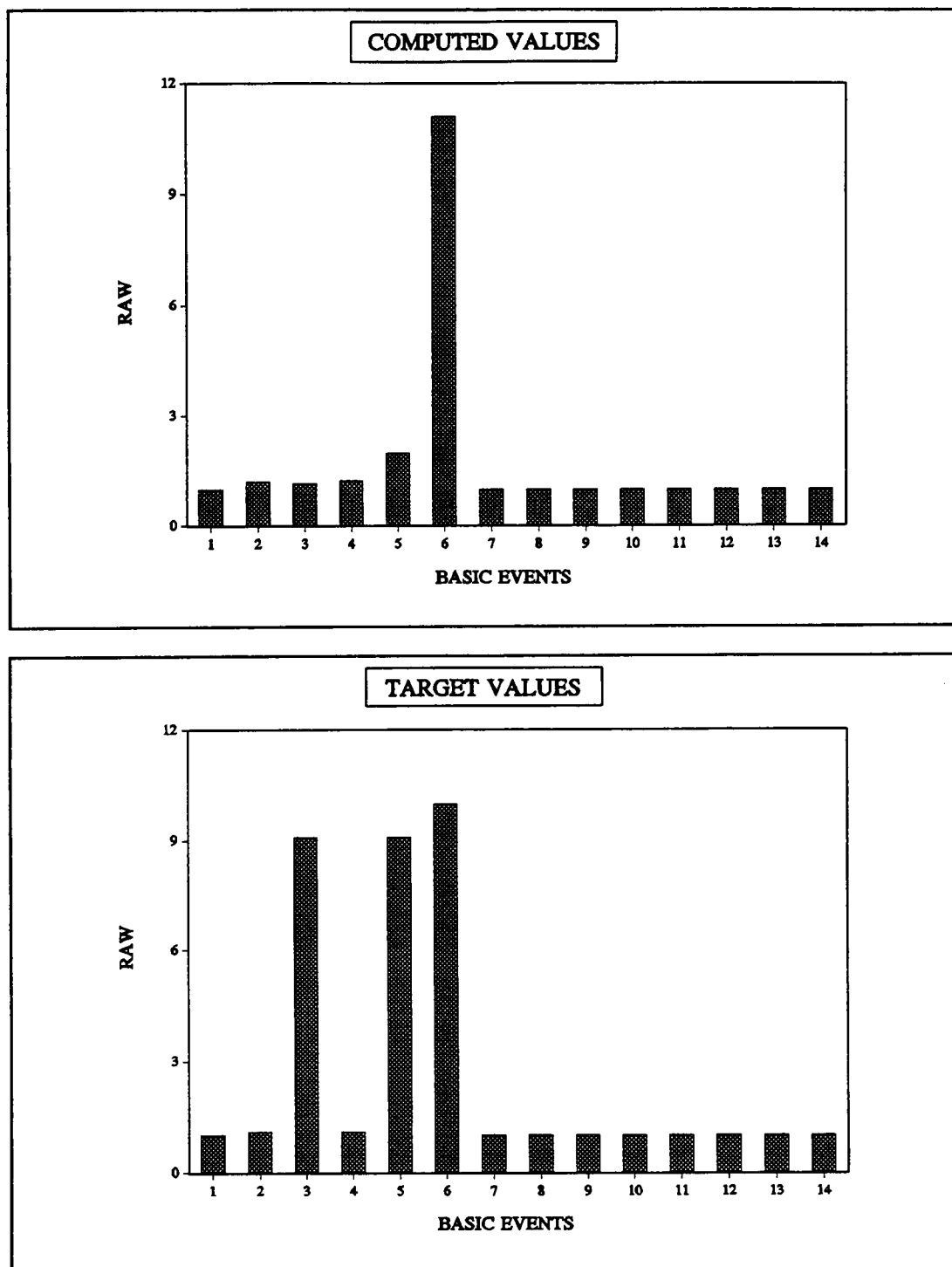


Figure 17 - RAWs of individual basic events as computed by the BPN for the configuration with only the event number one guaranteed

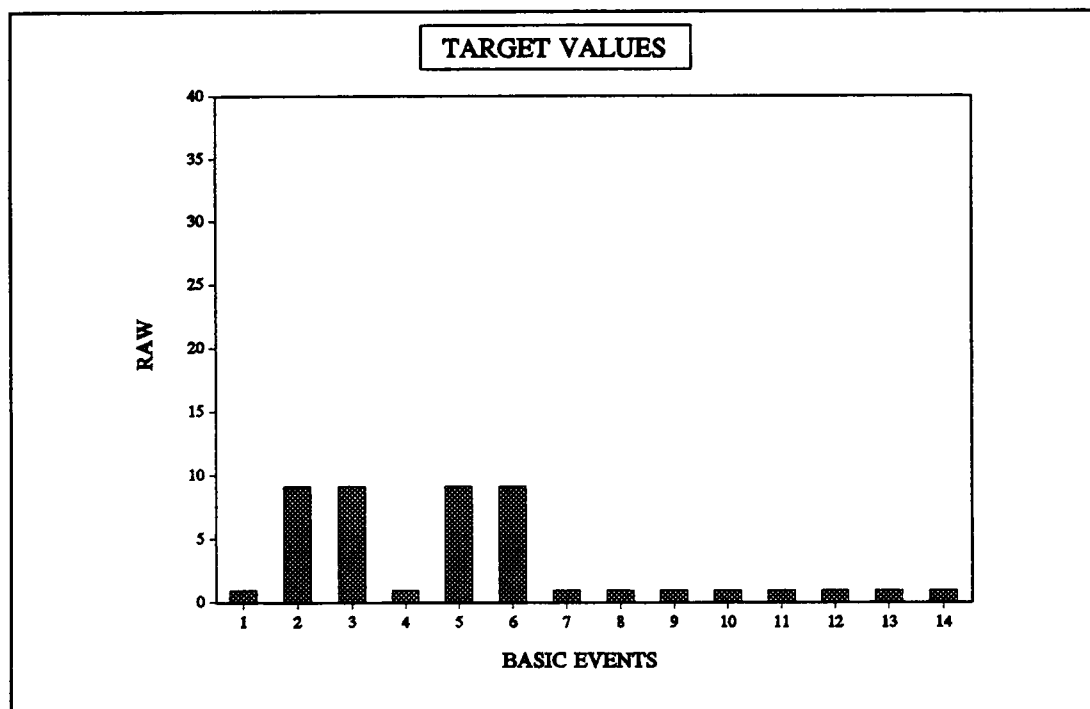
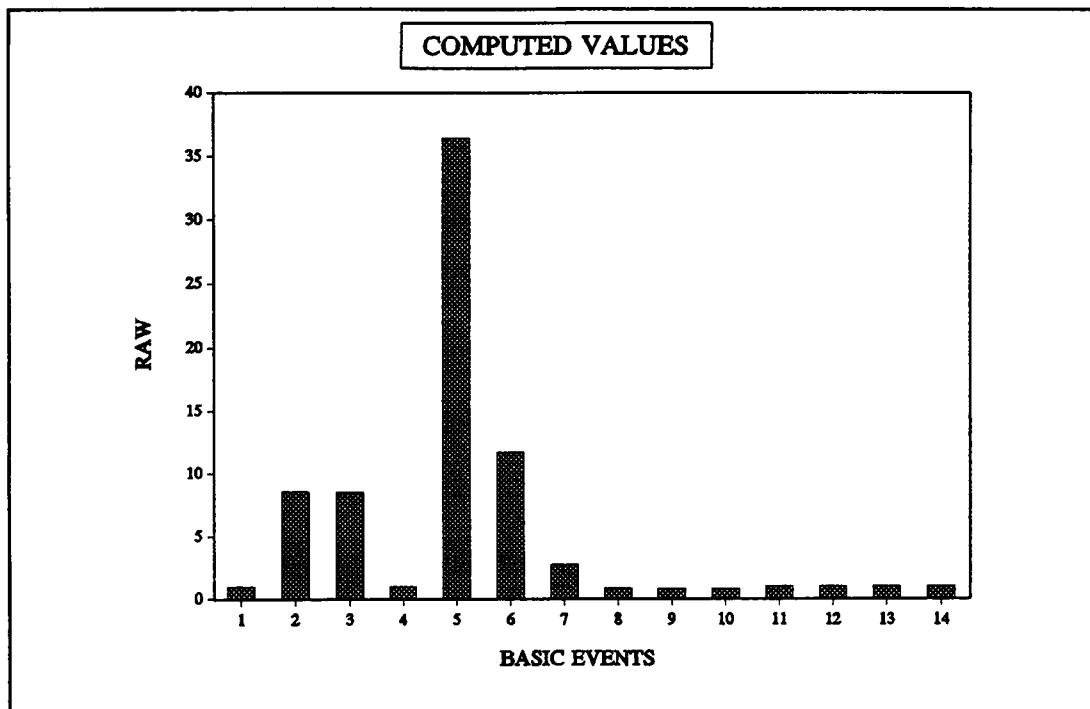


Figure 18 - RAWs of individual basic events as computed by the BPN for the configuration with the events number one and number four guaranteed

correctly so far; to improve performance of the BPN, these configurations would have to be added to the training set. Unfortunately, it is very likely that similar flaws as those identified for the investigated configurations would be found for many other configurations as well. Consequently, to train the BPN to reach an acceptable performance would require a huge training set which would be clearly beyond a practically manageable limit for any real risk monitor.

5.3.1.5. BPN Knowledge Analysis

Another step in the development of a BPN application can be the analysis of the way in which the BPN stores the knowledge acquired during training. Although this step is not really necessary, it is always useful to have some understanding of the structure of BPN knowledge. A good understanding of this structure can sometimes help in further training when some remaining flaws are to be removed.

In general, the BPN knowledge is stored in its weights. It is usually well distributed between all the weights and so it is rather rare to be able to observe any clear patterns [54,73]. The understanding of the BPN knowledge structure is especially difficult when the input vectors are continuously valued, as it is the case for the problem of concern.

The first natural step of the knowledge analysis of the trained BPN is a direct inspection of the values of its weights. The values of the weights in connections between input units and hidden units for the best performing BPN trained for the test problem are given in Table 6. There are some observations which can be based on these values. First of all, the fact that none of the weights is extremely large (as, e.g., a hundred or more) or very close to zero, shows that there are no excessive units in the trained BPN and all the weights have their place in representation of the problem. Another interesting observation is the high level of similarity between the weights in connections from the input units number 11, 12, 13, and 14 (in fact, they are precisely identical for three of these units). These input units represent four identical basic events with voting-logic connection, so this similarity seems to be logical. The influence of each of these events on the resulting core-damage frequency is exactly the same and it is thus only natural that the identical activation of the output unit is achieved by ensuring the identical activation of the hidden units. A very similar effect can be observed, to some extent, for the weights in connections from the input units number one and two, which represent identical events in different trains of the modeled NPP. Nevertheless, there is no such effect observable for the input units three and four, and especially not for five and six which represent the events with the same

Table 6 - Weights of the hidden units

Input Units	Hidden Units				
	1	2	3	4	5
bias	0.36	-1.96	-0.28	-0.59	0.02
1	1.87	0.73	-0.51	-0.51	3.27
2	1.51	0.27	-0.55	-0.55	3.27
3	3.28	2.18	0.85	-0.79	3.18
4	-1.34	0.14	-0.12	-0.79	3.36
5	-11.79	-4.01	-1.14	-0.41	2.06
6	-7.15	-1.8	-0.46	-0.52	16.11
7	1.31	0.24	1.63	3.03	0.74
8	-1.05	-0.53	1.47	3.08	-1.11
9	0.01	-0.7	2.67	0.86	-4.61
10	0.01	-0.55	2.67	0.92	-4.61
11	-0.05	1.55	-0.91	-0.29	-0.04
12	-0.05	1.55	-0.9	-0.26	-0.05
13	-0.05	1.55	-0.9	-0.26	-0.05
14	-0.05	1.55	-0.9	-0.26	-0.05

relation, but whose weights are quite dissimilar. Even more striking differences can be observed for the weights in connections from the input units seven and eight, and nine and ten, though these input units represent pairs of the identical redundant events in each of the two NPP trains. While the weights from the units seven and eight are completely different, the weights from the units nine and ten are very similar (and, in addition, there is no similarity at all between the weights of the two pairs). It can be concluded that, in some cases the BPN fully recognized similar influence of certain input units and reacted by assigning similar values to the weights from these inputs, while in other cases the similarities of the input units have not been projected into similarity of their weights. One possible reason for this phenomenon can be a rather random and therefore non-symmetrical distribution of the input-output vector pairs in the training set. Nevertheless, this observation does not say anything about the BPN success or failure; it only means that the BPN did not chose the way of knowledge representation logical to humans, and simply used its own different way which, by no means, is to be considered automatically incorrect.

The values of the weights in connections between the hidden units and the output unit are given in Table 7. There is an interesting observation which can be made from the values of

Table 7 - Weights of the output unit

Hidden Units	Output Unit
bias	0.07
1	-2.23
2	4.84
3	2.22
4	2.14
5	2.23

these weights. As only the weight from the hidden unit number one has a negative value, this unit would have to be strongly activated for all the configurations with the resulting core-damage frequency close to the lower limiting frequency (i.e. the frequency of the nominal configuration of 1×10^{-6}). To obtain higher frequency values required for many other configurations, the activation of this unit will have to be low, and/or the activations of the other hidden units will have to be rather high to compensate for its influence.

In any case, it is rather difficult to extract any really valuable information directly from the values of the BPN weights. The levels of activation of hidden units strongly depend on the specific input vectors presented to the BPN

input layer. In fact, the levels of activation of the different hidden units characterize individual features detected in the input vectors. The specific activations of the hidden units then cause corresponding activations of the output units producing the output of the BPN. Thus, to be able to understand the way in which the BPN recognizes individual features in the input vectors, it is necessary to analyze the levels of activation of hidden units for some selected input vectors. A proper selection of these input vectors is crucial for obtaining any meaningful information. The activations of hidden units of the best performing trained BPN when processing the input vectors representing a variety of different configurations are shown in the following graphs. The graphs the observations made when inspecting values of the BPN weights described above.

The most natural choices for the analysis seem to be the input vectors representing the boundary configurations. The activations of the BPN hidden units for the nominal configuration, and the configuration with all basic events guaranteed are shown in Figure 19 and Figure 20, respectively. As it can be seen in Figure 19, the expected strong activation of the hidden unit number one for the nominal configuration, which yields the lowest possible resulting core-damage frequency, really showed up; this unit was, actually, the only one activated at all for this configuration. This pattern of

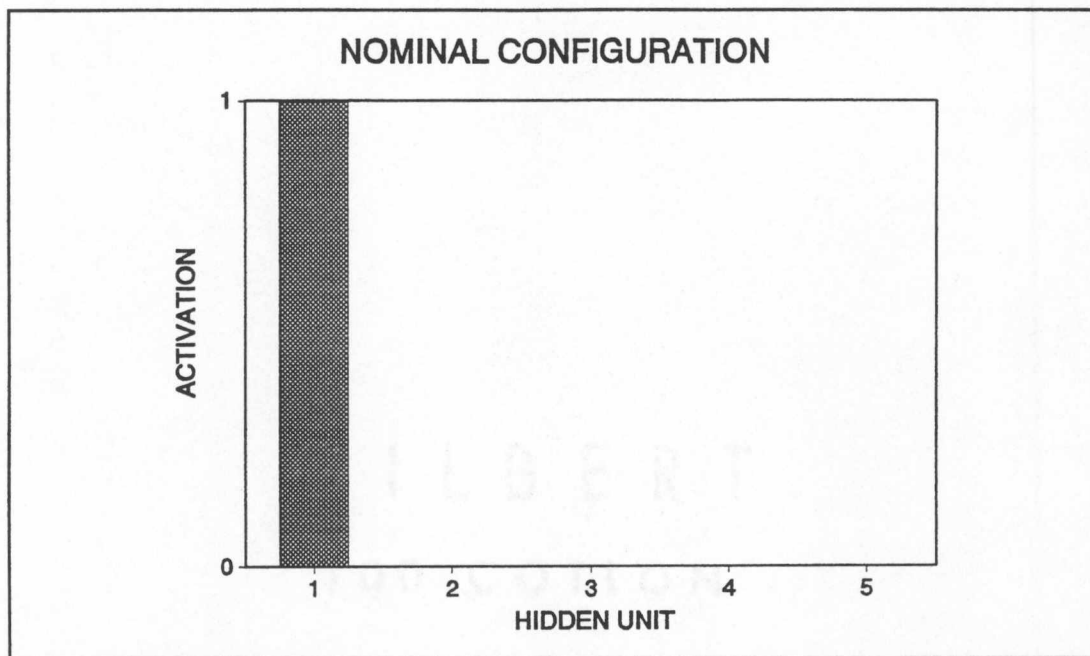


Figure 19 - Activations of hidden units for the nominal configuration

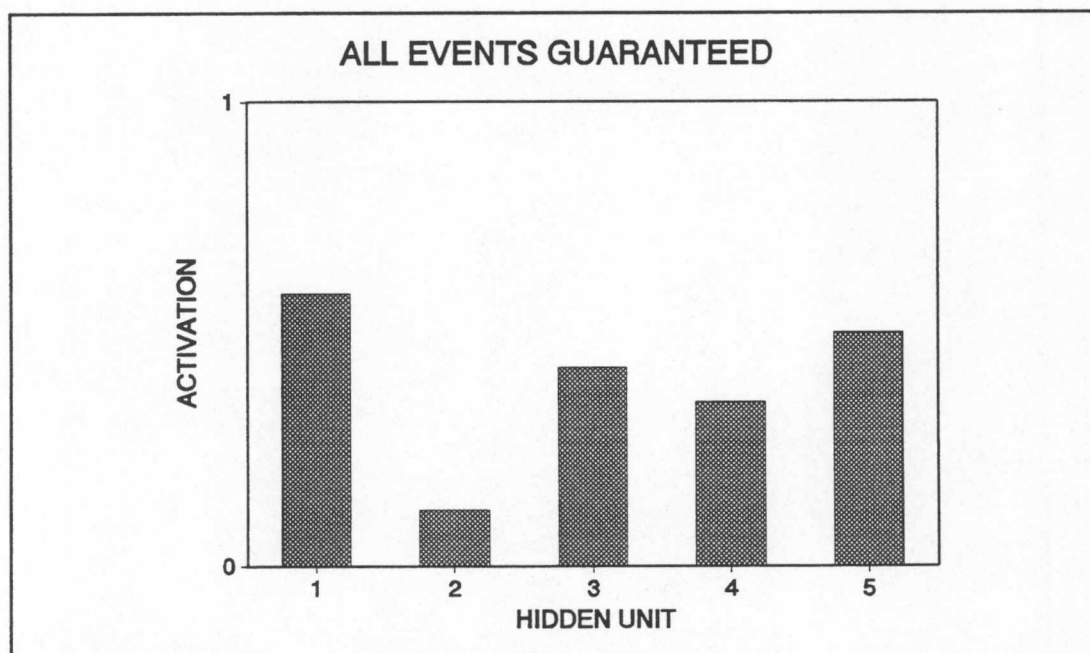


Figure 20 - Activations of hidden units for the configuration with all basic events guaranteed

hidden units activations is a classical case of the so-called "grandmother cell", when one unit is fully activated (i.e. its activation is almost one) and the other units are not activated at all [74]. The way how the BPN produced higher values of resulting frequencies is demonstrated in Figure 20. It was achieved by decreasing the activation of the hidden unit number one, and by increasing the activation of the other hidden units. The hidden unit number one was thus the only one detecting the specific features in the input vectors which resulted in the output frequency of 1×10^{-6} , and the other units were activated only when the output frequency was supposed to be higher. Such a transparent way of feature detection by the BPN is rather rare in practice and so it deserves further exploration.

The activations of the individual hidden units for some other configurations (all of them are the configurations with only one of the basic events guaranteed) yielding the core-damage frequencies close to the value of 1×10^{-6} are shown in Figure 21. As can be seen, the "grandmother-cell" effect, discussed before, held almost perfectly for all these configurations. The next Figure 22 shows some configurations (including the two remaining configurations with only one of the basic events guaranteed) yielding the core-damage frequencies close to the value of 1×10^{-5} . The last Figure 23 of this series shows some configurations yielding the core-

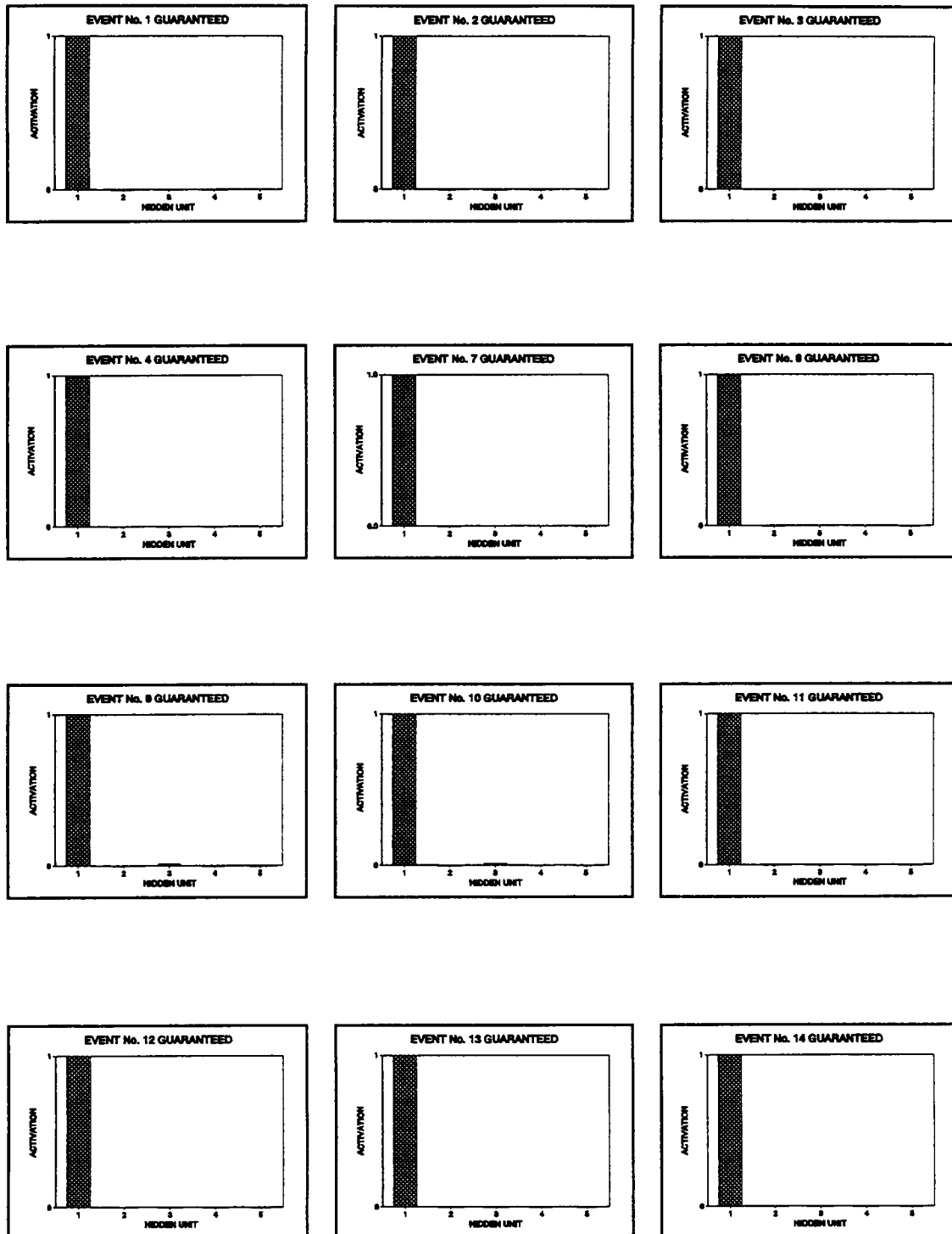


Figure 21 - Activations of the hidden units for the configurations yielding core-damage frequencies of the order of magnitude of 10^{-6}

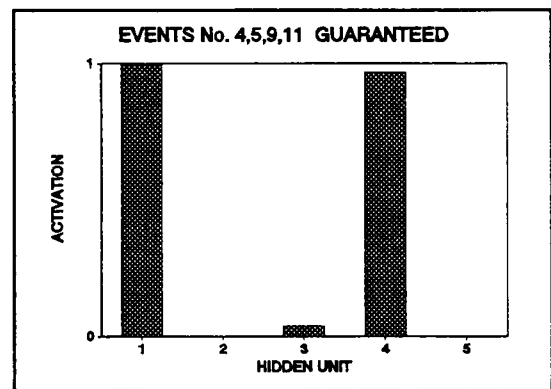
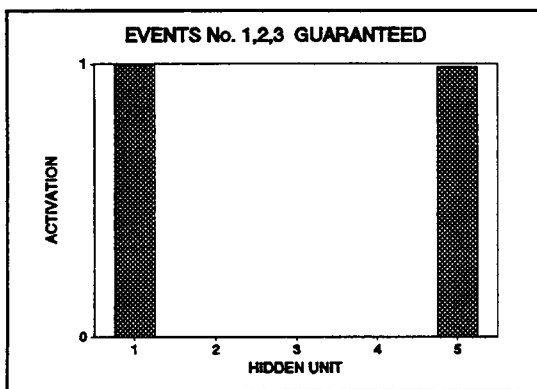
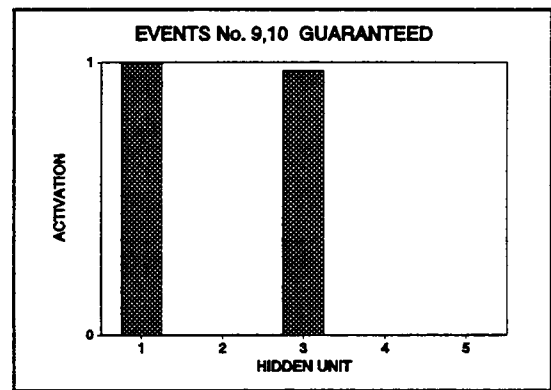
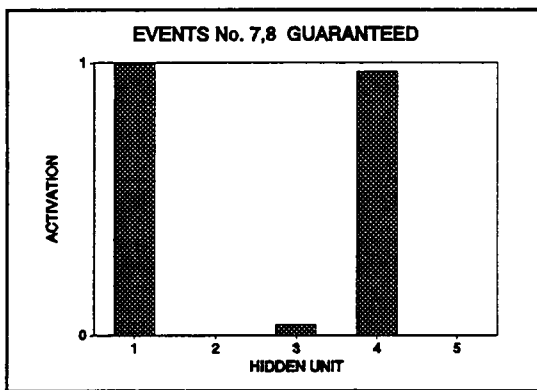
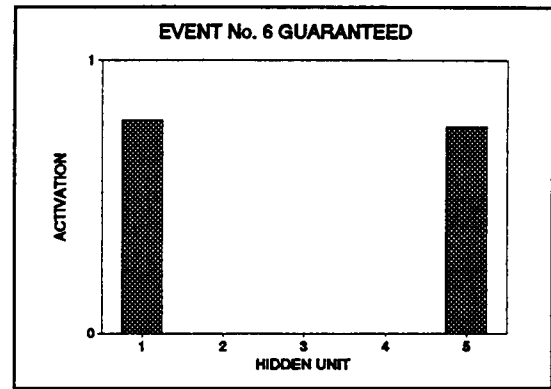
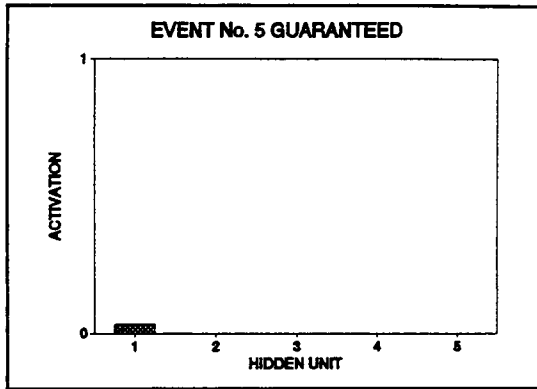


Figure 22 - Activations of the hidden units for the configurations yielding core-damage frequencies of the order of magnitude of 10^{-5}

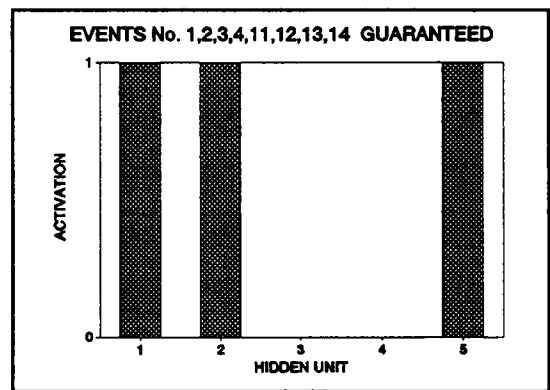
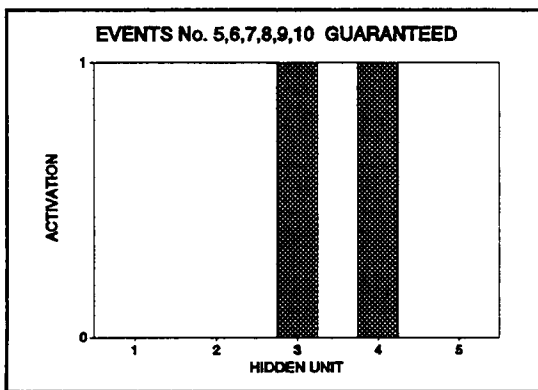
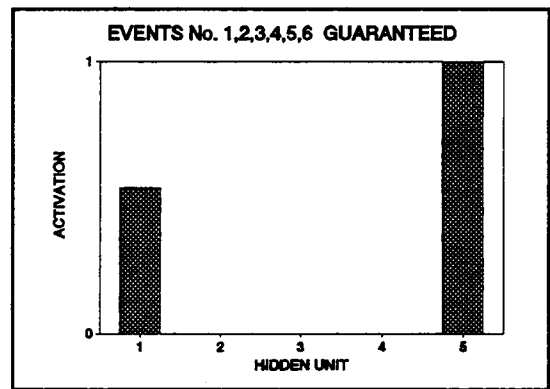
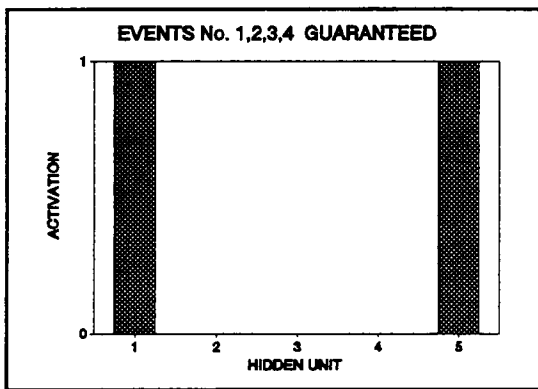
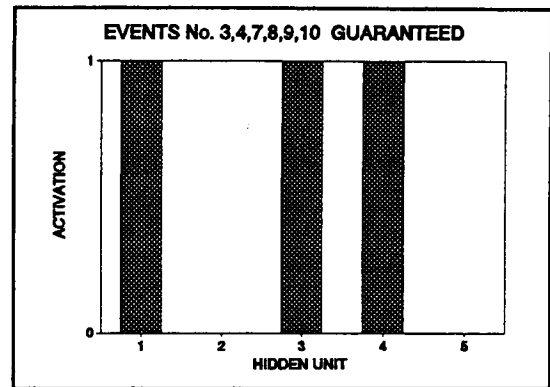
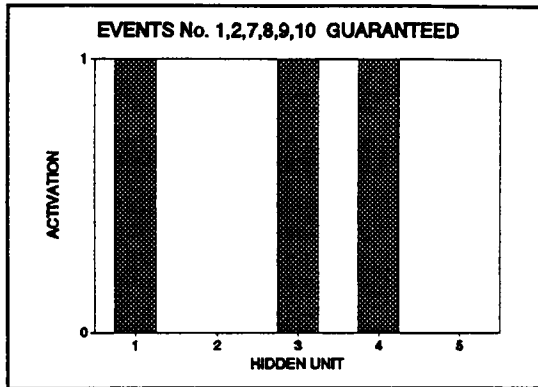


Figure 23 - Activations of the hidden units for the configurations yielding core-damage frequencies of the order of magnitude of 10^{-4}

damage frequencies close to the value of 1×10^4 . In Figure 23, the top two graphs represent the configurations correctly evaluated by the BPN, the center two graphs the configurations whose frequencies were underestimated (these patterns correspond to the configurations number one and four indicated in Table 5 as the two main flaws of the trained BPN), and the bottom two graphs the configurations whose frequencies were overestimated (these patterns correspond to the configurations number seven and ten in Table 5).

The patterns of activations of the BPN hidden units shown in the individual graphs in Figure 22 and Figure 23 demonstrate different ways how the BPN achieves increasing of the activation of the output unit. There are some interesting observations which can be made about these ways. One such observation is a striking difference between the hidden-unit activation patterns for the configurations with only the basic event number five guaranteed, and only the basic event number six guaranteed, respectively. Another observation is the strong activation of hidden unit number four when the two redundant basic events number seven and eight are guaranteed, the strong activation of unit number three when the events number nine and ten are guaranteed, and the strong activation of both these units when all these events are guaranteed. Still another observation is the same pattern obtained in the left bottom graph in Figure 22 and the left center graph in

Figure 23; as these Figures actually show configurations which yield different frequencies, this result indicates incorrect output of the BPN. The problem seems to be in the failure of the BPN to reflect the influence on the resulting frequency of guaranteeing the basic event number four in addition to the events one, two, and three. To improve the BPN performance, some input-output vector pairs with the basic event four considered guaranteed should probably be added to the training set. Nevertheless, although these observations are interesting, it is still very difficult to find really clear information directly useful for development of a better performing BPN.

5.3.2. Functional Link Network

The general description of the functional link network (FLN) structure and operation is given in section 4.2.2 of this study. The FLN is a special extension of the BPN based on an enhancement of the input vectors. This enhancement consists in an augmentation of the input vectors with additional elements representing some nonlinear transformations of their original elements. The objective of the added input vector elements is to directly provide the FLN with information which the BPN has to only extract from the input-output vector pairs used for its training. The transformations used for computation of the values of these

added elements are thus required to somehow reflect the nature of the approximated function.

As described in section 5.2.3.1, quantification of the PSA model involves mainly multiplication of the basic-event probabilities. Because these probabilities form also individual elements of the input vectors, the useful transformation to be used for computation of the additional elements of the FLN input vectors is clearly multiplication of these individual probabilities. Unfortunately, even for a moderate number of original elements this approach produces an enormous number of the new elements to be added to the input vectors. For the test PSA model, its input vector, which originally comprised of 14 elements, would have to be augmented with new elements representing 91 pairs, 364 triples, 1001 quadruples, and so on, of the original elements. It is quite clear that the FLN using so many input units would be very difficult to train in spite of an expected simplification of its task. Thus, there is no hope of achieving any better performance of the FLN, than the performance of the BPN described in section 5.3.1.

5.3.3. Modular Neural Network

The general description of the modular neural network (MNN) structure and operation is given in section 4.2.3. The MNN is

another extension of the BPN consisting in breaking the problem into several sub-problems addressed by separate BPNs whose individual results are combined into a final solution. This section describes the approach used and the results obtained when applying the MNN to the test problem.

5.3.3.1. MNN Design

When designing the MNN for any practical application, there are several decisions which have to be made. It has to be decided what ANN paradigm will be used for the expert networks of the MNN, how many of these expert networks will be used, and what the topology of the MNN gating network will be.

For function approximation, in general, it is recommended that expert networks without any activation function in their output units be used [56]. Another general recommendation suggests that a single expert network has to have a structure such that it cannot possibly solve the problem, addressed by the MNN as a whole, only by itself [56]. The experiments carried out for the test model confirmed both these general recommendations. When the expert networks used were three-layer BPNs using the sigmoid activation function in their output unit, after some training all the input vectors were processed by a single expert network, i.e. the MNN effectively degenerated into a simpler ANN consisting in this

single expert network (this happened even in the cases when the expert networks contained only one hidden unit, though this was the topology not sufficient to achieve any good performance with the BPN - see section 5.3.1.1). For all other designs, the MNN split the task among several expert networks as desired. When the expert networks were simple perceptrons (i.e., in fact, degenerated BPNs with two layers only) using the sigmoid activation function in their output unit, the MNN usually performed relatively well on the training set but did not achieve good generalization. In contrast, the MNN employing the expert networks not using the activation function in their output unit usually (but not always) performed slightly poorer on the training set but always achieved better generalization. The best general performance was achieved when the expert networks consisted in the two-layer perceptrons; addition of a hidden layer to the expert networks did not result in any improvement of the MNN performance but caused considerable troubles in its training. The best performing trained MNN employed expert networks consisting of 14 input units and a single output unit which only summed weighted signals coming from the input units.

There is no guidance in the literature concerning the number of expert networks to be used in a practical application. Clearly, this parameter is problem specific. The experiments with utilization of different numbers of

expert networks conducted for the test problem showed that this parameter influenced mainly the MNN generalization. The best performing trained MNN comprised five expert networks.

As far as the gating network is concerned, it is generally recommended to use a topology with only two layers [56]. Once again, the experiments carried out for the test model confirmed this recommendation. When the hidden layer was used in the gating network, it always caused considerable trouble with the MNN training resulting in the trained MNN always performing much poorer than the MNNs using the two-layer gating network. The best performing trained MNN also used the two-layer gating network.

5.3.3.2. Data Preprocessing for the MNN

As the MNN is just an extension of the BPN, it requires the same basic data preprocessing as the BPN. Processing of input signals in the MNN again essentially consists in a summation of their values. Therefore, the elements of the input vectors and also the only element of the output vectors need to be preprocessed using the decadic logarithmic transformation (cf. section 5.2.3.1).

On the other hand, in spite of the relation of the MNN and the BPN, the MNN output units do not use any activation

functions. The MNN thus can directly produce any output values required and there is no necessity for the re-scaling of the values of the output vector element.

5.3.3.3. MNN Training

As it is the case for the BPN, identification of the procedure allowing successful training of the MNN in practical applications is a difficult task requiring a lot of experimentation. Once again, there is no universal procedure to apply. In addition, the MNN paradigm is still so new that almost no guidance for its practical application can be found in the existing literature. On the other hand, the close relation of the MNN with the BPN allows the use of the experience acquired when training the BPN as a guidance for the MNN training.

The experiments with training the MNN for the test model showed that the procedure giving the best results in the training of the MNN is somewhat similar and somewhat different from the one identified for the BPN. The MNN appeared to have even more problems with distinguishing between the local and global (or "nearly-global") minima. The initial pre-training with the reduced training set, which proved to be very useful for the BPN training (cf. section 5.3.1.3), did not have the desired effect for the MNN training. Also, the MNN appeared

to be even more sensitive to the overtraining than the BPN. Thus, the cross-validation method identified as very helpful for the success of the BPN training proved to be even more useful for the MNN training. In case of the MNN, the learning curve obtained for the testing set always showed a distinctive minimum identifying the point when the overtraining started to take place. Indeed, the courses of the learning curves obtained for the training and testing sets of the input-output vector pairs were good indicators of the success of the training including the generalization of the trained MNN. For example, when a relatively low error was quickly achieved on the training set, it was an early indication of the MNN converging into a local minimum. Or, when the error on the testing set was oscillating for relatively many training epochs, it indicated another local minimum leading to a poor generalization of the trained MNN. In turn, the MNN promising relatively good performance was typically indicated by a balanced initial error on the training and testing set, a quick reaching of the point of overtraining, and a steady increase of the error on the testing set after this point.

The training and testing sets of the input-output vector pairs used for the MNN training were the same as those used for the training of the BPN (but, without reinforcement of any configuration - cf. section 5.3.1.3). These sets are given in Table 2 and Table 3, respectively (the events considered

guaranteed in the individual configurations are identified by their serial numbers assigned to them in Table 1). The best performing MNN trained for the test problem required only 2,100 training epochs while utilizing the batch update of the MNN weights and the momentum method of weight updating (cf. section 5.3.1.3).

5.3.3.4. Verification and Validation of the Trained MNN

As in the case of the BPN, the success of the MNN training for the test problem was, first of all, verified with the whole training set used for the training. The percentile errors of the resulting core-damage frequency values (i.e. the relative errors of their computed values in percents of the desired target values) computed by the best performing trained MNN for the configurations included in the training set are shown in Figure 24. As it can be seen, the errors of computed frequencies are rather high and frequently exceed the 10% tolerance. Thus, it is clear that the MNN did not succeed to learn the training set before the point, where it started losing its generalization properties due to overtraining.

Nevertheless, the real success of the MNN training is better judged by its performance while processing the input vectors never used for the training. For the test problem, the best performing trained MNN was validated first with the

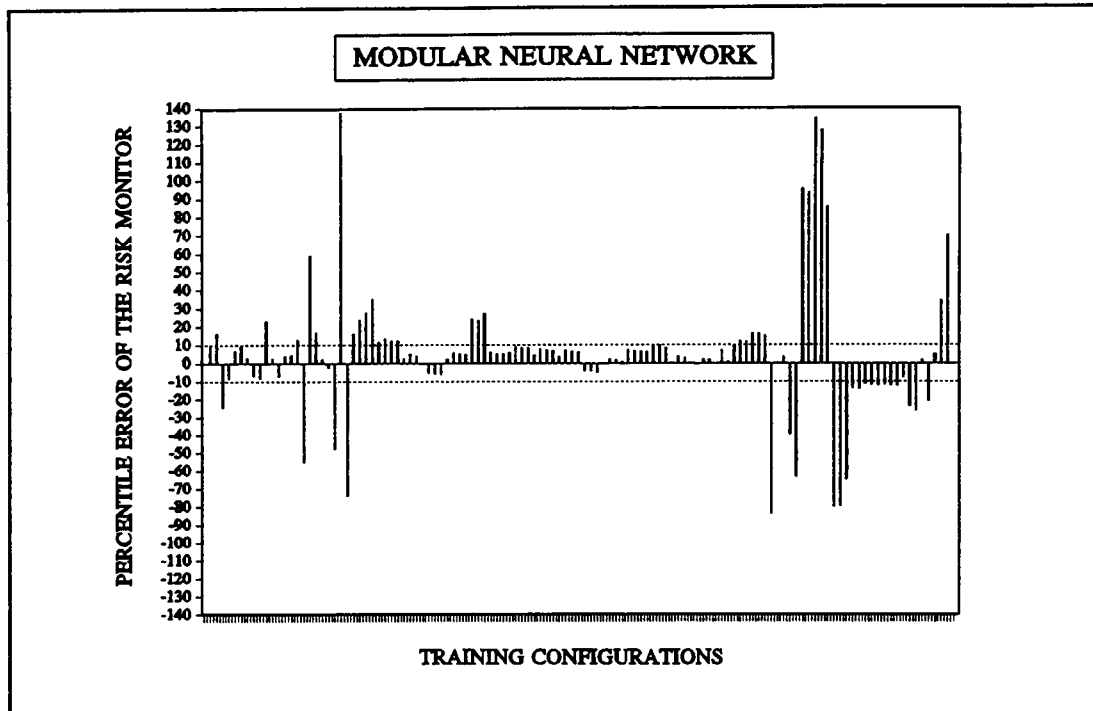


Figure 24 - Percentile errors of the core-damage frequency values computed from the MNN outputs for the training configurations

testing set already used during the training. The percentile errors of the resulting core-damage frequency values computed for the testing configurations are shown in Figure 25. In contrast to the BPN, the performance of the MNN on the testing set does not appear to be that much different from its performance on the training set. This result testifies to the good generalization provided by the MNN. The only exceptions are the huge errors obtained for the configuration number ten, where the MNN yielded the core-damage frequency of 8.23×10^{-4} , and the configuration number 15, where it yielded the value of

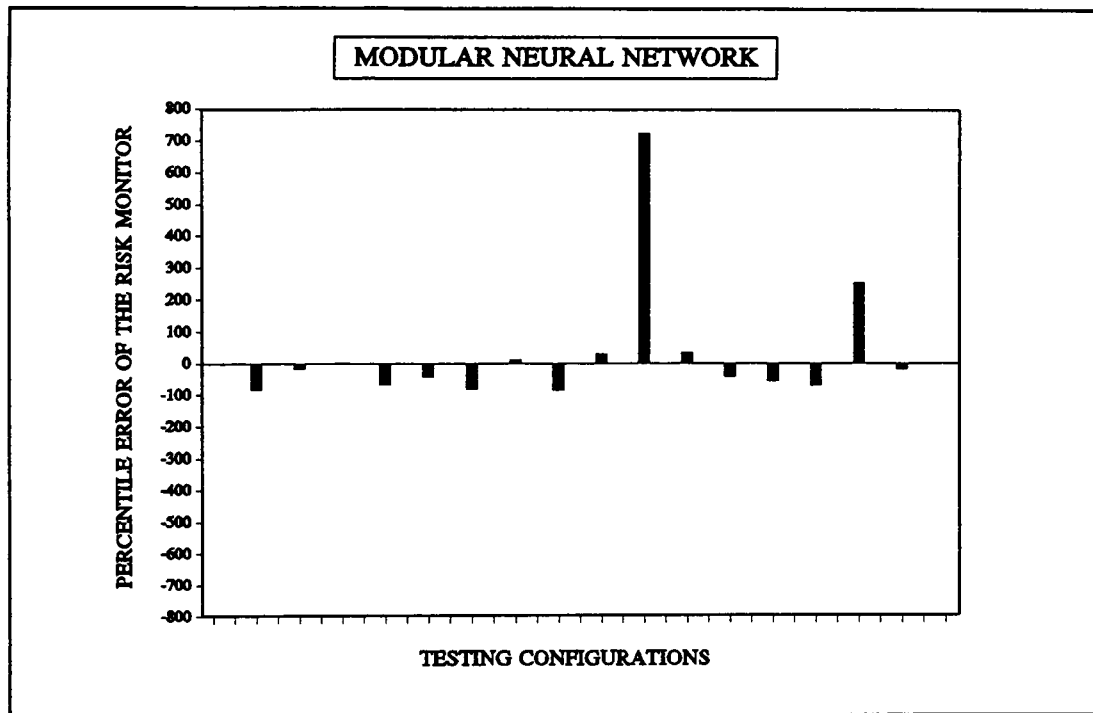


Figure 25 - Percentile errors of the core-damage frequency values computed from the MNN outputs for the testing configurations

3.55×10^{-4} (the correct value was 1×10^{-4} in both these cases). Nevertheless, as discussed for the similar results of the BPN in section 5.3.1.4, these error are not actually so severe, because they concern the configurations which can hardly develop in practice, and the errors can also be easily canceled out.

Once again as for the BPN, the performance of the best performing trained MNN was also validated on computing the RAWs of individual basic events for the configuration with

only the event number one guaranteed, and the configuration with the events number one and number four guaranteed. The results of the MNN computation are shown in the top graphs in Figure 26 and Figure 27, respectively. The bottom graphs in Figures 26 and 27 represent the corresponding correct RAW values. It is clear from these graphs that the MNN performance in this task was quite poor. The errors of the computed core-damage frequencies for individual configurations were so large that the resulting RAWs did not provide a correct picture of the relative importances of the individual basic events at all.

The overall MNN performance on all the data used for its verification and validation showed that the MNN was generally incapable of providing the core-damage frequencies within the desired error tolerance. In addition, it is unlikely that its performance could be significantly improved, even if more sample input-output vector pairs were used for its training. The MNN thus does not appear to be a suitable paradigm for the problem of interest.

5.3.4. Radial Basis Function Network

The general description of the radial basis function network (RBFN) structure and operation is given in section 4.2.4. This section describes the approach used and the

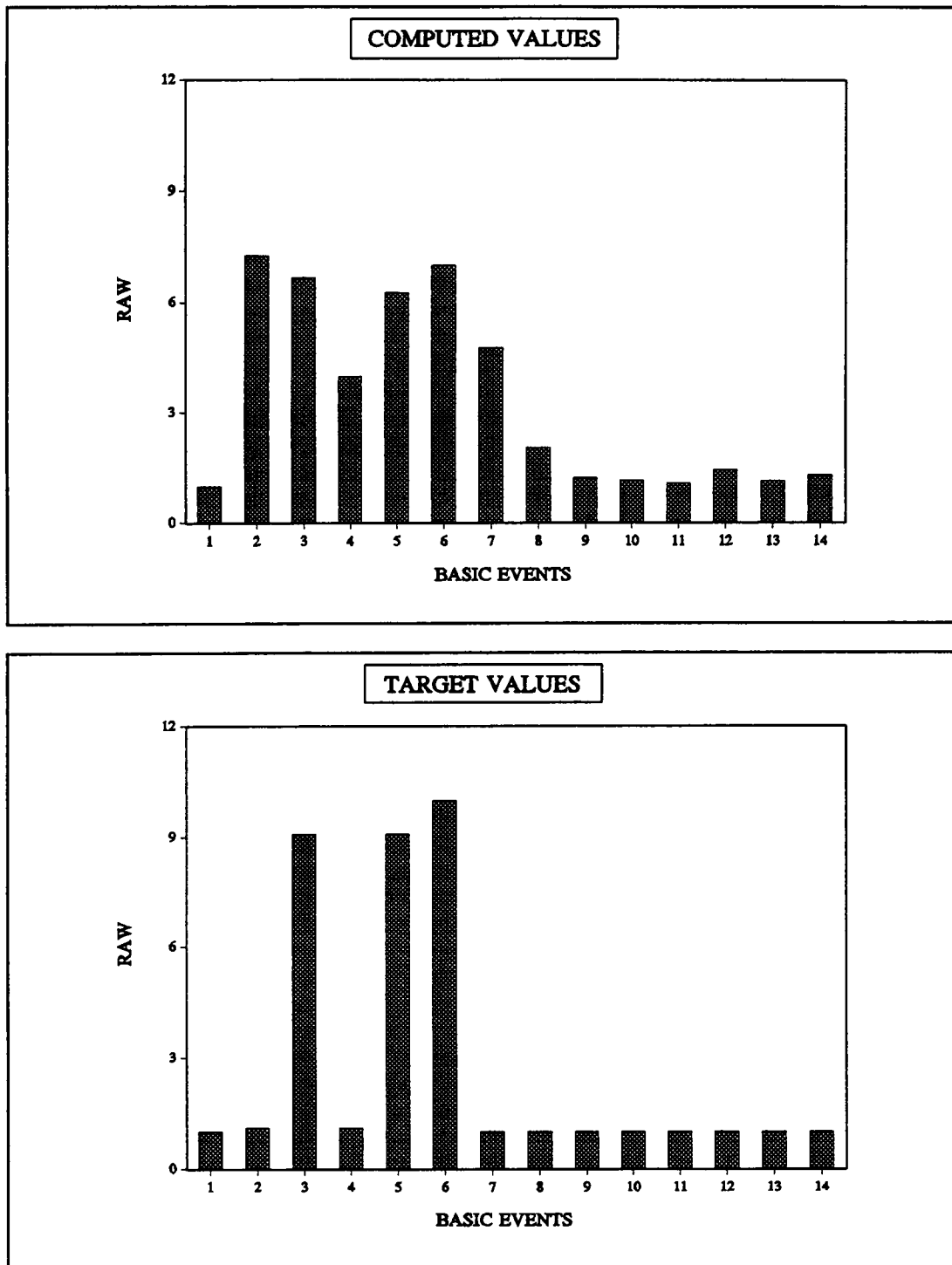


Figure 26 - RAWs of individual basic events as computed by the MNN for the configuration with only the event number one guaranteed

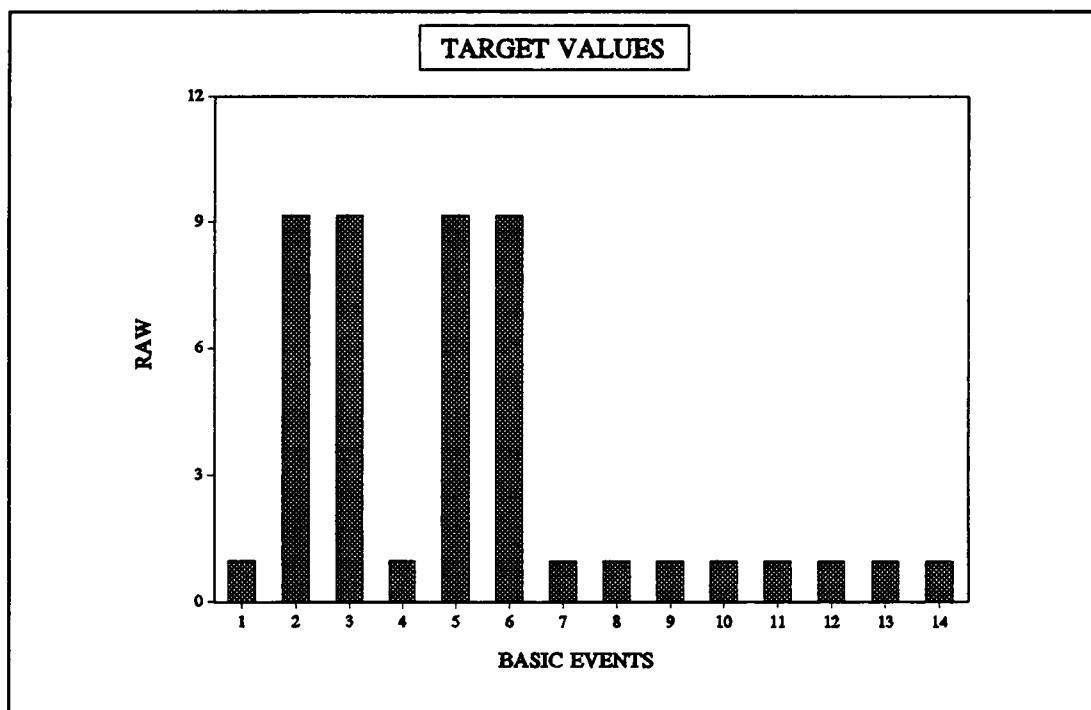
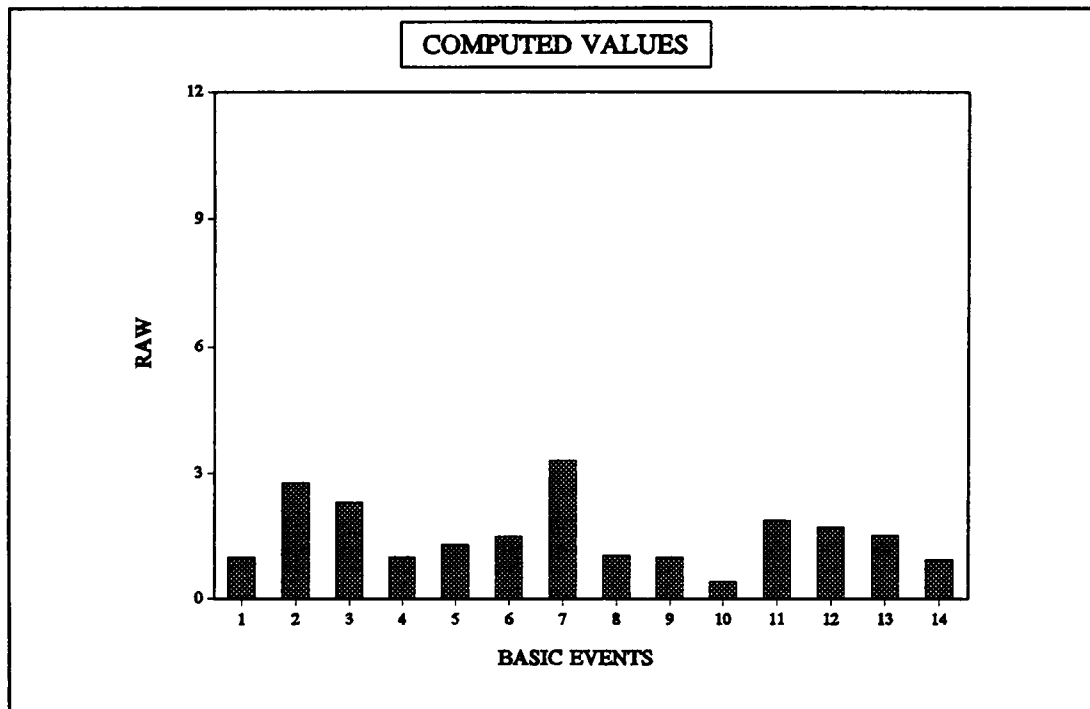


Figure 27 - RAWs of individual basic events as computed by the MNN for the configuration with the events number one and number four guaranteed

results obtained when applying the RBFN to the test problem. This approach is completely different from the one used by the BPN and its extensions, the FLN and the MNN.

5.3.4.1. RBFN Design

When designing the RBFN for a practical application, there is basically only one decision which has to be made. This decision is choosing of the number of pattern units of the RBFN. When using a fixed set of the input vectors for the RBFN training, the number of pattern units determines both the generalization properties of the trained RBFN and the error of its outputs for the training set. The training output error decreases steadily with the number of the pattern units, and it can be even eliminated when choosing the number of pattern units equal to the number of samples in the training set; this "ideal" is obviously very impractical. In contrast, the RBFN generalization properties are the best for a considerably lower number of the pattern units. In case of a well clustered input space, this number is naturally equal to the number of actual clusters in the input space. For an essentially un-clustered input space (unfortunately, rather typical for function approximation applications), the optimum number of pattern units is the number which ensures an approximately even coverage of the input space. Typically, the number of natural clusters in the input space is not known

when the training starts, and the optimum number of the pattern units has to be determined experimentally using the method of trial and error. For the test problem (while using the training set with 119 input vectors) the RBFNs with 10, 20, 23, 24, 25, 26, 27, 30, 35, 40, 50, and 120 pattern units were trained. The best overall performance was achieved with the RBFNs using 25 pattern units.

5.3.4.2. Data Preprocessing for the RBFN

In spite of the significant difference between the RBFN and the BPN, the operations performed by the RBFN output unit still include a summation of the weighted signals coming from the pattern units. Thus, it can be expected that although the positive effect of the preprocessing of elements of the input and output vectors using the decadic logarithmic transformation (cf. section 5.2.3.1) may not be so dramatic as for the BPN, its utilization should still facilitate training of the RBFN initially randomized weights and improve performance of the trained RBFN. This expectation was rather nicely proved by the experiments carried out. Performance of the trained RBFNs (with the same number of pattern units, of course) on the training set was almost the same regardless of whether the logarithmic transformation was used or not (yet, it was, actually, little worse when the transformation was not used) because the performance of the RBFN on the training set

depended mainly on the proper coverage of the input space determined by the number of the RBFN pattern units. For the testing set, however, performance of the trained RBFN was significantly better when the logarithmic transformation was used. The reason for this phenomenon was that the input vectors in the testing set did not necessarily match reasonably well the center of radial basis function of any single pattern unit, and the output produced by the RBFN was generated as a summation of the weighted activations of several pattern units. The RBFNs trained with the input-output vector pairs inherently more suitable for this operation were thus better prepared for this task. The preprocessing of the values of elements of the input and output vectors in the training set by using the decadic logarithmic transformation thus proved to be useful even for training of the RBFN.

As the RBFN output unit does not use any activation function, the RBFN can directly produce any output values required. Therefore there was no necessity of any additional re-scaling of the values of the output vector element.

5.3.4.3. RBFN Training

Similarly to the MNN, the RBFN paradigm is still relatively new, and so there is very little guidance provided in the

existing literature for its practical application. This generally makes identification of a suitable training procedure for the RBFN rather difficult. On the other hand, the training of the RBFN is relatively straightforward and does not offer many possible options.

The main task of the RBFN training involves an unsupervised training aimed at the determination of proper values of parameters of the radial basis functions assigned to the individual pattern units. The success of this training generally depends on a suitable choice of the clustering algorithm, which is utilized for division of the input vectors included in the training set into classes and determination of the centers of the radial basis functions, and a suitable choice of the algorithm used for determination of the widths of the radial basis functions. Nevertheless, the lack of practical experience with the RBFN has also projected itself to the implementation of the RBFN paradigm as provided in the NeuralWorks Professional II/PLUS software [70]. This software, in fact, does not provide any choice of these algorithms and automatically employs the k-means algorithm, and the two-nearest neighbor algorithm, respectively. These algorithms, however, proved to be suitable for the task and there was no reason to believe that any substantial improvement could be obtained when utilizing any other algorithms.

The rest of the RBFN training consists in the supervised training of the output unit using an algorithm for minimization of the error of the produced outputs. This training is practically identical to the one utilized when training the MNN output unit (and thus very similar to the BPN training too). Therefore, it was no surprise, that the same procedure as the one identified suitable for the MNN training proved to be successful for the RBFN as well. The success of the RBFN training did not require the initial pre-training with the reduced training set (cf. section 5.3.1.3), but did require the frequent cross-validation of current performance of the RBFN with the testing set. The learning curve obtained for the testing set always showed a quite distinctive minimum identifying the point when the overtraining started to take place (indeed, this effect appeared to be even stronger for the RBFN than in the case of the MNN). Also similarly to the MNN, the error on the testing set oscillating for many training epochs was an indicator of a local minimum leading to poor generalization of the trained RBFN. A balanced initial error on the training and testing set, a quick reaching of the point of overtraining, and a steady increase of the error on the testing set after this point, then indicated the RBFN promising the best achievable overall performance.

Again, the training and testing sets of the input-output vector pairs used for the RBFN training were the same as used

for the training of the BPN and the MNN (with the exception that no reinforcement of training of any configuration was used - cf. section 5.3.1.3). These sets are given in Table 2 and Table 3, respectively. The best performing RBFN trained for the test problem required about 11,500 training epochs in the second supervised training stage while utilizing the batch update of the MNN weights and the momentum method of weight updating (cf. section 5.3.1.3).

5.3.4.4. Verification and Validation of the Trained RBFN

As in the case of the BPN and the MNN, the success of the RBFN training for the test problem were, first of all, verified with the whole training set used for its training. The percentile errors of the resulting core-damage frequency values computed by the best performing trained RBFN for the configurations included in the training set are shown in Figure 28. As it can be seen, the errors of the computed frequencies were quite high and almost always exceeded the 10% tolerance. Therefore it has to be concluded that the RBFN, selected for its best overall performance, did not succeed to learn the training set at all. In fact, even when the RBFN was simply trained with the training set as long as its error on this set was decreasing, its final performance on the training set was still rather poor. Of course, when the RBFN was provided with more pattern units, its performance on the

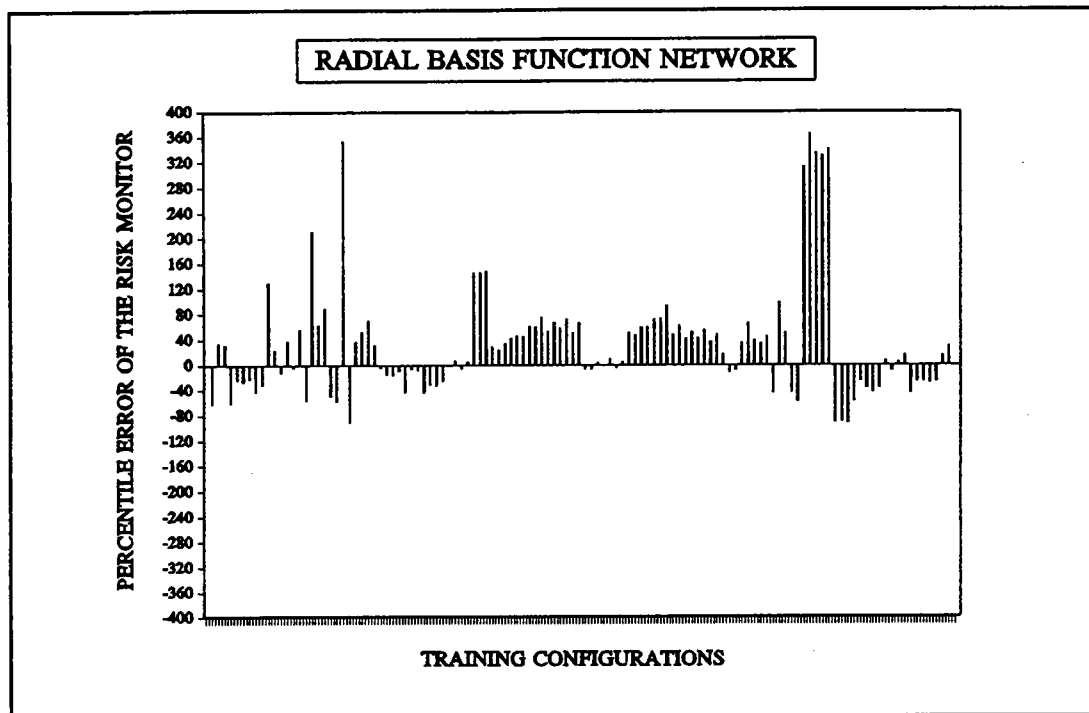


Figure 28 - Percentile errors of the core-damage frequency values computed from the RBFN outputs for the training configurations

training set improved; yet, this improvement was always achieved on the expense of its performance on the testing set and the other input vectors not included in the training set.

The next step of the RBFN verification and validation was checking of its performance while processing the input vectors never used for its training. The best performing trained RBFN was again first validated with the testing set already used during its training. The percentile errors of the resulting core-damage frequency values computed for the testing

configurations are shown in Figure 29. Performance of the RBFN on the testing set appeared to be actually better than its performance on the training set (very likely only a coincidence) but it was still rather unsatisfactory because the errors again almost always exceeded the 10% tolerance. In addition, it was especially undesirable that the majority of the errors actually indicated an underestimation of the resulting core-damage frequencies. Such a result was unacceptable for practical purposes and it also meant that there was no chance to cancel these errors out.

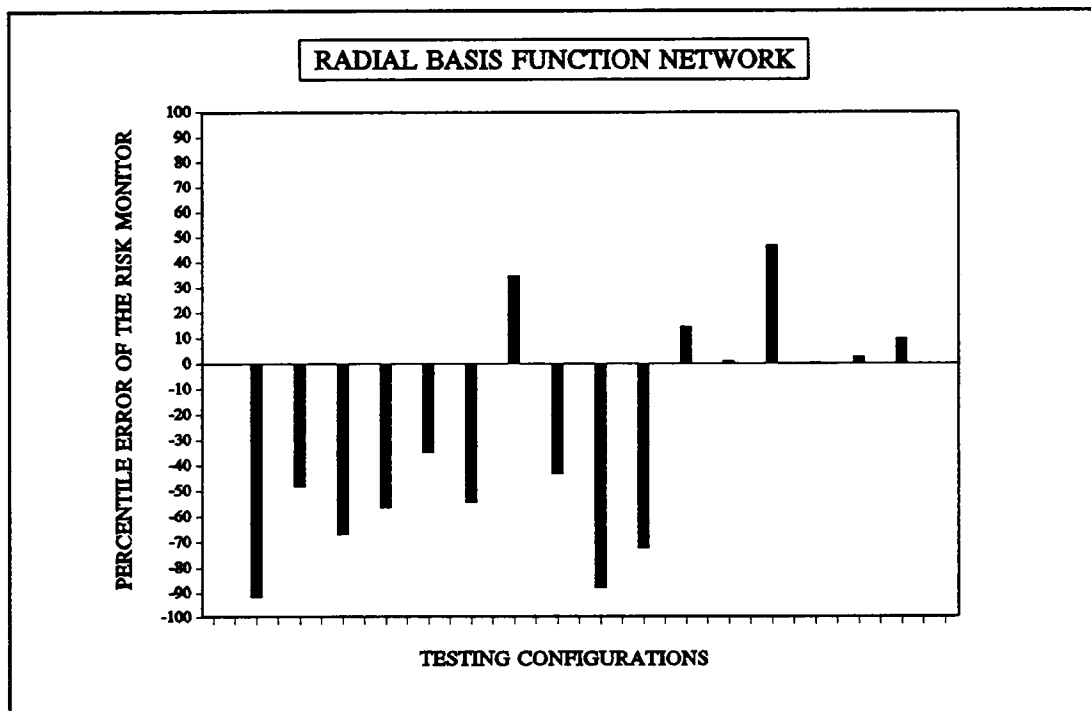


Figure 29 - Percentile errors of the core-damage frequency values computed from the RBFN outputs for the testing configurations

Once again as for the BPN and the MNN, the performance of the best performing trained RBFN was also validated on computing the RAWs of individual basic events for the configuration with only the event number one guaranteed, and the configuration with the events number one and number four guaranteed. The results of the RBFN computation are shown in the top graphs in Figure 30 and Figure 31, respectively. The bottom graphs in Figures 30 and 31 represent the corresponding correct RAW values. It is clear from these graphs that the RBFN performance in this task was again very poor. The errors of the computed core-damage frequencies for individual configurations were so big that the resulting RAWs did not provide a correct picture of relative importances of individual basic events.

The overall performance of the RBFN on all the data used for its verification and validation showed that it was completely incapable of producing the core-damage frequencies within the desired error tolerance. Its performance might be improved only by the addition of a very significant number of additional input-output vector pairs into the training set. The necessary number of these additional training samples would, however, clearly exceed an amount of data which is practically feasible to obtain. The RBFN thus appear to be a paradigm quite unsuitable for solving the problem of interest.

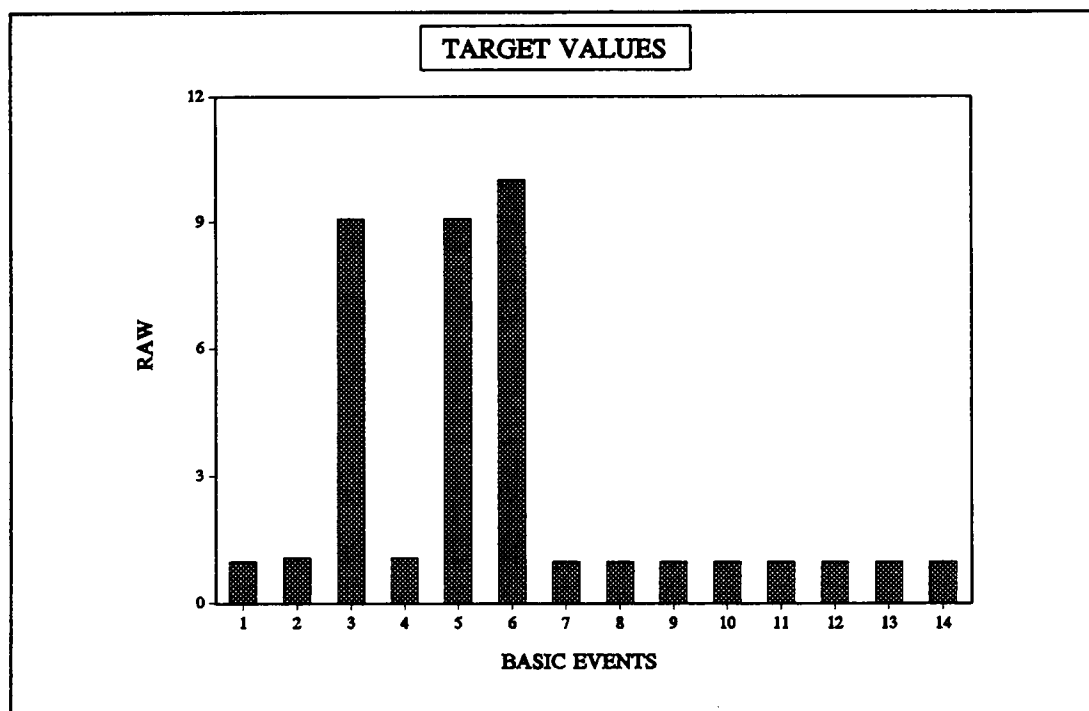
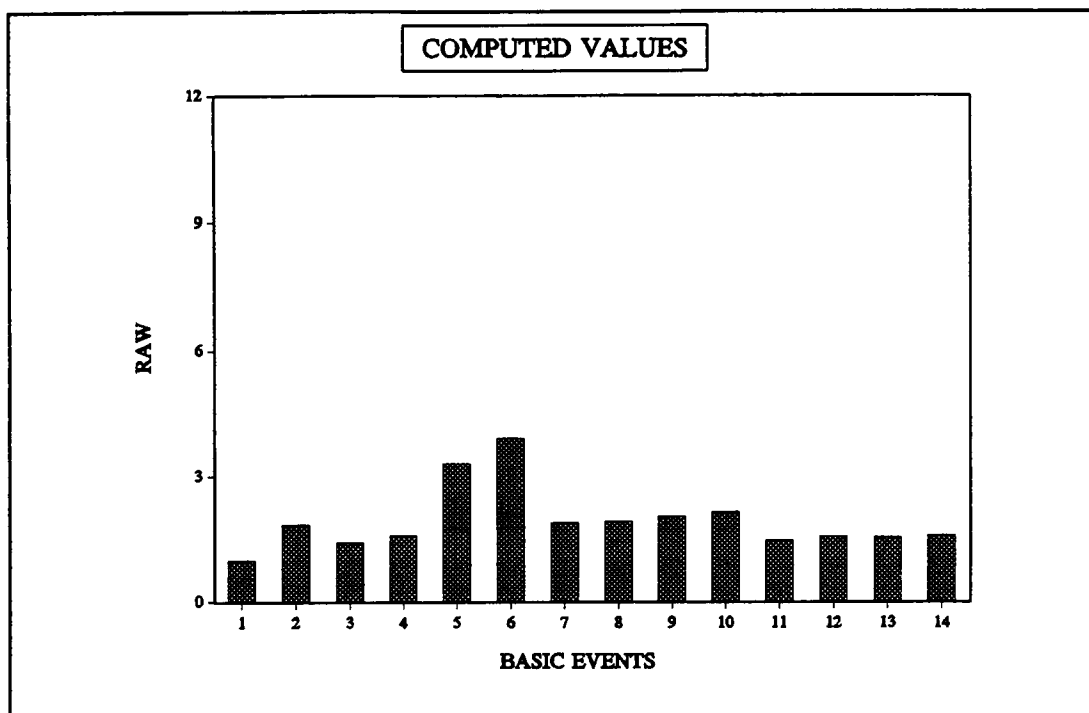


Figure 30 - RAWs of individual basic events as computed by the RBFN for the configuration with only the event number one guaranteed

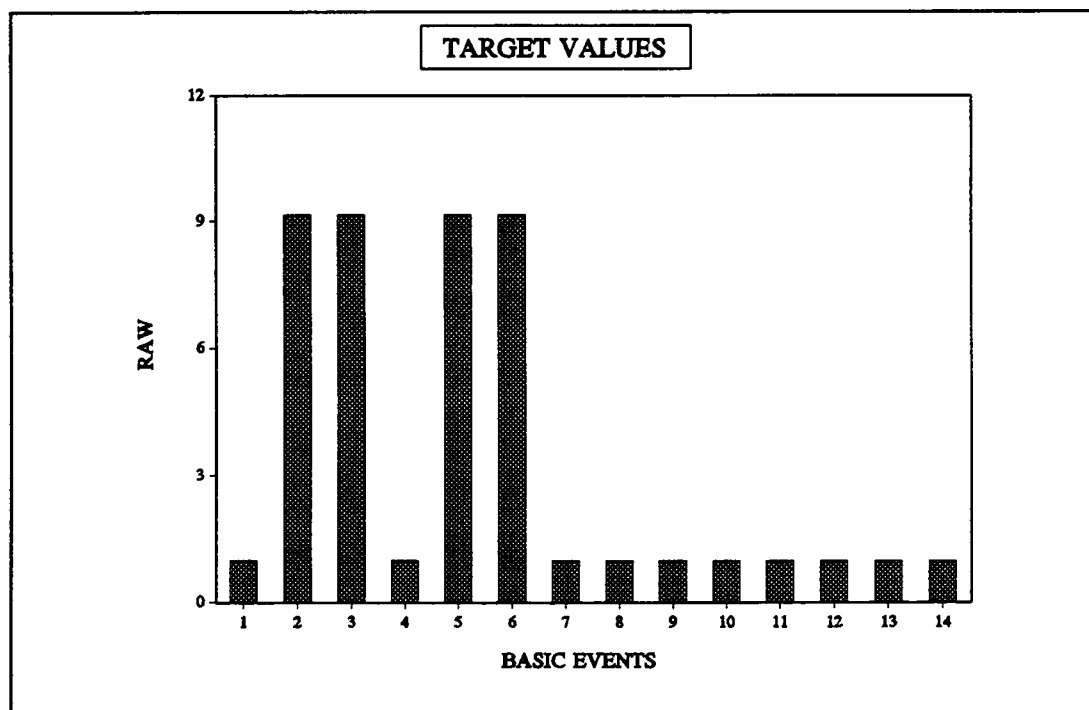
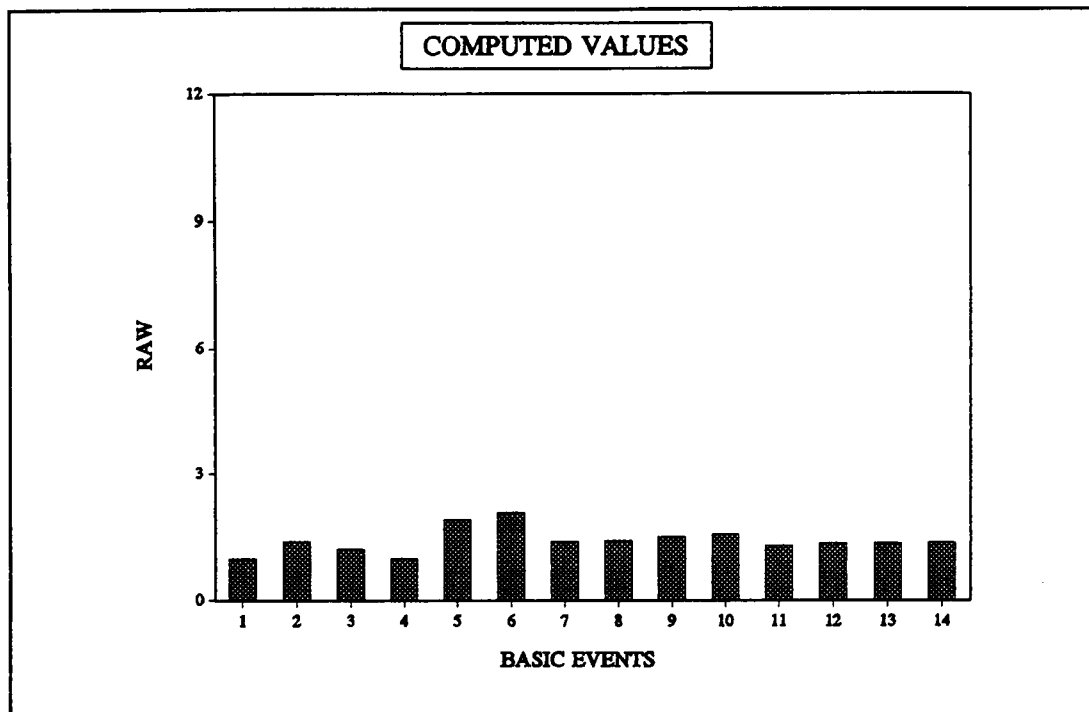


Figure 31 - RAWs of individual basic events as computed by the RBFN for the configuration with the events number one and number four guaranteed

5.3.5. Generalized Regression Neural Network

The general description of the generalized regression neural network (GRNN) structure and operation is given in section 4.2.5. This section describes the approach used and the results obtained when applying the GRNN to the test problem. Since the GRNN is just a special extension of the RBFN, there are also some similarities in the approaches used when applying these two paradigms.

5.3.5.1. GRNN Design

When designing the GRNN for any practical application, there are usually two decisions which have to be made. These decisions determine number of the GRNN pattern units, via specification of the maximal radius of the clusters in the input space, and the width of the radial basis functions of the GRNN pattern units. Fortunately, suitable values of these parameters can be determined one by one, because later setting of the radial basis function width does not change the impact of the value of the cluster maximal radius chosen before.

The first decision, i.e. choosing of the maximal radius of the clusters, predetermines number of the clusters distinguished in the input space by the clustering algorithm. This parameter is very important as it determines the performance of the trained GRNN on the training set.

Unfortunately, its setting is obviously very problem specific, and the rare references available are still rather theoretical and provide no guidance for this task. Therefore, many experiments were necessary to determine a suitable value for this parameter for the test problem. The values from 0.05 all the way to 10 were gradually tested and the value of 3.4 was finally identified as the one yielding the best overall results. For this value, the clustering algorithm identified 20 different clusters in the input space and, accordingly, built the GRNN pattern layer containing 20 pattern units. Nevertheless, even in this stage it was necessary to consider the main goal of the design, i.e. development of the GRNN capable of good generalization. Thus, the resulting GRNN topology rather ironically represented the solution aimed to achieving good generalization properties of the resulting GRNN. In contrast, the best performance solely on the training set could be achieved when setting a very small value of the maximal cluster radius which would result in creating the GRNN with one pattern unit for each input vector in the training set. Such a GRNN would be, naturally, unable to generalize at all and its performance on any input vectors not included in the training set would be inevitably very poor.

The next decision to be made is choosing of the width of the radial basis functions of the pattern units. This parameter determines overall performance of the trained GRNN

on the input vectors not included in the training set. Its setting is again very problem specific, but at least some guidance is already available in the existing literature [59,60]. Using this knowledge, the different values of the width from the range of 0.05 thru 1.5 were tried for the test problem; the best generalization of the GRNN was obtained for the value of 0.9.

5.3.5.2. Data Preprocessing for the GRNN

In contrast to the RBFN, there is no adjusting of the initially randomized weights involved in the GRNN training. Therefore, it could be expected that the preprocessing of elements of the input and output vectors using the decadic logarithmic transformation (which proved to be so useful for all other ANN paradigms considered so far) would not have any effect on the performance of the trained GRNN. The experiments conducted fully confirmed this expectation and really no differences were observed.

Thus, almost no preprocessing of the data is needed for the successful training of the GRNN. Nevertheless, it is still necessary to transform values of the output vector element using the logarithmic transformation so that the approximately uniform percentile error of the GRNN output along the range of the possible output values can be achieved.

5.3.5.3. GRNN Training

The training of the GRNN is essentially accomplished after a single pass over the samples included in the training set. From a practical point of view, however, the GRNN training consists of two separate steps. During these steps the maximal radius of clusters in the input space, and the width of the radial basis functions of the pattern units are determined, respectively (cf. section 5.3.5.1).

The first step of the training consisted in experimentation with different possible values of the cluster maximal radius while checking the resulting performance of individual trained GRNN only with the input-output vector pairs included in the training set. This way an optimum value of this parameter was gradually determined. To enable comparison of performance of different ANN paradigms on the test problem, the same training set as used for training of the other paradigms was used again (however, comparing to the BPN, no reinforcement of training of any configuration was used - cf. section 5.3.1.3). The configurations included in the set are given in Table 2 (the events considered guaranteed in the individual configurations are identified by their serial numbers assigned to them in Table 1).

The second step consisted in experimentation with different possible values of the radial basis function width while

keeping the cluster maximal radius fixed to the value determined in the previous step. During this step, however, the performance of the individual trained GRNN was checked only with the input-output vector pairs not included in the training set. This allowed the determination of an optimum value of the second GRNN parameter. The set used for this task for the test problem was again the same, as the one used in training of the other considered ANN paradigms for occasional testing of their current performance. The configurations included in this set are given in Table 3.

5.3.5.4. Verification and Validation of the Trained GRNN

As in the case of all ANN paradigms considered before, the success of the GRNN training for the test problem were, first of all, verified on the set used for the training. The percentile errors of the resulting core-damage frequency values computed by the best performing trained GRNN for the configurations included in the training set are shown in Figure 32. As it can be seen in Figure 32, errors of the computed frequencies were very high and exceeded the 10% tolerance practically for all the considered configurations. Thus, it has to be concluded that the GRNN designed to provide good generalization properties, did not succeed in learning the sufficiently correct mapping of the training set. Of course, if the GRNN would be allowed to fully concentrate on

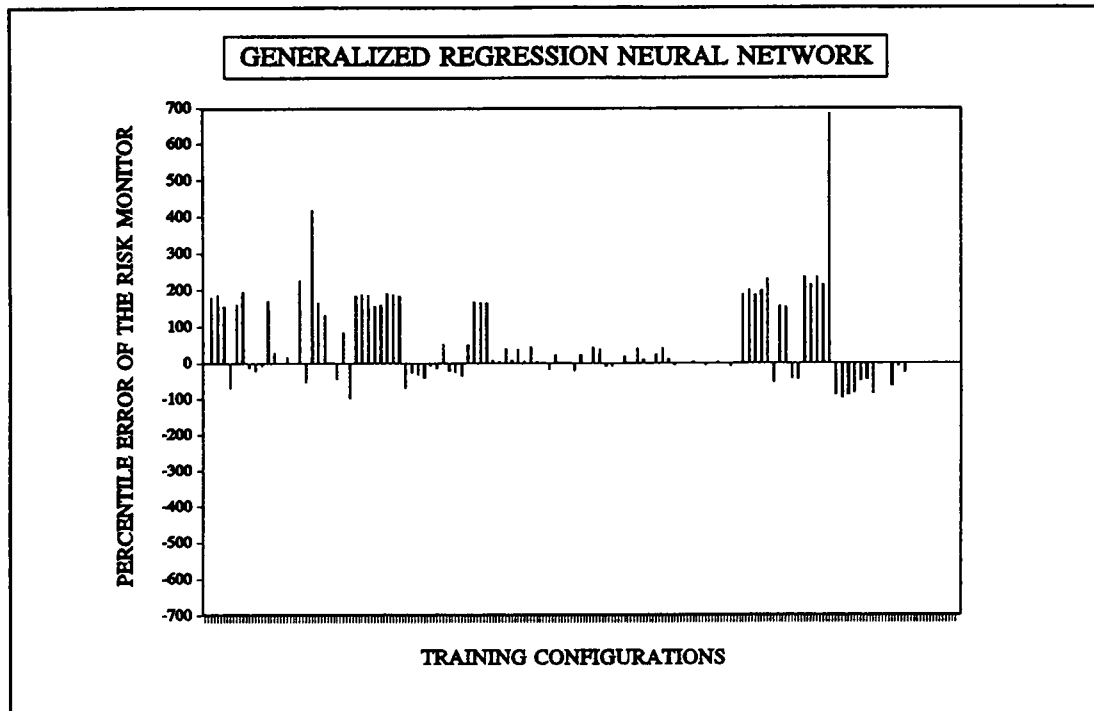


Figure 32 - Percentile errors of the core-damage frequency values computed from the GRNN outputs for the training configurations

learning only the training set, it would be able to achieve an absolutely perfect performance (i.e. without any error at all). Yet, this approach would naturally result in a complete GRNN failure when presented with any input vector not included in the training set.

Next step of the GRNN verification and validation was the checking of its performance while processing the input vectors not used for its training. The best performing trained GRNN was again first validated with the set used as the testing set

in training of the other considered ANN paradigms. The percentile errors of the resulting core-damage frequency values computed for these "testing" configurations are shown in Figure 33. The performance of the GRNN on this set was about the same as on the training set overviewed before. Thus it can be concluded that the desired generalization capability of the GRNN was reached. The performance of the GRNN was, however, again very unsatisfactory.

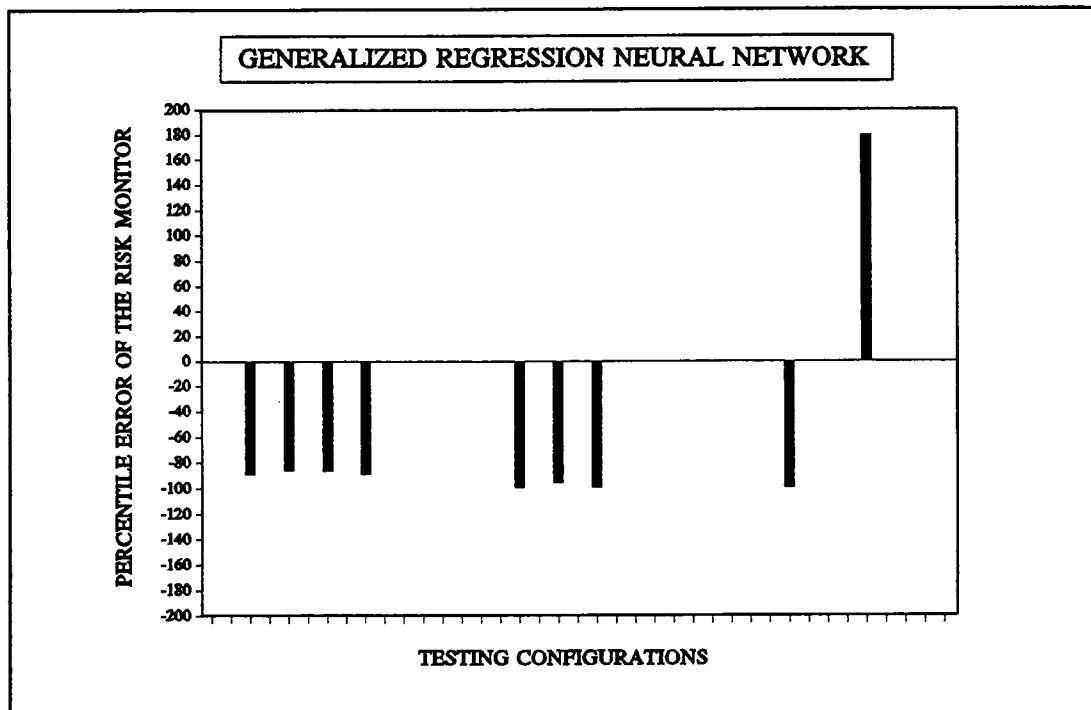


Figure 33 - Percentile errors of the core-damage frequency values computed from the GRNN outputs for the testing configurations

As with all the previous paradigms, the last step in the validation of the performance of the best performing trained GRNN was computing the RAWs of individual basic events for the configuration with only the event number one guaranteed, and the configuration with the events number one and number four guaranteed. The results of the GRNN computation are shown in the top graphs in Figure 34 and Figure 35, respectively. The bottom graphs in Figures 34 and 35 represent the corresponding correct RAW values. As it is clear from these graphs, the GRNN performance in this task was a complete failure. Clearly, the errors of the computed core-damage frequencies for individual configurations were so big and so randomly distributed that the resulting RAWs did not provide any information at all.

The overall GRNN performance on all the data used for its verification and validation showed that when using the given training set, the GRNN was incapable of producing the core-damage frequencies within the desired error tolerance. Its performance might be improved only by the addition of many new input-output vector pairs into the training set. However, the necessary number of these additional training samples necessary would inevitably exceed the amount of data which is practically plausible to obtain. Thus, also the GRNN is a quite unsuitable paradigm for the problem of interest.

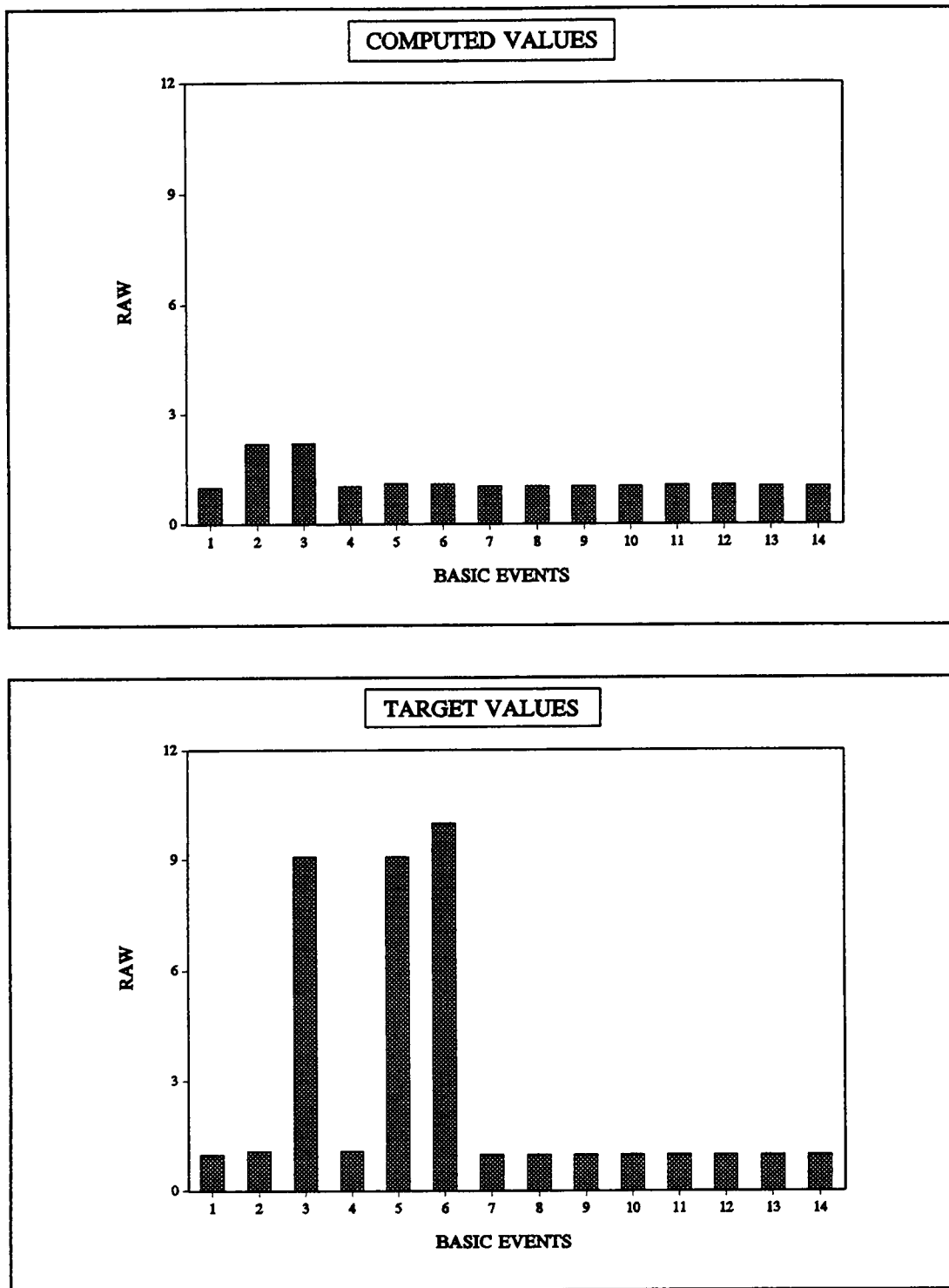


Figure 34 - RAWs of individual basic events as computed by the GRNN for the configuration with only the event number one guaranteed

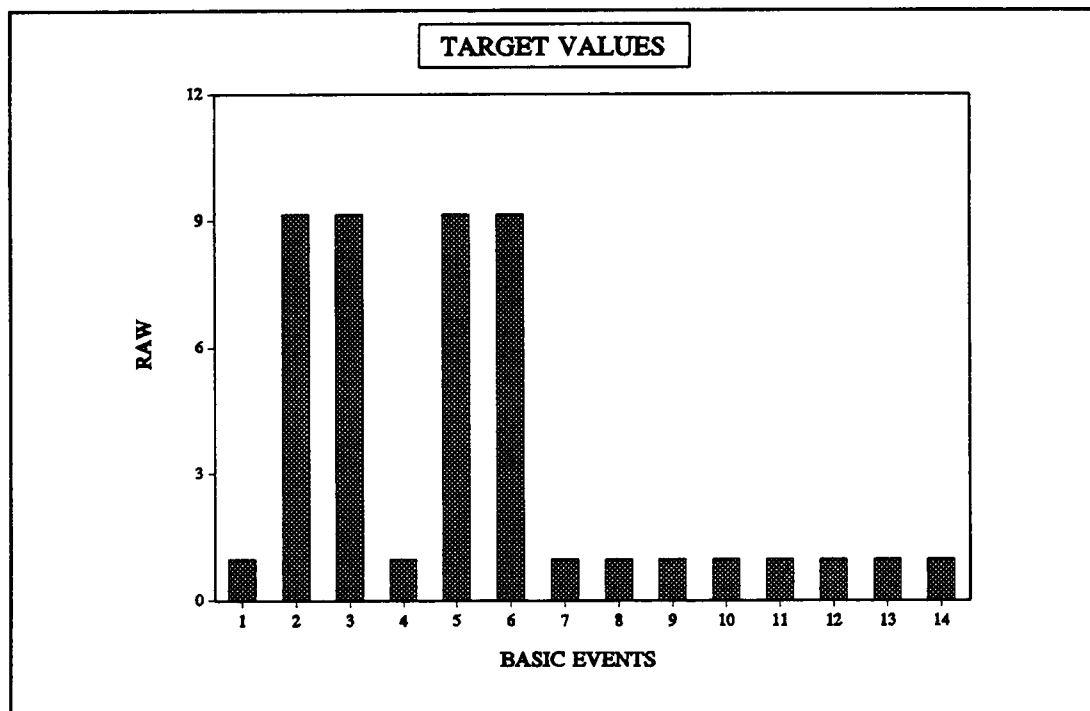
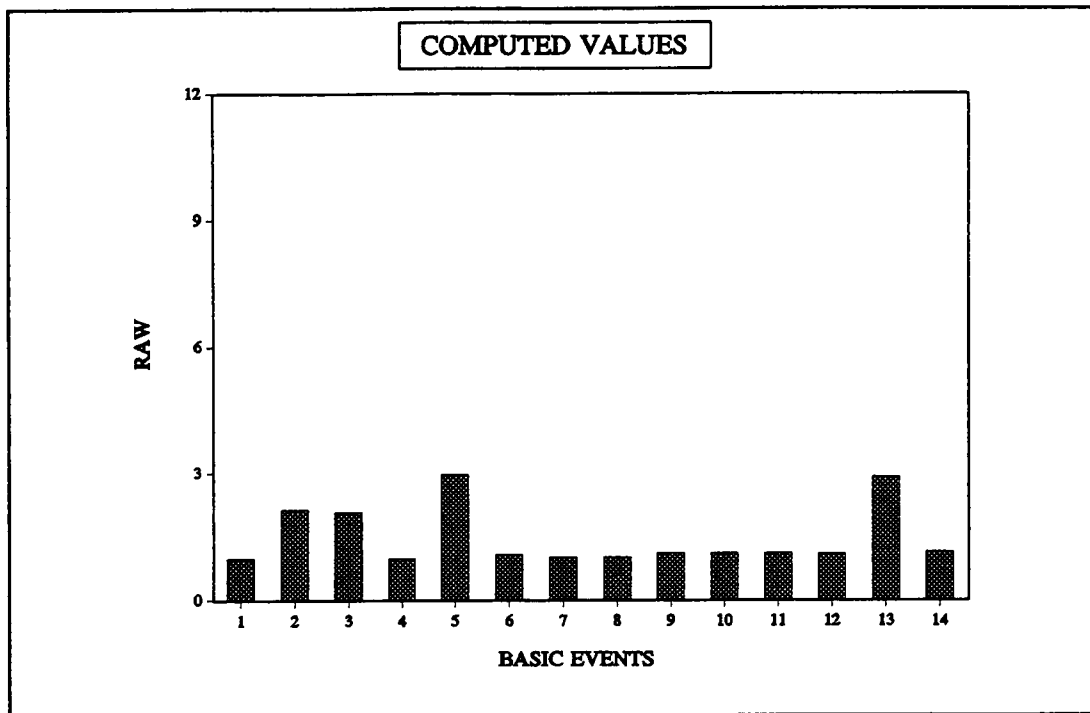


Figure 35 - RAWs of individual basic events as computed by the GRNN for the configuration with the events number one and number four guaranteed

5.4. Evaluation of ANNs Performance on the Test PSA Model

In the previous sections, potential of several different ANN paradigms for utilization in the application of concern was investigated using the rather simple test PSA model. The paradigms considered included the backpropagation network (BPN), the functional link network (FLN), the modular neural network (MNN), the radial basis function network (RBFN), and the generalized regression neural network (GRNN). In this section, the performances of these ANN paradigms are compared, and their overall performance is evaluated.

5.4.1. Comparison of the Individual ANN Paradigms

First of all, of the listed ANN paradigms which were all potentially useful for function approximation applications, the utilization of the FLN was not attempted. After considering the specific application, it was realized that there was no real hope of the FLN achieving any better performance than the BPN for the application of concern (the FLN is just a special extension of the BPN - cf. section 5.3.2). Thus, only the four remaining paradigms were actually applied to the test problem.

Numerous ANNs were trained for each of these paradigms while testing the influence of specific settings of various

variable parameters of these paradigms. Of the ANNs trained for each of the ANN paradigms, the one yielding the best overall results was chosen as a representative of the particular paradigm. The overall percentile errors of the core-damage frequency values produced by these representative ANNs are compared in Figure 36 (the adjective "percentile" here indicates that these errors represent the relative errors of the values produced by these ANNs in percents of the desired target values).

The individual rows of graphs in Figure 36 represent the overall percentile errors for different sets of configurations of the basic-event probabilities of the test PSA model. The graphs in the top row show the errors on the set used for training of all the ANNs, the middle row graphs show the errors on the set used for testing of the ANNs during their training, and the bottom row graphs show the errors on the set used for the final validation of the trained ANNs (this set is named the "validating set" and comprises 14 configurations which need to be evaluated when computing the RAWs for the configuration with the basic event number one guaranteed, i.e. the configurations with the event number one and always one more basic event - including itself - considered guaranteed; the events are identified by their serial numbers assigned to them in Table 1). In each of the rows, the left graph shows the average percentile errors of the core-damage frequency

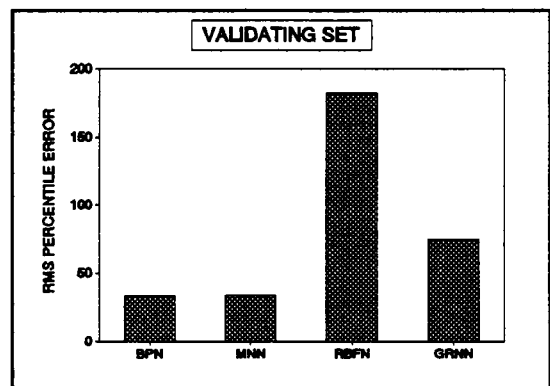
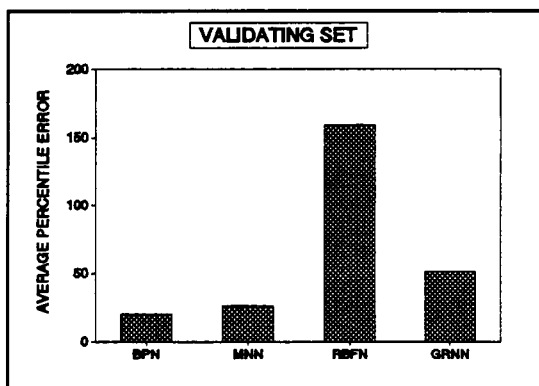
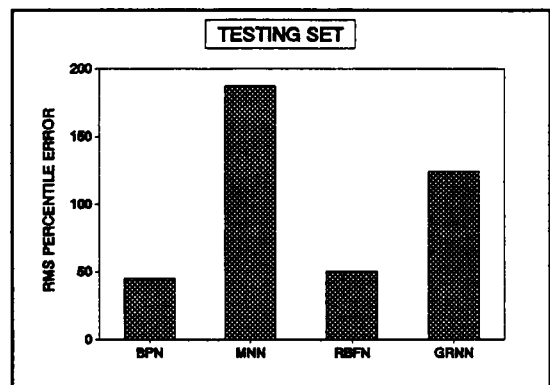
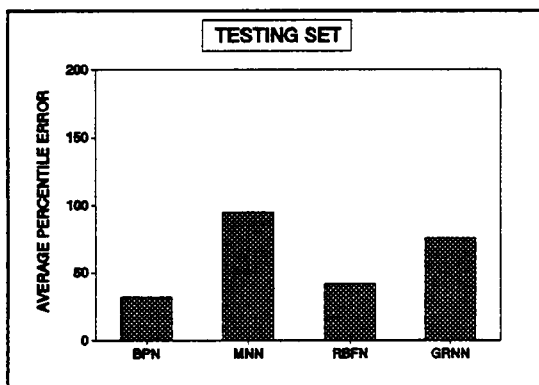
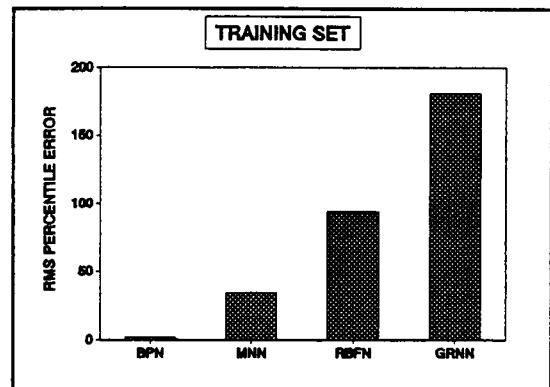
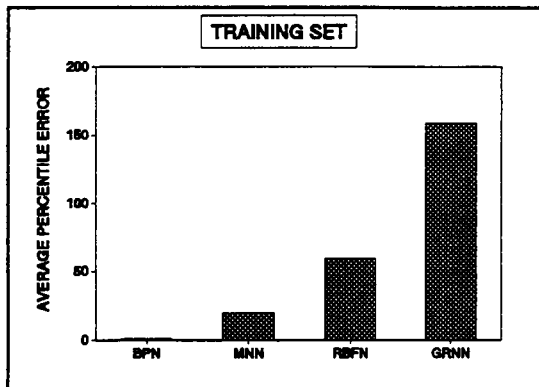


Figure 36 - Overall percentile errors of the core-damage frequency values computed by the selected ANN paradigms for different sets of the model configurations

values over the whole particular set (i.e. the average value of absolute values of the individual errors obtained for each of the samples in the set), and the right graph shows the root-mean-squares (RMS) percentile errors of the core-damage frequency values over the whole particular set, as computed by the individual ANNs. The RMS error in comparison with the average error gives a higher credit to the magnitude of the individual errors, i.e. of the two sets with the same average error, the one which includes the samples with the biggest errors of the both sets has a higher value of the RMS.

It is quite clear from Figure 36, that the BPN achieved the best performance of all the considered ANN paradigms (in fact, it returned the lowest error value of all the considered paradigms in every single case shown). The second best performance can be probably assigned to the MNN, though it appeared to have some difficulties, especially when processing the testing set. The performance of both the RBFN and GRNN was significantly poorer.

As the procedures developed for training the individual ANNs were aimed at achieving the best possible generalization properties of the trained ANNs, the best indicator of the success of the ANN is, possibly somewhat ironically, its performance on the training set. The best ANN generalization properties represent its state when it provides the best

overall approximation of the relation between its input and output space. The errors in mapping the training input-output vector pairs, which are imprinted in the ANN structure in this state, thus measure the ability of the ANN to learn this relation from the training set used. The magnitudes of these errors generally characterize the level of incompleteness of the training set, i.e. they measure the amount of additional data samples which would have to be added to the training set to achieve better or generally acceptable performance. Hence, as the same training set was used for all the considered ANN paradigms, the performance of these paradigms can be assessed as inversely proportional to the value of their average (or RMS) percentile errors on the training set. It can be seen at each of the top graphs in Figure 36, that the best performance was achieved with the BPN, the MNN was the next, the RBFN was the still next, and the GRNN was the last.

It is also somewhat interesting that the overall structure of the errors shown in Figure 36 seems to be rather similar for all the data sets and all the ANN paradigms considered (with the exception of the MNN on the testing set), because the RMS errors show the same pattern as the average errors, while having slightly higher values (these higher values of the RMS errors simply indicate that the errors for some of the samples in the sets were considerably higher than the average error on these sets). Nevertheless, this certainly does not

mean that all the ANNs considered had problems with mapping of the same samples in the data sets (as it can be verified, e.g., by comparing Figure-14 top graph, Figure 24, Figure 28, and Figure 32).

Nevertheless, a real challenge for the ANNs in the application of interest is their performance while computing the RAWs of the model basic events because it particularly clearly reflects any deviations of the computed core-damage frequencies from their correct values. Using again the best-performing ANN trained for each of the considered paradigms, the RAWs for the individual basic events of the test model were computed for two different configurations: the configuration with only the event number one guaranteed, and the configuration with the events number one and number four guaranteed. The RAWs for the individual basic events computed from the results produced by the individual ANN paradigms for these configurations are shown in Figure 37 and Figure 38, respectively (the top graph in each Figure shows the correct RAW values for the given configuration). It can be readily seen that none of the selected ANNs was able to produce the results allowing computation of the sufficiently correct RAW values (possibly with an exception of the BPN results for the configuration with the events numbers one and four guaranteed).

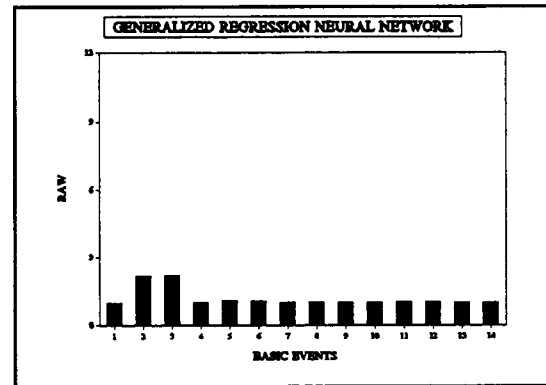
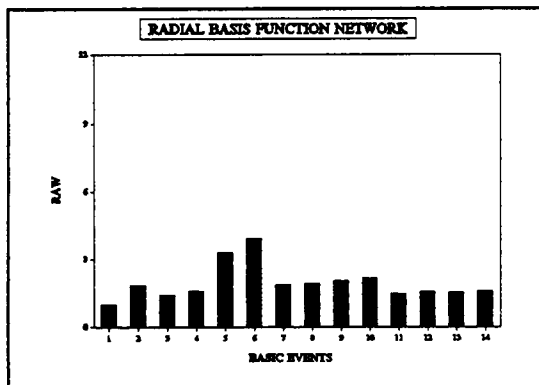
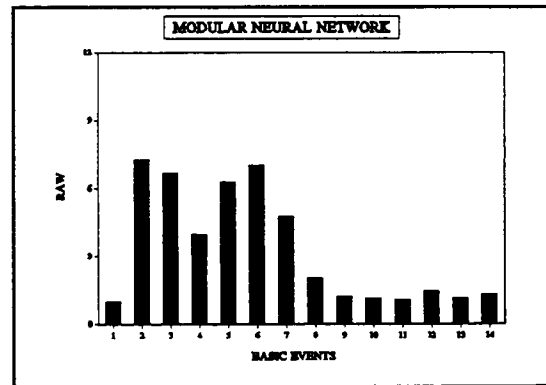
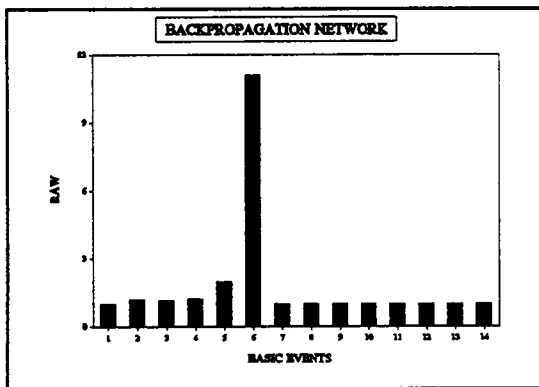
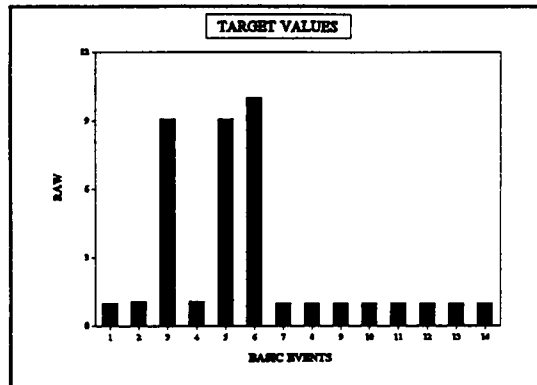


Figure 37 - RAWs of individual basic events as computed by different ANNs for the configuration with only the event number one guaranteed

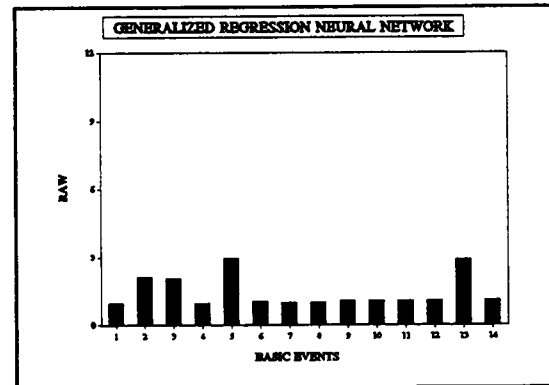
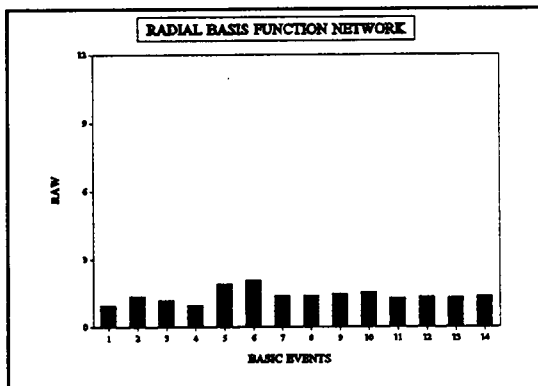
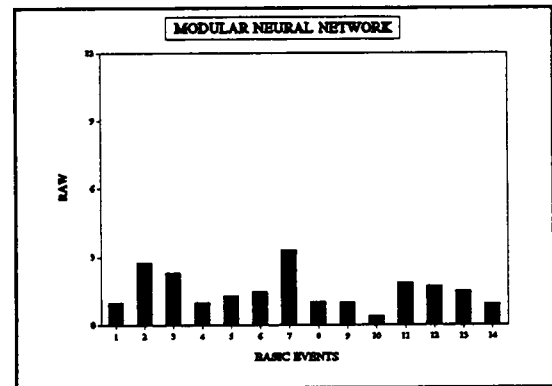
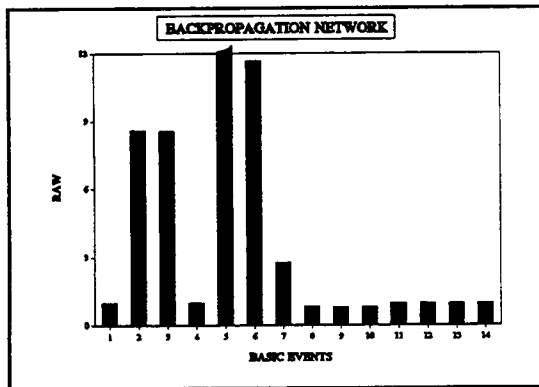
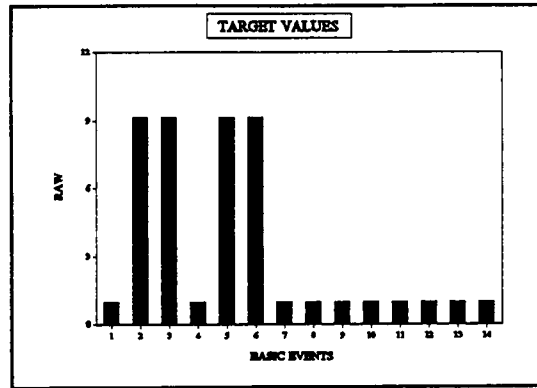


Figure 38 - RAWs of individual basic events as computed by different ANNs for the configuration with the events number one and number four guaranteed

5.4.2. Reasons for the Difficulties Encountered by Individual ANN Paradigms

The quantification of the small test PSA model appeared to be a quite easy task for the PSA code employed (i.e., the Tree-Master code - [62]). The general reason for the failure of the ANNs to emulate this quantification successfully consisted clearly in an insufficiently representative coverage of the input space by the samples comprising the data set used for the training of the ANNs. This is documented by some quite big errors of the core-damage frequencies as computed for various test-model configurations by the trained ANNs representing the considered paradigms (see Figures 14, 16-18, 24-35). In order to achieve better performance of the ANNs, the number of samples included in the training set would have to be increased considerably.

In case of the BPN, the proper extension of the training set alone could probably be sufficient for achieving its better performance. The quite good success of the BPN in learning the training set actually used for its training (see Figure 14) demonstrated that the BPN would likely be capable of learning the PSA model quantification with an arbitrary

precision*. Nevertheless, the training set actually used comprised already as many as 119 input-output vector pairs. To ensure the required precision of the computed core-damage frequencies (i.e. within the error margin of ten percent) for all possible configurations of the basic-event probabilities, it would be necessary to use the training set with several hundred of the input-output vectors pairs. Though it would certainly be possible to accomplish this task, it would require prohibitively many trial calculations with the Tree-Master code and would be rather impractical.

Concerning the MNN, the reference [52] suggests that for any problem solvable by the ANNs using the supervised training, the MNN performs at least as well as the BPN. Nevertheless, the results obtained for the application of concern demonstrate that this statement is incorrect**. It appears that the performance of the MNN can actually be worse than the performance of the BPN. A probable general reason for this phenomena is that it is not always possible to divide the original problem into several simple sub-problems which can be efficiently solved with "lower-level BPNs" (i.e. the

* It is meant here, that the BPN would be capable to achieve this result quite routinely within a reasonable time. Of course, according to the Kolmogorov Theorem the BPN is definitely able to learn perfectly any function, but this theory does not say that training of such a BPN is always practically feasible [72].

** Of course, providing that the trivial case, when the expert networks used in the MNN have the same structure as the successful BPN and the MNN thus consequently degenerates to the single BPN (cf. section 4.2.3), is not counted as an equal performance of the MNN and the BPN.

BPNs with a somewhat simpler structure, such as the BPNs using less hidden layers or "less nonlinear" activation functions, than the original BPN), which would form the MNN expert networks. Specifically for the problem of concern, the problem appeared to be solvable with a basic type of the BPN (i.e. the three-layer BPN using the sigmoid activation functions). Thus, to prevent degeneration of the MNN during training, it was necessary not to use any activation function in the MNN expert networks. However, the unsatisfactory performance of the trained MNN showed that it was not possible to partition the problem correctly enough into the sub-problems solvable with the ANNs so simple (the input space of the individual sub-problems would have to consist of the areas separable by linear decision boundaries). In any case, this result shows that there is actually no hope that the MNN could ever reach better performance than the BPN for the application of concern.

Regarding the RBFN, there is a general warning given in reference [70], that for the function approximation applications an efficient solution of the problem may require the use of unbounded activation functions. The RBFN, however, uses the radial activation functions which are always bounded. The performance of the trained RBFN on the test problem, highly unsatisfactory in comparison with the BPN, showed that the warning precisely applies for the application of interest.

In fact, the RBFN is mainly used for the pattern recognition applications and its successful utilization in the function approximation applications is rather rare. The reason for this is a strong dependence of the RBFN on the structure of the input space which has to be to a certain extent clustered. As the widths of the RBFN radial basis functions are normally set in a way that prevents overlapping of the areas of the strong activation of the individual pattern units, it is very difficult for the RBFN to generalize well for the input vectors not used during its training (cf. section 4.2.4). It appears, that this was precisely the trouble for the test problem. In any case, it is clear that even a considerable extension of the training set would not possibly help to achieve an acceptable performance of the RBFN for the application of concern.

The GRNN basically performs a statistical analysis of the data comprising its training set while using a technique called regression analysis, which estimates the most probable hyperplane to fit some continuously varying parameters in the analyzed data. The training of the GRNN is accomplished during a single pass through this analyzed data. Thus, for training to be successful, the data must represent all the possible cases especially well. If there are any cases missing in the data, the GRNN would not only fail to learn the correct mapping for the given part of the input space, but it

would very likely produce really poor estimates of the output vectors for all the input vectors from this input space area. The GRNN is only capable of generalization between the centers of the radial basis functions that are reasonably close to one another, and it produces quite misleading results for any outlying input vectors. The insufficiently representative coverage of the input space provided by the samples comprising the training data set used for training of the ANNs on the test problem, which creates the problems in training of all the ANNs considered, has thus especially damaging effect on the performance of the GRNN. The GRNN, in general, has the potential to achieve an arbitrarily precise approximation of any function, and it would be without any doubt capable to learn the PSA model quantification within the desired margin of errors. Nevertheless, to achieve this, it would require an extremely large number of samples included in the training set, the samples which are very difficult to attain. In addition, to evaluate the large input space correctly, the GRNN would also need to have many pattern units. But, among the ANNs the GRNN represents a single most computer-time-intensive paradigm and the expected large number of necessary pattern units would likely slow its operation down to a level where it would not provide any considerable improvement compared to the original PSA code computation. Thus, it is clear that also the GRNN is not practically usable for the application of concern.

5.4.3. General Reason for the Observed ANN Difficulties

Behind all the specific difficulties encountered by the individual considered ANN paradigms there is, actually, a single common cause which concerns the function approximation applications in general. The literature dealing with the ANNs routinely claims that many of the ANN paradigms can be successfully used for function approximation - e.g. [46-61]. Though this statement is certainly not incorrect, the experience with the application investigated in this study shows that it is somewhat oversimplified or incomplete.

The ANNs to be used for function approximation have to utilize the supervised training. Their training is thus based on an associative learning, i.e. they are learning to associate individual input vectors with the output vectors corresponding to them. During the training, the ANNs are not learning to perform some logical operations on their inputs, but rather act as simple pattern associators or classifiers [75]. Hence, in this respect, there is no substantial difference between the pattern recognition (classification) and the function approximation applications. Yet, from a practical point of view there is an important difference present: the output space for the function approximation applications is supposed to be continuous, while for the pattern recognition application it consists of a number of

discrete patterns only. The ANN trained for the function approximation application is still learning to associate individual input vectors with an inevitably limited number of classes represented by its outputs. However, this number of classes is usually much higher, while generally depending on the range of the individual required outputs, and on the allowed error of these outputs.

Naturally, the difference described above is also reflected in difficulties encountered in training. While the ANNs for the pattern recognition applications can be relatively easily trained with examples of all possible output patterns, the same approach cannot be obviously used when training the ANNs for function approximation applications. Therefore, to be able to succeed in function approximation applications, the ANNs have to be able to generalize properly the continuous relation between the input and output space, while trained with an inevitably limited set of specific examples of this relation. This is generally a very difficult task and the problems become overwhelming when the output space (even when represented by a single value) is rather broad and the allowed error of the produced output vector is rather low.

Hence, it can be concluded that the ANN paradigms currently known are inherently suitable primarily for the applications requiring classification of the input space into a discretized

limited output space. When the ANN is to be used for a function approximation application (i.e. classification into the continuous output space), it is necessary to realize that the algorithms used for its training inevitably involve an "uncertainty principle". There is no practically feasible way of a successful training of the ANN which would achieve both production of a relatively broad range of the required output values, and maintenance of a relatively low error of these values along this range. Theoretically of course, both these capabilities can be achieved simultaneously and the considered ANNs can approximate an arbitrary function with an arbitrary precision (as it follows from the Kolmogorov Theorem [72,76]). In practice, however, the successful training of such an ANN is not feasible. The ANNs can thus be expected to provide either of these capabilities alone, but only at the expense of the other.

Unfortunately, the problem of the application investigated in this study is that its solution can be considered successful only when both the conditions mentioned above are fulfilled simultaneously. The problem is thus inherently unsuitable for solving using the currently known ANNs.

5.5. Expected ANNs Performance on a Full-Scope PSA Model

The ANNs are generally known to perform better on more complex problems than on the simple ones. The reason for this phenomenon is that it typically becomes less easy for the ANN to become overtrained as the relation between the input and output space grows more and more complicated.

Unfortunately, it is rather unlikely that this phenomenon will be of any help for the application of concern because the major problem identified for it on the small-scale test problem consisted in the requirement to provide the ANN with the training set comprising relatively many examples of the relation between the input and output space. As the test problem was designed with an intention to model all important features of full-scope PSA models (cf. section 5.1.), it can be expected that the main difference of the full-scope PSA model comparing to the test PSA model will consist in an increase of the number of basic events of the model (it will be certainly hundreds of them, but it may be also easily over a thousand). Yet, a majority of these basic events represent independent (input) variables of the function to be approximated by using the trained ANN (cf. section 5.2.2) and each of these input variables will thus have to be represented in the ANNs by a corresponding input unit. In addition, it is clear that the expected increase of the number of these basic

events will not result in an only a linear increase of the number of possible configurations of their probabilities, but that this increase will be rather factorial. Thus, it is to be expected that the problems encountered when training the ANNs for the small test PSA model would not be reduced for the full-scope PSA models, but would be, in fact, even enlarged.

Besides that, considering an important role of the RAWs in operation of the risk monitor (see section 3.3.3.3), the observed failure of the ANNs to correctly compute their values for the test problem deals a very serious blow to the initial assumption that the ANNs are capable of successful emulation of calculations done by the PSA codes. There is no reason to believe that performance of the ANNs in this task would be more successful when dealing with a full-scope PSA model.

6. CONCLUSIONS AND RECOMMENDATIONS

On the basis of the experience gained when exploring the performance of selected ANN paradigms on the test problem it can be concluded that only the BPN showed some potential to handle the task. However, the expected volume of samples of the input-output space relation necessary for successful training of the BPN makes even its utilization very impractical at this time. Thus, although a successful utilization of the ANNs in the application of concern is not impossible, it has to be concluded that it is not practically feasible in the present state of knowledge of the ANNs.

It is recommended that before undertaking to utilize the ANNs for emulation of the PSA model quantification using the PSA code, the investigator should explore methods by which the following issues can be successfully handled:

1. The associative learning utilized by the currently known ANN paradigms usable for function approximation does not appear to be suitable for the tasks requiring production of the ANN output values covering a relatively broad range and involving a relatively small error, such as the approximation of the PSA model quantification. It would be necessary to find a new ANN paradigm capable of coping with this task. For instance, the probabilistic neural networks

implement Bayes optimal classification scheme and allow thus its effective practical utilization, which was prohibitively computer time and memory intensive before [54]. In a similar manner a new ANN paradigm might be developed to implement effective function approximation.

2. The PSA quantification involves basically the multiplication of failure probabilities of individual basic events of the PSA model, and the final summation of the obtained multiples. It is rather difficult to train an ANN to perform this operation. It might be possible to find transformations capable of facilitating this task in a similar manner as the logarithmic transformation converts multiplication into a summation.
3. The PSA model represents a highly nonlinear function involving discontinuous course associated with logical relations of influences of individual basic events of the PSA model on the computed core-damage frequency. This complex nature of the PSA models would cause problems with proper verification and validation of the trained ANN. It would be necessary to develop a well-advised strategy for accomplishing the dependable verification and validation.

REFERENCES

REFERENCES

- [1] U.S. NUCLEAR REGULATORY COMMISSION: Generic Letter 88-20, Individual Plant Examination for Severe Accident Vulnerabilities; US NRC, Washington, DC, USA, November 23, 1988.
- [2] J.W. Hickman et al.: PRA Procedures Guide - A Guide to the Performance of Probabilistic Risk Assessment for Nuclear Power Plants; Report NUREG/CR-2300, US NRC, Washington, DC, USA, January 1983.
- [3] INTERNATIONAL ATOMIC ENERGY AGENCY: Procedures for Conducting Probabilistic Safety Assessments of Nuclear Power Plants (Level 1) - A Safety Practice; Safety Series Report No. 50-P-4, IAEA, Vienna, Austria, 1992.
- [4] U.S. NUCLEAR REGULATORY COMMISSION: Reactor Safety Study - An Assessment of Accident Risks in U.S. Commercial Nuclear Power Plants; Report WASH-1400 (NUREG-75/014), US NRC, Washington, DC, USA, October 1975.
- [5] J.L. vonHerrman, P.J. Wood, J.P. Gaertner, I.B. Wall: The Practical Application of PRA - An Evaluation of Utility Experience and USNRC Perspectives; *Reliability Engineering and Systems Safety*, 24 (1989), pp. 167-198.
- [6] U.S. NUCLEAR REGULATORY COMMISSION: Severe Accident Risks - An Assessment for Five US Nuclear Power Plants - Final Summary Report; NUREG-1150, US NRC, Washington, DC, USA, December 1990.
- [7] K.N. Fleming: Lessons Learned in Applying PSA Technology to Diverse Risk Management Applications; In: Report of the IAEA Technical Committee Meeting "Advances in Reliability Analysis and Probabilistic Safety Assessment", Budapest, Hungary, September 7-11, 1992; Report IAEA-J4-TC-606.3, IAEA, Vienna, Austria, 1993.

- [8] S. Hirschberg: Prospects for Probabilistic Safety Assessment; *Nuclear Safety*, 33 (1992), pp. 365-380.
- [9] S. Hirschberg, G. Johanson, P.K. Samanta: Requirements and Capabilities of PSA for Safety Management Applications; In: Proceedings of the "International Topical Meeting on Probabilistic Safety Assessment - PSA'93", Clearwater Beach, FL, USA, January 26-29, 1993.
- [10] A.A. Dykes et al.: U.S. Nuclear Industry Efforts in Utilizing PSA for Technical Specifications Changes; In: PSA Applications - Preliminary Report of the IAEA Technical Committee Meeting on "Procedures for Use of PSA for Optimizing NPP Operational Limits and Conditions", Barcelona, Spain, September 20-23, 1993; Report IAEA-J4-TC-855, IAEA, Vienna, Austria, 1993.
- [11] G. Andre, J-P. Sursock, T. Leserman, R. Thierry: Methods for Applying PRA to Technical Specification Improvements; In: PSA Applications - Preliminary Report of the IAEA Technical Committee Meeting on "Procedures for Use of PSA for Optimizing NPP Operational Limits and Conditions", Barcelona, Spain, September 20-23, 1993; Report IAEA-J4-TC-855, IAEA, Vienna, Austria, 1993.
- [12] B. Tomic: Risk Based Optimization of Maintenance - Methods and Approaches; In: P. Kafka, J. Wolf (Eds.): Safety and Reliability Assessment - An Integral Approach; Proceedings of the "European Safety and Reliability Conference", Munich, Germany, May 10-12, 1993; Elsevier Science Publishers, Amsterdam, Holland, 1993.
- [13] R. Pahladsingh: Using PRA Techniques in Plant-Design, Plant-Operation, and Information Systems; In: Proceedings of the "Second ASME/JSME International Conference on Nuclear Engineering", San Francisco, CA, USA, March 21-24, 1993.
- [14] S.P. Kalra: On the Shutdown Risk Management in Nuclear Power Plants; In: Proceedings of the "Second ASME/JSME International Conference on Nuclear Engineering", San Francisco, CA, USA, March 21-24, 1993.

- [15] D. Currier: An IKBS Approach to Aid in Nuclear Accident Management; In: Proceedings of the "Second ASME/JSME International Conference on Nuclear Engineering", San Francisco, CA, USA, March 21-24, 1993.
- [16] R. Doremus: SAMSON - Severe Accident Management System Online Network; In: Proceedings of the "Second ASME/JSME International Conference on Nuclear Engineering", San Francisco, CA, USA, March 21-24, 1993.
- [17] INTERNATIONAL ATOMIC ENERGY AGENCY: PSA Applications - Preliminary Report of the IAEA Technical Committee Meeting on "Procedures for Use of PSA for Optimizing NPP Operational Limits and Conditions", Barcelona, Spain, September 20-23, 1993; Report IAEA-J4-TC-855, IAEA, Vienna, Austria, 1993.
- [18] INTERNATIONAL ATOMIC ENERGY AGENCY: Risk-Based Application of Nuclear Power Plant Technical Specification Improvements - Report of the Consultants Meeting, Vienna, Austria, August 31 - September 4, 1992; Report IAEA-J4-CS53/92, IAEA, Vienna, Austria, 1992.
- [19] P.K. Samanta, W.E. Vesely, I.S. Kim: Toward Risk-Based Control of Nuclear Power Plant Configurations; *Nuclear Engineering and Design*, 134 (1992), pp. 355-370.
- [20] J.Q. Kirkman, J.B. Fussel, D.J. Campbell: PRISIM at Arkansas Nuclear One, Unit 1 - Daily, in-Plant Use of PSA Information; *Reliability Engineering and Systems Safety*, 22 (1988), pp. 441-454.
- [21] B.E. Horne: The Use of Probabilistic Safety Analysis Methods for Planning the Maintenance and Testing Unavailabilities of Essential Systems at Heysham 2 AGR Power Plant; In: Report of the Technical Committee Meeting on "Use of Probabilistic Safety Assessment to Evaluate Nuclear Power Plant Technical Specifications", Vienna, Austria, June 18-22, 1990; Report IAEA-TECDOC-599, IAEA, Vienna, Austria, April 1991.

- [22] N. Mitchison, A. Poucet, J-P. Nordvik, M. Wilikens: A Real-Time Expert System for Risk Prevention - The FORMENTOR Project; In: P. Kafka, J. Wolf (Eds.): Safety and Reliability Assessment - An Integral Approach; Proceedings of the "European Safety and Reliability Conference", Munich, Germany, May 10-12, 1993; Elsevier Science Publishers, Amsterdam, Holland, 1993.
- [23] P. Kafka, H. Kunitz: The Use of PC Based Probabilistic Safety Models; In: Proceedings of an International Symposium "Use of Probabilistic Safety Assessment for Operational Safety - PSA'91", Vienna, Austria, June 3-7, 1991; Report IAEA-SM-321, IAEA, Vienna, Austria, 1992.
- [24] T.A. Morgan: Development and Use of the San Onofre Safety Monitor; In: PSA Applications - Preliminary Report of the IAEA Technical Committee Meeting on "Procedures for Use of PSA for Optimizing NPP Operational Limits and Conditions", Barcelona, Spain, September 20-23, 1993; Report IAEA-J4-TC-855, IAEA, Vienna, Austria, 1993.
- [25] W.J. Puglia, D. Galagher, P. Amico, B. Atefi: Risk-Based Technical Specifications - Development and Application of an Approach to the Generation of a Plant Specific Real-Time Risk Model; Report NUREG/CR-5925, US NRC, Washington, DC, USA, October 1992.
- [26] W.J. Puglia, B. Atefi, D.W. Gallagher: PSA for on-Line Maintenance Planning and Configuration Management; *Nuclear Plant Journal*, May-June 1993, pp. 78-81.
- [27] P. Kafka: Living PSA/Risk Monitor - Current Developments; In: Report of the IAEA Technical Committee Meeting "Advances in Reliability Analysis and Probabilistic Safety Assessment", Budapest, Hungary, September 7-11, 1992; Report IAEA-J4-TC-606.3, IAEA, Vienna, Austria, 1993.

- [28] J. Rumpf: SAIS Application to VVER Reactors;
In: Report of the IAEA Technical Committee Meeting
"Advances in Reliability Analysis and Probabilistic
Safety Assessment", Budapest, Hungary, September 7-11,
1992; Report IAEA-J4-TC-606.3, IAEA, Vienna, Austria,
1993.

- [29] H.P. Balfanz, H.P. Berg: The German Living PSA Tool -
SAIS; *Nuclear Plant Journal*, May-June 1993, pp. 86-90.

- [30] G.R. Moir: On-Line Use of PSA for Risk-Based
Configuration Control; In: PSA Applications -
Preliminary Report of the IAEA Technical Committee
Meeting on "Procedures for Use of PSA for Optimizing
NPP Operational Limits and Conditions", Barcelona,
Spain, September 20-23, 1993; Report IAEA-J4-TC-855,
IAEA, Vienna, Austria, 1993.

- [31] R. Nakai, Y. Kani: LIPSAS - A Living PSA System for an
LMFBR; In: Proceedings of an International Symposium
"Use of Probabilistic Safety Assessment for Operational
Safety - PSA'91", Vienna, Austria, June 3-7, 1991;
Report IAEA-SM-321, IAEA, Vienna, Austria, 1992.

- [32] T.W. Kim, S.H. Han, K.J. Yoo: Optimal Success Path
Generation System Based on Integrated Reliability
Rules; *Nuclear Technology*, 86 (1989), pp. 35-39.

- [33] K. Parsaye, M. Chignell: Expert Systems for Experts;
John Wiley & Sons, New York, NY, USA, 1988.

- [34] K. Pedersen: Expert Systems Programming - Practical
Techniques for Rule-Based Systems; John Wiley & Sons,
New York, NY, USA, 1989.

- [35] C.T. Kitzmiller, J.S. Kowalik: Coupling Symbolic and
Numeric Computing in Knowledge-Based Systems; *AI
Magazine*, Vol. 8 - No. 2 (Summer 1987), pp. 8-90.

- [36] B. Atefi, D.W. Gallagher: Feasibility Assessment of a
Risk-Based Approach to Technical Specifications; Report
NUREG/CR-5742, US NRC, Washington, DC, USA, May 1991.

- [37] W.E. Vesely: A Time-Dependent Methodology for Fault Tree Evaluation; *Nuclear Engineering and Design*, 13 (1970), pp. 337-360.
- [38] T. Mankamo, M. Kosonen: Continued Plant Operation Versus Shutdown in Failure Situations of Standby Safety Systems; In: Report of the Technical Committee Meeting on "Use of Probabilistic Safety Assessment to Evaluate Nuclear Power Plant Technical Specifications", Vienna, Austria, June 18-22, 1990; Report IAEA-TECDOC-599, IAEA, Vienna, Austria, April 1991.
- [39] D. Hamilton, K. Kelley, C. Culbert: State-of-the-Practice in Knowledge-Based System Verification and Validation; *Expert Systems with Applications*, 3 (1991), pp. 403-410.
- [40] S. Lee, R.M. O'Keefe: Developing a Strategy for Expert System Verification and Validation; *IEEE Transactions on Systems, Man, and Cybernetics*, 24 (1994), pp. 643-655.
- [41] W.R. Adrion, M.A. Branstad, J.C. Cherniavsky: Validation, Verification, and Testing of Computer Software; *Computing Surveys*, 14 (1982), pp. 159-192.
- [42] S.H. Kirani, I.A. Zualkerman, W.-T. Tsai: Evaluation of Expert System Testing Methods; *Communications of the ACM*, Vol. 37 - No. 11 (November 1994), pp. 71-81.
- [43] L. Adelman: Experiments, Quasi-Experiments, and Case Studies - A Review of Empirical Methods for Evaluating Decision Support Systems; *IEEE Transactions on Systems, Man, and Cybernetics*, 21 (1991), pp. 293-301.
- [44] D.V. Hillman: Integrating Neural Nets and Expert Systems; *AI Expert*, Vol. 5 - No. 6 (June 1990), pp. 54-59.
- [45] Z. Guo, R.E. Uhrig: Use of a Neural Network in an Expert System to Enhance Nuclear Power Plant Safety, *Transactions of the American Nuclear Society*, 61 (1990), pp. 215-216.

- [46] R.P. Lippman: An Introduction to Computing with Neural Nets; *IEEE ASSP Magazine*, Vol. 4 - No. 2 (April 1987), pp. 4-22.
- [47] P.D. Wasserman: Neural Computing - Theory and Practice; Van Nostrand Reinhold, New York, NY, USA, 1989.
- [48] J.E. Dayhoff: Neural Network Architectures - An Introduction; Van Nostrand Reinhold, New York, NY, USA, 1990.
- [49] A.J. Maren et al.: Handbook of Neural Computing Applications; Academic Press, San Diego, CA, USA, 1990.
- [50] M. Caudill: Avoiding the Great Back-Propagation Trap; *AI Expert*, Vol. 6 - No. 7 (July 1991), pp. 29-35.
- [51] D.E. Rumelhart, J.L. McClelland: Parallel Distributed processing - Explorations in the Microstructure of Cognition - Foundations, Volume 1; MIT Press, Cambridge, MA, USA, 1986.
- [52] NEURALWARE: Neural Computing - A Technology Handbook for Professional II/Plus and NeuralWorks Explorer; NeuralWare, Pittsburgh, PA, USA, 1993.
- [53] Y.-H. Pao: Functional Link Nets - Removing Hidden Layers; *AI Expert*, Vol. 4 - No. 4 (April 1989), pp. 60-68.
- [54] T. Masters: Practical Neural Network Recipes in C++; Academic Press, San Diego, CA, USA, 1993.
- [55] R.A. Jacobs, M.I. Jordan, S.J. Nowlan, G.E. Hinton: Adaptive Mixtures of Local Experts; *Neural Computation*, 3 (1991), pp. 79-87.
- [56] S. Haykin: Neural Networks - A Comprehensive Foundation; Macmillan College Publishing Company, New York, NY, USA, 1994.

- [57] J. Moody, C.J. Darken: Fast Learning in Networks of Locally Tuned Processing Units; *Neural Computation*, 1 (1989), pp. 281-294.
- [58] D.R. Hush, B.G. Horne: Progress in Supervised Neural Networks; *IEEE Signal Processing Magazine*, Vol. 10 - No. 1 (January 1993), pp. 8-39.
- [59] P.D. Wasserman: Advanced Methods in Neural Computing; Van Nostrand Reinhold, New York, NY, USA, 1993.
- [60] D.F. Specht: A General Regression Neural Network; *IEEE Transactions on Neural Networks*, 2 (1991), pp. 568-576.
- [61] M. Caudill: GRNN and Bear It; *AI Expert*, Vol. 8 - No. 5 (May 1993), pp. 28-33.
- [62] A. Wild: Tree-Master - A Fault Tree Handling System Designed Specifically for Your Personal Computer; Users Manual.
- [63] H.E. Lambert: Measures of Importance of Events and Cut Sets in Fault Trees; In: R.E. Barlow, J.B. Fussel, N.D. Singpurwalla (Eds.): Reliability and Fault Tree Analysis - Theoretical and Applied Aspects of System Reliability and Safety Assessment; Proceedings of the "Conference on Reliability and Fault Tree Analysis", Berkeley, CA, USA, September 3-7, 1974; SIAM, Philadelphia, PA, USA, 1975.
- [64] W.E. Vesely, T.C. Davis: Two Measures of Risk Importance and Their Application; *Nuclear Technology*, 68 (1985), pp. 226-234.
- [65] D. Bailey, D. Thompson: Developing Neural-Network Applications; *AI Expert*, Vol. 5 - No. 9 (September 1990), pp. 34-41.
- [66] J. Lawrence: Data Preparation for a Neural Network; *AI Expert*, Vol. 6 - No. 11 (November 1991), pp. 34-41.

- [67] T. Crooks: Care and Feeding of Neural Networks; *AI Expert*, Vol. 7 - No. 7 (July 1992), pp. 36-41.
- [68] R. Stein: Preprocessing Data for Neural Networks; *AI Expert*, Vol. 8 - No. 3 (March 1993), pp. 32-37.
- [69] J.A. Anderson, K.T. Spoehr, D.J. Bennett: A Study in Numerical Perversity: Teaching Arithmetic to a Neural network; In: D.S. Levine, M. Aparicio IV (Eds.): *Neural Networks for Knowledge Representation and Inference*; Lawrence Erlbaum Ass., Hillsdale, NJ, USA, 1994.
- [70] NEURALWARE: Reference Guide - Software Reference for Professional II/Plus and NeuralWorks Explorer; NeuralWare, Pittsburgh, PA, USA, 1993.
- [71] M. Caudill: Neural Network Training Tips and Techniques; *AI Expert*, Vol. 6 - No. 1 (January 1991), pp. 56-61.
- [72] R. Hecht-Nielsen: *Neurocomputing*; Addison-Wesley, Reading, MA, USA, 1990.
- [73] M. Caudill: Using Neural Nets, Part 1 - Representing Knowledge; *AI Expert*, Vol. 4 - No. 12 (December 1989), pp. 34-41.
- [74] M. Caudill: Neural Networks Primer, Part 1; *AI Expert*, Vol. 2 - No. 12 (December 1987), pp. 46-52.
- [75] J.A. Anderson: Data Representation in Neural Networks; *AI Expert*, Vol. 5 - No. 6 (June 1990), pp. 30-37.
- [76] J. Wray, G.G.R. Green: Neural Networks, Approximation Theory, and Finite Precision Computation; *Neural Networks*, 8 (1995), pp. 31-37.

Vita

Vaclav Hojny was born in Havlickuv Brod in former Czechoslovakia (now the Czech Republic) on August 26, 1956. In September 1975 he entered the Czech Technical University of Prague, Czechoslovakia where he received a Master of Science degree in Mechanical Engineering in July 1980.

In August 1980 he joined the staff of the Nuclear Research Institute at Rez near Prague, Czechoslovakia and worked there as researcher and senior researcher till June 1993. He worked in the field of nuclear safety performing safety reliability analyses and Probabilistic Safety Assessment of Czechoslovak nuclear power plants.

In August 1993 he entered the University of Tennessee at Knoxville accepting a research assistantship. He received his Master of Science degree in Nuclear Engineering in August 1995.