AGARD-LS-210

AGARD

ADVISORY GROUP FOR AEROSPACE RESEARCH & DEVELOPMENT
7 RUE ANCELLE, 92200 NEUILLY-SUR-SEINE, FRANCE

AGARD LECTURE SERIES 210

Advances in Soft-Computing Technologies and Application in Mission Systems

(les Avancées des technologies du calcul symbolique et les applications aux systèmes numériques de gestion de la mission)

The material in this publication was assembled to support a Lecture Series under the sponsorship of the Mission Systems Panel and the Consultant and Exchange Programme of AGARD presented on 17-18 September 1997 in North York, Canada; 22-23 September 1997 in Amsterdam, The Netherlands; 25-26 September 1997 in Madrid, Spain; and 6-7 October 1997 in Ankara, Turkey.

NORTH ATLANTIC TREATY ORGANIZATION

Published September 1997

Distribution and Availability on Back Cover

AGARD-LS-210

AGARD

ADVISORY GROUP FOR AEROSPACE RESEARCH & DEVELOPMENT
7 RUE ANCELLE, 92200 NEUILLY-SUR-SEINE, FRANCE

AGARD LECTURE SERIES 210

Advances in Soft-Computing Technologies and Application in Mission Systems

(les Avancées des technologies du calcul symbolique et les applications aux systèmes numériques de gestion de la mission)

The material in this publication was assembled to support a Lecture Series under the sponsorship of the Mission Systems Panel and the Consultant and Exchange Programme of AGARD presented on 17-18 September 1997 in North York, Canada; 22-23 September 1997 in Amsterdam, The Netherlands; 25-26 September 1997 in Madrid, Spain; and 6-7 October 1997 in Ankara, Turkey.

North Atlantic Treaty Organization
Organisation du Traité de l'Atlantique Nord

The Mission of AGARD

According to its Charter, the mission of AGARD is to bring together the leading personalities of the NATO nations in the fields of science and technology relating to aerospace for the following purposes:

- Recommending effective ways for the member nations to use their research and development capabilities for the common benefit of the NATO community;
- Providing scientific and technical advice and assistance to the Military Committee in the field of aerospace research and development (with particular regard to its military application);
- Continuously stimulating advances in the aerospace sciences relevant to strengthening the common defence posture;
- Improving the co-operation among member nations in aerospace research and development;
- Exchange of scientific and technical information;
- Providing assistance to member nations for the purpose of increasing their scientific and technical potential;
- Rendering scientific and technical assistance, as requested, to other NATO bodies and to member nations in connection with research and development problems in the aerospace field.

The highest authority within AGARD is the National Delegates Board consisting of officially appointed senior representatives from each member nation. The mission of AGARD is carried out through the Panels which are composed of experts appointed by the National Delegates, the Consultant and Exchange Programme and the Aerospace Applications Studies Programme. The results of AGARD work are reported to the member nations and the NATO Authorities through the AGARD series of publications of which this is one.

Participation in AGARD activities is by invitation only and is normally limited to citizens of the NATO nations.

Published September 1997

ISBN 92-836-1061-X

Printed by Canada Communication Group Inc.
(A St. Joseph Corporation Company)
45 Sacré-Cœur Blvd., Hull (Québec), Canada K1A 0S7

Advances in Soft-Computing Technologies and Application in Mission Systems

(AGARD LS-210)

Executive Summary

There is a paradigmatic complementary shift from symbolic AIKB techniques to so called Soft Computing technologies. The new paradigm is based on modelling the unconscious, cognitive and reflexive function of the biological brain.

In contrast to conventional Hard Computing, Soft Computing addresses the pervasive imprecision of the real world. This is accomplished by consideration of the tolerances for imprecision, uncertainty and partial truth to achieve tractable, robust and low cost solutions for complex problems.

Important related computing methodologies and technologies include among others fuzzy logic, neuro-computing, evolutionary and genetic algorithms. It becomes increasingly evident that combinations of these technologies are advantageous in many applications. A viable step towards intelligent machines can be expected that offer autonomous knowledge acquisition and processing self-organization and structuring, as well as associative rule generation for goal oriented behaviour in rarely predictable scenarios.

Soft Computing has the potential for unprecedented functional and operational advantages. As an important enabling technology it will be more or less part of any Mission System and many subsystems.

There is a potential for considerable dual-use and spin-in opportunities from civil information technology state-of-the-art.

The Lecture Series covers fundamental aspects as well as application issues, it addresses decision makers and those who develop, test, deliver, certify and apply corresponding systems or subsystems.

This Lecture Series, sponsored by the Mission Systems Panel of AGARD, has been implemented by the Consultant and Exchange Programme.

Les avancées des technologies du calcul symbolique et les applications aux systèmes numériques de gestion de la mission

(AGARD LS-210)

Synthèse

Les techniques symboliques AIKB sont en train d'évoluer de façon complémentaire et paradigmatique vers les technologies dites "bioinformatiques". Le nouveau paradigme est basé sur la modélisation des fonctions inconscientes, cognitives et réflexives du cerveau biologique.

Contrairement à l'informatique classique, la bioinformatique aborde le problème de l'imprécision omniprésente du monde réel. Ceci implique la considération des tolérances à appliquer à l'imprécision, à l'incertitude et à la vérité partielle, afin de trouver des solutions souples, robustes et de coût modique aux problèmes complexes qui se posent.

Parmi les technologies et méthodologies informatiques pertinentes l'on peut citer la logique floue, la neuroinformatique et les algorithmes évolutifs et génétiques. Il est de plus en plus évident que des amalgames de ces technologies seraient avantageux pour bon nombre d'applications. Il y a lieu de s'attendre à des initiatives valides, visant des machines intelligentes, qui permettront l'acquisition et l'exploitation autonome des connaissances, l'organisation et la structuration automatiques, ainsi que la génération associative de règles pour le comportement orienté vers un but dans des scénarios qui seront rarement prévisibles.

La bioinformatique a le potentiel de procurer des avantages fonctionnels et opérationnels sans précédent. En tant que technologie habilitante elle fera plus ou moins partie de tout nouveau système de conduite de mission et de bon nombre de sous-systèmes.

Il existe des possibilités considérables de dualité et de retombées venant des nouvelles technologies de l'information du secteur civil.

Ce cycle de conférences couvre les aspects fondamentaux de la question, ainsi que les applications. Il s'adresse aux décideurs et à tous ceux qui développent, testent, livrent et homologuent les systèmes et sous systèmes correspondants.

Le Cycle de Conférences No. 210 de l'AGARD a été organisé par le Panel Systèmes de conduite de mission, sous l'égide du Programme des consultants et d'échanges.

Contents

	Page
Executive Summary	iii
Synthèse	iv
List of Authors/Speakers	vi
	Reference
Introduction and Overview by Dr. U. Krogmann	**I**
Techniques for Computational and Machine Intelligence SOFT COMPUTING by Dr. U. Krogmann	**1**
Introduction to Neural Networks Basic Theory and Application Potential for Mission Systems by Professor D.J. Collins	**2**
Fundamentals of Fuzzy Logic, fuzzy inference and fuzzy control by Mr. B. Bouchon-Meunier	**3**
Paper 4 withdrawn	
Hybrid Architectures for Intelligent Systems, Learning Inference Systems by Mr. B. Bouchon-Meunier	**5**
Dual Waveband Infra-Red Target Tracking and Acquisition System by Mr. M. Bernhardt, Ms. M. Welch and Dr. D.L. Toulson	**6**
Application of Neural Network to Reconfiguration of Damaged Aircraft by Professor D.J. Collins and Lt. Commander S. Dror	**7**
Paper 8 withdrawn	
Towards Autonomous Unmanned Systems by Dr. U. Krogmann	**9**

List of Authors/Speakers

Lecture Series Director: Dr U. KROGMANN
BGT
Bodenseewerk
Gerätetechnik
GmbH
Postfach 10 11 55
D-88641 Überlingen

Professor Daniel J. COLLINS
Department of Aeronautics & Astronautics
Naval Postgraduate School
699 Dyer Road (Rm 137)
Monterey, California 93943-5106
USA

Ms B. BOUCHON-MEUNIER
LIP6, CNRS-UPMC
Case 169, 4 place Jussieu
75252 Paris Cedex 05
France

Mr. M. BERNHARDT
SAS Dept
Rm 1026 Bldg A5
DERA
Farnborough
Hants GU14 OLX
UK

Co-Authors

Lt Commander Shahar DROR
P.O. Box 1609
REUT 71908
Israel

Ms Meryl WELCH
Room 1026
A5 Building
DERA Farnborough
Hampshire GU14OLX
UK

Dr. Darren TOULSON
Wheatstone Laboratory
King's College
Strand
London WC2R 2LS
UK

Introduction and Overview

U. Krogmann
Bodenseewerk Gerätetechnik GmbH
Postfach 10 11 55
D-88641 Überlingen

Tactical systems are implemented as Integrated Mission Systems (IMS) such as air and space defence systems. As shown in the lower part of Fig. 1, the key elements of an IMS are platforms with sensors and effectors, ground based components with communication, command and control etc.. Mission management constitutes the functional process within and among Integrated Mission Systems (Fig. 1). It will provide the capability for rapidly gathering, distributing and integrating large quantities of available information and will allow rapid strategic and tactical decision-making on the missions as well as carrying out the resulting actions from what has been decided and what is required for dealing effectively with these missions, in order to perform the functionalities appropriately in unpredictable and uncertain scenarios.

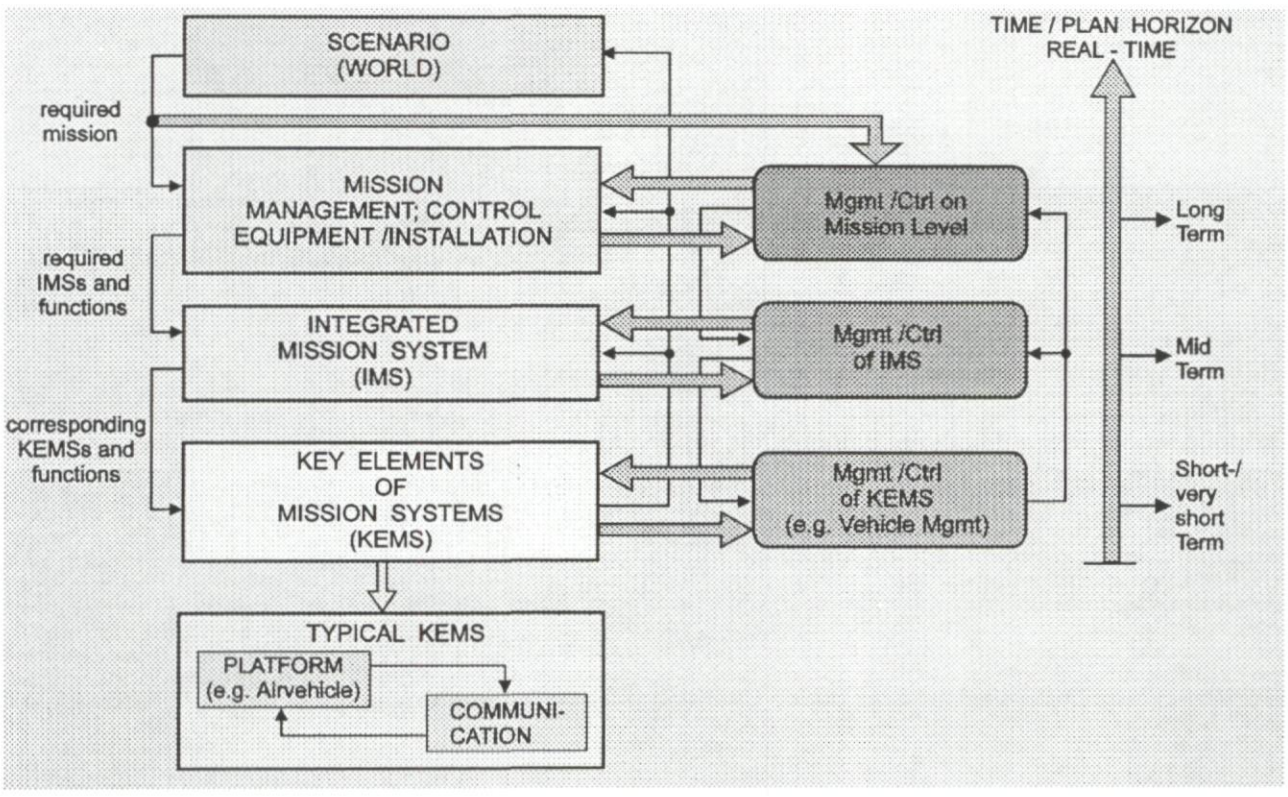


Figure 1: Mission system and mission management structure

Paper presented at the AGARD MSP Lecture Series on "Advances in Soft-Computing Technologies and Applications in Mission Systems", held in North York, Canada, 17-18 September 1997; Amsterdam, The Netherlands, 22-23 September 1997; Madrid, Spain, 25-26 September 1997; Ankara, Turkey, 6-7 October 1997, and published in LS-210.

Taking for example an airvehicle as a mission system element, Fig. 2 depicts how mission control interacts with the functional levels of the guidance and control (G.a.C.) process of the airvehicle. It can be seen that G.a.C. problems extend over several hierarchically structured levels and the communication functions between these levels. The represented interconnection of the different functional levels (scenario, mission, trajectory, airvehicle state) can be conceived of as a hierarchically structured control system. The objects on which G.a.C. functions are performed on the mentioned levels represent the control plants. Information processing by which actuation is generated from sensor information on all levels represents the controller functions which are often also called the recognize-act-cycle functions.

It requires functions such as recognizing and assessing the situation, defining action goals, generating optimum or favorable solutions, decision-making, planning and finally performing as well as monitoring of actions. Hence, behavior levels of mental capabilities such as skill, rule, knowledge based functions [1] can be assigned to the functional levels (Fig. 2).

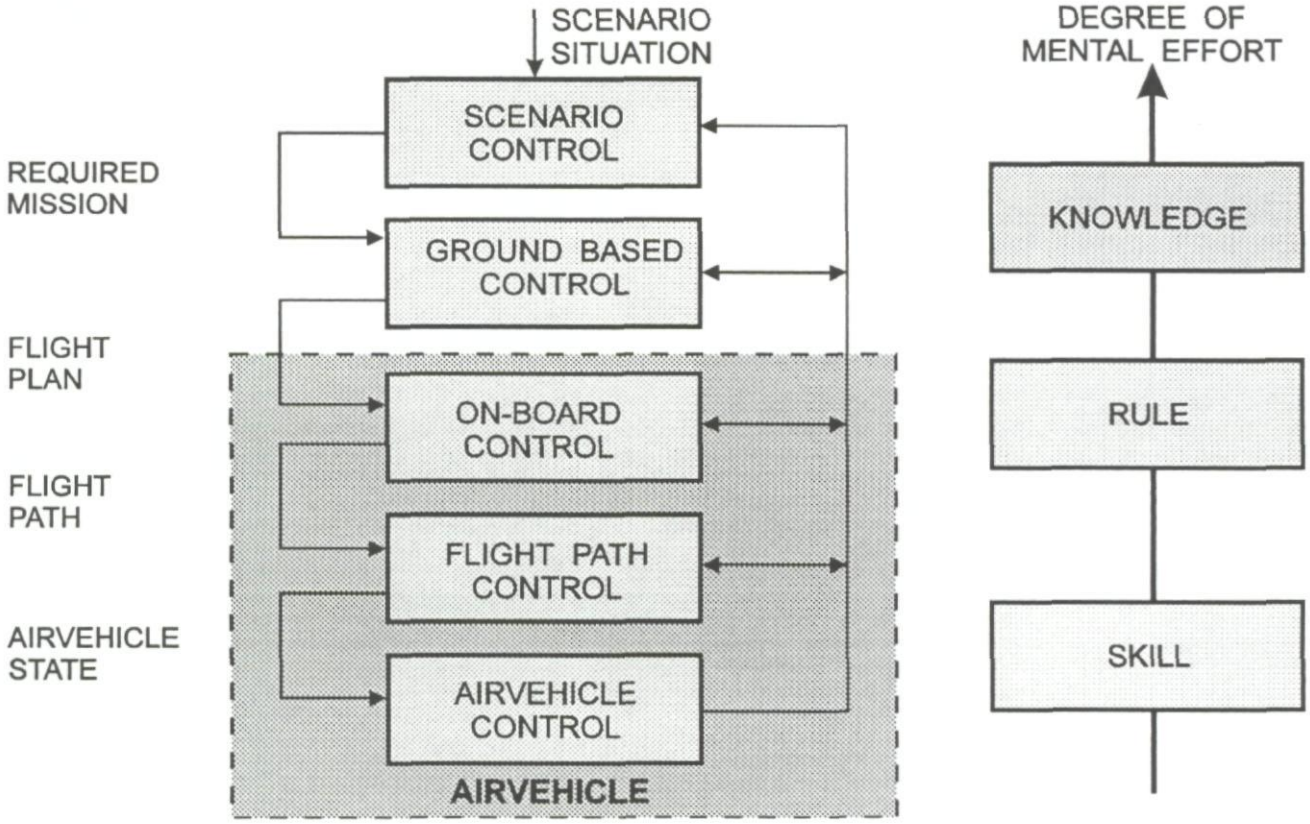


Figure 2: Cascaded airvehicle G.a.C. structure

For reasons of human limitations in more demanding dynamic scenarios and in the operation of complex, highly integrated systems, there is the neccessity for extended automation of these functions on higher levels such as trajectory control as well as mission management and control. Furthermore, the implementation of intelligent functions on lower levels such as the fusion and interpretation of sensor data, multifunctional use of sensor information and advanced nonlinear learning control becomes inevitable.

This is accomplished through the introduction of new computational and machine intelligence techniques (CMI). Moreover, CMI will be the most important prerequisite for less manned air operations extending as far as fully autonomous tactical platforms.

Although the development of hardware and software for computers of conventional v. Neumann architecture has continued for more than 20 years and the performance of today´s processors is 25.000 times better than in the 1970s, the dynamics of this development is going on as well.

However, there is a complementary shift from conventional computing techniques, including symbolic AI/KB techniques, to so-called soft computing technologies. The new paradigm is based on modelling the unconscious, cognitive and reflexive function of the biological brain. This is accomplished by massively parallel implementation in networks as compared to program/software based information processing in conventional sequential architectures.

In contrast to the conventional method, soft computing addresses the pervasive imprecision of the real world. This is obtained by consideration of the tolerances for imprecision, uncertainty and partial truth to achieve tractable, robust and low-cost solutions for complex problems.

Important related computing methodologies and technologies include among others fuzzy logic, neuro-computing, as well as evolutionary and genetic algorithms. With those a viable step towards intelligent machines can be expected that offer autonomous knowledge acquisition and processing, self-organization and structuring as well as associative rule generation for goal-oriented behavior in rarely predictable scenarios. The new techniques will yield computational and machine intelligence which offers the user the opportunity for cognitive automation of typical „recognition-act cycle" activities on various functional and operational levels.

The last two decades have witnessed a very strong growth of CMI techniques. These techniques have already been applied to a variety of problems to deliver efficient solutions to the benefits of the user. Certainly there are relationships between CMI and other fields such as those shown on top of Fig. 3. Moreover, numerous disciplines have contributed to the area of soft computing where some are mentioned at the bottom of Fig. 3.

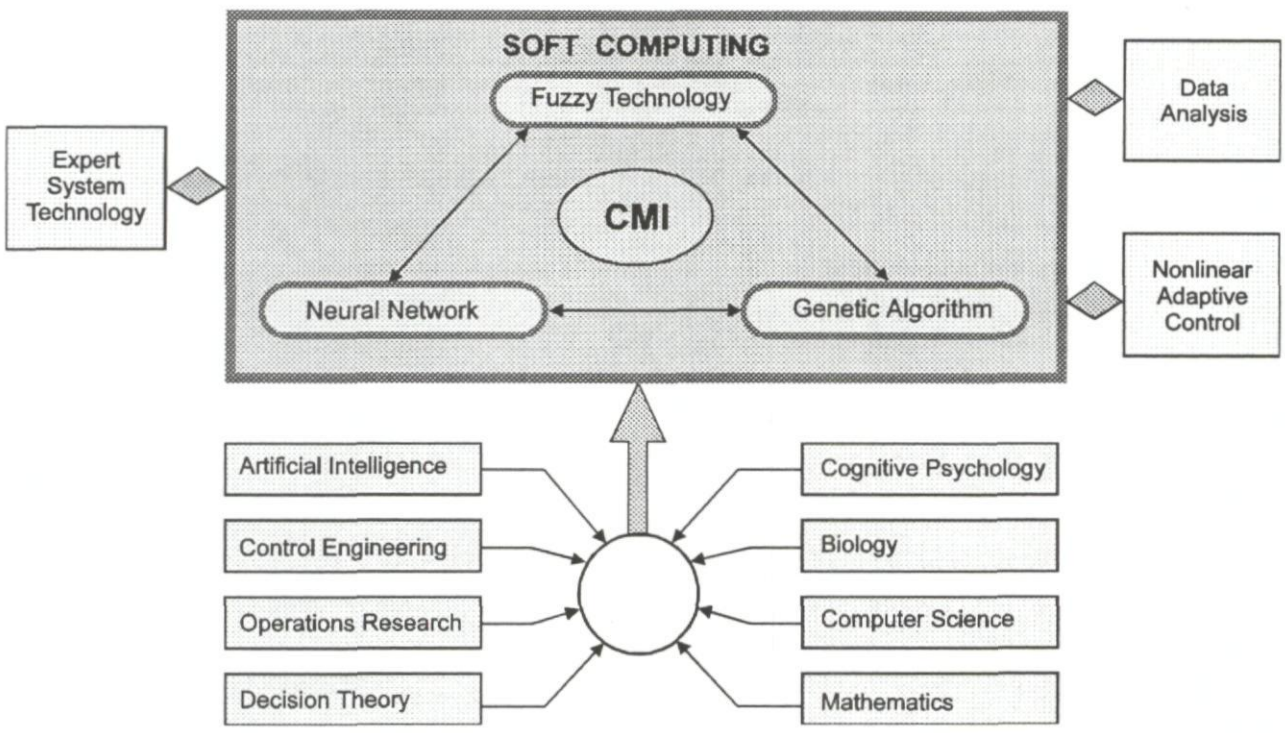


Figure 3: Relationships to and contributions from other areas and soft computing/CMI

The Mission Systems Panel (MSP) of AGARD felt it particularly important and timely to facilitate, foster and strengthen communication and cooperation between scientists, practitioners and decision-makers in various disciplines.

In this context the present lecture series is seen as a valuable step where the objective is to introduce soft computing as the basis for CMI and to briefly familiarize the participants with important related technologies and techniques as well as applications.

The first lecture introduces and overviews the soft computing techniques enabling CMI. Based on brainlike structures the most important techniques of soft computing such as fuzzy logic, artificial neural networks and genetic algorithms are briefly described. The treated approaches form the basis for adaptive, learning control, as well as advanced automation and artificially intelligent machines.

The series continues with an introduction and review of neural networks, their basic theory and important network models. It also looks at the application potential for mission systems.

Fundamentals of fuzzy logic, fuzzy inference and fuzzy control are treated in the third lecture, which is followed by a presentation of the basics of genetic algorithms and evolutionary computing.

A paper on hybrid architectures for intelligent systems concludes the part which is dealing with fundamental aspects. Various ways of combining fuzzy and neural elements for classification and control applications are presented.

The second part of the lecture series is devoted to applications of soft computing technologies in mission systems. Contributions address sensor signal processing, guidance and control problems, mission management and simulation issues as well as autonomous systems.

The first lecture demonstrates a combination of Kalman filter and maximum likelyhood techniques with neural networks for acquiring and tracking targets.

It is followed by a paper dealing with the application of neural networks and fuzzy techniques in the area of guidance and control. The emphasis is on reconfiguration of damaged aircraft.

The last paper but one considers the application of genetic algorithms and evolutionary strategies for mission management, simulation and autonomous systems.

Finally, the last paper summarizes important enabling techniques and technologies for the implementation of future autonomous systems. Main emphasis is placed on information technology with its soft computing techniques.

Literature:

[1] J. Rasmussen
Skills, rules and knowledge
IEEE Transactions an Systems, Man and Cybernetics
Vol. 13, No. 3, May/June 1983

Techniques for Computational and Machine Intelligence SOFT COMPUTING

U. Krogmann
Bodenseewerk Gerätetechnik GmbH
Postfach 10 11 55
D-88641 Überlingen

SUMMARY

There is a paradigmatic complementary shift from symbolic AI/KB techniques to so called Soft Computing technologies. The new paradigm is based on modelling the unconscious, cognitive and reflexive function of the biological brain. In contrast to the conventional Hard Computing, Soft Computing addresses the pervasive imprecision of the real world. This is accomplished by consideration of the tolerances for imprecision, uncertainty and partial truth to achieve tractable, robust and low cost solutions for complex problems.

The objective of this paper is to introduce Soft Computing and to briefly familiarize the reader with important Soft Computing techniques. This is accomplished by first of all defining what is ment by computational and artificial intelligence and why as well as for which functions computational intelligence is needed. Based on brainlike structures the most important techniques of Soft Computing such as fuzzy logic, artificial neural networks and genetic algorithms are briefly described. The approaches treated provide the basis for adaptive, learning control, advanced automation and artificially intelligent machines.

1 INTRODUCTION

Dealing with the title of this paper, the following question arises immediately: What is computational, machine or more generally artificial intelligence? In relation to the issues and topics treated here, the following answer shall be given.

- Systems/units **have no artificial intelligence** if a program/software "injects" them with what they have to do and how they have to react to certain pre-specified situations.
- Systems/units **have artificial intelligence** if their "creator" has given them a structure - not only a program - allowing them to organize themselves, to learn and to adapt themselves to changing situations.

Moreover, the next important question suggests itself: Why computational intelligence?

For reasons of human limitations in more demanding dynamic scenarios and in the operation of complex, highly integrated systems, there is the necessity for extended automation of functions on higher levels such as mission management and control.

Furthermore, the implementation of intelligent functions on lower levels such as the fusion and interpretation of sensor data, multifunctional use of sensor information and advanced nonlinear learning control become inevitable.

Paper presented at the AGARD MSP Lecture Series on "Advances in Soft-Computing Technologies and Applications in Mission Systems", held in North York, Canada, 17-18 September 1997; Amsterdam, The Netherlands, 22-23 September 1997; Madrid, Spain, 25-26 September 1997; Ankara, Turkey, 6-7 October 1997, and published in LS-210.

Improved effectiveness in diagnostics and maintenance of systems, subsystems and modules can be achieved through the application of computational and machine intelligence (CMI).

Last but not least CMI will be the most important prerequisite for the emergence of future autonomous systems with the ability fo function as an independent unit or system over an extended period of time, performing a variety of actions necessary to achieve predesignated objectives while responding to stimuli produced by integrally contained sensors.

In general terms, CMI is concerned with the automation of the functions of the so-called recognize-act-cycle (Fig. 1), which in the past have predominatly been performed by man. Related knowledge-based functional elements of the goal-directed interactions of man and/or machines with the surrounding world, object or system are found on all functional levels of complex integrated mission systems as well as mission management systems. As an example, Fig. 2 shows a simplified block diagram of a mission control loop. The mission control functions include the guidance and control of one or more air vehicles (aircraft, missiles), and/or other processes and units (radar, communication, etc.) in order to influence the scenario in a goal-directed manner.

Performance of the controller activities - as shown in Fig. 2 - by a technical apparatus requires implementation of mental humanlike functions, such as skill-, rule-, knowledge based functions. This in turn requires means for the acquisition, encoding, representation, storage, processing and recall of knowledge. Consequently, in the following we are concerned with related enabling techniques and structures offering artificial intelligence through appropriate CMI.

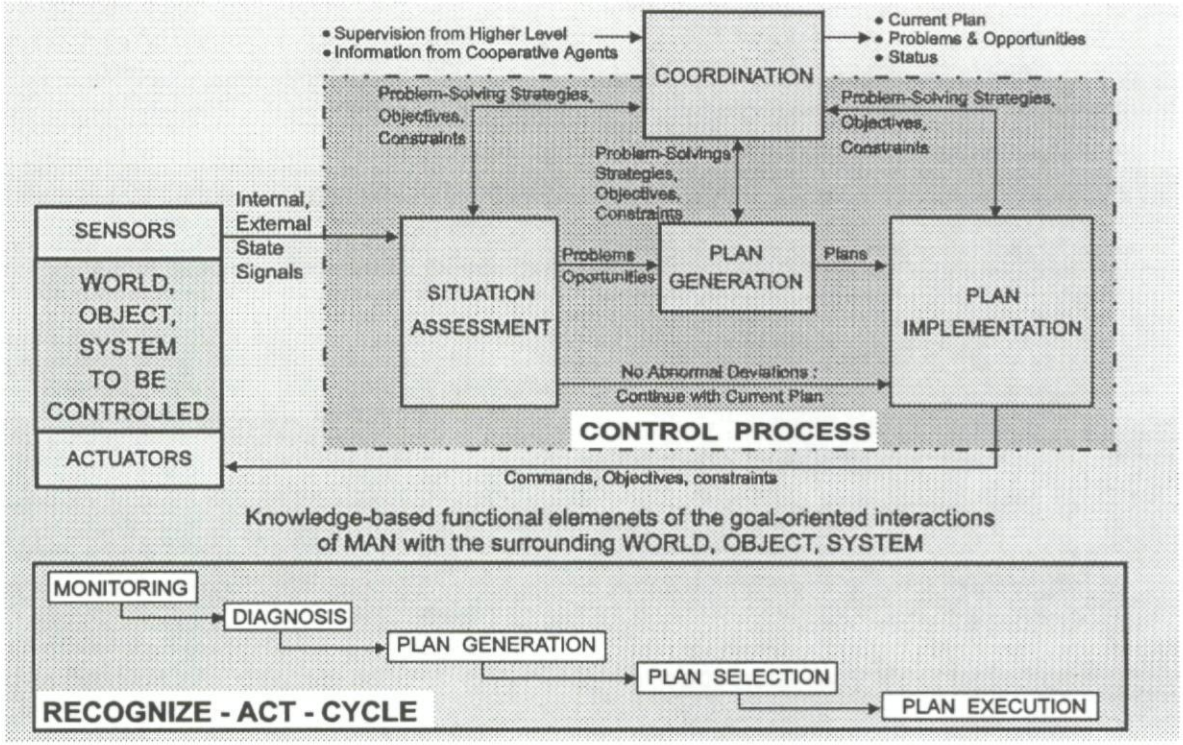


Figure 1: Functional elements of the Recognize-Act-Cycle

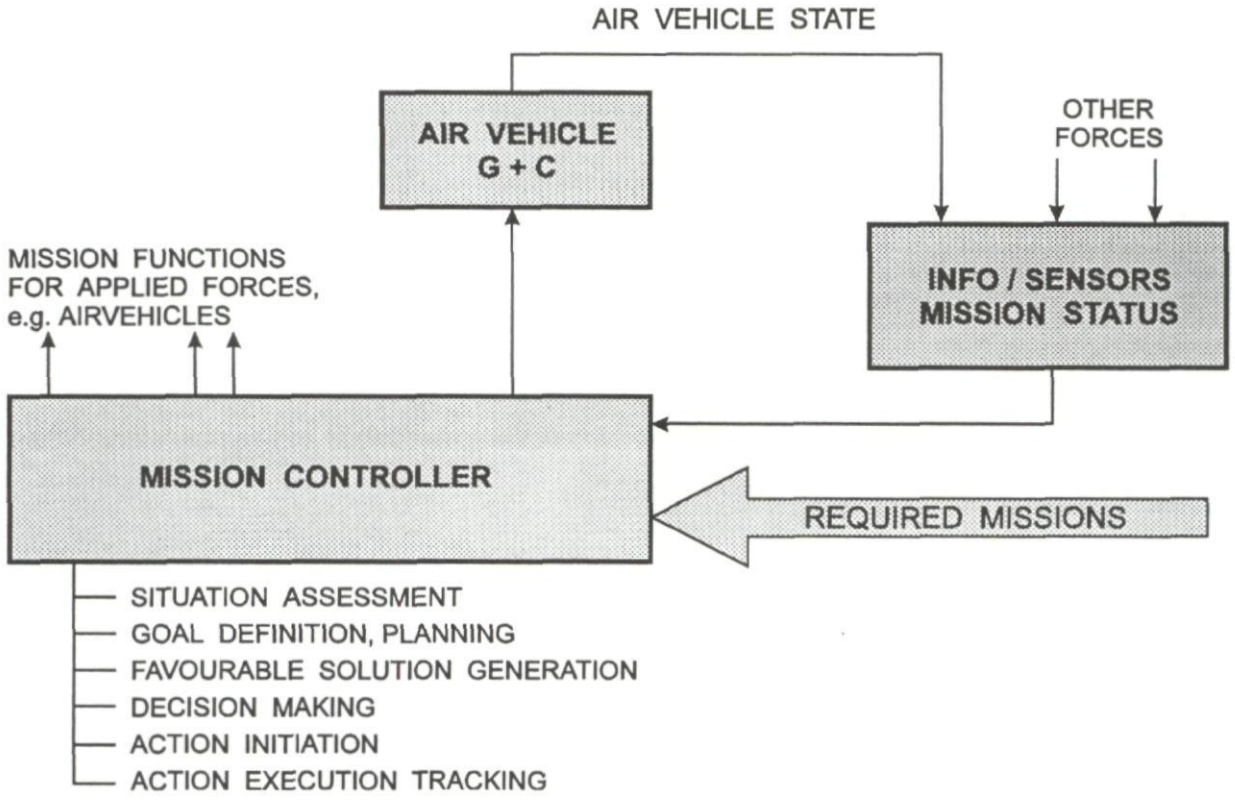


Figure 2: Mission control loop

2 PARADIGM SHIFT TO BRAINLIKE STRUCTURES

The expected unprecedented advances in computing based on the conventional architecture, where processing is performed sequentially, do not yield the performance required for computational and machine intelligence. This becomes ultimately evident if we compare a contemporary high performance computer (Cray YMP/8) with the brain of an insect, as represented in Fig 3. A vast amount of hardware and software consuming a lot of power is needed to build and operate the Cray-machine, as compared to the biological brain. What is even more impressive is the level of capabilities achieved with the latter. To make at least a small step towards a technical implementation of biological intelligence, as it is needed for future automation, new computational techniques and architectures must be considered and introduced. There is a paradigmatic complementary shift from symbolic artificial intelligence techniques to a new paradigm which is inspired by modeling the conscious and unconscious, cognitive and reflexive functions of the biological brain. The scientific paradigmatic shift regarding the engineering view of the biological brain is depicted in Fig. 4. Two different functions of the brain are to be looked at. First, there is the rational thinking with a function in conscious steps performed in a particular serial sequence. The digital computers we use today with a sequential processing of instructions listed in programs (computers in so-called von-Neumann architecture) were developed in the 1940s based on the investigation of sequentially conscious thinking. The conventional AI which manifests itself in the classical expert systems is an extension of this view towards symbolic processing. Recently, it has been extended from crisp to fuzzy representation leading to fuzzy expert systems.

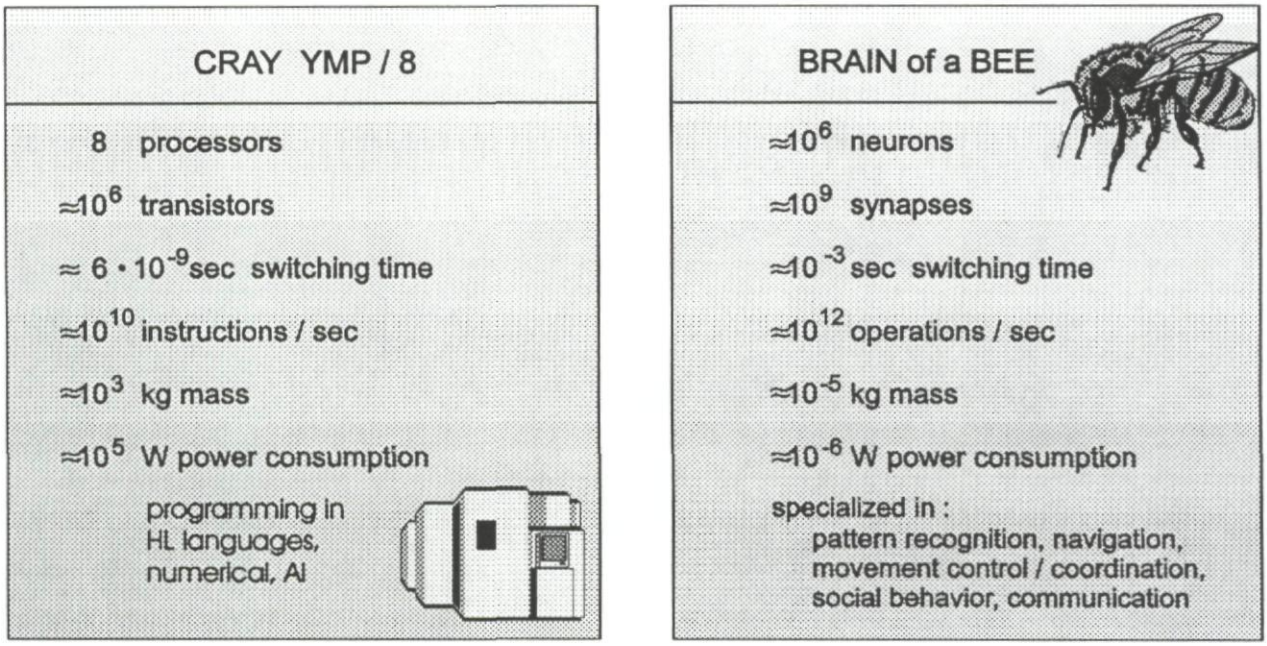


Figure 3: Comparison computer-brain of an insect

On the other hand, there are the much more complex structures of unconscious thinking or unconscious intelligence. Here, a lot of environment data are processed within the context of our sensory perception and characteristics extracted. The sensorimotor control of our motions as well as three-dimensional thinking are largely unconscious. The structures of unconscious thinking provide the basis for the enormous capacity of our memory. All of these functions performed unconsciously are running parallel in networks in which so-called neurons interact due to a close interconnection and by means of electrochemical processes. Consequently, the new engineering view looks at the brain as a dynamical system which can be adapted by learning. It is represented by artificial neural networks (ANN).

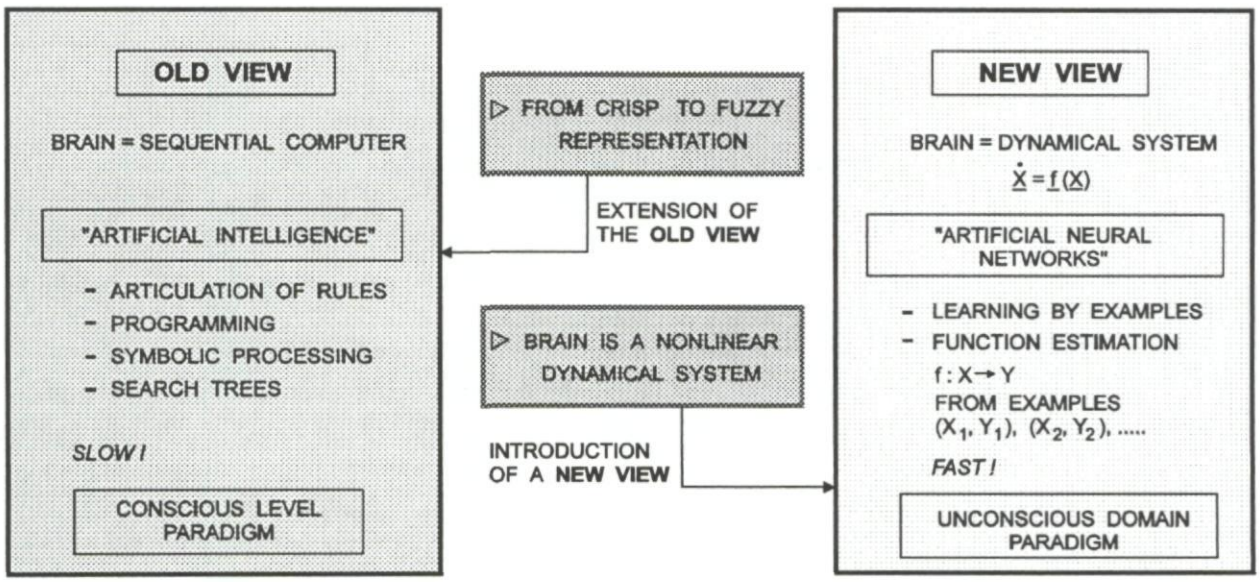


Figure 4: Engineering views of the biological brain

The levels of artificially intelligent information processing inspired by the biological brain paradigm are summarized in Fig. 5. Important related computing methodologies and technologies include inter alia fuzzy logic, neuro-computing and evolutionary and genetic algorithms. Apart from conventional analytical techniques they represent new powerful means for knowledge acquisition, representation and processing as shown in Fig. 6.

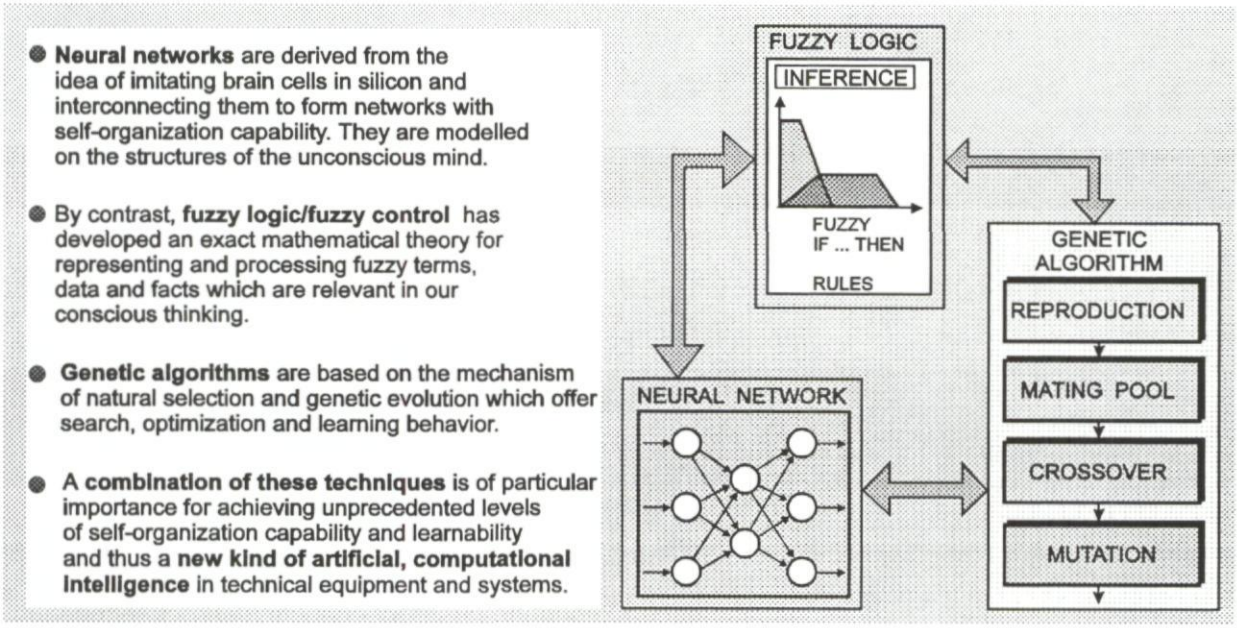


Figure 5: Information processing inspired by biology

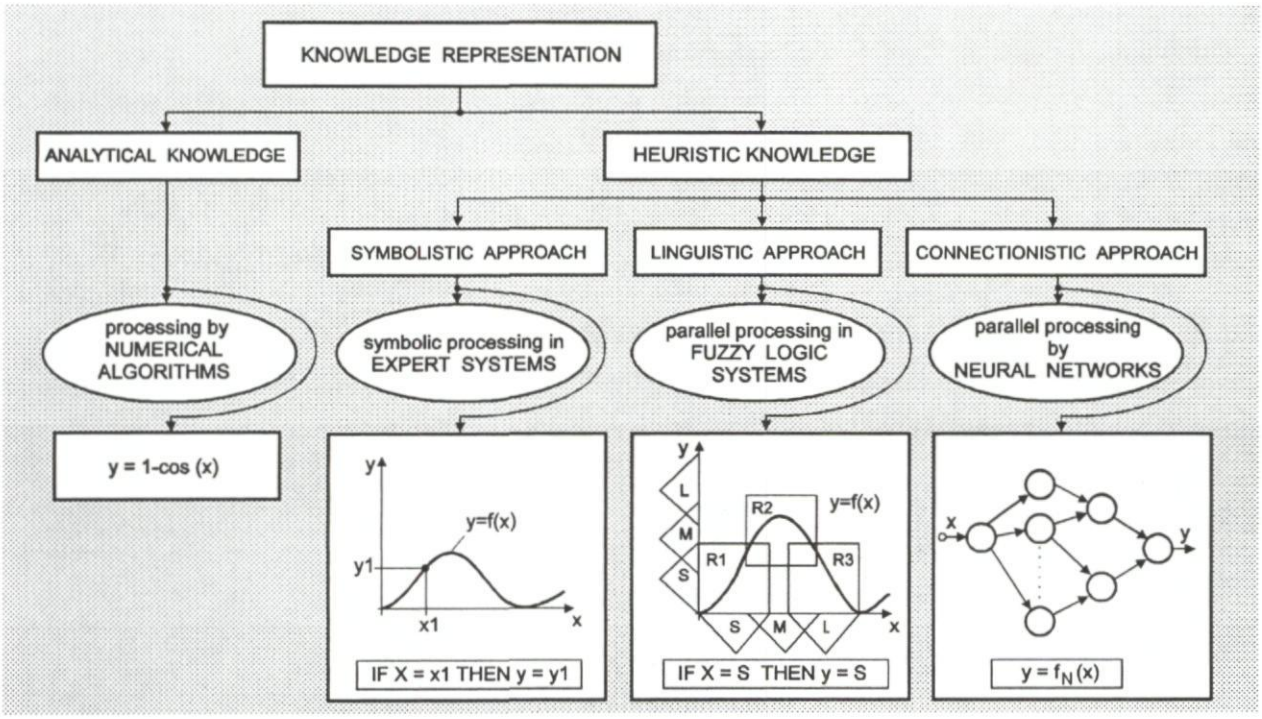


Figure 6: Knowledge representation and processing

Based on the above definition of AI, a system behavior is called intelligent, if the system emits appropriate problem-solving responses when faced with problem stimuli. In order to perform these responses, humans and animals estimate functions by associating a response y to a stimulus x, i.e. f: $x \rightarrow y$. As can be seen from Fig. 6, the computing methodologies shown can represent such functional relations, where each representation has its individual features and capabilities. Their basic functions shall be briefly summarized in the following.

3 SOFT-COMPUTING TECHNIQUES TO ENABLE COMPUTATIONAL AND MACHINE INTELLIGENCE (CMI)

3.1 Overview

As mentioned before, there is a paradigmatic complementary shift from symbolic AI/KB techniques to so called soft-computing technologies. The new paradigm is based on modeling the unconscious, cognitive and reflexive functions of the biological brain. In contrast to conventional hardcomputing, softcomputing addresses the pervasive imprecision of the real world. This is accomplished by consideration of the tolerance for imprecision, uncertainty and partial truth to achieve tractable, robust and low cost solutions for complex problems. Important related computing methodologies and technologies include fuzzy logic, neuro-computing, evolutionary and genetic algorithms. It becomes increasingly evident that combinations of these technologies are advantageous in many applications. A viable step towards intelligent machines can be expected that offer autonomous knowledge acquisition and processing, self-organization and structuring as well as associative rule generation for goal-oriented behavior in rarely predictable scenarios.

Neural networks are derived from the idea of imitating brain cells in silicon and interconnecting them to form networks with self-organization capability and learnability. They are modelled on the structures of the unconscious mind.

The theory of fuzzy logic provides a mathematical framework to capture the uncertainties associated with human cognitive processes, such as thinking and reasoning. Also, it provides a mathematical morphology to emulate certain perceptual and linguistic attributes associated with human cognition. Fuzzy logic provides an inference morphology that enables approximate human reasoning capabilities for knowledge-based systems. Fuzzy logic/fuzzy control has developed an exact mathematical theory for representing and processing fuzzy terms, data and facts which are relevant in our conscious thinking.

Genetic algorithms are based on the mechanism of natural selection and genetic evolution which offer search, optimization and learning behaviour.

A combination of these techniques is of particular importance for achieving unprecedented levels of self-organisation capability and learnability and thus a new kind of artificial intelligence in technical equipment and systems.

3.2 Fuzzy Logic

A unit based on fuzzy logic represents an associator that maps spatial or spatiotemporal multi-variable inputs to corresponding associated outputs [1]. The knowledge which relates inputs and outputs is expressed as fuzzy if-then rules in the form IF A THEN B, where A and B are linguistic labels of fuzzy sets determined by appropriate membership functions. By fuzzy if-then rules the qualitative aspects of human knowledge reasoning and decision processes can be modelled without the need for a precise quantitative description. Fuzzy logic units, also known as fuzzy-rule-based systems, fuzzy models, fuzzy associative memories (FAM) or fuzzy controllers, consist of five fundamental functional blocks as shown in Fig 7.

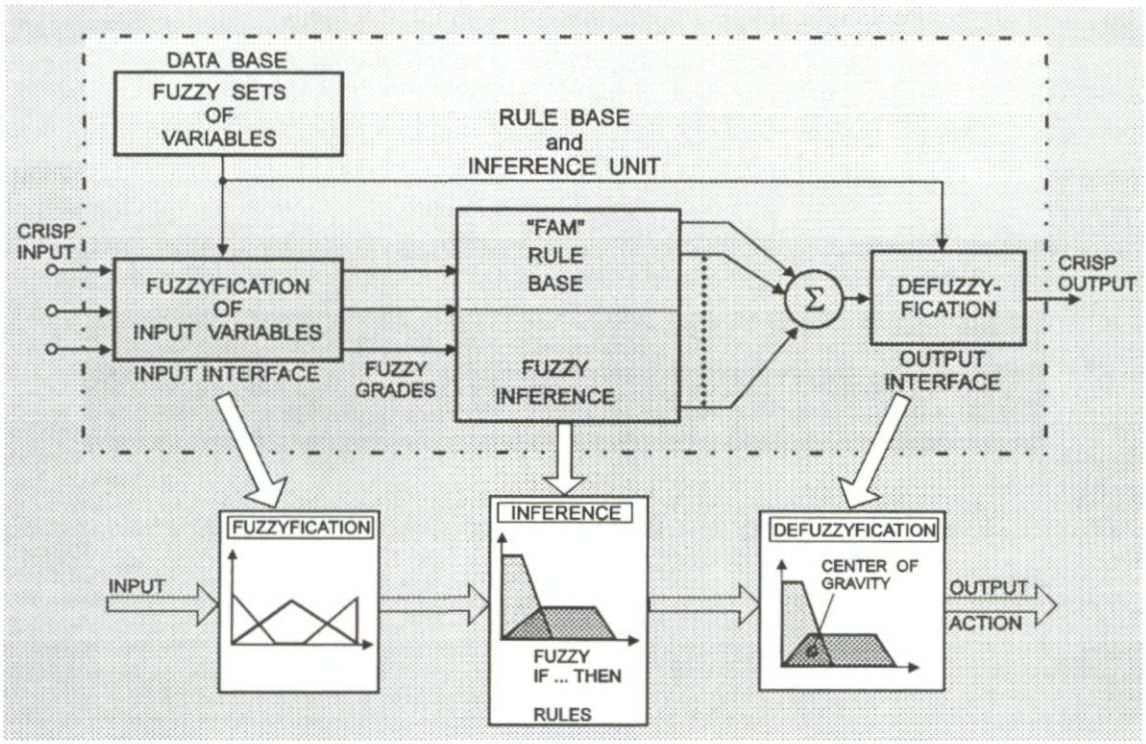


Figure 7: Fuzzy logic unit

The fuzzification interface transforms the crisp inputs into degrees of memberships to linguistic labels of fuzzy sets. A number of fuzzy if-then rules is contained in the rule base. They comprise the knowledge about the relationship between the antecedent and consequence variables. The membership functions of the linguistic fuzzy sets as used in the fuzzy rules are defined within the data base. Rule and data base are often jointly referred to as the knowledge base. The inference unit is the decision making element and performs the inference operation applying crisp mathematics of fuzzy logic. The fuzzy result of the inference process is transformed into a crisp output by the defuzzyfication block. By fuzzy logic processing, the vague imprecise rules can be given a specific meaning and outputs can be produced for inputs which only partially match the rules. This is a very powerful attribute.

Based on the consciousness paradigm, fuzzy rule based systems enable endomorphic real world modelling. With this technology, human behavior can be emulated in particular as far as reasoning and decision-making and control is concerned taking into account the pervasive imprecision of the real world. To quote Albert Einstein: "So far as the laws of mathematics refer to reality they are not certain. And so far as they are certain, they do not refer to reality." Thus, finally it can be concluded that fuzzy logic strongly supports realistic modelling and treatment of reality.

3.3 Artificial Neural Networks (ANN)

Neurocomputing is a fundamentally new kind of information processing and the first real alternative to the known computing techniques executed sequentially according to instructions stored in programs [2]. In contrast to programmed computing, in the application of neural networks the approach is learnt by the neural network used by mapping the mathematical correlations and constantly adapted during use. Neural networks are information processing structures composed of simple processor elements (PE) and networked with each other via unidirectional connections. Fig. 8 shows as an example a multi-layer network with exclusively forward connections. The PEs in a network all have the same function. They only differ in the values of the interconnection weights. The "knowledge" is contained in the variable interconnection weights. They are adjusted during a learning/training phase and continue to be adapted during operational use. The neural network learns the correlation between vectors of an input quantity and the related vectors of the output quantity. The basic functions of ANN are summarized in Fig. 9.

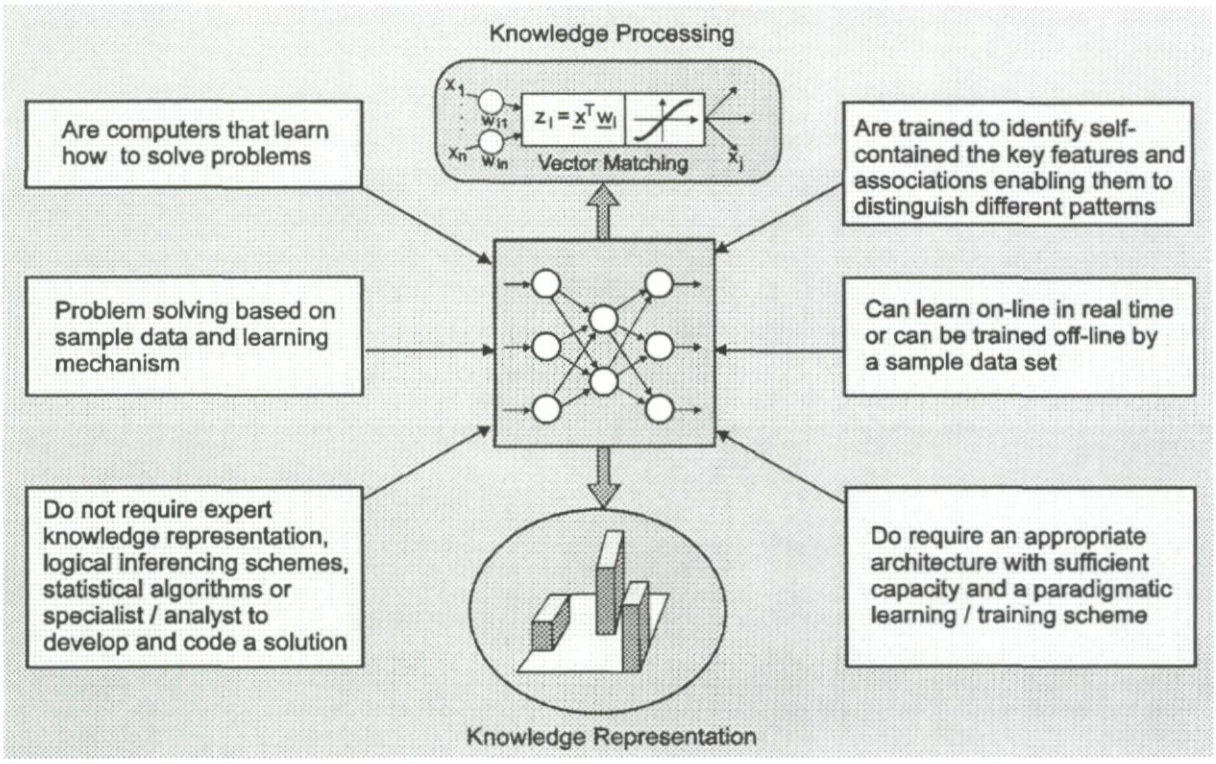


Figure 8: Artificial neural networks

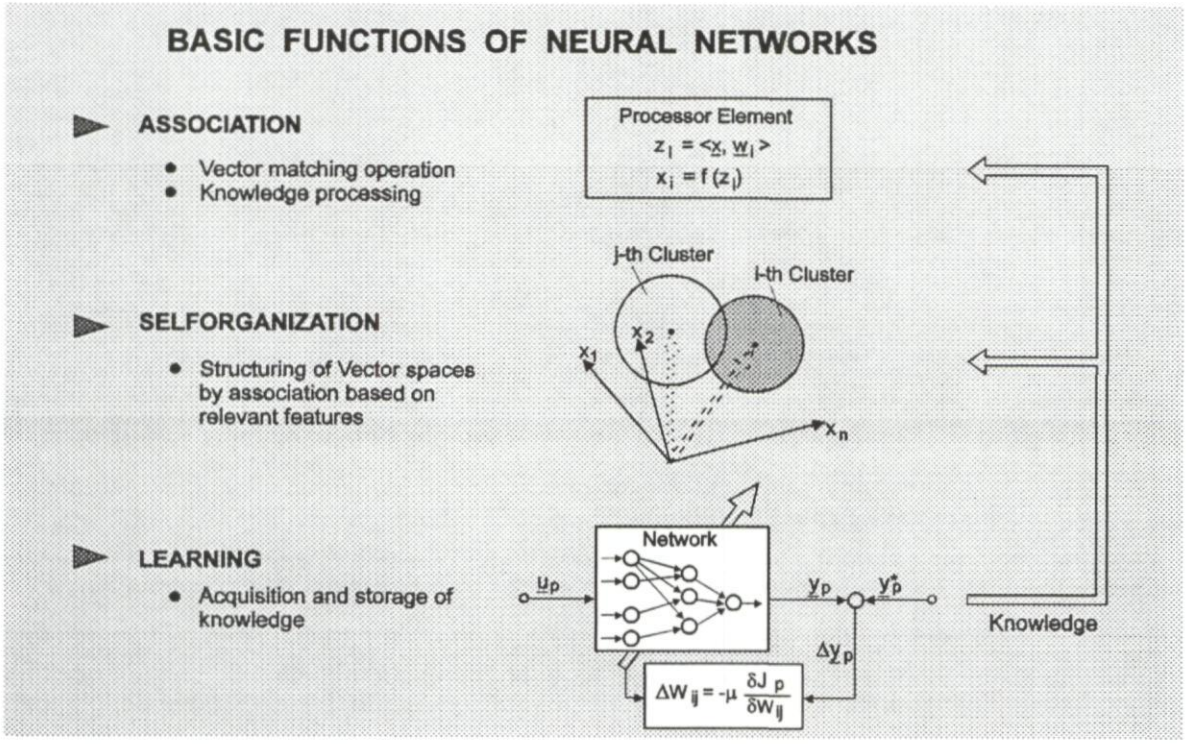
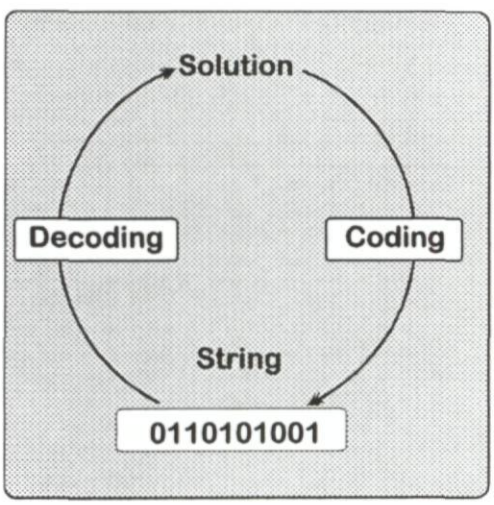


Figure 9: Basic neural network functions

With this capability the ANN represents an associator (like a fuzzy logic unit) that maps spatial or spatio-temporal multi-variable inputs to corresponding associated outputs. However, in contrast to a fuzzy-rule-based system the mapping function is learnt by the ANN. A network is characterized by the organized topology of interconnected PEs, the method of encoding information and the method of recalling the knowledge.

ANNs as derived from the unconscious biological brain paradigm are capable of acquiring, encoding, representing, storing, processing and recalling knowledge. These are important prerequisites for endomorphic real world modelling.



- Populate the computer with not just one, but a population of solutions, i. e. GAs search from a population of points not just from a single point.
- Use objective (fitness) functions, not derivatives.
- Make that population evolve through rules :
  - Independant from the problem.
  - That tend to generate better and better solutions.
  - Use of probabilistic transition rules.
- All the user needs to known is how to evaluate a solution (fitness).

Figure 10: Evolutionary computation

3.4 Genetic and Evolutionary Algorithms

Genetic and evolutionary algorithms represent optimization and machine learning techniques which initially were inspired by the processes of natural selection and evolutionary genetics [3]. They simulate biological evolution by representing possible solutions as chromosomes and use an evolution or fitness function, in analogy to natural selection, in order to determine the worth of that solution, i.e. selection and survival of the fittest.

As mentioned, to apply a genetic algorithm (GA), potential solutions are to be coded as strings on chromosomes (Fig. 10). The GA is populated with not just one but a population of solutions, i.e. GA search from a population of points rather than from a single point. By repeated iterations, a simulated evolution occurs and the population of solutions improves until a satisfactory result is obtained. The initial population evolves through rules which are independent of the problem, tend to generate better and better solutions and utilize probabilistic transition rules. As shown in Fig. 11 in a simplified form, this is accomplished by iteratively applying the genetic operators reproduction, crossover and mutation. Strings are copied and parts swapped as well as changed in a probabilistic manner by the genetic operators. As an illustrative example, the evolutionary development of a Neural Network is suggested in Fig. 11. The network structure and parameters are coded as strings on chromosomes. Crossover of „parent" networks generates offsprings, i.e. „children". These are evaluated by the fitness function and the fittest are retained for the next generation and so on. The network is optimally generated by time-lapsed simulation of biological evolution. All the user needs to know is how to evaluate a solution. For this purpose, an objective or fitness function has to be defined. It measures the performance or goodness after reproduction, i.e. for each generation. Thus, GAs use domain knowledge only through the objective function. The genetic operators are application independent. Stopping criteria can either be a maximum number of iterations or observation of an evolution below a given threshold.

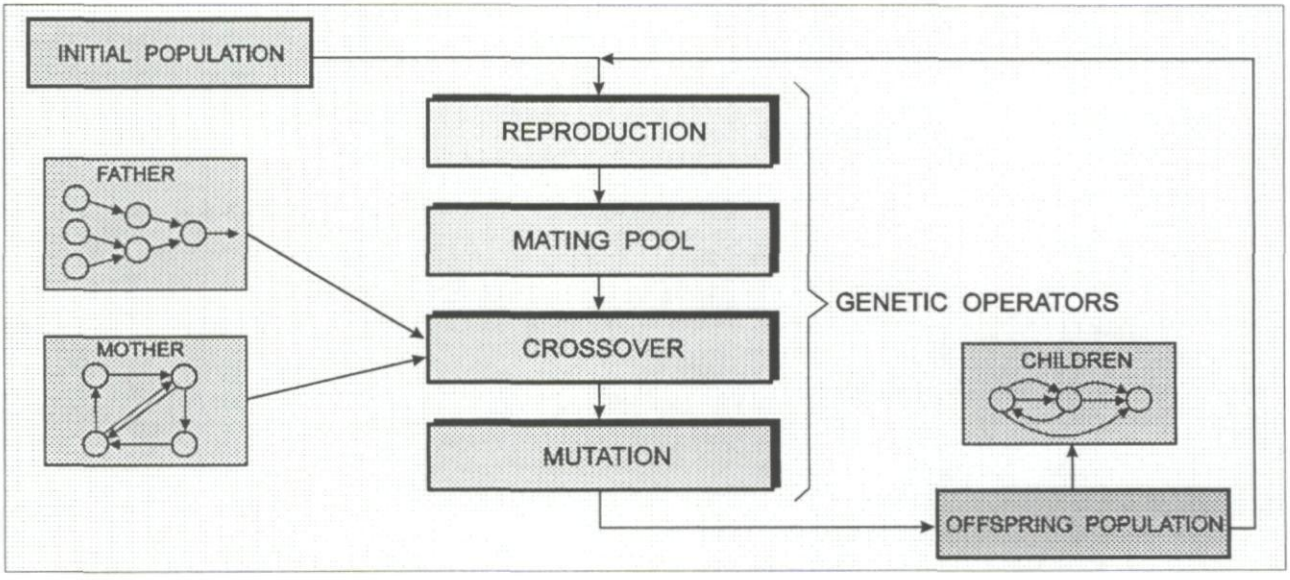


Figure 11: Genetic operations for a single generation

GAs have the potential to yield optimization methods that are global (avoid local minima) and general, (i.e. can be applied to different types of problems), that are robust with respect to problem modifications and last but not least intelligent, i.e., a deep understanding of the problem structure is not required. With GA, an optimum balance between efficiency and efficacy can be achieved for technical goal-directed systems. This is mandatory for a successful behavior in natural environment, i.e. real world. Computer simulation is a viable tool to optimize behavior oriented systems by utilizing genetic or evolutionary techniques. An ever-increasing processing speed enables the quick motion representation of events and processes for which nature needs millions of years.

3.5 Conclusions

It was mentioned that fuzzy and artificial neural network techniques enable the endomorphic modelling of real world objects and scenarios. Together with conventional algorithmic processing, classical expert systems, probabilistic reasoning techniques and evolving chaos-theoretic approaches, they enable the implementation of recognize-act cycle functions as shown in Fig 1.

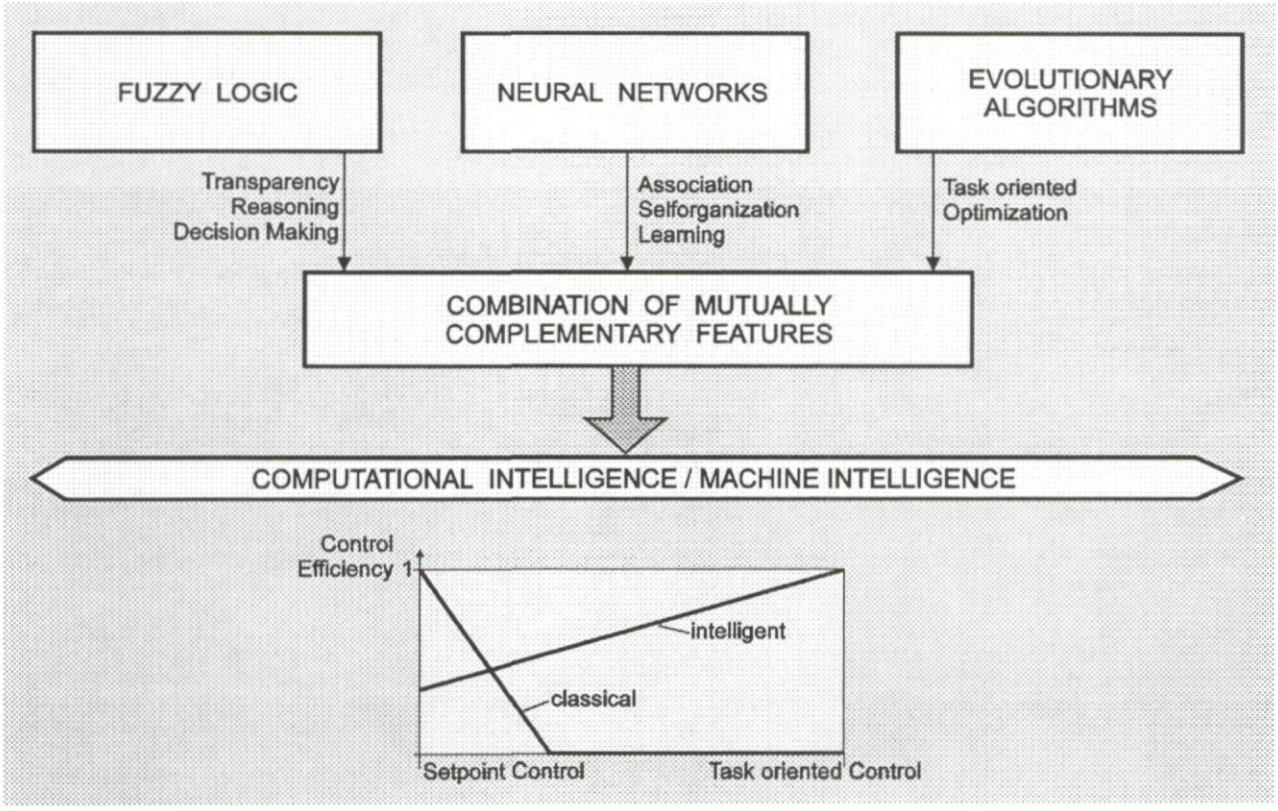


Figure 12: Complementary features of technologies

Genetic and evolutionary algorithms can be applied to generate and optimize appropriate structures and/or parameters to acquire, encode, represent, store, process and recall knowledge. This yields self-learning control structures for dynamic scenarios that evolve, learn from experience and improve automatically in uncertain environment. Ideally, they can be mechanized by a synergetic, complementary integration of fuzzy, neuro- and genetic techniques as shown in Fig. 12. Some integrated, cooperative approaches of fuzzy logic and ANN are briefly summarized in

Fig. 13. Fuzzy Neural Networks, for example, as shown in Fig. 14 allow the representation of heuristic knowledge by fuzzy rules taking into account the fuzzyness and imprecision of the real world and have the capability of learning, e.g. rules and/or membership functions. All very important attributes within the context of problems as treated here. Thus, Fuzzy Neural Networks have the potential to capture the benefits of the two fascinating fields of fuzzy logic and neural networks into a single capsule.

Soft-computing techniques support the move towards adaptive knowledge based systems which rely heavily on experience rather than on the ability of experts to describe the dynamic, uncertain world perfectly. Among other things this will yield so-called Case Based Reasoning (CBR) and Planning (CBP) modules where experience is gained, stored and retrieved when similar conditions are repeated, or modified to suit different situations. Thus, soft-computing techniques in conjunction with appropriate system architectures provide the basis for creating systems which map the definition of artificial intelligence.

Finally, it can be stated that the approaches treated here could form the basis of a new general theory and design methodology for adaptive, learning, nonlinear control, advanced automation and artificially intelligent machines.

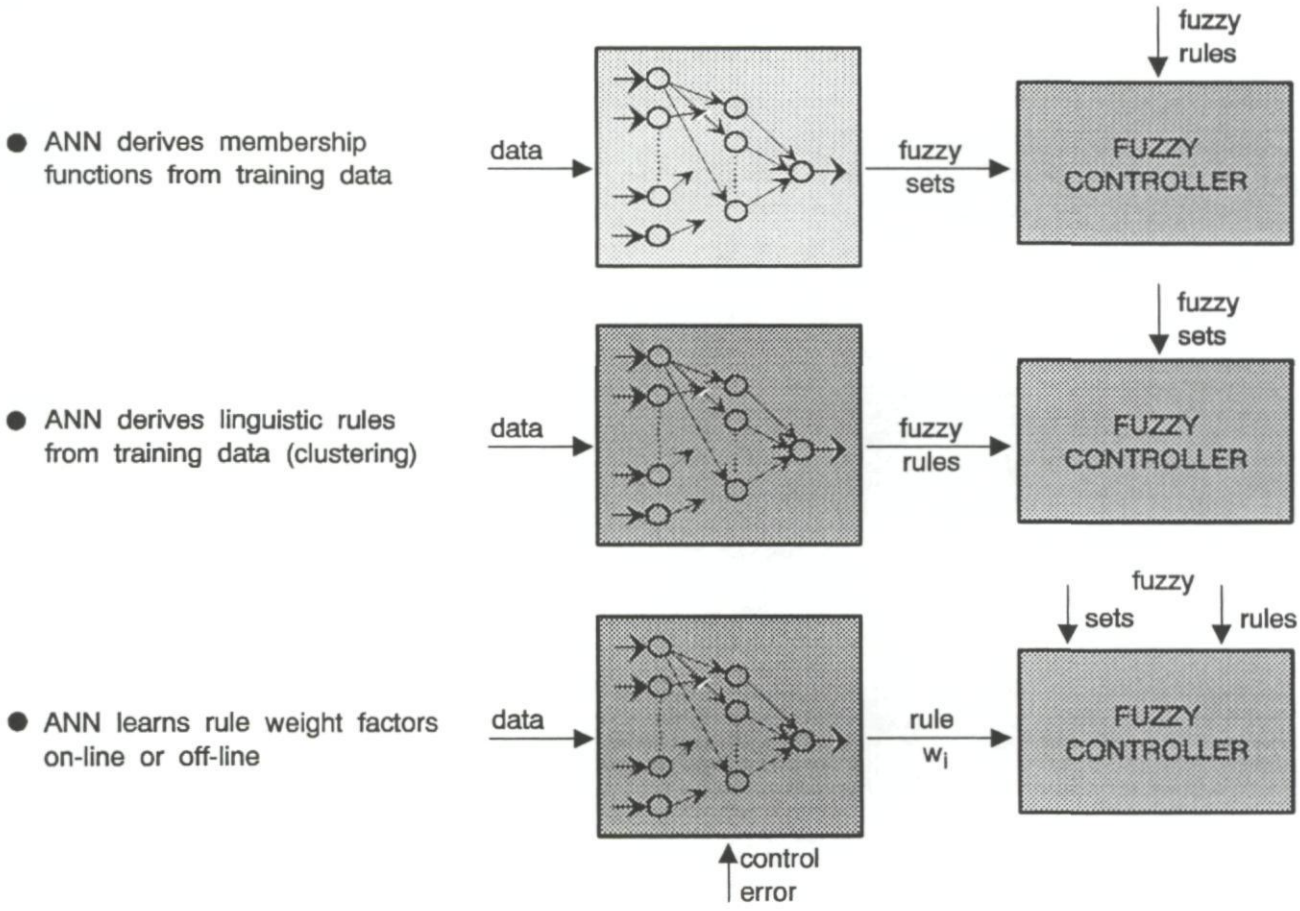


Figure 13: Integration of fuzzy logic and neural network

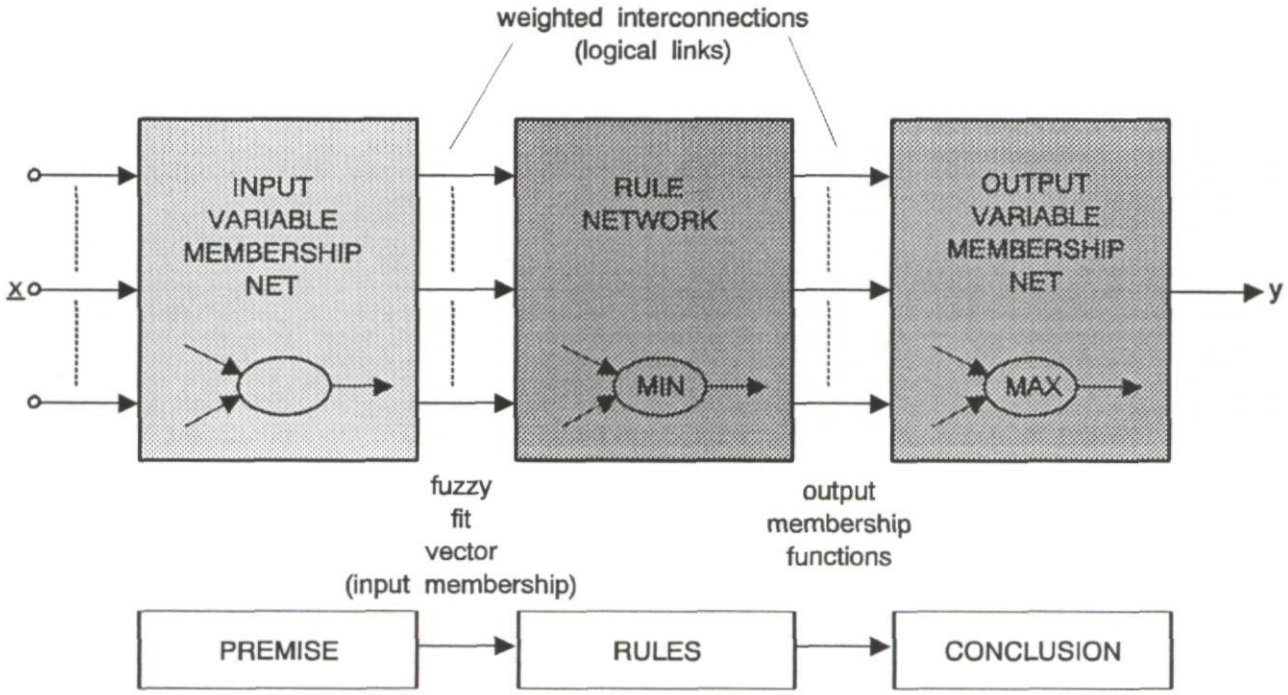


Figure 14: Structure of a fuzzy neural network

Literature:

[1] B. Kosko
Neural Networks and Fuzzy Systems
Prentice Hall Inc., 1992

[2] U. Krogmann
Introduction to Neural Computing
AGARD LS179, Monterey, USA, Oct. 1991

[3] E. Sanchez
Genetic Algorithms and Soft Computing
1. European Congress on Fuzzy and Intelligent Technologies
7.-10. Sept. 1993, Aachen, Germany

INTRODUCTION TO NEURAL NETWORKS BASIC THEORY AND APPLICATION POTENTIAL FOR MISSION SYSTEMS

Daniel J. Collins
Department of Aeronautics and Astronautics
Naval Postgraduate School
699 Dyer Road (Rm 137)
Monterey, California 93943-5106
U.S.A.

1. INTRODUCTION

This lecture is a brief introduction to neural networks. Modern approaches to complex engineering problems typically involve the development of a computer algorithm with a specified set of inputs. For a well posed problem the inputs specify a unique output or solution. Thus for the state variable linear quadratic problem in control theory with the plant specified one uses a Riccati equation to obtain a set of unique feedback gains.

In some complex problems the necessary algorithm may not be evident. Problems involving recognition or identification of a human face, for example, are extremely difficult for computers but relatively easy for people.

This recognition ability is innate with humans as is shown by the fact that at an extremely young age children recognize their parents.

Artificial Neural Networks (ANN) in some sense emulate the human brain. A neural network can learn, self organize and have associative memory properties. These properties are characteristic of human intellectual activity.

In contrast to a programed algorithm ANN's consists of a set of interconnected nodes which we will call neurons in analogy to the brain. The connections between nodes are weighted and the nodes have a local internal transfer function. By variation of the topology of the neural network different effects and systems can be developed.

Neural networks can be classified in two categories based on the learning method i.e., supervised and unsupervised learning. A further classification can be made based on feedforward networks and recurrent networks. In the rest of the lecture, we will discuss these different types of neural networks and highlight several practical applications [1, 2, 3, 4].

2. NEURON SPECIFICATION AND LOCAL TOPOLOGY

The neuron or node of the neural network is the basic computation or processing unit. All computations are done locally which makes for efficient computation. Consider the node 'N' given in figure 1.

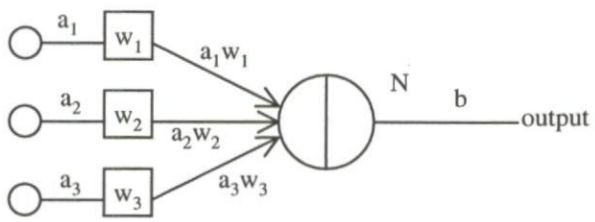


Figure 1 - Neuron Specification

The neuron or node labelled 'N' receives inputs from the three associated neurons. Their outputs a_i are attached to the principal neuron by a set of weighted connections indicated by w_i. The net input to the neuron which is normally summed is then

$$net_j = \sum a_i w_i \qquad (1)$$

Associated with N may be an activation level α. A new activation level can then be calculated using some threshold function that can be non-linear.

$$\alpha_{new} = F(\alpha_{old}, net_j) \qquad (2)$$

As example of a threshold function could be

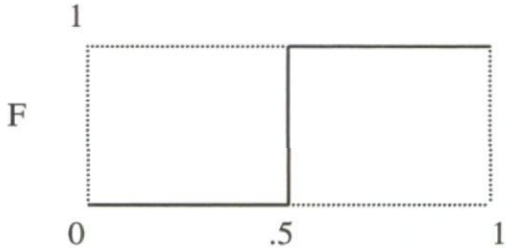


Figure 2 - Threshold Function

Paper presented at the AGARD MSP Lecture Series on "Advances in Soft-Computing Technologies and Applications in Mission Systems", held in North York, Canada, 17-18 September 1997; Amsterdam, The Netherlands, 22-23 September 1997; Madrid, Spain, 25-26 September 1997; Ankara, Turkey, 6-7 October 1997, and published in LS-210.

From the new activation level 'α_{new}' the output b of the neuron can be calculated using for example a sigmoid activation function

$$b = f(\alpha_{new}) \qquad (3)$$

where

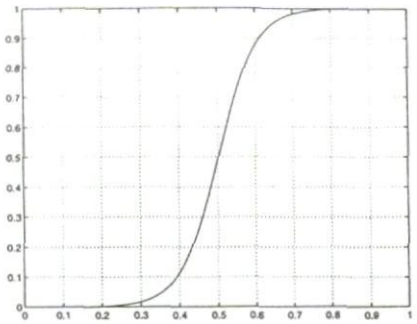


Figure 3 - Activation Function

The neuron N should be viewed as imbedded in a neural network where the topology of connections may vary widely and where also the inputs, the threshold function and the activation function may also vary. Different neural networks are developed as we shall see as we vary the general topoplogy.

The information content of the neural networks rests in the value of the weights which are changed based on a learning process. Again there are a wide number of learning processes associated with different neural networks. As we discuss different types of neural networks the concepts covered in this section should become clear.

3. FEEDFORWARD SYSTEMS

Active work on neural networks began in the 1950's with the invention of the perception [5]. The first investigations were limited to a single input layer and on output layer.

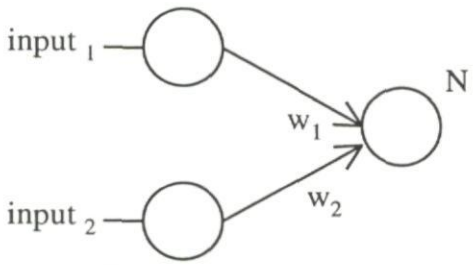


Figure 4 - Perceptron

This device by proper selection of the weights and the threshold function could represent a logical gate such as an 'AND' or 'OR' gate. For example if $w_1 = w_2 = .5$ and N has a threshold function of .6 then the neural network with binary inputs represents an 'AND' gate.

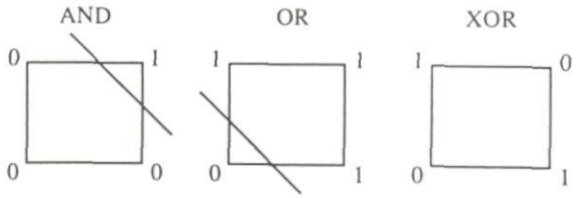


Figure 5 - Linear Separability

The perception had the difficulty that it could only handle linearly separable solutions as is illustrated in figure 5.

For linearly separable problem a guaranteed global solution exist.

Linearly separable problems are, however, not of great interest. (AND-OR gates). The XOR problem also shown in figure 5 cannot be solved by means of a simple perception. In order to solve the XOR problem an intermediate node (or hidden units) is needed as is shown in figure 6.

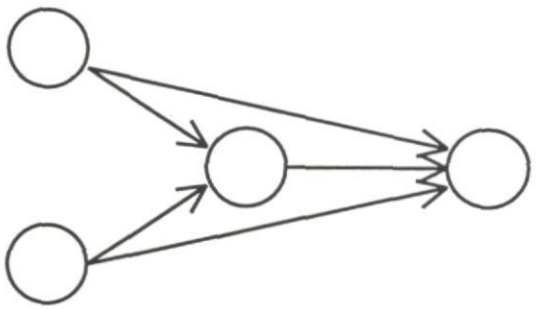


Figure 6 - XOR Solution

Unfortunately there did not exist a general method for determining the weights of hidden units until the development of the generalized delta rule by McClelland and Rumelhart [2]. This is a gradient descent method. The greatest development of neural networks occurred after the development of the generalized delta rule.

A general multilayer neural networks (figure 7)

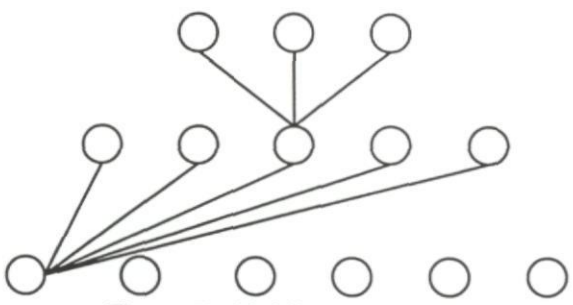


Figure 7 - Multilayer Network

can be used to simulate any nonlinear transfer function from the input layer to the output layer. Multilayer feedforward neural networks (sometimes referred to as multilayer perceptions) have been the basic neural networks used in the solution of a wide range of problems connected in particular to pattern recognition and function approximation. There are now available several professional systems which permit the rapid development of neural networks for specific engineering problems. There is also a Neural Network toolbox that can be incorporated in an instructional program.

4. SUPERVISED LEARNING (BACK PROGATION ALGORITHM)

The generalized delta rule is an example of supervised learning. We will later discuss unsupervised learning. In supervised learning there is a set of pairs of input patterns and output patterns. In the learning procedure the input pattern is applied to the neural network and the corresponding output pattern is calculated. If the output pattern created by the neural network is equal to the desired output pattern, then no learning takes place. If the output patterns differ then the weights of the neural network are changed to decrease the error between the desired output pattern and the neural network produced pattern. How the weights are changed is the learning rule for the particular neural network.

In the case where there are no hidden layers there are many variations in the learning rule. The delta rule (also called Widrow-Hoff rule) [6] is given by

$$\Delta w_{ji} = \eta(t_j - o_j)\, i_i = \eta\, \delta_j\, i_i \qquad \text{with} \quad \delta_i = (t_j - o_j) \tag{4}$$

where 't' represents the desired or target vector and 'o' represents the output vector from the neural network. t_j is the j th component of the desired output pattern. o_j is the j th component of the calculated output vector. i_i is the value of the i component of the input vector and Δw_{ji} is the calculated change in the weight connecting the i th input to the j th output. The constant η is referred to as the learning rate. We also assume at this stage linear activation functions.

This formulation is a generalization of the perception learning rule which has a guarenteed convergence theorem. We now following the development of Rumelhardt and McClelland and define the overall measure of error for a given pattern p as

$$E_p = \frac{1}{2} \sum_j (t_{pj} - o_{pj})^2 \tag{5}$$

The total error would then be

$$E = \sum E_p \tag{6}$$

Applying the chain rule one can obtain

$$\frac{\partial E_p}{\partial w_{ji}} = \frac{\partial E_p}{\partial o_{pj}} \frac{\partial o_{pj}}{\partial w_{ji}} \tag{7}$$

$$\frac{\partial E_p}{\partial o_{pj}} = -(t_{pj} - o_{pj}) = -\delta_{pj} \tag{8}$$

but

$$o_{pj} = \sum_i w_{ji}\, i_{pi} \tag{9}$$

and

$$\frac{\partial o_{pj}}{\partial w_{ji}} = i_{pi} \tag{10}$$

or

$$-\frac{\partial E_p}{\partial w_{ji}} = \delta_{pj}\, i_{pi} \tag{11}$$

This development clearly shows that the delta rule is a gradient descent method based on the error E or E_p

The extension of this method to hidden layers leads to the generalized delta rule. It can be shown that multiple layers with linear activation functions are equivalent to a single layer. We therefore introduce nonlinear activation functions but with the constraint that the functions are nondecreasing and differentiable functions of the total net input. Note that threshold function of figure 2 does not satisfy this constraint since they are not everywhere differentiable.

By apply much the same mathematical procedure as above one can develop with

$$net_{pj} = \sum_i w_{ji}\, o_{pi} \qquad (12)$$

two recurrsive equations for calculating the weight changes for any layer in the neural network. The equations are

$$\delta_{pj} = (t_{pj} - o_{pj})\, f_j'(net_{pj}) \qquad (13)$$

(note applied at output layer)

and

$$\delta_{pj} = f_j'(net_{pj}) \sum_k \delta_{pk} w_{kj} \qquad (14)$$

(note applied at hidden layer)

The generalized delta rule has the same form as the delta rule and the δ_{pj} are use in equation 4 to calculate the changes in the weight.

Thus the application of the generalized delta rule involves two passes through the network. In the forward pass the output pattern is calculated for a given input vector. The error vector is then calculated and in a backward pass through the network the weights are progressively changed to reduce the error.

Although we are guaranteed a global minimum in the case of the delta rule, we have no such guarantee in the case of the generalized delta rule. It is possible to get trapped in local minimum. Several techniques are available to minimize this problem. Experience seems to indicate that for large networks local minimum are not a serious problem.

We are now at a point where we can describe how to train and test a neural network. One must have a set of data consisting of matched input output pairs. One selects a given pair and places the input data in the input layer. This information is propagated through the network to the output level. A comparison of the actual output with the desired output creates an error vector which is then used in a backpropagation through the network to systematically change the weights. This procedure is also referred to as the backpropagaton algorithm. After multiple presentations of data pairs the network has 'learned' the system under consideration. By 'learned' one means that a set of weights has been obtained such that for a given input the desired output is obtained.

Assuming that the data set consists of three sub classes I, II, III, then after the training of the neural network in Figure 5 an exemplar from class I would activate say the first neuron in the output layer, etc.

If the neural network is properly trained it can more than just identify members of the training set. The network has abstracted the essential fixtures of the data and will correctly identify other exemplars of class I data which it has not been trained on. It is possible to overtrain a network so that it memorizes to some extent the training data. In this case the ability of the neural network to generalized to new data may be compromised.

There are some practical problem which we have not considered such as what is the size of the input layer and how large should the hidden layer be? We need also to normalize the input data to prevent divergence of the network. Further unlike the perception which could guarantee an optional solution it is possible to obtain suboptional solutions using the back-propagation algorithm. We will have further discussion of these factors under the applications section of the lecture.

5. RECURRENT NETWORK - FEEDBACK SYSTEMS

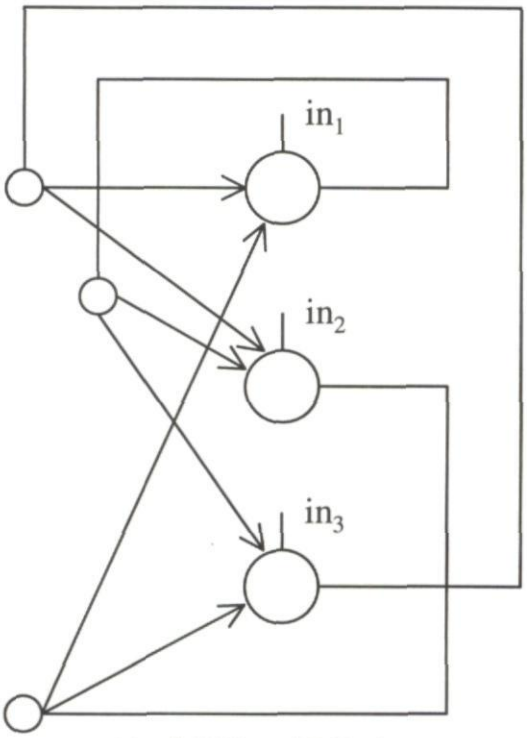
In the feedforward network there is no feedback of the output signal to the current or previous layers. Recurrent network are those involving feedback. This feedback in the modeling of a discrete system can have previous knowledge of the state of the system. There is therefore an element of memory introduced into the neural network when we consider recurrent networks.

Although recurrent networks are potentially more powerful than feedforward, networks, they introduce the question of stability. Under certain conditions the network can either converge, oscillate or become unstable.

Hopfield in 1982 [7] introduced the idea of storing information in a dynamically stable network. The Hopfield network is a recurrent network that is credited with reviving interest in neural networks. The neurons in the network have two binary states, 1 and -1 the activation function is therefore the sigmum function. The weight matrix denoted by W is symmetric

$$W = W^T \qquad (15)$$

Self excitation is not permitted. A Hopfield network of 3 nodes is shown in Figure 8.



Hopfield Net of 3 Nodes
Figure 8

A Hopfield net is a single layer recurrent network which has associate memory capability. A probe vector with elements ±1 is placed on the nodes of the network. Each neuron is accessed randomly to determine its activation level with respect to a given threshold. If above the threshold the neuron switches to +1. If it is below the threshold it switches to -1. If already in the appropriate state it remains there. The sequential random changes in the neuron insures stability of the network.

With N neuron one has 2^N possible outputs or states of the network. These states can be viewed as the vertices of a hypercube thus for 2 nodes one has 4 states shown in figure 9.

Figure 9 - Hypercube

The number of stable states is determined by the current inputs, the network weight and the threshold value. The network has an associate memory capability in that partial or incorrect inputs stabilize to the vertex closes to the desired one.

Hopfield also pointed out a isomorphism between his neural network and the Ising model of magnetic materials. This permitted the application of existing theory to the analysis of neural networks. Detailed analysis of stability of Hopfield neural networks involves the use of Lyapunov functions. One of the major contributions of Hopfield was to suggest an appropriate Lyapunov function for such analysis. This analysis was a major help in the analysis of the stability of recurrent nets.

A major limitation of the hopfield network is a limitation on the number of patterns that can be stored. The storable patterns are small with respect to the number of neurons

6. UNSUPERVISED LEARNING

6.1 Competative Learning

Competative learning all have the property that a competition is involved between units before learning is done. A mechanism exist such that only one neuron is on at a time. Figure 10 shows a simple competitive learning network, where the first layer permits the input of a desired pattern. This layer is fully connected to the output layer. Between the nodes of the output layer are lateral inhibitory connections, indicated by dotted lines. The neuron with the highest activations turns off the other nodes.

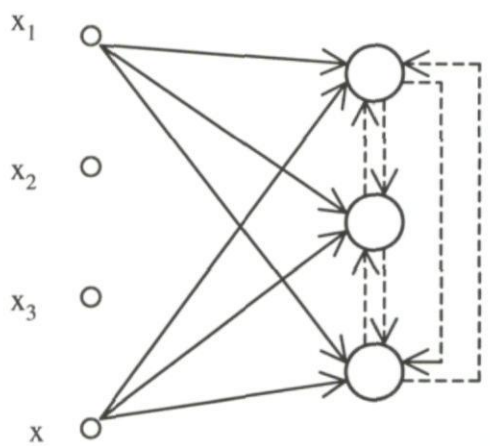


Figure 10. Single Layer Competative Network

The output of the winning neuron is set equal to one, while all other neurons have their output set to zero. In order to limit the maximum strength of any neuron the weights can be normalized

$$\sum_i w_{ji} = 1 \qquad (16)$$

It is also possible to normalize the input patterns which with current software is easy to accomplish. A

neuron learns by essentially shifting its weights in the direction of the input pattern . The competative learning rule is that Δw_{ji} is applied to w_{ji} with

$$\Delta w_{ji} = \begin{cases} \eta (x_i - w_{ji}) & \text{if neuron } j \text{ wins} \\ 0 & \text{if neuron } j \text{ loses} \end{cases} \qquad (17)$$

The effect of this learning rule is easily seen if we view the patterns as vectors on an N dimensional sphere where N is the number of nodes. In figure 11 the patterns are represented by x 's and the normalized weight vectors by o 's [8].

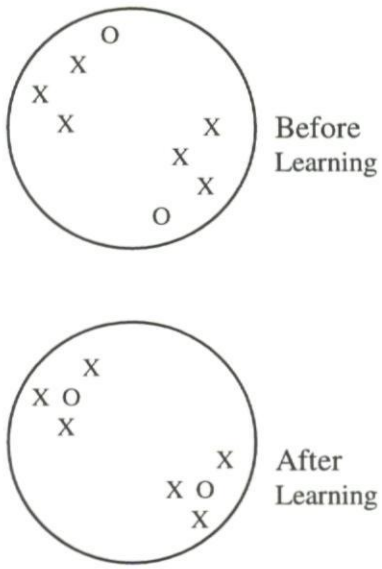


Figure 11 - Pattern Geometry

The learning rule shifts the weights in the direction of each of the patterns. Effectively the nodes specialize in a given set of patterns. Each of the nodes is then being selected for particular features in the input patterns.

6.2 Self-Organizing Network

Self organizing feature maps have two fascinating catagories of application. The first category deals with understanding and modeling of the method that the brain uses to process sensory data. Some successful modeling of visual perception has been based on self organizing feature maps (SOFM). The second category is in practical applications such as speech recognition, robot control and vector coding.The SOFM model developed by T, Kahonen (Self-Organization and Associate Memory) [9] is actually a vector coding algorithm. We will say more on this later.

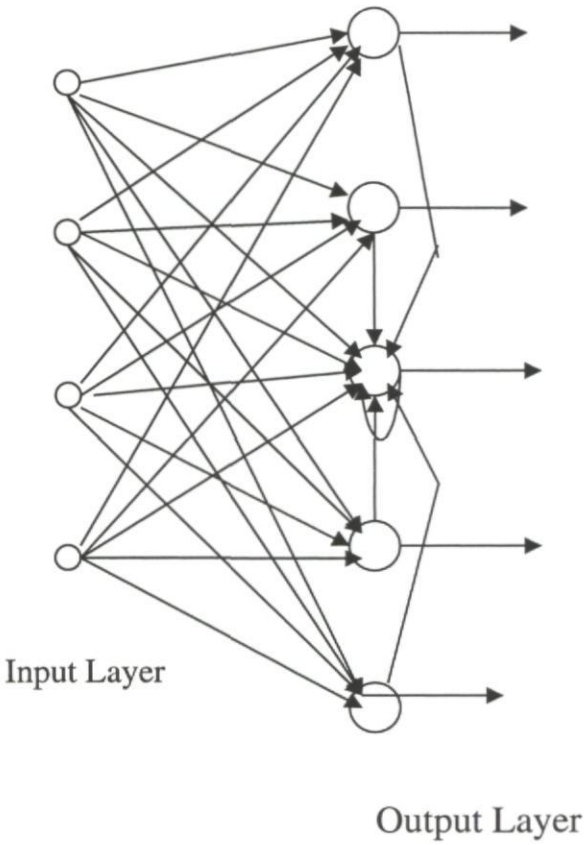


Figure 12 One Dimensional Lattice

In a SOFM the neurons are placed in a one dimensional lattice or a two dimensional lattice. Figure 12 is an example of a one dimensional lattice.

The SOFM is an unsupervised mapping of the input space on to the lattice, forming a topographic map of the input space. The location of an output neuron in the topographic map depends on the features of the input space.

In addition to the feedforward connections there are lateral feedback connections which are both excitatory and inhibitory in analogy to biological activity. In figure 12 are show the lateral connections for the center neuron. This includes a self feedback loop.

The lateral interactions are characterized by an interaction function called a Mexican Hat by Kohonen. In practice this function is approximated by the stepwise function a shown in figure 13.

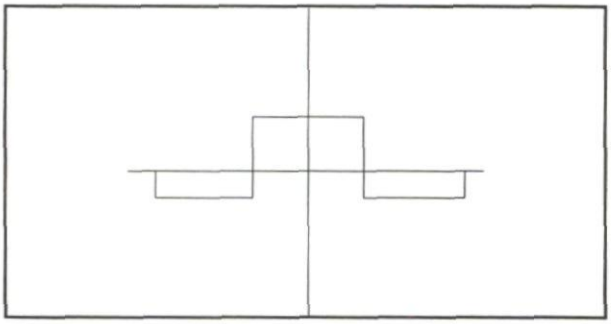


Figure 13 - Approximation to Mexican Hat Function

Thus the lateral interaction between neurons is characterized by a short range excitation area, followed by a weaker inhibitory area. The generation of the topographic map develops activity areas around firing neurons. The local clusters have been termed activity bubbles by Kohonen. The location of these activity bubbles depends on the input vectors or patterns.

SOFM are a competative learning system.. The neuron compete among themselves for activation and the activated neuron on cluster is a winner take all neuron(s).

Assume that we set up input vectors or patterns **x** with dimension P. In this case the number of terminals in the input layer would be P. For each neuron j in the output layer one has a weight vector $\mathbf{w}_j$ with p components since the network is fully connected.

The net input to neuron j from the input layer is then

$$net_j = \sum_{l=1}^{P} w_{jl} x_l \qquad (18)$$

One also needs to account for the lateral connections. Let

$$c_{j,-k} \text{----} c_{j,o} \text{---} c_{j,k} \qquad (19)$$

be the lateral feedback weights where k defines the region of influence of the firing neuron.

The output of neuron j then can be given by

$$y_j = f(net_j + \sum_{i=-k}^{i=k} c_j y_{j-i}) \qquad (20)$$

Where f is a nonlinear activation function such that $y_j \geq 0$ and y_j has a limited value.

Typically the equation for y_j is solved by relaxation on the corresponding difference equation which could be written as

$$y_j(n+1) = f(net_j + \beta \sum_{i=-k}^{i=k} c_{ji} y_{j-i}(n)) \qquad (21)$$

and β controls the rate of convergence.

One still needs to specify the SOFM algorithm. This algorithm transforms the signal pattern of a given dimension to a one- or two dimensional discrete map. Consider the input vector x and the weight vector w_j previously defined. A simple method of matching the two vectors is to take the scalar produce

$$w_j^T x \qquad for\ all\ j \qquad (22)$$

and select the largest value as the matching neuron. This assumes that each neuron has the same threshold level.

The weight vector w_j is often normalized and in this case an equivalent method is to find the minimum 2-norm of the different vectors

$$l(x) = \min_j \|x - w_j\|_2 \qquad j=1\text{---}N \qquad (23)$$

Associated with the winning neuron is a neighborhood

$$N_{l(x)}(n) \qquad (24)$$

which is a function of time n. Kohonen has suggested that this neighborhood should be fairly large at the beginning and then shrink with time.

Having selected the winning neuron we now need to discuss the adaptation algorithm for such a neuron. in order to prevent the neurons from saturating it is necessary to introduce a forgetting term in the equation for change. In the discrete case one has

$$w_j(n+1) = \begin{cases} (w_j(n) + \eta(n)[x(n) - w_j(n)]) & j \in N_{l(x)}(n) \\ w_j(n) & j \notin N_{l(x)}(n) \end{cases} \qquad (25)$$

This is the same form that we introduced for competative learning in the previous section. Note that the learning rate and the size of the neighborhood are functions of n.

6.3 Vector Quantization (VQ)

This section shows the relationship between SOFM and vector quantization. Vector quantization is connected with the idea of data compression or dimensionality reduction. Video teleconferencing (VTC) depends on vector quantization. Figure 14 shows the process

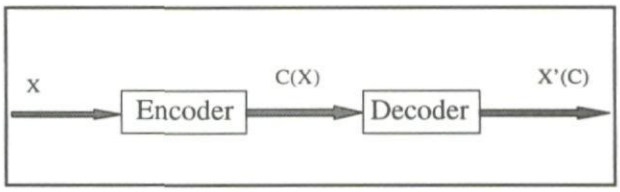


Figure 14 - Vector Coding

The image x in VTC is compressed by means of the encoder. it is then transmitted to the distance site where it is decoded to yield a representation of the original image x'(c).

The quality of the distance image x'(c) is a function of the encoding and decoding process.

In the VQ theory **x** is considered as random variable from an input set X with an implied probability density function f(x). One defines the average distortion by the integral [4].

$$D = \frac{1}{2}\int_{-\infty}^{\infty} dx f(x) \; \|x-x'\|_2^2 \qquad (26)$$

Where

$$\|x-x'\|_2^2 \qquad (27)$$

is the norm of the difference between the input vector and the reconstructed vector. The LBG algorithm [10] minimizes D in a two step process. It first choose a code c to minimize

$$\|x-x'(c)\|_2^2 \qquad (28)$$

It then uses c to calculate x'(c) as the centroid of the input vectors that satisfies equation 28.

The LBG algorithm is closely related to the SOFM algorithm. This relationship [11] can be shown by introducing noise into the system as shown in Figure 15.

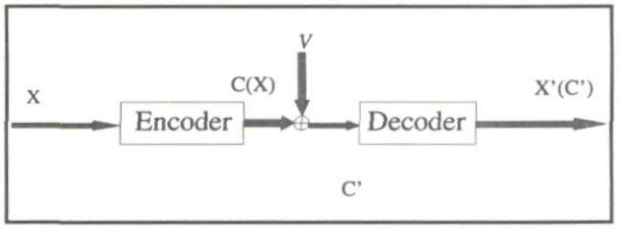


Figure 15 - Coding with Noise

A modified distortion integral is obtained

$$D_{\text{mod}} = \frac{1}{2}\int_{-\infty}^{\infty} dx f(x) \int_{-\infty}^{\infty} dv p(v) \|x-x'[c(x)+v]\|_2^2 \qquad (29)$$

Differentiating this integral with respect to x'(c) one has

$$\frac{\partial D_{\text{mod}}}{\partial x'(c)} = -\int_{-\infty}^{\infty} dx f(x) p(c' - c(x)) [x - x'(c)] \qquad (30)$$

The latter part of this integral is the distortion measure corresponding to equation 26 and the reconstruction vector is calculated from

$$x'(c) = \frac{\int_{-\infty}^{\infty} dx f(x) p(c' - c(x) x}{\int_{-\infty}^{\infty} dx f(x) p(c' - c(x))} \qquad (31)$$

The correspondence between VQ and SOFM algorithm is given in the following table

Table I

VQ	SOFM
c(x)	i(x)
x'(c')	w_i
p(c' - c(x_i))	$N_{i(x)}$

7. EPILOGUE ON NEURAL NETWORKS

Clearly it is not possible in a single article to cover all aspects of neural networks but before going on to the more applied aspects of this article it would be useful to mention some item that have been neglected.

In feedforward networks there are many modifications to the backpropogation algorithm. Current software normally permits the easy use of these modifications. Several feedforward networks, Grossberg Network, Radical Basis Networks,

Functional Link Networks, among others, have not been considered.

With respect to competitive networks we have not considered ART networks or the counterpropagation network.

With respect to dynamic associative memory networks we have not considered the Boltzman machine or bidirectional memory.

Finally there are many design problems connected with the implementation of a neural network. Although we will touch on some aspects of the design process in the application part of this article, it is useful to list some of the problems here. What is, for example, a minimum realization of a ANN? How many modes does one need? How many layers of nodes? How long should one train a network.?

We now turn to applying neural networks to some practical problems. We first consider modeling of nonlinear systems, and identification and control of dynamic systems.

8. GENERAL APPLICATION OF ANN

One of the first applications of ANN was used in the determination of credit loans. Banks use some 30 factors in determining a loan {salary, age, stability of employment, etc.). The network is trained by a human expert on a wide sample of loan cases. After training the ANN can be used to determine whether the applicant receives a loan. This type of feedforward network trained by a human expert could have an important application in complex processes such as the operation of a blast furnance which involves an element of art. There is a difficulty in obtaining the acceptance of the use of the neural networks by the experts. Recently it has been announced that neural networks are being used to track credit card payments with a view of detecting changes in the pattern of payments by the holder of the card.

One of the classical optimization problems is the "traveling salesman" problem. (TSP). The TSP is said to be NP-complete. The problem is easy to state. The salesman has a given number of cities to visit. The problem is to visit each city once and return to the starting city. The optimal tour of the cities is the one that has the shortest length. There is no known optimal method for selecting the shortest tour, short of calculating the length of all tours. Hopfield and Tank in 1985 [11] used a Hopfield network to calculate a solution to the problem. The method was based on a arbitrarily constructed Lyapunov function. Recent work indicates that a valid solution can be obtained, which although not optimal, may be quite good.

Neural networks have been used also in stock market predictions with mixed success. That the success is mixed is not surprising. The inputs to a neural network must contain features on which the neural network has been trained. One can look on these features as defining a set of vectors which define the space within which the neural network can function properly. Valid responses from a neural network cannot be expected when the inputs to the neural network do not contain features on which the neural network has been trained. Thus if the "state" of the stock market changes then a neural network trained on the previous state will cease to give valid outputs.

9. ANN IN ADAPTIVE CONTROL

Model following or model reference adaptive controllers are ideally suited to the use of neural networks. The eighth paper in this series gives a detailed discussion of the use of neural networks in model following. We will discuss here some ideas not covered in that presentation.

The ability of a neural network to model a plant is perhaps the first object that must be obtained when emulating a dynamic system. The typical system in controls is a linear time invariant (LTI) system represented by the equations

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx + Du \end{aligned} \tag{32}$$

The ANN is not limited to the LTI model. Nonlinear system models may also be emulated. It is interesting to consider the emulation results both in the frequency domain and in the time domain. Brunger [12] has emulated the longitudinal dynamics of the A4D aircraft. The model has 4 states given by [u, q, α, θ] In Figure 16 the frequency response of the original system and the trained ANN is shown. Only the angle of attack α and the pitching angle θ are shown. A random binary signal was used in the training. The network was trained for 100,000 random binary inputs. It should be noted that proper emulations require that the system be excited over the frequency range and amplitude range of the control. The signal characteristics need to comply with the concept of persistent excitation. Further one needs to scale and normalize the neural network in order to have successful emulation. Again current software facilitate these actions.

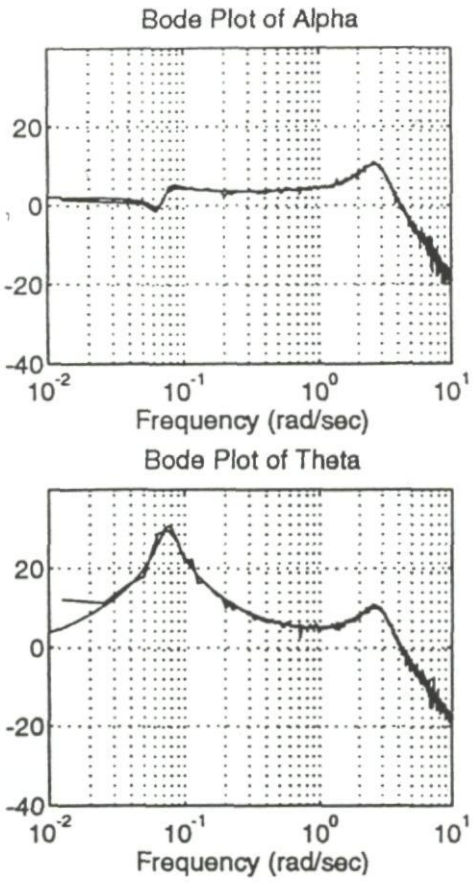


Figure 16 - Neural Network & System Bodes

The random binary input was used to obtain persistence of excitation. The excitation can be applied at the digital sampling rate. It is difficult to properly excite very low frequencies and one can see that the emulation diverges here in the graphical comparisons of Figure 16. Determination of the input power spectral density can directly show if one has proper excitation of the system.

The time response of both the ANN and the LTI model for α and θ are shown on the same graph in Figure 17. It can be seen that the two curves on each graph essentially are identical. It might be though that the low order of the system facilitated the emulation of the plant. In another study Bertrand [13] the X29, a demonstration fighter with a forward swept wing was emulated by an ANN. The plant model had 14 states and with the addition of an $H\infty$ controller the system emulated had 30 states. Figure 18 shows the time response of the pitching rate q of the ANN and original system. Again excellent emulation has been obtained. The frequency domain comparisons were also excellent.

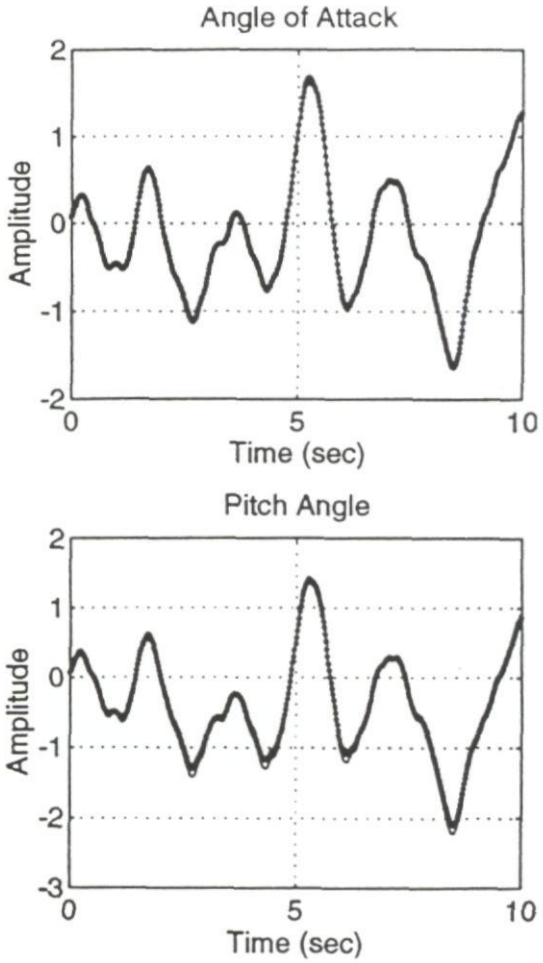


Figure 17 - Neural Network and System Time Response

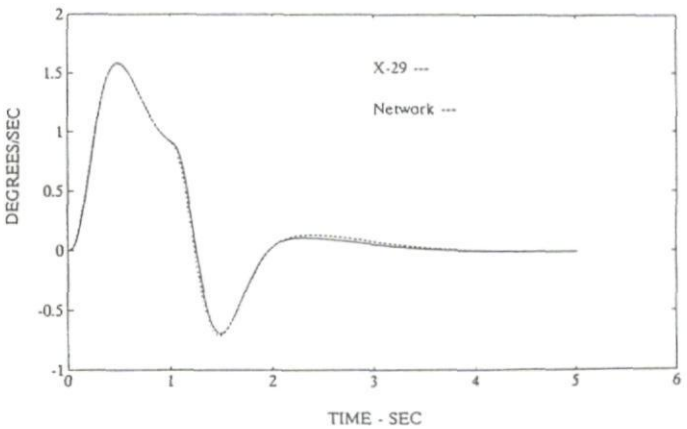


Figure 18 - X-29 q Response

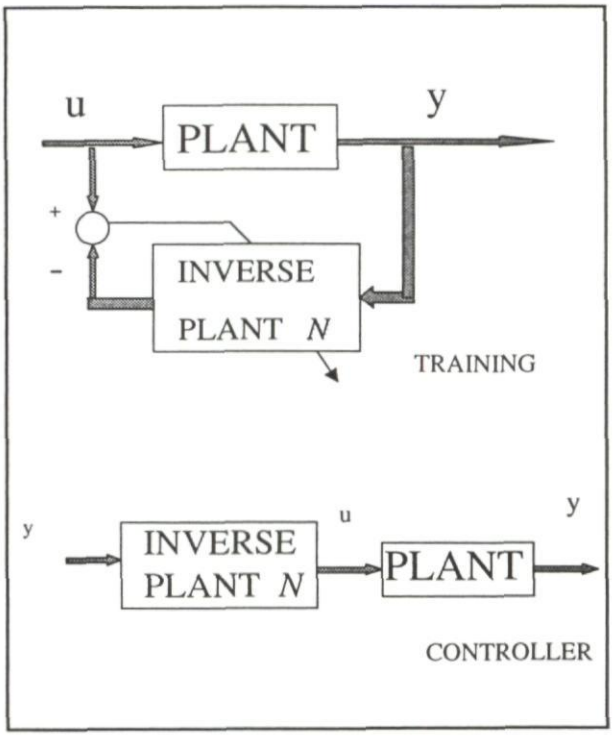


Figure 19 Inverse Dynamics

One method of control which is somewhat difficult classically but is relatively easy from a neural network viewpoint is that of inverse dynamics.

The inverse controller is an open loop controller as shown in Figure 19. The desired output y is given to the inverse which generates the need inputs for the plant. Typical aerodynanic transfer functions have non-minimum phase zeros (i.e., zeros in the right hand plane). The non-minimum phase zero becomes an unstable pole in the inverse plant. If the zero is not too far in the right hand plane, it is possible to obtain the inverse of the plant. Even if the inverse of the plant is unstable, over a finite time the plant is a bounded input, bounded output (BIBO) problem. For the A4D aircraft the stable 4th order plant has a non-minimum phase zero at 3.65 in the z plane. It was possible to obtain the inverse of the plant in this case. Figure 20 shows the input to the inverse neural network and the output of the system plotted on the same graph. Figure 21 shows the error between the desired output and the actual output . Note that the scale on the error is 10^{-4}. In the case of X29 with an unstable pole at 9.85 in the z plane, the instability grows so fast that the basic modes of the system are not excited. In this case emulation of the unstable inverse was not successful. The original X29 plant is also unstable with a pole in the s domain at 1.95. It was possible to model the unstable X29 plant by an ANN.

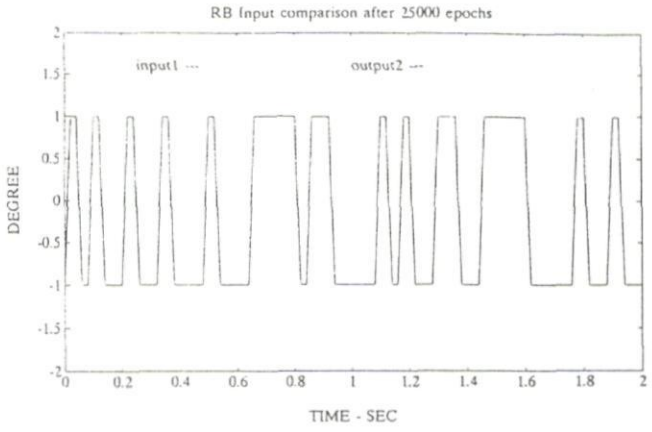


Figure 20 Input-Output Comparison

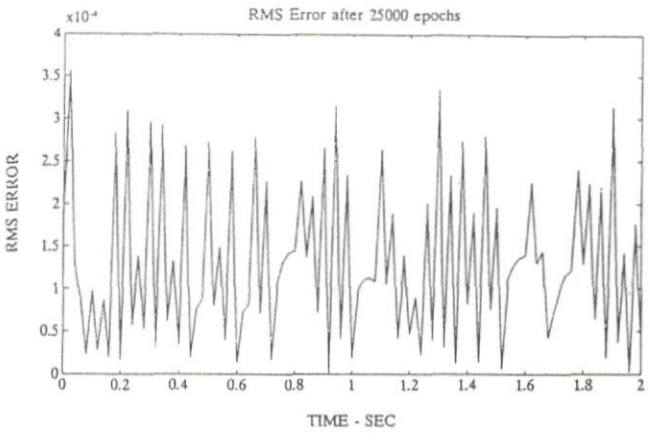


Figure 21 RMS Error Input-Ouput

10. IONOSPHERIC MODELING

Over the horizon radar is possible by reflecting radar beams off the ionosphere. To be effective it is necessary to have a model of the ionosphere. Three ionospheric layers appear quite regularly E, F_1, F_2. Neural networks [14] can be used to model the ionosphere. In this example we will highlight some of the data handling procedures necessary to the proper training of a neural network. Available for analysis were 146 radar soundings of the ionosphere. These soundings were taken every 10 minutes during a 24 hour period.

To determine the effectiveness of a neural network one needs a test set of data different from that of the training set. The data was divided into a training set of 74 exemplers and a test set of 72 exemplers.

Care must be taken in this division that all features that characterize the data are contained in the training set. A neural network cannot recognize a feature it has not been trained on. Each record consisted of an input with two time delays (or echos) for each of 449 frequencies that were taken. Thus the input layer had 889 processing elements while the output layer returned E, F_1, F_2 layers critical frequency and F_2 layer peak height determined by an expert. Care must also be taken that one does not overtrain the network and thus lose the ability of the network to generalize. This can be checked by plotting the error both on the training set and the test set as a function of training cycles. The training set error should continually decrease to some plateau value. The test global error may also at first decrease but when overtraining begins the error on the test set will begin to increase. Training should stop just before this effect starts.

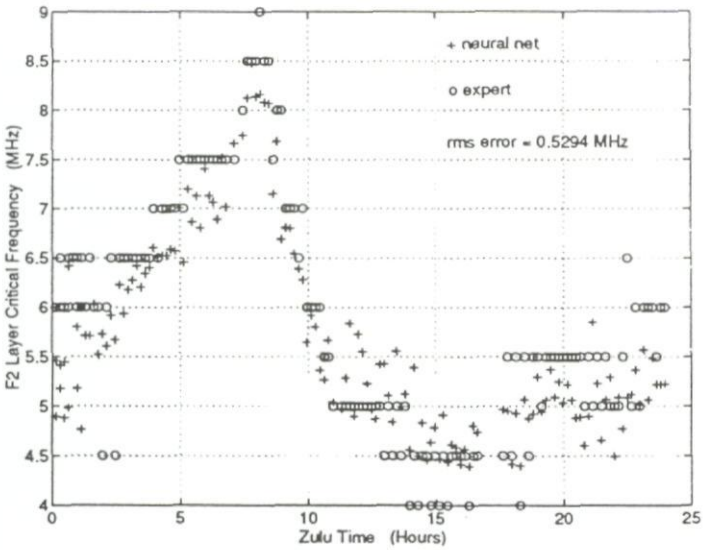


Figure 22 - F_2 Layer - Model Network

Figure 22 shows an example of the ANN prediction of F_2 layer critical frequency as a function of Zulu time for the test set. Predictions for the F_1 and F_2 peak were not as good. The fault may, however, be not with the neural network but with the expert modeling of these quantities.

11. SUMMARY

This article has given a brief introduction to neural networks. Feedforward and recurrent networks have been discussed. Supervised and Unsupervised learning outline. Several applications of neural networks to adaptive control and modeling have been highlighted.

REFERENCES

1. AGARD Lecture Series 179
Artificial Neural Network Approaches in Guidance and Control, September 1991.
2. Rumelhart, D.E. and McClelland, J.L.
Parallel Distributed Processing, Vol. I, Vol. II. The MIT Press, Cambridge, Massashusetts 1986.
3. Bose, N.K. and Liang, P.
Neural Networks Fundamentals With Graph, Algorithms, and Applications, McGraw-Hill, Inc., 1996.
4. Haykin, S.
Neural Networks- A Comprehensive Foundation, MacMillan College Publishing Company, NY 1994
5. Rosenblatt, F.
Principles of Neurodynamics, Spartan Press, Washington, DC 1961
6. Widrow, B. and Hoff, M.E.
Adaptive Switching Circuits 1960 IRE WESCON Convention, Record, NY IRE Part 4, pp. 96-104, 1960
7. Hopfield, J.J.
Neural Networks and Physical System With Emergent Collective Computational Properties. Proceeding of National Academy of Sciences, Vol. 79, pp. 2354-2558, 1982.
8. Rumelhart, D.E. and Zipzer, D.
Feature Discovery by Competitive Learning. Cognitive Science 9, pp. 75-112, 1985.
9. Kohonen, T.
Self-Organization and Associative Memory Springer-Verlag, New York, 1989.
10. Linde, Y., Buzo, A. and Gray, R.M.
An Algorithm for Vector Quantizer Design IEEE Transactions on Communication COM-28 84-95 1980.
11. Hopfield, J. and Tank, D.
"Neural" Computation of Decisions in Optimization Problems.
Biological Cybernetics, vol. 52, pp 141-152, 1985.
12. Brunger, C.A.
Artificial Neural Network Modeling of Damaged Aircraft, Naval Postgraduate School, Monterey, CA, MS Degree, March 1994.
13. Bertrand, D.J.S.R.
Application of Neural Network to Adaptive Control Theory for Super Augmented Aircraft, Naval Postgraduate School, Monterey, CA, MS Degree, December 1991.
14. Pinkepank, J.A.
The Applicability of Neural Networks to Ionospheric Modeling in Support of Relocatable Over-the-Horizon Radar, Naval Postgraduate School, Monterey, CA, MS Degree, September 1994.

Fundamentals of Fuzzy Logic, fuzzy inference and fuzzy control

B. Bouchon-Meunier
LIP6, CNRS-UPMC
Case 169, 4 place Jussieu
75252 Paris Cédex 05, France
bouchon@laforia.ibp.fr

1. INTRODUCTION

Fuzzy set theory was introduced 30 years ago by L.A. Zadeh, with the purpose of formalizing the representation and management of imprecise or approximate knowledge, in order to facilitate the management of very complex systems, for instance systems involving human components. It provides tools to process imperfect information.

There exist several types of imperfect knowledge:

- uncertain information: there exist a doubt about the validity of the piece of information, for instance "it may rain this afternoon". Uncertainty is mainly due to a random behavior of some phenomenon or to a relative reliability of an observer or a device providing the information.

- imprecise information: the information is difficult to express clearly or precisely, for instance "it rained most of the afternoon", either because the measurement of a quantity is not precise enough (no count of the rain duration), or because a piece of knowledge has no precise value ("he cannot go far" cannot be considered equivalent to any precise information such as "he cannot go farther than 300 meters")

- incomplete information: some piece of information is missing, for instance the elements of an image hidden by a spot on the image or a list of exceptions to a given general rule, for instance "with certain exceptions, the battalion commander arrives every day at 8:00". Incompleteness entails uncertainty about the hidden or missing elements.

Fuzzy set theory is a tool to manage imprecise information and, associated with possibility theory, it enables to manage subjective uncertain information. Some incomplete information can also be approached by fuzzy set and possibility theory.

The purpose of fuzzy set theory is to allow graded membership of an element to a given class and to avoid breaks between close situations. It allows to deal with partial membership to a class, ill-defined categories, gradual knowledge, heterogenous types of data (symbolic and numerical data in a given system). Fuzzy logic is specially interesting to use in the case of complex systems, and particularly systems with a human component.

In this paper, we introduce the basic elements of fuzzy set theory : definition of fuzzy sets, various operations on fuzzy sets, fuzzy relations, fuzzy arithmetics. We present possibility theory which allows to manage non-probabilistic uncertainty and leads to a form of reasoning on imperfect knowledge when used in synergy with fuzzy set theory. We present the bases of this approximate reasoning and we show how it can be used in inferences or in control.

2. DEFINITION OF FUZZY SETS

For a given universe X, a classical subset C is defined by a characteristic function χ_C lying in {0, 1}, while a *fuzzy set* A is defined by a membership function

$$f_A : X \dashrightarrow [0, 1]$$

such that $f_A(x)$ indicates the degree of membership of the element x of X to the fuzzy set A. A classical (or *crisp*) subset of X is then a particular case of fuzzy set. The following notation is classical:

$$A = \sum_{x \in X} f_A(x) / x, \text{ if X is countable,}$$
$$A = \int_X f_A(x) / x, \text{ otherwise.}$$

Some particular elements are interesting to describe a fuzzy set :

- its support: $\text{supp}(A) = \{x \in X / \ f_A(x) \neq 0\}$

- its height: $h(A) = \sup_{x \in X} f_A(x)$

- its kernel or core: $\ker(A) = \{x \in X / \ f_A(x) = 1\}$. Fuzzy sets with a non-empty kernel and then a height equal to 1 are called *normalized.*

- its cardinality: $|A| = \sum_{x \in X} f_A(x)$.

We denote by $\mathbf{F}(X)$ the set of all fuzzy sets of X. Singletons are particular fuzzy sets A of X, very useful in fuzzy control, such that there exist an element a in X such that $f_A(a) = 1$ and $f_A(x) = 0$ for any $x \neq a$.

In the ***example*** given in Figure 1, with a continuous membership function, we have Ker(A) = [28, 32], supp(A) = [24, 36], h(A) = 1. An example on a discrete universe can be the following, regarding the choice of a location for some purpose, with X = {Wood, Field, Village}: A = 0.3 / W + 0.7 / F + 0.1 / V.

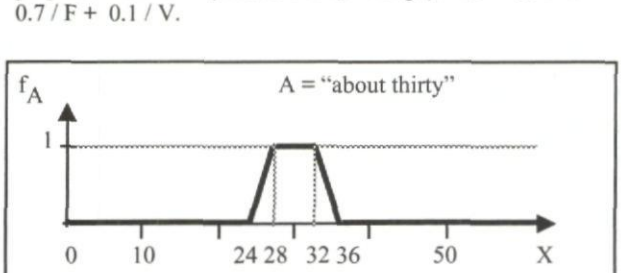


Figure 1. Example of a fuzzy set

Let us remark that a fuzzy set can have several interpretations, depending on the situation: partial membership ($f_A(x)$ is the membership degree of x to the class A), preferences ($f_A(x)$ is the degree of preference attached to x), typicality ($f_A(x)$ is the degree of typicality of x in the class A), possibility ($f_A(x)$ is the degree of possibility that x is the value of a variable defined on X).

3. OPERATIONS ON FUZZY SETS

Fuzzy set theory can be regarded as a generalization of classical set theory. It is then necessary to define operations on fuzzy sets extending the operations on crisp subsets of X.

Let A and B be two fuzzy sets of X. They are *equal* if and

Paper presented at the AGARD MSP Lecture Series on "Advances in Soft-Computing Technologies and Applications in Mission Systems", held in North York, Canada, 17-18 September 1997; Amsterdam, The Netherlands, 22-23 September 1997; Madrid, Spain, 25-26 September 1997; Ankara, Turkey, 6-7 October 1997, and published in LS-210.

only if

$$\forall x \in X \qquad f_A(x) \leq f_B(x).$$

A is *included* in B, and we note $A \subseteq B$, if and only if

$$\forall x \in X \qquad f_A(x) \leq f_B(x).$$

Inclusion defines a partial order on fuzzy sets of X, with the empty set ($\forall x \in X \ f_A(x) = 0$) as smallest element and the universe itself ($\forall x \in X \ f_A(x) = 1$) as greatest element.

3. 1. Basic definitions

The *intersection* of A and B is defined as the fuzzy set $C = A \cap B$ of X with the following membership function:

$$\forall x \in X \qquad f_C(x) = \min (f_A(x), f_B(x)).$$

The *union* of A and B is defined as the fuzzy set $D = A \cup B$ of X with the following membership function:

$$\forall x \in X \qquad f_D(x) = \max (f_A(x), f_B(x)).$$

The properties of intersection and union of crisp subsets of X are preserved by these definitions :

- associativity of $\cap$ and $\cup$,
- commutativity of $\cap$ and $\cup$,
- $A \cup \emptyset = A$, $A \cup X = X$,
- $A \cap X = A$, $A \cap \emptyset = \emptyset$,
- $A \cup B \supseteq A \supseteq A \cap B$,
- distributivity

$A \cap (B' \cup B'') = (A \cap B') \cup (A \cap B'')$

$A \cup (B' \cap B'') = (A \cup B') \cap (A \cup B'')$,

- $|A| + |B| = |A \cap B| + |A \cup B|$.

We now define the *complement* of a fuzzy set A of X as the fuzzy set A^c of X with the following membership function:

$$\forall x \in X \qquad f_{A^c}(x) = 1 - f_A(x).$$

This definition preserves most of the properties available in classical set theory, for instance:

- $(A \cap B)^c = A^c \cup B^c$,
- $(A \cup B)^c = A^c \cap B^c$,
- $(A^c)^c = A$,
- $|A| + |A^c| = |X|$.

The only properties of the complement of a fuzzy set which are different from their correspondent in classical set theory are the following :

- $A^c \cap A \neq \emptyset$
- $A^c \cup A \neq X$.

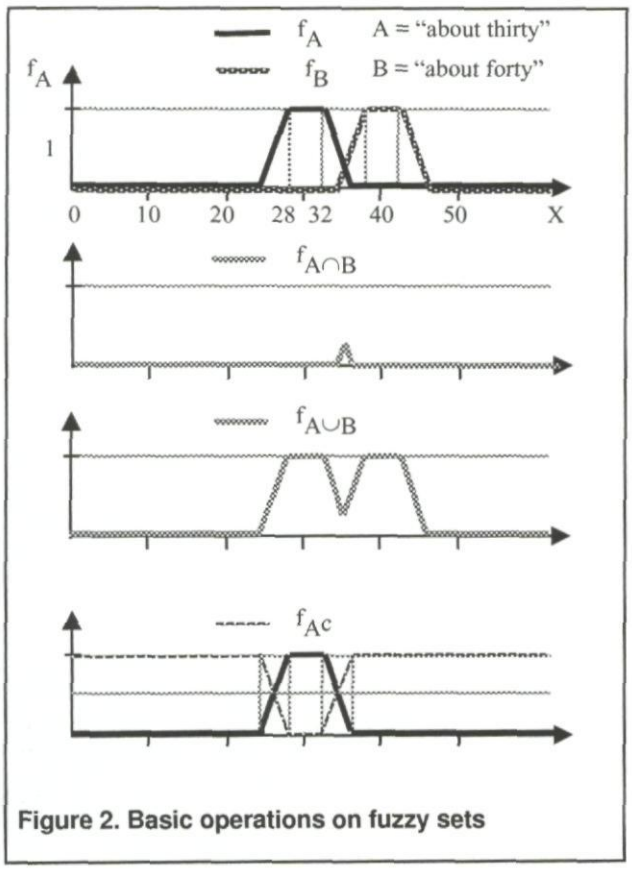


Figure 2. Basic operations on fuzzy sets

3. 2. Other definitions

The choice of operators min and max to define the intersection and union of fuzzy sets is justified by the fact that, together with the complement, they preserve almost all the properties satisfied in classical set theory. They have been pointed out as the solution of functional equations taking into account the most important chracteristics expected from a union and an intersection. Nevertheless, in some cases, it can be interesting to loose some other properties and to use definitions of intersection and union with a slightly different behavior.

The most common alternate operators are triangular norms T: $[0,1] \times [0,1] \rightarrow [0,1]$ (t-norms) to define the intersection and triangular conorms $\perp$: $[0,1] \times [0,1] \rightarrow [0,1]$ (t-conorms) to define the union. These operators have been introduced in probabilistic metric spaces and they have the following properties in common:

i) $T(x,y) = T(y,x)$, $\perp(x,y) = \perp(y,x)$ (commutativity)

ii) $T(x,T(y,z)) = T(T(x,y),z)$, $\perp(x,\perp(y,z)) = \perp(\perp(x,y),z)$ (associativity)

iii) $T(x,y) \leq T(z, t)$, $\perp(x,y) \leq \perp(z,t)$, if $x \leq z$ and $y \leq t$ (monotonicity)

Their neutral elements are different:

iv) $T(x, 1) = x$, $\perp(x, 0) = x$ for any x in $[0,1]$

It is easy to check that min is a t-norm and max a t-conorm. Intersection and union are defined as follows:

- $C = A \cap_T B$ with $f_C(x) = T(f_A(x), f_B(x))$
- $C = A \cup_T B$ with $f_C(x) = \perp (f_A(x), f_B(x))$.

The most classical t-norms and t-conorms are defined in Table 1.

t-norm	*t-conorm*	*name*
$\min(x,y)$	$\max(x,y)$	Zadeh
$x \cdot y$	$x+y-xy$	probabilistic
$\max(x+y-1, 0)$	$\min(x+y, 1)$	Lukasiewicz
$\frac{xy}{\gamma+(1-\gamma)(x+y-xy)}$	$\frac{x+y-xy-(1-\gamma)xy}{1-(1-\gamma)xy}$	Hamacher ($\gamma > 0$)
x si y=1 y si x=1 0 sinon	x si y=0 y si x=0 1 sinon	Weber

Table 1. Main dual t-norms and t-conorms

They satisfy:

- $T_{Weber}(x,y) \le T(x,y) \le \min(x,y)$
- $\max(x,y) \le \perp(x,y) \le \perp_{Weber}(x,y)$.
- The Lukasiewicz operators satisfy

$A^c \cap A = \emptyset$, $A^c \cup A = X$, $A \cap A \neq A$, $A \cup A \neq A$

- The probabilistic operators satisfy

$A^c \cap A \neq \emptyset$, $A^c \cup A \neq X$, $A \cap A \neq A$, $A \cup A \neq A$.

Pairs of t-norms and t-conorms can be considered as dual in the following sense, preserving the De Morgan laws for the corresponding fuzzy set operations:

- $1-T(x, y) = \perp(1-x, 1-y)$
- $1-\perp(x, y) = T(1-x, 1-y)$.

4. ALPHA-CUTS OF FUZZY SETS

It is sometimes necessary to associate fuzzy sets to crisp sets, in order to make a decision for instance. The most classical way is to choose a threshold α in [0, 1], considered as a lower bound of the membership degrees taken into consideration. The *α–cut* A_α of a fuzzy set A of the universe X is defined as:

$A_\alpha = \{x \in X / f_A(x) \ge \alpha\}$.

If we increase the threshold, we are more exacting with regard to the membership of an element to the considered class and less elements of X correspond to our demand. This means that the α-cuts are nested:

if $\alpha' \ge \alpha$ then $A_\alpha \supseteq A_{\alpha'}$.

The following properties are satisfied by α-cuts :

- $(A \cap B)_\alpha = A_\alpha \cap B_\alpha$,
- $(A \cup B)_\alpha = A_\alpha \cup B_\alpha$,
- if $A \supseteq B$ then $A_\alpha \supseteq B_\alpha$
- $A_0 = X$, $A_1 = \mathrm{Ker}(A)$.

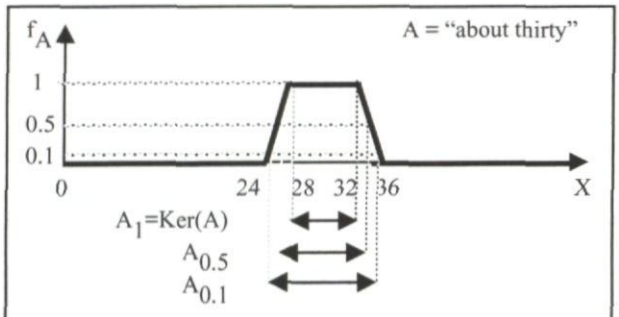


Figure 3. Some α -cuts of the fuzzy set A

The family of all the α-cuts of a given fuzzy set A of X provides a representation of A by means of the following theorem:

$\forall x \in X \quad f_A(x) = \sup_{\alpha \in]0,1]} \alpha \cdot \chi_{A_\alpha}(x)$,

where χ_{A_α} is the characteristic function of the α-cut A_α.

5. CARTESIAN PRODUCT AND PROJECTION OF FUZZY SETS

When several universes are considered simultaneously, for instance several criteria to make a decision, attributes to describe an object, variables to control a system, it is necessary to define the cartesian product of fuzzy sets of the various universes.

Let us consider universes $X_1, X_2, \ldots, X_r$, their cartesian product $X = X_1 \times X_2 \times \ldots \times X_r$, the elements of which are r-tuples $(x_1, x_2, \ldots, x_r)$, with $x_1 \in X_1, x_2 \in X_2, \ldots, x_r \in X_r$.

From fuzzy sets $A_1, A_2, \ldots, A_r$, respectively defined on $X_1, X_2, \ldots, X_r$, we construct a fuzzy set of X denoted by $A = A_1 \times A_2 \times \ldots \times A_r$, considered as their *cartesian product*, with membership function:

$\forall x=(x_1, \ldots, x_r) \in X, f_A(x) = \min(f_{A1}(x_1), \ldots, f_{Ar}(x_r))$.

Let us consider again the choice of a location, X_1 = {Wood, Field, Village}, with $A_1 = 0.3 / W + 0.7 / F + 0.1 / V$, and X_2 = {North of the road, South of the road}, with $A_2 = 0.6 / N + 0.4 / S$. A global view of the preferences on the cartesian product $X = X_1 \times X_2$ is given by the cartesian product of A_1 and A_2:

$A = 0.3 / (W,N) + 0.3 / (W,S) + 0.6 / (F,N) + 0.4 / (F,S) + 0.1 / (V,N) + 0.1 / (V,S)$

The converse operation provides a description on some of the available universes, given a global description on their universes.

Let us consider again universes $X_1, X_2, \ldots, X_r$, their cartesian product X. We suppose that we are given a fuzzy set A of X. The *projection* of A on $X_a \times X_b \times \ldots \times X_k$, with $1 \le a < b < \ldots < k \le r$, denoted by Proj(A), is defined by:

$\forall x_a \in X_a \ldots x_k \in X_k$,

$f_{Proj(A)}(x_a \ldots x_k) = \sup_{1 \le i \le r,\ i \neq a \ldots j \neq k} f_A((x_i, \ldots, x_j))$.

More particularly, if we restrict ourselves to two universes (r = 2), the *projection* of A on X_1 is defined by:

$\forall x_1 \in X_1, f_{Proj X_1(A)}(x_1) = \sup_{x_2 \in X_2} f_A((x_1, x_2))$.

Let us remark that, in the case of normalized fuzzy sets, if A is the cartesian product $A_1 \times A_2$, then its projection on X_1 is A_1 and its projection on X_2 is A_2.

For ***example***, if we have a global description of preferences $A = 0.3 / (W,N) + 0.5 / (W,S) + 0.6 / (F,N) + 0.4 / (F,S) + 0.1 / (V,N) + 0.2 / (V,S)$, we can deduce two marginal descriptions of preferences

$A_1 = 0.5 / W + 0.6 / F + 0.2 / V$ on X_1,

$A_2 = 0.6 / N + 0.5 / S$ on X_2.

6. THE ZADEH EXTENSION PRINCIPLE

Fuzzy sets of X are imperfect information about the elements of X. For instance, instead of observing x precisely, we can only perceive a fuzzy set of X with a high membership degree attached to x. The methods which would be available to manage the information regarding X in the case of precise information need to be adapted to be able to manage fuzzy sets.

Let us consider a mapping φ from a first universe X to a second one Y, which can be identical with X. The *Zadeh extension principle* defines a fuzzy set B of Y from a fuzzy set A of X, in agreement with the mapping φ, in the following way (figure 4):

$\forall y \in Y \quad f_B(y) = \sup_{x \in \varphi^*(y)} f_A(x) \quad$ if $\varphi^*(y) \neq \emptyset$

$f_B(y) = 0 \quad$ if $\varphi^*(y) = \emptyset$,

with $\varphi^*(y) = \{x \in X / y = \varphi(x)\}$ if $\varphi : X \to Y$,

and $\varphi^*(y) = \{x \in X / y \in \varphi(x)\}$ if $\varphi : X \to 2^Y$ (i.e φ is multivalued).

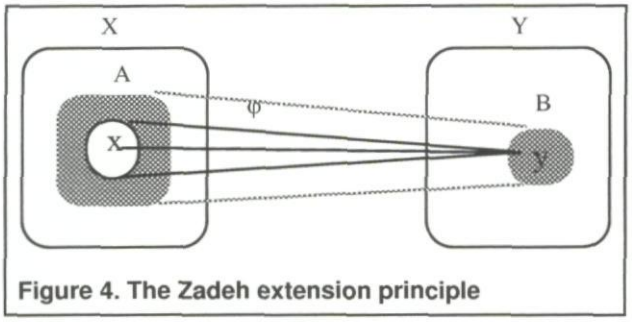


Figure 4. The Zadeh extension principle

If A is a crisp subset of X reduced to a singleton {a}, the Zadeh extension principle constructs a fuzzy set B of Y reduced to $\varphi(\{a\})$.

If φ is a one-to-one mapping, then

$\forall y \in Y \quad f_B(y) = f_A(\varphi^{-1}(y))$.

If X is the cartesian product of universes $X_1 \times X_2 \times \ldots \times X_r$ and A is the cartesian product of fuzzy sets of these universes $A = A_1 \times A_2 \times \ldots \times A_r$, the Zadeh extension principle associates a fuzzy set B of Y with A as follows:

$\forall y \in Y$

$f_B(y) = \sup_{x \in \varphi^*(y)} \min (f_{A1}(x_1), \ldots, f_{Ar}(x_r))$ if $\varphi^*(y) \neq \emptyset$

$f_B(y) = 0$ if $\varphi^*(y) = \emptyset$.

For ***example***, let us consider the fuzzy set A representing "about thirty" on the universe $X = \mathbb{R}^+$, as defined in figure 1. If we know that the value of variable W defined on X is greater than the value of variable V and that the value of V is about thirty, we can characterize the value of W by the fuzzy set B obtained by applying the extension principle to the *order* relation on $\mathbb{R}$. We have $Y = \mathbb{R}^+$ and $\varphi(x) = \{y \in Y / y \geq x\}$. We get :

$\forall y \in Y \quad f_B(y) = \sup_{\{x \in X / y \geq x\}} f_A(x)$,

which yields $f_B(y) = 0$ if $x \leq 24$, $f_B(y) = y/4 - 6$ if $24 \leq y \leq 28$, and $f_B(y) = 1$ if $y \geq 28$.

Another ***example*** of application of the extension principle defines a *distance* between imprecise locations. Let us consider a set of points $Z = \{a, b, c, d\}$. The distance between any pair of points of Z is defined by a mapping $\varphi: Z \times Z \rightarrow \mathbb{R}^+$. If the points are observed imprecisely, we need to extend the notion of distance to fuzzy sets. We use the extension principle with $X = Z \times Z$ and $Y = \mathbb{R}^+$, and we get a fuzzy set C of $\mathbb{R}^+$ with its membership function defined for any $d \in \mathbb{R}^+$ by :

$f_C(d) = \max_{\{(x, y) \in X, x \neq y, \varphi(x, y) = d\}} \min (f_A(x), f_B(y))$
if $\{(x, y) \in X, x \neq y, \varphi(x, y) = d\} \neq \emptyset$

$f_C(d) = 0$ sinon.

For ***example***, let us suppose that the distance is defined by $\varphi(a, b) = \varphi(a, c) = 4$, $\varphi(a, d) = 5$, $\varphi(b, c) = 2$, $\varphi(b, d) = \varphi(c, d) = 3$. If we have imprecise locations in Z defined as $A = 0.8 / a + 0.1 / b + 0.4 / c + 0.1 / d$ and $B = 0.2 / a + 0.3 / b + 0.7 / c + 0.5 / d$, we obtain a description of the distance between A and B through the following fuzzy set of $\mathbb{R}^+$:

$f_C(d) = 0 / 0 + 0 / 1 + 0.3 / 2 + 0.4 / 3 + 0.7 / 4 + 0.5 / 5 + 0 / 6 + 0 / 7 + \ldots$,

which means that the distance between A and B is around 4.

The Zadeh extension principle is fundamental to extend to fuzzy sets all the concepts we are familiar with in classical set theory, for instance in arithmetics or reasoning.

7. FUZZY RELATIONS

Since fuzzy set theory represents a generalization of classical set theory, we need to generalize all the classical tools available to manage crisp data. Fuzzy relations are among the most important concepts in fuzzy set theory.

A *fuzzy relation* R between X and Y is defined as a fuzzy set of $X \times Y$. In the particular case where X and Y are finite, R can be described by means of the matrix M(R) of its membership degrees.

An ***example*** of fuzzy relation on $X = Y = \{x_1, x_2, x_3\}$ is defined by the following matrix $M(R_o)$:

x \ y	x_1	x_2	x_3
x_1	0.2	1	0.5
x_2	0	0.6	0.3
x_3	0	0.9	0.4

Another ***example*** of fuzzy relation can be defined on $X = Y = \mathbb{R}$ to represent approximate equality between real values, for instance with the following membership function (see Figure 5):

$\forall x \in X, \forall y \in X \quad f_{Re}(x,y) = 1/(1+(x-y)^2)$.

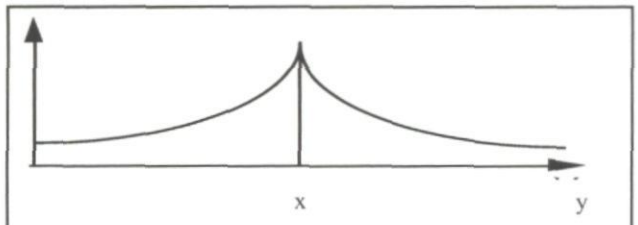


Figure 5. Representation of "approximately equal to"

7. 1. Combination of fuzzy relations

The *inverse of* a given relation R between X and Y is the fuzzy relation R^{-1} between Y and X defined by:

$\forall x \in X, \forall y \in Y \quad f_{R^{-1}}(y, x) = f_R(x, y)$.

If we have 3 universes X, Y et Z, it is useful to combine fuzzy relations between them. The *composition* of two fuzzy relations R_1 on $X \times Y$ and R_2 on $Y \times Z$ defines a fuzzy relation $R = R_1 \circ R_2$ on $X \times Z$, with membership function:

$\forall (x, z) \in X \times Z \quad f_R(x, z) = \sup_{y \in Y} \min(f_{R_1}(x, y), f_{R_2}(y, z))$.

This *max-min composition* is the basic way of combining fuzzy relations. If we have special requirements about the behavior of the combined fuzzy relation, it is possible to replace the operator min by another t-norm * (*max-* composition*).

This definition is compatible with the classical composition of relations when R_1 and R_2 are not fuzzy. It is particularly easy to compute when the universes are finite, since the calculus of $M(R)$ from $M(R_1)$ and $M(R_2)$ can then be regarded as a matrix product, with the operators max replacing the addition and min replacing the product.

With the ***example*** of relation R_o above-mentioned, we obtain $R_o \circ R_o$ with the following matrix $M(R_o \circ R_o)$:

x \ y	x_1	x_2	x_3
x_1	0.2	0.6	0.4
x_2	0	0.6	0.3
x_3	0	0.6	0.4

7. 2. Properties of fuzzy relations

The main utilizations of fuzzy relations concern the representation of resemblances ("almost equal") or orders ("really smaller"). We need to define general classes of fuzzy relations suitable for such representations, based on particular properties of fuzzy relations.

The fuzzy relation is *symmetrical* if:

$\forall(x, y) \in X \times X \quad f_R(x, y) = f_R(y, x).$

It is *reflexive* if:

$\forall\, x \in X \quad f_R(x, x) = 1.$

It is *transitive* if:

$R \supseteq R \circ R,$

and more particularly *max-min transitive* if we use the max-min composition of fuzzy relations:

$\forall(x, z) \in X \times X \quad f_R(x,z) \geq \sup_{y \in X} \min(f_R(x,y), f_R(y,z)).$

The fuzzy relation is *antisymmetrical* if:

$\forall(x, y) \in X \times X, f_R(x, y) > 0$ and $f_R(y, x) > 0 \Rightarrow x = y.$

The property of *max-min transitivity* is particularly easy to check when X is finite, since we have to check that each element of the matrix M(R o R) is not greater than the corresponding element of M(R).

For instance, the relation R_o is transitive. The fuzzy relation R_e is reflexive and symmetrical.

For any threshold $\alpha \in]0, 1]$, we define the α-cut R_α of R, such that $f_{R_\alpha}(x,y) = 1$ if $f_R(x,y) \geq \alpha$ and $f_{R_\alpha}(x,y) = 0$ otherwise. It can be proven that R is symmetrical (respectively max-min transitive) if and only if R_α is a symmetrical (respectively transitive) crisp relation, for any $\alpha \in]0, 1]$. This proves that the definitions we have chosen are compatible with the classical definitions.

7. 3. Similarity relations

A *similarity relation* is a symmetrical, reflexive and max-min transitive fuzzy relation.

It corresponds to the idea of resemblance.

For any threshold $\alpha \in]0, 1]$, the α-cut R_α of R is a classical equivalence relation on X. When α varies from 0 to 1, we obtain nested partitions of X from the equivalence classes of R_α. For any level α, x et y belong to the same class of the partition if and only if they resemble each other with a degree at least equal to α.

If the similarity relation R is such that $f_R(x,y) = 1$ if and only if x=y, we can associate it with a distance d defined on X and lying in [0, 1] as follows:

$\forall(x, y) \in X \times X \quad d(x,y) = 1 - f_R(x,y).$

This distance is ultrametric since it satisfies:

$\forall(x, y, z) \in X \times X \times X \quad d(x,y) \leq \max(d(x,z), d(z,y)).$

Similarity relations are mainly involved in classification and clustering.

An ***example*** of similarity relation is given in Figure 6.

x \ y	x_1	x_2	x_3	x_4
x_1	1	0.3	0.9	0.5
x_2	0.3	1	0.3	0.3
x_3	0.9	0.3	1	0.5
x_4	0.5	0.3	0.5	1

M(R) = M(R o R)

α-cuts of R

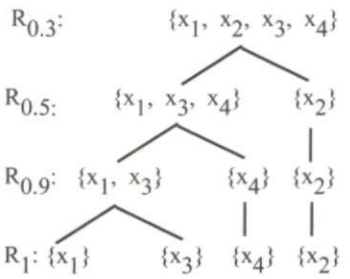


Figure 6. Example of similarity relation and partitions of alpha-cuts

7. 4. Fuzzy order relations

A *fuzzy preorder* is a reflexive and transitive fuzzy relation R.

If R is also antisymmetrical, it is a *fuzzy order relation.* It corresponds to the idea of ordering or anteriority.

For any threshold $\alpha \in]0, 1]$, the α-cut R_α of an order fuzzy relation R is a partial order on X.

An ***example*** of fuzzy order relation is given by the following matrix M(R):

$$M(R) = \begin{array}{c} \begin{matrix} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \end{matrix} \\ \begin{bmatrix} 1 & 0.8 & 0.2 & 0.6 & 0.6 & 0.4 \\ 0 & 1 & 0 & 0 & 0.6 & 0 \\ 0 & 0 & 1 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 1 & 0.6 & 0.4 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{matrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{matrix} \end{array}$$

Fuzzy order relations are useful in decision-making, for instance for the analysis of preferences or for temporal ordering.

8. FUZZY ARITHMETICS

Many applications use the universe $\mathbb{R}$ of real numbers, with fuzzy sets representing imprecise measurements of real-valued variables (distance, weight...). The membership function are generally chosen as simple as possible, compatible with the intuitive representation of approximations. This simple form of functions corresponds to convex fuzzy sets.

A fuzzy set on X is *convex* if it satisfies the following condition:

$\forall(x,y) \in \mathbb{R} \times \mathbb{R}, \forall z \in [x, y] \quad f_F(z)) \geq \min(f_F(x), f_F(y)).$

A fuzzy set is convex if and only if its α-cuts are convex subsets of $\mathbb{R}$.

A *fuzzy quantity* Q is a normalized fuzzy set of $\mathbb{R}$.

A *modal value* of Q is an element m of $\mathbb{R}$ in the kernel of Q, such that $f_Q(m) = 1$.

A *fuzzy interval* I is a convex fuzzy quantity. It corresponds to an interval of $\mathbb{R}$ with imprecise boundaries.

If its membership function is upper semi-continuous, its α-cuts are closed intervals of $\mathbb{R}$ for any $\alpha \in]0, 1]$.

A *fuzzy number* M is a fuzzy interval with an upper semi-continuous membership function, a compact support and a unique modal value. It corresponds to an imprecisely known

real value.

8. 1. Main arithmetic operations

It is often necessary to compute addition or product of imprecisely known real values. For instance, the number of present elements is approximately 30 and approximately 20 are expected, what will be the resulting number of elements? Symbolically, we can conclude that it will be approximately 50, but we need to formalize this operation to define automatic operations for more complex problems, such as fuzzy operational research, for instance. We use the Zadeh extension principle to extend the classical arithmetic operations to fuzzy quantities.

A *unary operation* Δ can be defined on fuzzy quantities from a unary operation φ defined on $\mathbb{R}$. For any fuzzy quantity Q, it yields another fuzzy quantity ΔQ with the following membership function:

$\forall z \in \mathbb{R}$ $f_{\Delta Q}(z) = \sup_{z=\varphi x} f_Q(x)$.

Particular unary operations provide :

- the *opposite* -Q of a fuzzy quantity Q, such that:

$\forall z \in \mathbb{R}$ $f_{-Q}(z) = f_Q(-z)$

- the *inverse* 1/Q of Q, such that:

$\forall z \in \mathbb{R}$, $z \neq 0$ $f_{1/Q}(z) = f_Q(1/z)$.

- the *exponent* of Q, denoted by exp(Q), such that

$\forall z \in \mathbb{R}$ $f_{\exp(Q)}(z) = f_Q(\ln z)$

- the *product by a scalar* λ different from 0, denoted by λQ, such that

$\forall z \in \mathbb{R}$ $f_{\lambda Q}(z) = f_Q(z/\lambda)$.

For ***example***, if Q is a fuzzy number with modal value 4 and support [3, 5], -Q is a fuzzy number with modal value -4 and support [-5, -3], 1/Q is a fuzzy number with modal value 1/4 and support [1/5, 1/3].

A *binary operation* * can be defined for fuzzy quantities for any binary operation ψ on $\mathbb{R}$. From two fuzzy quantities Q and Q', it yields another fuzzy quantity Q*Q' with membership function:

$\forall z \in \mathbb{R}$ $f_{Q*Q'}(z) = \sup_{\{(x,y)/\ z=x\psi y\}} \min(f_Q(x), f_{Q'}(y))$.

If the operation ψ is commutative, * is also commutative.

If the operation ψ is associative, * is also associative.

Particular binary operations provide the following fuzzy operations:

- the *addition* of fuzzy quantities Q and Q', denoted by $Q \oplus Q'$, such that

$f_{Q \oplus Q'}(z) = \sup_{\{(x,y)\ /\ z=x+y\}} \min(f_Q(x), f_{Q'}(y))$,

- the *product* of fuzzy quantities Q and Q', denoted by $Q \otimes Q'$, such that:

$f_{Q \otimes Q'}(z) = \sup_{\{(x,y)\ /\ z=x \times y\}} \min(f_Q(x), f_{Q'}(y))$

- the *subtraction* of fuzzy quantity Q' to Q, denoted by $Q \ominus Q'$, such that:

$f_{Q \ominus Q'}(z) = \sup_{\{(x,y)\ /z=x-y\}} \min(f_Q(x), f_{Q'}(y))$

- the *quotient* of fuzzy quantity Q by Q', denoted by $Q \oslash Q$, such that:

$f_{Q \oslash Q'}(z) = \sup_{\{(x,y)\ /z=x/y\}} \min(f_Q(x), f_{Q'}(y))$

- the *maximum* of fuzzy quantities Q and Q', denoted by max(Q, Q'), such that:

$f_{\max(Q, Q')}(z) = \sup_{\{(x,y)\ /\ z = \max(x, y)\}} \min(f_Q(x), f_{Q'}(y))$

- the *minimum* of fuzzy quantities Q and Q', denoted by min(Q, Q'), such that:

$f_{\min(Q, Q')}(z) = \sup_{\{(x, y)\ /\ z = \min(x, y)\}} \min(f_Q(x), f_{Q'}(y))$

These operations are such that

- $Q \ominus Q' = Q \oplus (-Q')$
- $Q \oslash Q' = Q \otimes (1/Q')$.
- $Q \otimes Q' = (-Q) \otimes (-Q')$
- $(-Q) \otimes Q' = Q \otimes (-Q') = -(Q \otimes Q')$
- $Q \oplus 0 = 0 \oplus Q = Q$
- $Q \otimes 1 = 1 \otimes Q = Q$
- $Q \otimes 0 = 0 \otimes Q = 0$
- $Q \otimes (Q' \oplus Q'') \leq (Q \otimes Q') \oplus (Q \otimes Q'')$,

with equality if the supports of Q' and Q'' are both positive or both negative.

- $Q \otimes (1 / Q) \neq 1$ (1 is the modal value of $Q \otimes (1 / Q)$)
- $Q \oplus (-Q) \neq 0$ (0 is the modal value of $Q \oplus (-Q)$)
- max(Q, Q') = Q if and only if min(Q, Q') = Q'.

8. 2. L-R fuzzy numbers

It is interesting to use particular forms of membership functions for which the main operations are easily computable. It is the case for so-called L-R fuzzy intervals.

An *L-R fuzzy interval* I is a fuzzy quantity with a membership function f_I defined by means of four real parameters (m, m', a, b), with a and b strictly positive, and two functions L et R, defined on $\mathbb{R}^+$, lying in [0, 1], upper semi-continuous, non-increasing, such that:

- L(0) = R(0) = 1
- L(1) = 0 or L(x)>0 $\forall x$ with $\lim_{x \to \infty} L(x) = 0$,
- R(1) = 0 ou R(x)>0 $\forall x$ avec $\lim_{x \to \infty} R(x) = 0$.

The membership function of an L-R fuzzy interval defined by m, m', a and b is then :

$f_I(x) = L((m-x)/a)$ if $x \leq m$,

$f_I(x) = 1$ if $m < x < m'$,

$f_I(x) = R((x-m')/b)$ if $x \geq m'$,

We note $I = (m, m', a, b)_{LR}$. It can be interpreted as "approximately between m and m' ".

The particular case of an L-R fuzzy interval $(n, n, a, b)_{LR}$ is an *L-R fuzzy number* denoted by $M = (n, a, b)_{LR}$. It can be interpreted as "approximately n".

Fuzzy quantities have often trapezoidal or triangular membership functions (figure 7). They are then L-R fuzzy intervals or numbers, with R(x) = L(x) = max (0, 1-x). It is also possible to use functions such as $\max(0, 1-x^2)$, $\max(0, 1-x)^2$ or exp(-x) for instance to define R and L.

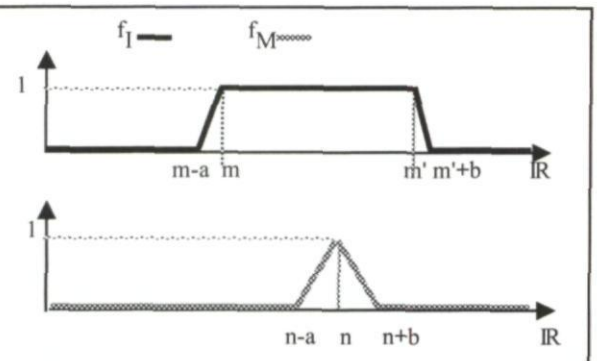


Figure 7. Triangular fuzzy number M and trapezoidal fuzzy interval I

Given two L-R fuzzy intervals defined by the same functions L and R, denoted by
$I = (m, m', a, b)_{LR}$ and $J = (n, n', c, d)_{LR}$, the main unary and binary operations defined in section 8.1:

$-I = (-m', -m, b, a)_{RL}$

$I \oplus J = (m+n, m'+n', a+c, b+d)_{LR}$

$I \ominus J = (m-n', m'-n, a+d, b+c)_{LR}$ if L=R

$I \otimes J$ is generally not an L-R fuzzy interval, but it is possible to approximate it by the following L-R fuzzy interval :

$I \otimes J = (mn, m'n', mc+na, md+nb)_{LR}$,

when the supports of I and J are included in $\mathbb{R}^+$ and a and b are small with respect to m, c and d are small with respect to n.

8. 3. Fuzzy relations on fuzzy quantities

Fuzzy values of variables can be compared by means of fuzzy relations, weakened versions of equality or order for instance.

Let V and W be two variables, N a fuzzy quantity which is the values of V, R a relation between the value of V and the value of W. We can deduce the value M of W as follows:

$\forall x \in \mathbb{R}\ f_M(x) = \sup_{y \in \mathbb{R}} \min (f_R(x, y), f_N(y))$.

For ***example***, let us suppose that the relation "y is really greater than x" between two real numbers x and y is represented by the fuzzy relation R with membership function

$\forall (x,y) \in \mathbb{R}^2\ f_R(x,y) = \min (1, (y-x) / ß)$ if $y \geq x$, $ß \geq 0$,
$= 0$ sinon,

for a positive parameter ß .

If the length of a field is really greater than its width, and the value of the length is given by a fuzzy triangular number N = (n, a, a), for $0<a<n$, then we can deduce the value of the width as a fuzzy trapezoidal number M = (0, n-ß, 0, a+ß) (Figure 8).

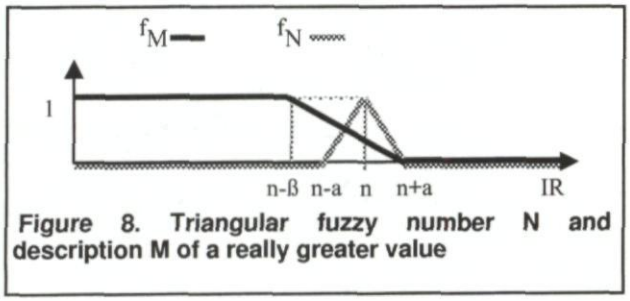


Figure 8. Triangular fuzzy number N and description M of a really greater value

9. MEASURES OF POSSIBILITY AND NECESSITY

Fuzzy set theory provides a representation of imprecise knowledge. It does not present any immediate representation of uncertain knowledge, which is nevertheless necessary to reason with imprecise knowledge. Let us consider the precise and certain rule "if the purchase is made after the 20th of a month, it will be paid at the end of next month". An imprecise information such as "the purchase is made around the 20th of a month" leads to an uncertain conclusion "we are not certain that the purchase will be paid at the end of next month". This simple example proves that imprecision and uncertainty are closely related.

Possibility theory has been introduced in 1978 by L.A. Zadeh to represent non-probabilistic uncertainty linked with imprecise information, in order to enable reasoning on imperfect knowledge. It is based on two measures defined for any subset of a given universe X, the possibility and the necessity measure.

9. 1. Possibility measure

We define a *possibility measure* as a mapping $\Pi : [0,1]^X \rightarrow [0, 1]$, such that:

i) $\Pi(\emptyset) = 0$, $\Pi(X) = 1$,

ii) $\forall A_1 \in [0,1]^X, A_2 \in [0,1]^X \ldots$

$\Pi(\cup_{i=1,2\ldots} A_i) = \sup_{i=1,2\ldots} \Pi(A_i)$,

In the case of a finite universe X, we can reduce ii) to ii') which is a particular case of ii) for any X:

ii') $\forall (A, B) \in [0,1]^X \times [0,1]^X\ \Pi(A \cup B) = \max (\Pi(A), \Pi(B))$.

$\Pi(A)$ represents to which extent it is possible that the subset (or event) A of X occurs. If $\Pi(A) = 0$, A is impossible, if $\Pi(A) = 1$, A is absolutely possible.

We remark that the possibility measure of the *intersection* of two subsets of X is not determined from the possibility measure of these subsets. The only information we obtain from i) and ii) is the following:

$\forall (A, B) \in [0,1]^X \times [0,1]^X\ \Pi(A \cap B) \leq \min (\Pi(A), \Pi(B))$.

Let us remark that two subsets can be individually possible ($\Pi(A) \neq 0$, $\Pi(B) \neq 0$), but jointly impossible ($\Pi(A \cap B) = 0$).

Let us consider the ***example*** of the date of an attack. X = {monday, tuesday…sunday}. We suppose that it is absolutely possible that the attack occurs on monday or tuesday, relatively possible that it occurs on wednesday, impossible that it occurs on thursday.

$\Pi(\{\text{monday, tuesday}\}) = 1$, $\Pi(\{\text{wednesday}\}) = 0.8$, $\Pi(\{\text{thursday}\}) = 0$.

We deduce that it is absolutely possible that the attack occurs on monday, tuesday or wednesday.

$\Pi(\{\text{monday, tuesday, wednesday}\}) = \max(1, 0.8) = 1$.

It is also absolutely possible that the attack occurs on monday, tuesday or thursday.

$\Pi(\{\text{monday, tuesday, thursday}\}) = \max(1, 0) = 1$,

$\Pi(\{\text{wednesday, thursday}\}) = \max(0.8, 0) = 0.8$,

but the intersection {thursday} of these two subsets {monday, tuesday, thursday} and {wednesday, thursday} of X corresponds to a possibility measure equal to 0.

We deduce from conditions i) and ii) that

- $\prod$ is *monotonous* with respect to the inclusion of subsets of X:

if $A \supseteq B$ then $\prod(A) \geq \prod(B)$.

- if we consider any subset A of X and its complement A^c, at least one of them is absolutely possible. This means that either an event or it contrary is absolutely possible:

$$\forall A \in [0,1]^X \quad \max(\prod(A), \prod(A^c)) = 1,$$

$$\prod(A) + \prod(A^c) \geq 1.$$

9. 2. Possibility distribution

A possibility measure is completely defined if we assign a coefficient in [0, 1] to any subset of X. It is easier to define if we restrict ourselves to the singletons of X and we use condition ii) to deduce the other coefficients.

A *possibility distribution* is a mapping $\pi : X \rightarrow [0, 1]$, satisfying the normalization condition:

$$\sup_{x \in X} \pi(x) = 1.$$

Possibility measure and distribution can be associated. From a possibility distribution π, assigning a coefficient to any element of X, we construct a possibility measure assigning a coefficient to any subset of X as follows:

$$\forall A \in [0,1]^X \quad \prod(A) = \sup_{x \in A} \pi(x).$$

Conversely, from any possibility measure $\prod$, we construct a possibility distribution as follows:

$$\forall x \in X \quad \pi(x) = \prod(\{x\}).$$

For ***example***, let us consider the universe X of integers and the possibility that a given number of persons is present in a given room. From the size of the room, we deduce the possibility distribution π given in Figure 9, where $\pi(x)$ represents the degree of possibility that x persons can be in the room. We deduce the possibility measure $\prod$, for instance $\prod([70, 82\ ]) = 1$, $\prod([90, 100]) = 0$ and $\prod([a, b]) = \pi(a)$ for any $0 \leq a \leq b$. Let us take the oppportunity of this example to point out the difference between possibility distribution and probability distribution, which could be only determined by additional information. More particularly, we remark that $\prod([0, 70]) = \prod([71, 100]) = 1$, which means that two complementary events are both absolutely possible.

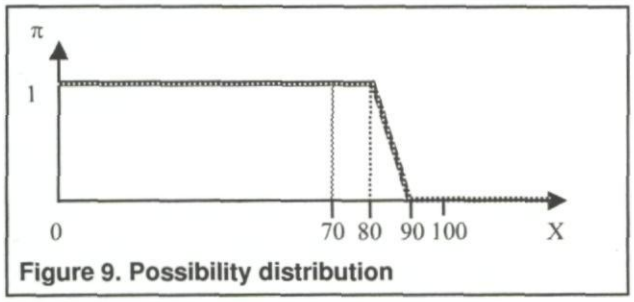


Figure 9. Possibility distribution

In the case of two universes X and Y, we need to define to which extent a pair (x,y) is possible, with $x \in X$ and $y \in Y$.

The *joint possibility distribution* $\pi(x,y)$ on the cartesian product $X \times Y$, is defined for any $x \in X$ and $y \in Y$ and it expresses to which extent x and y can occur simultaneously.

The global knowledge on $X \times Y$ through the joint possibility distribution π provides a marginal information on X and Y by means of the *marginal possibility distribution* on X, for instance:

$$\forall x \in X \ \pi_X(x) = \sup_{y \in Y} \pi(x,y).$$

or on Y:

$$\forall y \in Y \ \pi_Y(y) = \sup_{x \in X} \pi(x,y).$$

These equalities imply the following inequality:

$$\forall x \in X \ \forall y \in Y \ \pi(x,y) \leq \min(\pi_X(x), \pi_Y(y)).$$

We remark that a joint possibility distribution provides uniquely determined marginal distributions, but the converse is false. Determining a joint possibility distribution π on $X \times Y$ from possibility distributions π_X on X and π_Y on Y requires to have information about the relationship between events on X and Y. If we have no information, π cannot be known exactly.

The universes X and Y are *non-interactive* if
$\forall x \in X \ \forall y \in Y \ \pi(x,y) = \min(\pi_X(x), \pi_Y(y))$.

This possibility distribution is the greatest among all those compatible with π_X and π_Y. Two variables respectively defined on these universes are also called *non-interactive.*

The effect of X on Y can also be represented by means of a *conditional possibility distribution* $\pi_{Y/X}$ such that:

$$\forall x \in X \ \forall y \in Y \ \pi(x,y) = \pi_{Y/X}(x,y) * \pi_X(x),$$

for a combination operator *, generally the minimum or the product.

For ***example***, if we consider again the universe X of days of a week and we add the universe Y of hours in a day. π_Y is the possibility degree that somebody is in the room at time y. For instance, $\pi_Y(y) = 0$ for $y \in$]12 p.m., 5 a.m.[, $\pi_Y(y) = 1$ for any other y. For a number x of persons and a time y, we define the possibility degree $\pi(x,y)$ that the pair (x,y) is possible. Y has clearly an influence on X and the universes are interactive. We can have $\pi_X(60) = 1$, $\pi_Y(7) = 1$, but $\pi(60, 7) = 0{,}05$. We can also define the conditional possibility degree $\pi_{Y/X}(x,y)$ that the time is y, given that x persons are in the room. For instance $\pi_{Y/X}(60,18) = 1$, compatible with $\pi(60, 18) = 1$ by $\pi(60, 18) = \min(\pi_{Y/X}(60,18), \pi_X(60))$.

9. 3. Necessity measure

A possibility measure provides an information on the fact that an event A can occur, but it is not sufficient to describe the uncertainy about this event. For instance, if $\prod(A) = 1$, A is absolutely possible, but we can have $\prod(A^c) = 1$, which proves that we have an absolute uncertainty about A. We complete the information on A by means of a measure of necessity on X.

A *necessity measure* is a mapping $\prod : [0,1]^X \rightarrow [0, 1]$, such that:

iii) $N(\emptyset) = 0$, $N(X) = 1$,

iv) $\forall A_1 \in [0,1]^X, A_2 \in [0,1]^X \ldots$

$N(\cap_{i=1,2\ldots} A_i) = \inf_{i=1,2\ldots} N(A_i)$,

Property iv) is reduced to the following in the case of two events:

iv') $\forall (A, B) \in [0,1]^X \times [0,1]^X \ N(A \cap B) = \min(N(A), N(B))$.

Necessity measures are monotonous with regard to set inclusion:

if $A \supseteq B$, then $N(A) \geq N(B)$.

The necessity degree of the union of subsets of X is not known precisely, but we know a lower bound:

$\forall (A, B) \in [0,1]^X \times [0,1]^X \; N(A \cup B) \geq \max (N(A), N(B))$.

We deduce also from iii) and iv) a link between the necessity measure of an event A and its complement A^c:

$\forall A \in [0,1]^X \quad \min (N(A), N(A^c)) = 0$

$N(A) + N(A^c) \leq 1$.

9. 4. Duality between possibility and necessity measures

The properties required from possibility and necessity measures show that they are linked together. They are *dual* in the following sense. For a given universe X and a possibility measure Π on X, the measure defined by:

$$\forall A \in [0,1]^X \quad N(A) = 1 - \Pi (A^c),$$

where A^c is the complement of A in X, is a necessity measure on X.

While $\Pi(A)$ evaluates the possibility of A to occur, N(A) represents the certainty we have in this occurrence. We are certain that A occurs (N(A) = 1) if and only if A^c is impossible ($\Pi(A^c) = 0$) and then $\Pi(A) = 1$.

If Π is defined from a possibility distribution π, we can define its dual necessity measure by:

$$\forall A \in [0,1]^X \quad N(A) = \inf_{x \notin A} (1 - \pi(x)).$$

The duality between Π and N appears also in the following relations, satisfied $\forall A \in [0,1]^X$:

- $\Pi(A) \geq N(A)$
- $\max (\Pi(A), 1 - N(A)) = 1$,
- if $N(A) \neq 0$, then $\Pi(A) = 1$,
- if $\Pi(A) \neq 1$, then $N(A) = 0$.

With the previous ***example***, the certainty on the fact that at most 80 persons are in the room equals $N([0, 80]) = 1 - \Pi([81, 100]) = 1 - \pi(81) = 0.1$.

9. 5. Possibility and necessity measures of fuzzy sets

Possibility and necessity measures have been defined for crisp subsets of X, not for fuzzy sets.

In the case where fuzzy sets are observed, analogous measures are defined to compare an observed fuzzy set F to a reference fuzzy set A of X.

The *possibility* of F *relative to* A is defined as:

$$\Pi(F; A) = \sup_{x \in X} \min (f_F(x), f_A(x)).$$

We remark that $\Pi(F; A) = 0$ indicates that $F \cap A = \emptyset$, and $\Pi(F; A) = 1$ indicates that $F \cap A \neq \emptyset$.

The dual quantity defined by $N(F; A) = 1 - \Pi(F^c; A)$ is the necessity of F with regard to A, defined as:

$$N(F; A) = \inf_{x \in X} \max (f_F(x), 1 - f_A(x)),$$

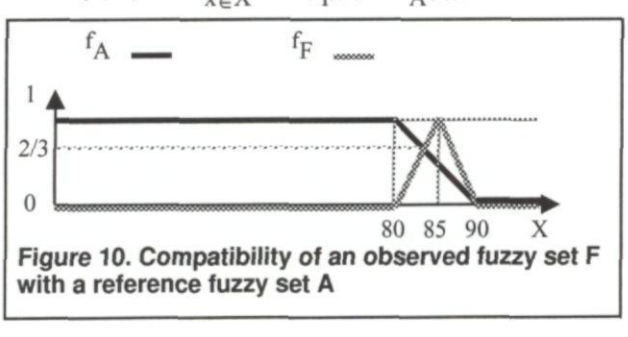


Figure 10. Compatibility of an observed fuzzy set F with a reference fuzzy set A

These coefficients are used, among others, to measure to what extent F is suitable with A.

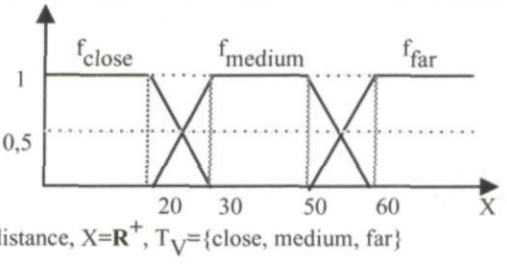
For ***example***, with the universe X of integers, we can evaluate the compatibility of an observed number of persons represented by F = "around 85" with the reference fuzzy set A = "approximately less than 80" (Figure 10). We get $\Pi(F;A) = 2/3$ and $N(F;A)=0$.

10. LINGUISTIC VARIABLES AND FUZZY PROPOSITIONS

Possibility theory, as presented in section 9, is restricted to crisp subsets of a universe. The purpose of its introduction was to evaluate uncertainty due to inaccuracy. We need to establish a link between both approaches.

10. 1. Linguistic variables

A *linguistic variable* is a 3-uple (V, X, T_V), defined from a variable V (distance, size, length…) defined on a universe X. and a set $T_V = \{A_1, A_2 \ldots\}$ of fuzzy characterizations of V. For instance, with V = distance, we can have T_V = {far, medium, close}. We use the same notation for a linguistic characterization and for its representation by a fuzzy set of X (figure 11).



V=distance, X=$\mathbf{R}^+$, T_V={close, medium, far}

Figure 11. Example of linguistic variable (V, X, T_V)

The set T_V corresponds to basic characterizations of V. We need to construct more characterizations of V to enable efficient reasoning from values of V.

A *linguistic modifier* is an operator M yielding a new characterization m(A) from any characterization A of V, in such a way that $f_{m(A)} = t_m(f_A)$, for a mathematical transformation t_m associated with m.

For a set M of modifiers, $M(T_V)$ denotes the set of fuzzy characterizations deduced from T_V.

For ***example***, with M = {rather, very}, we obtain $M(T_V)$ = {very far, very medium, very close, rather far…} from T_V = {far, medium, close}.

Classical linguistic modifiers are defined by (figure 12):

- $f_{m(A)}(x) = f_A(x)^2$ (very) introduced by L.A. Zadeh
- $f_{m(A)}(x) = f_A(x)^{1/2}$ (more or less) introduced by L.A. Zadeh
- $f_{m_1(A)}(x) = \min (1, \lambda f_A(x))$, for $\lambda > 1$ (approximately),
- $f_{m_2(A)}(x) = \max (0, \upsilon\varphi(x) + 1 - \upsilon)$, , for a parameter υ in [1/2, 1] (about).
- $f_{m_3(A)}(x) = \min (1, \max (0, \varphi(x) + \beta))$, with $0 < \beta < 1$ (rather)
- $t_{m_4}(f_A(x)) = f_A(x+\alpha)$, for a real parameter α (really or rather, depending on the sign of α)

where φ is the function identical with f_A on its support and extending it out of the support.

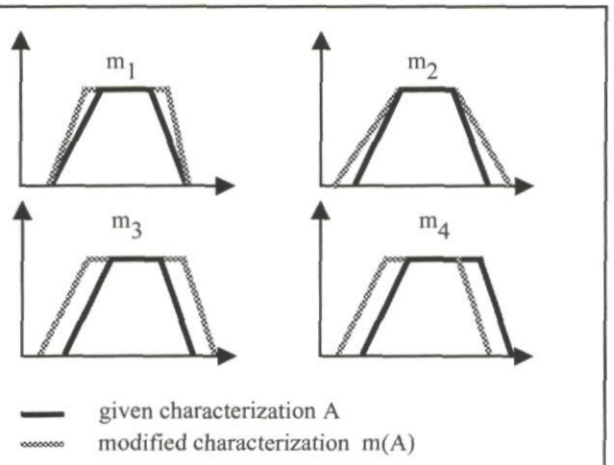


Figure 12. Examples of linguistic modifiers

10. 2. Fuzzy propositions

We consider a set L of linguistic variables and a set M of linguistic modifiers.

For a linguistic variable (V, X, T_V) of L, an *elementary proposition* is defined as "V is A", by means of a normalized fuzzy set A of X in T_V or in $M(T_V)$.

The more suitable the precise value of V with A, the more true the proposition "V is A". The *truth value* of an elementary fuzzy proposition "V is A" is defined by the membership function f_A of A.

A *compound fuzzy proposition* is obtained by combining elementary propositions "V is A", "W is B"... for non-interactive variables V.

The simplest fuzzy proposition is a *conjunction* of elementary fuzzy propositions "V is A and W is B" (for instance "the distance to the town is far and the size of the place is big"), for two variables V and W respectively defined on universes X and Y. It is associated with the cartesian product A × B of fuzzy sets of X and Y, characterizing the pair (V, W) on X × Y. Its truth value is defined by $\min(f_A(x), f_B(y))$ or more generally $T(f_A(x), f_B(y))$ for a t-norm T, in any (x,y) of X × Y. Such a fuzzy proposition is very common in rules of knowledge-based systems and in fuzzy control.

Analogously, we can combine elementary propositions by a *disjunction* of the form "V is A or W is B" (for instance "the distance to the town is far or the size of the place is big"). The truth value of the fuzzy proposition is defined by $\max(f_A(x), f_B(y))$, or more generally $\perp(f_A(x), f_B(y))$ for a t-conorm $\perp$, in any (x,y) of X × Y.

An *implication* between two elementary fuzzy propositions provides a fuzzy proposition of the form "if V is A then W is B" (for instance, "if the distance to the town is far then the position is acceptable") and we will study carefully this form of fuzzy proposition because of its importance in reasoning in a fuzzy framework.

More generally, we can construct fuzzy propositions by conjunction, disjunction or implication on already compound fuzzy propositions.

A fuzzy proposition based on an implication between elementary or compound fuzzy propositions, for instance of the form "if V is A and W is B then U is C" is a *fuzzy rule*. "V is A and W is B" is its *premise* and "U is C" its *conclusion*.

10. 3. Possibility distribution associated with a fuzzy proposition

The concepts of linguistic variable and fuzzy proposition are useful for the management of imprecise knowledge when we associate them with possibility distributions to represent uncertainty.

A fuzzy characterization A such as "far" is given in a prior way and its membership function f_A indicates to what extent each element x of X belongs to A. A fuzzy proposition such as "the distance is far" is a posterior information, given after an observation, which describes to what extent it is possible that the exact distance is any element of X.

An elementary fuzzy proposition induces a *possibility distribution* $\pi_{V, A}$ on X, defined from the membership function of A by:

$$\forall x \in X \quad \pi_{V, A}(x) = f_A(x).$$

From this possibility distribution, we define a possibility and a necessity measure for any crisp subset of X, given the description of V by A:

$$\Pi_{V, A}(D) = \sup_{x \in D} \pi_{V, A}(x),$$
$$N_{V, A}(D) = 1 - \Pi_{V, A}(D^c).$$

Analogously, a compound fuzzy proposition induces a possibility distribution on the cartesian product of the universes. For instance, a fuzzy proposition such as "V is A and W is B", with V and W defined on universes X and Y, induces the following possibility distribution:

$$\forall x \in X, \forall y \in Y \quad \pi_{(V,W), A \times B}(x,y) = \min(f_A(x), f_B(y)).$$

In the case of an uncertain fuzzy proposition such as "V is A, with an uncertainty ε", for $A \in T_V$, no element of the universe X can be rejected and every element x of X has a possibility degree at least equal to ε. Such a fuzzy proposition is associated with a possibility distribution $\pi'(x) = \max(\pi_{V, A}(x), \varepsilon)$ (Figure 13).

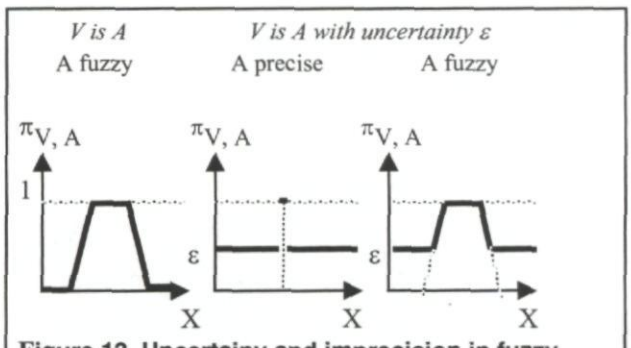


Figure 13. Uncertainy and imprecision in fuzzy propositions

11. FUZZY LOGIC

The use of imprecise and/or uncertain knowledge leads to reasoning in a way close to human reasoning and different from classical logic. More particularly, we need:

- to manipulate truth values intermediate between absolute truth and absolute falsity

- to use soft forms of quantifiers, more gradual than the universal and existential quantifiers $\forall$ and $\exists$

- to use deduction rules when the available information is imperfectly compatible with the premise of the rule.

For these reasons, fuzzy logic has been introduced with the following characteristics:

- propositions are fuzzy propositions contructed from sets L of linguistic variables and M of linguistic modifiers,

- the truth value of a fuzzy proposition belongs to [0, 1] and it is given by the membership function of the fuzzy set used in the proposition

- fuzzy logic can be considered as an extension of classical logic and it is identical with classical logic when the propositions are based on crisp characterizations of the variables.

11. 1. Fuzzy implications

Let us consider a fuzzy rule "if V is A then W is B", based on two linguistic variables (V, X, T_V) and (W, Y, T_W).

A *fuzzy implication* associates to this fuzzy rule the membership function of a fuzzy relation R on $X \times Y$ defined as:

$\forall (x, y) \in X \times Y \quad f_R(x,y) = \Phi(f_A(x), f_B(y))$,

for a function Φ chosen in such a way that, if A and B are singletons, then the fuzzy implication is identical with the classical implication.

There exist many definitions of fuzzy implications. The most commonly used are summarized in Table 2. There can be classified in general classes, according to their behavior. One of the best-known classes is based on a t-norm T as follows:

$f_R(x,y) = T^*(f_A(x), f_B(y))$,

with T^* the quasi-inverse of T, defined as:

$\forall (u,v) \in [0,1]^2 \quad T^*(u,v) = \sup \{w \in [0,1],\ T(u,w) \le v\}$.

Particular cases of this class are Brouwer-Gödel implication, with $T(u,v) = \min(u,v)$, Goguen implication with $T(u,v) = u.v$ and Lukasiewicz implication with $T(u,v) = \max(u+v-1, 0)$.

This means that we can choose the t-norm T to define the *conjuntion* of fuzzy propositions compatible with each fuzzy implication.

Truth value $f_R(x,y)$	*Name of the implication*
$1 - f_A(x) + f_A(x) . f_B(y)$	Reichenbach R^R
$\max(1 - f_A(x), \min(f_A(x), f_B(y))$	Willmott R^W
$\begin{cases} 1 \text{ si } f_A(x) \le f_B(y) \\ 0 \text{ otherwise} \end{cases}$	Rescher-Gaines R^{RG}
$\max(1 - f_A(x),\ f_B(y))$	Kleene-Dienes R^{KD}
$\begin{cases} 1 \text{ si } f_A(x) \le f_B(y) \\ f_B(y) \text{ otherwise} \end{cases}$	Brouwer-Gödel R^{BG}
$\begin{cases} \min(f_B(y)/ f_A(x), 1) \text{ si } f_A(x) \neq 0 \\ 1 \text{ otherwise} \end{cases}$	Goguen R^G
$\min(1 - f_A(x) + f_B(y), 1)$	Lukasiewicz R^L
$\min(f_A(x),\ f_B(y))$	Mamdani R^M *
$f_A(x) . f_B(y)$	Larsen R^P *

* These quantities do not generalize the classical implication, but they are used in fuzzy control to manage fuzzy rules

Table 2. Fuzzy implications

11. 2. Generalized modus ponens

Generalized modus ponens is an extension of the scheme of reasoning called modus ponens in classical logic. For two propositions p and q such that $p \Rightarrow q$, if p is true, we deduce that q is true. In fuzzy logic, we use fuzzy propositions and, if p' is true, with p' approximately identical with p, we want to get a conclusion, even though it is not q itself.

Generalized modus ponens (g.m.p.) is based on the following propositions:

Rule	if V is A then W is B
Observed fact	V is A'
Conclusion	W is B'

The membership function $f_{B'}$ of the conclusion is computed from the available information: f_R to represent the rule, $f_{A'}$ to represent the observed fact, by means of the so-called combination-projection rule:

$\forall y \in Y \quad f_{B'}(y) = \sup_{x \in X} T(f_{A'}(x), f_R(x,y))$,

for a t-norm T called *generalized modus ponens operator.*

The choice of T is determined by the compatibility of the generalized modus ponens with the classical modus ponens: if A = A', then B = B'.

The most usual g.m.p operators suitable with this condition are summarized in Table 3.

g.m.p operators T	*fuzzy implication R*
$T_{Lukasiewicz}(u,v) = \max(u+v-1, 0)$	$R^R, R^W, R^{RG}, R^{KD}, R^{BG}, R^G, R^L, R^M, R^P$
$T_{Zadeh}(u,v)=\min(u,v)$	R^{RG}, R^{BG}, R^M, R^P
$T_{probabilistic}(u,v)=u.v$	$R^{RG}, R^{BG}, R^G, R^M, R^P$
T	$f_R(x,y) = T^*(f_A(x), f_B(y))$

Table 3. Generalized modus ponens operators associated with fuzzy implications

11. 3. Fuzzy inferences

The choice of a fuzzy implication is based on its behavior. An example of comparison of their behaviors is given in Figure 14. We remark that some fuzzy implications entail an uncertainty about the conclusion (Kleene-Dienes), while some others provide imprecise conclusions (Reichenbach, Brouwer-Gödel, Goguen). Some of them entail both types of imperfection (Lukasiewicz).

Let us consider the following ***example***:

Rule: "if the distance to the town is medium then the position is acceptable",

with the universe of distances $X = \mathbb{R}^+$ and the universe of degrees of acceptability Y = [0,1].

Observation: the distance to the town is 27 km.

Conclusion:

- it is relatively certain that the position is acceptable (R^{KD})
- the position is rather acceptable (R^{R}, R^{BG}, R^{G})
- it is relatively certain that the position is rather acceptable (R^{L})

(Figure 14)

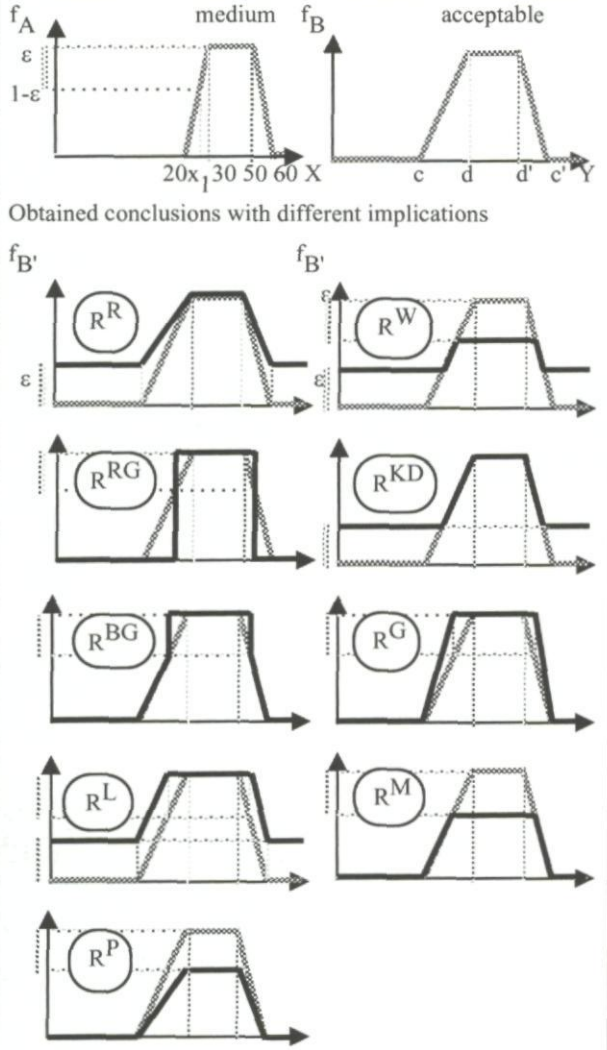


Figure 14 - Conclusion obtained from a precise observation

Fuzzy inferences are used in rule-based systems, when there exist imprecise data, when we need a flexible system, with representation of the linguistic descriptions depending on the environment of the system or its conditions of utilization, when we cope with categories with imprecise boundaries, when there exist subjective variables described by human agents. Expert systems have been realized in medical diagnosis, financial engineering, civil engineeering, military decision-making, for instance.

12. FUZZY CONTROL

Fuzzy control has the same purpose as conventional control: to control a process or, more generally, a system, by acting on some variables, with a given objective.

But fuzzy control has the following particularities:

- The mathematical knowledge of the system is not necessary. The basic knowledge is generally provided by an operator or an expert of the system in a linguistic form. For instance, conventional control of a car needs to know the equations describing the functioning of the engine. Fuzzy control needs to know how a human agent drives the car.

- Some variables describing the system are subjective, for instance linked with human senses (view, touch...), or more complex, such as comfort or beauty.

Fuzzy control is interesting because of the following properties:

- it can be used when the system is complex, ill-known.
- several expert opinions can be easily aggregated
- several objectives can be followed simultaneously
- it is flexible and easily adaptable to new conditions of utilization
- it is able to resist to disturbances or fluctuations of the system parameters.

12. 1. General form of a fuzzy controller

A fuzzy controller can be regarded as a fuzzy rule-based system. It is described in Figure 15. The knowledge base contains a set of fuzzy rules involving the variables describing any state of the system (input variables) in their premises, and the variables used to act on the system (output variables) in their conclusions. At a given time, sensors provide precise values of input variables. The fuzzy interface establishes a link between these crisp values and fuzzy sets used in the knowledge base. The crisp interface transforms the fuzzy conclusions provided by fuzzy inferences in the fuzzy reasoning module into crisp values of output variables to be used to act on the system.

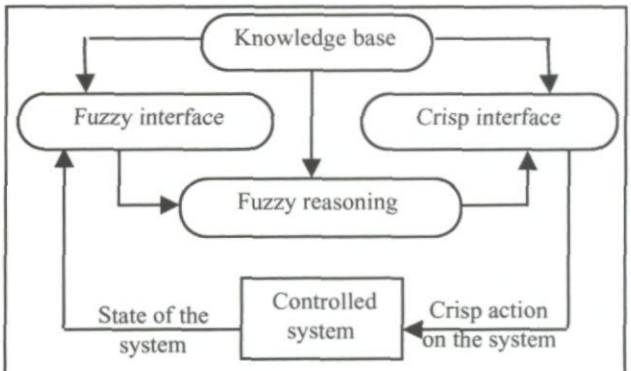


Figure 15. General form of a fuzzy controller

Let us consider linguistic variables (V_1, X_1, T_1), (V_2, X_2, T_2)... associated with the input variables and a linguistic variable (W, Y, T) defined on the ordered universe Y associated with the output variable W. We use fuzzy partitions of X_i, i = 1, 2: each element A_{1i} , A_{2i} of the universe X_i has a non-null membership degree for at least one fuzzy set of T_i. Figure 11 gives an example of such a fuzzy partition, for the variable X_i = distance, with A_{1i} = close, A_{2i} = medium, A_{2i} = far.

The rules have the general following form:

(R1) If V_1 is A_{11} and V_2 is A_{12} then W is B_1

(R2) If V_1 is A_{21} and V_2 is A_{22} then W is B_2,

.....

For ***example***, with a driving system:

(R1) if the distance to previous car is close then the speed reduction is big

(R2) if the distance to previous car is medium and the road is slippery then the speed reduction is big

...

For a given set of input variable values, we obtain a crisp value of the output variable W in 3 steps:

S1) For each rule (Rj), the fuzzy reasoning module, based on fuzzy inferences as indicated in section 11, provides an intermediate characterization B'_j of W.

S2) These intermediate characterizations are aggregated by means of a t-conorm if Mamdani or Larsen implication is chosen, a t-norm for any other fuzzy implication. A fuzzy characterization B' of W is then obtained.

S3) An element y_o of the universe Y is chosen as representative of B'. Several methods are available for this *defuzzification* step. The most classical ones are:

- the center of area: $y_o = [\int_Y f_{B'}(y)\, y dy] / [\int_Y f_{B'}(y)\, dy]$

- the mean of maxima: y_o is the mean of the elements y of Y with the greatest membership degre $f_{B'}(y)$.

12. 2. Mamdani-type fuzzy control

The Mamdani-type fuzzy control is the most classical fuzzy control. It corresponds to a set product method rather than to a reasoning method, but it can be interpreted as a fuzzy inference also. It is based on Mamdani "implication" and the minimum as a g.m. p. operator. It is usually used with the center of area as a defuzzification method.

An ***example*** of its functioning is given in Figure 16.

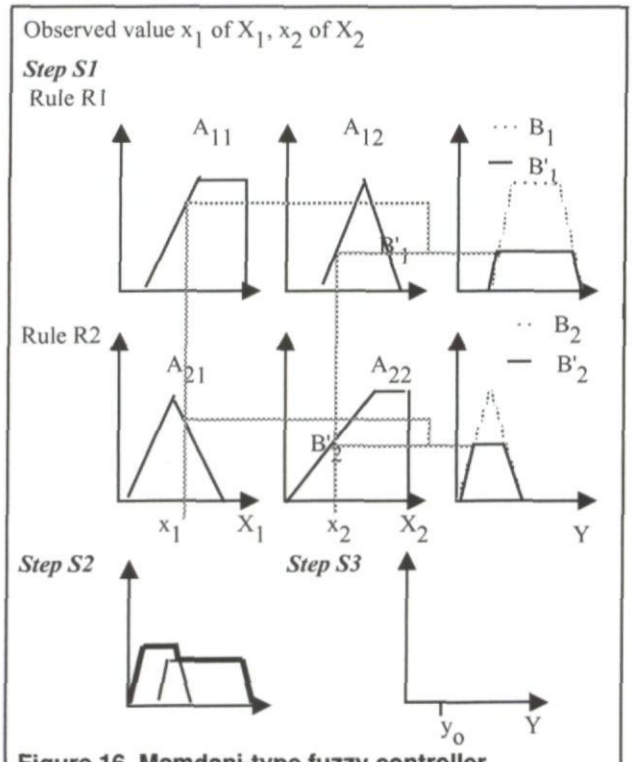


Figure 16. Mamdani-type fuzzy controller

12. 3. Utilizations of control

There exist other forms of fuzzy control, such as the interpolation method, the Sugeno-Takagi method. The developments about theory and methodology are various and provide a serious framework for the control of complex systems.

Applications of fuzzy control have a wide range, from the control of chemical plants to robotics, from autonomous vehicles to home appliance.

13. CONCLUSION

It is interesting to use fuzzy set theory when at least one of the following problems occurs:

- we have to deal with imperfect knowledge,

- a precise modelling of a system is difficult,

- we have to cope with both uncertain and imprecise knowledge,

- we have to manage numerical knowledge (numerical values of variables, "100 meters") and symbolic knowledge (descriptions of variables in natural language, "far") in a common framework,

- human components are involved in the studied system (observers, users of the system, agents) and bring approximative or vague descriptions of variables, subjectivity (degree of risk, aggressiveness of the other participants), qualitative rules ("if the speed is too high, reduce the speed") , gradual knowledge ("the closer, the more dangerous"),

- we have to take into account imprecise classes, ill-defined categories ("vulnerable position"),

- we look for a flexible management of knowledge, adaptable to the environment or to the situation we meet,

- the system is evolutionary, which makes difficult to describe precisely each of its states.

REFERENCES

[1] Bouchon-Meunier B., La logique floue, Que Sais-Je ? n°2702, Presses Universitaires de France, 2nd edition, 1994

[2] Bouchon-Meunier B., La logique floue et ses applications, Addison-Wesley, Paris, 1995

[3] Dubois D., Prade H., Fuzzy sets and systems, theory and applications, Academic Press, 1980.

[4] Dubois D., Prade H., Théorie des possibilités, applications à la représentation des connaissances en informatique, Masson, 2ème édition, 1987.

[5] Dubois D., Prade H., Yager R.R., Readings in Fuzzy Sets for Intelligent Systems, Morgan Kaufmann, San Mateo, CA, 1993.

[6] Klir G., Folger T., Fuzzy sets, uncertainty and information, Prentice Hall, 1988.

[7] Yager R.R., Filev D.P., Essentials of fuzzy modeling and control, John Wiley and Sons, 1994

[8] Yager R.R., Ovchinnikov S., Tong R.M., Nguyen H.T. (eds.), Fuzzy sets and applications, selected papers by L.A. Zadeh, John Wiley & Sons, 1987.

[9] Zimmermann H.J., Fuzzy set theory and its applications, Kluwer, Dordecht, 1985.

Hybrid Architectures for Intelligent Systems, Learning Inference Systems

B. Bouchon-Meunier
LIP6, CNRS-UPMC
Case 169, 4 place Jussieu
75252 Paris Cédex 05
France

1. INTRODUCTION

Fuzzy systems are very efficient since they are generally based on expert knowledge, they are very robust and flexible and they manage interpretable knowledge. The main problem in their construction is the tuning of all the parameters involved in their construction : shapes and parameters of membership functions, number of classes in fuzzy partitions of the universes of definition of variables (number of available characterizations for each variable), number of rules, values of the rule conclusions when they are crisp. While it is interesting to allow the designer of the fuzzy system to make some of the choices, the automatic tuning of the other parameters is a valuable help. In the case where training sets of data are available, it is possible to use automatic methods for the learning part of a fuzzy system.

Neural networks are powerful for learning and it seems natural to consider them as a possible tool for such automatic methods. It is nevertheless interesting to mention that the association of fuzzy and neural approaches is not restricted to such methods and there exist various types of hybrid systems based on their combination. We present the main complementary utilizations of these techniques in section 2.

Many other solutions have been pointed out for the learning part of the construction of fuzzy systems. The most important directions are presented in section 3.

Among these methods, we have treated separately genetic algorithms, which provide many kinds of combinations with fuzzy logic-based systems, similarly to neural networks. Section 4 deals with these approaches.

Regarding the definition of membership functions, which takes a part in the determination of fuzzy systems, direct methods have been proposed, avoiding the need of a training set necessary for automatic learning methods. They are summarized in section 5.

Finally, we show in section 6 that special types of fuzzy systems themselves can be considered as learning systems, based on induction from examples, providing a means of generalizing knowledge regarding the assignement of a class or a decision to a situation, on the basis of previously studied situations.

2. HYBRID SYSTEMS: FUZZY LOGIC AND NEURAL NETWORKS

Hybrid systems are useful to take advantages and avoid disavantages of several methodologies by a complementary utilization. It is the case, for instance with fuzzy logic and neural networks and we give the main joint utilizations of these two approaches. In most of the cases, they are constructed to solve learning problems.

Neural networks, as well as fuzzy systems, are universal approximators of continuous functions and this property is a reason to use them to learn a structure from a training set, since it enables the user to interpolate between the values of a discrete set to determine a continuous function, for instance a transfer function from an input space into an output space [22] [36].

Fuzzy systems are very efficient since they are generally based on expert knowledge, they are very robust and flexible and they manage interpretable knowledge. The main problem in their construction is the tuning of all the parameters involved in their construction : shape and parameters of membership functions, number of classes in fuzzy partitions of the universes of definition of variables (number of available characterizations for each variable), number of rules, values of the conclusions when they are crisp. While it is interesting to allow the designer of the fuzzy system to make some of the choices, the automatic tuning of the other parameters is a valuable help. In the case where training sets of data are available, it is possible to use neural networks for the learning part of a fuzzy system.

Several methods of supervised learning exist, with various degrees of fusion between the two methodologies [35]. They give rise to very different architectures. We present three of them from the strongest to the weakest degree of fusion.

In methods based on a strong fusion, a neuronal representation of fuzzy rules is used (sections 2. 1, 2. 2, 2. 3). The neurons are introduced at a low level of calculus in fuzzy rules. The neurons of the various layers do not perform the same operations, which can be different from those of classical neurons.

In methods based on a weak fusion (section 2. 4), membership degrees are computed by neural networks. The expressibility of fuzzy rules is generally lost. Classical neurons are used.

The last methods (section 2. 5) are based on a joint utilization of two methodologies without any fusion.

Hybrid systems are also used for non-supervised learning (section 2. 6)

2. 1. Neuronal representation of fuzzy rules for control

Let us consider a fuzzy control system, based on general fuzzy rules such as

(R1) If V_1 is A_{11} and V_2 is A_{12} then W is b_1

(R2) If V_1 is A_{21} and V_2 is A_{22} then W is b_2...

with A_{1j}, A_{2j}... fuzzy characterizations of V_j, given in a fuzzy partition of its universe of definition, j = 1, 2, and b_1, b_2... precise values of W.

The *fuzzy inference process* can be decomposed into 4 elementary operations:

- For a given rule Ri:
  - Computation of the truth value of "V_1 is A_{i1}" and of "V_2 is A_{i2}" corresponding to the observed values

Paper presented at the AGARD MSP Lecture Series on "Advances in Soft-Computing Technologies and Applications in Mission Systems", held in North York, Canada, 17-18 September 1997; Amsterdam, The Netherlands, 22-23 September 1997; Madrid, Spain, 25-26 September 1997; Ankara, Turkey, 6-7 October 1997, and published in LS-210.

of V_1 and V_2.

- Computation of the truth value of the whole premise of the rule "V_1 is A_{i1} and V_2 is A_{i2}"
- Computation of the conclusion regarding W.

- Aggregation of the conclusions obtained from all the available rules with the same variable W in their conclusions.

The general idea of an hybrid system is to construct an architecture achieving the operations. Every elementary operation is computed by a neuron. The architecture is analogous to a multilayer perceptron. Each layer corresponds to one of the four steps we mentioned and a special type of neurons is associated with this layer, in order to perform the operation required for the given step : computation of a membership degree, computation of a logical operation... Some of the weights are given, others are adjustable (Figure 1).

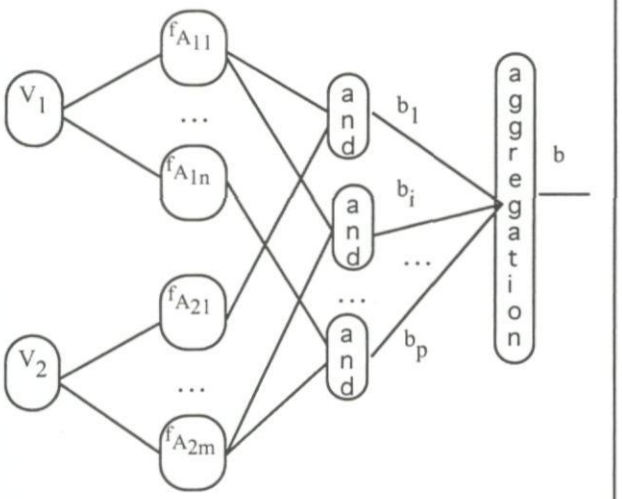


Figure 1. Example of neural network for fuzzy control

• The input layer transmits the observed values of V_1 and V_2 to neurons containing the definitions of the fuzzy characterizations $A_{11}, A_{21} \ldots, A_{12}, A_{22} \ldots$

• The first hidden layer computes the truth value of each elementary fuzzy proposition "V_1 is A_{i1}", "V_2 is A_{i2}"...

• The second hidden layer aggregates these truth values for each rule, providing the truth value of the conjunction of elementary fuzzy propositions "V_1 is A_{i1} and V_2 is A_{i2}", which gives the firing degree a_i of each rule Ri.

• The synaptic value b_i represents the value of W in rule Ri, i = 1, 2... The value b of the output variable W is the mean value of these synaptic values, weighted by the firing degrees (aggregation step):

$$b = [\Sigma_i a_i b_i] / [\Sigma_i a_i]$$

This network provides the automatic adjustment of the synaptic values b_i and of parameters of the membership functions $f_{A_{ij}}$.

This architecture is associated with a *supervised learning algorithm*. The parameters of the network are modified to reduce the error, i.e. the difference between the desired output and the network output, for a given set of examples.

Such a learning algorithm is for instance proposed in [21] with the following choices:

- gaussian membership functions f_A, with parameters α and β such that:

$$f_A(x) = \exp(-(\alpha x+\beta)^2)$$

- conjunction operator:

$$\text{and}(x,y)=[1 + \exp(-(x+y-1.5)/\gamma)]^{-1}$$

which is an approximation of Lukasiewicz t-norm by a continuous function.

Semi-supervised learning is also possible by completing the previous architecture [3]. The available information is then a binary signal indicating that the system works or fails. There is no possible measurement of the difference between expected and obtained value. A reinforcement learning technique is used to tune the fuzzy controller, which learns to propose actions leading to a good functioning of the system and also to evaluate the quality of its past actions. The general scheme of the method is given in Figure 2.

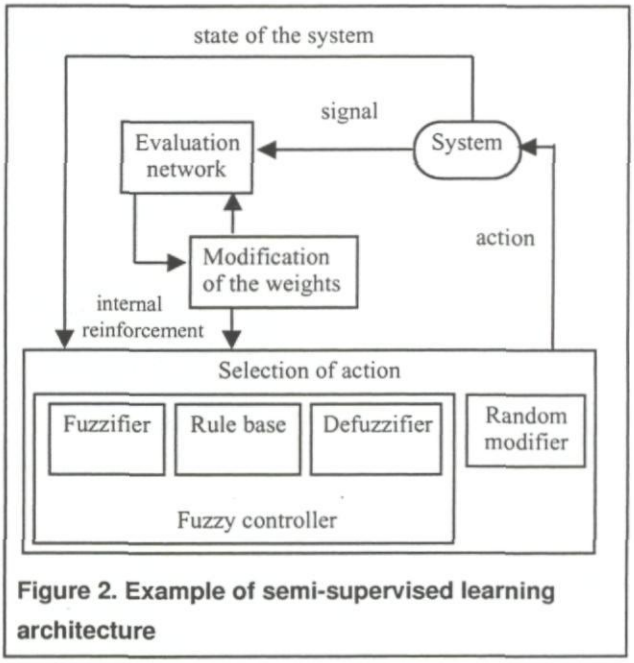


Figure 2. Example of semi-supervised learning architecture

The structure is based on two modules:

- a fuzzy controller, which determines the action necessary for a given state of the system. This controller is associated with a neural network to evaluate a confidence degree associated with this action.

- a neural network evaluating the effect of the last action on the system.

For a given state of the system, the fuzzy controller determines an action from fuzzy rules and the neural network computes the confidence degree of this action.

This action is randomly modified. The smallest the confidence degree, the more important the modification.

The neural network provides the new internal reinforcement measure.

The weights of the neural networks are then modified, depending on the reinforcement: if the reinforcement is positive, the weights are reinforced, if the reinforcement is negative, the weights are weakened.

2. 2. Neuronal representation of fuzzy rules for reasoning

In the case of reasoning, the data available to fire the rules are not precise values of the variables, but fuzzy sets of their universes of definition [55], [26], [27], [28].

The rules have the same form as rules in section 2. 1, with fuzzy propositions in the conclusions:

(Ri) If V_1 is A_{i1} and V_2 is A_{i2} then W is B_i...

The observations yield fuzzy values "V_1 is A'_1", "V_2 is A'_2", where A'_1 and A'_2 are respective fuzzy sets of the universes of V_1 and V_2, supposed to be discrete. These fuzzy sets are generally represented by the list of membership degrees.

• Each neuron of the input layer of the neural network receives one of these fuzzy sets (Figure 3). If $\{x_1,...x_n\}$ is the considered universe, the weights $f_{A_{ij}}(x_k)$, $1 \le k \le n$, represent one of the given fuzzy sets, determined during the learning phase. An input value A'_j is a fuzzy set defined by membership degrees $f_{A'_j}(x_k)$, $1 \le k \le n$. The neuron compares A'_j to A_{ij} and gives a real value α_j. The operations realized in the neuron are different from ordinary neuron operations, in order to be compatible with fuzzy set operations. For instance, the comparison degree can be the following:

$\alpha_j = \sup_{1 \le k \le n} \min(f_{A'_j}(x_k), f_{A_{ij}}(x_k))$.

The number of input neurons is equal to the number of fuzzy sets in the rules.

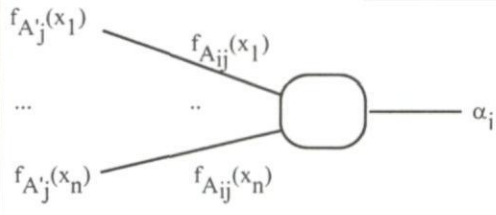


Figure 3. Representation of a fuzzy set by synaptic values

• A first hidden layer contains the fuzzy inference rules, analogously to the neural network presented in section 2. 1.

• The output of the network is a fuzzy set B of the universe of definition of variable W. This universe is supposed to be discrete and denoted by $\{y_1,...y_m\}$. B is defined by its membership degrees $f_B(y)$. The number of neurons in the output layer is m (Figure 4).

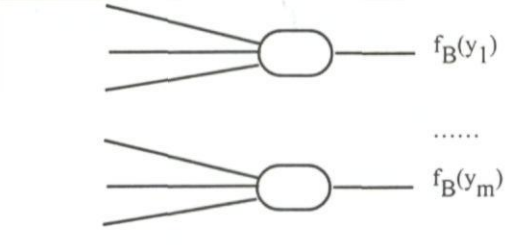


Figure 4. Output layer for the calculus of a fuzzy set

The number of rules can be automatically determined, for instance by starting from all possible rules obtained from all possible characterizations of variables and, then, eliminating the useless rules.

2. 3. Advantages and disadvantages of the association between fuzzy logic and neural networks for rule-based systems

In the methods presented in sections 2. 1 and 2. 2, respective properties of fuzzy logic and neural networks are preserved, lost or improved because of their association. Globally, we can conclude that the association is beneficial. More precisely, we give the list of interesting properties we can take into acount.

Properties of fuzzy logic :

- linguistic rules and structured knowledge are preserved in the association.

- the shape of membership functions, the number of fuzzy descriptions (classes of a fuzzy partition of the universe) available for each variable, the fuzzy implication and the operators used for the conjunction are still freely chosen.

- learning of parameters and automatic optimization of

  • membership functions in premises of rules

  • conclusions

  • number of rules

is now possible.

Properties of neural networks

- learning capacities are maintened

- problems regarding slow convergence and local minima are still present

- synaptic values and neurons are interpretable

- the number of neurons in each layer is automatic

- we loose a part of robustness of the neural network

Interpolation properties of both fuzzy and neural systems are preserved.

2. 4. Partial utilizations of neural networks in a fuzzy rule-based system

Another solution to associate fuzzy logic and neural networks consists in the construction of neural networks corresponding to the following modules of a fuzzy rule-based system :

- Fuzzification and calculus of elementary propositions

- Fuzzy inference [39]

- Aggregation of conclusions and defuzzification

In some cases, it is possible to compute *grades of membership* for some elements of the universe, from a training set. Then, a neural network is used to interpolate the membership function on the whole universe. The number of neural networks is the number of fuzzy sets used in the fuzzy rules. The output of each network is further used by the fuzzy system.

The advantage of this technique is to avoid the choice of a shape for the membership functions, and to provide fuzzy sets well suited with the data. The number of fuzzy sets in each fuzzy partition is still to be determined.

Another solution is to use a neural network to *replace conjunction and generalized modus ponens*. It is necessary

to have a training set consisting of triples ($f_{A_{i1}}(x_1)$, $f_{A_{i2}}(x_2)$, $f_{B_i}(y)$), for input values x_1 and x_2 of V_1 and V_2, and output value y.

The advantage of such a method is to avoid the prior determination of the rules. The disadvantage is the lack of interpretability of the results in terms of rules.

The two previous techniques have been used by [51] in the construction of an expert system, with again rules of the form:

(Ri) If V_1 is A_{i1} and V_2 is A_{i2} then W is B_i...

Each pair (A_{i1}, A_{i2}) can be regarded as an area of the cartesian product $X \times Y$ of the universes V_1 and V_2 associated with rule Ri. The limits of the area are linear, corresponding to bounds of the kernel and the support of the fuzzy sets (Figure 5).

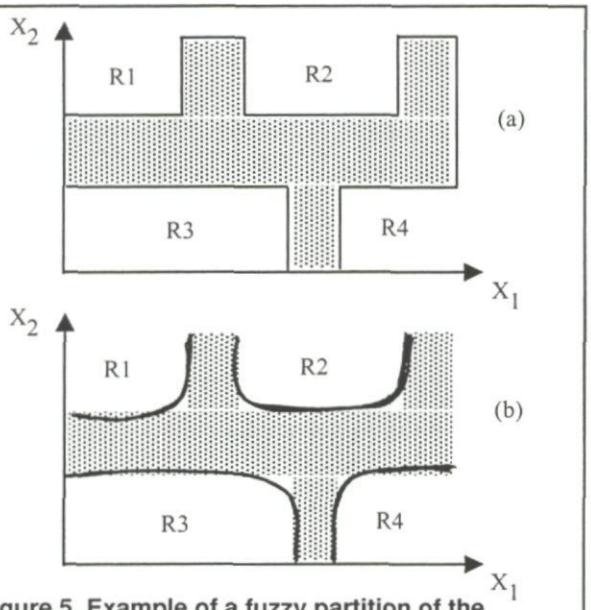


Figure 5. Example of a fuzzy partition of the cartesian product of universes obtained by fuzzy rules (a) and by neural network (b).

In many cases, it is meaningless to split the universes this way. A neural network is then used to split the universe in a less rigid way.

To achieve this soft partitioning, the training set is clustered into disjoint classes, corresponding to the functioning of the system and a corresponding control value. We construct a neural network with input neurons receiving input values of the system, and each output network is associated with a class. The output values of neurons are in [0, 1]. The network gives a 1 output when the input data correspond to the associated class. The ouput value can be in]0, 1[, and it is interpreted as a degree of membership w_k to a class. If several neurons give a non-null value, the input is in the fuzzy boundaries of several classes.

The control value y_k is also learnt by a neural network NN_k with one output, constructed for each class C_k.

The control value B for a given input is computed as the mean of the values computed by all the networks, weighted by the membership grades (Figure 6):

$b=[\Sigma_k w_k y_k] / [\Sigma_k w_k]$

The advantages and disadvantages of such an association between neural and fuzzy techniques are the following.

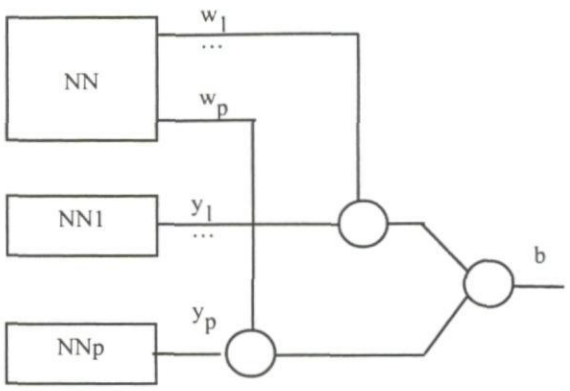


Figure 6. Architecture for the learning of membership degrees and weights of rules

Properties of fuzzy logic:

- the learning and automatic optimization of membership functions is now realized

- the operators of implication and conjunction by the designer of the system are still chosen by the designer of the system

- the knowledge is not structured any more and there are no interpretable rules

Properties of neural networks:

- robustness and learning capacities are preserved

- slow convergence, local minima, choice of the number of neurons in the hidden layers are still problems

Interpolation properties of both fuzzy and neural systems are preserved.

2. 5. Joint utilization of neural networks and fuzzy logic

Some systems use both techniques without any fusion between them. Each technique is used for a phase of the system, according to its capabilities. Fuzzy and neural modules can then be used sequentially. In this kind of independant utilization of both techniques, each of them preserves its qualities and its defaults.

The first scheme contains a neural network followed by a fuzzy system.

For instance, a neural network can be used for clustering or image processing and a fuzzy system for a decision-making phase afterwards.

Another kind of system takes advantage of the efficiency of a neural network for the preparation of data or the detection of patterns in an image. Then a fuzzy system is able to interprete and use these patterns, on the basis of expert knowledge.

A neural network can also be used to recognize patterns in a noisy framework and work as an elementary classifier, and a fuzzy system is then used to refine the clustering.

In the case of a fuzzy control system, the input variables can be determined by means of a neural network when they cannot be measured direcly. An example of such a system is given in Figure 7.

Another organization of both techniques consists in a

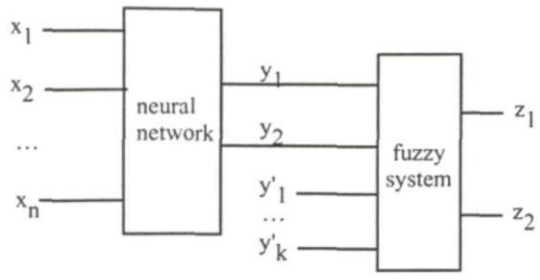


Figure 7. Example of a series of neural network and fuzzy system

preliminary utilization of fuzzy system, followed by a neural network.

For instance, a fuzzy system is used to transform the available data in the training set before a neural network is used for clustering. The learning capacities of the neural network are then improved.

In the case of fuzzy control, a neural network can be used after the fuzzy system, in order to tune the obtained output values, according to further available knowledge. The inputs of the neural network are the outputs of the fuzzy system and additional variables, its outputs are errors on the outputs of the fuzzy system (Figure 8).

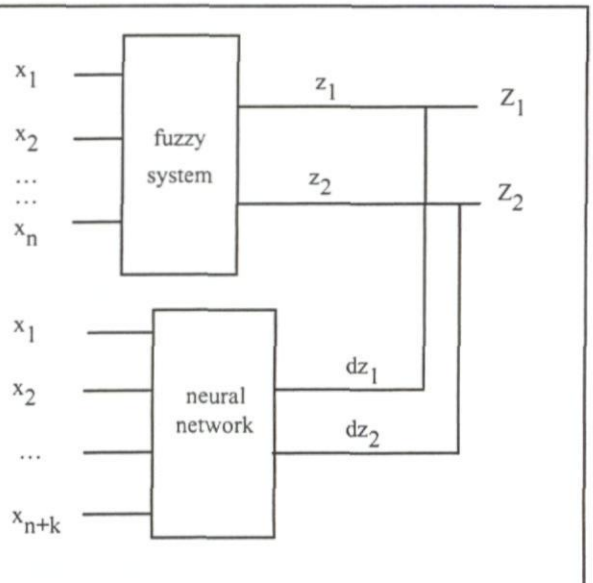


Figure 8. Example of a series of fuzzy system and neural network

2. 6. Non-supervised learning

In the case of non-supervised learning, a competitive algorithm is used; for each example proposed to the network, only one neuron is modified. Such algorithms can be used for fuzzy clustering, in data analysis or pattern recognition [4], [30].

3. LEARNING OF A FUZZY CONTROL SYSTEM STRUCTURE

Learning of fuzzy system structures can be achieved by several approaches. We focuse on control systems.

3.1. Self learning

Self learning has been used in fuzzy control, to learn control rules and membership functions of adaptive controllers.

The first controller which learns himself how to tune its functioning was proposed by Procyk et Mamdani [43]. They called it Self-Organizing Process Controller (S.O.C.) (Figure 9). It observes its environment and it uses the results of its actions to improve them, by means of metarules. A set of prior rules is first given to the system, and it is tuned through error control. Learning is achieved on-line, the rules

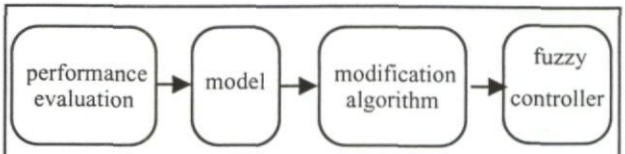


Figure 9. General architecture of the Self Organizing Process Controller

are tuned during the functioning and they can follow the modifications of the process to control.

It is also possible to tune only the membership functions used in the rules.

Learning can also be achieved off-line. The controller is used when the tuning has provided an acceptable error [48].

Another technique consists in the on-line adaptation of a controller which was well tuned for a given environment and which is used in diffferent conditions because of perturbations or parameter variations of the process [47].

3.2. Utilization of a fuzzy model of the process

In the case of fuzzy control, it is possible to use a model of the process to construct the control system [49]. Prior descriptions of characteristics of the process are constructed by means of fuzzy rules describing its behavior.

Then, it is for instance necessary to construct control rules from the model, by compensating an unsatisfying functioning of the system [51].

The form of the rules is generally the following [53] :

(Ri) If V_1 is A_{i1} and V_2 and A_{i2} ... and V_m is A_{im}
then $W_i = p_{i0} + p_{i1}V_1 + p_{i2}V_2 + \ldots + p_{im}V_m$,

for $1 \leq i \leq n$.

The first parameter to be determined is the number n of rules. Then, we must determine fuzzy partitions of the input universes [50], membership functions of the fuzzy values A_{ij} of input variables and coefficients p_{ij} defining the

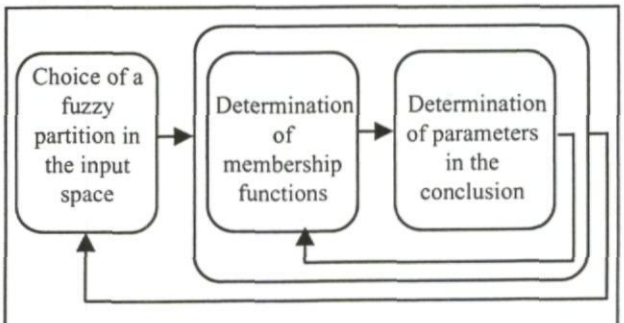


Figure 10. Identification of fuzzy rules

conclusion parts of the rules. Those elements are clearly not independent.

The identification of the number of rules is realized empirically, on the basis of expert knowledge or by a heuristic optimization.

The identification of membership functions is generally realized empirically or by means of a non linear programming method.

The identification of the coefficients is obtained by error minimization, corresponding to the smallest difference between output of the model and desired output (figure 10).

3. 3. Fuzzy relational equations

Fuzzy rule-based systems can be considered from a relational point of view [40] [57]. Let us consider the simplified model of n rules:

if V is A_i then W is B_i, i =1, ... n

and an observation : V is A

With a Mamdani approach, we get a conclusion: W is B, defined as:

$f_B(y) = \sup_x \min(f_A(x), R(x,y))$,

with $R(x,y) = \sup_{1 \le i \le n} \min(f_{A_i}(x), f_{B_i}(y))$.

This can be rewritten as the composition of fuzzy relations

$B = A \circ R$.

We now suppose that we are given a training set of data (A^k, B^k), representing pairs of input values A^k of V and output values B^k of W.

The problem is then to identify the fuzzy relation R optimal under a criterion of closeness, with respect to the training set.

If the rules are well chosen, A^k and B^k must satisfy the above-mentioned equation for any k :

$B^k = A^k \circ R^k$

and the identification of the model is then equivalent to the identification of the relations R^k solutions of these k equations. The theory of fuzzy relational equations provides the following solutions:

$R^k = A^k \alpha B^k$,

where α is an operator defined as :

$u \alpha v = 1$ if $u \le v$, $u \alpha v = v$ if $u > v$.

It can be proved that the intersection of the obtained fuzzy relations is a solution of the identification problem and we can choose:

$R = \min_k (A^k \alpha B^k)$.

The main problem with this method lies in the assumption that the input-output pairs fit well the general relational equation. In fact, the assumption does not always correspond to the reality, because of noisy data. Several solutions have been suggested [41], for instance regarding the identification problem as an optimization problem.

3. 4. Clustering methods

Clustering is a means of determining either the membership functions of a fuzzy system [29], or directly the rules of the system.

In the first case, the membership grades are defined through the suitability degree of a given element with the class to which it has been assigned. A membership grade can also be determined from the distance between a given element and the prototype of its class. The prototype has a membership grade equal to 1 in this case.

3. 5. Optimization

Optimization techniques can help tuning fuzzy rules. Parameters are determined to provide the minimum output error, computed from a training set of input and output values obtained experimentally.

For instance, let us consider the output variable y, I the vector of input variables and A the vector of parameters : parameters of membership functions (bounds of the core and the support in the case of trapezoidal functions, mean and standard deviation in the case of gaussian functions) and also the weights of rules if necessary.

The relationship between input and output can be written $y = f(I, A)$ and optimal values of parameters are those minimizing the difference $\sum_k [y_k - f(I_k, A)]$, with k the index of the experiment, I_k the vector of input values for this experiment, y_k the obtained output value. We can use classical optimization methods, such as the least mean squared method or the gradient descent method. More original methods can also be used, for instance a rapid calculus of partial derivatives [31].

4 GENETIC ALGORITHMS

Among optimization methods, genetic algorithms and evolution strategies are particularly useful to determine the parameters of fuzzy systems, either to determine the best membership functions of fuzzy sets or, more generally, to learn fuzzy rules [22].

We are given a population of vectors, called chromosomes, corresponding to a solution of the problem. For instance, in the case where we need to optimize memberhip functions, a chromosome represents the parameters of all the membership functions, corresponding to their shapes and positions. In the case where we need to learn fuzzy rules, all the parameters are in the chromosomes, including the values of conclusion part [2]. Variable length genotypes can be used [18].

Each chromosome corresponds to a coding of the parameters allowing to define the functions or the rules. This initial population is generally ramdom. Then, random transformations are defined, such as mutation and recombination. These transformations make the population of vectors evolve towards increasingly better regions of the search space. The successive use of the transformations promotes or discriminates some chromosomes, by means of a measure of fitness, allowing to determine which structures will be used to form new ones at the next step. Mutation provides new kinds of chromosomes, selection eliminates the less interesting chromosomes, the predominance of some characteristics with regard to others is favored by means of successive recombinations.

The result of all the transformations is the best possible parameter vector for the fuzzy system.

Similarly to the association of neural networks and fuzzy logic, there exist various types of combination of genetic algorithms and fuzzy logic-based systems. The main directions can be classified as follows [17]:

- fuzzy genetic algorithms: fuzzy logic techniques are used to improve genetic algorithms.

- fuzzy clustering: genetic algorithms are used to optimize fuzzy clustering.

- fuzzy optimization: fuzzy genetic algorithms are used in optimization problems.

- fuzzy neural networks: genetic algorithms are used for the optimization of fuzzy neural networks or genetic algorithms and neural networks are used together to determine the structure of fuzzy systems.

- fuzzy classification: fuzzy classifiers are generated or improved by means of genetic algorithms

- learning: genetic algorithms are used for concept learning, for the learning of fuzzy rules or more generally for the learning of fuzzy logic-based systems.

- fuzzy methods: genetic algorithm are used to help using fuzzy techniques, such as fuzzy relational equations or fuzzy regression analysis.

The domains of applications of the joint use of genetic algorithms and fuzzy logic are various. Among the most intensively studied are the following:

- fuzzy logic control, the more active domain in this matter

- fuzzy expert systems

- fuzzy information processing and database querying

- fuzzy decision making

- fuzzy pattern recognition

- fuzzy image processing.

5. DIRECT LEARNING OF FUZZY SETS

In many cases, the parameters of a fuzzy system are defined heuristically. It is not an easy task to determine membership functions and there exist several direct approaches which can help determining their parameters.

The shape of membership functions represents the first choice to make. This choice is very often made in a prior way. It has been proved empirically that the shape has not an important influence on the results of fuzzy if-then rule-based systems in the case of expert systems. Comparisons of results are sometimes done between two optimized systems with different shapes of membership functions, after the other parameters have been determined.

We present the main techniques available to determine the parameters of membership functions, generally for a given shape. Let us suppose that we look for the membership function f_A of a fuzzy set A of a given universe X.

5. 1. Statistical and probabilistic methods

The simplest methods to determine membership functions are based on statistics.

An elementary principle to obtain the degree of membership of an element x of X to A consists in the *"yes-no" method* [5] [23] [19] [20] [58]. Each member of a given population is asked to answer a binary question. For instance, if X is a set of distances, and A a fuzzy set associated with the linguistic label "long", a question such as "Is it true that a distance of 1.2 km is long ?" provides the membership degree of the value 1.2 km in A, defined as the proportion of answers "yes" obtained in the given population.

In another method [54], the members of a population are asked to *estimate the fuzzy set*: any member p gives a crisp subset A(p) of X which represents the linguistic label A. The membership grade of any element x of X in the fuzzy set associated with A is the frequency of the occurrence of x in the subsets A(p), for any p in the population.

A third method [14] is based on *normalized histograms*. The elements representative of a linguistic label A are gathered in a class. An histogram of the frequencies fr(x) of elements x in this class is realized and normalized as follows:

$$\forall x \in X \quad f_A(x) = fr(x) \; / \max_{x \in X} fr(x)$$

Then, this function is smoothed, by a gaussian filtering for instance, and gives the membership degree $f_A(x)$ of x in the fuzzy set representing A.

We must be careful with these methods which could suggest a statistical view of fuzzy sets. It is clear that $f_A(x)$ has nothing to do with the probability of x, it is only related with the probability that somebody in a given population thinks that x is a representative of the class A. These methods cannot be used for any membership function.

A more sophisticated method [23] consists in a *function with threshold*. The first step consists again in "yes-no" questions asked to an expert, such as "Does the element x belong to the class A ?" for a given crisp class of X. Different elements of X are proposed to the expert, until we get a thresholded function, for instance the function indicated in figure 11. We think that the expert may have given a wrong number for the threshold, but the more we move off this value, the smallest the probability of error. The error probability is represented by a function E, which is generally chosen gaussian. Finally, the membership function is obtained by means of the convolution of the function with threshold and E.

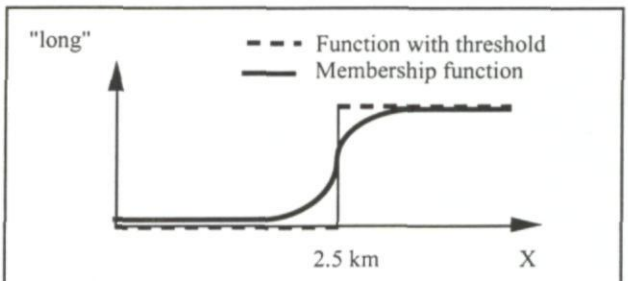


Figure 11. Function with threshold and membership function

It is also possible to derive a membership function from a *probability density*. The probability p(x) that any element x of X is characterized by the linguistic label A is supposed to be known [16]. The membership function f_A of the fuzzy set representing A is proportional, with a multiplicative coefficient λ to be determined, to the probability density. A threshold is chosen, from which the membership function equals 1, when the probability density is high enough.

$f_A(x) = \lambda \, p(x)$ if $\lambda \, p(x) < 1$,

$f_A(x) = 1$ if $\lambda \, p(x) \geq 1$.

The coefficient λ is chosen in such a way that :

$$\int_{\lambda \, p(x) < 1} p^2(h) \, dh + \int_{\lambda \, p(x) \geq 1} p(h) \, dh - c = 0$$

for a level c of confidence in the fact that the elements x of X belong to A, with $c \in [0, 1]$.

5. 2. Psychometrical methods

In these methods, an expert is asked to provide a numerical representation of a linguistic label.

The simplest method determines the *support and core* of the membership function. The expert is asked to determine the elements of X which certainly belong to A (the core ker(A) of the fuzzy set representing A) and the elements of X which do not certainly belong to A (the complement of the support supp(A) of the fuzzy set in X). The expert gives four parameters (p_1, p_2, p_3, p_4) and a membership function is constructed with value 1 on $[p_2, p_3]$ and with a null value out of $[p_1, p_4]$. Its shape is chosen arbitrarily (linear, with parts of parabola...) on the subset supp(A)-ker(A), non decreasing on $[p_1, p_2]$, non increasing on $[p_3, p_4]$ (figure 12).

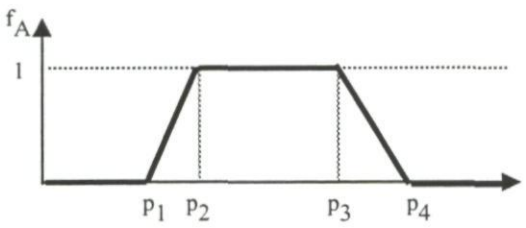


Figure 12. Support and core determination of a membership function

A *structural quantization* is also possible to determine a membership function [58]. A partition of X in three subsets X_o, ker(A), X_f is constructed by experts, with:

$X_o = X - \text{supp}(A)$, $X_f = \text{supp}(A) - \text{ker}(A)$.

We define :

$\forall x \in X_f \quad f_A(x) = \text{Card}(\{y \in X_f / x >_A y\}) / \text{Card}(X_f)$

where $x >_A y$ means that "x belongs to A" is more true than "y belongs to A".

This method is limited to finite universes X.

A similar method [38] is based on *two weak orders* $\geq_A$ on the elements of X and $\geq'_A$ on the pairs of elements of X.

$x_1 \geq_A x_2 \Leftrightarrow f_A(x_1) \geq f_A(x_2)$,

where $x_1 \geq_A x_2$ means that «x_1 is at least as much A as x_2 is A» or «"x_1 is A" is at least as true as "x_2 is A"».

$|x_2\ x_1| \geq'_A |x_4\ x_3| \Leftrightarrow f_A(x_2) - f_A(x_1) \geq f_A(x_4) - f_A(x_3)$,

where $|x_2\ x_1| \geq'_A |x_4\ x_3|$ means that «"x_2 is more A than x_1" is at least as true as "x_4 is more A than x_3"».

Two bounds x_m and x_M are considered, such that:

$\forall x \in X\ x \geq_A x_m$, and x_m is certaintly not in A

$\forall x \in X\ x_M \geq_A x$, and x_M is certaintly in A.

x_m and x_M are chosen arbitrarily among all the elements x satisfying these definitions. x_m and x_M belong respectively to X - supp(A) and ker(A):

$f_A(x_m) = 0$ and $f_A(x_M) = 1$.

To obtain the membership function, an axis is presented to the expert, with an explanation about the fact that x_m is an element which does surely not belong to A and x_M is an element which surely belong to A, and the segment between these two points on the axis works as a truth scale for the proposition « x is A ». The same question is asked several times to the same expert, rather than asked to several experts, in order to obtain a possible interpretation of the answers of the expert (Figure 13 a). Three methods are available.

• direct evaluation: the elements of X are presented to the expert one after the other. He puts a mark on the segment $[x_m, x_M]$ in such a way that the chosen value represents the truth value of the proposition «x is A».

• reverse evaluation: a mark is put arbitrarily in a position c on the segment $[x_m, x_M]$. Then the elements of the ordered universe X are enumerated in the increasing order. The expert stops the enumeration when he thinks that the last presented element x_1 of X corresponds to the truth value c. In the case of trapezoidal functions, a given mark c can be associated with two elements of X and the enumeration goes on after the first stop in x_1, until the expert stops it again in

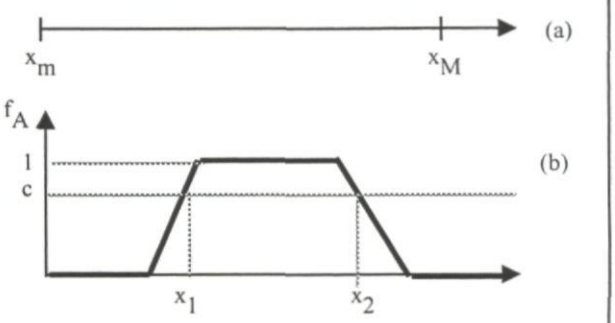


Figure 13. Direct and reverse evaluation of membership function

x_2 (figure 13b).

This reverse evaluation can be used to confirm the results of the direct evaluation.

• comparison of intervals: the experts are asked to compare pairs of intervals, for any randomly chosen pair of elements of X.

These three methods can be used individually. Practical experiments prove that they produce similar membership functions.

A classical knowledge elicitation method is the *repertory grid* [7], [42]) and it is used to ask an expert to explicit the meaning of vague terms, by means of other vague terms. It can be used to provide a numerical representation of vague terms. The linguistic results of the repertory grid are translated into a numerical representation.

For each linguistic value a, the expert is asked to give an opposite value b, for instance "short" for "long", "weak" for "strong"... A bipolar scale is obtained this way, which is graduated from 1 to 5, for instance, as follows:

1 = very a, 2 = a, 3 = medium, 4 = b, 5 = very b.

The scale can be graduated from 1 to 3 or to 7...

A method based on the construction of *α-cuts* has also been proposed [46] to determine a membership function. An α-cut of the fuzzy set A is a subset of X with membership grades at least equal to α, for $0 \leq \alpha \leq 1$.

We suppose that we are given an ordered list of degrees $\alpha_1 \leq \alpha_2 \leq \ldots \leq 1$ and we create the subsets of X of levels α_1, α_2..., with $A_\alpha = \{x \mid f_A(x) \geq \alpha, x \in X\}$, with $A_1 = \text{ker}(A)$. The membership function is deduced from the α-cuts (Figure 14)

There exist some other methods to construct membership functions directly [59] [15], we have presented the main

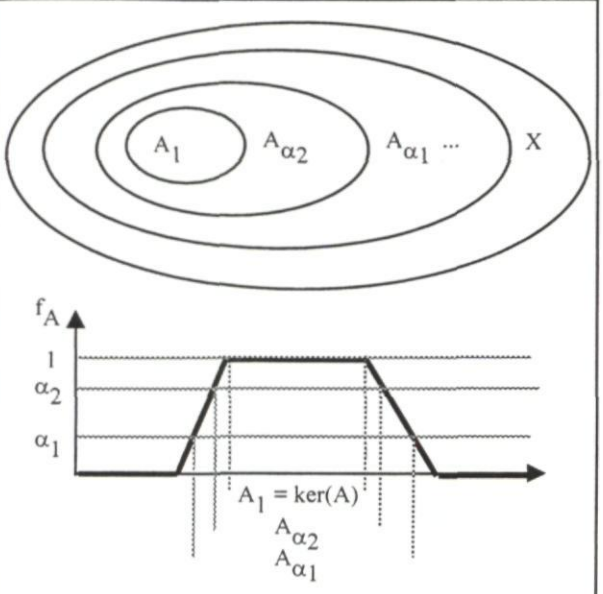


Figure 14. Nested α–cuts of A

directions. Let us remark that the functions which are constructed have generally a non-empty core, and the reference universe X is supposed to be ordered.

Such numerical representations are interesting to aggregate several interpretations of the same linguistic term and to reach a common representation by consensus, by means of one of the available aggregation methods. They are also interesting to take into account the relativity of the linguistic terms. For instance, "far" has a different meaning for a child walking in the street or for a pilot in a plane and the numerical representation reflects the environment of the expert asked to determine the membership function.

6. FUZZY LEARNING

Inductive reasoning is very useful in the case where a training set of examples is available and we want to generalize the knowledge obtained from these examples to any new case. It is based on a relationship between characteristics of every completely known situation and a decision or a class which has been assigned to this situation. Inductive reasoning methods construct general schemes representing this relationship, in order to enable the user to determine the class or decision corresponding to any new situation described by means of its characteristics. It provides a learning of decision-making.

6.1. General inductive learning method

We suppose that we are given a training set of examples described by means of attributes $A_1,\ldots, A_n$ defined on universes $U_1,\ldots, U_n$ (for instance, the size, the location, the environment ... of a position). The descriptions are either numerical (precise or approximative) or symbolic (given by a linguistic term) values. Each example is also associated with a class or decision c_i taken in a list $C = \{c_1,\ldots c_m\}$.

For instance, we have the training set given in Table 1.

New examples are proposed to the system, only associated with values of the attributes. A class of C must be assigned to them. For instance, the size is large, the location is dowtown and the environment is "houses". Which class will be assigned to this position?

size	*location*	*environment*	*class*
1223 m^2	close to town	houses	not acceptable
small	downtown	houses	not acceptable
540 m^2	close to town	fields	acceptable
1100 m^2	12 km from town	fields	acceptable
284 m^2	downtown	park	not acceptable
...	...	...	...

Table 1 : Example of a training set

In the case of symbolic values of attributes, methods such as ID3 [44] or C4.5 [45] allow to solve this problem. They are based on the construction of a tree of attributes, which represents the selection of most relevant attributes to determine the class. Attributes are associated with vertices of the tree and possible values of a given attribute are associated with the edges coming out of the vertex. Once the tree is constructed, we determine to which values of attributes indicated on the tree the new case corresponds (Figure 15).

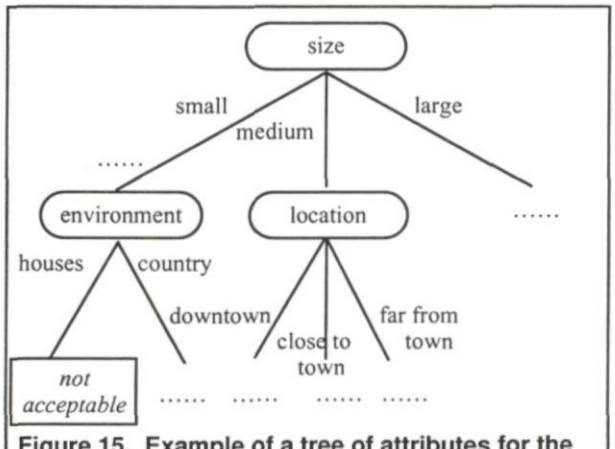


Figure 15. Example of a tree of attributes for the identification of the class of a position

In many cases, the values of attributes are imprecise or qualitative, or the values are numerical-symbolic, which means that they are numerical for some examples and symbolic for other ones. The classes can also be imprecise, with some situations belonging to several classes. The classical methods must be transformed to take care of these problems.

A solution consists in merging both kinds of values in a fuzzy representation.

6. 2. Fuzzy inductive learning

The tree of attributes is constructed from the root to the leaves, by choosing at each step, the attribute which provides the best information with respect to the identification of classes in the training set [12].

Each attribute A_i is associated with a partition of its possible values. In the case of a symbolic attribute, the elements of the partition are directly obtained from the values met in the training set. In the case of a precise numerical attribute, the partition is obtained by a splitting U_i in a small number of intervals. In the case of imprecise or numerical-symbolic values, a fuzzy partition is used, for instance, with n_i fuzzy

classes v_{ij} of U_i with membership functions f_{ij}. Such a representation accepts non-fuzzy representation as a particular case (with membership functions lying in {0, 1}). Such a partition is generally provided by experts of the domain. It can also be constructed automatically, for instance by a mathematical morphology-based method [33] [34] or by means of genetic algorithms.

In the case where the value of an attribute A_i is unknown for a given example, it can be replaced by a membership function equal to 1 in every point of U_i.

An attribute is chosen at the root of the tree, optimizing the following fuzzy entropy among all attributes:

$E^*(A_i) = - \sum_j P^*(v_{ij}) \sum_k P^*(c_k / v_{ij}) \log P^*(c_k / v_{ij})$,

where P* is the fuzzy probability of the event "the value of A_i is v_{ij}":

$P^*(v_{ij}) = \sum_u f_{ij}(u)\, p(u)$,

$P^*(c_k / v_{ij}) = P^*(c_k, v_{ij}) / P^*(v_{ij})$,

where p is the probability density estimated from the frequencies in the training set.

The root is then associated with this chosen attribute A_i, and edges coming out of the root are associated with all values v_{ij}, $j = 1, \ldots n_i$. The training set is splitted into n_i sub-training sets and the process is repeated at the terminal vertex of every constructed edge, on the corresponding sub-training set.

The construction stops when a class can be assigned to the end of a path (figure 16).

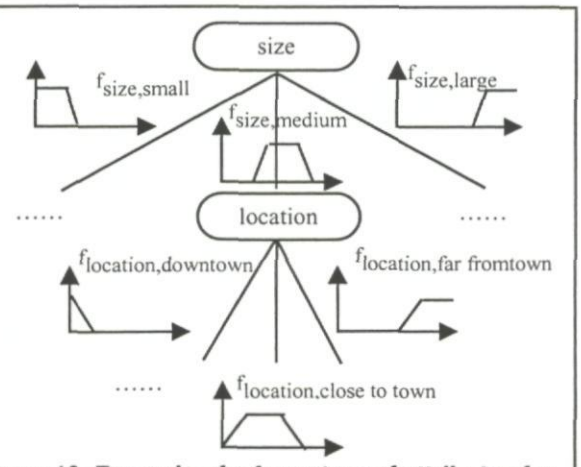


Figure 16. Example of a fuzzy tree of attributes for the identification of the class of a position

It is possible to express the knowledge derived from the tree of attributes by means of fuzzy rules as follows: let $A_{l_1}, \ldots, A_{l_p}$ be the attributes on a path of the tree from the root to a terminal vertex. If the value of A_{l_r} is $f_{l_r j_r}$ on this path and c_k is the class assigned to the terminal vertex, the rule has the form:

if A_{l_1} is $f_{l_1 j_1}$ and ... A_{l_p} is $f_{l_p j_p}$ then the class is c_k.

Once the tree of attributes is constructed, we are able to assign a class to any new example e described with a fuzzy (or crisp as a particular case) value of every attribute A_i represented by a membership function g_i on U_i. We need to compare the description of e with the descriptions associated with all paths of the tree. We use a measure of satisfiability, for instance the following, to measure the fitness of g_i with f_{ij}, with $j = 1, \ldots n_i$:

$S(g_i, f_{ij}) = (\int_{U_i} \min(g_i(u), f_{ij}(u))du) / (\int_{U_i} g_i(u)du)$,

for any attribute on a given path.

Then, we aggregate these degrees obtained for all the attributes $A_{l_1}, \ldots A_{l_p}$ of this path.

We obtain a compatibility degree, for instance as follows:

$Deg(g_{l_1}, \ldots, g_{l_p} / f_{l_1 j_1}, \ldots, f_{l_p j_p}) = \prod_{r=1 \ldots p} S(g_{l_r}, f_{l_r j_r})$.

Finally, we associate example e with the class c_k of C assigned to the terminal vertex of the path with the greatest compatibility degree. We can also use this compatibility degree as a membership grade defining a fuzzy class assigned to e, in the case where no dominant class appears.

6.3. Utilizations of fuzzy inductive learning

A real world application of this method can be found in [8] for cancer diagnosis from mammographical images.

It is possible to use such an approach to deduce imprecise categories from imprecise descriptions [13] [37] or to choose a possibilistic approach [32] to produce an inductive learning algorithm which is robust with regard to knowledge uncertainties. It can also be used to elicit deduction rules for a fuzzy expert system or control system [55].

In the case where there exist ill-classified examples in the training set, fuzzy quantifiers (fuzzy representations of "generally" or "in most cases", for instance) can be used to take into account descriptions which satisfy almost all the examples, for instance [25].

7. CONCLUSION

The association of several advanced techniques of knowledge processing creates a synergy between the advantages of these techniques which has been recently pointed out, particularly by L.A. Zadeh, the father of fuzzy logic, who promotes what is now called "soft computing". The main techniques providing such a synergy are fuzzy logic, neural networks and genetic algorithms, but some others can be efficiently used, such as probabilistic methods for instance.

In particular, fuzzy systems are confronted with the problem of learning or tuning their parameters. Using neural networks or genetic algorithms provides solutions to this problem, but there exist other kinds of methods to solve this key problem.

We have presented the main streams of development of such combined utilizations of techniques and learning methods, but this paper does not pretend to be exhaustive, since there exist a large number of papers dealing with this topic, with many different approaches, the possibilities of associating techniques are countless and the subject is then very rich.

8. REFERENCES

[1] Aladenise N., Bouchon-Meunier B., Acquisition de connaissances imparfaites : mise en évidence d'une fonction d'appartenance, Revue Internationale de Systémique 11, 1 (1997)

[2] Bäck T., Kursawe F., Evolutionary algorithms for fuzzy logic : a brief overview, Proceedings of the fifth international conference IPMU . Information Processing and Management of Uncertainty in Knowledge-Based Systems (1994), pp. 659-664.

[3] Berenji H., A Reinforcement Learning-Based

Architecture for Fuzzy Logic Control, Int. J. Approximate Reasoning 6, pp. 267-292, 1992.

[4] Bezdek J. C., Computing with uncertainty, IEEE Communications Magazine 30, 9, 1992, pp. 24-36

[5] Black M., Vagueness, Philosophy of Science 4 (1937) pp. 427-455.

[6] Bobrowicz O., Choulet C., Haurat A., Sandoz F., Tebaa M., A method to build membership functions, application to numerical/symbolic interface building, Proceedings of the fifth international conference IPMU, Information Processing and Management of Uncertainty in Knowledge-Based Systems (1990), pp. 136-142.

[7] Boose J. H., A knowledge acquisition program for expert systems based on personal construct psychology, International Journal of Man-Machine Studies 23 (1985) pp. 495-525.

[8] Bothorel S., Bouchon-Meunier B., Muller S., A fuzzy logic based approach for semiological analysis of microcalcifications in mammographic images, International Journal of Intelligent Systems, 1997 (to appear).

[9] Bouchon-Meunier B., La logique floue, Que sais-je? n° 2702, Presses Universitaires de France, 2ème édition (1994).

[10] Bouchon-Meunier B., La logique floue et ses applications, Addison-Wesley (1995).

[11] Bouchon-Meunier B., Marsala C., Ramdani M., Inductive learning and fuzziness, Scientia Iranica 2. 4., pp. 289-298, 1996

[12] Bouchon-Meunier B., Marsala C., Ramdani M., Learning from imperfect data, in H. Prade, D. Dubois, R.R. Yager (eds.), Fuzzy set methods in information engineering : a guided tour of applications, JohnWiley & Sons, pp. 139-148, 1997.

[13] Botta M., Giordana A., Saitta L., Learning Fuzzy Concept Definitions, Proc. 2nd International IEEE Conference on Fuzzy Systems, San Francisco (1993)

[14] Chauvin S., Evaluation des théories de la décision appliquées à la fusion de capteurs en imagerie satellitaire, Thèse de Doctorat d'Université, Nantes (1995).

[15] Chen J.E., Otto K.N., Constructing membership functions using interpolation and measurement theory, Fuzzy Sets and Systems 73 (1995) pp. 313-327.

[16] Civanlar M.R., Trussell H.J., Constructing membership functions using statistical data, Fuzzy Sets and Systems 18 (1986) pp. 1-13.

[17] Cordón O., Herrera F., Lozano M., A classified review on the combination fuzzy logic-genetic algorithms. Bibliography 1989-1995, in : Genetic algorithms and fuzzy logic systems, Soft Computing Perspectives, (E. Sanchez, T. Shibata and L.A. Zadeh, eds.), World Scientific 1997, pp. 209-240.

[18] Fagarasan F., Negoita M. G., A genetic algorithm with variable length geotypes, applications in fuzzy modeling, Proceedings 4th European Conference EUFIT, Aachen 1996, pp. 405-409

[19] Gaines B.R., Fuzzy and probability uncertainty logics, Information and Control 38 (1978) pp. 297-323.

[20] Giles R., The concept of grade of membership, Fuzzy Sets and Systems 25 (1988) pp. 297-323.

[21] Glorennec P.Y., Tutoriel sur le neuro-flou, , Congrès sur les sous-ensembles flous, Nîmes, 1992.

[22] Heider H., Tryba V., Mühlenfeld E., Automatic design of fuzzy systems by genetic algorithms, Proceedings of the Fifth International Conference IPMU, Information Processing and Management of Uncertainty in Knowledge-Based Systems (1994), pp. 665-670.

[23] Hisdal E., Are grades of membership probabilities ? Fuzzy Sets and Systems 25 (1988) pp. 325-348.

[24] Hisdal E., Reconciliation of the Yes-No versus grade of membership dualism in human thinking, in : Readings in Fuzzy Sets for Intelligent Systems (Dubois D., Prade H., Yager R.R., eds.), 1993, pp. 854-860.

[25] Kacprzyk J., Iwanski C., Fuzzy Quantifiers in Inductive Learning with Perception of Errors in Data, Proc. 1st International IEEE Conference on Fuzzy Systems, San Diego (1992).

[26] Keller J.M., Krishnapuram R., Rhee F. C.-H., Evidence aggregation networks for fuzzy logic inference, IEEE Trans. Neural Networks, 3, 5, pp. 761-769, 1992.

[27] Keller J.M., Yager R.R., Tahani H., Neural network implementation of fuzzy logic, Fuzzy Sets and Systems 45, pp. 1-12, 1992.

[28] Keller J.M., Tahani H., Implementation of conjunctive and disjunctive fuzzy logic rules with neural networks, International Journal of Approximate Reasoning, 6, pp. 221-240, 1992.

[29] Krishnapuram, R., Generation of membership functions via possibilistic clustering, Proceedings of the third IEEE Conference on Fuzzy Systems, IEEE World Congress on Computational Intelligence - vol. 2 (1994) pp. 902-908.

[30] Krishnapuram, R., Keller J.M., A possibilistic approach to clustering, IEEE Transactions on Fuzzy Systems 1, 2, 1993, pp. 98-110.

[31] Maeda A., Someya R., Funabashi M., A Self-Tuning Algorithm for Fuzzy Membership Functions Using Computational Flow Network, Proceedings IFSA Congress, Bruxelles, volume Artificial Intelligence, pp. 129-132 (1991).

[32] Maher P.E., Saint-Clair D., Uncertain reasonning in an ID3 machine learning framework. 2nd FUZZ-IEEE Congress, San Francisco, pp. 7-12, 1993.

[33] Marsala C., Fuzzy partition inference over a set of numerical values. Rapport 95/22, LAFORIA-IBP, 1995.

[34] Marsala C., Bouchon-Meunier B., Fuzzy partitioning using mathematical morphology in a learning scheme, 5th FUZZ-IEEE Congress, New Orleans, 1996

[35] Ménage X., Système flou pour le contrôle de qualité, Thèse de l'Université Paris VI, 1996.

[36] Ménage X., Hartani R., Synthèse des méthodes d'association des techniques neuronales et des techniques floues, rapport technique LAFORIA 93/23 (1993).

[37] Narazaki H., Ralescu A.L., A Method for Inducing Category Membership Function from Examples, Actes International Conference IPMU'92, Palma de Mallorca (1992) .

[38] Norwich A.M., Turksen I.B., A model for the measurement of membership and the consequences of its empirical implementation, Fuzzy Sets and Systems 12 (1984) pp. 1-25.

[39] Pal S.K., Mitra S., Multilayer perceptron, fuzzy sets and classification, IEEE Trans. Neural Networks, 3, 5, pp. 683-697, 1992.

[40] Pedrycz W., Identification in fuzzy systems, IEEE Trans. Systems, Man and Cybernetics 14, pp. 361-366, 1984.

[41] Pedrycz W., Hirota K., Takagi T., Fuzzy associative

memories: concepts, architectures and algorithms, Proceedings of IFES'91, Yokohama, 1991, pp. 163-174.

[42] Plaza E., Alsina C., Lopez de Mantaras R., Aguilar J., Agusti J., Consensus and knowledge acquisition, Proceedings of the fifth international conference IPMU. Information Processing and Management of Uncertainty in Knowledge-Based Systems (1986), pp. 294-306.

[43] Procyk T.J., Mamdani E.H., A Linguistic Self-Organizing Process Controller, Automatica 15, pp. 15-30 (1979).

[44] Quinlan J.R., Induction of decision trees. Machine Learning 1(1) pp. 86-106, 1986.

[45] Quinlan J.R., C4.5: Program for machine learning. Morgan Kaufmann, San Mateo, Ca., 1993.

[46] Ralescu D., A survey of the representation of fuzzy concepts and its applications", in : Gupta, M.M., Ragade, R.K., Yager, R.R., Advances in fuzzy set theory and applications, North-Holland, 1979.

[47] Shi-Zhong He, Shaohua Tan, Chang-Chieh Hang, Pei-Zhuang Wang (1993) Control of Dynamical Processes using an on-line Rule-Adaptative Fuzzy Control System, Fuzzy Sets and Systems 54, pp.11-22.

[48] Smith S.M., Comer D.J., Automated Calibration of a Fuzzy Logic Controller using a Cell State Space Algorithm, IEEE Control Systems Mag., August, pp. 18-28 (1991).

[49] Sugeno M., An Introductory Survey of Fuzzy Control, Information Science 36, pp. 59-83 (1985) .

[50] Sugeno M., Murakami K., An experimental study on fuzzy parking control using a model car, in : Industrial Applications of Fuzzy Control, M. Sugeno (ed.), Elsevier Science Pub. North Holland, 1985, pp. 125-138.

[51] Sugeno M., Takagi T., A New Approach to Design of Fuzzy Controller, in : P.P. Wang, S.K. Chang (dir.) Advances in Fuzzy sets, possibility theory and applications, Plenum, New York (1983).

[52] Takagi H., Fusion Technology of Fuzzy Theory and Neural Networks - Survey and Future Directions, Proc. Intern. Conf. on Fuzzy Logic and Neural Networks, IIzuka, pp. 13-26 (1990).

[53] Takagi T., Sugeno M., Fuzzy Identification of Systems and its Applications to Modeling and Control, IEEE Trans. Systems, Man and Cybernetics, 15, 1, pp. 116-132 (1985) .

[54] Wang P.Z., "From the fuzzy statistics to the falling random subsets", in Advances in Fuzzy Sets, Possibility Theory and Applications, Wang, P.P. (ed.) (1983) pp. 81-96.

[55] Weber R., Automatic Acquisition for Fuzzy Control Applications, Proc. International Symposium on Fuzzy Systems, Iizuka (1992) .

[56] Yager R.R., Connectives and Quantifiers in Fuzzy Sets, Fuzzy Sets and Systems 40, 1, pp. 39-75 (1991).

[57] Yager R.R., Filev D.P., Essentials of fuzzy modeling and control, John Wiley 1994

[58] Zhang L., Structural and functional quantization of vagueness, Fuzzy Sets and Systems 55 (1993) pp. 51-60.

[59] Zimmerman H.-J., Zysno P., Quantifying vagueness in decision models, European Model Of Operational Research 22 (1985) pp. 148-158.

Dual Waveband Infra-Red Target Tracking and Acquisition System

M Bernhardt, M Welch, DERA Farnborough
D L Toulson, King's College London

1. INTRODUCTION

The introduction of Infra-Red (IR) systems onto military fast-jet platforms, particularly Forward Looking Infra-Red (FLIR) has extended the operational envelope of these aircraft by giving them a night flying capability. FLIR systems commonly use the 8-12 micron wavelength band of the IR spectrum as this is part of the IR spectrum where there is good atmospheric transmission. Conveniently, objects at room temperature have peak IR emissions within this band too. Some FLIR systems use the 3-5 micron band, which also has good atmospheric transmission and where hotter objects, such as aircraft exhaust plumes, have their peak emission.

Originally FLIR systems were conceived as navigation aides for night flying operations, but it became apparent that there was the possibility of using them for more than this. By applying image processing and analysis techniques to the images, targets of interest to the pilot could be automatically selected and cued to the aircrew. The term Automatic Target Recognition (ATR) is often used to describe this process.

The field of ATR is large, involving many approaches and a wide variety of mathematical and computational techniques. In this paper we describe an ATR system using FLIR imagery which has the following features:

- It uses two wavebands of digital FLIR imagery simultaneously to extract additional scene information by data fusion
- It is designed to work for targets that occupy only a few pixels in the image
- It uses a neural network for initial clutter rejection
- It uses a novel probabilistic tracking methodology
- It uses a committee of neural networks for final confirmation of targets
- It is being evaluated on real flight trials data

The structure of this paper is as follows: in section two we give a brief outline of a simple ATR process and describe its limitations; Section 3 discusses the improved architecture we propose, its extension to two wavebands of FLIR imagery and the novel aspects of the neural networks and their training; in Section 4 we discuss the preliminary results and ongoing evaluation; in Section 5 we describe a number of novel enhancements to the neural networks that have been investigated; finally Section 6 concludes and summarises the paper.

2. A CONVENTIONAL ATR SYSTEM

Before discussing the novel ATR system that is the main subject it is useful to describe a simpler system in order to introduce ATR in general and illustrate the shortcomings of simple approaches, which we attempt to

Paper presented at the AGARD MSP Lecture Series on "Advances in Soft-Computing Technologies and Applications in Mission Systems", held in North York, Canada, 17-18 September 1997; Amsterdam, The Netherlands, 22-23 September 1997; Madrid, Spain, 25-26 September 1997; Ankara, Turkey, 6-7 October 1997, and published in LS-210.

overcome in the more advanced system. The task of all the ATR systems that we shall consider is to cue the location of ground based military vehicle targets to the pilot of an airborne (usually fast-jet) platform. Figure 1 shows a block diagram of an ATR processing chain which takes as its input a stream of sequential images from a single waveband FLIR system.

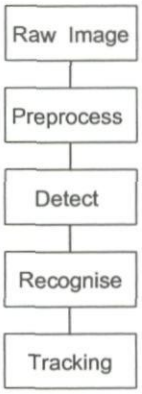


Figure 1. Simple ATR process

In this processing chain data flows from top to bottom.

The first stage is the input of the 'raw image'. This is a frame from the digitised FLIR imagery. Typically the number of pixels will vary from 256x256 up to 1536x1024 (for one band of the FLIR used to evaluate the more sophisticated algorithm). The digitisation accuracy can also vary: 8 bits is common, but 12 or 14 bit systems are also available.

The pre-processing stage is necessary because FLIR imagery often contains artefacts (such as banding or swathing) which depend on the specific sensor technology employed in the FLIR. Simple image processing techniques can remove these unwanted artefacts.

The detection processing stage is more complex. There are a very large number of techniques available, ranging from simple matched filter approaches, to highly sophisticated non-linear adaptive processing. The output of this stage of the system is a list of coordinates, within this frame of the image stream, of potential targets, or target-like objects.

One could stop here and declare to the aircrew the results of the detection stage. Unfortunately there is usually a high False Alarm Rate (FAR) associated with the detection processing, which renders this option impractical. There are many ways of reducing the false alarms. Simple techniques such as first checking the consistency of detections over time can remove many false alarms. Experience has shown that even by employing a number of simple techniques the false alarm rate, although dramatically reduced, is still too high to be practical. To remedy this, it is the function of the recognition box in Figure 1 to further reduce the false alarm rate. One very effective technique which we have employed in the recognition stage is a neural network.

The neural network is a standard Multi Layer Perceptron (MLP). A patch of 9x9 pixels is extracted from the image centred on the location of each detection and the 81 image intensity values used as the inputs to the network. The output layer of the network consists of two neurons, one for targets and one for non-targets. The network is trained on a large number of sample patches from image sequences and optimised for the number of neurons in the hidden layer. Typically an 81-16-2 neuron network was found to be optimal. When evaluating the performance of the system different image sequences were used so that all data passed to the network was previously unseen.

Finally the pixel coordinates of detections which are classified by the network as targets are passed to the tracking stage. Various types of tracking are possible including the more sophisticated tracker described later in this paper. Typically though, a standard Kalman filter approach is used.

The above ATR process is certainly practical and has been implemented in software and

real-time flightworthy hardware, however there are a number of possible improvements that can be made.

The most severe limitation of this processing chain is its inability to cope reliably with low contrast targets which are only detected intermittently. This is because the system only tries to recognise targets that have passed the threshold of the detection process, no attempt is made to classify a target which although tracked for many frames, suddenly fails to pass the detection threshold for a few frames. Such targets are often lost. The new proposed architecture does not suffer from this defect.

The other potential improvement that can be made to this ATR system is to make use of dual-waveband FLIR imagery. That is two separate image sequences that are both spatially and temporally registered. The two separate wavebands contain complimentary information about targets and backgrounds which aides the discrimination process. In addition one might expect such a system to be robust in its performance over a wider range of weather conditions as the atmospheric transmission of each band is affected differently by the prevailing weather.

3. NEW ATR ARCHITECTURE

3.1 Overview

In order to improve over the simple system presented in Section 2 a new architecture was devised with a number of novel features. In this section we first discuss the single waveband version of the architecture in some detail. Its extension to two wavebands, which is quite straightforward is given in Section 3.4. Section 3.2 gives details of the probabilistic tracking methodology, and Section 3.3 discusses the neural network processing aspects of the system. Figure 2 shows a block diagram of the new architecture:

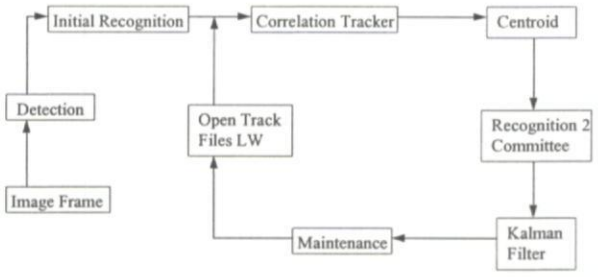


Figure 2 Enhanced ATR Architecture

The main difference between this architecture and the one given in Section 2 is the closed tracking loop. This enables targets which may have only been detected in one frame to be continuously tracked. Within the tracking loop is an embedded classification process which can keep trying to classify the object even if it is no longer passing the detection threshold. It can do this because the tracker combines a correlator with a Kalman filter, and is capable of tracking low contrast targets.

The other significant difference is the splitting of the classification process into two parts. The first is an MLP which has been trained on a low ratio of targets to false alarms, thus it is good clutter rejector, but not necessarily as good at recognising true targets. The second neural network (which is actually a committee network) has been trained on objects which pass the first network. Thus its training set consists of a much higher target to false alarm ratio, making it more discriminating. The details of these networks are presented in Section 3.3.

3.2 The Tracking Process

The tracking process consists of four boxes in the diagram shown in Figure 2: Correlation Tracker; Centroiding; Kalman Filter and maintenance. Although shown as separate boxes for convenience on the diagram they all fit within one mathematical framework, which is described below.

Consider a sequence of S consisting of T images. Let S be denoted as a set containing the individual image frames i.e.

$$S = \{ I_t, \quad t = 1...T \} \tag{1}$$

Suppose that a set of potential targets are detected in each image frame. Let this set be denoted D_t

$$D_t = \{ \mathbf{d}_{ti} \; i = 1...N_D(t) \} \tag{2}$$

where $\mathrm{d_{ti}}$ is the image plane vector of the i^{th} detected object in frame I_t and $N_D(t)$ is the number of detections in frame t.

A *trackfile* is a set of temporarily associated target locations corresponding to a single real object in the image plane. It will describe the motion of a tracked object from the point at which it is first detected by the system until the point at which it is no longer detectable. Let the contents of the i^{th} trackfile at time t, be denoted by the set $T_i(t)$. At the time t each trackfile T_i will contain a set of observed locations

$$T_i(t) = \{\mathbf{x}_{ti}, \mathrm{t} = \mathrm{t_s}, ..., \mathrm{t_e}\} \tag{3}$$

where t_s is the *initialisation time* for the trackfile (where the point at which the object is first detected by the system) and t_e is either the current time t or the time of the final observation of the target. Note that the target locations $\mathbf{x}_{ti}$ are not necessarily the same as the detected locations $\mathbf{d}_{ti}$.

A common approach to tracking is to update trackfiles at time t on the basis of the latest detections D_t. This is usually achieved using a two stage process. The detections are first *associated* with the trackfiles to find which detections correspond to which trackfiles. After association, the Kalman Filter estimates of the current locations and the one frame ahead predictions are updated. Tracks that do not have detections associated with them estimate the current location by using the Kalman Filter in *closed loop* mode.

Detections that do not correspond to any existing trackfile cause a new trackfile to be opened. Trackfiles that consistently fail to attract associated detections are deleted.

In this paper we propose a joint probabilistic framework for tracking using both the spatial and radiometric aspects of the target. This process does not require a continuous stream of target locations to maintain the tracking process. We shall then move on to examine the use of a probabilistic centroiding technique to reduce target *drift* over time.

Consider a trackfile $T_i(t)$ that is open at time t. Suppose we have just observed the $t + 1^{th}$ image, I_{t+1}. We wish to update our estimate of the current position of the tracked target in the light of the new image data I_t. Let the probability that the target has moved a distance Δx in the image plane between frames t and $t + 1$ be $p(\Delta x)$. This probability $p(\Delta x)$ may be written, using Bayes' Rule,

$$p(\Delta x \mid I_t) = \frac{p(I_t \mid \Delta x)\, p(\Delta x)}{p(I_t)} \tag{4}$$

where $p(I_t)$ is a normalising term that may be expressed as the integral of the product of the conditional and prior densities,

$$p(I_t) = \int_{\Delta x} p(I_t \mid \Delta x)\, p(\Delta x) d\Delta x \tag{5}$$

To find the probability $p(\Delta x \mid I_t)$ we need to calculate the following two densities

- $p(I_t \mid \Delta x)$ - The probability of the new image data being observed given a particular target motion hypothesis.
- $p(\Delta x)$ - The prior probability that the target will be observed to have a given inter-frame translation based on *a priori* information about the target available *before* the new image is observed.

Estimating $p(I_t \mid \Delta x)$

The density $p(I_t \mid \Delta x)$ relates to the probability of observing the new image I_{t+1} given an assumed target motion Δx between frames.

To estimate this we shall formulate a model of pixel intensities and use this model to derive an expression for the probability of observing the image (locally about the target) for a given inter-frame motion.
Consider two pixel patches of size M x M extracted from image frames I_t and I_{t+1}. Let the first patch, extracted from the t^{th} image, be centred about the location of the target in the t_{th} frame. Let the second patch, extracted from the $t+1^{th}$ image, be centred about the location of the target in the t^{th} frame *plus* the assumed inter-frame target motion Δx. Let the pixels of the first image patch be denoted by the vectors **p**

$$\mathbf{p} = (x,y) \quad x = 1...M \quad y = 1...M \tag{6}$$

Similarly let the pixels of the second patch be denoted by the vectors $\mathbf{p}_{\Delta x}$ where Δx is the assumed inter-frame motion.

Now, let us model the pixel intensities of the patch themselves as consisting of a signal with some additive Gaussian Noise. So, if the intensity of pixel **p** is denoted $i(\mathbf{p})$ then,

$$i(\mathbf{p}) = s(\mathbf{p}) + N(0, \sigma^2) \tag{7}$$

where $s(\mathbf{p})$ is the intensity of the signal at pixel location **p**.

Now consider the intensity differences of corresponding pixels in the two patches. Let us denote the size of these differences by $i_{\Delta x}(\mathbf{p})$.

$$i_{\Delta x}(\mathbf{p}) = i(\mathbf{p}) - i(\mathbf{p}_{\Delta x}) \tag{8}$$

If the patches have an offset Δx such that they are both centred upon the physical target (i.e. correctly registered) then the signal intensities of corresponding pixels in the two patches should be approximately the same. In this case the differences of the intensities of corresponding pixels, $i_{\Delta x}(\mathbf{p})$ will be a Random Variable that is the difference between two normal distributions.

Assuming independence we can now express the *likelihood* of observing the intensities of the two patches given an assumed inter-frame motion of Δx between frames as

$$p(I_t \mid \Delta\mathbf{x}) = \prod_{\mathbf{p}} p(i_{\Delta x}(\mathbf{p}) \mid \Delta\mathbf{x})$$

$$= \frac{1}{2\sigma\sqrt{\pi}} \exp\left(-\frac{i_{\Delta x}(\mathbf{p})}{\sqrt{2}\sigma^2}\right) \tag{9}$$

Estimating $P(\Delta\mathbf{x})$
If a target is tracked using a Kalman Filter then, at a given time t, the filter provides the following information

- A filtered estimate of the target's current state $\mathrm{x}(t|t)$.
- An estimate of the target's predicted state in the next frame $\mathrm{x}(t+1|t)$.
- An estimate of the uncertainty in the predicted state, expressed as a covariance matrix $P(t+1|t)$.

The value we wish to calculate, $p(\Delta\mathrm{x})$, is the prior belief that a target will move by the vector Δx between frames. We can estimate this by a probability density function constructed from the tracking predictions $\mathrm{x}(t+1|t)$ and the covariance matrix $\mathrm{P}(t+1|t)$.

Let the estimated inter-frame motion of a target obtained from the Kalman Filter be denoted x_k,

$$\mathrm{x}_k = \mathrm{x}(t+1|t) - \mathrm{x}(t|t) \tag{10}$$

The probability (according to the Kalman prediction) of the target moving by a vector Δx between frames is then

$$p(\Delta\mathrm{x}) = exp\left((\mathrm{x}_k - \Delta\mathrm{x})^{\mathrm{T}} \Sigma^{-1} (\mathrm{x}_k - \Delta\mathrm{x})\right) \tag{11}$$

where Σ is the covariance of the target location.

Target Centroiding

A difficulty encountered in tracking targets using solely their inter-frame correlations is the tendency for the position of the target to *drift* over time. This is due to the lack of any explicit criteria as to which part of the target is required to be tracked. For instance, suppose a trackfile is initialised with a location that corresponds to the physical position of the engine of a vehicle. Now suppose that in the next frame, the correlation tracker erroneously generates a new target location corresponding to the exhaust plume of the vehicle. In the subsequent frames the correlation tracker will then attempt to find the location of the exhaust plume of the target rather than the engine. The tracker has effectively *forgotten* the part of the physical object it was originally tracking.

One way in which this effect can be reduced is to add additional criteria to the selection of target locations. For instance, if we require that the location of a target should correspond to the geometric centre of the object, then we can continuously correct the location of the targets using a centroiding process. This will prevent the targets' locations *drifting* over time. The centroiding correction will be applied after the joint Kalman Filter/ correlation association has been made.

Consider a pixel patch extracted about a target. Let us model the intensity of pixels in the patch as being generated by a finite mixture of two distributions, background and target, i.e.

$$p(i) = \sum_{\alpha=1}^{2} \pi_\alpha p(i|\alpha) \qquad (12)$$

where π_1 is the prior probability that a pixel is a target pixel, π_2 is the probability of a pixel being a background pixel, $p(i|1)$ is the probability that a pixel has intensity i given that it is a target pixel and $p(i|2)$ is the probability that a pixel has intensity i given that it is a background pixel.

Let us also assume that the class conditional probability distribution functions may be modelled by Gaussian Distribution

$$p(i|\alpha) = \frac{1}{\sqrt{2\pi}\sigma_\alpha} \exp\left(\frac{i-\mu_\alpha}{\sigma_\alpha^2}\right) \quad (13)$$

where μ_α and $\sigma^2{}_\alpha$ are the mean and variances of the conditional distributions.

We have derived a method of finding these distributions using the EM (Expectation Maximisation) algorithm [3].

Once we have calculated the probabilities of each pixel being a part of the target, we can then find the spatial moments of the pixel target probabilities.

$$c = \sum_{\mathbf{p}} p(i(\mathbf{p})|1)\mathbf{p} \qquad (14)$$

This centroid value is then used to correct for any drifting of the tracker.

The final part of the tracking process to note is the maintenance function which simply kills trackfiles whose time weighted probability of being targets (according to the neural net) has fallen below a threshold value. This is necessary to prevent the indefinite proliferation of trackfiles.

This tracking process has been implemented and tested and has been shown to be reliable and robust. We now move on to discuss the neural network aspects of the system.

3.3 Neural Networks

Referring to Figure 2 there are two neural networks involved in the classification process. Before moving on to discuss the specifics we discuss some more general issues regarding the networks and their training.

To train an MLP network to correctly discriminate between targets and false alarms it is necessary to obtain a large set of labelled image patches. These classified patches are obtained by allowing a human operator to manually label sets of target cues generated by the system, i.e. objects passing the threshold of the detection process, as targets or non-targets. These patches and manual classifications are then stored as the training set. The network may then be trained using Backpropagation of Error [8]. At its simplest the training process involves the following iterative procedure

- Present the network with an image patch from the training set
- Produce an output response
- Compare this output response with the desired (manually classified response)
- Alter the weighted connections between processing nodes such that the error between the desired and actual network responses is reduced

Presenting all of the patterns in the training set and altering the weights once is referred to as an *epoch* of the training algorithm. The summed error of classifying all of the training examples is referred to as the *training error*. There are various methods for deciding when to stop training of the network. The method used here was to have another (different) set of manually classified patches, known as the validation set, which are presented to the network at the end of each epoch. The summed error in classifying these patches, known as the *cross-validation error* is measured and training continued until this error is minimised. This is done to avoid overtraining, i.e. learning the specific training set, but losing the ability to generalise to similar data.

The training sets obtained by this process of manually classifying the detection filter responses usually contain at least ten times as many false alarms as targets due to the deliberately low thresholding chosen for the detection process. Training an MLP on such data is prone to difficulties caused by the local minima of the objective training error function obtained by the network trivially classifying all the input patches as false alarms. A number of researchers faced with similar problems have opted to artificially weight the training set to contain equal numbers of targets and false alarms. However, although this may lead to better training performance, the performance will degrade when the network is expected to classify the actual data given to it by the target cuer which will contain the original ratio of targets to false alarms. In this case a disproportionately high number of false alarms will be mis-classified as targets due to their incorrect relative weighting in the training set. To avoid this problem we have devised the following strategy of *class unbiasing*.

Class Unbiassing

Consider a training set consisting of T target examples and F false alarms. The usual error function minimised using back-propagation is

$$E(t) = \frac{1}{2}\sum_i \left|\vec{o}_i(t) - \vec{d}_i\right|^2 \qquad (15)$$

where d_i represents the desired vector output from the network when presented with the *ith* training vector and $o_i(t)$ the network response vector after t epochs. We propose the following modified error function

$$E(t) = \frac{1}{2}\sum_i \left|\vec{o}_i(t) - \vec{d}_i\right|^2 C_i(t) \quad (16)$$

where $C_i(t)$ is an exponentially decaying class un-biassing term defined to be

$$C_i(t) = \begin{cases} \left(\dfrac{F}{T+F}\right)^{(1+t/T_0)^{-1}} & \text{target} \\ \left(\dfrac{T}{T+F}\right)^{(1+t/T_0)^{-1}} & \text{nontarget} \end{cases} \qquad (17)$$

where T_0 is a constant determining the rate of decay of the un-biassing.

The effect of this un-biassing term is that, at the beginning of training, contributions to the overall error from the smaller number of target examples are increased. This prevents the tendency of the network to classify everything as a false alarm at the start of training. However as training progresses the class un-biassing effect is relaxed and after a sufficiently large number of iterations the modified error term (16) tends to the usual error term (15). This technique has been shown to lead to more robust training using both the *backprop with momentum* [8], and the *quick-prop* [9] training algorithms.

There is a further interesting feature associated with the training of these networks, and that is the size of the data-set. We are using real trials data which is expensive to collect and we therefore wish to make the best possible use of this data. For training and evaluation purposes it is desirable to use large quantities of data for each experiment. Typically this might be a few minutes worth of FLIR imagery which would be divided up into three sets: The training set - from which patches are extracted for training the network; the validation set - to determine when to stop training; and the test set for evaluating the performance of the overall ATR algorithm. Typically we must deal with training sets that consist of half a million patterns (each of which is a 9x9 array of pixels). Even with fast computers this is a formidable training problem and so a method was devised to reduce the size of the training set, whilst minimising the effect on training performance. This technique is known as *data-set collapsing*.

Data-set Collapsing

When a neural network trains it is establishing decision boundaries between different classifications in a high dimensional space, in our case 81-dimensional. It is natural to imagine that data points close to these boundaries are more important to the learning process than data points which are deep within regions of one classification.

A common approach to this has been to select *boundary examples* to train the network. These are patterns that lie close in input space to a discrimination boundary between classes. The problem with such an approach is that it will distort the class conditional probability density functions of the training samples. Here we present a slightly different approach that attempts to maintain, as far as possible, the original distribution of the training data. Instead of eliminating non-boundary patterns, we instead attempt to replace clusters of them by single weighted example patterns. The algorithm proceeds as follows.

1. Select a training pattern $\mathbf{x_0}$
2. Locate the nearest k training patterns contained in the same training set, x_i , i=1,...,k. Near is defined in the Euclidean sense: $d_i=\|\mathbf{x}_0-\mathbf{x}_i\|$.
3. If all of the k nearest neighbour training patterns are from the same class (all targets or false alarms) then perform collapse (step 4) otherwise return to step 1.
4. Replace the selected pattern $\mathbf{x_0}$ and the k nearest neighbours $\mathbf{x_i}$, i=1,..., k with a single training pattern corresponding to the component-wise mean of the k+1 vectors, i.e.

$$\mathbf{m} = \frac{\sum_{i=0}^{k} \mathbf{x}_i}{k+1}$$

   and give the collapsed vector $\mathbf{m}$ a weight of k+1.
5. Repeat step 1.

This procedure continues until all of the training patterns have either been considered for collapse or have already been collapsed.

Experiments with this technique have shown that it works well typically giving less than a

1% degradation in learning performance for a factor of 10 reduction in the size of the training set. Although training takes place using a collapsed data-set the cross validation error is computed using un-collapsed data.

Optimising Number Of Hidden Nodes

A recurring difficulty with the MLP architecture is the question of the number of hidden nodes to use. The number of input nodes is determined by the data representation - 81 in our case for the pixel intensities of the 9x9 patch. The number of output nodes is determined by the number of classes - 2 in our case. In order to obtain an estimate for the best number of hidden nodes we adopted a 'brute force' approach. Networks were trained using a number of different hidden nodes and the best number of hidden nodes chosen on the basis of the training error. This method indicated that the best number of hidden nodes was close to 16, so this architecture was chosen for the final system.

The above techniques were used to train the network labelled as 'Initial Recognition' in Figure 2. The networks used to form the committee for the recognition stage within the tracking loop were trained in a similar way, except that their training sets were restricted to patches that 'Initial Recognition' had classified as targets.

The Committee Network

A committee of neural networks is simply a number of networks each trained on the same data, but starting from different randomised weight configurations. The outputs of the members of the committee are then combined in some way to produce a classification decision.

The reason for implementing the committee structure is to increase the accuracy and generalisation capability of the system. Each network within the structure has been trained into a different local minima of the error function. This implies that each network is modelling the data in a different way. We have no a-priori way of knowing which way of modelling the data is the best, so in an ideal world we would wish to integrate or average over all possible models that fitted the data in order to find the least biased classification system. In practise such an average is not possible, but it can be approximated by averaging over a number of networks - hence the idea of a committee of neural networks.

There are several methods that one might use to combine the outputs of the members of the committee to produce a decision, these include:

- Majority Voting - In this case each network makes a hard classification locally and the most common classification decision is accepted
- Simple Averaging - the outputs from each network are averaged with equal weighting
- Weighted averaging - as above except unequal weights are used to bias the average. This method requires a means to determine the weighting factors
- Non-linear combination - the output responses are combined in a non-linear way, possibly by means of another neural network.

In the light of the above discussion about the philosophy behind the committee structure we opted for the weighted averaging approach. To determine the weighting factors to use it was decided to use the training errors associated with each of the networks, as these give an indication of the relative 'confidence' we should place on each network's output. The weighting factors were chosen to be proportional to the reciprocal of the training errors for each of the networks in the committee. Naturally the sum of the weighting factors was set to unity.

3.4 The Dual Waveband Scheme

There are many ways in which the ATR system described in this section could be generalised to more than one waveband of FLIR imagery. The method chosen must depend on the nature and characteristics of the FLIR used to provide the input data. The trials data available for the training and evaluation of this system was as follows

3-5 micron FLIR
This was a 256x256 pixel staring array imager with a 16x16 degree field of view. The output was digitised in-flight at the head amplifier to an accuracy of 14 bits.

8-12 micron FLIR
This produces a 1536x1024 pixel image and is based on the UK scanning SPRITE technology. The field of view was 25x16 degrees. The output was digitised at the head amplifier to 12 bit accuracy. Due to data recorder bandwidth limitations not all of the field of view can be recorded. A 1536x400 pixel region was recorded.

The two FLIR systems were flown on a large lab aircraft (Andover) mounted one above the other, with frame synchronisation between them. This leads to a difficult registration problem due to the following factors

- Different spatial resolutions
- Different optical distortions
- Imprecisely known translation and rotation between the two systems

Much work was invested in registering the images. A good registration was achieved, but not close enough for any 'image fusion' schemes (such as principal component analysis) to work. For this reason it was decided that detection would be done independently in each of the bands and the fusion would be carried out at the tracking and secondary recognition level.

For the initial study a simple data fusion scheme was chosen in which the trackfile list was common between the two bands and both waveband's recognition networks were included in the committee.

This scheme allows targets detected only in one band to be tracked and recognition attempted in both bands. Further refinements are possible, particularly in the tracking process (inter-band correlation for example), but these were not implemented in the initial study.

4. INITIAL RESULTS

The system was trained on part of the data-set gathered during flight trials. A different part of the data-set was used for performance evaluation. In order to assess the performance of the system a comparison is necessary. The following systems were also trained and evaluated for comparison

1. A system based on the architecture described in section 2 - our previous 'best' system. This used the 8-12 micron FLIR imagery as its input
2. As system 1, but using the 3-5 micron imagery
3. The single waveband version of the new architecture. This used the 8-12 micron imagery as input
4. As system 3, but using the 3-5 micron imagery
5. The dual waveband fusion system

We consider two performance metrics:

Correct Cueing Probability (CCP)
This is the total percentage of all the targets in each frame of the entire data-set which are correctly cued. It is the same as the correct cueing probability per frame. Note that to be cued the target must be declared as a target by the recognition committee.

False Alarm Rate (FAR)
This is the average number of non-targets that are cued as targets (i.e. incorrectly recognised by the committee) per frame.
In presenting the results we do not give the absolute performance values, but the relative improvements over system 1, i.e. percentage increase in FAR and CCP.

Results relative to system 1

System	CCP	FAR
1	0	0
2	12%	36%
3	63%	18%
4	58%	32%
5	75%	30%

In all cases there is a slight increase in false alarm rate. This is because the systems are all operating at different points on their Receiver Operating Characteristic (ROC) curve, which measures the trade-off between false alarm rate and correct classification probability. Generating the complete ROC for each system is a very computationally expensive task and has not been done for this preliminary investigation.

It can be deduced from the relative performance figures given that the more advanced ATR system (system 3 & 4) is better than the simple processing chain (system 1&2), and that the dual waveband fusion scheme gives a further performance gain, particularly in CCP whilst maintaining FAR.

5. FUTURE WORK

The work presented in this paper is just the initial evaluation of this algorithm. A number of further trials sorties using the large lab aircraft have yet to be analysed. In addition a number of sorties have been flown with the two FLIR sensors mounted in pods on a fast-jet (RAF Tornado). These additional sorties have also to be analysed to build up a good performance database.

In addition to the further characterisation of the existing algorithm there have been separate investigations carried out which could be included as enhancements to the programme.

Wavelet Neurons
In the current system the pixel intensities of the 9x9 patch are used directly as the inputs to the MLP. This is probably not the best representation for the input space.

Wavelets are a convenient method for representing features of an image which are compact and localised. And may give better results. This lead to the concept of the wavelet neuron.

Consider a mother wavelet function $\phi(t)$, where for convenience of presentation we work in one dimension. The children are the translated and dilated forms of this function

$$\phi^{\tau,\zeta}(t) = \frac{1}{\sqrt{\zeta}}\phi\left(\frac{t-\tau}{\zeta}\right) \quad (18)$$

By selecting a suitable subset of child wavelets the discrete wavelet transform is obtained for the projection **y** of a signal **x** onto the wavelet space. The *ith* component of **y** is given by

$$y_i = \sum_{k=1}^{N} x_k \phi^{\tau_i,\zeta_i}(k) \quad (19)$$

Now consider a typical sigmoidal activation function of an MLP neuron $\varphi(x)$. The output of a neuron y_i is dependent on the activation of the nodes in the previous layer x_k and weighted connections between the neurons and the previous layer $\omega_{k,i}$, i.e.

$$y_i = \varphi\left(\sum_{k=1}^{N} x_k \omega_{k,i}\right) \quad (20)$$

Noting the similarity between (19) and (20) we can implement the discrete wavelet transform as the first layer of an MLP where the weights connecting each wavelet node to the input layer are constrained to be discrete samples of a particular child wavelet and the activation function is the identity transformation. In fact we may ignore the weights connecting the wavelet node to the input layer and instead characterise the neuron by the values of scale and translation τ and ζ. The interconnecting weights between the input and wavelet layers are then determined solely by the parameters of the wavelet node.

It is possible to derive the back-propagation algorithm for these neurons and we have shown that they train successfully and that training is stable.

A further refinement has been developed which prevents wavelet node duplication by ensuring that all the wavelet neurons in the input layer are orthogonal. This is achieved by adding a term to the training error function which is proportional to the overall overlap of the discrete wavelets at each node

A number of experiments have been undertaken to compare the performance of standard MLPs with the performance of an MLP whose first layer consists of wavelet neurons and the results on the trials data are promising.

Weight and Node Elimination

Optimising the architecture for neural networks is a difficult problem. The crude 'brute force' technique used to determine the number of nodes in the hidden layer can certainly be improved upon and we have adopted a technique due to Williams [10]. The method uses a Laplacian prior on the weights. This manifests itself in training as a term added to the error function

$$E_W = \sum_{i,j} |\omega_{i,j}| \qquad (21)$$

A consequence of this prior is that during training, weights are forced either to adopt equal data error sensitivity or are forced to zero. This naturally leads to *skeletonisation* of the network. During this process if all of the weights associated with a neuron are forced to zero then that node is removed from the network.

Experiments with this technique used in conjunction with the wavelet neurons have also shown considerable promise.

6. SUMMARY AND CONCLUSIONS

In this paper we have described a novel architecture for an ATR system. The key features of the system are

- Dual waveband FLIR imagery is used
- A new integrated Bayesian tracking methodology is used
- A committee of neural networks is used for target discrimination
- Initial evaluation is on real data

We have also proposed a number of novel enhancements that can be employed within this system including

- Wavelet neurons
- Automatic architecture optimisation for MLP

REFERENCES

[1] Y. Bar-Shalom, T.E. Fortmann **Tracking And Data Association** Academic Press, New York, 1988

[2] P.S. Maybeck, R.L. Jensen, D.A. Hernly **An Adaptive Extended Kalman Filter For Target Image Tracking** IEEE Transactions On Aerospace And Electronic Systems AES-17, p.173-180, 1981

[3] A.P. Dempster, N.M. Laird, D.B. Rubin **Maximum Likelihood Estimation From Incomplete Data Via the EM Algorithm (with discussion)** Journal Of

The Royal Statistical Society, B 39, pp. 1-38

[4] R.E. Kalman **A New Approach to Linear Filtering and Prediction Problems** Trans. ASME, J. Basic Engineering, Vol. 82, pp. 34-45, 1960

[5] D.L. Toulson, J.F. Boyce **Data Fusion For Target Determination Using A Committee Network** *The Role of Intelligent Systems in Defence* Aeronautical Society, March 27-28 1995

[6] D.L. Toulson, J.F. Boyce **Fused Target Recognition Using A Committee Of Neural Networks** World Congress on Neural Networks, Washington, July 1995

[7] D.L. Toulson, J. F. Boyce **Moving Object Tracking Using Camera Motion Corrected Kalman Filtering**. Fourth International Conference On Image Processing And Its Applications (IAPR-92). pp. 81-85

[8] D. E. Rummelhart, G. E. Hinton, R. J. Williams **Learning Internal Representations By Error Propagation. In Parallel Distributed Processing**. Chapter 8. MIT Press 1986

[9] S. E. Fahlman. **Faster Learning Variations On Back-Propagation: An Empirical Study**. Proceedings of the 1988 Connectionist Models Summer School. pp. 38-51. Morgan Kaufmann.

[10] P. M. Williams **Bayesian Regularisation and Pruning Using a Laplace Prior** Neural Comput **5** 1993.

APPLICATION OF NEURAL NETWORK TO RECONFIGURATION OF DAMAGED AIRCRAFT

Daniel J. Collins
Department of Aeronautics and Astronautics
Naval Postgraduate School
699 Dyer Road (Rm 137)
Monterey, California 93943-5106
U.S.A.

and

Shahar Dror
Lieutenant Commander, Israeli Navy

1. INTRODUCTION

Although the title of this lecture refers to using artificial neural networks (ANN) in reconfiguration algorithms of damaged aircraft, the analysis applies in general to the identification and control of non-linear time-varying dynamical systems. This lecture addresses the problem of emulation and control of a fighter aircraft by means of ANN. Four models for describing non-linear MIMO dynamical systems are described. Based on this description a combined feedforward and recurrent neural network is structured to emulate the system. A procedure is given to emulate multiple systems in a single neural network. By the introduction of a minimal realization of the network, the complexity of the network is greatly reduced without degradation of the operating performance of the network. The aircraft model used in the demonstration of the neural network is the longitudinal node of the F/A-18A fighter aircraft. The work developed here is based on Shakar Dror [1].

The analysis may also be viewed as a neural network adaptive controller for a non-linear MIMO time varying system. A direct model reference control system is used with the neural network trained by means of the backpropagation algorithm.

2. SYSTEMS MODELING OF NONLINEAR DYNAMIC SYSTEMS

There are several different models of dynamical systems. In control theory the basic representation of a dynamic, system particularly of MIMO models, is that derived from a state space representation. In what follows we will limit our discussion to discrete state space models.

Four classes of unknown nonlinear discrete dynamical systems have be suggested by Narenda and Parthasarathy [2]. In all these models the measured output y(k) is assumed to be a linear combination of the states and possibly a feedforward control. The state at time k+1 ie., x(k+1) is derived from the state x(k) and the excitation at u(k). Both as indicated are evaluated at the previous time step.

The nomenclature adopted here uses symbols that are usually used in control theory. In addition, upper-case italic bold letters followed by brackets represent vectorial functions, subscripted with *l* for a linear function, and with *nl* for a non-linear function. Upper-case bold letters with or without succeeding parentheses, represent matrices. The state vector is denoted as *x*, the measured output vector is denoted as *y*, the input sequence is *u*, the discrete time is represented by *k*, and θ are the unknown parameters of the system.

2.1 Model I

$$\begin{cases} x(k+1) = F_l[k,\theta_f]x(k) + G_{nl}[k,\theta,u(k)];\ x(0) = x_0 \\ y(k) = C(k)\,x(k) + D(k)\,u(k) \end{cases} \tag{1}$$

In this nonlinear model, the state vector depends linearly on its prior state and nonlinearly on the input signals. The output is a linear combination of the state and input vectors. This model is very common in many mechanical systems where the inputs are subjected to nonlinearities such as saturation, backlash (dead zone), and switching (bang-bang).

2.2 Model II

$$\begin{cases} x(k+1) = F_{nl}[k,\theta_f,x(k)] + G_l[k,\theta_g]u(k);\ x(0) = x_0 \\ y(k) = C(k)x(k) + D(k)u(k) \end{cases} \tag{2}$$

Paper presented at the AGARD MSP Lecture Series on "Advances in Soft-Computing Technologies and Applications in Mission Systems", held in North York, Canada, 17-18 September 1997; Amsterdam, The Netherlands, 22-23 September 1997; Madrid, Spain, 25-26 September 1997; Ankara, Turkey, 6-7 October 1997, and published in LS-210.

Here the nonlinear function F_{nl} acts on the previous states, while the inputs are linear. This is a very popular model in systems where kinematics are involved (nonlinear trigonometric functions), systems with viscous friction, e.g., airplanes. This model or its linearized version, is very common in implementations of control actions, u, on the system F.

2.3 Model III

$$\begin{cases} x(k+1) = F_{nl}[k,\theta_f,x(k)] + G_{nl}[k,\theta_g,u(k)]; \; x(0) = x_0 \\ y(k) = C(k)x(k) + D(k)u(k) \end{cases} \quad (3)$$

In model III, both the input values and the previous state vector are subjected to two separate but additive nonlinear functions. Real-world examples for this model can be simply a combination of examples from models I and II.

2.4 Model IV

$$\begin{cases} x(k+1) = H_{nl}[k, \theta_h, x(k), u(k)]; \; x(0) = x_0 \\ y(k) = C(k)x(k) + D(k)u(k) \end{cases} \quad (4)$$

This is the most general form that represents the current state as a nonlinear function of both the previous states and inputs combined. Although it is the most general, it is the most difficult to deal with, and in practice, approaches described in models I to III are mostly used.

For every suggested model there is a corresponding neural network structure. The structure characterizes the physical behavior of the model, and is trainable by the generalize delta rule. The representations and their generalizations are described in the following section.

3. NEURAL NETWORK REPRESENTATION OF NON-LINEAR DYNAMICAL SYSTEM

3.1 System Topology

For emulation, the dynamic system has to be reconstructible in the linear case or realizable in the nonlinear case. To represent linear or nonlinear dynamical system a recurrent network must be used. The major problem is to find a training algorithm which insures convergence of the system.

It is convenient to classify dynamic in terms of the order of the input and output vectors and by the relationship of the output vector to the state vector.

Category A Input-Output Orders

a. Single input, single output (SISO)
b. Single input, multi output (SIMO)
c. Multi input, single output (MISO)
d. Multi input, multi output (MIMO)

Category B Output to State Vector Relation

a. Output vector full state representation
b. Output vector partial state representation

Full state variable output (Ba) are rather trivial cases to emulate or identify using ANN. Systems involving a single input (Aa, Ab) are also somewhat straight forward since there is a theoretical solution for the one step ahead prediction and control that is easily adapted to neural network implementation.

This lecture address the more difficult problem of finding a neural network for MIMO that is trainable by means of the back-propagation algorithm. The emulation of a dynamic system is shown in figure (1).

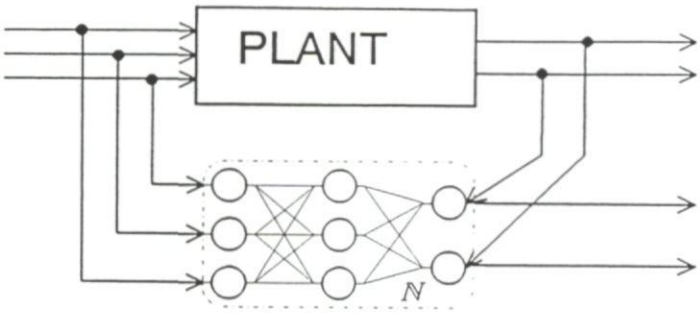


Figure 1 - A Neural Network Emulation of Dynamical System

The inputs of the plant are also presented to the neural networks. Training continues until the errors in the output reach an acceptable level.

3.2 Inner Structure

The specific inner structure of the ANN is given in figure 2.

This inner structure can emulate models I-III discussed in section 2. Three inputs and two outputs have been selected only for purposes of illustration. The circles are nodes or neurons while the solid connecting lines represent adjustable weights. The dash - double-dot line (- .. -) represents a fixed unit value weight.

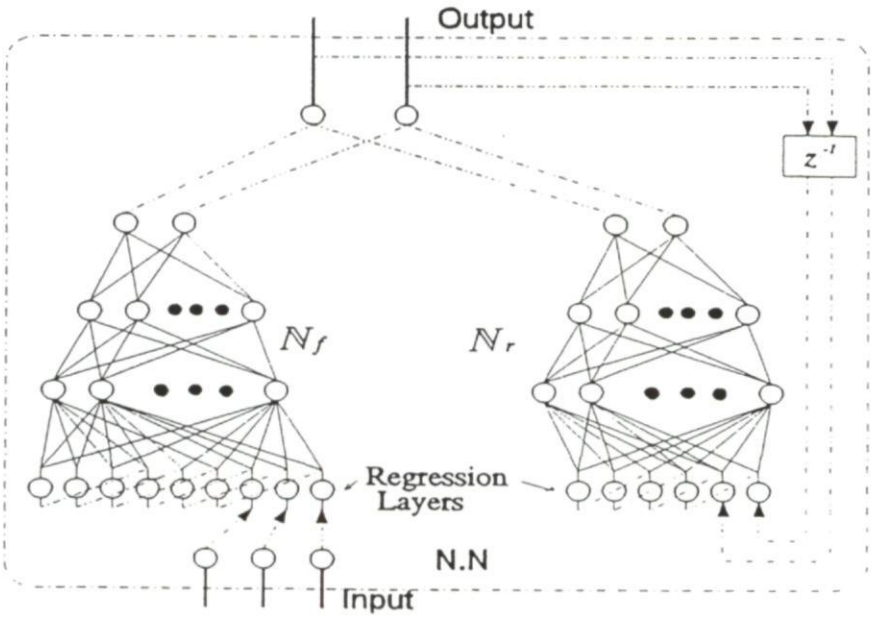


Figure 2 - Artificial Neural Network Emulator for Models I-III

In the simpler case of state variable feedback, it is possible to write the model III equation as

$$y(k+1) = \tilde{F}_{nl}[k, \theta_f, y(k)] + \tilde{G}_{nl}[k, \theta_g, u(k)] \quad (5)$$

Where θ_f and θ_g represent unknown parameters. It can be seen that the neural network structure is then

$$y(k+1) = N_r[y(k)] + N_f[u(k)] \quad (6)$$

Where N_r is the recurrent network shown the the right side of figure 2 and N_f is the feedforward network on the left side of figure 2.

The network N_r is an extended form of the Hopfield recurrent net. The Hopfield net normally has a fully connected output layer to the input layer by a one-step delay line. The z^{-1} denotes a one step delay.

In the case of output feedback, it is more difficult to determine N_r. The condition of reconstructibility insures however that present output values can be predicted from past values of the output. The need for previous values leads to the introduction of regression layers.

The order of the regression layer is determined by the system that needs to be emulated. In the illustration given in figure 2 shows two regression layers of order three. On the left side the input values are regressed and on the right side the output values are regressed. Although the regression layers are inner layers of the neural network for all practical purposes they function as the input layers to the neural network.

The neural network models the physical system nicely in that N_r models the $\tilde{F}$ function and N_f models the $\tilde{G}$ function in the equation 5. This separation of functions gives more insight to the system when analyzing the trained network and the weight matrices.

3.3 Trainability Aspects

A series - parallel scheme has been used in the training of the network. While training one has a desired output denoted by y^d, the procedure is to use this desirable output in the regression layers so that one has

$$y(k+1) = N[y^d(k), u(k)] \quad (7)$$

Essential one has a simple feedforward network. This method has been shown to be stable.

Since the input and output values are assume to be bounded (BIBO) and there are no non linearities in the feedback loop, the standard back propagation algorithm can be used. After training or in the recall phase, the outputs are regressed i.e.

$$y(k+1) = N[y(k), u(k)] \quad (8)$$

4. ANN FOR LINEAR TIME-INVARIANT SYSTEM

Consider the set of equations given by

$$x(k+1) = Fx(k) + Gu(k)$$
$$y(k) = Cx(k) \qquad (9)$$
$$x(o) = x_o$$

These are the same set of equations which we will later use to describe the F/A18A. In this particular case we assume that the matrix C has full rank and is invertible. This specific system can be represented by Figure 3 where the output layer is fully connected to itself.

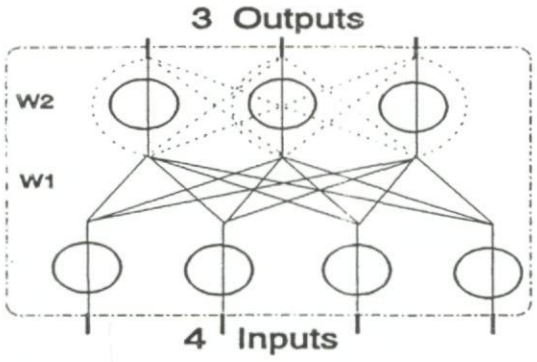


Figure 3 - A Neural Network Representation of LTI System

Normally the weights to which the network converges are unknown in advance but in this particular case one can write

$$y(k+1) = CFC^{-1}y(k) + CGu(k)$$
$$y(o) = Cx(o) \qquad (10)$$

The weights shown in the figure as w1 and w2 converge as expected to w1 = CG and w2 = CFC^{-1} with a residual error approximating the numerical precision of the machine performing the calculations. Thus in this particular case one can write down the neural network weights directly without going through the training process.

5. ANN IN ADAPTIVE CONTROL

The general idea in adaptive control is to find a regulator that will compensate for uncertainties in the controlled plant to form an overall stable system. As described by Astron and Wittenwork [3], there are five basic approaches to adaptive control. We shall limit our discussion to model-reference adaptive systems.

Model -reference adaptive controllers (also called model following) involve methods to adjust the parameters of the controller so that the overall closed loop system will behave closely to a prescribed system. Model-reference systems can further be divided into direct and indirect adaptive controllers, shown in Figure 4.

In direct control the parameters of the controllers are changed directly based on some measure of the output error as shown in Figure 4

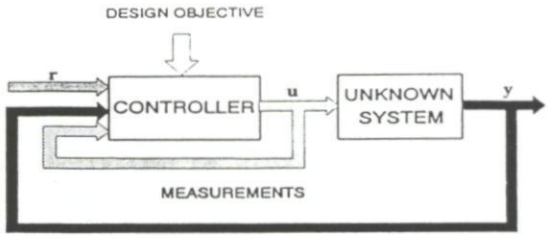


Figure 4a - Direct Control

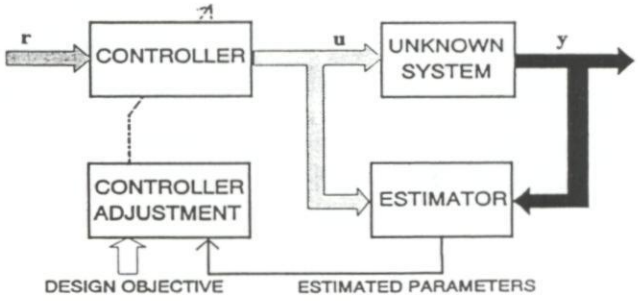


Figure 4b - Indirect Control

In indirect control the parameters of the unknown system are estimated and then used to update the parameters of the controller. Both systems direct and indirect result in a nonlinear controller even for a LTI system. The method used in this investigation was that of direct control.

Model-reference adaptive control is based on a reference model that specifies the behavior of the overall system. The controller parameters are updated based on the error between the outputs of the reference model, y^d, and the plant, y. Figure 5. shows the general architecture of model-reference adaptive controller, where, r, is the command signal fed into the reference model and the controller, and u, is the control actions obtained from the controller and excite the plant.

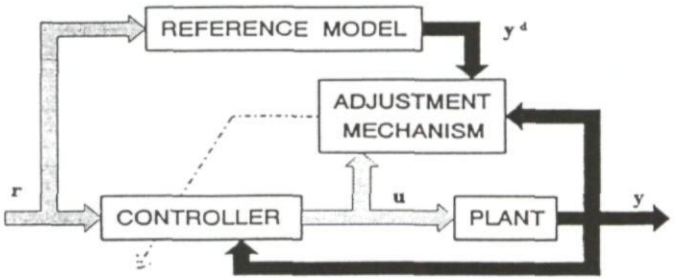


Figure 5 - Model-Reference Adaptive Control Architecture

Two loops are involved. The inner loop specifies the feedback controller, whereas the outer loop adapts the controller's parameters. From the beginning of the training the control actions are in the desired trend to bring the plant to follow the reference model. The procedure is directed to work on the desired domain which is the output of the plant, despite the fact that the controller does not have direct access to plant output, which is usually needed for training artificial neural networks. Since it is continuously learning to supply control actions in the desired direction, it can be used for varying systems. The problem is to ensure stable learning procedure over the possible changes in the system, which is not guaranteed.

Direct model-reference adaptive control overcomes the need to first emulate the plant by utilizing the back-propagation through the plant. A schematic of the direct model-reference controller is given in Figure 6. The neural network controller is trained based on an error, transformed backwards through the plan, and effects the plant output directly. This method was used by Psaltis, Sidris and Yamamura for coordinate transformation [4], and by Saerens and Soquet for different constant dynamical systems [5]. Ha, Wei, and Bessolo [6] supposedly used the method for control of a linearized longitudinal approximation of the F16 aircraft.

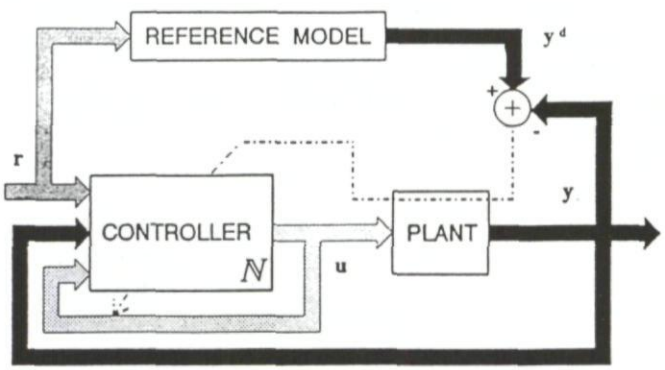


Figure 6 - Direct Model-Reference Adaptive Control

The direct model-reference adaptive control with the extensions described in this section is the method used in this research. The specific implementation of the control scheme is presented in the coming section.

The dynamical systems are assumed to be output-controlled under all the changed conditions for the purpose of this design. The adaptive control scheme is of the model-reference type. The reference model represents the desired response of the overall controlled dynamical system. The reference model is usually described by a linear dynamical system from $r \in R^l$ to $y \in R^n$, simply because the properties of linear systems are well defined. It is assumed that the reference model is chosen such that for a given bounded command signal, r, there exists a control action, $u \in U$, exciting the plant, such that the output y follows y^d. The control actions spanned by $U \in R^m$ and are bounded in accordance and as required by the plant. This in turn makes the controller be a bounded system by itself. Under the specified control actions, $u \in U$, the neural network controller treats the "non-exact" tracking., by reaching the best controller in a least-mean-square-sense. The changing plant is also assumed to remain within the same class of dynamical systems and retain the same inputs and outputs, i.e., l, m and n are constant for the problem.

The ANN based controller design is adopted from classical control. A feedback controller $N_{fk} \in R^n \rightarrow R^m$ and a forward shaping filter $N_{fd} \in R^l \rightarrow R^m$, as shown in Figure 7 schematically. The two parts of the controller can be combined into one network which accomplishes both functions, having as inputs the command signal r, and the plant output y, resulting in the control action u. The dash-dot (-.-) line in Figure 7 indicates the combined controller, denoted N_c. The training of the network is done as decribed in Figure 6 for the model reference approach.

The controller in the model-reference scheme is continuously adjusting to change the parameters in the network to drive the plant to follow the reference output. As mentioned earlier, it is desirable to shorten the transition time of the adaption between changes of the plant.

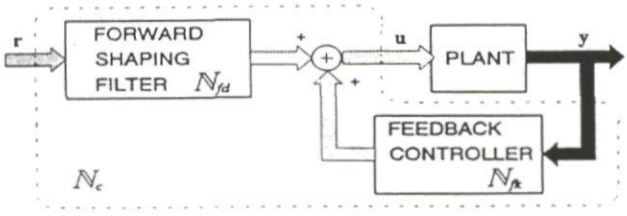


Figure 7 - Control Architecture

Training can be accomplished by defining a finite set of probable changes, each with its corresponding reference model and a resultant controller. Each model is trained off-line. As discussed in the multiple-systems emulator, a single network is used to incorporate all of the controllers creating a unified controller. Inputs to the controller network are used to relate the controller to the proper plant or new plant. For linear MIMO systems described in a state-space observer canonical form, controllability implies identifiability. A sensible choice of physical information to describe the plant, even in the non-linear case, is past measurements of input and output of the plant , denote by type information in figure 8.

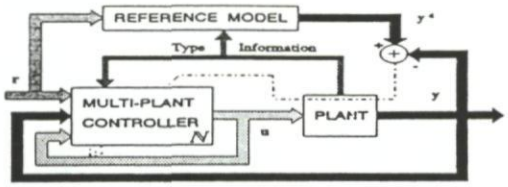


Figure 8 - Adaptive Control with Type Information

Having the information about the current plant in physical terms enables the network to generalize a suitable controller for the cases where the real plant differ somewhat from the plant the network was trained on. Consequently the network is valid for a variety of changes with a finite number of pre-trained conditions, unless the given change is in some sense linearly independent of the models used to train the network. The structure of the unified controller is herein presented.

6. ANN STRUCTURE FOR THE UNIFIED CONTROLLER

The unified controller can be introduced in a network very similar in structure to the multi-system emulaltion network. Figure 8 shows the general block diagram of the training. One can see that the information on which a plant is currently controlled is passed to the reference as well, enabling a possibly different reference model for every plant. The network Nc, not only trained as a simple controller, but also makes a synthesis of type information to generate proper control actions to the corresponding plant.

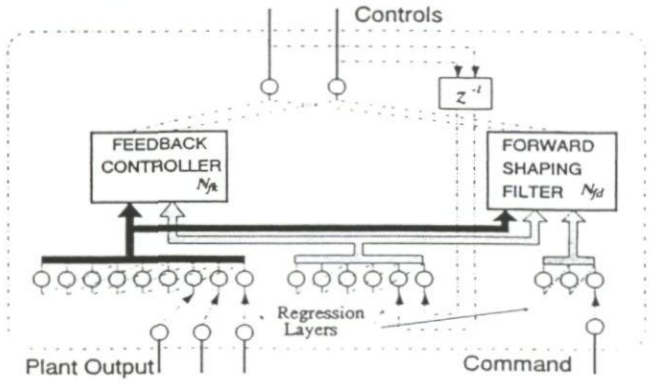


Figure 9 - General Structure of ANN Unified Controller

In the case of dynamical systems, the information about the type of the plant happens to be the same as needed for creating the controller itself. Previous measurements of the commands, output and controls are provided inside the network by using regression layers. Figure 9 depicts the classical controller as appears in Figure 7 with additional type information combined into a single network as in Figure 8.

The example given in Figure 9 shows a controller network with one command input, two control actions as an input to a plant with three outputs, all of which are fed into regression layers of order three. Each of the sub-networks, N_{fd} and N_{fk} in Figure 9, may have its own dynamics based on the inputs to that block. The blocks of the feedback controller and the forward shaping filter incorporates networks of the same form that appear in Figure 2 . The type of information, i.e., the regressed value of the inputs and outputs of the plant are supplied to both sub-networks. The z^{-1} block represent a delay of one time unit, and is inherent in the structure of the network.

7. ANN EMULATION AND CONTROL DEMONSTRATION

This section presents a test case to examine implementation of emulation and control using artificial neural networks as developed in the previous chapters.

The F/A-18A fighter aircraft was chosen as a demonstration MIMO dynamical system due to the availability of multiple control surfaces leading to a rather complex dynamics, see Figure 10 below. A damaged aircraft was assumed to represent a change in the system, which the neural network emulates and controls. Thus, the neural network

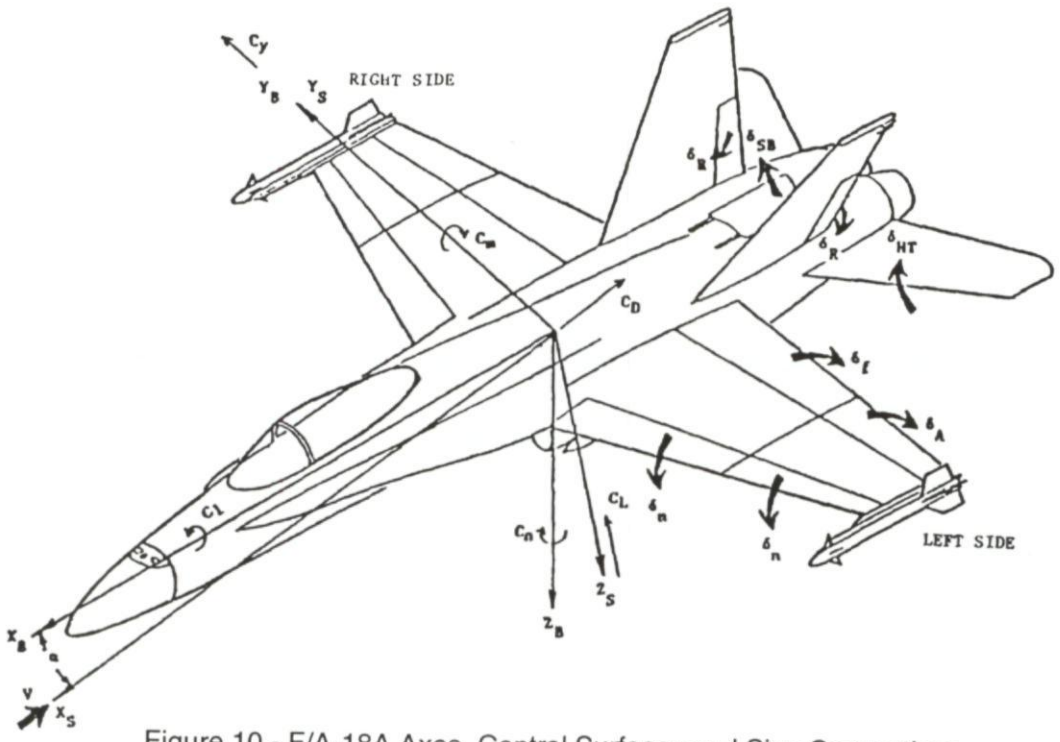


Figure 10 - F/A-18A Axes, Control Surfaces and Sign Conventions

forms a reconfigurable adaptive controller for an impaired aircraft. The closed-loop performance is dictated by a reference model that meets the conditions required by military specification MIL-F-8785C. The discussion continues with a description of the open-loop dynamic model of the F/A-18 followed by specific damage mechanism which were treated in this research.

7.1 F/A-18A Longitudinal Dynamical Model

The F/A-18A linearized longitudinal dynamics with asymmetrically-saturated control inputs, provides a 25th order dynamical system with six control inputs and three output variables. The dynamical model includes the airframe dynamics as well as actuator and senor dynamics. The continuous state-space model was developed from data in [7] for a flight condition of Mach 0.6 and altitude 10,000 feet, in escort configuration. The aircraft equations of motion are taken from [8] in stability axes and trim conditions, assuming no gusts. The actuator and sensor models are based on the model developed by Rojek [9].

Figure 10 shows the F/A-18 with the control surfaces and describes the sign conventions for both deflections and aerodynamic coefficients. The terms in body axes are denoted $(*)_B$ and $(*)_S$ denote stability axes. The control surfaces affecting the longitudinal dynamics are the stabilators (dst), leading edge flaps (dle) and trailing edge flaps (dte).

The continuous state-space representation of the aircraft, which includes the sixteenth order actuator dynamics, fourth order airframe dynamics, and fifth order sensor dynamics is given in the 25th order augmented system as follows

$$\begin{aligned} \dot{x} &= Fx + Gu \\ y &= Hx + Du \end{aligned} \tag{11}$$

The control vector, u, is the actuator deflection commands [deg],

$$u = (\delta_{st_r} \;\; \delta_{st_l} \;\; \delta_{le_r} \;\; \delta_{le_l} \;\; \delta_{te_r} \;\; \delta_{te_l})^T \tag{12}$$

where st, le and te denote the stabilators, leading edge and trailing edge, and the subscripts r, l refer to the right and left sides respectively.

The measured output y is given by

$$y = (\; q \;\; n_z \;\; \alpha \;)^T \tag{13}$$

where q is the pitch rate [deg/sec], n_z the normal acceleration. [g's] and α is the angle of attack [deg].

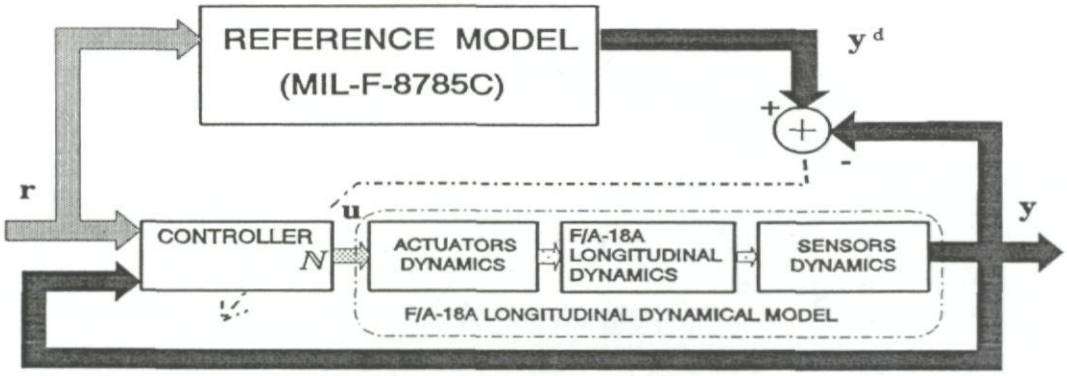


Figure 11 - Model-Reference Control Architecture for the F/A-18A

7.2 F/A -18A Damaged Longitudinal Dynamic Model

The damage mechanism considered in this example is a malfunction in the actuation of the left stabilator. The left stabilator does not respond to actuator commands and remains fixed at the trim condition.

7.3 Control Refererence Model

The reference model for the model-following controller is based on the military specification for flying qualities of piloted airplanes, MIL-F-8785C. The general configurations is shown in Figure 11 The same reference model is selected for both damaged and undamaged F/A-18 models.

The reference is a forth order linear model, that defines desirable behavior for the short period and phugoid modes. The frequency and damping are

- Short period

$$\omega_{sp} = 4.3 \text{ [rad/sec]}; \quad \zeta_{sp} = 0.6$$

- Phugoid

$$\omega_{ph} = 0.02 \text{ [rad/sec]}; \quad \zeta_{ph} = 0.4$$

The reference model is given in discrete state-space form as

$$\begin{cases} z(k+1) = Fz(k) + Gr(k) \\ y^d(k) = Hz(k) + Dr(k) \end{cases} \tag{14}$$

The input command, r, is the pilot pitch stick commands p_x. The output vector, y^d, consists of,

$$y^d = \left\{ q^d \; n_z^d \; \alpha^d \right\}^T \tag{15}$$

here q^d is the reference pitch rate [deg/sec], the desired normal acceleration n_z^d = [g's] and α^d is the reference angle of attack [deg].

8. SIMULATION RESULTS AND ANALYSIS

The control results include two parts. Initially a model follower controller is designed for each of the undamaged and damaged aircraft separately. Then the two controllers are combined into a single network. The unified network is an adaptive controller. Based on the input information, the network generates the required control actions such that the plant output will behave similar to the reference model. The input information includes the pilot command p_x, the plant output feedback, and prior measurements of plant input (network output) and output as indicators of plant type, of either undamaged or damaged aircraft.

8.1 Model-Reference Controller - Frequency Domain Results and Discussion

The spectral response of the pitch rate q, the normal acceleration n_z and the angle of attack AoA of the undamaged (plant #1) and damaged (plant #2) aircraft due to pilot commands p_x is given in Figure 12.

This data is obtained from two separately trained neural networks for the damaged and undamaged aircraft.

Each graph shows the analytic frequency response - the solid line, which is the Bode plot of the reference system outputs due to the input, together with the desired values and network output values, both analyzed using FFT, while exciting the input

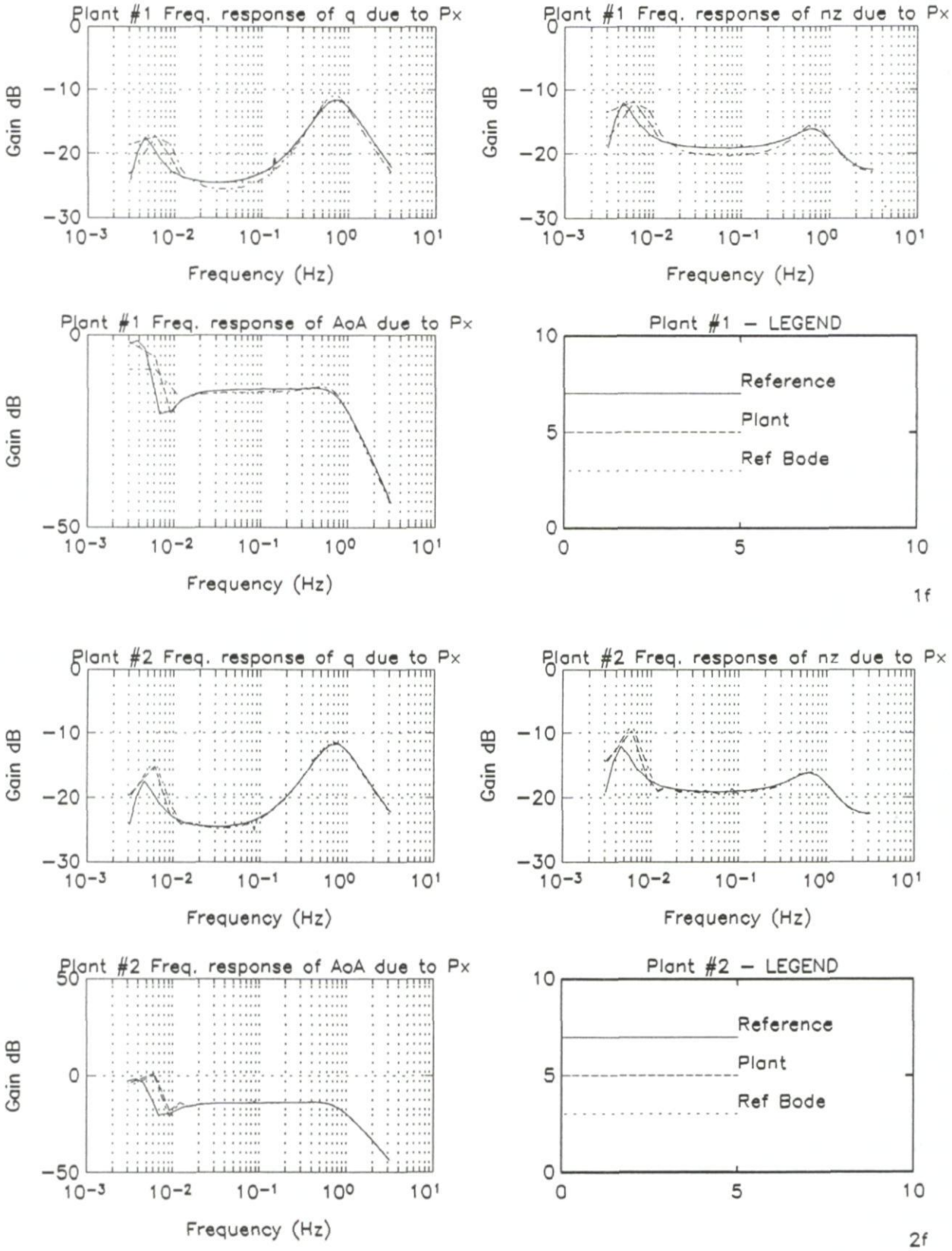


Figure 12 - Spectral Response of Undamaged and Damaged Controlled Aircraft Due to Pilot Longitudinal Stick Command, Single Controller

(pilot) command with random binary sequence (RBS). The desired value shown by the dashed lines are calculated using the reference model, and the dash-dot line is the neural network output.

It was desired to have a single neural network (unified controller) which would control both the damaged (Plant #2) and undamaged aircraft (Plant #1) simultaneously. To accomplish this the neural network was randomly trained on examplars of the Plant #1 and Plant #2. This included type information which identified the plant being considered.

Figure 13 presents the same information as in Figure 12 using the unified controller, which is capable of controlling both the undamaged and damaged plants simultaneously. The normalization of the models to range of ± 1 cause a shift of about - 15dB from the zero dB line.

The observations from the spectral results in Figures 12 and 13 are the following.

a. Each plant is controlled such that the overall system behave very close to the reference model over the entire spectral range of interest. The difference at the low frequencies is a problem of persistent excitation. For piloted aircraft control usually the indicated difference at the low end does not present a control problem. The human pilot compensate for the low frequency dynamics with no accessional control effort.

b. The spectral behavior of the unified controller in Figure 13 is similar for both models, with good agreement to the analytical reference, although less accurate than the single controller shown in Figure 12. The difference between the unified and single controllers is due to the greater complexity of the unified controller.

8.2 Model-Reference Controller - Time Domain Results and Discussion

The performance of the unified controller in the time domain is tested by presenting a full stick RBS input as the pilot command.

For the first 10 seconds Plant #1 or the undamaged plant is simulated. At 10 seconds damage is considered to have occurred. The neural network adaptive controller (unified controller) detects the change and initiates the correct control action for Plant #2, the damaged plant.

The response of the network compared to the reference model is presented in Figure 14 with the corresponding actuators commands in Figure 15.

Each graph shows the desired and network output for the command. The error between the two is presented in the lower plot of each pair. Note that the output errors are at the order of 10^{-2} or less. The relative error of all input-output relations is about 1-5%.

9. Further Analysis of Damaged Aircraft

It might be thought that the present method is limited to the specific damage that has been modelled in the training. In a further study Brunger [11] investigated a damaged A4D. The damage was assumed to be a reduction in the control effectiveness $M_{\delta\epsilon}$ by 70%. The neural network of the same structure as that developed by Dror [1] in this lecturer was trained on the undamaged and the damaged aircraft. Similar results to those just discussed in the frequency and time domain were obtained for the damaged and undamaged aircraft.

In addition it was found that the unified neural networks controller could detect and control damage intermediate between the damage and undamaged aircraft. Figure 16 shows the response of the neural network to an aircraft with damage caused by a 30% reduction in $M_{\delta\epsilon}$ rather than the 70% reduction on which the network was trained.

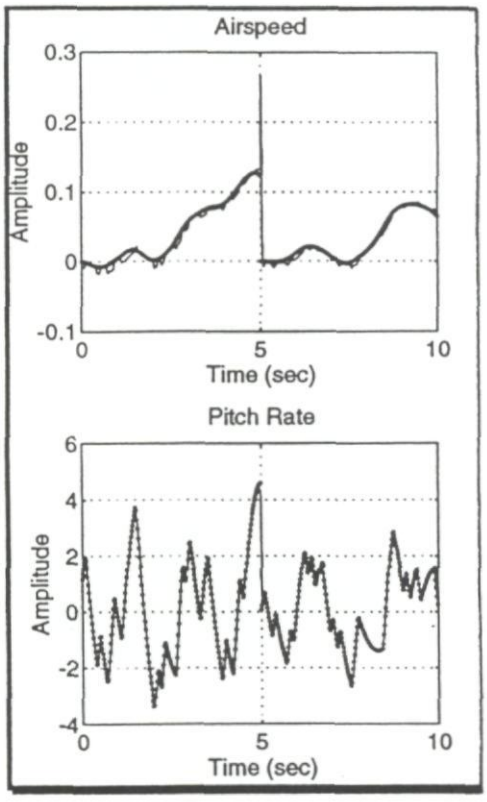


Figure 16a - Airspeed Pitch Rate

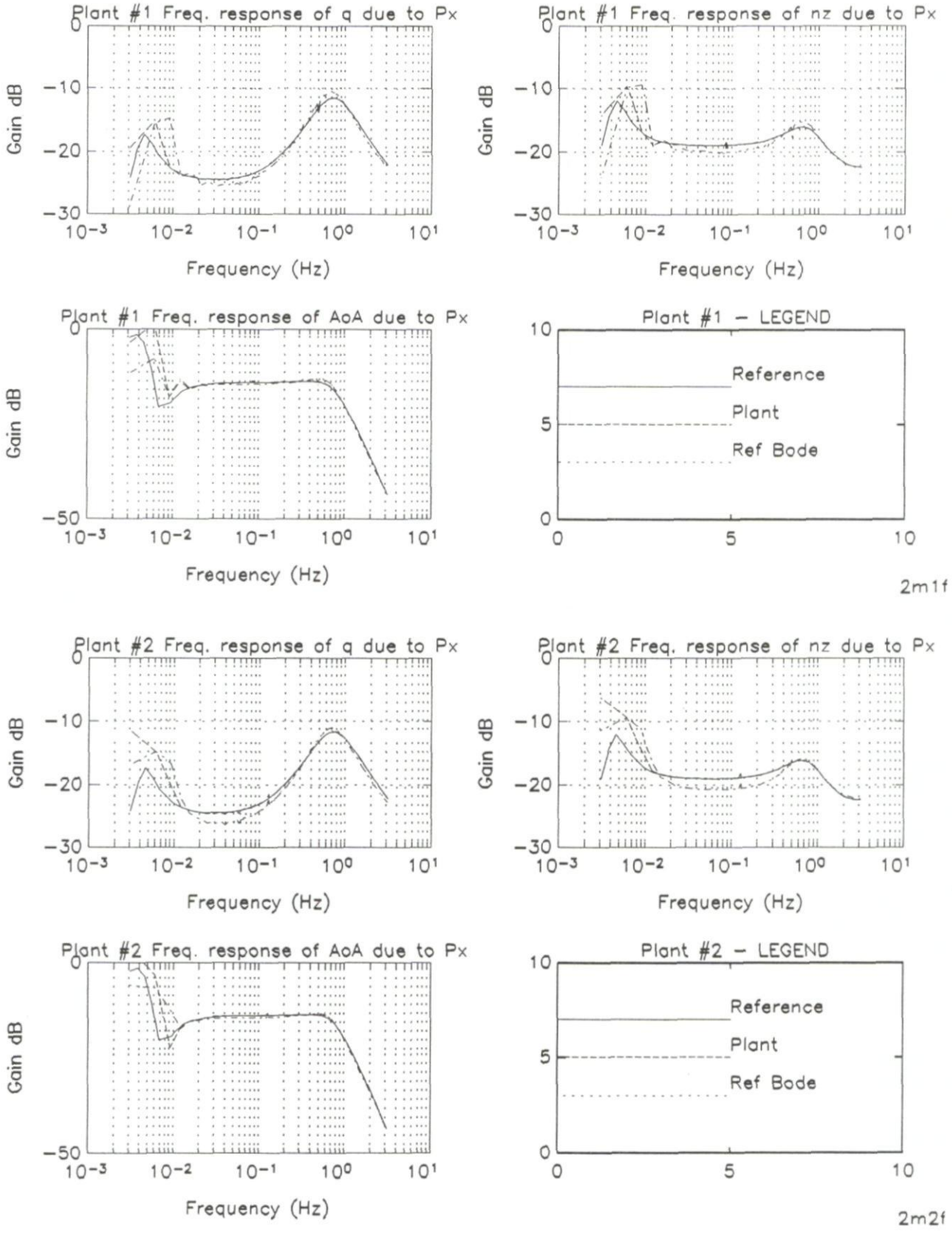


Figure 13 - Spectral Response of Undamaged and Damaged Controlled Aircraft Due to Pilot Longitudinal Stick Command, Unified Controller

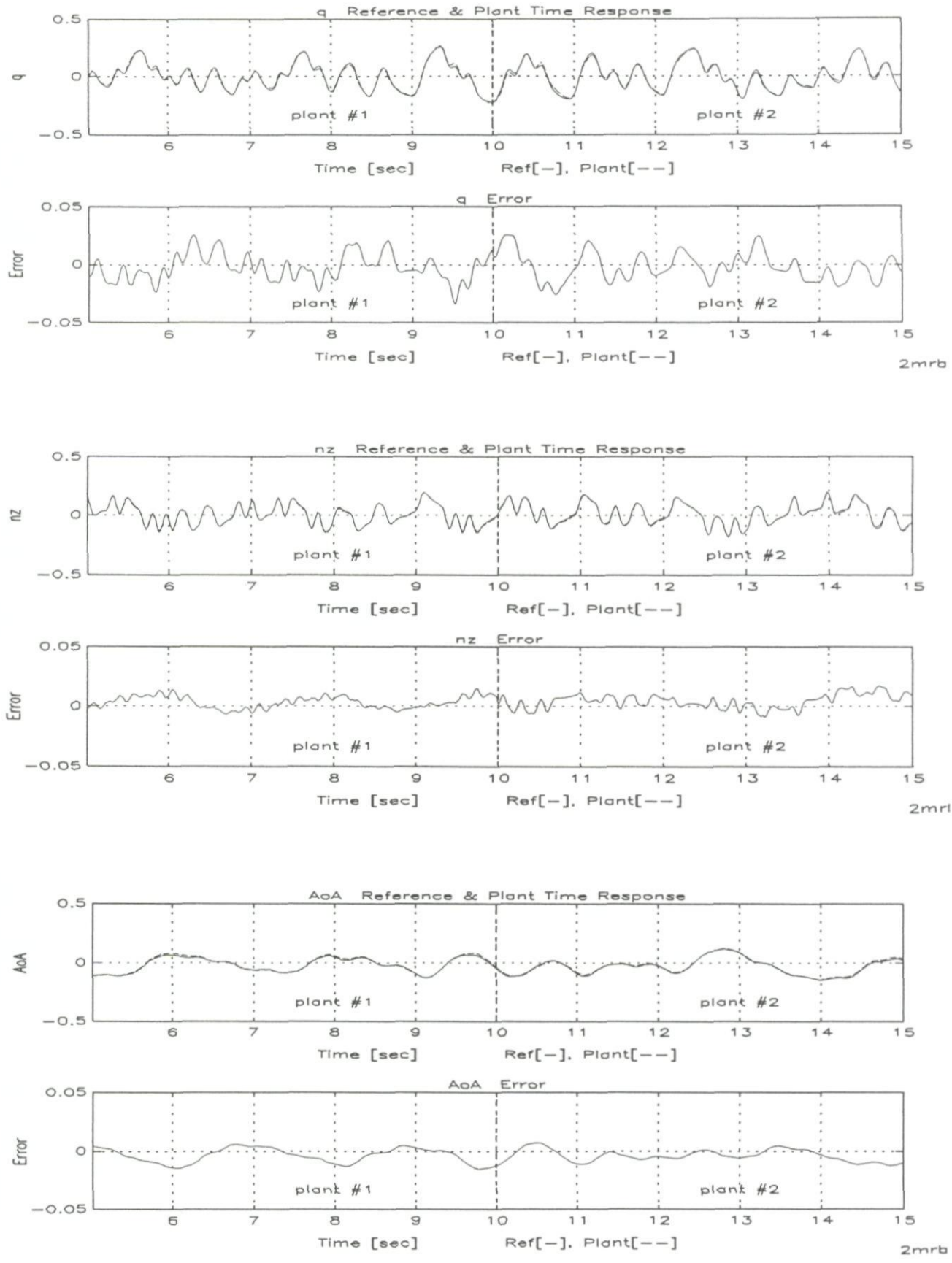


Figure 14 - Time response of Controlled Time Varying Aircraft Due to Pilot RBS Stick Excitation, Unified Controller

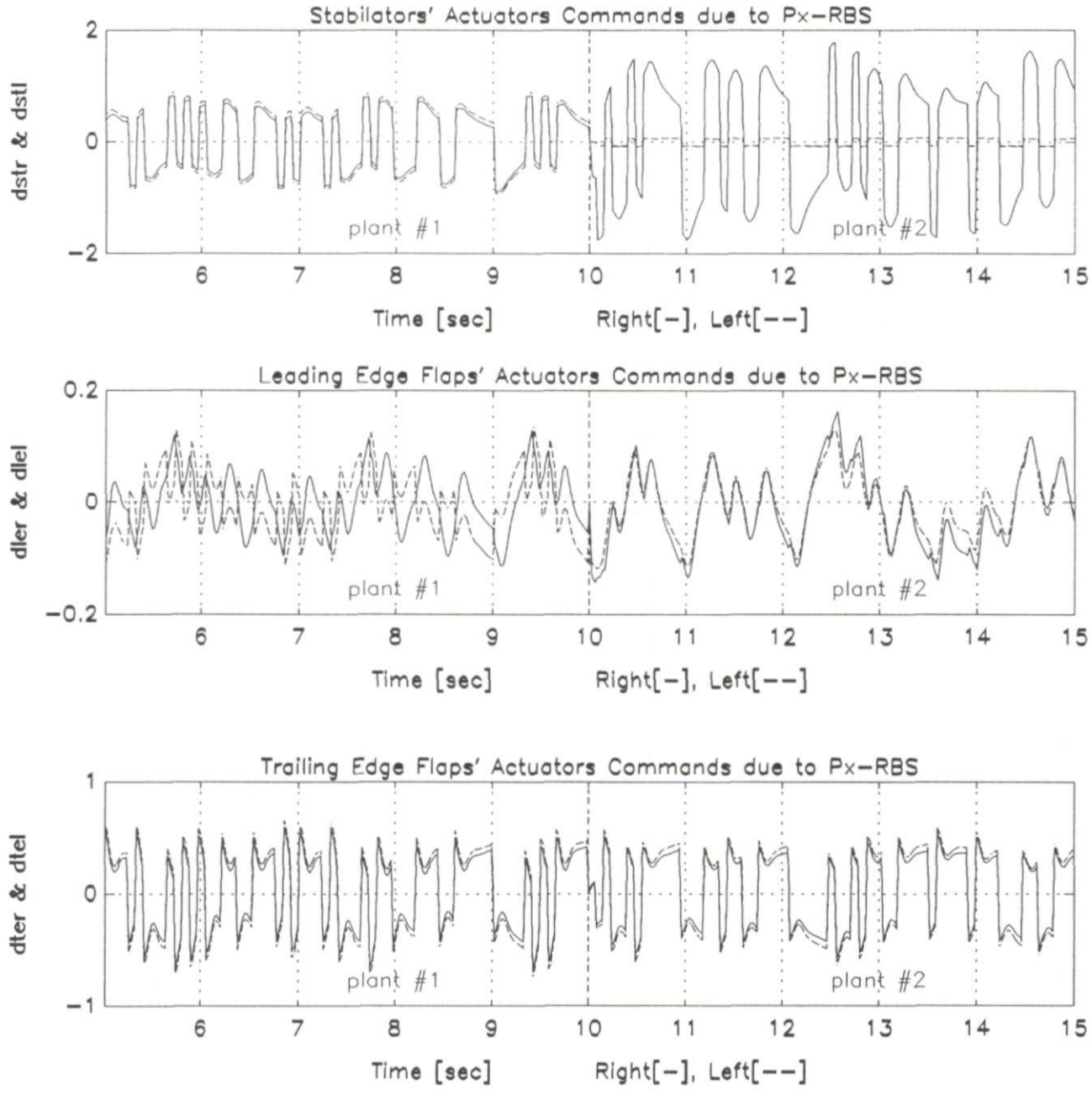


Figure 15 - Actuation Commands of Controlled Time Varying Aircraft Due to Pilot RBS Stick Excitation, Unified Controller

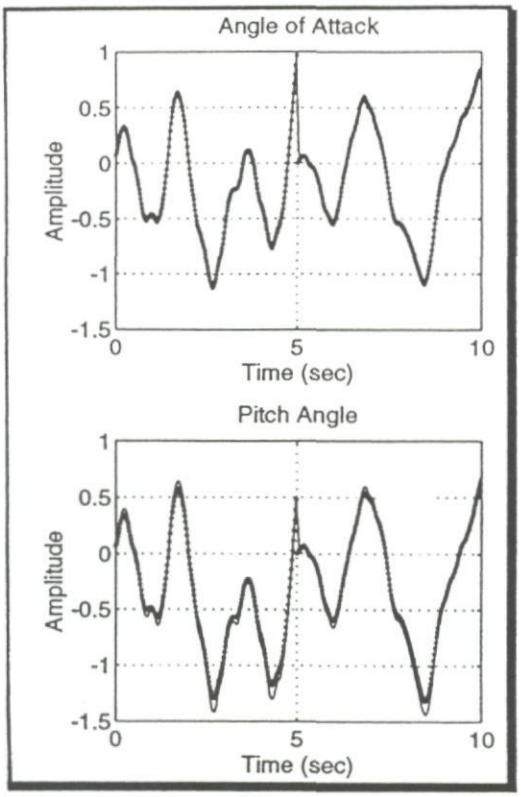


Figure 16b - Angle of Attack Pitch Angle

The damage occurs at 5 seconds. It can be seen that the neural network detects and controlls the 30% damaged aircraft properly. This was found true for control effectiveness changes from 0% to 90% in $M_{\delta e}$.

10. SUMMARY

This work considered dynamical systems in association with artificial neural networks. The basic theory of adaptive controllers based on neural network was presented. Methods for emulation and control of dynamical systems using neural networks were developed. The important results are summarized here

10.1 Main Results

This research established a general approach for identification and emulation of non-linear MIMO time-varying dynamical system. Four general MIMO dynamical model are presented. Based on the selected model a corresponding network structure was introduced. The structure associated a combination of feedforward and recurrent artificial neural networks. A serial-parallel training scheme was used to train the network. A single network was shown to be capable of representing multiple dynamical systems from the same class, under the balanced random training procedure. The model switching in the procedure did not affect the accuracy of the network.

Further in the research a neural network was trained to control a MIMO dynamical system using a back-propagation approach through the plant training procedure, which is a variation of the basic back-propagation algorithm. The network was trained in a model-follower control architecture without a direct access to the desired values and without the need for specific knowledge on the controlled system. A single network controller was trained to represent multiple controllers, thus forming an adaptive controller. No estimation process of the plant was involved in the procedure.

Artificial neural networks were found to be a powerful tool for system identification and in designing adaptive controllers.

REFERENCES

1. Dror, S.
Identification and Control of Non-Linear Time-Varying Dynamical Systems Using Artificial Neural Networks, Naval Postgraduate School Ph.D., September 1992.
2. Narenda, K.S. and Parthasarathy, K.
Identification and Control of Dynamical Systems Using Neural Networks, IEEE Tran. Neural Networks, Vol. 1, No. 1, pp. 4-27, March 1990.
3. Aström, K. J. and Wittenmark, B
Adaptive Control, Addison-Wesley, 1989.
4. Psattis, D., Sidris A., and Yamamura A. A.
A Multilayered Neural Network Controller, IEEE Control Systems Magazine, pp. 17-21, April 1988.
5. Saerens, M., Soquet, A.
A Neural Controller Based on Back Propagation Algorithm, Proc of First IEE Inter Conf. on ANN, pp. 211-215, London 1989.
6. Ha, C. M., Wei, Y. P. and Bessolo J.A.
Reconfigurable Aircraft Flight Control System via Neural Networks, AIAA 92-1075, 1992.
7. McDonell Douglas Corporation.
Report No. MDC A 7813. F/A-18A, Flight Control System Design Report Vol I & II, System Description and Theory of Operation, December 1982.
8. McRuer, D., Ashkenas I., Graham D.
Aircraft Dynamics and Automatic Control, Princeton, New Jersey, 1990.
9. Rojek, F. W.
Development of a Mathematical Model that Simulates the Longitudinal and Later-Directional Response of the F/A-18 for the Study of Flight Control Reconfiguration, MS degree, Naval Postgraduate School, Monterey, California, 1986.

10. Military Specifications, Flying Qualities of Piloted Airplanes, MIL-F-8785C, 5 November 1980.
11. Brunger, C. A. Artificial Neural Network Modeling of Damaged Aircraft, M.S. degree, Naval Postgraduate School, Monterey, California 1994.

Towards Autonomous Unmanned Systems

U. Krogmann
Bodenseewerk Gerätetechnik GmbH
Postfach 10 11 55
D-88641 Überlingen

SUMMARY

The development, procurement and utilization of defense systems will in future be strongly influenced by affordability. A considerable potential for cost reduction is seen in the extended use of unmanned systems. This paper will describe important enabling techniques and technologies as a prerequisite for the implementation of future autonomous systems with goal- and behavior-oriented features. Main emphasis is being placed on information technology with its soft-computing techniques. The treatment of conceptional system approaches will be followed by design considerations and then a global methodology for the engineering of future autonomous systems will be dealt with. Critical experiments for technology evaluation and validation will be mentioned together with a brief description of the main focus in future research.

1 INTRODUCTION

Tactical systems are implemented as Integrated Mission Systems (IMS) such as e.g. air and space defense systems. As shown in Fig. 1, the key elements of IMS are - among others - platforms with sensors and effectors, ground based components with communication, command and control etc.

In technology, evolutionary progress is generally determined by the interaction between the "Requirements Pull (RP)" and the "Technology Push (TP)" (Fig. 2). Ever increasing requirements for more and more complex systems and their functions activate individual key technologies within the technological basis available or possibly to be created. However, new technologies - such as currently the new Information Technology (IT) and Micro Technology (MT) - exert pressure towards increased requirements for new systems. This is connected with the objective of implementing unprecedented system characteristics, thus gaining a technical and operational lead, and this, if possible, at reduced life-cycle costs.

In the future progress primarily will be driven by economic aspects rather than by technological advances alone. Within this context "affordability" is of decisive importance. Autonomous unmanned tactical systems surely are a viable step to cope with the cost reduction challenge and to improve cost effectiveness in the future.

It is impossible to treat all important related techniques and technologies within the scope of this paper. The key notion „autonomy“ is intimately connected with advances in Information Technology. Therefore emphasis is placed on this aspect. The implementation of all needed functions into an Autonomous System is also a great challenge in terms of miniaturization and integration techniques. However, this can only be treated here very briefly. The objective of this paper is to present computational and machine intelligence technologies and concepts enabling the realization of future autonomous systems. This part of IT is the catalyst for the transition to new technologies in almost all areas of mission systems.

Paper presented at the AGARD MSP Lecture Series on "Advances in Soft-Computing Technologies and Applications in Mission Systems", held in North York, Canada, 17-18 September 1997; Amsterdam, The Netherlands, 22-23 September 1997; Madrid, Spain, 25-26 September 1997; Ankara, Turkey, 6-7 October 1997, and published in LS-210.

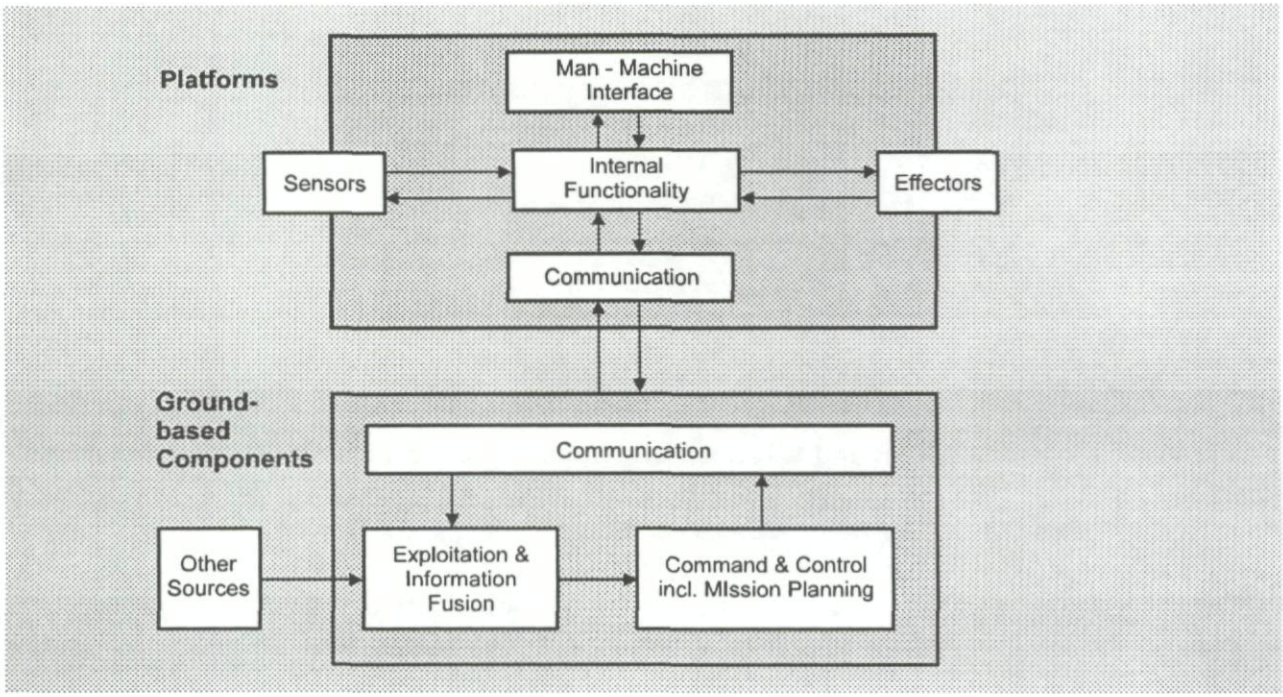


Figure 1: Key elements of integrated mission systems

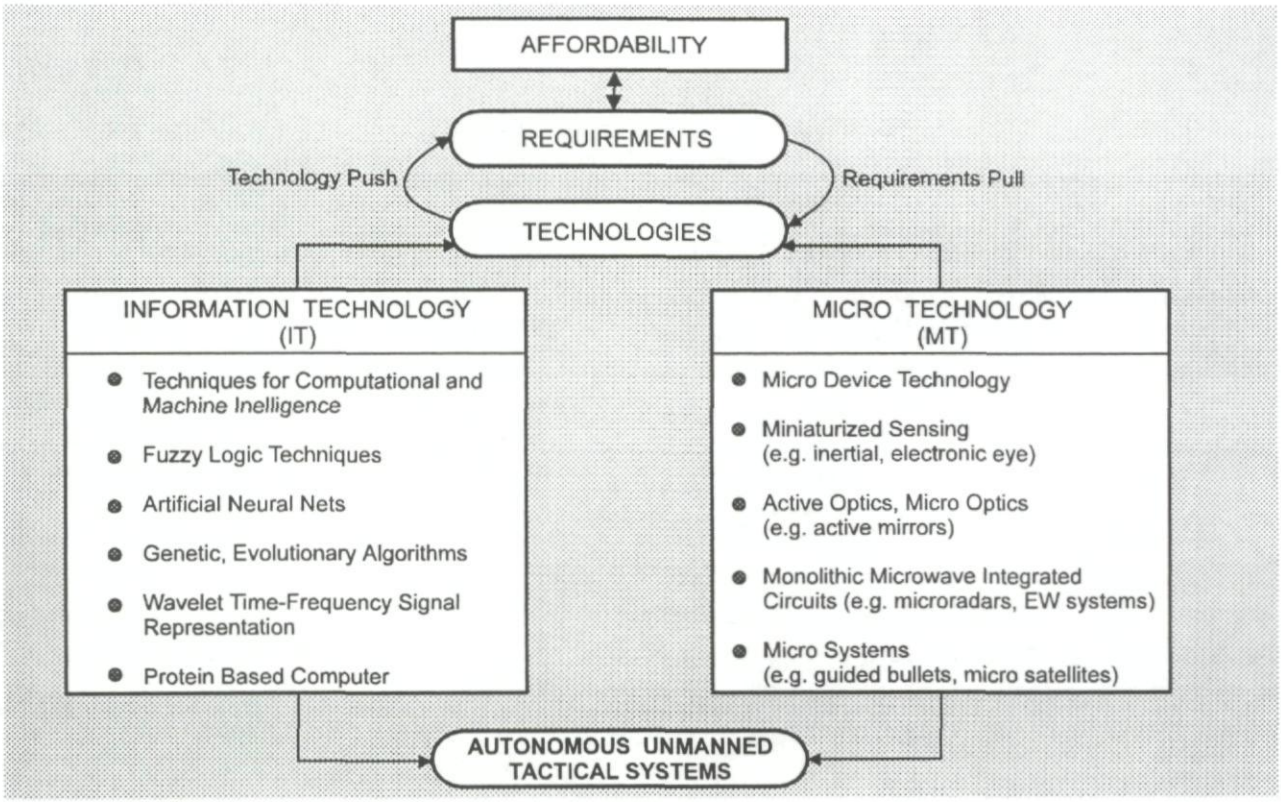


Figure 2: Key enabling technologies

2 AUTONOMOUS SYSTEMS

As specified in [1], autonomy is the ability to function as an independent unit or element over an extended period of time, performing a variety of actions necessary to achieve predesignated objectives while responding to stimuli produced by integrally contained sensors. The following characteristics are therefore typical of an autonomous, behavior-oriented system:

- An "environment" (real world) is allocated to the system
- There is an interaction between the system and the environment via input and output information and possibly output actions
- The interactions of the system are concentrated on performing tasks within the environment according to a goal-directed behavior, with the system adapting to changes of the environment.

The interaction of the systems with the surrounding world (Fig. 3) can be decomposed into the following elements of a recognize-act-cycle (or stimulus-response-cycle).

- Recognize the actual state of the world and compare it with the desired state (which corresponds to the goal of the interaction). (MONITORING)
- Analyse the deviations of actual and desired state. (DIAGNOSIS)
- Think about actions to modify the state of the world (PLAN GENERATION)
- Decide the necessary actions to reach the desired state (PLAN SELECTION)
- Take the necessary actions to change the state of the world (PLAN EXECUTION)

To perform these functions, first of-all appropriate sensor and effector systems must be provided, as mentioned earlier. In the case of unmanned autonomous systems information processing means must be incorporated that apply machine intelligence to perform the tasks mentioned.

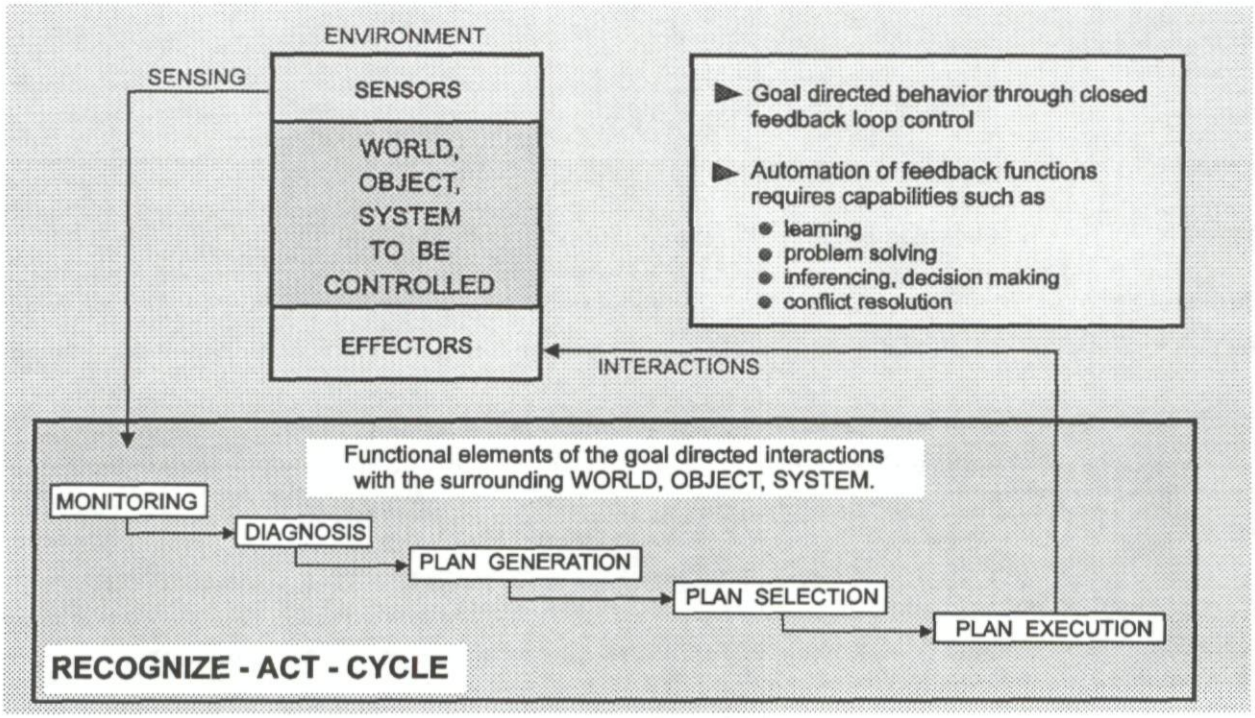


Figure 3: Interaction autonomous system - real world

For the purpose of full autonomy of future systems the metamorphosis to new structures of autonomous systems based on rapidly advancing Information Technologies (IT) represents the viable constituent and results in the concept of „Behavior-oriented Autonomous Systems (BOAS)". Behavior in this context means a suitable, goal-directed sequence of activities which, as a whole, represent a behavioral pattern. This concept can be adopted as metaphor for the imagination and implementation of a future or may be far future autonomous unmanned tactical system, that has to operate in complex, uncertain, unstructured, and none-benign environment.

An autonomous system must contain knowledge about the environment and the objects operating in it to be able to fulfill its characteristics. As shown in Fig. 4, an information-processing unit processes this knowledge using methods of artificial intelligence (AI). With the following definition, AI is the key characteristic of an autonomous system: Systems have artificial intelligence if their „creator" has given them a structure - not only a program - allowing them to organize themselves, to learn and to adapt themselves to changing situations. In this context it is important to mention that systems have no artificial intelligence if a program/software „injects" them with what they have to do and how they have to react to certain pre-specified situations.

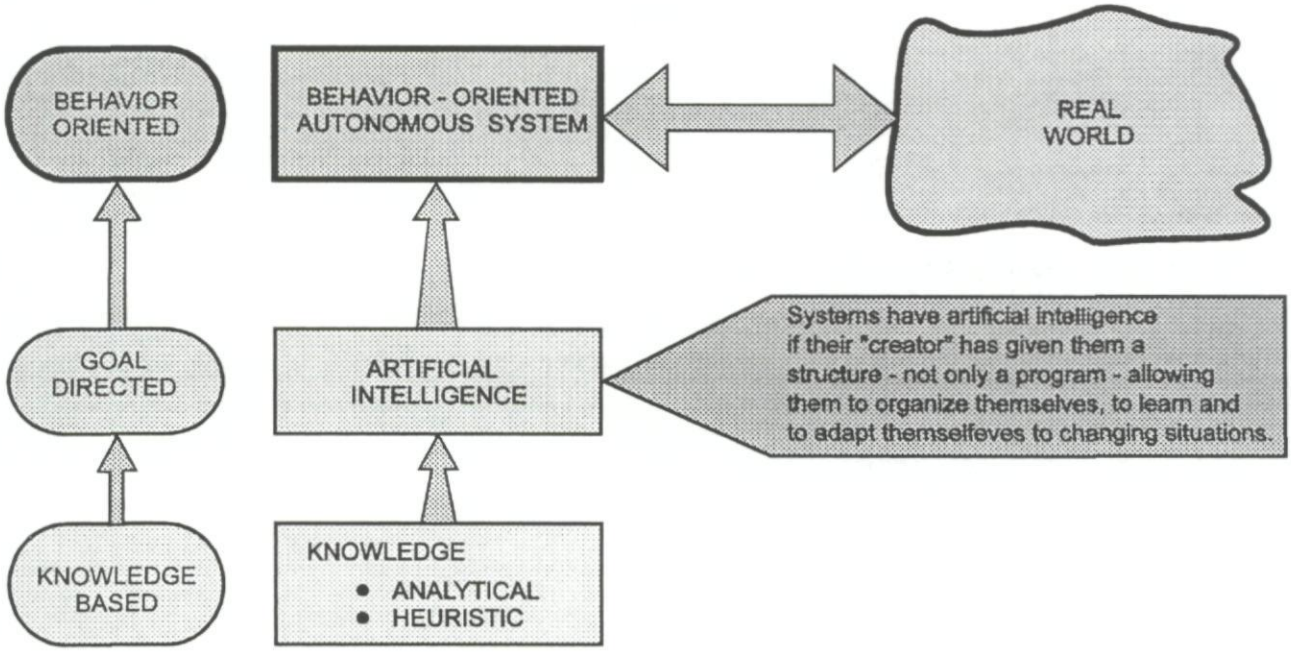


Figure 4: From knowledge to goal-directed autonomous behavior

The goal-directed interaction of the system with real-world objects is made possible through the technical implementation of the endomorphic system concept. This is based on a phenomenological approach where the ability of man to form internal models in the form of heuristic knowledge (experience knowledge) can be used as paradigm. The upper part of Fig. 5 shows how man intervenes in a partial area of the real world containing the object of interest in order to acquire knowledge and maps this partial area in his own sphere by modelling. The knowledge required to accomplish a task or to solve a problem is acquired by man both a-priori by training and on line by learning while performing the task. The problem is solved by processing this knowledge by means of processes on different mental functional levels. The respective partial area of the real world (e.g. object) is influenced by actions resulting from this. Through perception, the influence, in turn, activates cognitive learning processes, based on the goals set, for continuous adaptation of the knowledge required for task performance.

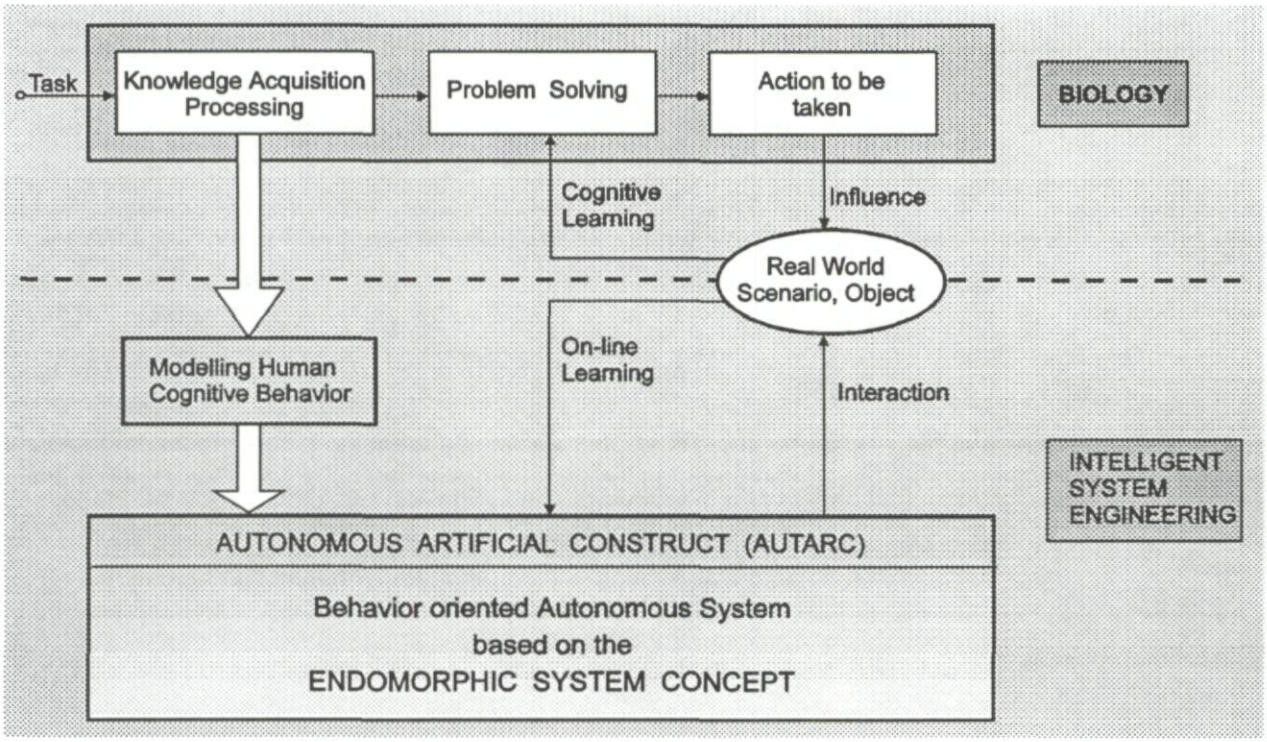


Figure 5: Modelling human cognitive behavior

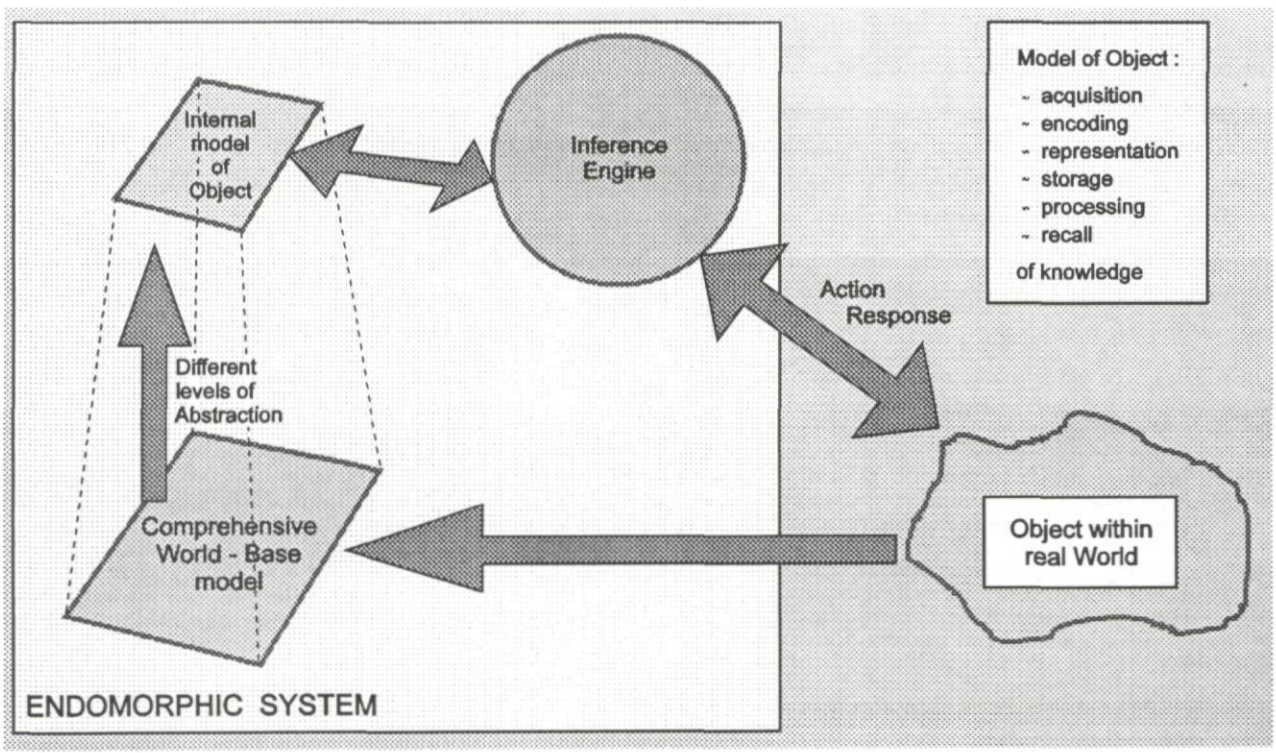


Figure 6: Endomorphic system concept

The technical reproduction of these functions in an Autonomous Artificial Construct (AUTARC) is achieved by implementing the endomorphic system concept [1] as represented in Fig. 6 in a simplified form. This is accomplished by establishing an endomorphism between real-world objects. The inference engine inquires at its internal models for necessary information prerequisite to goal-directed interaction with the real-world objects. The applied internal model is abstracted from a comprehensive model of the world and object. The Artificial Intelligence (AI) element of the system (see also Fig 4) comprises a domain-dependent knowledge and data base (real-world modeling) and a domain independent inference mechanism. In general the world-base model comprises different models such as continuous models, discrete-event models, rule-based models. The recognition - act-cycle, involving sensing, planning and execution, can thus invoke different models in an optimum way.

The technical implementation of the AI functions concerns the acquisition, the representation and storage of knowledge as well as the processing of knowledge to achieve goal-directed behavior. Enabling techniques for these tasks are treated in the following paragraph.

3 ENABLING NEW INFORMATION TECHNOLOGY

3.1 PARADIGM SHIFT TO BRAINLIKE STRUCTURES

The expected unprecedented advances in computing based on the conventional architecture, where processing is performed sequentially, do not yield the power for computational and machine intelligence. This becomes ultimately evident if we compare a contemporary high performance computer (Cray YMP/8) with the brain of an insect, as represented in Fig 7. A vast amount of hardware and software consuming a lot of power is needed to build and operate the Cray-machine, as compared to the biological brain. What is even more impressive is the level of capabilities achieved with the latter. To make at least a small step towards a technical implementation of biological intelligence, as it is needed for autonomous systems, new computational techniques and architectures must be considered and introduced. There is a paradigmatic complementary shift from symbolic artificial intelligence techniques to a new paradigm, which is inspired by modelling the conscious and unconscious, cognitive and reflexive function of the biological brain. Important related computing methodologies and technologies include inter alia fuzzy logic, neuro-computing and evolutionary and genetic algorithms. The basic functions of these technologies shall be briefly summarized in the following.

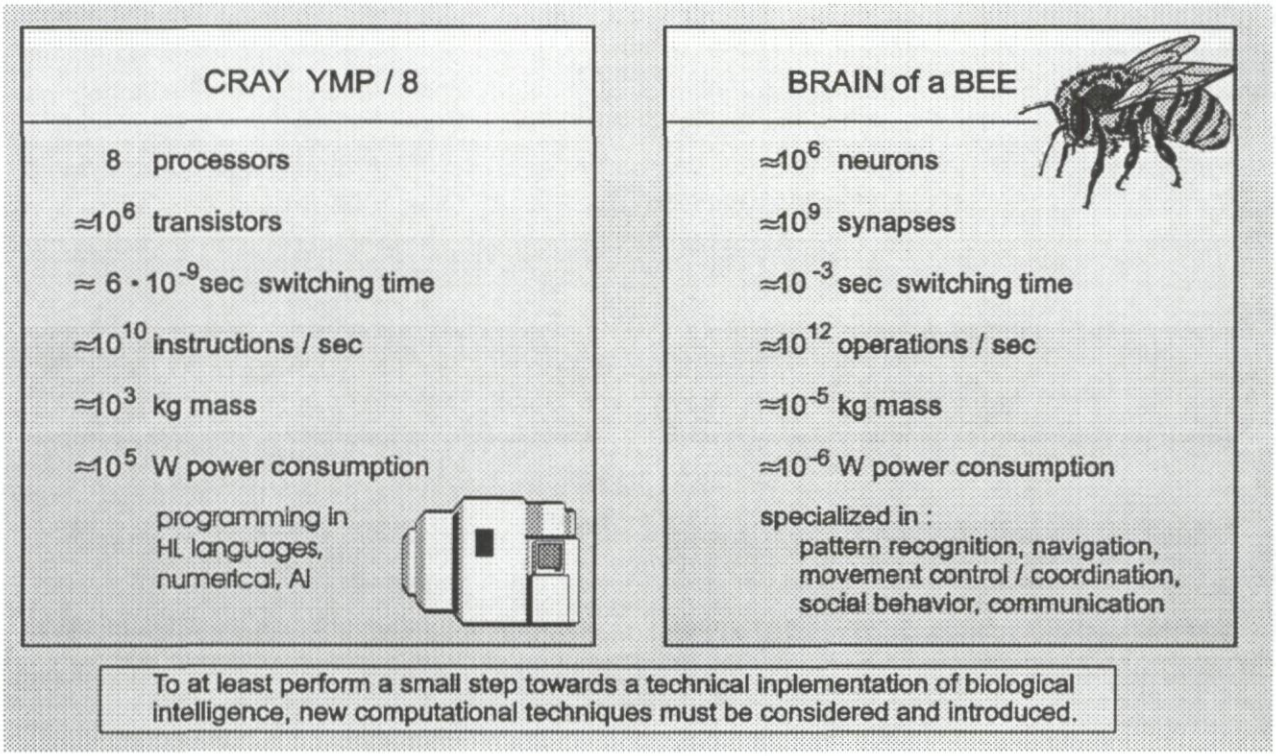


Figure 7: Comparison computer - brain of an insect

3.2 SOFT-COMPUTING TECHNOLOGIES

3.2.1 Fuzzy Logic

The theory of fuzzy logic provides a mathematical framework to capture the uncertainties associated with human cognitive processes, such as thinking and reasoning. Also, it provides a mathematical morphology to emulate certain perceptual and linguistic attributes associated with human cognition. Fuzzy logic provides an inference morphology that enables approximate human reasoning capabilities for knowledge-based systems. Fuzzy logic/fuzzy control has developed an exact mathematical theory for representing and processing fuzzy terms, data and facts which are relevant in our conscious thinking.

A unit based on fuzzy logic represents an associator that maps spatial or spatiotemporal multivariable inputs to corresponding associated outputs [2]. The knowledge which relates inputs and outputs is expressed as fuzzy if-then rules at the form IF A THEN B, where A and B are linguistic labels of fuzzy sets determined by appropriate membership functions. By fuzzy if-then rules the qualitative aspects of human knowledge reasoning and decision processes can be modelled without the need for a precise quantitative description. Fuzzy logic units, also known as fuzzy-rule-based systems, fuzzy models, fuzzy associative memories (FAM) or fuzzy controllers, consist of five fundamental functional blocks as shown in Fig 8. The fuzzification interface transforms the crisp inputs into degrees of memberships to linguistic labels of fuzzy sets. A number of fuzzy if-then rules is contained in the rule base. They comprise the knowledge about the relationship between the antecedent and consequence variables. The membership functions of the linguistic

fuzzy sets as used in the fuzzy rules are defined within the data base. Rule and data base are often jointly referred to as the knowledge base. The inference unit is the decision making element and performs the inference operation applying crisp mathematics of fuzzy logic. The fuzzy result of the inference process is transformed into a crisp output by the defuzzyfication block. By fuzzy logic processing the vague imprecise rules can be given a specific meaning and outputs can be produced for inputs which only partially match the rules. This is a very powerful attribute.

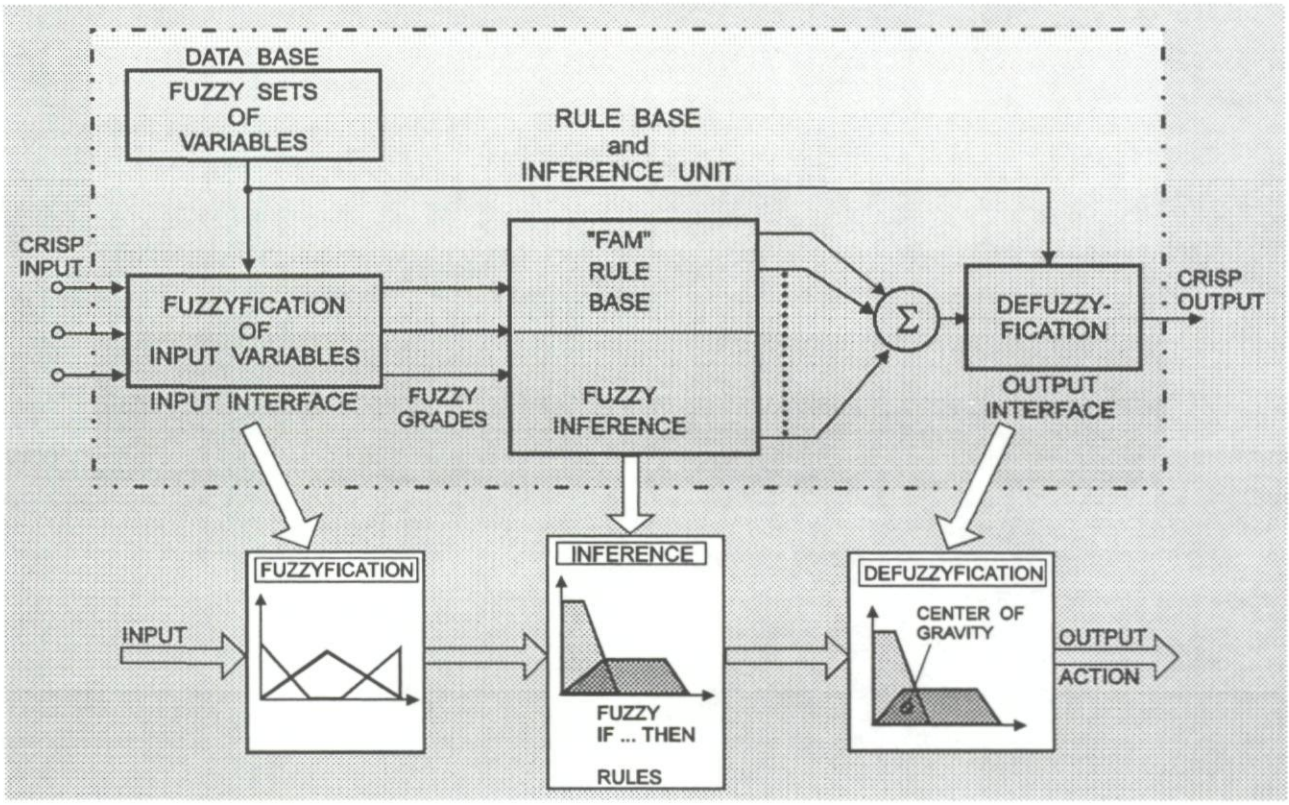


Figure 8: Fuzzy logic unit

Fuzzy rule based systems enable endomorphic real world modelling. With this technology human behavior can be emulated in particular as far as reasoning and decision making and control is concerned taking into account the pervasive imprecision of the real world. Fuzzy logic strongly supports realistic modelling and treatment of reality.

3.2.2 Artificial Neural Networks (ANN)

Neural Networks are derived from the idea of imitating brain cells in silicon and interconnecting them to from networks with self-organization capability and learnability. They are modeled on the structures of the unconscious mind.

Neurocomputing is a fundamentally new kind of information processing [3]. In contrast to programmed computing, in the application of neural networks the solution is learnt by the network by mapping the mathematical functional relations. Neural networks are information processing structures composed of simple processor elements (PE) and networked with each other via unidirectional connections. Fig. 9 shows as an example a multi-layer network with exclusively forward

connections. The "knowledge" is contained in the variable interconnection weights. They are adjusted during a learning or training phase and continue to be adapted during operational use. With this capability the ANN represents an associator (like a fuzzy logic unit) that maps spatial or spatio-temporal multi-variable inputs to corresponding associated outputs. However, in contrast to a fuzzy-rule-based system the mapping function is learnt by the ANN. Neural Networks are capable of acquiring, encoding, representing, storing, processing and recalling knowledge. These are important prerequisites for endomorphic real world modelling (see Fig 6).

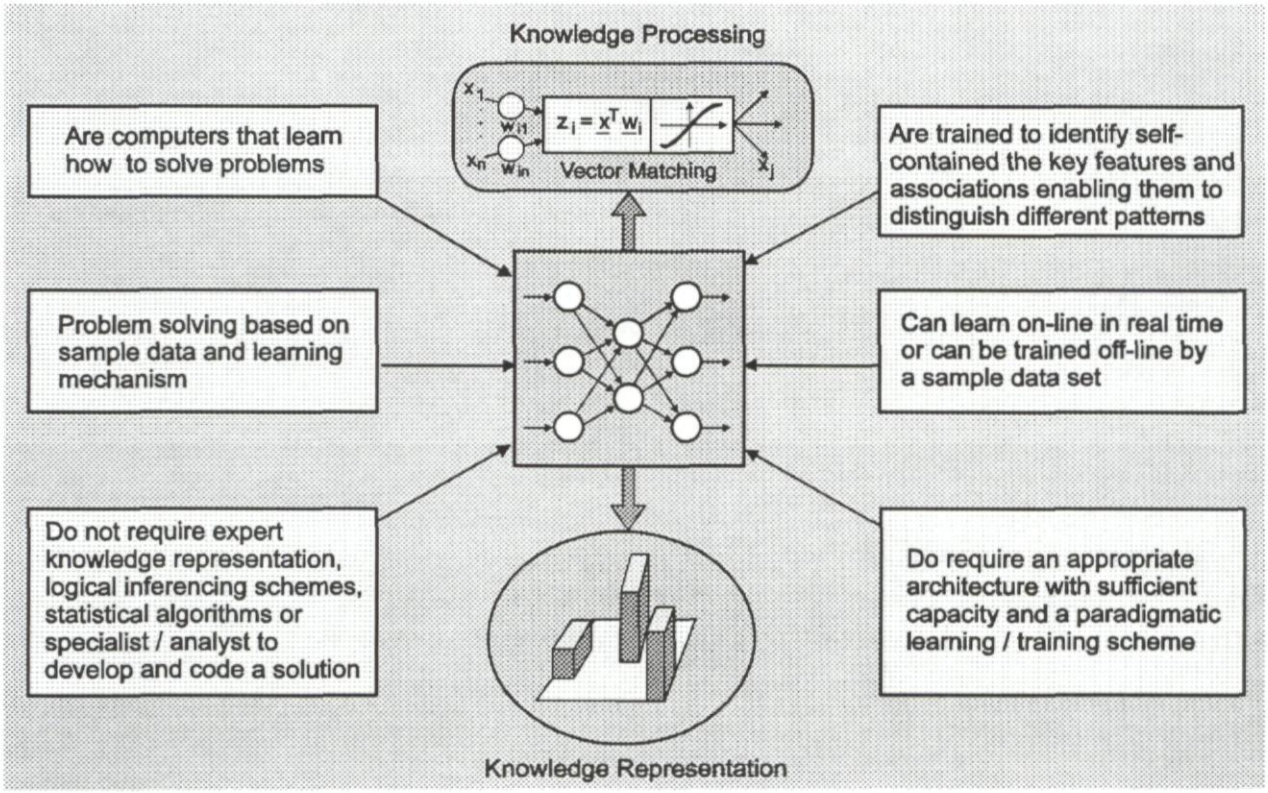


Figure 9: Artificial neural networks

3.2.3 Genetic and Evolutionary Algorithms

Genetic and evolutionary algorithms represent optimization and machine learning techniques, which initially were inspired by the processes of natural selection and evolutionary genetics [4].

To apply a genetic algorithm (GA) potential solutions are to be coded as strings on chromosomes. The GA is populated with not just one but a population of solutions, i.e. GA search from a population of points rather than from a single point. By repeated iterations a simulated evolution occurs and the population of solutions improves, until a satisfactory result is obtained. The initial population evolves through rules which are independent of the problem, tend to generate better and better solutions and utilize probabilistic transition rules. As shown in Fig 10 in a simplified form this is accomplished by iteratively applying the genetic operators reproduction, crossover and mutation. Strings are copied and parts swapped as well as changed in a probabilistic manner by the genetic operators. As an illustrative example, the evolutionary development of a Neural Network is suggested in Fig 10. The network structure and parameters are coded as strings on chromosomes. Crossover of „parent" networks generates offsprings, i.e. „children". The network is

optimally generated by time-lapsed simulation of biological evolution. All the user needs to know is how to evaluate a solution. For this purpose an objective or fitness function has to be defined. It measures the performance or goodness after reproduction, i.e. for each generation.

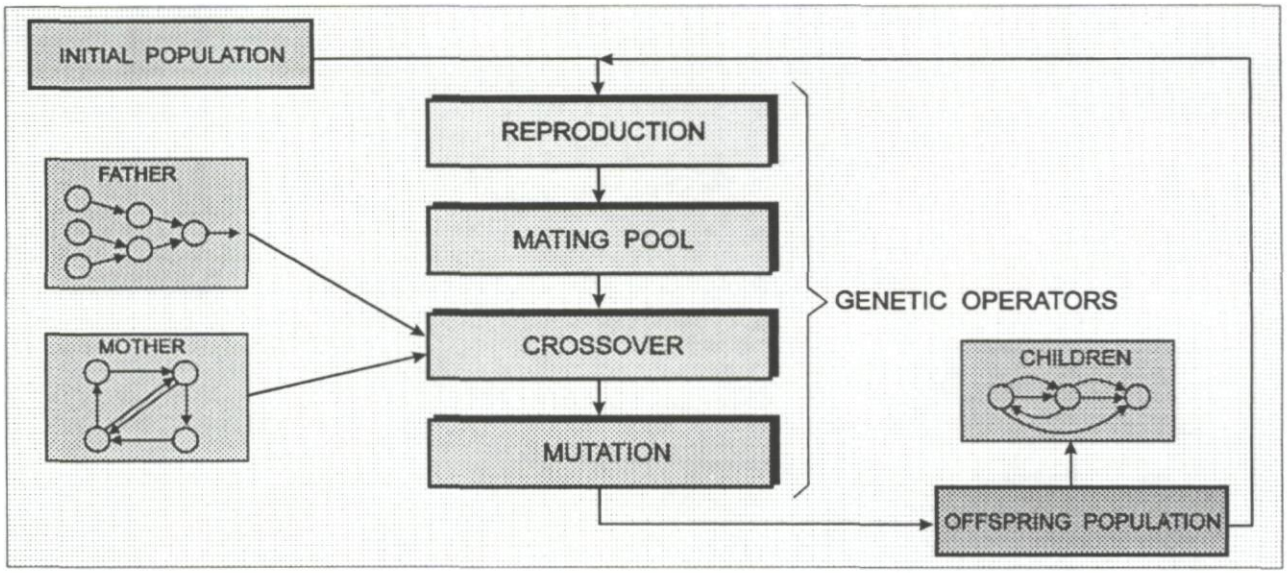


Figure 10: Genetic operations for a single generation

Computer simulation is a viable tool to optimize behavior oriented systems by utilizing genetic or evolutionary techniques. Ever increasing processing speed enables the quick motion representation of events and processes, for which nature requires millions of years.

3.2.4 Conclusions

It was shown that fuzzy and artificial neural network techniques enable the endomorphic modelling of real world objects and scenarios. Together with conventional algorithmic processing, classical expert systems, probabilistic reasoning techniques and evolving chaos-theoretic approaches they enable the implementation of recognize-act cycle functions as shown in Fig 3. Genetic and evolutionary algorithms can be applied to generate and optimize appropriate structures and/or parameters to acquire, encode, represent, store, process and recall knowledge. This yields self-learning control structures for dynamical scenarios that evolve, learn from experience and improve automatically in uncertain environment. Ideally, they can be mechanized by a synergetic complementary integration of fuzzy, neuro and genetic techniques. These soft-computing techniques support the move towards adaptive knowledge based systems which rely heavily on experience rather than on the ability of experts to describe the dynamic, uncertain world perfectly. In contrast to conventional computing, soft-computing addresses the pervasive imprecision of the real world. This is accomplished by consideration of the tolerances for imprecision, uncertainty and partial truth to achieve tractable, robust and low cost solutions for complex problems.Thus, soft-computing techniques in conjunction with appropriate system architectures provide the basis for creating the above-mentioned Behavior-Oriented Systems. In the following, this will be looked at in somewhat greater detail.

4 CONCEPTUAL IDEAS FOR AUTONOMOUS SYSTEMS

4.1 System architectures

The viable architecture must represent the organization of the systems intelligence and capability to behave, to learn, to adapt and to reconfigure in reaction to new situations in order to perform in accordance with its functionalities. Based on fundamentally different philosophies regarding the organisation of intelligence, two different architectures can be basically considered (Fig. 11). With the well known top-down approach as prevalently used to date a hierarchically functional architecture results. It structures the system in a series of levels or layers following the concept of increasing precision with decreasing intelligence when going from top to bottom. Implementation is characterized by the fact that for as many contingencies as possible the allocated system behavior is fixed in top-down programming. This method is "unnatural" and, according to the definition given above, does not lead to artificial intelligence of the system. In fact, the real world is so complex, imprecise and unpredictable that the direct top-down programming of behavioral functions soon becomes almost impossible.

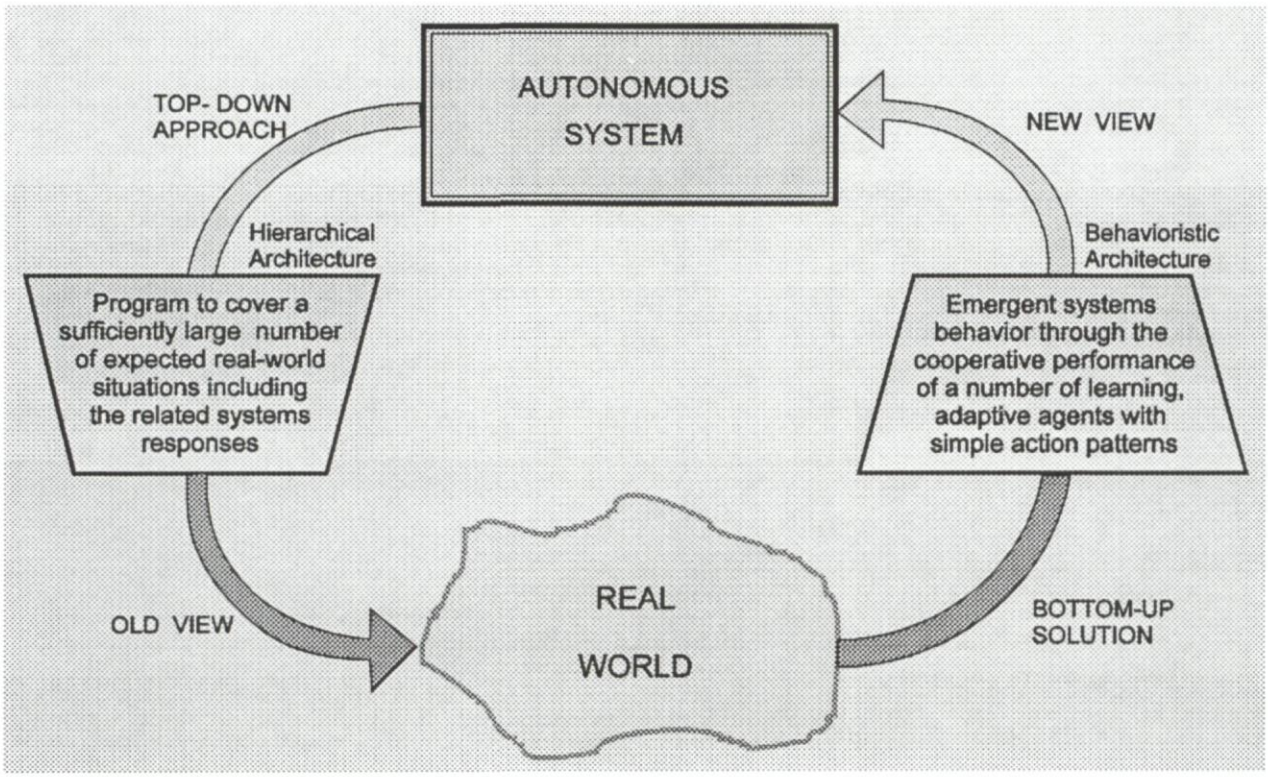


Figure 11: Organization of the systems intelligence

Considerably different from the hierarchical structure is the subsumption architecture. It is based upon building functionality and complexity from a number of simple, parallel, elemental behaviors. It is sometimes called the behaviorist architecture and is based on a buttom-up approach. In this approach, so-called agents are implemented with the most simple action and behavior patterns possible so that the resulting emergent system behavior corresponds to the desired global objective. The system is able to adapt itself to changing situations in the environment by learning.

The specific local intelligence of the individual agents generates a global intelligent behavior of the integrated overall system. Multi-agent systems are very complex and hard to specify in their behavior. Therefore there is the need to endow these systems with the ability to adapt and learn. This learning capability is very important, because otherwise a multi-loop nonlinear control problem of very high order must be solved. The current state of the art precludes a solely analytic solution for such a complex system. For this reason we invoke at the field of biology.

The biological nerve system is the living example for the fact that strongly meshed systems of an extremely high order can adopt stable states. Moreover, without supervised control, these biological systems are able to act purposely and task oriented. By an extensive comprehension of the biological paradigm, the brain, we must try and strive to recognize the regularities which might be of decisive use to us for the stabilization and self-organization of highly integrated complex dynamic systems.

A simplified block diagram of an autonomous system based on such a concept of cooperative AI/KB-Agents, is depicted in Fig 12. The objective is to implement as many simple agents as possible with the associated behavior pattern, which then make the system act in a flexible, robust and goal-oriented manner in its environment through their additively complementary interaction. To enable the generation of emergent characteristics it must be ensured that the agents can influence each other mutually. Emergent functionality is one of the major fields of research dedicated to behavior-oriented systems.

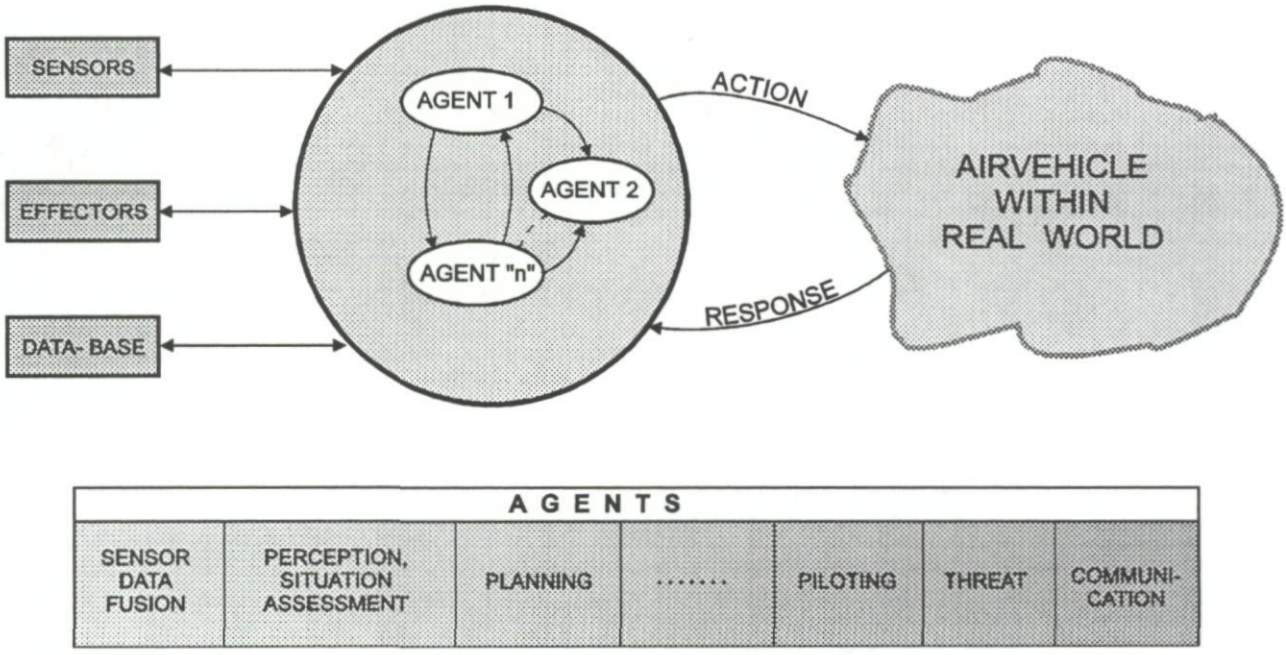


Figure 12: Endomorphic system representation by agents

4.2 Agents

Intelligent HW/SW agents will fuse sensor information, monitor critical variables, generate optimized plans, alert operators through communication to problems as they arise and recommend optimized solutions in real time. Response agents capture basic data, communication (forecast

and other information) and apply optimization technology to generate new plans based on changed conditions and states. Remote Operators can consider recommended plans in a "what if" manner, i.e. making changes to the agents suggestions or accepting new recommendations. Last, but not least, agents will offer learning capability and will be designed almost automatically in the future.

The structure of an agent of the lower functional level is shown in Fig. 13 using the example of an air vehicle piloting agent. It contains two neural networks (control [CN], plant model [PN]) and a fuzzy neural network (performance [FN]). The networks are interconnected in a network system for cooperative function. The CN has learnt in a training phase to stably fly the air vehicle in all aerodynamic ranges. The PN as nonlinear, dynamic model of the air vehicle maps its flight characteristics in the complete flight envelope. The PN continuously adapts itself to changes of the air vehicle by continued on-line learning. The FN is used to allocate a performance function dependent on the aerodynamic state of flight. Via the performance feedback, the CN learns to adapt itself to the desired, feasible performance, taking into account the current dynamic behavior of the air vehicle. The collective behavior of the individual networks finally consists in allocating action signals to - partially sensory - input signals, which on the whole, yields a learning, dynamic control behavior.

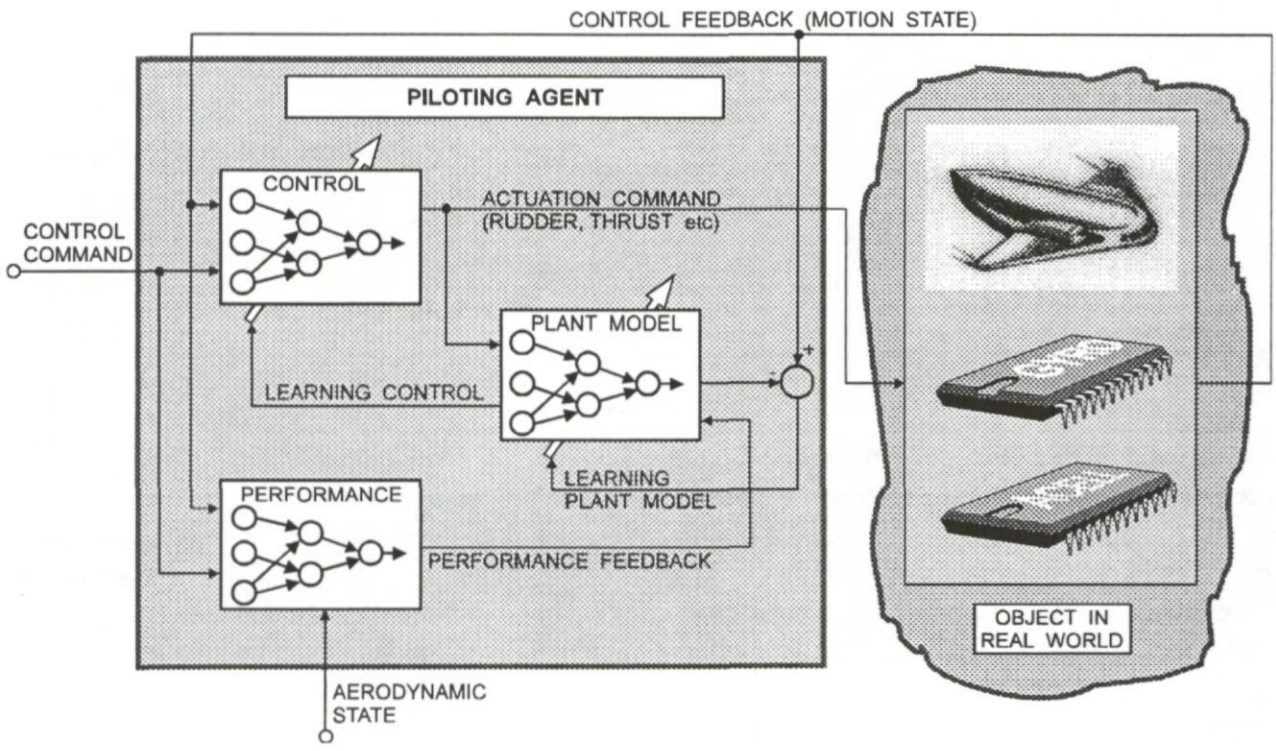


Figure 13: Structure of the piloting agent

Owing to the necessary precise description and representation of knowledge about action-effect-correlations on upper functional levels (e.g. planning, mission management) in complex, dynamic scenarios, the automation of such functions has been difficult until now. The use of neuro-fuzzy systems (agents) enables a new kind of acquisition, representation and adaptation of the knowledge concerned. Fig. 14 shows this in a very simplified form for a knowledge-based planning process.

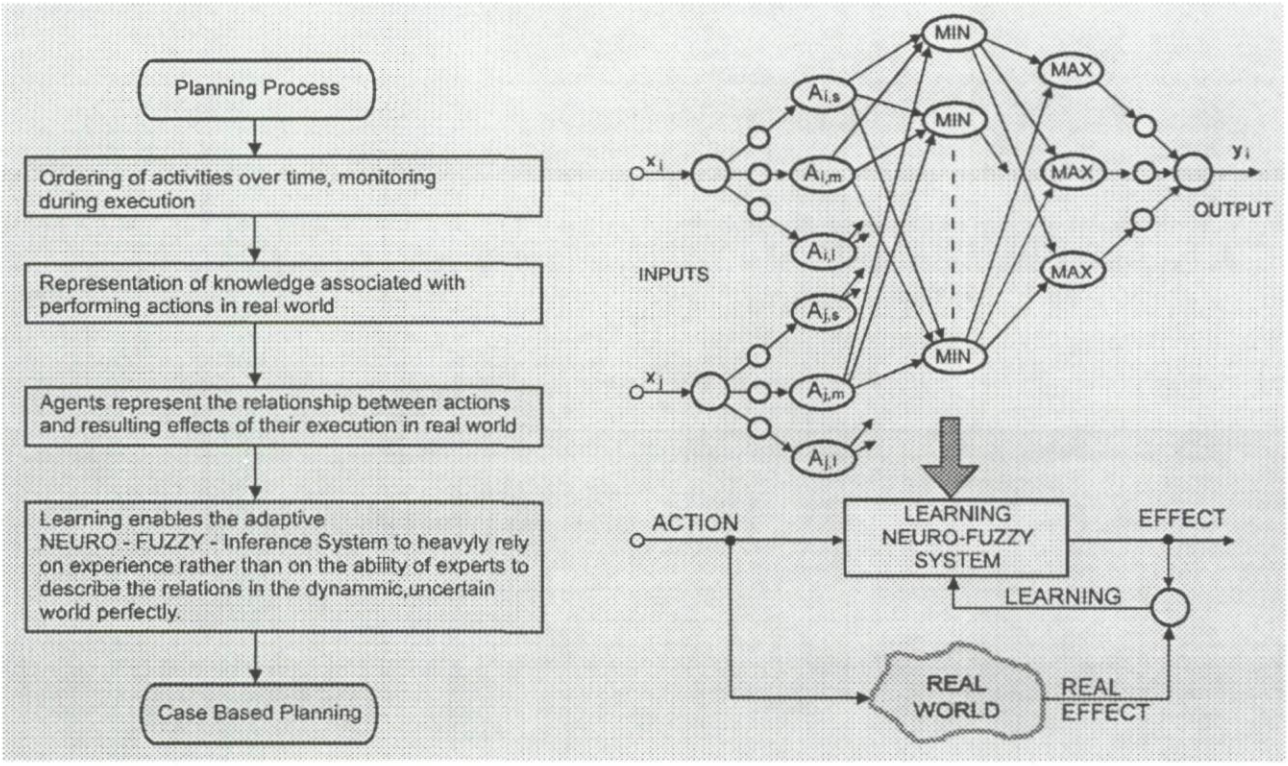


Figure 14: Knowledge based planning

4.3 Design Considerations

As explained, behavior-oriented agent systems can have a hierarchical structure or work with a cooperative structure with distributed supervision. For practical application in the near future, a mutually complementary integration of both approaches can be expected. This, because there is no doubt that a - presumably complex-balance exists between the learnt and the genetically determined parts in the behavior of biological organisms. A similar situation is likely to arise in the engineering process of autonomous systems. The problem will be to decide what should be explicitly programmed and what should be left for the system to learn from experience. However, it is the Author's opinion that in the long run the parallel, subsumption structure will be the prevalent architecture for behavior oriented AUTARC systems. As summarized in Fig. 15, applying the endomorphic system concept, case based reasoning and planning as well as supervised and reinforcement type learning, it can be implemented by extensive use of soft-computing technologies. These and many other issues will be part of a new discipline, which by analogy to „Knowledge Engineering“ is sometimes referred to as „Behavior Engineering“. Its objective is to provide methodologies and tools for developing and applying autonomous systems, which is a very complex engineering enterprise.

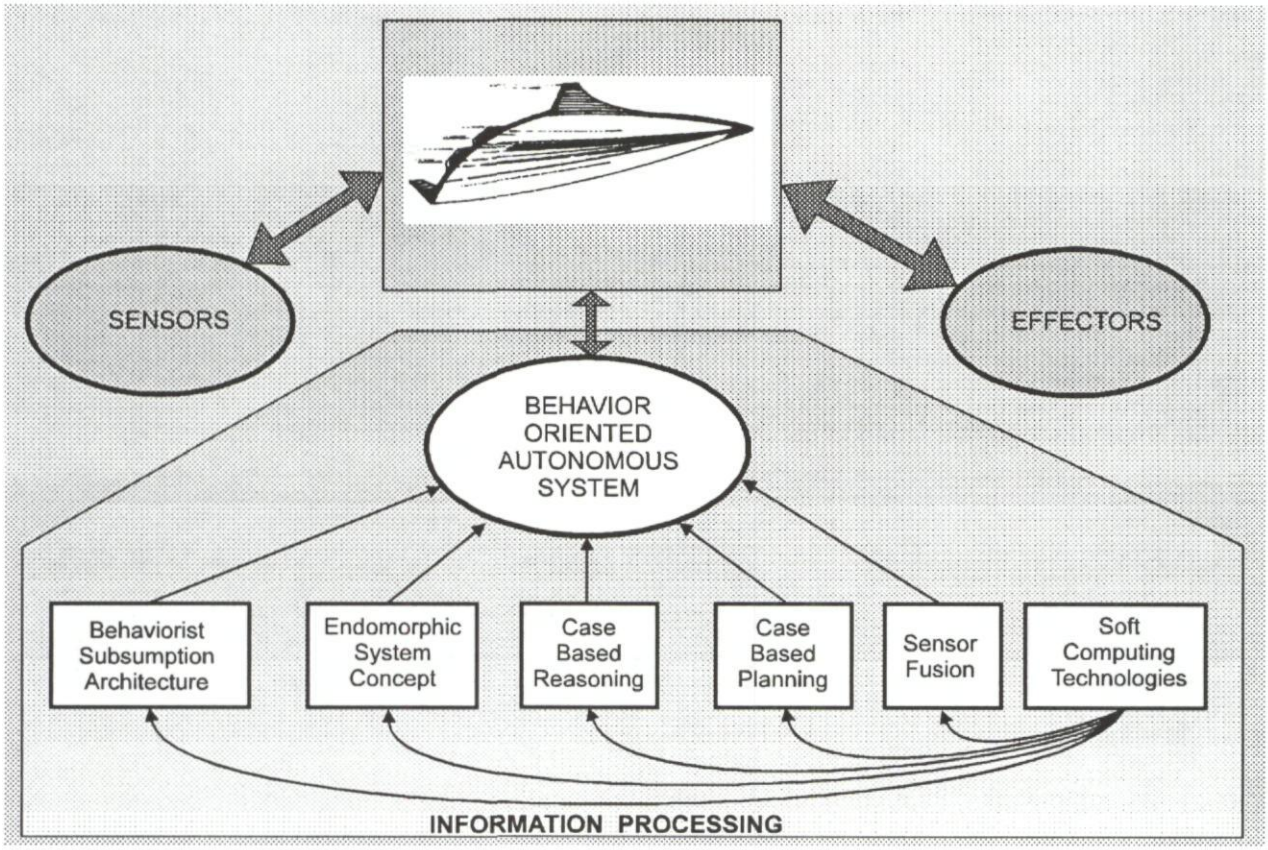


Figure 15: Main processing elements

Like in Engineering, it is also an indispensable prerequisite for an AUTARC that it is designed, constructed and trained according to a strict methodical approach. Fig. 16 shows such an approach in a very simplified form from today's technological point of view.

It starts with the description of the physical AUTARC, its application, the initial environment, and the behavior requirements, with the latter being usually informally stated in natural language. The following behavior analysis is one of the major tasks. This step involves the decomposition of the target behavior in simple behavioral components and their interaction. Part of the specification is the architecture of the intelligent control system. It is the second key point during the engineering process. With the specification all information is available to design, implement and verify a nascent AUTARC, which is endowed with all its hardware and software components, however, prior to any training.

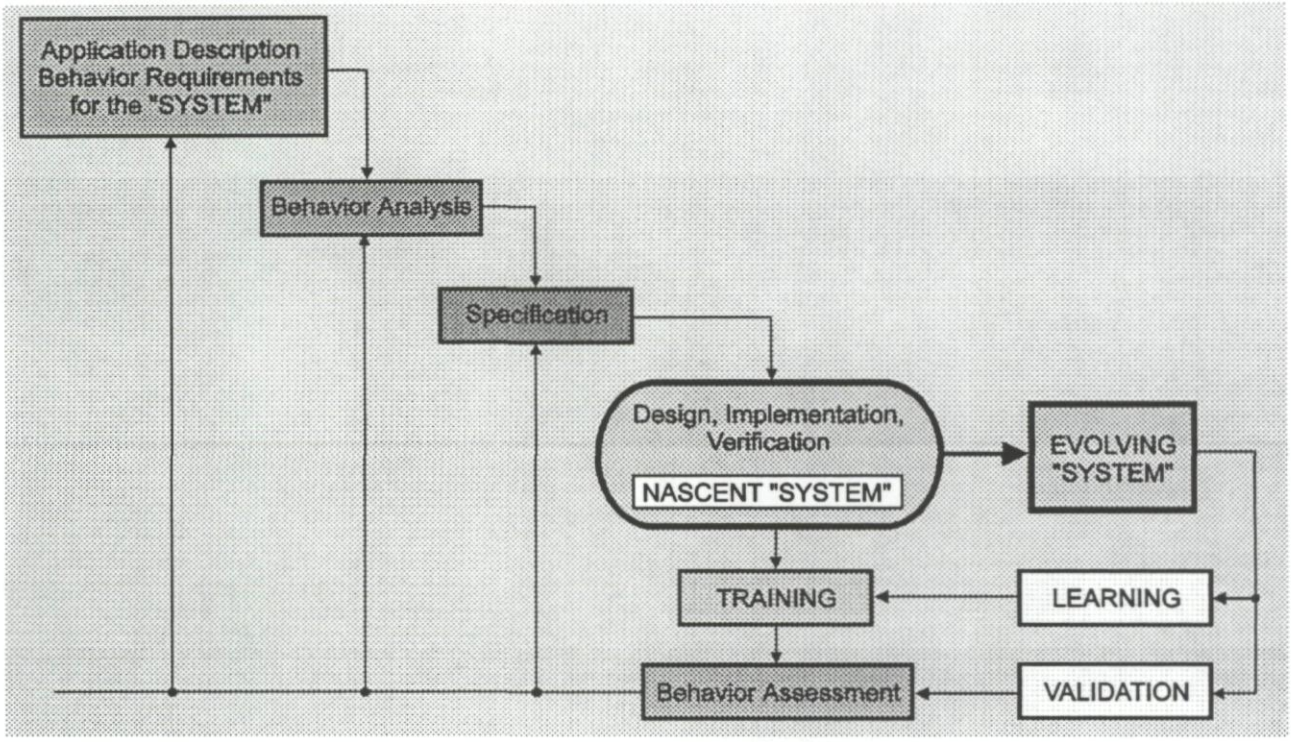


Figure 16: Global methodology in the engineering of the behavior oriented system

Based on a suitable training strategy the system acquires its knowledge during a training phase which is mandatory and prerequisite for appropriate behavior of the system. Training can usually be speeded up applying simulation including virtual reality. Within this context environments can be used that are much more changeable than the real ones.

After completion of training the behavior is assessed with respect to correctness (target behavior), robustness (target behavior vis-à-vis changing environment) and adaptiveness. Based on this assessment, further iterations during the engineering steps might become necessary in order to make the satisfactorily behaving AUTARC evolve from them in a step by step sense.

5 IMPLEMENTATION ISSUES

As shown in Fig. 17, high performance on-board computing will be made possible by the arrangement of scalable parallel computers in network structures. Powerful optic switchable network links will connect several networks to form large cooperative network structures. The hierarchically supervised performance of these structures will be augmented by a goal directed learning feature. Thus, supervised cooperative networks will replace command-control (master-slave) structures, which supports the implementation of multiple agent system concepts.

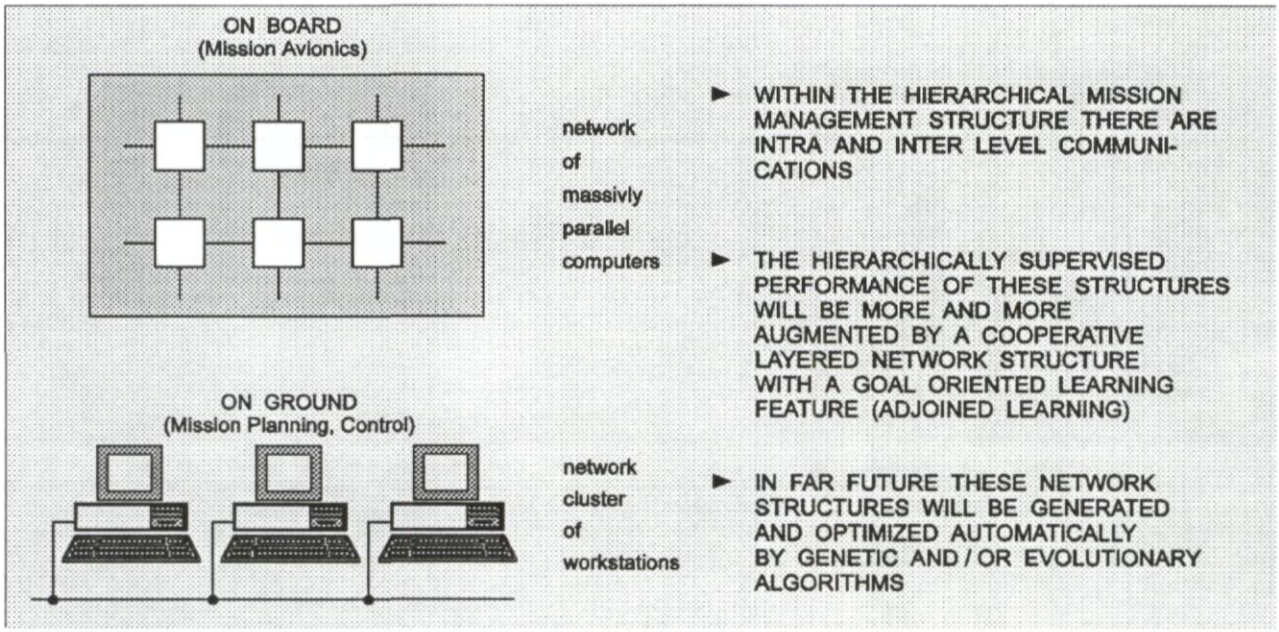


Figure 17: Computational network structure

There will be neuro-computers (in 5 to 10 years) whose capacity will be approximately equal to that of the biological neuronal network of an insect (one million neurons, one billion synapses). They will be implemented electronically or optically (cubic centimeter volume).

Further implementation issues like [6]

- hardware for computational and machine intelligence
- software technology, software generation techniques
- autonomous control technology
- integrated system structures and functions

could not be treated here. It is referred to the Literature, e.g. [6].

6 EMERGENCE OF AUTONOMOUS SYSTEMS

Advancing Technologies are essential for achieving unprecedented capabilities for new systems at affordable cost. Looking at Fig. 18 (upper left) the yield/cost ratio is plotted against the commulative expenses for old and new technologies (e.g. Information Technology). Considering the general performance potential, the transition to new technologies is mandatory to offer new opportunities and improved yield/cost ratios.

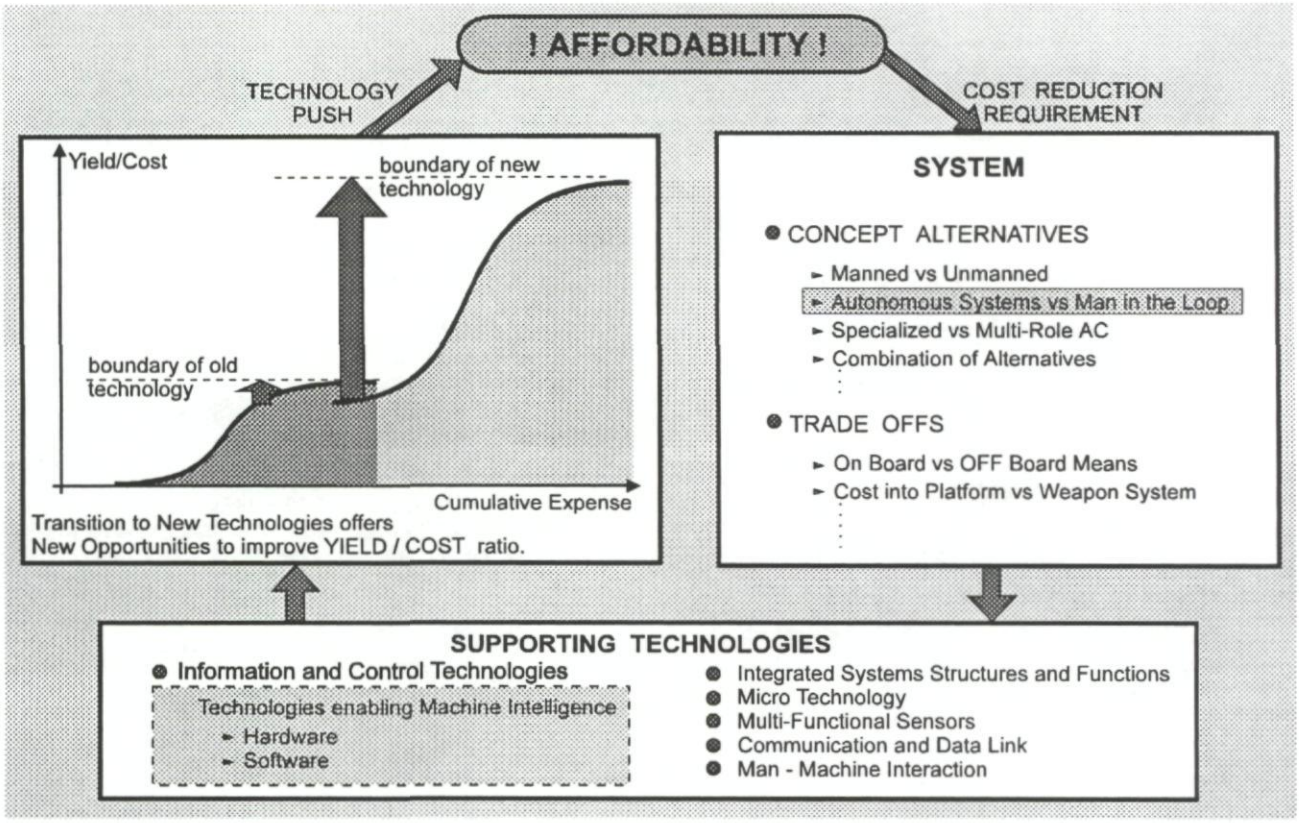


Figure 18: Requirements pull versus technology push

The approaches treated here will significantly contribute to the advent of autonomous tactical systems with revolutionary capabilities. They will be become feasible through rapidly advancing Information-, Micro- and Sensor- as well as Communication-Technologies. New platform and weapon technologies, which cannot be used within a manned system, will become applicable, such that new systems will be so far beyond anything within contemporary experience, that its architecture and some technologies may prove radically different from anything envisioned today. Beyond the enabling technologies further technical issues such as

- maturity assessment
- system concepts
- critical experiments
- validation, certification techniques
- future research focus

shall be emphasized, because they critically influence the emergence of autonomous systems. Stepping back to the first chapter and recalling the interdependence of the Requirements Pull and the Technology Push it is of paramount importance for research planners to identify applications and requirements indicating the indispensable need for such systems and their capabilities. In this context the Uninhabited Tactical Aircraft (UTA) concept of variable autonomy currently under investigation, offers an ideal platform to perform critical experiments for the evaluation, validation and possibly certification of techniques and technologies.

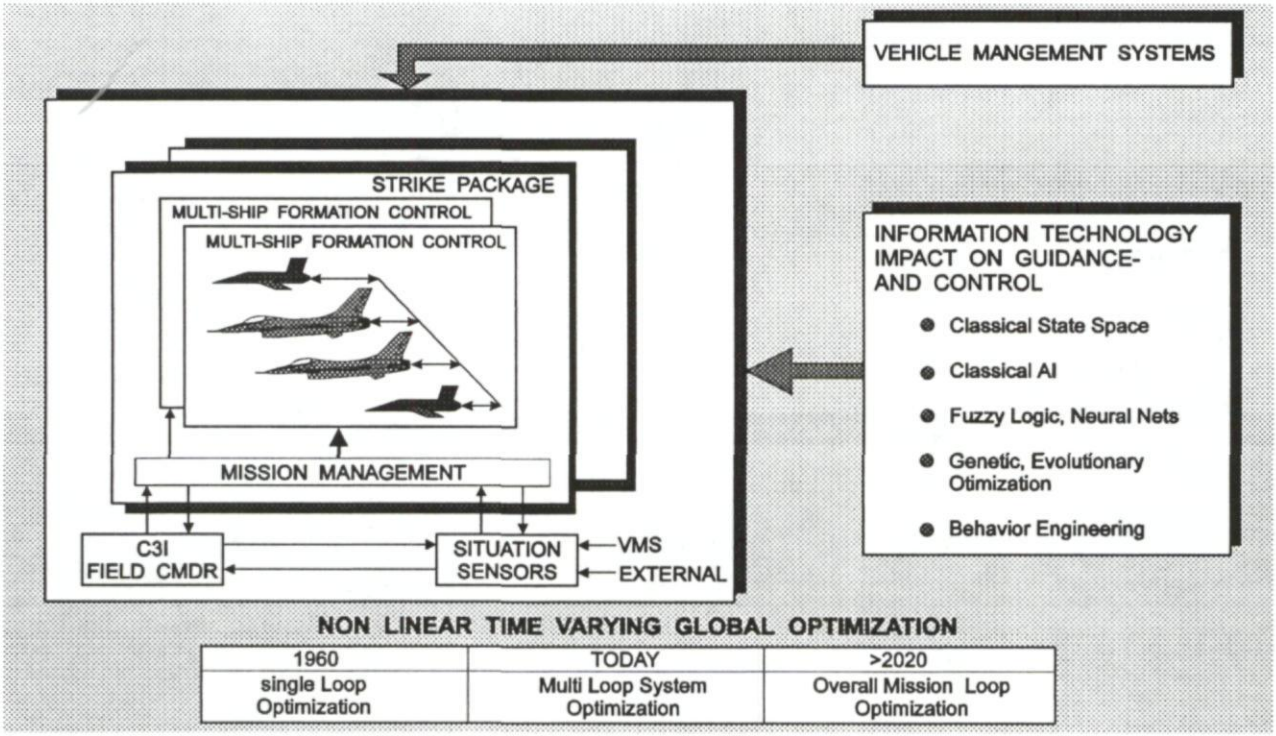


Figure 19: Integrated battle management system

Recalling the affordability issue, the technologies enabling autonomous systems as treated here together with other supporting technologies, such as those shown in Fig. 18, can foster thoughts towards mission systems concept alternatives and trade-offs with the potential for cost reduction. Several technologies will be viable for the development of such systems. However, the most important challenges are in the area of system integration. Fig. 19 shows for example the concept of an Integrated Battle Management System, where as multi-ship formation autonomous unmanned vehicles are internetted with manned aircraft. The involvement of users in conception, development and demonstration will be essential as the overall mission loop and its optimization evolves. Without the claim to be nearly complete further related research should be focussed in areas such as

- LOS/SATCOM data links for high maneuverability
- smart skins, multifunctional structures
- information fusion
- autonomous 360° situation awareness and associated sensors
- autonomous planning and routing
- adaptive autonomy management

One of the fundamental concepts in behaviour oriented systems is the partitioning and subsumption of the overall task into as many subtasks as possible and performed by corresponding system elements. Generalizing this concept it can also be envisioned that instead of a complex AUTARC a group of simply structured, miniaturized airborne robots/vehicles with dedicated sub-behaviors are cooperating together in order to perform the overall task (Fig. 20). In addition to the potential for cost reduction this concept offers improved mission reliability.

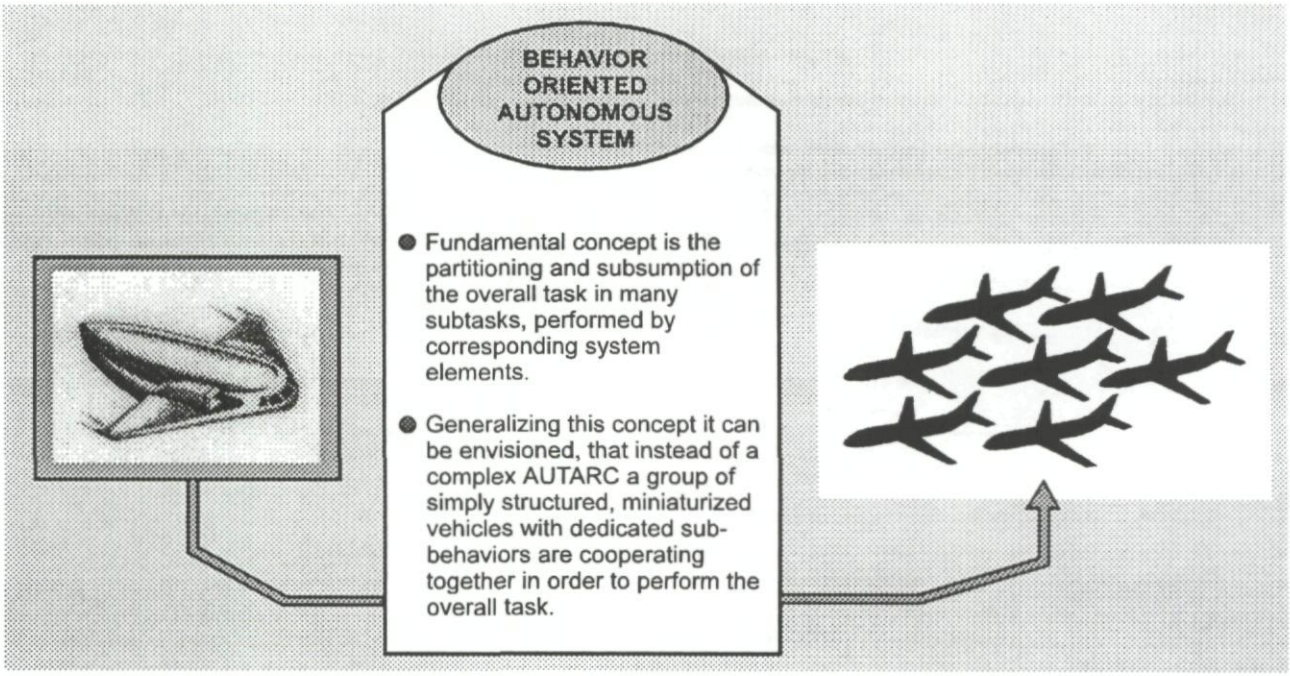


Figure 20: Emergent capabilities from simple behaviors

7 IMPLICATIONS ON MANNED SYSTEMS

As far as manned systems are concerned, the technologies stressed in this paper enable the design of tools that help the human brain to have better ideas, to generate better solutions and to respond faster to cope with complex situations. Corresponding new support systems will increase the operational tempo (bandwidth) of the human plus machine system, such that the recognize-act-cycle gains a time advantage as compared to that of the opponent. (Fig. 21)

Simultaneously, Life-Science research will discover new ways to further move forward the limits of human mental and related physical capabilities. This will be based on results regarding human brain structures and functions. With new approaches to model the human brain by brain-like structures, implemented in technical constructs (biotechnical in far future), new ways for the interaction of the human with intelligent machines will yield considerable improvements in efficiency and efficacy (see concept in Fig. 19).

8 FINAL REMARKS

Finally, the question arises whether it is really possible to predict scientific and technical developments over rather long periods of time. Are such attempts legitimate and necessary? In this connection, it must be taken into account that - in particular in predictions concerning the development of Information Technology - there is a close correlation between the situation to be predicted and the social and sociological environment. The problem here is that the latter can hardly be predicted with sufficient precision because this is a complex, dynamic process with many influencing factors.

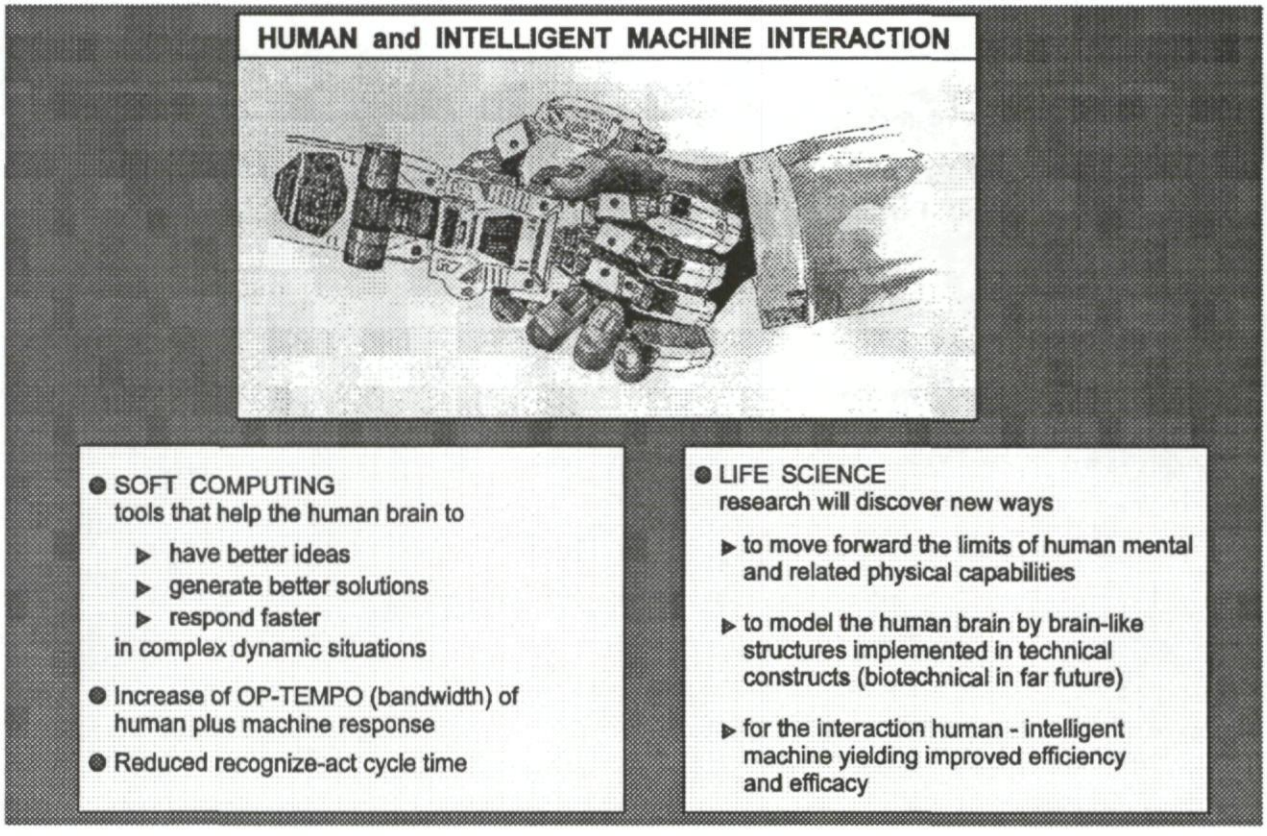


Figure 21: Human and machine interaction

However it can be stated undoubtedly, that significant changes are currently taking place in the new IT, MT and other technological areas as far as functional capabilities, performance, characteristics and cost are concerned. These changes will influence the users of these technologies and the supporting industries as well as their technical and organisational structures. Organizational structures have always reflected system structures. The rate of change and related realizations will exceed normal evolution and will have great social impacts accompanying the technological and functional advances. Instead of spin-offs considerable spin-in effects from commercial research and industry will impact military applications. Simultaneously a global availability of commercial High-Tech must be assumed.

In order to accommodate all this, the strategies of users and industry must be adapted accordingly. Looking at the interdependence of requirements, technologies, procurement processes and time behavior, 20 years is a short period. WE MUST BEGIN NOW!

Literature:

[1] B.P. Zeigler et. al
Model-Based Architecture for High Autonomy Systems

S.G. Tzafestas (ed): Engineering Systems with Intelligence,
Kluwer Academic Publishers, Netherlands.

[2] B. Kosko
Neural Networks and Fuzzy Systems
Prentice Hall Inc., 1992

[3] U. Krogmann
Introduction to Neural Computing
AGARD LS179, Monterey, USA, Oct. 1991

[4] E. Sanchez
Genetic Algorithms and Soft Computing
1. European Congress on Fuzzy and Intelligent Technologies
7.-10. Sept. 1993, Aachen, Germany

[5] L. Steels, D. Mc Farland
Artificial Life and autonomous robots
Tutorial 11th. European Conference on Artificial Intelligence, Amsterdam, 1994

[6] U. Krogmann et. al.
Aerospace 2020, Vol. 3
Critical Enabling Technologies
AGARD, Paris, 1997

REPORT DOCUMENTATION PAGE

1. Recipient's Reference	2. Originator's Reference	3. Further Reference	4. Security Classification of Document
	AGARD-LS-210	ISBN 92-836-1061-X	UNCLASSIFIED/ UNLIMITED

5. Originator Advisory Group for Aerospace Research and Development
North Atlantic Treaty Organization
7 rue Ancelle, 92200 Neuilly-sur-Seine, France

6. Title

Advances in Soft-Computing Technologies and Application in Mission Systems

7. Presented at/sponsored by

The Mission Systems Panel and the Consultant and Exchange Programme of AGARD, presented on 17-18 September 1997 in North York, Canada; 22-23 September 1997 in Amsterdam, The Netherlands; 25-26 September 1997 in Madrid, Spain; and 6-7 October 1997 in Ankara, Turkey.

8. Author(s)/Editor(s)	9. Date
Multiple	September 1997
10. Author's/Editor's Address	**11. Pages**
Multiple	112

12. Distribution Statement There are no restrictions on the distribution of this document. Information about the availability of this and other AGARD unclassified publications is given on the back cover.

13. Keywords/Descriptors

Artificial intelligence
Software engineering
Precision
Mathematical models
Fuzzy sets
Neural nets
Algorithms
Genetics
Integrated systems
Mission effectiveness
Computer architecture
Target acquisition
Infrared tracking
Airframes
Damage

14. Abstract

Contains the papers presented at a Lecture Series on 'Soft Computing' technologies. Soft computing addresses the pervasive imprecision of the real world by consideration of the tolerances for imprecision, uncertainty and partial truth to achieve tractable, robust and low-cost solutions for complex problems.

Topics covered include soft computing for computation and machine intelligence, neural networks, fuzzy logic, inference and fuzzy control, hybrid architectures for intelligent and learning inference systems, and applications to target tracking and acquisition and the reconfiguration of damaged aircraft.

The Lecture Series was sponsored by the Mission Systems Panel of NATO's Advisory Group for Aerospace Research and Development (AGARD).

AGARD

NATO OTAN

7 RUE ANCELLE • 92200 NEUILLY-SUR-SEINE

FRANCE

Télécopie 0(1)55.61.22.99 • Télex 610 176

DIFFUSION DES PUBLICATIONS

AGARD NON CLASSIFIEES

L'AGARD détient un stock limité de certaines de ses publications récentes. Celles-ci pourront éventuellement être obtenus sous forme de copie papier. Pour de plus amples renseignements concernant l'achat de ces ouvrages, adressez-vous à l'AGARD par lettre ou par télécopie à l'adresse indiquée ci-dessus. *Veuillez ne pas téléphoner.*

Des exemplaires supplémentaires peuvent parfois être obtenus auprès des centres de diffusion nationaux indiqués ci-dessous. Si vous souhaitez recevoir toutes les publications de l'AGARD, ou simplement celles qui concernent certains Panels, vous pouvez demander d'être inclus sur la liste d'envoi de l'un de ces centres.

Les publications de l'AGARD sont en vente auprès des agences de vente indiquées ci-dessous, sous forme de photocopie ou de microfiche. Certains originaux peuvent également être obtenus auprès de CASI.

CENTRES DE DIFFUSION NATIONAUX

ALLEMAGNE
Fachinformationszentrum Karlsruhe
D-76344 Eggenstein-Leopoldshafen 2

BELGIQUE
Coordonnateur AGARD-VSL
Etat-major de la Force aérienne
Quartier Reine Elisabeth
Rue d'Evere, 1140 Bruxelles

CANADA
Directeur - Gestion de l'information
(Recherche et développement) - DRDGI 3
Ministère de la Défense nationale
Ottawa, Ontario K1A 0K2

DANEMARK
Danish Defence Research Establishment
Ryvangs Allé 1
P.O. Box 2715
DK-2100 Copenhagen Ø

ESPAGNE
INTA (AGARD Publications)
Carretera de Torrejón a Ajalvir, Pk.4
28850 Torrejón de Ardoz - Madrid

ETATS-UNIS
NASA Center for AeroSpace Information (CASI)
800 Elkridge Landing Road
Linthicum Heights, MD 21090-2934

FRANCE
O.N.E.R.A. (Direction)
29, Avenue de la Division Leclerc
92322 Châtillon Cedex

GRECE
Hellenic Air Force
Air War College
Scientific and Technical Library
Dekelia Air Force Base
Dekelia, Athens TGA 1010

ISLANDE
Director of Aviation
c/o Flugrad
Reykjavik

ITALIE
Aeronautica Militare
Ufficio del Delegato Nazionale all'AGARD
Aeroporto Pratica di Mare
00040 Pomezia (Roma)

LUXEMBOURG
Voir Belgique

NORVEGE
Norwegian Defence Research Establishment
Attn: Biblioteket
P.O. Box 25
N-2007 Kjeller

PAYS-BAS
Netherlands Delegation to AGARD
National Aerospace Laboratory NLR
P.O. Box 90502
1006 BM Amsterdam

PORTUGAL
Estado Maior da Força Aérea
SDFA - Centro de Documentação
Alfragide
2700 Amadora

ROYAUME-UNI
Defence Research Information Centre
Kentigern House
65 Brown Street
Glasgow G2 8EX

TURQUIE
Millî Savunma Başkanliği (MSB)
ARGE Dairesi Başkanliği (MSB)
06650 Bakanliklar-Ankara

AGENCES DE VENTE

NASA Center for AeroSpace Information (CASI)
800 Elkridge Landing Road
Linthicum Heights, MD 21090-2934
Etats-Unis

The British Library Document Supply Division
Boston Spa, Wetherby
West Yorkshire LS23 7BQ
Royaume-Uni

Les demandes de microfiches ou de photocopies de documents AGARD (y compris les demandes faites auprès du CASI) doivent comporter la dénomination AGARD, ainsi que le numéro de série d'AGARD (par exemple AGARD-AG-315). Des informations analogues, telles que le titre et la date de publication sont souhaitables. Veuiller noter qu'il y a lieu de spécifier AGARD-R-nnn et AGARD-AR-nnn lors de la commande des rapports AGARD et des rapports consultatifs AGARD respectivement. Des références bibliographiques complètes ainsi que des résumés des publications AGARD figurent dans les journaux suivants:

Scientific and Technical Aerospace Reports (STAR)
STAR peut être consulté en ligne au localisateur de ressources uniformes (URL) suivant:
http://www.sti.nasa.gov/Pubs/star/Star.html
STAR est édité par CASI dans le cadre du programme NASA d'information scientifique et technique (STI)
STI Program Office, MS 157A
NASA Langley Research Center
Hampton, Virginia 23681-0001
Etats-Unis

Government Reports Announcements & Index (GRA&I)
publié par le National Technical Information Service
Springfield
Virginia 2216
Etats-Unis
(accessible également en mode interactif dans la base de données bibliographiques en ligne du NTIS, et sur CD-ROM)

Imprimé par le Groupe Communication Canada Inc.
(membre de la Corporation St-Joseph)
45, boul. Sacré-Cœur, Hull (Québec), Canada K1A 0S7

AGARD

NATO OTAN

7 RUE ANCELLE • 92200 NEUILLY-SUR-SEINE

FRANCE

Telefax 0(1)55.61.22.99 • Telex 610 176

DISTRIBUTION OF UNCLASSIFIED

AGARD PUBLICATIONS

AGARD holds limited quantities of some of its recent publications, and these may be available for purchase in hard copy form. For more information, write or send a telefax to the address given above. *Please do not telephone.*

Further copies are sometimes available from the National Distribution Centres listed below. If you wish to receive all AGARD publications, or just those relating to one or more specific AGARD Panels, they may be willing to include you (or your organisation) in their distribution.

AGARD publications may be purchased from the Sales Agencies listed below, in photocopy or microfiche form. Original copies of some publications may be available from CASI.

NATIONAL DISTRIBUTION CENTRES

BELGIUM
Coordonnateur AGARD — VSL
Etat-major de la Force aérienne
Quartier Reine Elisabeth
Rue d'Evere, 1140 Bruxelles

CANADA
Director Research & Development
Information Management - DRDIM 3
Dept of National Defence
Ottawa, Ontario K1A 0K2

DENMARK
Danish Defence Research Establishment
Ryvangs Allé 1
P.O. Box 2715
DK-2100 Copenhagen Ø

FRANCE
O.N.E.R.A. (Direction)
29 Avenue de la Division Leclerc
92322 Châtillon Cedex

GERMANY
Fachinformationszentrum Karlsruhe
D-76344 Eggenstein-Leopoldshafen 2

GREECE
Hellenic Air Force
Air War College
Scientific and Technical Library
Dekelia Air Force Base
Dekelia, Athens TGA 1010

ICELAND
Director of Aviation
c/o Flugrad
Reykjavik

ITALY
Aeronautica Militare
Ufficio del Delegato Nazionale all'AGARD
Aeroporto Pratica di Mare
00040 Pomezia (Roma)

LUXEMBOURG
See Belgium

NETHERLANDS
Netherlands Delegation to AGARD
National Aerospace Laboratory, NLR
P.O. Box 90502
1006 BM Amsterdam

NORWAY
Norwegian Defence Research Establishment
Attn: Biblioteket
P.O. Box 25
N-2007 Kjeller

PORTUGAL
Estado Maior da Força Aérea
SDFA - Centro de Documentação
Alfragide
2700 Amadora

SPAIN
INTA (AGARD Publications)
Carretera de Torrejón a Ajalvir, Pk.4
28850 Torrejón de Ardoz - Madrid

TURKEY
Millî Savunma Başkanliği (MSB)
ARGE Dairesi Başkanliği (MSB)
06650 Bakanliklar-Ankara

UNITED KINGDOM
Defence Research Information Centre
Kentigern House
65 Brown Street
Glasgow G2 8EX

UNITED STATES
NASA Center for AeroSpace Information (CASI)
800 Elkridge Landing Road
Linthicum Heights, MD 21090-2934

SALES AGENCIES

NASA Center for AeroSpace Information (CASI)
800 Elkridge Landing Road
Linthicum Heights, MD 21090-2934
United States

The British Library Document Supply Centre
Boston Spa, Wetherby
West Yorkshire LS23 7BQ
United Kingdom

Requests for microfiches or photocopies of AGARD documents (including requests to CASI) should include the word 'AGARD' and the AGARD serial number (for example AGARD-AG-315). Collateral information such as title and publication date is desirable. Note that AGARD Reports and Advisory Reports should be specified as AGARD-R-nnn and AGARD-AR-nnn, respectively. Full bibliographical references and abstracts of AGARD publications are given in the following journals:

Scientific and Technical Aerospace Reports (STAR)
STAR is available on-line at the following uniform resource locator:
http://www.sti.nasa.gov/Pubs/star/Star.html
STAR is published by CASI for the NASA Scientific and Technical Information (STI) Program
STI Program Office, MS 157A
NASA Langley Research Center
Hampton, Virginia 23681-0001
United States

Government Reports Announcements & Index (GRA&I)
published by the National Technical Information Service
Springfield
Virginia 22161
United States
(also available online in the NTIS Bibliographic Database or on CD-ROM)

Printed by Canada Communication Group Inc.
(A St. Joseph Corporation Company)
45 Sacré-Cœur Blvd., Hull (Québec), Canada K1A 0S7

ISBN 92-836-1061-X