PAGE 1

Form Approved
OMB No 0704 01

(2)

1 AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3 REPORT TYPE AND DATES COVERED
		FINAL/30 SEP 89 TO 29 DEC 92

4 TITLE AND SUBTITLE
ARTIFICIAL INTELLIGENCE METHODOLOGIES IN FLIGHT RELATED DIFFERENTIAL GAME, CONTROL AND OPTIMIZATION PROBLEMS

5. FUNDING NUMBERS
3484/D7
AFOSR-89-0518

6. AUTHOR(S)
DR GUNZBURGER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
WASHINGTON UNIVERSITY
DEPT OF SYSTEMS SCIENCE AND MATHEMATICS
CAMPUS BOX 1040
ONE BROOKINGS DRIVE
ST. LOUIS, MO 63130-4899

8. PERFORMING ORGANIZATION REPORT NUMBER
AFOSR-[illegible] 0487

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
AFOSR/NM
110 DUNCAN AVE, SUTE B115
BOLLING AFB DC 20332-0001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFOSR-89-0518

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE: DISTRIBUTION IS UNLIMITED

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

Artificial intelligence methodologies have been applied to the modeling and implementation of control systems and differential games problems. To be more specific, artificial neural networks, a multiple instruction multiple data parallel processor tuned by connection weights, are used to model a control system or used as an identifier/controller which functions as a mapping between two information domains. Significant advances have been achieved in applying differential games theory to practical problems.

14 SUBJECT TERMS			15 NUMBER OF PAGES 151
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT	18 SECURITY CLASSIFICATION OF THIS PAGE	19 SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAR(SAME AS REPORT)

FINAL TECHNICAL REPORT
(September 30, 1989 - September 29, 1992; [with extension, -December 31,1992])

Submitted to
Air Force Office of Scientific Research
Building 410, Bolling AFB, DC 20332

by

Ervin Y. Rodin, P.I.
Professor and Director

Center for Optimization and Semantic Control
Department of Systems Science and Mathematics
Campus Box 1040, Washington University
One Brookings Drive
St. Louis, MO 63130-4899

in connection with

Grant AFOSR 89-0518:

ARTIFICIAL INTELLIGENCE METHODOLOGIES IN FLIGHT RELATED DIFFERENTIAL GAME, CONTROL AND OPTIMIZATION PROBLEMS

January 31, 1993

FINAL TECHNICAL REPORT

EXECUTIVE SUMMARY

This is the final report for grant AFOSR-89-0518. While its principal portion is the last part of this report, it may be appropriate to summarize first the achievements of the past three years. We begin by reprinting the

1. 1989-90 Annual Technical Report Executive Summary; and the

2. 1990-91 Annual Technical Report Executive Summary.

The above two items give an overview of our activities and achievements during the first two years of this grant.

The main part of the current report is entitled

3. Artificial Intelligence Methodologies for Aerospace and Other Control Systems;

it is the culmination of a project by the Principal Investigator and one of his students. The major subjects addressed in this report are the following:

3a. Neural Networks Approach to Control Systems;

3b. Differential Games with Neural Networks;

3c. Aircraft Control in the Presence of Windshear;

3d. Optimal Control in a Layered Defense System.

Concerning our other activities in the course of the year: we attempted to report on most of them by letter to AFOSR, as they occurred (Copies of the letters enclosed). Examples are presentations given by our group and publications, of which we have already provided copies to AFOSR. We list these here in chronological order:

4. Teaching Neural Networks Nuclear Physics;
(an extensive undergraduate project)

5. System Identification with Dynamic Neural Networks;
(preprint)

6. Control and Disturbance Rejection with a Dynamic Neurocontroller;
(preprint)

7. Maneuver Prediction in Air Combat Via Artificial Neural Networks;
(reprint)

8. Adjacency of the 0-1 Knapsack Problem;
(reprint)

9. On Differential Games With Neural Networks;
(Proceedings of an AFOSR Workshop)

10. Character Recognition: Qualitative Reasoning and Neural Networks;
(reprint)

11. Collision Avoidance and Low-Observable Navigation in a Dynamic Environment;
(reprint)

12. An Optimization Algorithm with Probabilistic Estimation;
(preprint)

There were two particular activities with which we were involved extensively in the course of the past year:

13. Organizing several reciprocal visits, presentations and discussions between our group and the Analysis Group at HQ MAC, Scott AFB;

14. Co-sponsoring "ANNIE '92": an international conference on artificial neural networks in engineering.

Finally, we also established a working relationship with three St. Louis area companies:

ESCO Corporation:
(Collaborating with them to develop an AI assisted and PC based unarmed aircraft defense system)

United Vanlines:
(Helping them to develop optimal routing and scheduling algorithms)

St. Louis Post Dispatch:
(Attempting to apply the AI technology developed by us to the communication needs of the 21st century)

ANNUAL TECHNICAL REPORT EXECUTIVE SUMMARY

The principal portion of this Annual Technical Report is a work by the Principal Investigator and one of his students:

Semantic Control in Continuous Systems:
Applications to Aerospace Problems.

This report discusses our new methodology for dealing with time dependent control and optimization problems; and, in particular, its application to combat path planning in the presence of multiple opposing radar coverage, with time dependent scheduling problems and with flight and fire control via logic programming.

Preceding this report we are presenting a brief discussion, intended to explain and justify why we decided to broaden our original proposed aims and begin to consider

Stochastic Optimization Problems.

The importance of this extension seems particularly evident in the light of the tasks and missions of

Desert Sword .

In the course of the reporting period we also submitted to AFOSR copies of the writeups of two additional projects that we have completed. We are attaching copies of the relevant covering letters here. These consisted of the following:

Flight and Fire Control with Logic Programming
by Ervin Y. Rodin and D. Geist; Comp. and Math. with Applications,
Vol. 20, No. 9/10, pp. 15-27, 1990.

Methods for Stochastic Optimization
by Di Yan and H. Mukai; a Technical Report by the
Center for Optimization and Semantic Control.

We also transmitted copies of the doctoral dissertation of another student of the Principal Investigator. While that person was not supported by this grant, and his work then appeared to have no relevance to the project at hand, we felt that since the Artificial Intelligence methodologies employed in that dissertation were derived from ours, it may be appropriate to present those results to the AFOSR. Now, however, with the threat of large scale Iraqui sabotage of Middle Eastern oil fields a possibility, that dissertation may become very relevant indeed:

Acidic Deposition Control Through an Artificial Intelligence Method
by Ji-Shing Lin.

We were also proud to report in the course of the past year that one of our graduate students, Kevin Ruland, who has been involved with our research projects for two years now, was awarded the very prestigious

Mercury Seven Fellowship.

An additional item of possible relevance here is that the P.I. was elected in the course of the reporting period to be an Associate Fellow of the

American Institute of Aeronautics and Astronautics,

and also a member of the Advisory Committee of its St. Louis chapter.

Finally, we are glad to report that our direct contacts and collaboration with various elements of the

United States Air Force

have been increasing and it seems that our group is becoming progressively more useful to them. In this regard, we can list the following accomplishments for the period of this report:

1. Several working visits by USAF/MAC personnel at our facilities; and several visits by us at **Scott Air Force Base,** in order to discuss reserch problems and results attained by us. (See attached letter by Col. J.D. Graham, and the page after it.)

2. Volunteer Service Agreement between Scott AFB and Washington University, as proposed and implemented by the P.I. under this Grant.

3. A Washington University - MAC Intern Program's description.

4. The nomination of the P.I., Dr. Ervin Y. Rodin, to membership in the AF Scientific Advisory Board, by Lt. General A.J. Burschnick.

5. We also list here the Scott AFB/MAC operational projects on which we are currently working:

Closure Optimization
Defense Courier Service
Aeromedical Evacuation

6. Finally, we should also mention in this section that we are making excellent progress on the development of integer constraint relaxation paradigms, which will be particularly useful for the KORBX computer of Scott AFB

Finally, we should mention here that, in addition to our previous good working relationship with Rockwell International, we have also developed close contacts and mutual interests with McDonnell Douglas and with Emerson Electric. Scientists and engineers from these companies now regularly visit with us: one such group visit took place in conjunction with our full-day presentation for visitors from

HQ, Strategic Air Command.

We are attaching a one-page informational sheet about that also.

ANNUAL TECHNICAL REPORT EXECUTIVE SUMMARY

This is the annual report for the second year of our current three-year grant: thus, several of our major projects are in the midst of being developed. It may be appropriate, therefore, to begin this report with brief descriptions of those projects that we expect to conclude in the course of the coming year. There are actually three such projects, each of which will become a doctoral dissertation under the guidance of the Principal Investigator:

1. Polyhedral Computations For Many-To-Many Routing Problems; Applications To Air Transport;

2. Artificial Intelligence Methodologies In Control Systems;

3. System Identification With Dynamic Neural Networks.

We begin our report by providing brief descriptions of the current status of our research for each of the above subjects.

Several of our projects have their genesis in our collaborative efforts with the CINCMAC Analysis Group of HQ/MAC at Scott AFB. The formal arrangement of this collaboration was set out in a Volunteer Service Agreement between 375 MSSQ/MSCS Scott AFB and Washington University, the details of which were included in our annual report last year. During this past year 2 groups of two senior students each, and one group of three students performed studies relating to the following MAC problems:

1. Defense Courier Routing Problem;

2. Closure Optimization;

3. Operational Support Aircraft Vehicle Scheduling Problem.

We presented these reports to the technical staff of the CINCMAC Analysis Group in the course of one of our regular meetings with them; in fact, we also presented to them an entire written report about the first two of these, with copies also provided to AFOSR. For this reason, we are including in this report only the first few pages of that transmittal. However, we are attaching here a copy of the third and shortest report, which was not submitted to the AFOSR.

It may be appropriate to mention that a fourth senior student group was also working on a project related to this grant (but not related to our Scott AFB oriented work). Only the cover page of their report, entitled

Situation Assessment In Medium Range Air Combat,

is included in this Annual Report.

Several of our research results were prepared in the course of this past year for publication. We are including some of these in this report. The publication on

i. Adjacency of the 0-1 Knapsack Problem,

is a byproduct of our work on *Semantic Control In Continuous Systems: Applications To Aerospace Problems*,, which was presented in last year's Annual Report. The next item,

ii. Differential Games and Neural Nets,

was developed jointly with McDonnell Douglas Missiles Systems Company scientists, and it is an ongoing project and collaborative effort.

Our attempts to utilize neural networks in control, optimization and differential game type problems led us to the realization, that much more powerful, self-tuning networks of this type should be developed. This led to our first report on

iii. Neural Networks With Local Memory For Control Systems,

which is our next enclosure.

We reported last year on our work on *Tactical Air Combat Maneuvers: Recognition And Guidance Via Neural Networks.* This past year we attempted to utilize that same technology, to identify the output of an arbitrary "black box". A first step in that direction was our next included item, consisting of our work on

iv. Character Recognition: A New Approach Using Neural Networks.

The last item in this section is in fact the longest one: a detailed report on the

v. Application of Semantic Control To A Class Of Pursuer-Evader Problems.

This was a project which we undertook jointly with scientists from the ESCO Corporation. From our point of view, the importance of the research here was in proving the feasibility of creating a rule based expert system, which is capable of calling on exact optimization algorithms as subroutines, and which can be implemented on small computers. This is still an ongoing project: we expect to provide further results in next year's report. (Note: we are not including here the lengthy appendices to this work, which consist of detailed computer listings.)

Washington
WASHINGTON UNIVERSITY IN ST. LOUIS

Center for Optimization and Semantic Control

Friday, January 31, 1992

Dr. Neal Glassman
AFOSR/NM
Bldg. 410, Bolling AFB
Washington, DC 20332-6448

Dear Neal:

As I may have mentioned to you in the past, I am always trying to get as many undergraduates as possible to get involved in our various research projects. Some of these result in nice outcomes, and some are just so-so.

During the past year I encouraged one such undergraduate to try his hand at using neural nets for a physics related problem. Since his results were pretty nice, I decided to send you and Arje a few copies, enclosed here.

With belated good wishes for the new year and best personal regards,

Sincerely yours,

Ervin Y. Rodin
Professor and
Director, COSC

enc.: 3 copies of *Teaching Neural Networks Nuclear Physics*

Washington University
Campus Box 1040
St. Louis, Missouri 63139-4899
Tel: (314) 889-6007, -5806
FAX: (314) 726-4434

Washington
WASHINGTON UNIVERSITY IN ST. LOUIS

Center for Optimization and Semantic Control

Monday, June 1, 1992

Dr. Arje Nachman
AFOSR/NM
Bldg. 410, Bolling AFB
Washington, DC 20332-6448

Dear Arje:

Since a portion of our research under our present grant involves the tuning and utilization of neural networks, and since we have made some nice strides in that direction, we decided to publish some of our results related to this area. So, to bring this to your early attention, I am sending you attached two preprints from our Center:

1. *System Identification With Dynamic Neural Networks;* and
2. *Control and Disturbance Rejection With A Dynamic Neurocontroller.*

With best regards,

Sincerely yours,

Ervin Y. Rodin
Professor and
Director, COSC

enc.: 3 copies each of 1. and 2. above

cc. Dr. Neal Glassman

Washington University
Campus Box 1040
St. Louis, Missouri 63139-4899
Tel: (314) 889-6007, -5806
FAX: (314) 726-4434

Washington
WASHINGTON UNIVERSITY IN ST. LOUIS

Center for Optimization and Semantic Control

Monday, July 6, 1992

Dr. Arje Nachman
AFOSR/NM
Bldg. 410, Bolling AFB
Washington, DC 20332-6448

Dear Arje:

I am sending you attached three copies of another paper, for which support by both AFOSR 870252 and AFOSR 890158 was acknowledged:

Mancuver Prediction In Air combat Via Artificial Neural Networks,

by myself and S. M. Amin.

With best regards,

Sincerely yours,

Ervin Y. Rodin
Professor and
Director, COSC

enc.: 3 reprints

cc. Dr. Neal Glassman

Washington University
Campus Box 1040
St. Louis, Missouri 63139-4899
Tel: (314) 889-6007, -5806
FAX: (314) 726-4434

Washington
WASHINGTON UNIVERSITY IN ST. LOUIS

Center for Optimization and Semantic Control

Wednesday, August 26, 1992

Dr. Arje Nachman
AFOSR/NM
Bldg. 410, Bolling AFB
Washington, DC 20332-6448

Dear Arje:

I am sending you attached three copies of a paper, for which support by AFOSR 890158 was acknowledged:

Adjacency of the 0-1 Knapsack Problem,

by D. Geist and myself.

With best regards,

Sincerely yours,

Ervin Y. Rodin
Professor and
Director, COSC

enc.: 3 reprints

cc. Dr. Neal Glassman

Washington University
Campus Box 1040
St. Louis, Missouri 63139-4899
Tel: (314) 889-6007, -5806
FAX: (314) 726-4434

Washington
WASHINGTON UNIVERSITY IN ST LOUIS

Center for Optimization and Semantic Control

Tuesday, September 8, 1992

Dr. Neal Glassman
AFOSR/NM
Bldg. 410, Bolling AFB
Washington, DC 20332-6448

Dear Neal;

My apologies for responding to your request late; however, I was out of town when the messages arrived. So now here is the information for the period requested:

Publications:

"On Differential Games With Neural Networks" with Y. Wu), AFOSR Workshop of Theory and Applications of Nonlinear Control, St. Louis, MO, 1991.

"Character Recognition: Qualitative Reasoning and Neural Networks" (with Y. Wu and S. M. Amin), Math. and Comp. Modelling, Vol 16, No. 2, pp. 95-104, 1992.

"Collision Avoidance And Low-Observable Navigation In A Dynamic Environment" (with S.M. Amin and C. Ruan), Math. and Comp. Modelling Vol 16, No. 5, pp. 77-98, 1992.

Graduate Students Supported:

Kevin Ruland; Michael Meusey; James Revetta; Mark Monical.

Undergraduate Students Supported:

Travis Cusick.

Postdoctoral Associates Supported:

S. Massoud Amin.

External Honors Received:

Elected Associate Fellow of the American Institute of Aeronautics and Astronautics.

I hope this will meet your requirements.

Best regards,,

Ervin Y. Rodin
Professor and
Director, COSC

Washington University
Campus Box 1040
St. Louis, Missouri 63130-4899
Tel (314) 889-6007, -5800

Washington
WASHINGTON UNIVERSITY IN ST LOUIS

Center for Optimization and Semantic Control

Wednesday, November 25, 1992

Dr. Arje Nachman
AFOSR/NM
Bldg. 410, Bolling AFB
Washington, DC 20332-6448

Dear Arje:

I am sending you attached two copies of a report, for which support by AFOSR 890158 was acknowledged:

An Optimization Algorithm With Probabilistic Estimation,

by D. Yan and H. Mukai. A slightly different version of the report will also appear in the Journal of Optimization Theory and Applications.

With best regards,

Sincerely yours,

Ervin Y. Rodin
Professor and
Director, COSC

Washington University
Campus Box 1040
St. Louis, Missouri 63149-4899
Tel (314) 889-6007, -5806
FAX (314) 726-4434

Washington
WASHINGTON UNIVERSITY IN ST. LOUIS

Center for Optimization and Semantic Control

Tuesday, September 22, 1992

Dr. Neal Glassman
AFOSR/NM
Bldg. 410, Bolling AFB
Washington, DC 20332-6448

Dear Neal:

I am sending you attached the program for ANNIE '92, a conference in which we are involved. Our Center is a Sponsor of the Conference; my colleague, M. Amin and I are on the Organizing Committee; I am chairing a session and we are presenting two papers. Both of these papers carry acknowledgement of support by AFOSR.

With best regards,

Sincerely yours,

Ervin Y. Rodin
Professor and
Director, COSC

enc.: 1 program

cc. Dr. A. Nachman

Washington University
Campus Box 1040
St. Louis, Missouri 63139-4899
Tel: (314) 889-6007, -5806
FAX: (314) 726-4434

TECHNICAL REPORT

Center for Optimization and Semantic Control

P.O.Box 1040
WASHINGTON UNIVERSITY
St. Louis, Missouri 63130-4899

ARTIFICIAL INTELLIGENCE METHODOLOGIES FOR AEROSPACE AND OTHER CONTROL SYSTEMS

by

Ervin Y. Rodin

and

Yuanlan Wu

January, 1993

Another version of this report was submitted to the Sever Institute of Technology by the second author, under the direction of the first author, in partial fulfillment of his requirements for the degree of Doctor of Science.

ABSTRACT

ARTIFICIAL INTELLIGENCE METHODOLOGIES FOR AEROSPACE AND OTHER CONTROL SYSTEMS

Artificial intelligence methodologies have been applied to the modeling and implementation of control systems and differential games problems. To be more specific, artificial neural networks, a multiple instruction multiple data parallel processor tuned by connection weights, are used to model a control system or used as an identifier/controller which functions as a mapping between two information domains. Based on a new paradigm of neural networks consisting of Neurons With Local Memory (NLMs), the representation of a control system by neural networks is discussed. Using this representation, the basic issues of complete controllability and observability for the system are addressed. A separation principle of learning and control is presented for Networks with NLMs (NNLM). The result shows that the weights of the network will not affect its dynamics. The principle may be utilized to prespecify the steady state properties of the system. Modeled by NNLM, the resulting system is a typical nonlinear one which, through rigorous mathematical analysis, is shown to be locally linearizable via a regular static state feedback and a nonlinear coordinate transformation.

Significant advances have been achieved in applying differential games theory, a theory dealing with most of conflicts in daily life, economics, military affairs, etc., to practical problems. In this dissertation, this theory has been thoroughly addressed from a new point of view. A configuration, based on the paradigm of semantic control, is proposed, which can be used to derive two paradigms of differential games with neural networks. Generally, two neural networks are used in each of these two paradigms. One network is called the neural-identifier and it is used to identify the control strategy of one's opponent. The other one is the neural-controller which, taking the estimate of the control strategy of one's opponent, outputs the control value for oneself. The issue of existence of solutions is discussed. To demonstrate the effectiveness of the method, a simulation experiment was carried out and studied for a pursuit-evasion game problem.

In Chapter 3 a learning control algorithm is developed. The algorithm can be used to evaluate the weight of a neural controller in the paradigms proposed in the chapter or in the control systems. Using the learning control algorithm, we study the aircraft control problem in the presence of wind shear.

In Chapter 4 we shall discuss another aspect of artificial intelligence techniques in control systems: rule-based system in a class of pursuit-evasion game problems. The pursuit-evasion game problems can be converted to classical optimal control problems. The optimal control solution is obtained. The solution offers several advantages such as significant time-saving in implementation. Further research directions are addressed in the last chapter.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ARTIFICIAL INTELLIGENCE METHODOLOGIES FOR AEROSPACE AND OTHER CONTROL SYSTEMS

1. Introduction

Much effort has been directed during the past two decades in attempting a merger of the areas of Artificial Intelligence and Automatic Control [4, 24, 56, 79]. Such a merger would combine the rigorous, precise, and analytical foundation of automatic control theory with the heuristic, qualitative and efficient reasoning aspects of artificial intelligence. Such a merger would provide a practical, powerful mechanism and framework and effective computational tools for the modeling and analysis of fuzzy, time-dependent, noisy and uncertain models describing physical phenomena. The theory and applications of such a merger does show the power of the efforts in this area for modeling various real processes [25, 76, 77, 79]. This study attempts to combine the techniques of these two areas to develop new tools, to enhance the analysis of a mature control theory, and to apply these techniques to real-time processes. We begin in the next section to discuss some basic issues concerning the combination of artificial intelligence and automatic control.

1.1. Intelligent Control

In this section, we shall summarize the history, research efforts, and application aspects of intelligent control. In particular, we shall explore its relationship with adaptive control, semantic control [76], more closely related expert control [6], and knowledge-based control systems [87].

Among others, Saridis [79] gave a formal definition for what he termed an *Intelligent Machine*:

Definition 1.1 *Intelligent Machines are machines that are designed to perform anthropomorphic tasks with minimum interaction with a human operator.*

Intelligent control then is the function that drives an intelligent machine. Intelligent control can also be considered as a fusion between mathematical and linguistic methods and algorithms applied to systems and processes. Intelligent control, which is hierarchically distributed, is composed of three basic levels of control: the organization level, the coordination level, and the execution level (see Figure 1.1).

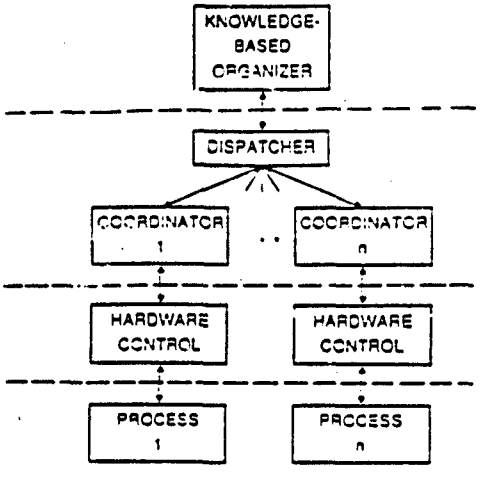


Figure 1.1: Hierarchical Intelligent Control System

The organization level is designed to perform such operations as receiving and reasoning with commands, planning, making high level decisions from long-term memories, providing feedback, and exchanging long-term memory. Probabilistic models are provided for a mechanism so that it can select an appropriate task for a given command. The concepts of commands, task commands, events, activities, random variables associated with events, and functions are then introduced to specify analytically the functions of the organizer. This level, which is designed to imitate functions of human behavior, may be treated as an element of knowledge-based systems. Thus, knowledge representation, knowledge flow, and knowledge processing and management are the main activities on this level.

The coordination level is an interactive structure serving as an interface between the organization and execution level. It formulates the control problems associated with the most probable, complete and compatible plan formulated in the organization levels. Several individual coordinators are associated with specific hardware execution devices. Each of them performs a pre-specified number of different functions. Two types of feedback information exist for each coordinator: [i] off-line feedback information which is fed to the organization level and is stored in long-term memory; and, [ii] on-line or real-time feedback information which is issued by each executor in the execution level, received by the coordinator and stored in short-term memory. The on-line feedback information may also be used by other coordinators for the evaluation of the overall accrued cost of the coordinate level.

The execution level executes the appropriate control functions. In particular, optimal control theory with a non-negative functional of the systems states or

with an entropy H(u) for a particular control action u(x, t) is discussed by Saridis [79]. However, various control schemes may be employed on this level.

To represent the uncertainty which may be present on each of these levels, Saridis introduced the concept of entropy, a probabilistic measure of uncertainty. All levels of a hierarchical intelligent control are measured by entropies and their rates. With the introduction of entropy, the theory of hierarchically intelligent controls may be stated as the following:

The theory of an Intelligent Machine may be postulated as the mathematical problem of finding the right sequence of decisions and controls for a system structured according to the principle of increasing precision with decreasing intelligence (constraint) such that it minimizes its total entropy.

Although Saridis is the first one who has worked on Intelligent Control Theory in a systematic way, and has attempted to to lay a mathematical foundation for the theory, other people have also actively worked on this area. Among these people are Åström [5], Fu [24], Wiener [99] and Meystel [59, 60, 61, 62]. In [5], Åström discussed the issues of intelligent control from a more practical and application-oriented point of view. Primarily aiming at PID Controllers, Automatic Tuning (e.g., relay autotuner), Adaptive Control, and Expert Control, he reviewed briefly the history of automatic control. He explored the realistic issues of practical real-time processes, such as sampling period, model structure, uncertainty, disturbance, and, in favor of PID controller and self-tuning regulator, he discussed the ideas and application areas of Automatic Tuners and Adaptive Controllers. Unlike Saridis, who combined the techniques of AI, Operational Research and conventional Control Theory and treated the issues of intelligent control in a more analytical and systematic way, Åström mainly discussed the problems of an

automatic tuner and adaptive controller, implying that the virtual part of intelligent control lies in the system's capacity to adapt to a time-varying/unknown environment, be convenient to end-users, have an automation of regulator parameter tuning, and assume less prior information for the system to be controlled.

Aithough a common point can be observed for both Åström and Saridis - adaptation to unknown/time-varying environments - Åström emphasizes incorporating more human intelligence into the process controllers as in the position of an instrument engineer while Saridis views the intelligent control as an overall structure of the whole organization as in the position of a Chief Executive Officer. In this sense, the category that Åström discussed as intelligent control falls into what Saridis termed "Executive Level"; however, Åström treated various issues in a more detailed, practical and realistic way.

It is interesting to know that there exists another type of control system capable of intelligence: an Expert Control System (see Figure 1.2). An intelligent control system with the function of supervision and containing a knowledge-base, is categorized by Åström [5, 6] as an Expert Control System. The object of expert control is to encode knowledge representation and decision capabilities to allow for automatic intelligent decisions and recommendations rather than by preprogrammed logic. The development of an expert control system is motivated by the fact that heuristics plays an important role in PID regulators. Thus, a more efficient, robust, yet cruder way of implementing heuristics may be needed. Designing an expert control system, which has the capacity to orchestrate a range of different control algorithms for different control goals, seems to be the right answer. Analogous to an Expert System in the field of Artificial Intelligence, an

expert control system consists of the system data base, the rulebase, the inference engine, the user interface, and the planning process.

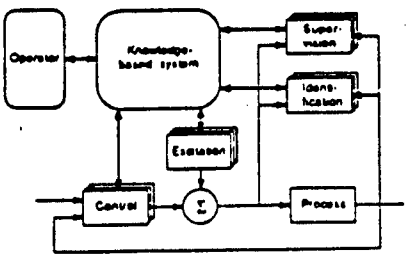


Figure 1.2: Block Diagram of An Expert Control Systems

In an expert control system, the system data base contains constraints on operational sequencing; facts (or static data such as sensor measurements, tolerances, operating thresholds, etc.); evidence (or dynamic data such as sensors, instrument engineering reports, and laboratory and test reports); hypotheses, which are generated and stored in the data base, e.g., various state estimates; and goals (either static goals or dynamic goals: static goals include the wide array of performance objectives; dynamic goals are those established on-line).

The rule-base of an expert control system contains production rules, such as if-then rules. The conditions of the rules are usually facts and hypotheses from the data base while the results of the rules are the actions, such as activation of controllers. The rules may also be viewed as functions operating on the strategies. The inference engine has the same meaning as its definition in the traditional expert system, which functions according to different strategies.

An important element of an expert control system is planning. In view of the difference between the conventional control systems and the expert control systems which deal with a process in a more ambiguous, more qualitative way, the

planning process of an expert control system should be implemented according to this difference. Various algorithms are provided for supervision, analysis and signal generation. An expert control system, which separates the control algorithms from the logic, decides when to use a particular algorithm. The planning may be viewed as an action of search in a logic network, forming a path to reach the goals. Its function involves issuing a command to change the production goals and change the process with its requirements.

Comparison of an expert control system with an autotuner, which is what Åström meant by an intelligent controller, is given by Åström in [5]. Although both schemes can tune the parameters for a conventional controller, e.g. a PID controller, an expert control system usually has a more efficient way of interacting with a human operator because of its supervision functionality, linguistic interaction capacity, and listing capacity. Thus, depending on each individual application problem, one can choose an appropriate scheme of control system structure.

Another effort at combining AI techniques and control theory has been in developing the knowledge-based nested hierarchical controller [62] for the analysis and design of autonomous robots [59]. The structure of a multi-resolutional (pyramidal) nonhomogeneous system of knowledge representation interacting with a planning/control system was introduced by Meystel. A structure of this type gives not only unique capabilities of knowledge representation but also a number of powerful algorithmic capabilities, such as a joint planning/control structure, planning in traversability spaces, minimum-time dynamic navigation, knowledge-based control, and others which are promising for autonomous intelligent machines. This type of controller, which employs joint multi-resolutional planning-control procedures, algorithms of enhanced nested dynamic programming, the

hybrid world representation, and linguistic clauses, has been implemented in an intelligent mobile robot IMAS-2 [59].

Knowledge for the choice of control is represented as a descriptive structure in a fuzzy linguistic representation space (FLR-Space). This structure is obtained in the form of a semantic network from a set of texts. The nodes and the relations in the structure can be evaluated numerically. Time behavior can be associated with the structure, and hence, a function or a sequence $f(t_0), \cdots, f(t_f)$ can be considered as a trajectory of motion starting with the initial state and ending at a fixed state. Although other control schemes can be considered, so far only cost-optimal control processes have been studied in [59]. The control strategies are obtained via a sequence of Hamitonians $H_1 \supset H_2 \supset \cdots \supset H_i$ for each of the levels of the hierarchy. The algorithm of nested dynamic programming provides the major mechanism for obtaining the control strategies. Nonhomogeneous models which are not in the form of a system of algebraic and/or differential equations are used for various reasons[59]. If the analytical model is unknown, one can usually organize a pseudo-analytical model using tabulated data. It seems natural to consider the use of production systems (PS) for matching the linguistic nature of the original world description.

Recent works by Rodin [76] on Semantic Control Theory have been successful in several cases, such as [25, 77]. As another important and unique approach to combining AI techniques and control theory, Semantic Control theory allows: (1) the system to adapt to varying/unknown environments, (2) enhancing human-machine interaction, and (3) for on-line planning/goal selection. A semantic control system usually consists of three parts (see Figure 1.3): [i] Identifier; [ii] Goal

Selector: and [iii] Adaptor. Their functions, when applied to a situation governed by differential games (for instance), are as follows:

(i) Identifier: The Identifier block identifies through sensors and a knowledge base the differential game, parameters, targets (if any) and role of each player.

(ii) Goal Selector: The Goal Selector solves the differential game chosen by the Identifier block. The results are the optimal trajectories, barriers and controls.

(iii) Adaptor: The Adaptor determines the controls that cause each player to "best" follow the optimal trajectory determined by the Goal Selector.

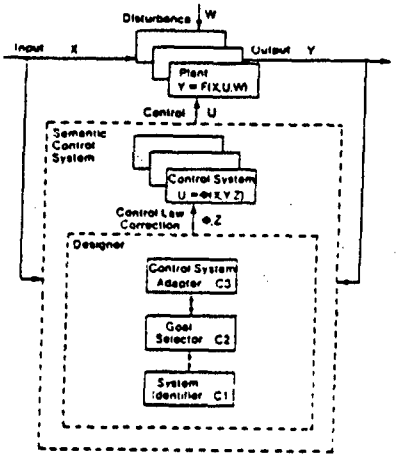


Figure 1.3: Semantic Control Paradigm

Applications of semantic control theory in problems of air-combat, a class of pursuit-evasion game, can be found in [25, 77]. From these applications, one can see that the theory does provide a powerful, fundamental framework and mechanism for modeling a real and complex system as well as provide on-line adaptation to an unknown environment, on-line goal selection and implementation of lower-level execution functions.

From above, we have seen that several intelligent control schemes have been proposed in the area of intelligent control. They are Intelligent Control by Saridis [79], Automatic Tuning by Åström [5], Expert Control Systems by Åström [6, 5], Knowledge-based Neted Hierarchical Controller by Meystel [6-] and Semantic Control Theory by Rodin [76]. These schemes are proposed from different points of view to deal with many complex practical systems. They have been successful in a variety of applications [5, 25, 59, 77, 79]. A common point of these schemes is their capability of adapting to the changing environment. In other words, they have the so-called learning capacity. Thus, it is nature to consider the mechanism to realize this learning capacity. Neural networks for control systems seem to be ideal for such mechanisms. In the next section, more details about the current research efforts in this area will be given.

1.2. Neural Networks for Control Systems

In this section, a brief review and survey is given concerning current works and expected future research trends in the area of neural networks for control systems. The topics to be discussed include the most recent works in this area, different types of neural controllers and various applications of these controllers. They are [i] Learning Controllers, [ii] Recurrent Neural Networks for Control Systems, [iii] Reinforcement Learning Controllers, [iv] Relationship Between Adaptive Controllers and Neural Controllers, [v] Modeling and Identification, and [vi] Cerebellum Model Articulation Controllers. Most of the works focus on applications of neural networks in known/unknown nonlinear systems with/without noise, for the purpose of either control or identification. Although this discussion is far from complete in covering all aspects of the work in this area, it does indeed include

the major trends at the current time. In what follows, we shall discuss different topics separately.

Learning Controller

Most recently, Hoskins *et al* [36] presented an iterative constrained inversion technique to find the control inputs to a plant. Although the nature of their work is similar to the work in [26], several advantages are observed in [36]. First, the proposed controller responds on-line to changes in the plant dynamics. More interestingly, the proposed controller is applied to generate a neural-network-based model reference adaptive controller (NN-MRAC), which is used to control the spring-mass-damper system in which the position respon[illegible]e reference command is the same as a target controller. Second, by removing t[illegible]ural network from the direct feedback path and replacing direct feedback with an estimate and optimization, Hoskins is also the first to attempt to consider the analytical treatment of the stability of the closed-loop system, which is important but has no mature solution in the current literature. Third, he also considered the issue of "Smooth Control". "Smooth control" is generally required in some applications. That is, the control value computed at the current step should not vary too much from the control value at a previous step. This is particularly true in robot control problems. For the redundant robot control problem, one requirement is to avoid abrupt changes of the gesture in response to the slow end-effect movement of the arm. This requirement is not satisfied in previous works applying neural networks for the inverse kinematics problems. Although a two-stage learning strategy may be an answer to this problem, the works by Hoskins and his coworkers did show an advantage in this regard.

Reinforcement Learning

Reinforcement learning is one of the major neural network approaches to learning control [43]. Although these methods originated from studies of animal learning and in early learning control works [58], they have now been an active area of research in neural networks and machine learning. In [43], Sutton, Barto and Williams explained these methods as a synthesis of dynamic programming and stochastic approximation methods and focused their discussion on the Q-learning method which was originally presented in [94] by Watkins. An active-critic learning system contains two distinct subsystems: one to estimate the long-term utility for each state and another to learn to choose the optimal action in each state. A Q-learning system maintains estimates of utilities for all state-action pairs and makes use of those estimates to select actions. They viewed these methods as an example of a direct adaptive optimal control algorithm, i.e. as an on-line Dynamic Programming method and a computationally inexpensive approach to direct adaptive optimal control, which determines the control without first forming a system model.

Recurrent Neural Networks for Control Systems

Also recently, Nikolaou *et al* [73, 74] published their research for identifying and modeling a chemical process. In their work, a recurrent neural network consisting of dynamic neurons whose behavior is governed by the following set of differential equations

$$\frac{dx_i}{dt} = -\frac{x_i}{T_i} + \frac{F_i(\sum_j w_{ij}x_j)}{T_i} + \frac{u_i}{T_i}, \tag{1.1}$$

where i = 1, 2, ..., n, is used to model chemical processes with severe nonlinearity. After the network has been said to approximate the process well enough, a nonlinear controller based on the works of Isidori [38] is used to control the chemical process so that the resulting system is linear, and thus, various synthesis methods for linear systems can be used for the purpose of control. Their approach has been shown to be successful by applying the network to a model and identifying a continuously stirred reactor (CSTR).

Although their work still falls into the category of identifying and modeling a system (process) utilizing the interpolation property of a neural network, one of the unique features of their work lies in using the internal information of the neural networks, namely, using the states of neurons to construct the nonlinear controller. This approach reveals a new aspect of the work in the area for neural networks for control: how to efficiently and effectively make use of the intelligence of the neural networks themselves for control systems or how to utilize the internal information of the network, instead of viewing the network as generic mapping, so that the memory capacity and learning capacity of neural networks can be more fully utilized.

It turns out that, in the current literature on neural networks for control systems, very few people put an emphasis on this point of view. A tremendous amount of work has been done using feedforward neural networks as generic mappings, and then demonstrating that such a mapping, now replaced by the fancier name "neural networks", worked fine for some particular problems [8, 20, 70]. Typical work has been in the inverse kinematics problems [30, 65]. If the plant is known *a priori*, teaching the inverse dynamics of a plant to a feedforward neural network appears easier since the input-output behavior of the plant can be utilized

as teaching signals. However, in contrast to the inverse kinematics problem, in most of constrained control applications, the functional expression of the forward mapping ϕ is generally unknown. In that case, a two-stage learning strategy has been proposed in [21, 42, 46, 70]. In the two-stage learning strategy, two neural networks are used. One is trained to learn the forward mapping or dynamics of a nonlinear system. The other one is trained as a neural controller. The approach seems promising for this kind of problem.

Therefore, the works by Nikolaou *et al* [73, 74] have offered a unique approach in this direction.

Relationship between Adaptive Controllers and Neural Controllers

There is a common point between well-developed Adaptive Controllers and Neural Controllers: adjusting their parameters on-line or recussively. Thus, in many cases, neural controllers are very closely related to adaptive controllers. Due to this reason, it is natural to consider neural controllers and adaptive controllers together and explore their relationship. Among researchers working in this particular area are Narendra[68, 69], Hoskins [36], Chen [14], Karakasoglu [45], Guha [31], Bialasiewicz [11], and Sztipanovits [89]. Narendra explored how well-established adaptive identification and control techniques can be applied to the analysis and synthesis of dynamic systems, which contain neural networks as subsystems. Different combinations of neural networks and linear systems are considered as models for identification and adaptive control. Detailed analysis and discussion about those issues are given by him and Parthasarathy in [69]. Chen [14] used a different approach to neural networks for self-tuning control systems. Two neural networks are used for approximating the nonlinear terms of a NARMAX model. The weights were adjusted such that the error between the output of the actual

plant and the output of the neural network, and the error of the output signal of the plant and the predefined signal tend to be minimized.

Unlike a self-tuning control scheme which usually requires *a priori* information such as process model order, deadtime, and disturbance characteristics as well as the assumption of linearity of process, a neural controller has the advantage that it usually does not require *a priori* information about the process to be controlled. Comparisons have been made between the two schemes in [14, 48, 50, 91], and attempts have been made to combine techniques in these two areas [36, 48].

Modeling and Identification

There are many neural networks applications for modeling and identifying nonlinear systems [1, 15, 28, 51, 74, 85, 86]. Most of the work on neural networks for identification and modeling has been in using the property of universal approximation of feedforward networks (e.g., [16, 35, 88, 98]). A typical scheme is to use the error between the output of the network and the output of the unknown system to update the connection weights of the network at each step. Various optimization methods may be employed to reduce the output error by adjusting the interconnection weights. Among them are the gradient descent algorithm, the conjugate gradient algorithm and Davidson's algorithm. Depending on how the weights are updated, there are two different schemes for training the neural networks: Pattern Learning [78, 97] and Batch Learning [32, 96]. Pattern learning is the method in which the weights of the network are adapted immediately after each pattern is fed in. The other method, however, takes all the data as a whole batch, and the network is not updated until the entire batch of data is processed. Qin *et al* [75] discussed the relationship between Pattern Learning and Batch Learning for dynamic system identification. Four basic learning methods

have been used for their work, depending on the schemes of the system identification using neural networks. In [69], Narendra and Parthasarathy discussed the use of neural networks for dynamical system identification and control. *Generalized Neural Networks* have been proposed, which are various combinations of linear dynamic systems and feedforward networks. Chen *et al* [15] have developed a *prediction error algorithm* for system identification, in which the networks are primarily used as universal approximations for nonlinear systems.

A unique approach has been employed by Specht in [85]. A one-pass neural network learning algorithm similar to [84] has been used to estimate continuous variables. Depending on the variables used, the networks can be utilized for prediction, modeling, mapping, and interpolation, or as a controller. Specht discussed the memory-based network that provides estimates of continuous variables and converges to the underlying (linear or nonlinear) regression surface. This network, called *General Regression Neural Network* (GRNN), is a one-pass learning algorithm with a highly parallel structure. Thus, the network features fast learning that does not require an iterative procedure and a highly parallel structure. Among the advantages of GRNN, the network "learns" in one pass through the data and can generalize from examples as soon as they are stored.

Although most of the work in this direction is based on the property of universal approximation of feedforward neural networks, several specific neural network architectures have been used: [i] Feedforward Neural Networks (e.g., [1, 15, 51]); [ii] GRNN [85]; and [iii] Recurrent Neural Networks (e.g., [74]). Different architectures of neural networks find their use for various purposes of applications. For example, the neural networks proposed by Nikolaou *et al* have the advantage that

the internal variables can be readily used for constructing a linearizing controller such that the overall system is linearized.

Likewise, the neural network approach for identifying and modeling nonlinear systems has the advantage that no *a priori* information about the model structure is needed. Important works for modeling and identification using neural networks can be also found in [93, 97].

Cerebellum Model Articulation Controller (CMAC)

Another type of neural network for control systems which is worthy of mentioning is the so-called Cerebellum Model Articulation Controllers (CMACs) [67]. It was invented in 1975 by James Albus [2], then with the National Bureau of Standards. Albus's scheme was based on a model of human memory and human neuromuscular-control principles. The term Cerebellum Model Articulation Controller, or CMAC, is often interpreted to mean Cerebellar Arithmetic Computer. CMACs were originally developed for robot control, and they have been popularized by a group at the robotics laboratory of electrical and computer engineering at the University of New Hampshire under the direction of W. Thomas Miller III.

CMACs enjoy the reputation of having a much faster training time (several orders of magnitude) than Feedforward Neural Networks (FFNNs) trained by backpropagation [13], yet give the same performance as FFNNs. This property is particularly useful for real-time learning and control problems, e.g. in an adaptive flight-control system. CMAC neural networks are aiso capable of effectively organizing and implementing a multi-dimensional function approximation in a computationally efficient manner using traditional computing architectures.

A unique approach to neural network controller design has been employed by Kraft and Campagna [50] to study the performance of this type of controller in a nonlinear system corrupted with noise. The basic idea behind their work is to generate an approximation to a characteristic system surface from input-output measurements and then use the surface as feedforward information to calculate the appropriate control signal. The characteristic system surface is, in fact, the system equation representing the known/unknown plant to be controlled. If the values of the system parameters were known, the surface could be precalculated and stored in memory. Then, given the control objective (i.e. the desired position in memory), it would be possible to look up in memory the correct control signal. When the system parameters are unknown, the surface must be "learned" from input-output data in real time. The controller, similar to the work of Miller [66], uses a memory update algorithm which updates the values of a group of memory locations near a selected memory cell during each control cycle, using the concept of generalization.

Kraft and Campagna [50] compared this type of neural network controller with two traditional adaptive control systems: Self-tuning Regulator and Model Reference Adaptive Controllers (MRACs) in a study of the behavior of a first-order system with/without nonlinearity, presented with/without noise. Results showed that the CMAC neural controller performed equally well in the presence of noise, and worked extremely well for a nonlinear system, compared with the two traditional adaptive controllers. Although, unlike MRACs, this controller has no guarantee for stability analysis, implementation speed comparisons favored the neural network approach because the control signal can be generated virtually as a table look-up procedure. Moreover, with the neural network controller approach, no *a priori* information about the system to be controlled is needed. Thus, the

neural network controller is suitable for a wide class of nonlinear systems. Above all, their results indeed reveal some interesting aspects of neural network approach for control systems.

More recently, using B-Spline receptive field functions in conjunction with more general CMAC weight addressing, Lane *et al* [52] developed higher-order CMAC neural networks that can learn both functions and function derivatives. The number of weights addressed in computing a network output grow exponentially with the number of input dimensions. Back-propagation BMACs with higher-order reception field functions on only selected network inputs and Spline-Net network architectures were proposed as potential solutions to problems of more modest size, producing piecewise linear and additive function approximations.

1.3. Organization of the Dissertation

The purpose of this study is to model and to analyze control systems aided by neural networks. The approaches attempt to explore use of the features of parallel architecture in control systems. It is organized into five chapters.

The second chapter is a study in modeling control systems using neural networks which have a highly parallel structure and are capable of learning and storing information. The study is in the spirit of fully utilizing the intelligence of the networks and the pattern of processing information in parallel inside the networks. We go beyond using the universal approximation property of neural networks, and also consider the internal state information of the recurrent neural networks so that a control system can be modeled using this highly parallel structure of computation mechanism. Based on a new paradigm of neural networks

consisting of Neurons With Local Memory (NLMs), the representation of a control system by neural networks is discussed. Using this representation, the basic issues of complete controllability and observability for the system are addressed. A separation principle of learning and control is presented for NNLM. The result shows that the weights of the network will not affect its dynamics. The principle may be utilized to prespecify the steady state properties of the system. Modeled by NNLM, the resulting system is a typical nonlinear one that, through mathematical analysis, can be shown to be locally linearizable via a regular static feedback and a nonlinear coordinate transformation. Although theorectical results in Chapter 2 are not directly used in Chapter 3, they do have potential applications for the differential game problems. For example, pursuit-evasion games can be modeled by NNLMs while controllers can then be designed using various techniques.

The third chapter of the dissertation is to develop another new paradigm and tools for applying neural network techniques in traditional differential game problems. During the past few years, attempts have been made to utilize the powerful qualitative reasoning and heuristic search capacities in the area of artificial intelligence to overcome the difficulties of applying differential game theory in practical problems, such as cumbersome computations [77, 82, 95]. A configuration, based on the paradigm of Semantic Control, is proposed. It can be used to derive two paradigms of differential games with neural networks. Two neural networks are used in each of these two settings. One network is called the neural-identifier which is used to identify the control strategy of the opposing player. The other one is the neural-controller which, taking the estimate of the control of the other player, outputs the real control value for its own player. The issue of existence of solution is discussed. To demonstrate the effectiveness of the method, a simulation experiment is carried out and studied for a pursuit-evasion problem. In

this chapter, a learning control algorithm is developed. The algorithm can be used to evaluated the weights of a neural controller in the paradigms proposed in the chapter or in other control systems. Using the learning control algorithm, we study the aircraft control problem in the presence of windshear.

The fourth chapter is a study of optimal control and optimization problems in the Layered Defense Project. The Layered Defense Project is a cooperative effort between the Center for Optimization and Semantic Control at Washington University and the Electronics and Space Corporation. Based on the semantic control theory, the project is to model and study a class of pursuit-evasion game problems. The third part of this dissertation discusses the optimal control problems arising from the project. Classical line-of-sight coordinates are employed to model the game situation. Based on a similar study in [18], an optimal control law was derived for the one-pursuer and one-evader case. A non-derivative optimization method is used for finding the optimal initial costates for the optimal control law.

The fifth chapter summarizes our work. The main contributions of this dissertation are enumerated and future research directions are presented in this chapter.

2. Neural Networks Approach to Control Systems

This chapter presents new approach to neural networks for control systems. Based on a paradigm of neural networks - NNLM - consisting of neurons with local memory, a control system represented by neural networks is discussed. With this representation, the basic issues of complete controllability and observability for the system are addressed. For the first time, a separation principle of learning and control is presented for NNLM, and the principle shows that the weights of the network will not affect its dynamics. Because of the nonlinearity of the network, it is natural to consider the issue of linearization around a local equilibrium point. A detailed and rigorous analysis of the local linearization via a regular static feedback and a nonlinear coordinate transformation is given in the final section.

2.1. Background

This beginning section will briefly recall three types of neurons commonly used in feed-forward and recurrent neural networks. They are McCulloch-Pitts neurons, Grossberg's neurons and Hopfield's neurons. The well-known McCulloch-Pitts neurons, which take the weighted sum of inputs and give the output through a transfer function, are the basic elements in feedforward neural networks. They have been widely and successfully used, and their structure is well known. Although there are various architectures to connect the neurons (see the structures in backpropagation networks, Kohonen networks and Hopfield networks), the basic elements — the neurons — remain the same.

When he studied the famous dog-saliva-food biological phenomenon, Grossberg proposed a new type of learning rule, known as the Grossberg Learning Law, as well as a new type of neuron in order to attempt to mathematically formulate Hebb's law. His approach, in turn, attempted to explain the classical conditioning behaviors discovered by Pavlov. Since we shall not discuss the learning law in detail, interested readers are referred to [47].

The neurons proposed by Grossberg are not simply of the McCulloch-Pitts type, as their outputs are described by a different set of equations. Consider a neuron which has a number of inputs coming from other neurons in the network, as well as an external input coming from outside the network. The following equation describes the dynamics of the ith neuron

$$\frac{dy_i(t)}{dt} = -ay_i(t) + I_i(t) + \sum_{\substack{j=1 \\ j \neq i}}^{n} w_j y_j(t) \tag{2.1}$$

where $y_i(t)$ is the output of the ith neuron, $I_i(t)$ is an external input to the ith neuron, and w_j is the weight connecting the output of the ith neuron to the input of some other neuron. The difference between the McCulloch-Pitts neurons and those proposed by Grossberg is clear since "dynamics" are incorporated in each of the Grossberg neurons. These dynamics are represented by a positive constant a which controls the decay of the output in the absence of any other input. Thus, a may also be called a forgetting factor. This type of neuron, together with Grossberg's learning law, give a plausible mathematical formulation for Hebb's law and thus form a satisfactory connection with Hebb's learning theories.

Later (in 1984), John Hopfield proposed a general structure for a continuous deterministic model. This structure is known as the Hopfield model. A Hopfield model is a two-layer network in which the neurons in the hidden layer are fully

connected to each other. The input-output relationship of the ith neuron in the network, realized by an amplifier, is described by the set of nonlinear dynamic equations

$$\begin{aligned} C_i \frac{du_i}{dt} &= \sum_{j \neq i} T_{ij} v_j - \frac{u_i}{\tau_i} + I_i \\ v_i &= g_i(u_i) \end{aligned} \tag{2.2}$$

where C_i is the total input capacitance of the amplifier, T_{ij} is the strength of the connection from the output of the jth amplifier to the input of the ith amplifier, u_i is the input to the ith amplifier, and v_j is the output of the jth amplifier. Also, τ_i is a resistance value, I_i is the sigmoidal transfer function of the ith amplifier, and g_i is the sigmoidal transfer function of the ith amplifier, assuming a negligible response time. A commonly known property of the Hopfield network is that the state of the network can be attracted to an equilibrium point corresponding to a local minimum of the energy function and hence the network can be used to implement a content addressable memory. Based on this property, Hopfield networks have been used satisfactorily for traveling salesman problems [33, 34], for an A/D converter [90], signal decomposition [90], linear programming [90], and various combinatorial optimization problems [90].

As we shall see in subsequent sections, the neurons introduced below are different from McCulloch-Pitts neurons, Grossberg neurons and Hopfield neurons. In some sense, they are closest to the neurons in the Hopfield model as they can be viewed as a discrete-time version of the neurons in the Hopfield model. But unlike those in the Hopfield model, these neurons are used in a feedforward network in which the well-known backpropagation algorithm can be employed to change the weights. More importantly, they are used here in a novel attempt to represent

a control system by a neural network. In fact, the idea of representing the internal states as a state vector is not new. Hopfield used the same idea when he used state vector to construct an energy function in his network. He proved, by using Liapunov stability theory, that the state will eventually converge to a local equilibrium in state space, which corresponds to a local minimum of the energy function. The neurons proposed below are used for representing a control system and are the basic elements for a feedforward network which, unlike the Hopfield network, does not have an equilibrium.

2.2. Neurons with Local Memory (NLM)

The term, Neurons with Local Memory (NLM), comes from the presence of dynamics inside each of the neurons we are interested in. The incorporation of dynamics inside each neuron is the main distinction between these neurons and the conventional McCulloch-Pitts neurons. As we shall see below, this type of neural network facilitates much of the subsequent analysis of neural networks for control systems. The incorporation of dynamics in each neuron results in the flow of outputs from neurons even without any inputs. Thus, the NLMs may also be termed dynamical neurons or active neurons.

Interestingly, a similar idea has been used by Nikolaou *et al* in [73, 74] to identify the dynamics of a continuously stirred reactor (CSTR). In their work, Nikolaou *et al* used a neural network whose neurons have the following set of differential equations

$$\frac{dx_i}{dt} = -\frac{x_i}{T_i} + \frac{F_i(\sum_j w_{ij}x_j)}{T_i} + \frac{u}{T_i}, \qquad (2.3)$$

for i = 1, 2, ..., n. Although their work has been successful in identifying the dynamics of CSTR, they have not discussed basic issues of a control system such

as controllability and observability. In this study, we shall discuss the basic issues of control systems associated with this type of neural networks in a more analytical and systematic way.

A typical representation of an input-output relationship for the conventional McCulloch-Pitts neurons is written as

$$y_k^j = f^j(u_k^{1,j}, ..., u_k^{n_j,j}), \qquad k \in Z, \tag{2.4}$$

where the y's and u's are the outputs and the inputs respectively. Also, Z is the set of positive integers, the subscript k denotes the time step k, and the superscript j denotes the jth neuron. A typical form for f^j of (2.4) can be written as

$$y_k^j = s_j(\sum_{i=1}^{n_j} w_{ji} u_k^{ij}), \tag{2.5}$$

where s_j is a sigmoidal function, and the w_{ij}'s are the synaptic weights.

A basic structure of an NLM is shown in Figure 2.1, where j denotes the jth neuron. The quantities $y_k^j, u_k^{1,j}, ..., u_k^{n_j,j}$ are the output and inputs to the neuron at time step k, respectively. Also, z^{-1} denotes the backshift operator and s_j^{-1} denotes the inverse of the transfer function for the neuron j.

The output y_k^j of an NLM can be written as

$$y_k^j = s_j(a^j s_j^{-1}(y_{k-1}^j) + c^j \sum_{i=1}^{n_j} w_{ji} u_k^{i,j}), \tag{2.6}$$

where a^j is a scalar whose value represents the dynamics in neuron j, c^j is another scalar, and the w_{ji}'s are the weights of connection from other neurons to neuron j. By setting $a^j = 0$ and $c^j = 1$, we immediately obtain the conventional input-output relationship for the McCulloch-Pitts neurons. It follows that the input-output relationship of a conventional neuron is actually a special case of that of NLM.

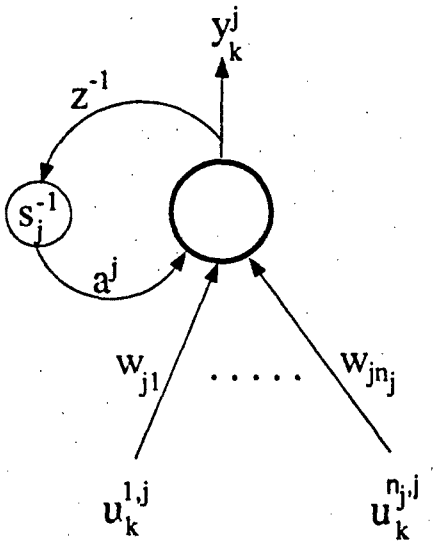


Figure 2.1: Basic Structure of a NLM

An alternative and more informative input-output representation of an NLM can be given by introducing an internal state variable x_k

$$\begin{aligned} x_k^j &= a^j x_{k-1}^j + \sum_{i=1}^{n_j} w_{ji} u_k^{i,j}, \\ y_k^j &= s_j(c^j x_k^j), \qquad\qquad k \in Z, \end{aligned} \tag{2.7}$$

from which (2.6) can be derived easily. The system equation (2.7) is called the node system. Again setting $a^j = 0$ and $c^j = 1$ in (2.7), we obtain the input-output relationship of a conventional neuron.

The advantage of the representation (2.7) over the representation (2.6) is apparent by introducing the internal state x_k^j. The system (2.7) actually has the standard state equation and output equation familiar to control engineers. For convenience, we still adopt the same name, "state equation", for the x equation. The role of a^j in (2.7) is clear from the familiar control theory. For example, a necessary condition for the node system to be asymptotically stable is that the a^j's lie inside the unit disc in the complex plane. Even though the state

equation in (2.7) is linear and time-invariant, the output equation is nonlinear, which complicates further analysis. Although we may assume that s^j is linear, which is the case in part of our following analysis, we shall generally consider s_j to be nonlinear, e.g. a commonly used sigmoidal function.

2.3. Networks with NLMs (NNLM)

Having defined the basic structure for NLM in the previous section, we can now construct a neural network whose elements are NLMs. We shall denote the NNLM with m inputs, n hidden nodes and p outputs by $N_{m,n,p}$. For simplicity, we only consider the single-input and single-output (SISO) system in this section. The generalization to the multi-input and multi-output (MIMO) system is straightforward. Meanwhile, the input to the network has generally arbitrary values.

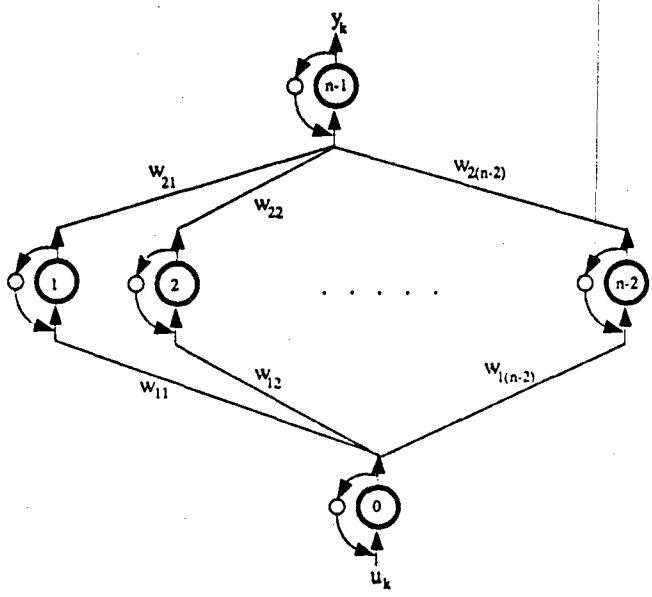


Figure 2.2: General Structure of NNLM

A general structure for NNLM is shown in Figure 2.2 . The state equations are:

$$\text{node } 0: \begin{cases} x_k^0 &= a^0 x_{k-1}^0 + u_k, \\ y_k^0 &= s_0(c^0 x_k^0), \end{cases}$$

$$\text{node 1, ..., node n-2}: \begin{cases} x_k^i &= a^i x_{k-1}^i + w_{1i} y_k^0, \\ y_k^i &= s_2(c^i x_k^i), \qquad i = 1, 2, \ldots, n-2, \end{cases}$$

$$\text{node n-1}: \begin{cases} x_k^{n-1} &= a^{n-1} x_{k-1}^{n-1} + \sum_{i=1}^{n-2} w_{2i} y_k^i, \\ y_k &= s_3(c^{n-1} x_k^{n-1}), \end{cases}$$

where the a^is are scalars representing the dynamics of the ith node system, the s_j's are the transfer functions, which are generally sigmoidal functions, and the w_{ij}'s are the synaptic weights for the path connecting adjacent layers.

Assuming for a moment that the transfer functions s_0, s_2 and s_3 are all linear and defining the state-variable vector $\mathbf{x}_k^T$ by $\mathbf{x}_k^T = [x_k^0, \ldots, x_k^{n-1}]$, we can represent the node system in a more concise form by

$$\begin{aligned} \mathbf{x}_k &= \mathbf{A}\mathbf{x}_{k-1} + \mathbf{B}u_k, \\ y_k &= \mathbf{C}\mathbf{x}_k, \end{aligned} \tag{2.8}$$

where

$$\mathbf{A} = \begin{bmatrix} a^0 & 0 & 0 & \cdots & 0 & 0 \\ w_{11}c^0a^0 & a^1 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ w_{1(n-2)}c^0a^0 & 0 & 0 & \cdots & a^{n-2} & 0 \\ a^0\tilde{a} & w_{21}c^1a^1 & w_{22}c^2a^2 & \cdots & w_{2(n-2)}c^{n-2}a^{n-2} & a^{n-1} \end{bmatrix},$$

$$\mathbf{B}^{\mathbf{T}} = \begin{bmatrix} 1 & w_{11}c^0 & \cdots & w_{1(n-2)}c^0 & \tilde{a} \end{bmatrix},$$

$$\mathbf{C} = \begin{bmatrix} 0 & 0 & \cdots & 0 & c^{n-1} \end{bmatrix},$$

$$\hat{a} = \sum_{i=1}^{n-2} w_{1i} w_{2i} c^0 c^i.$$

Equation (2.8) represents a linear state and output equation with the transfer matrix being a lower-triangular one. By assigning $a^i (for 0 \leq i \leq n-1)$ in A, we can alter the dynamics in (2.8). Assuming that $a^{n-1} \neq a^i$ for i = 0, ..., n-2, we define the quantity a_c as follows

$$a_c = \sum_{i=1}^{n-2} w_{1i} w_{2i} \frac{c^i}{a^{n-1} - a^i}. \tag{2.9}$$

The quantity a_c plays a key role in our subsequent discussions. As mentioned in the begining of this section, $N_{m,n,p}$ denotes the NNLM with m inputs, n hidden nodes and p outputs. Based on the analysis on controllability and observability in the next section, we immediately have the following:

Theorem 2.1 *Suppose that*

[i] *All transfer functions s_j are linear.*

[ii] $w_{ij} \neq 0$ *for all i,j.*

[iii] $c^i \neq 0$ *for all i.*

[iv] $a_c \neq 0$ *and* $a_c < \infty$.

[v] $a^i \neq a^j$ *for* $i \neq j$.

Then, any strictly proper SISO linear system with real and nonrepeating eigenvalues can be realized by $N_{1,n-2,1}$, where the a^is are the eigenvalues of the system.

Proof: Because the system (2.8) is completely controllable and observable (see theorems 2.2 and 2.3 in the next section), the transfer fu... on $\mathbf{C}(sI - \mathbf{A})^{-1}\mathbf{B}$

has no pole-zero cancellation between its numerator and denominator. Since the order of the denominator polynomial is n, it represents a typical nth-order rational transfer function.

Q.E.D.

We give an example for the case n=4. The matrices **A** and **B** in this case are

$$\mathbf{A} = \begin{bmatrix} a^0 & 0 & 0 & 0 \\ w_{11}c^0a^0 & a^1 & 0 & 0 \\ w_{12}c^0a^0 & 0 & a^2 & 0 \\ a^0\tilde{a} & w_{21}c^1a^1 & w_{22}c^2a^2 & a^3 \end{bmatrix}, \mathbf{B} = \begin{bmatrix} 1 \\ w_{11}c^0 \\ w_{12}c^0 \\ \tilde{a} \end{bmatrix},$$

$$\mathbf{C} = \begin{bmatrix} 0 & 0 & 0 & c^3 \end{bmatrix},$$

where $\tilde{a} = w_{21}c^1w_{11}c^0 + w_{22}c^2w_{12}c^0$ and the transfer function is

$$\frac{c^3(b_3s^3 + b_2s^2 + b_1s + b_0)}{(s-a^0)(s-a^1)(s-a^2)(s-a^3)}, \tag{2.10}$$

and

$$\begin{aligned} b_3 &= \tilde{a}. \\ b_2 &= -\tilde{a}(a^0 + a^1 + a^2) + a^2c^0c^2w_{12}w_{22} + a^1c^0c^1w_{11}w_{21} + a^0\tilde{a}. \\ b_1 &= \tilde{a}(a^0a^2 + a^0a^1 + a^1a^2) - (a^0 + a^1)a^2c^0c^2w_{12}w_{22} - \\ &\quad (a^0 + a^2)a^1c^0c^1w_{11}w_{21} - a^0a^1c^0c^2w_{12}w_{22} - a^0a^2\tilde{a}. \\ b_0 &= -\tilde{a}a^0a^1a^2 + a^0a^1a^2c^0c^2w_{12}w_{22} + a^0a^1a^2c^0c^1w_{11}w_{21} + \\ &\quad a^0a^1a^2c^0c^2w_{12}w_{22}. \end{aligned}$$

Thus, by properly choosing $w_{11}, w_{12}, w_{21}, w_{22}$, we can realize a 4th order linear system. A block diagram for the realization is shown in Figure 2.3.

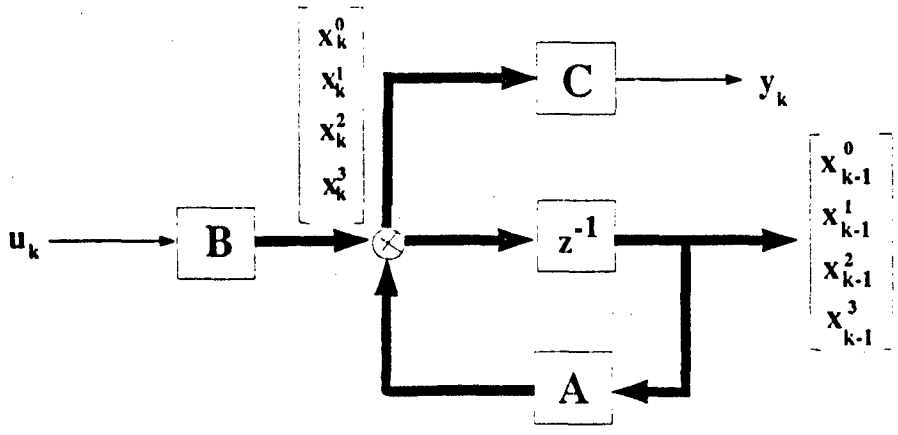


Figure 2.3: Linear System Representation of Order 4

2.4. Controllability and Observability

The basic issues of controllability and observability for the system (2.8) will be discussed in this section. For the definitions of controllability and observability, interested readers may refer to [44]. We have the following:

Theorem 2.2 *Suppose that*

[i] $w_{ij} \neq 0$ *for all* i,j,

[ii] $c^i \neq 0$ *for all* i,

[iii] $a_c \neq 0$ *and* $a_c < \infty$.

Then, the system (2.8) is completely controllable if and only if the following inequalities hold

$$a^i \neq a^j \quad for \quad i \neq j, \qquad i,j = 1,2,....,n-1. \tag{2.11}$$

Proof: (**Sufficiency**) We shall prove sufficiency by using the Popov-Belevitch-Hautus rank test (see [44]). Let A_1 be defined as follows

$$A_1 = [sI - A \quad B] = \begin{bmatrix} s-a^0 & 0 & 0 & \dots & 0 & 0 \\ -w_{11}c^0a^0 & s-a^1 & 0 & \dots & 0 & 0 \\ -w_{12}c^0a^0 & 0 & s-a^2 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ -w_{1(n-2)}c^0a^0 & 0 & 0 & \dots & s-a^{n-2} & 0 \\ -a^0\tilde{a} & -w_{21}c^1a^1 & -w_{22}c^2a^2 & \dots & -w_{2(n-2)}c^{n-2}a^{n-2} & s-a^{n-1} \end{bmatrix}$$

$$\begin{bmatrix} 0 & 1 \\ 0 & w_{11}c^0 \\ 0 & w_{12}c^0 \\ \vdots & \vdots \\ 0 & w_{1(n-2)}c^0 \\ s-a^{n-1} & \tilde{a} \end{bmatrix}.$$

Obviously A_1 has rank n if s is not an eigenvalue of A_1. For $s = a^0$ and if $a^0 \neq a^i$ for $i \geq 1$, multiplying the last column by a^0 and adding it to the 1st column yields

$$A_2 = \begin{bmatrix} a^0 & 0 & 0 & \dots & 0 & 1 \\ 0 & a^0-a^1 & 0 & \dots & 0 & w_{11}c^0 \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \ddots & 0 & w_{1(n-2)}c^0 \\ 0 & -w_{21}c^1a^1 & -w_{22}c^2a^2 & \dots & a^0-a^{n-1} & \tilde{a} \end{bmatrix},$$

which has rank n.

For $s = a^0$ and if $a^0 = a^i$ for some i, deleting the nth column of matrix A_1 yields a matrix A_3:

$$A_3 = \begin{bmatrix} 0 & 0 & 0 & \dots & 0 & 1 \\ -w_{11}c^0a^0 & a^0-a^1 & 0 & \dots & 0 & w_{11}c^0 \\ -w_{12}c^0a^0 & 0 & a^0-a^2 & \dots & 0 & w_{12}c^0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ -w_{1(n-2)}c^0a^0 & 0 & 0 & \dots & a^0-a^{n-2} & w_{1(n-2)}c^0 \\ -a^0\tilde{a} & -w_{21}c^1a^1 & -w_{22}c^2a^2 & \dots & -w_{2(n-2)}c^{n-2}a^{n-2} & \tilde{a} \end{bmatrix}.$$

After elementary transformations are performed on the matrix A_3, its last row becomes [0, 0, ... 0, *, 0, ..., 0]. Then, performing another series of elementary

transformations on the resulting matrix yields the following

$$A_4 = \begin{bmatrix} a^0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & a^0 - a^1 & \ddots & \vdots & \vdots & \vdots & \cdots & \vdots & \vdots \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & \cdots & a^0 - a^{i-1} & \vdots & 0 & \cdots & 0 & * \\ 0 & \cdots & \cdots & 0 & \vdots & a^0 - a^{i+1} & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & a^0 - a^{n-2} & 0 \\ 0 & 0 & \cdots & 0 & * & 0 & \cdots & 0 & 0 \end{bmatrix},$$

which has rank n.

For $s = a^i (1 \leq i \leq n-2)$, deleting the nth column of A_1 yields

$$A_5 = \begin{bmatrix} a^i & 0 & 0 & \cdots & 0 \\ 0 & a^i - a^1 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \vdots & \vdots & \vdots & a^i - a^{i-1} & 0 \\ \vdots & \vdots & \vdots & 0 & 0 \\ \vdots & \vdots & \vdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & -w_{21}c^1a^1 & -w_{22}c^2a^2 & \cdots & -w_{2(i-1)}c^{i-1}a^{i-1} \end{bmatrix}$$

$$\begin{bmatrix} 0 & 0 & \cdots & 0 & 1 \\ 0 & 0 & \cdots & 0 & w_{11}c^0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \vdots & \vdots & \vdots & w_{1i}c^0 \\ 0 & \ddots & \vdots & \vdots & w_{1(i+1)}c^0 \\ a^i - a^{i+1} & \vdots & \ddots & \vdots & w_{1(i+2)}c^0 \\ \vdots & \vdots & \vdots & a^i - a^{n-2} & \vdots \\ -w_{2i}c^ia^i & -w_{2(i+1)}c^{i+1}a^{i+1} & \cdots & -w_{2(n-2)}c^{n-2}a^{n-2} & \tilde{a} \end{bmatrix}.$$

After performing a series of elementary transformations, it is not hard to show that the rank of the matrix is again n.

For the case $s = a^{n-1}$, deleting the nth column of A_1 and performing one elementary transformation on A_1 yields

$$A_6 = \begin{bmatrix} a^{n-1} & 0 & 0 & \dots & 0 & 1 \\ 0 & a^{n-1} - a^1 & 0 & \dots & 0 & w_{11}c^0 \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \ddots & a^{n-1} - a^{n-2} & w_{1(n-2)}c^0 \\ 0 & -w_{21}c^1a^1 & -w_{22}c^2a^2 & \dots & -w_{2(n-2)}c^{n-2}a^{n-2} & \tilde{a} \end{bmatrix}.$$

Again, performing another series of elementary transformations on A_6, one obtains

$$A_7 = \begin{bmatrix} a^{n-1} & 0 & 0 & \dots & 0 & 1 \\ 0 & a^{n-1} - a^1 & 0 & \dots & 0 & w_{11}c^0 \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \ddots & a^{n-1} - a^{n-2} & w_{1(n-2)}c^0 \\ 0 & 0 & 0 & \dots & 0 & \tilde{a}_c \end{bmatrix}.$$

where $\tilde{a}_c$ is

$$\tilde{a}_c = a_c c^0 a^{n-1}. \tag{2.12}$$

Therefore, A_7 has also rank n.

(Necessity) Necessity is proved by contradiction. Letting $a^i = a^j$ for some $i \neq j$, i, j=1, 2, ..., n-1 yields a matrix $A^{\#}$ which has rank less than n.

Q.E.D.

REMARKS:

It is easy to see from the proof that the condition $a^0 = a^i (1 \leq i \leq n-1)$ is allowed. Thus the system is still controllable even for repeated eigenvalues $a^0 = a^i$ for some i between 1 and n-1. Notice that $a_{n-1} \neq a_i$ for i = 0, ... n-2 is only a sufficient

condition for the theorem. The following example shows that the assumption may not be necessary.

Considering a case where n = 2, we have the state equations

$$x_k^0 = a^0 x_{k-1}^0 + u_k, \tag{2.13}$$

$$x_k^1 = a^1 x_{k-1}^1 + w_{11} c^0 x_{k-1}^0, \tag{2.14}$$

$$y_k^1 = c^1 x_k^1. \tag{2.15}$$

This system is obviously controllable no matter what the values of a^0 and a^1 are. Thus, letting $a^0 = a^1 =$ constant, we still have a controllable system.

A similar result regarding the observability of the system is obtained.

Theorem 2.3 *Suppose that*

[i] $w_{ij} \neq 0$ *for all i,j,*

[ii] $c^i \neq 0$ *for all i,*

[iii] $a_c \neq 0$ *and* $a_c < \infty$.

Then, the system (2.8) is completely observable if and only if the following inequalities hold

$$a^i \neq a^j \quad for \quad i \neq j. \qquad i, j = 1, 2, \ldots, n-1. \tag{2.16}$$

Proof: Necessity and sufficiency can be proved again by using the Popov-Belevitch-Hautus rank test, namely, by checking the rank of the matrix $[C^T \quad (sI - A)^T]^T$. Similar arguments lead to the conclusion of this theorem, with the only difference being that the column transformations are changed to corresponding row transformations.

Q.E.D.

REMARK:

The result on observability of the systems holds only under the assumption that the transfer functions of the node system are all linear.

2.5. Separation of Learning and Control

In this Section, we shall discuss the effects of the weights on the overall performance of the system. In Section 2.3, we showed that some of the entries of matrices A and B in (2.8) contain the weights of the network. This seems to imply that the weights could affect the dynamics of the system. However, this turns out not to be the case. In fact, the transfer function (2.10) tells us a very important fact that the weights of the network will only affect the numerator of the system and do not affect the eigenvalues of the system will not be affected. In general, we have the following

$$\text{Transfer Function} \;=\; \frac{d(s;\mathbf{w},\mathbf{a},\mathbf{c})}{\prod_{i=0}^{n-1}(s-a_i)}, \tag{2.17}$$

where $\mathbf{w} = (w_{ij})_{n\times n}$, $\mathbf{a} = (a^0,...,a^{n-1})$, $\mathbf{c} = (c^0,c^1,...,c^{n-1})$ and d(s; **w**, **a**, **c**) is a polynomial of order n-1 whose coefficients are the linear combination of entries of matrices **w**, **a** and **c**. This property will be formally stated as follows:

Property 2.1 *The dynamics of the system will not be affected by changing the weights of the network.*

Based on this property and the fact that the NLMs are extensions of the McCulloch-Pitts neurons, we obtain the Separation Principle of Learning and Control, stated below. The importance of this principle lies in the fact that before we actually use the system, we can set all a^i's to be zero. We then train the

network using the backpropagation algorithm with a prespecified training set so that the network has the desired stationary property. After training is done, the parameters a^i can be resumed and thus the network will function as a normal system.

SEPARATION PRINCIPLE OF LEARNING AND CONTROL

The training process of an NNLM and the control process after training can be separated.

2.6. Linearization via Transformation of Coordinates and Nonlinear Feedback

In Section 2.3, we saw that our representation resulted in a nonlinear discrete-time system. There are many reasons for linearizing a nonlinear system, and many publications in the literature [29, 53, 54, 72] discuss this problem. Before we proceed, let us look at our discrete-time system whose nonlinearity arises from the nonlinear transfer functions. In general, the transfer functions in input nodes of a neural network are linear. Thus, the state-space description of our system has the form

$$\begin{aligned}
x_k^0 &= a^0 x_{k-1}^0 + u_k, \\
x_k^1 &= w_{11} c^0 a^0 x_{k-1}^0 + a^1 x_{k-1}^1 + w_{11} c^0 u_k, \\
&\vdots \\
x_k^{n-2} &= w_{1(n-2)} c^0 a^0 x_{k-1}^0 + a^{n-2} x_{k-1}^{n-2} + w_{1(n-2)} c^0 u_k, \\
x_k^{n-1} &= a^{n-1} x_{k-1}^{n-1} + \sum_{i=1}^{n-2} w_{2i} s_2(c^i a^i x_{k-1}^i + c^i w_{1i} c^0 a^0 x_{k-1}^0 + c^i w_{1i} c^0 u_k), \\
y_k &= s_3(c^{n-1} x_k^{n-1}).
\end{aligned} \tag{2.18}$$

The above equations can be written in the following form:

$$\mathbf{x_k} = \mathbf{f}(\mathbf{x_{k-1}}, \mathbf{u_k}), \tag{2.19}$$

where $\mathbf{x_k} = (x_k^0, \cdots, x_k^{n-1})$ and $\mathbf{f}(\mathbf{x_{k-1}}, \mathbf{u_k}) = (f_0(\mathbf{x_{k-1}}, \mathbf{u_k}), \cdots, f_{n-1}(\mathbf{x_{k-1}}, \mathbf{u_k}))$ is a vector of the equations which are defined above.

From the above, we know that the overall system consists of a linear sub-system cascaded by a nonlinear subsystem together with a nonlinear output equation (see Figure 2.4). This in turn implies that the overall system is a nonlinear one.

It is natural to consider the problem of locally linearizing the above system via coordinate transformations and nonlinear feedback. In general, not all nonlinear systems can be so linearized. A necessary and sufficient condition will be given in Section 2.6.2. Once a linearized system is obtained, it is very easy to implement a nonlinear control law to have the system track some desired signal.

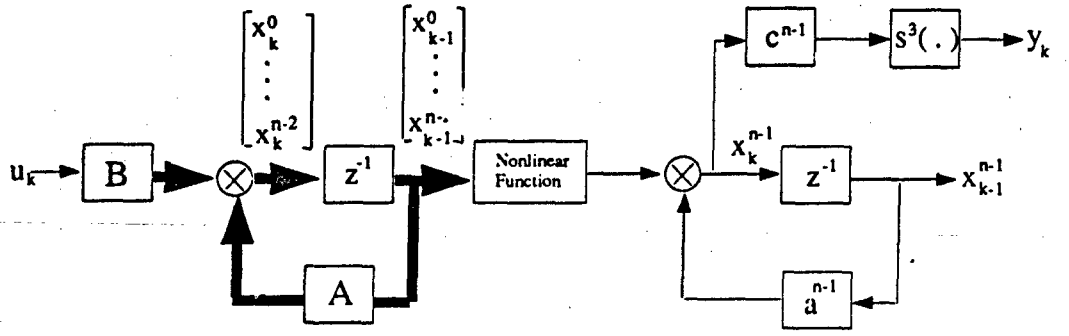


Figure 2.4: Nonlinear System Representation of Order n

2.6.1. Preliminary

We consider a smooth discrete-time nonlinear dynamic system

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_{k+1}), \tag{2.20}$$

where $\mathbf{x}_k = (x_k^0, x_k^1, ..., x_k^{n-1})$ and $\mathbf{u}_k = (u_k^0, u_k^1, ..., u_k^{m-1})$ are smooth local coordinates for the state-space M and input space U respectively. Before discussing feedback linearizability for (2.20), we introduce the notion of a regular static state feedback. We call a relation

$$\mathbf{u}_{k+1} = \alpha(\mathbf{x}_k, \mathbf{v}_{k+1}), \tag{2.21}$$

a regular static state feedback whenever $\frac{\partial \alpha}{\partial v}(\mathbf{x}_k, \mathbf{v}_{k+1})$ is nonsingular at every point $(\mathbf{x}_k, \mathbf{v}_{k+1})$. Notice that this implies locally a one-to-one relation between the old inputs $\mathbf{u}_{k+1}$ and the new controls $\mathbf{v}_{k+1}$. We can now formulate the notion of feedback linearizability for (2.20):

Definition 1:

Let $(\mathbf{x_0}, \mathbf{u_0})$ be an equilibrium point for (2.20), i.e. $\mathbf{x_0} = f(\mathbf{x_0}, \mathbf{u_0})$. The system (2.20) is feedback linearizable around $(\mathbf{x_0}, \mathbf{u_0})$ if there exists

(a) A coordinate transformation $S : V \in R^n \rightarrow S(V) \subset R^n$ defined on a neighborhood V of $\mathbf{x_0}$ with $S(\mathbf{x_0}) = \mathbf{0}$;

(b) A regular feedback $\mathbf{u} = \alpha(\mathbf{x}, \mathbf{v})$ satisfying $\alpha(\mathbf{x_0}, \mathbf{0}) = \mathbf{u_0}$ and defined on a neighborhood $V \times O$ of $(\mathbf{x_0}, \mathbf{0})$ with $\frac{\partial \alpha}{\partial v}(\mathbf{x}, \mathbf{v})$ nonsingular on $V \times O$,

such that in the new coordinates $\mathbf{z} = S(\mathbf{x})$ the closed loop dynamics are linear

$$\mathbf{z}(k+1) = A\mathbf{z}(k) + B\mathbf{v}(k), \tag{2.22}$$

for some matrices A and B.

At this point, let us look at the equilibrium points of our nonlinear system. For the system (2.20) it is not hard to show that the $\mathbf{x}^*, \mathbf{u}^*$ satisfying $f(\mathbf{x}^*, \mathbf{u}^*) = \mathbf{x}^*$

have the form

$$\begin{aligned} x^{0*} &= \frac{u^*}{1-a^0}, \\ x^{1*} &= \frac{1}{1-a^1}[w_{11}c^0a^0x^{0*} + w_{11}c^0u^*], \\ &\vdots \\ x^{(n-1)*} &= \frac{1}{1-a^{n-1}}[\sum_{i=1}^{n-2} w_{2i}s_2(c^ia^ix^{i*} + c^iw_{1i}c^0a^0x^{0*} + c^iw_{1i}c^0u^*)]. \quad (2.23) \end{aligned}$$

Therefore, $x^{0*}, x^{1*}, ..., x^{(n-2)*}$ are all linear functions of u^* but $x^{(n-1)*}$ is not.

2.6.2. Necessary and Sufficient Conditions for Local Linearization via Transformation of Coordinates and Nonlinear Feedback

In this section, we are going to use Grizzle's necessary and sufficient conditions [72] to prove that our nonlinear system is locally linearizable to a controllable linear system. Before we formally give the result in the next section, let us look at a sequence of distributions given by Grizzle in [72]. This sequence will be instrumental in the solution of the feedback linearization problem for (2.20).

Let $\pi : M \times U \rightarrow M$ be the canonical projection and K the distribution defined by

$$K = \ker f_*. \quad (2.24)$$

where $M \subset R^n$, $U \subset R^m$ and f_* is the dual vector space homomorphism from $TM \times TU$ to TM.

Algorithm 2.1

Assume f_* has full rank around $(\mathbf{x_0}, \mathbf{u_0})$.

Step 0: Define the distribution D_0 in a neighborhood of $(\mathbf{x_0}, \mathbf{u_0})$ in $M \times U$ by

$$D_0 = \pi_*^{-1}(0), \tag{2.25}$$

Step i+1: Suppose that around $(\mathbf{x_0}, \mathbf{u_0})$ $D_i + K$ is an involutive constant dimensional distribution on $T(M \times U)$. Then define in a neighborhood of $(\mathbf{x_0}, \mathbf{u_0})$

$$D_{i+1} = \pi_*^{-1} f_*(D_i), \tag{2.26}$$

and stop if $D_i + K$ is not involutive or constant dimensional.

The effectiveness of the above algorithm rests upon the following observation.

Lemma 2.1 *Let (x_0, u_0) be an equilibrium point of (2.20), and assume that f_* has full rank around (x_0, u_0). Let D be an involutive constant dimensional distribution on $M \times U$ such that $D + K$ is also involutive and constant dimensional. Then there exists a neighborhood O of (x_0, u_0) such that $f_*(D|_O)$ is an involutive constant dimensional distribution around x_0.*

Based on the above algorithm and lemma, Grizzle [72] states necessary and sufficient conditions for locally linearizing a nonlinear system to a controllable one.

Theorem 2.4 (Grizzle) *Consider the discrete-time nonlinear system (2.20), about the equilibrium point (x_0, u_0). The system (2.20) is linearizable around (x_0, u_0) to a controllable linear system if and only if Algorithm 2.1 applied to the system (2.20) gives distributions $D_0, ..., D_n$ such that $dim(D_n) = n + m$.*

The proof of the above lemma and theorem can be found in [72]. In the next section. we are going to show that our nonlinear system satisfies the conditions of the above theorem and thus the system is locally linearzable.

2.6.3. Main Result

Now. let us consider our nonlinear system (2.20) in which f(x. u) has the form

$$f(x,u) = \begin{bmatrix} a^0x^0 + u \\ w_{11}c^0a^0x^0 + a^1x^1 + w_{11}c^0u \\ \vdots \\ w_{1(n-2)}c^0a^0x^0 + a^{n-2}x^{n-2} + w_{1(n-2)}c^0u \\ a^{n-1}x^{n-1} + \sum_{i=1}^{n-2} w_{2i}s_2(c^ia^ix^i + c^iw_{2i}c^0a^0x^0 + c^iw_{1i}c^0u) \end{bmatrix}, \quad (2.27)$$

where $x = (x^0, x^1, \cdots, x^{n-1})$ and u is a scalar. Before we present the main theorem, we shall stat and prove some lemmas, which will be used later.

Lemma 2.2 *Consider the nonlinear system (2.20) and the nonlinear function f(x,u) of (2.27). If* $a^i \neq 0$ *for* $0 \leq i \leq n-1$, *then* f_* *has full rank around the equilibrium point* (x^*, u^*).

Proof: By noting that $f : M \times U \rightarrow M$ is given by (2.27) , we can evaluate $f_* : TM \times TU \rightarrow TM$ by considering the natural basis $(\frac{\partial}{\partial x^0}, \ldots, \frac{\partial}{\partial x^{n-1}})$ in TM and $\frac{\partial}{\partial u}$ in TU, where $M \in R^n$, $U \in R$, and TM and TU are the tangent spaces for M and U respectively. Let $Z^1, Z^2, ..., Z^n$ be the basis in the image of f_*. Then,

$$f_* \begin{pmatrix} \frac{\partial}{\partial x^0} \\ \frac{\partial}{\partial x^1} \\ \vdots \\ \frac{\partial}{\partial u} \end{pmatrix} = A \begin{pmatrix} Z_1 \\ Z_2 \\ \vdots \\ Z_n \end{pmatrix},$$

where $A = (a_{ij})$, given by $a_{ij} = \frac{\partial f^j}{\partial x^i}$, i, j = 0, 1, ..., n-1, and $a_{nk} = \frac{\partial f^k}{\partial u}$, k=0, 1, ..., n-1.

Thus,

$$A = \begin{bmatrix} a^0 & w_{11}c^0a^0 & \cdots & w_{1(n-2)}c^0a^0 & \sum_{i=1}^{n-2} w_{2i}s_2'(.)c^i w_{1i}c^0a^0 \\ 0 & a^1 & \cdots & 0 & w_{21}s_2'(.)c^1a^1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & a^{n-2} & w_{2(n-2)}s_2'(.)c^{n-2}a^{n-2} \\ 0 & 0 & \cdots & 0 & a^{n-1} \\ 1 & w_{11}c^0 & \cdots & w_{1(n-2)}c^0 & \sum_{i=1}^{n-2} w_{2i}s_2'(.)c^i w_{1i}c^0 \end{bmatrix}.$$

Since $a^i \neq 0$ for $0 \leq i \leq n-1$, rank(A) = n and thus f_* has full rank around (x^*, u^*).

Q.E.D.

Lemma 2.3 *Let the conditions in Lemma 2.2 be satisfied. Let D be a subspace in $TM \times TU$. Then*

$$dim(f_*(D)) = \begin{cases} dim(D) & \text{if } dim(D) \leq n, \\ n & \text{if } dim(D) = n+1. \end{cases}$$

Proof: Case (i) dim(D) ≤n:

Suppose that dim(D) = p ≤ n and let $Y^1, Y^2, \ldots, Y^p$ be the basis in D. Then, without loss of generality, we have

$$\begin{pmatrix} Y^1 \\ Y^2 \\ \vdots \\ Y^p \end{pmatrix} = P \begin{pmatrix} \frac{\partial}{\partial x^0} \\ \frac{\partial}{\partial x^1} \\ \vdots \\ \frac{\partial}{\partial x^{n-1}} \\ \frac{\partial}{\partial u} \end{pmatrix}, \qquad (2.28)$$

where $P \in R^{p\times(n+1)}$ and rank(P) = p. Then,

$$f_*\begin{pmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_p \end{pmatrix} = Pf_*\begin{pmatrix} \frac{\partial}{\partial x^0} \\ \frac{\partial}{\partial x^1} \\ \vdots \\ \frac{\partial}{\partial x^{n-1}} \\ \frac{\partial}{\partial u} \end{pmatrix} = PA\begin{pmatrix} Z_1 \\ Z_2 \\ \vdots \\ Z_n \end{pmatrix} = \hat{P}\begin{pmatrix} Z_1 \\ Z_2 \\ \vdots \\ Z_n \end{pmatrix} = \begin{pmatrix} W_1 \\ W_2 \\ \vdots \\ W_p \end{pmatrix}.$$

To prove that $f_*(D) = \text{span}(W_1, ..., W_p)$, it suffices to show that rank($\hat{P}$)=p. Indeed, the fact that rank(P)=p $\leq n$ and rank(A) = n implies that rank($\hat{P}$)=p. Thus, dim(f_*(D)) = dim (D).

Case(ii) dim(D) = n+1:

Then (2.28) still holds, but in this case, $P \in R^{(n+1)\times(n+1)}$ and rank(P)=n+1. The fact that rank($\hat{P}$) follows from Sylvester's inequality on the rank of the product of two matrices and the rank inequality for matrices $A \in R^{n\times n}, B \in R^{n\times m}(m \leq n)$

$$rank(A) + rank(B) - n \leq rank(AB). \tag{2.29}$$

Therefore, dim(f_*(D))=n.

Q.E.D.

Lemma 2.4 *Let π denote the canonical projection from $M \times U$ onto M given by $\pi(x,u) = x$, and let Q be a subset of TM with $dim(Q)=p \leq n$. Then $dim(\pi_*^{-1}(Q))=p+1$.*

Proof: Let $Y_1, ..., Y_p$ be a basis in Q. Then any vector field in Q can be represented by $\sum_{i=1}^{p} a_i Y_i$, that is,

$$(a^1, \ldots, a^p, 0, \ldots, 0) \begin{pmatrix} Y_1 \\ \vdots \\ Y_p \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

But

$$\begin{pmatrix} Y_1 \\ \vdots \\ Y_p \\ 0 \\ \vdots \\ 0 \end{pmatrix} = \begin{pmatrix} P_1 & P_2 \\ 0 & P_3 \end{pmatrix} \begin{pmatrix} \frac{\partial}{\partial x^1} \\ \vdots \\ \frac{\partial}{\partial x^n} \\ 0 \end{pmatrix} = P \begin{pmatrix} \frac{\partial}{\partial x^1} \\ \vdots \\ \frac{\partial}{\partial x^n} \\ 0 \end{pmatrix},$$

where $P_1 \in R^{p\times n}$, $P_2 \in R^{p\times 1}$, $P_3 \in R^{(n+1-p)\times 1}$ and rank(P_1)=p, rank(P_2)=1.

Let $[\frac{\partial}{\partial x^1}, \ldots, \frac{\partial}{\partial x^n}, \frac{\partial}{\partial u}]^T$ be a basis in $TM \times TU$, then

$$\pi_* \begin{pmatrix} \frac{\partial}{\partial x^1} \\ \vdots \\ \frac{\partial}{\partial x^n} \\ \frac{\partial}{\partial u} \end{pmatrix} = I \begin{pmatrix} \frac{\partial}{\partial x^1} \\ \vdots \\ \frac{\partial}{\partial x^n} \\ 0 \end{pmatrix},$$

where I is an identity matrix.

Thus,

$$\pi_*^{-1} \begin{pmatrix} a^1 & \cdots & a^n & a \end{pmatrix} \begin{pmatrix} \frac{\partial}{\partial x^1} \\ \vdots \\ \frac{\partial}{\partial x^{n-1}} \\ 0 \end{pmatrix} = \begin{pmatrix} a^1 & \cdots & a^n & a \end{pmatrix} \pi_*^{-1} \begin{pmatrix} \frac{\partial}{\partial x^1} \\ \vdots \\ \frac{\partial}{\partial x^n} \\ 0 \end{pmatrix}$$

$$= \begin{pmatrix} a^1 & \cdots & a^n & a \end{pmatrix} \begin{pmatrix} \frac{\partial}{\partial x^1} \\ \vdots \\ \frac{\partial}{\partial x^n} \\ \frac{\partial}{\partial u} \end{pmatrix} =$$

$$= \sum_{i=1}^{n} a^i \frac{\partial}{\partial x^i} + a\frac{\partial}{\partial u}, \tag{2.30}$$

and

$$\pi_*^{-1}\begin{pmatrix} Y_1 \\ \vdots \\ Y_p \\ 0 \\ \vdots \\ 0 \end{pmatrix} = P\pi_*^{-1}\begin{pmatrix} \frac{\partial}{\partial x^1} \\ \vdots \\ \frac{\partial}{\partial x^n} \\ 0 \end{pmatrix} = PI\begin{pmatrix} \frac{\partial}{\partial x^1} \\ \vdots \\ \frac{\partial}{\partial x^n} \\ \frac{\partial}{\partial u} \end{pmatrix}.$$

So rank(P) = p+1 implies that

$$\pi_*^{-1}\begin{pmatrix} Y_1 \\ \vdots \\ Y_p \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

has dimension p+1 or $\dim(\pi_*^{-1}(Q))$=p+1.

Q.E.D.

Theorem 2.5 *Consider the discrete-time nonlinear system (2.20) about the equilibrium point* (x^*, u^*). *If* $a^i \neq 0$ *for* $0 \leq i \leq n-1$, *then the system is linearizable around* (x^*, u^*) *to a controllable linear system.*

Proof:

Step 1: Using lemma 2.2, we see that f_* has full rank around the equilibrium (x^*, u^*). Therefore, we can apply algorithm 6.1 to compute D_i.

Let $K = \ker f_*$. Note that $f_* : TM \times TU \to TM$, and $TM \times TU \subset R^n \times R$ and $TM \subset R^n$. Therefore $f_*(\bar{a}^1, \cdots, \bar{a}^n, \bar{a}) = (\bar{a}^1, \cdots, \bar{a}^n, \bar{a})A$. The equality $f_*(\bar{a}^1, \bar{a}^2, \cdots, \bar{a}^n, \bar{a}) = \mathbf{0}$ implies that

$$(\bar{a}^1, \cdots, \bar{a}^n, \bar{a})A = \mathbf{0}. \tag{2.31}$$

Let

$$T = \begin{bmatrix} 1 & -w_{11}c^0 & \cdots & -w_{1(n-2)}c^0 & -\sum_{i=1}^{n-2} w_{1i}s_2(.)c^i w_{1i}c^0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & \cdots & 0 & 1 \end{bmatrix}$$

and

$$A^* = AT = \begin{bmatrix} a^0 & 0 & \cdots & 0 & 0 \\ 0 & a^1 & \cdots & 0 & w_{21}s_2'(.)c^1a^1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & a^{n-1} \\ 1 & 0 & \cdots & 0 & 0 \end{bmatrix}.$$

Now right-multiplying (2.31) by T yields:

$$\begin{pmatrix} \bar{a}^1 & \cdots & \bar{a}^n & \bar{a} \end{pmatrix} A^* = 0,$$

or

$$\begin{pmatrix} \bar{a} + a^0\bar{a}^1 & a^1\bar{a}^2 & \cdots & a^{n-2}\bar{a}^{n-1} & \sum_{i=1}^{n-2} w_{2i}s_2'(.)c^i a^i \bar{a}^{i+1} + a^{n-1}\bar{a}^n \end{pmatrix} = \mathbf{0},$$

from which we conclude that $\bar{a}^2 = \cdots = \bar{a}^{n-1} = \bar{a}^n = 0$. But $\bar{a} = -a^0\bar{a}^1$. So,

$$\begin{aligned} K &= \text{span}(\bar{a}^1 \frac{\partial}{\partial x^1} - a^0\bar{a}^1 \frac{\partial}{\partial u}). \\ &= \text{span}(\hat{Y}) \end{aligned}$$

where $\hat{Y} = \frac{\partial}{\partial x^1} - a^0 \frac{\partial}{\partial u}$ and dim(K)=1.

Step 2: Let $D_0 = \pi_*^{-1}(0)$. Then we have $D_0 = \text{span}(\frac{\partial}{\partial u})$ and $\dim(D_0)$=1 from Lemma 2.4. Let $D_{i+1} = \pi_*^{-1} f_*(D_i)$ for $0 \leq i < n$.

Suppose now $\dim(D_i)$=p and $X_1, \cdots, X_p$ are a basis for D_i. Then we have

$$\begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_p \end{bmatrix} = P \begin{bmatrix} \frac{\partial}{\partial x^1} \\ \vdots \\ \frac{\partial}{\partial x^n} \\ \frac{\partial}{\partial u} \end{bmatrix},$$

where $P \in R^{p \times (n+1)}$. Thus, $[X_1, \cdots, X_p, \tilde{Y}]$ is a basis for $D_i + K$, and

$$\begin{bmatrix} X_1 \\ \vdots \\ X_p \\ \tilde{Y} \end{bmatrix} = \begin{bmatrix} P \\ \cdots \\ \bar{p} \end{bmatrix} = \begin{bmatrix} \frac{\partial}{\partial x^1} \\ \vdots \\ \frac{\partial}{\partial x^n} \\ \frac{\partial}{\partial u} \end{bmatrix},$$

where $\bar{p}$=[1 0 0 ... 0 $-a^0$]. Obviously, $[X_i, X_j] = 0$ for $i \neq j$ and $[X_i, \tilde{Y}]$=0 for all i. Therefore, $D_i + K$ is involutive and has constant dimension. Repeatedly applying Lemma 2.2 and Lemma 2.3 on D_i, and using induction on i, we obtain a D_n whose dimension is n+1. It follows from Grizzle's necessary and sufficient condition that the nonlinear system is linearizable to a controllable linear system.

Q.E.D.

2.7. Discussion

The lack of rigorous mathematical representation of control systems in current paradigms of feedforward and recurrent neural networks is a drawback to the development of research on neural networks for control. The feedforward networks are known to work as a mapping between two information domains. Most of the current research in neural networks for control and related publications discuss

using this type of neural network to "learn" a model or a controller, which is usually either highly nonlinear or hard to implement. The results published show that these approaches are satisfactory in some cases. However, there is little development to attempt to relate the theory of classical and modern control systems to this type of neural network. Neural networks of this type are always treated as a "Black Box" and thus there is no direct contact with the "internal" information of the box. A classical linear control system, which may also be called a "Black Box", can be represented by a transfer function in the linear case, and thus the input-output performance can be studied thoroughly. This work exploits the "internal information" of the network and attempts to represent the control systems in terms of this information. Therefore, the network itself is not only a control system, but it is also capable of learning. In this case, the paradigm presented here may be viewed as an extension of current recurrent networks.

As quoted in [93] by Williams: "While much of the recent emphasis in the field has been a multilayer network having no feedback connections, it is likely that the use of recurrently connected networks will be of particular importance for applications to the control of dynamical systems". Indeed, because of the incorporation of feedback or dynamics inside the networks, the recurrent networks show great promise for the future of research on neural networks for the purpose of control. The property that the Hopfield net has a Constant Addressable Memory provides a way for implementing many practical problems; e.g., traveling salesman problems. Another particular type of recurrent network, *a settling network*, has also been widely recognized as important in connectionist circles. Such a network converges to a stable state from any starting state. The final state of such a network can be viewed as the solution to a certain constraint-satisfaction-type of search, as in *relaxation labeling*, or it might be viewed as a retrieved item

from a content-addressable-associative memory. Despite this, the ambiguity of information stored in networks hinders the networks direct use of the information, and thus there is very limited use for this type of network for control purposes.

Our attempt is to mathematically formulate the control systems inside the neural networks. We can easily represent each linear SISO system in the neural network, by introducing a small feedback loop INSIDE each neuron, rather than a feedback connection. For this paradigm of neural networks, we can directly use the internal states to construct a feedback control law. What is more important is that a network of this type is itself a system, but not an unknown "Black Box". Thus its input-output performance can be studied just as in the case of the classical control system . Based on this observation, many conventional synthesis methods can be directly borrowed to design the system. The stationary property of the system can be preassigned by means of learning, a unique feature that the classical control system does not have.

Of course, this is only a first step in this direction of research. There are still many interesting open problems, such as:

1. Designing a controller which is also a neural network of the same structure, and then applying the controller in the system modelled by the neural networks discussed in this chapter. It is interesting to study this type of mixed network and to explore its properties.

2. Carrying out research in the case of multi-variable system. It is straightforward to extend current results to a multi-input and multi-output system. However, extension of the results of linearizability is not trivial and requires further study.

3. Considering how to construct the training set. By the Separation Principle of Learning and Control, the systems of this chapter can be regarded as networks when the dynamic parameters are set to zero. Thus they have the capacity of learning. The construction of a training set is an interesting problem. Also by this same Principle, it is not hard to show that it is possible to construct a training set such that after the network has learned, it has the desired stationary properties. Thus, the problem of how to construct a training set so that the trained system has the desired stationary property needs to be investigated.

4. Considering the applications of results in this chapter to the differential game problems. Differential game problems can be modeled by an NNLM, and control strategy for each player can be obtained using various controller design techniques. In this case, the NNLM used for modeling the differential game has at least two inputs, since most differential games have at least two players. This application is very interesting and is worthy of further study. Although results in this chapter will not be directly used in the next chapter, they do have potential applications for such kinds of problems. For example, by modeling a differential game problem using an NNLM and by designing controllers using additional neural networks without local dynamics, we obtain a network consisting of several small neural networks. Studying such a network which consists of several small neural networks is also interesting. We hope that results of this kind will appear in the near future.

3. Differential Games with Neural Networks

Rufus Isaacs [40] first looked into the theory of modeling tactical encounters in what he termed "Differential Games" in his seminal Rand report [39]. Isaacs assumed a differential model for aircraft dynamics. He also assumed that the roles of pursuer and evader are fixed for the duration of an encounter. Only the pursuer was assumed to have weapon capabilities that could be modeled by a hyper-surface in the state space. For more than two decades since Isaacs' pioneering work, there have been many publications in the literature about this subject. However, the problems studied in the literature were different from that studied by Isaac in his original report. There are many practical systems which can be modeled by differential game problems. For example, cooperative and non-cooperative games are typical differential games in the area of economic systems. Up to now, it is commonly agreed upon that there has been a strong theoretical foundation in the field of differential games.

However, there are some reasons why differential games have not had widespread use in air combat whose arena is one of the most complex dynamic systems. In fact, the applications of conventional differential game theory in many practical problems are limited because of the following reasons. First, the classical solution of a differential game is based on simultaneous backwards integration of the state and adjoint equations, starting at the target set (terminal manifold) of the game, in order to fill the entire game space by the ensemble of optimal trajectories. The backwards integration is a rather direct operation as long as no singular surface of the game exists. In a simple game, with no more than two state variables, the

existence of the singular surface can be easily visualized. For a differential game of three independent state variables the same process becomes very cumbersome, and for dynamic models of higher dimension, it is virtually impossible. Second, at the other end, the numerical solution of a two-point boundary value problem associated with the differential game satisfies only the necessary conditions of optimality, and it cannot identify the singular surfaces of the game and has no tool to verify the sufficiency conditions of the game solution. Third, as illustrated by an example in [81], Shinar indicates that the frame of classical differential game theory is rather limited to accommodate even relatively simple models of a "real-world" dynamic conflict. To be more exact, the assumption that the roles of pursuer and evader are fixed during an encounter is not reasonable since all participants in an encounter have weapon systems that can be modeled by a continuous probabilistic function. Fourth, in future air combat most engagements will start at rather long (beyond visual) ranges. Thus, the initial conditions of the above described two-target game are generally in the "draw" zone. Therefore, the only guaranteed outcome of such a non-cooperative game is a "draw". This result, which denies the very essence of an air-to-air combat and consequently the justification of the high cost of advanced aircraft and missile development, is clearly unacceptable from an operational point of view.

As a consequence of the difficulties outlined in the previous paragraph, one is strongly tempted to search for an alternative approach to the analysis of dynamic conflicts. *A priori*, Artificial Intelligence seems to represent such an alternative. An interesting concept introduced in [77] is the OODA loop - Observe, Orient, Decide and Act - in a cyclical maneuver. The OODA concept is based on the fact that in current classical air combat at short range, pilots have been using their eyes as sensors and their brains to integrate the visual and sensory-supplied

information necessary to play the game. Although the OODA loop undoubtedly plays an important role in air combat pilotage, the limitations are obviously those of human ability in the supersonic combat environments. Humans are characterized by a limited processing rate - two events taking place in less than about one tenth of a second will generally be perceived as a single event. Another limitation concerning the processing rate involves the fact that an activity of integrated percepts, decision and motor action is performed. To overcome these difficulties, Rodin *et al* proposed in [77] an Artificial Intelligence approach to designing an operational on-board system, called "Tactical Decision Aid Expert Systems (TDAES)", to support pilots in tactical decision making processes. The Expert System generated an initial flight and action plan (initial mission). The optimal plan gets reevaluated and possibly changed every time when an unforeseen event takes place. The system employs a basic set of pursuit-evasion algorithms for suboptimal mission generation.

In their pioneering work, Rodin *et al* proposed in [77] the use of artificial intelligence methodology in air combat games. The work was based on the semantic control paradigm proposed in [76] by Rodin. In this paradigm, a control problem is broken into three blocks. Their functions, when applied to a situation governed by differential games (for instance), are as follows:

(i) Identifier: Identifier block identifies the differential game, parameters and role that the aircraft should assume in order to destroy an assigned target.

(ii) Goal Selector: The Goal Selector solves the differential game chosen by the Identifier block. The results are the optimal trajectories, barriers and controls.

(iii) Adaptor: The Adaptor determines the controls that causes the aircraft to "best" follow the optimal trajectory determined by the Goal Selector.

In [95], Weil used Artificial Intelligence methodologies to splice the solutions in low order one target models together in a sub-optimal fashion that will be useful in air combat. He explored the characteristics of a self modifying system. Among the methodologies he used, the artificial neural net approach is impressive. Several neural nets are in his approach. One net is the classification net that determines which differential game from the knowledge base will be chosen along with some of the criteria that might be used to make the decision. Once a differential game is chosen, another neural net is assigned to the game so that it can determine the parametrizations of the game. The method of generating the training set for each net is unique in the sense that, instead of generating the training set off-line by using traditional methods, the training process is done by closing the loop simulation forward and adjusting the weights by propagating errors backward in time. Training each individual net is relatively independent.

Based on the above semantic control paradigm, we present here a new approach to using neural networks in differential game problems. The approach reflects the fundamental phases in real-life conflicts. Instead of building up a knowledge base, we implement the neural nets in the Identifier and Adaptor blocks of the semantic control paradigm so that the approach is more general and is capable of working on-line during an entire encounter. Generating training sets for these two neural nets is different from other approaches [49, 95] in the sense that the nets in our approach do not learn the optimal trajectories generated by optimal controls but learn the mapping between two information regions.

This chapter first discusses the fundamental phases in real-life conflicts and gives a general configuration summarizing these phases. The configuration provides a basis for further analysis and the paradigm for differential games with neural networks. In Section 3.2, two paradigms of differential games with neural networks are given with emphasis on one player and two players respectively. These paradigms represent semantic control. One contribution of this approach is the Identifier, which takes the environmental information and outputs an estimate of the opponent's strategy. In Section 3.3, the INTERCHANGEABILITY conditions will be discussed. Interchangeability is a basic assumption throughout this study. In Section 3.4, several algorithms with the same basic frame will be given. A detailed discussion of each stage and of the implementation of the algorithms are also given. In Section 3.5, a detailed study of a pursuit-evasion game problem will be given. The simulation results are quite satisfactory. Discussion of the simulation results with the algorithms of different paradigms, direction of further research on this topic, and the conclusions will be given as well.

3.1. Motivation

In real-time life conflicts, the actions of two players usually consist of three phases: discovering the opponent, finding what this opponent is doing, and making a decision. Upon finding the opponent, the pilot will first identify his maneuver, and then react according to his opponent's perceived action. Sometimes, it is hard to distinguish between phases because some phases may last very short periods. These three phases are the same in all air combat problems, though the last two stages may repeat for the successive time intervals, e.g. the pilot may change

from evasion to pursuit upon finding that the action of his opponent changes from attack to withdrawal.

The above process is similar to the OODA loop outlined in the last section, and these phases can be formally stated as the following configuration. We assume that there are two players engaged during the entire encounter of the game. Each player employs the three phases in the process of making a control decision. We only give the phases for one player because the phases for the other player are the same. For each player, each individual phase may start at a different instant.

General Configuration of Phases in Real-life Conflicts:

Phase 1: Discovery of the opponent
Phase 2: Identification of the actions of this opponent
Phase 3: Making decision for next action according to this opponent's perceived strategy

Phase 1 is self-explanatory. Immediately after finding his opponent, the player identifies an approximation to the strategy and maneuver and obtains information about the range, bearing, heading and speed of his opponent. Gathering this information is accomplished in phase 2. In our discussion, examples of strategies may be pursuit, evasion, or disengage. Typical examples of maneuvers may be found in [57, 80]. Techniques used for identifying the tactical air combat maneuvers are in the forms of decision-making, scheduling, and control systems [10, 22, 37, 64]. Recent works in using neural networks in diverse ways for the classification of underwater sonar targets [27] and recognition of radar targets [71] show great promise in the area of maneuver identification. The information about the range, bearing, heading and speed of his opponent is generally gathered by means of mathematical devices. Many times, this information is crucial to the

player's control decision making. In phase 3, the player reacts according to his opponent's perceived action. His strategies might be pursuit, evasion, or disengage. Generally speaking, his strategy is some type of generic mapping of his opponent's strategy, which in some cases may be a continuous/discrete-time type of mathematical functions. In light of the above, his strategy could be

$$\text{strategy} = \begin{cases} \text{evasion} & \text{if his opponent is pursuing;} \\ \text{pursuit} & \text{if his enemy is escaping and a chance exists;} \\ \text{observation} & \text{if both sides choose to be peaceful.} \end{cases}$$

REMARKS:

1. In this configuration, one player should be in one and only one phase at any instant although sometimes it is hard to distinguish between phases because of their very short periods. For example, one player usually identifies what strategy his opponent is taking almost at the same time he discovers his opponent and thus it is usually hard to distinguish between phase 1 and phase 2 for this player.

2. At any instant, two players may be in different phases because the phases for each individual player may last for different periods. Moreover, the last two phases for each individual player would alternate as time passes. However, the mutual and continual observation of each player by his opponent is also assumed, as is their mutual ability to identify the opponent's maneuvers.

3. Although the dependence of next action on the opponent's strategy is usually not clear in general zero-sum differential game problems, this configuration is realistic for many applications of differential games and other real-life conflicts. In fact, in the general zero-sum differential game problems, each

player in such games is supposed to play his own optimal strategy regardless of what his opponent does. Failing to do so on one side may do the other side a favor by increasing or decreasing the relevant cost function. However, not only are there some categories of differential game problems in which the optimal strategy of one side depends on the strategy of the other side, but also real life seems to be like that. A pilot will rarely fly an absolute optimal trajectory and a good pilot will overcome a poor one by capitalizing on the latter's mistakes. Thus, he will formulate the plan of his actions in accordance to what his opponent happens to be doing.

To justify the general configuration given previously, we shall give the following simple two-player game problem in which E denotes one player (e.g. an evader) and P the other (e.g. a pursuer). In this example, we shall omit the phases 1 and 2 in the general configuration, since they are conceptually simple in this case, and emphasize the phase 3.

Example:

Consider a simple game problem in a 2-dimensional plane. $\wp$ is the upper half-plane and $\Re$ is the x-axis. The vecto-gram for E is shown in Figure 3.1; it has a downward unit vertical component and a horizontal headline of half-length u(x,y), a positive and smooth function. That of P is circular of radius w(x, y), again a smooth, positive function with always $w < u$ and, for some constant c, $w \leq c < 1$. The total velocity for x is to be the vector sum of a choice from each vectogram. Analytically all this means that the kinematic equations are

$$\dot{x} = u(x,y)\psi + w(x,y)sin\phi, \tag{3.1}$$

$$\dot{y} = -1 + w(x,y)cos\phi, \qquad -1 \leq v \leq 1. \tag{3.2}$$

The payoff will be terminal with H = x on $\Re$ (where y =0). Thus E will strive to have **x** reach $\Re$ at a point as far right as possible, and P similarly struggles for the left.

Always E will play his rightmost vector ($\psi = 1$ in the KE). At Figure 3.2 let XA be this vector. The line XB is tangent from X to a circle of radius w (w is reckoned at X) and center A. Then XB is a properly oriented semipermeable direction. If a family of curves is drawn (an ordinary differential equation solved) having these directions as that of their tangents at each point, these curves will be semipermea If each is labeled with the value of H at its meeting with $\Re$, the labelings will constitute V(**x**).

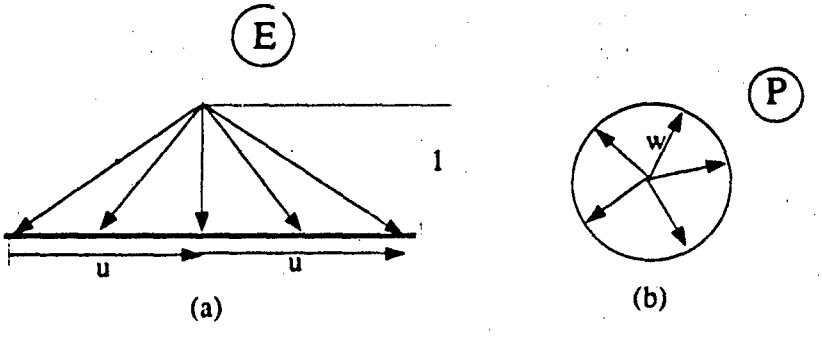


Figure 3.1: Vectograms of Players E and P

From Figure 3.2, it can be seen that

$$\begin{aligned} \beta &= \alpha - tg^{-1}\frac{1}{u} \\ &= \phi_1 - tg^{-1}\frac{1}{u} \\ &= \phi - \pi - tg^{-1}\frac{1}{u}, \end{aligned} \tag{3.3}$$

and

$$sin\beta = \frac{w}{\sqrt{1+u^2}}. \tag{3.4}$$

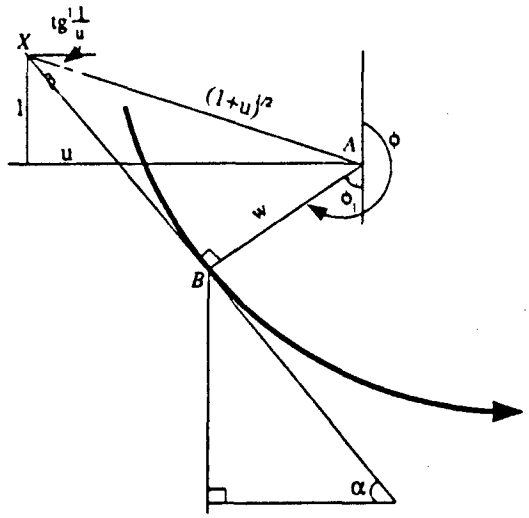


Figure 3.2: Trajectory

It follows from equations (3.3) and (3.4) that

$$\phi = \pi + tg^{-1}\frac{1}{u} + sin^{-1}\frac{w}{\sqrt{1+u^2}}. \tag{3.5}$$

If we can consider u(x,y) to be a control for the evader, it is clear from eq. (3.5) that the control for the pursuer depends on the control of the evader.

3.2. Architecture

A paradigm for differential games with neural networks is shown in Figure 3.3. It is called a one-player paradigm of differential games with neural networks because only one side of the game is considered. Figure 3.3 clearly represents the general configuration discussed in the last section although the first phase is omitted. The paradigm consists of two stages. Stage I represents one player's process of making a control decision, which imitates phase 2 and phase 3 in the general configuration. In this stage, the neural identifier takes the environmental information and gives the estimate of the opponent's control strategy. The input information to the

Neural Identifier is relatively independent of the system state variables because the network takes only the environmental information which usually comes from the sensor, e.g., from readings of an odometer. To represent phase 3 of the general configuration, we use another neural net, called Neural Controller, for the control purpose. Based on the estimate of the opponent's control strategy and the optimal criterion given by the user, the neural controller gives the optimal control for the player. The information of the state variables $\mathbf{x}$ is needed for evaluation of the optimal control because the neural net controller is usually viewed as a part of the system for stage I.

In stage II, because we only consider the one-player paradigm, we simply put a neural net controller in stage II, which represents the process of making a control decision for the other player. We assume no control over the player, and he can make a control decision based on his own criterion.

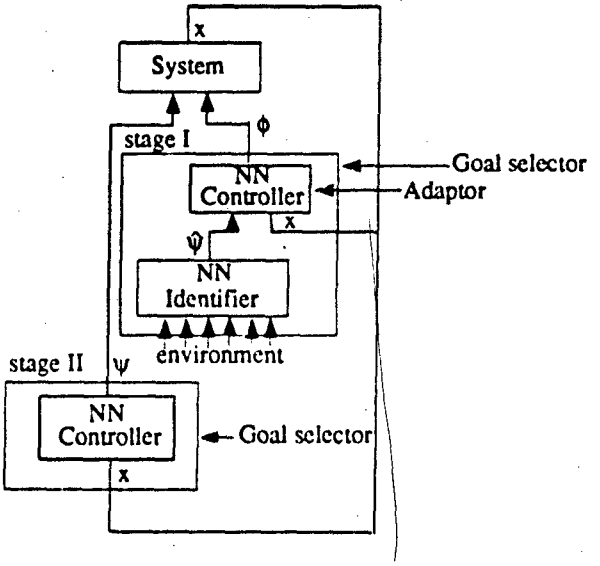


Figure 3.3: One-player Paradigm of Differential Games With Neural Networks (a)

The system is described by a set of differential equations

$$\dot{\mathbf{x}} = f(\mathbf{x}, \phi, \psi), \quad \mathbf{x}(t_0) = \mathbf{x}_0, \tag{3.6}$$

where $f(\mathbf{x}, \phi, \psi)$ is an n-dimensional vector of continuously differentiable functions and $\mathbf{x}$ is an n-dimensional state vector. The goal for both players is to choose ϕ and ψ to maximize and to minimize the cost function $J(\phi, \psi, \mathbf{x}_0)$, i.e.,

$$\max_{\phi \in \Phi} \min_{\psi \in \Psi} J(\phi, \psi, \mathbf{x}_0), \tag{3.7}$$

where Φ is the feasible set for ϕ and Ψ is the one for ψ. The optimal value is denoted by J_{optimal}

$$J_{\text{optimal}} = \max_{\phi \in \Phi} \min_{\psi \in \Psi} J(\phi, \psi, \mathbf{x}_0). \tag{3.8}$$

Now let us define J_ψ by

$$J_\psi = \max_{\phi \in \Phi} J(\phi, \psi, \mathbf{x}_0), \qquad \psi \in \Psi, \tag{3.9}$$

where again Φ and Ψ are the feasible sets which contain all possible values for ϕ, ψ respectively. We have

$$J_{\text{optimal}} = \min_{\psi \in \Psi} J_\psi. \tag{3.10}$$

Obviously, J_ψ is a functional of ψ. Replacing ψ by $\hat{\psi}$, the estimate of ψ, yields

$$J_{\hat{\psi}} = \max_{\phi \in \Phi} J(\phi, \hat{\psi}, \mathbf{x}_0), \qquad \hat{\psi} \in \Psi. \tag{3.11}$$

Equation (3.11) is important in the sense that it is the analytical representation of stage I in Figure 3.3. Namely, equation (3.11) is the representation for the player who tries to use ϕ to minimize $J(\phi, \hat{\psi}, \mathbf{x}_0)$ based on the estimate value $\hat{\psi}$. Changing the cost function $J(\phi, \hat{\psi}, \mathbf{x}_0)$ is equivalent to changing the goal

selector[76] in stage I. Based on the estimate of the control strategy ψ, one player makes a control decision using the optimal criterion (3.11).

If we exchange symbols ψ and ϕ in equation (3.11), it is equivalent to exchanging ϕ and ψ in Figure 3.3 and $J_{\hat{\phi}}$ is, in this case, representing the player who tries to use ψ to minimize $J(\hat{\phi}, \psi, \mathbf{x_0})$ based on the estimate value $\hat{\phi}$. To represent this, we define $\bar{J}_\phi$ and $\bar{J}_{\text{optimal}}$ by

$$\bar{J}_\phi = \min_{\psi \in \Psi} J(\phi, \psi, \mathbf{x_0}), \qquad \phi \in \Phi, \tag{3.12}$$

$$\bar{J}_{\text{optimal}} = \min_{\phi \in \Phi} \bar{J}_\phi. \tag{3.13}$$

Figure 3.4 shows the paradigm for this case. One question may be asked about the equivalence of Figure 3.3 and Figure 3.4. That is equivalent to asking whether

$$\bar{J}_{\text{optimal}} = J_{\text{optimal}}. \tag{3.14}$$

A more detailed discussion on this topic will be given in the next section.

Unlike the general scheme of differential games, the enemy's strategy need not be optimal in this paradigm. Instead of assuming that the enemy plays optimally, we shall consider all the possibilities that the opponent can take. This can be seen in equations (3.9), (3.10) and (3.11). In equation (3.9), J_ψ is a functional of ψ which takes all possible values from the feasible set Ψ.

The above paradigm assumes that the function $f(\mathbf{x}, \phi, \psi)$ is known and it is continuously differentiable to all its arguments. We do not eliminate the possibility that $f(\mathbf{x}, \phi, \psi)$ is unknown, in which case an additional identification process is needed to get the estimate of $f(\mathbf{x}, \phi, \psi)$. For simplicity, we assume that the function $f(\mathbf{x}, \phi, \psi)$ is known throughout this work.

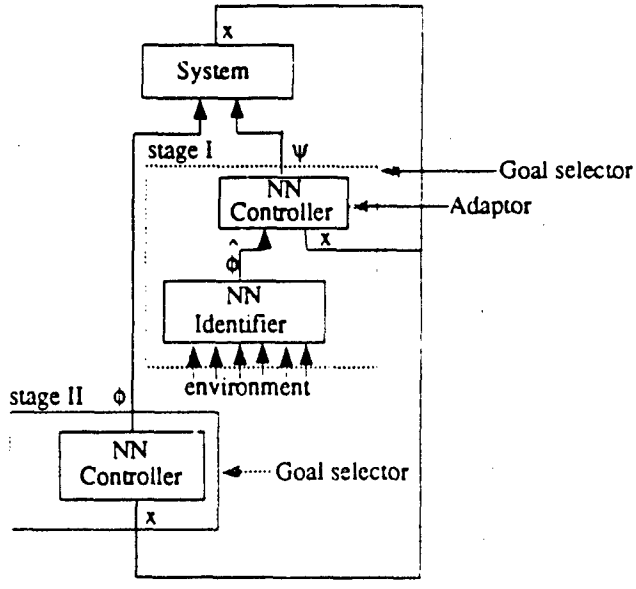


Figure 3.4: One-player Paradigm of Differential Games With Neural Networks(b)

3.3. Interchangeability

In this section, the following assumption will be justified

$$\bar{J}_{\text{optimal}} = J_{\text{optimal}}. \tag{3.15}$$

In general, the following statements are equivalent:

(a) ϕ and ψ are exchangeable in Figure 3.3 and Figure 3.4.

(b) $\bar{J}_{\text{optimal}} = J_{\text{optimal}}$,

(c) Interchangeability:

$$\max_{\phi\in\Phi}\min_{\psi\in\Psi} J = \max_{\phi\in\Phi}(\min_{\psi\in\Psi} J) = \min_{\psi\in\Psi}(\max_{\phi\in\Phi} J),$$

$$\text{subject to } \dot{x} = f(x, \phi, \psi).$$

Before we discuss the problem in detail, let us look at a simple example [12]. Define $L(u, v) = u^2 - 3uv + 2v^2$, $-1 \le u \le 1$, $-1 \le v \le 1$ and it is not hard to

see that

$$\max_v(\min_u L(u,v)) = 0, \tag{3.16}$$

and

$$\min_u(\max_v L(u,v)) = 2. \tag{3.17}$$

Therefore,

$$\max_v(\min_u L(u,v)) \neq \min_u(\max_v L(u,v)).$$

From this, we know that it is not always true that statement (c) holds. The condition that statement (c) or (a) holds is called the interchangeability condition.

In general, consider a continuous-time system whose equation is given by

$$\dot{\mathbf{x}} = A(t)\mathbf{x} + B^1(t)\mathbf{u^1(t)} + B^2(t)\mathbf{u^2(t)}, \quad t \geq 0, \tag{3.18}$$

$$L(\mathbf{u^1}, \mathbf{u^2}) = |\mathbf{x}(t_f)|^2_{Q_f} + \int_0^{t_f} \{|\mathbf{x}(t)|^2_{Q(t)} + |\mathbf{u^1}(t)|^2 - r(t)|\mathbf{u^2}(t)|^2\}dt, \tag{3.19}$$

where $Q_f \geq 0$, $Q(t) \geq 0$, $t \in [0, t_f]$, $r(t) > 0$, t_f is the terminal time, all matrices and $\mathbf{u^1}$ and $\mathbf{u^2}$ have piecewise continuous entries. The initial state $\mathbf{x_0}$ is known to both players.

Let Ξ^i denote the policy space of player i. Further, introduce the function J: $\Xi^1 \times \Xi^2 \rightarrow$ R by

$$J(\mu^1, \mu^2) = L(\mathbf{u^1}, \mathbf{u^2}), \tag{3.20}$$

$$\mathbf{u^i_k} = \mu^i_k(\cdot), \quad k \in \kappa; \quad \mu^i_{[1,k]} \in \Xi^i, i = 1, 2, \tag{3.21}$$

where we have suppressed the dependence on the initial state $\mathbf{x_0}$. The triplet $\{J; \Xi^1, \Xi^2\}$ constitutes the **extensive form** of the zero-sum dynamic game, in the context of which we can introduce the notion of a *saddle-point equilibrium.*

Definition 1:

Define the following quantities:

$$\bar{J} = \min_{\mu^1 \in \Xi^1} \max_{\mu^2 \in \Xi^2} J(\mu^1, \mu^2),$$

$$\underline{J} = \max_{\mu^2 \in \Xi^2} \min_{\mu^1 \in \Xi^1} J(\mu^1, \mu^2),$$

where $\bar{J}$ and $\underline{J}$ are the upper value and the lower value, respectively. Generally, we have the inequality $\bar{J} \geq \underline{J}$ in the context of static games.

Definition 2:

Given a zero-sum dynamic game $\{J; \Xi^1, \Xi^2\}$, in extensive form, a pair of policies $(\mu^{1*}, \mu^{2*}) \in \Xi^1 \times \Xi^2$ constitutes a saddle point solution if, for all $(\mu^1, \mu^2) \in \Xi^1 \times \Xi^2$,

$$J(\mu^{1*}, \mu^2) \leq J^* := J(\mu^{1*}, \mu^{2*}) \leq J(\mu^1, \mu^{2*}). \tag{3.22}$$

The quantity J^* above is called the value of the game, which is defined even if a saddle point solution does not exist, as

$$\bar{J} := \min_{\mu^1 \in \Xi^1} \max_{\mu^2 \in \Xi^2} J(\mu^1, \mu^2) = J^* = \max_{\mu^2 \in \Xi^2} \min_{\mu^1 \in \Xi^1} J(\mu^1, \mu^2) =: \underline{J} \tag{3.23}$$

Only when $\bar{J}$ and $\underline{J}$ are equal, as in eq. (3.23), is the value J^* of the game defined.

In [9], Basar considered the system described by eq. (3.18) and the cost function in eq. (3.19) and gave the following result on the open-loop saddle point solution [9]:

Lemma 3.1 *The quadratic objective functional $L(u^1, u^2)$ given by eq. (3.19), and under the state equation (3.18), is strictly concave in u^2 for every open-loop policy*

u^1 of Player A, if, and only if, the following Riccati differential equation does not have a conjugate point on the interval $[0, t_f]$

$$\dot{S} + A'S + SA + Q + \frac{1}{r}SB^2B'^2S = 0; \quad S(t_f) = Q_f. \tag{3.24}$$

We have the following result on the open-loop saddle-point solution [9]:

Theorem 3.1 *For the linear-quadratic differential game with open-loop information structure, let the condition of above Lemma be satisfied, and introduce the following Riccati differential equation*

$$\dot{Z} + A'Z + ZA + Q - ZB^1B'^1Z + \frac{1}{r}ZB^2B'^2Z = 0; \quad Z(t_f) = Q_f. \tag{3.25}$$

Then,

(i) *The Riccati differential equation (3.24) does not have a conjugate point on the interval $[0, t_f]$.*

(ii) *The game admits a unique saddle-point solution, given by*

$$u^{1*} = \mu^{1*}(t; x_0) = -B^1(t)'Z(t)x^*(t) \tag{3.26}$$

$$u^{2*}(t) = \mu^{2*}(t; x_0) = \frac{1}{r(t)}B^2(t)'Z(t)x^*(t), \quad t \geq 0, \tag{3.27}$$

where $x^*_{[0,t_f]}$ is the corresponding state trajectory, generated by

$$\dot{x}^* = (A - (B^1B'^1 - \frac{1}{r}B^2B'^2)Z(t))x^*; \quad x^*(0) = x_0. \tag{3.28}$$

(iii) *The saddle-point value of the game is*

$$L^* = L(u^{1*}, u^{2*}) = x_0'Z(0)x_0. \tag{3.29}$$

(iv) *If the Riccati equation (3.24) has a conjugate point in the open interval* $(0, t_f)$, *then the upper value of the game is unbounded.*

In this study, we would consider the system described by eq. (3.18), assuming that the interchangeability condition is satisfied. Thus, we have the following:

ASSUMPTION

$$\min_{\phi} \max_{\psi} J = \max_{\psi} \min_{\phi} J.$$

As stated in the theorem above, the assumption can be checked by the conjugate points of the Riccati differential equation (3.24). If this assumption is satisfied, the game saddle point and the calculus saddle point are equivalent [12].

3.4. Algorithm

Based on the discussions in the previous sections, several algorithms will be presented in this section. The basic idea is for one player to repeat phase 2 and phase 3 of the general configuration described in the first section while assuming no control over his opponent's strategy. For the other player, his strategy can be generated by some external generator. This process results in the one-player paradigm algorithm which will be described below. In practice, the other player may implement the same process of the phase 2 and phase 3 as well, which results in the two-player paradigm algorithm. The two-player paradigm algorithm of differential games with neural networks will be discussed later in this section. The details of each algorithm will be given. One should know that the basic frame of each algorithm remains the same, namely, to repeat stage I and stage II for one particular player or all players. In the one-player paradigm algorithm,

phases 2 and 3 of stage I for one player are repeated and we assumed no control over the strategy for the other player. In the two-player paradigm algorithm, we implement the same process of both stage I and stage II for each player. Finally, in algorithm 3, we implement the process in stage II based on the idea discussed in the last Section 3.2. Figure 3.5 shows the time step for the implementation.

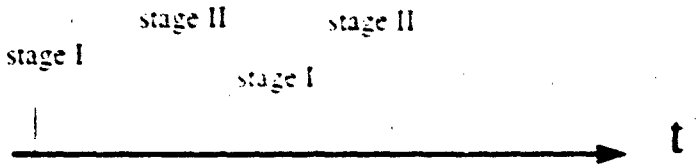


Figure 3.5. Time Steps For One Player

In what follows, the player to whom the algorithms are applied is called "own player", and his counterpart is "the other player".

Algorithm 1(One-player Paradigm):

Stage I: **(1)** Using the environmental information, identify the other player's control strategy.

(2) Based on the estimate of the other player's movement or strategy, generate a new control value for own player by using the optimal control law.

Stage II: Generate a new control value for the other player.

REMARKS:

1. The environmental information in stage I is usually numerical values of observable state variables, but that information could be curves generated

by state variables. One way to obtain the environmental information is to first obtain the characteristics of the segment of curves, e.g. the direction information and the sharpness of the turning corner, and then match any of this information with that stored in the database. The strategy assigned to any matched segment of curves will be said to be the strategy employed to generate that segment. Details of the method will not be given here. In the remainder of this dissertation, we will use the observable variables as the environmental information in stage I.

2. The control value in stage II could be generated by any control law (or maneuver) since the own player has no control over the opponent player. In this algorithm, changing control law on-line is allowed for the opponent player. The own player has the capability, realized by an identifier in stage I, of keeping track of the changes of control strategies of the opponent player.

3. In practice, the difference in the length of time intervals for stage I and stage II could be significant since an optimal control law is used in stage I and hence the time required to compute the optimal control could be much more than that in stage II.

4. In our simulations, we realized the identifier and controller using neural networks. There are many publications in the literature discussing the problems of neural controllers (e.g., [48, 95]). This type of controller is particularly useful for difficult control problems [3].

In what follows, we give a slightly different algorithm called a two-player paradigm algorithm. By assuming that two players use the same process of identification and control, we can construct, in Figure 3.6, a different paradigm of differential

games with neural networks. In this paradigm, instead of assuming an arbitrary controller structure in stage II, we use the same process for the opponent player.

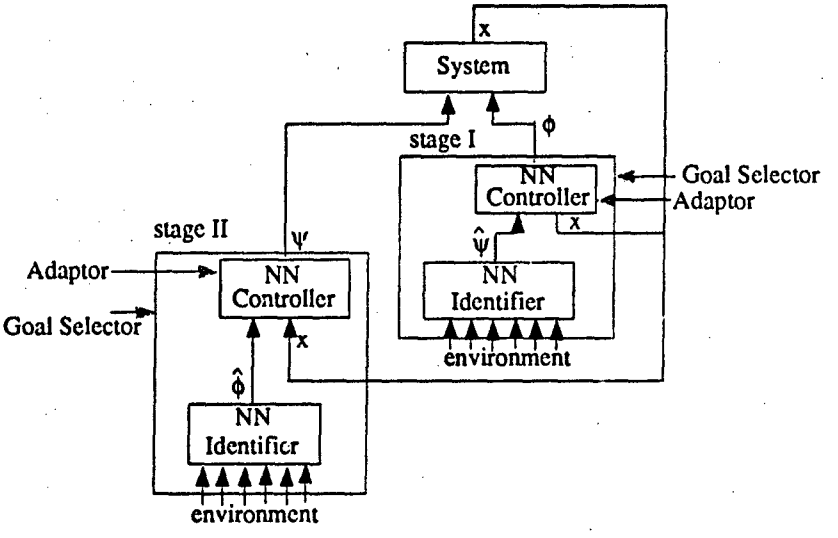


Figure 3.6: Two-player Paradigm of Differential Games With Neural Networks

In the following two algorithms, we assume that two players are engaged in the entire encounter. For simplicity and clarity, we denote "Player A" for one player and "Player B" for the other.

Algorithm 2(Two-player Paradigm):

Stage I: **(1)** Using environmental information, identify player B's strategy.

(2) Based on the estimate of player B's movement or strategy, generate a new control law for player A by using optimal control theory.

Stage II: **(1)** Using environmental information, identify player A's strategy.

(2) Based on the estimate of player A's strategy, generate a new control law for player B using optimal control theory.

Last, we give an algorithm based on the discussions of the last section. In this case, the optimal control in stage II could be very complicated and highly nonlinear even though the original system is linear. This will be illustrated in a pursuit-evasion game problem in the next section.

Suppose that the system for two players is described by

$$\dot{\mathbf{x}} = f(\mathbf{x}, \phi, \psi), \tag{3.30}$$

where $f(\mathbf{x}, \phi, \psi)$ is a vector of differentiable functions, $\mathbf{x}$ is the state vector of dimension n (n is a positive integer), ϕ is the control strategy for player A and ψ is the control strategy for player B.

The goal for both players is to maximize (or minimize) a cost function given by $J(\phi, \psi)$

$$\max_{\psi} \min_{\phi} J(\phi, \psi) \tag{3.31}$$

Again, we use $J_{\hat{\psi}}$ to denote the optimal value for $J(\phi, \hat{\psi})$ if $\hat{\psi}$, the estimate of player B's strategy, is known, that is

$$J_{\hat{\psi}} = \min_{\phi} J(\phi, \hat{\psi}). \tag{3.32}$$

With these notations, we have the following:

Algorithm 3 (Optimal Control Paradigm):

Stage I: **(1)** Using the environmental information, identify player B's strategy.

(2) Based on the estimate of player B's strategy, generate a control strategy by using optimal control theory.

Stage II: From the strategy used in stage I, construct a new problem

$$\max_{\psi} J_{\psi} \ (or \ \min_{\phi} J_{\phi}),$$

$$\text{subject to} \quad \dot{\mathbf{x}} = f(\mathbf{x}, \phi, \psi),$$

3.5. A Pursuit-Evasion Game Problem

3.5.1. The Problem

In a pursuit-evasion game[12], the pursuer's control is his acceleration, $a_p(t)$, normal to the initial line of sight (ILOS) to the evader. The evader's control is his acceleration, $a_e(t)$, also normal to the ILOS. The relative velocity along the ILOS is such that the normal time of closest approach is t_f. If v(t) is the relative velocity normal to the ILOS, and y(t) is the relative displacement normal to the ILOS, the equations of motion are

$$\dot{v} = a_p - a_e, \qquad v(t_0) = v_0; \tag{3.33}$$

$$\dot{y} = v, \qquad y(t_0) = 0. \tag{3.34}$$

The pursuer wishes to minimize the terminal miss, $|y(t_f)|$, whereas the evader wishes to maximize it, so the performance index may be taken as

$$J = \frac{1}{2}[y(t_f)]^2. \tag{3.35}$$

The accelerations of the pursuer and the evader are limited

$$\begin{aligned} |a_p| &\leq a_{pm}, \\ |a_e| &\leq a_{em}, \end{aligned} \tag{3.36}$$

where $a_{pm} > a_{em}$. The solution proceeds by first forming the Hamiltonian

$$H = \lambda_v(a_p - a_e) + \lambda_y v. \tag{3.37}$$

The adjoint equations are then

$$\dot{\lambda}_v = -\lambda_y, \qquad \lambda_v(t_f) = 0, \tag{3.38}$$

$$\dot{\lambda}_y = 0, \qquad \lambda_y(t_f) = y(t_f), \tag{3.39}$$

and the optimality conditions are

$$a_p(t) = -a_{pm}\text{sgn}\lambda_v, \tag{3.40}$$

$$a_e(t) = -a_{em}\text{sgn}\lambda_v. \tag{3.41}$$

The adjoint equations are easily integrated to yield

$$\lambda_v(t) = (t_f - t)y(t_f), \tag{3.42}$$

$$\lambda_y(t) = y(t_f) = \text{const.} \tag{3.43}$$

It is, therefore, clear that

$$\text{sgn } \lambda_v(t) = \text{sgn } y(t_f) = \text{const.} \tag{3.44}$$

Substituting eq. (3.44) into eq. (3.40) and eq. (3.41), and eqs. (3.40), (3.41) into eq. (3.33) and eq. (3.34) yields a simple set of differential equations whose solution may be written as

$$y(t_f) = v_0(t_f - t_0) - \frac{1}{2}(a_{pm} - a_{em})(t_f - t_0)^2\text{sgn } y(t_f), \tag{3.45}$$

which can be used for determining $y(t_f)$. Thus, we have

$$y(t_f) = \begin{cases} \frac{1}{2}(t_f - t_0)^2[\frac{2v_0}{t_f - t_0} - (a_{pm} - a_{em})] & \text{if } \frac{2v_0}{(t_f - t_0)(a_{pm} - a_{em})} > 1 \\ -\frac{1}{2}(t_f - t_0)^2[\frac{-2v_0}{t_f - t_0} - (a_{pm} - a_{em})] & \text{if } \frac{2v_0}{(t_f - t_0)(a_{pm} - a_{em})} < -1. \end{cases}$$

Substituting the above equation into eq. (3.40) yields the control for the pursuer as follows

$$a_p(t) = \begin{cases} -a_{pm} & \text{if } a_1 > 1, \\ a_{pm} & \text{if } a_1 < -1, \\ a_e + \frac{A - v(t_0)}{t_f - t_0} & \text{if } -1 < a_1 < 1, \end{cases} \tag{3.46}$$

where

$$a_1 = \frac{2v_0}{(t_f - t_0)(a_{pm} - a_{em})},$$

$$A = -v(t_0) - \frac{2y(t_0)}{t_f - t_0}.$$

3.5.2. Solution By Using Neural Networks

From the last section, we know that the differential game solution, for a_p is

$$a_p(t) = -a_{pm}\text{sgn}[y(t_f)],. \tag{3:47}$$

The one-player paradigm algorithm will be used for our study in which a_e is either a fixed value or a continuous function of time (see Figure 3.7).

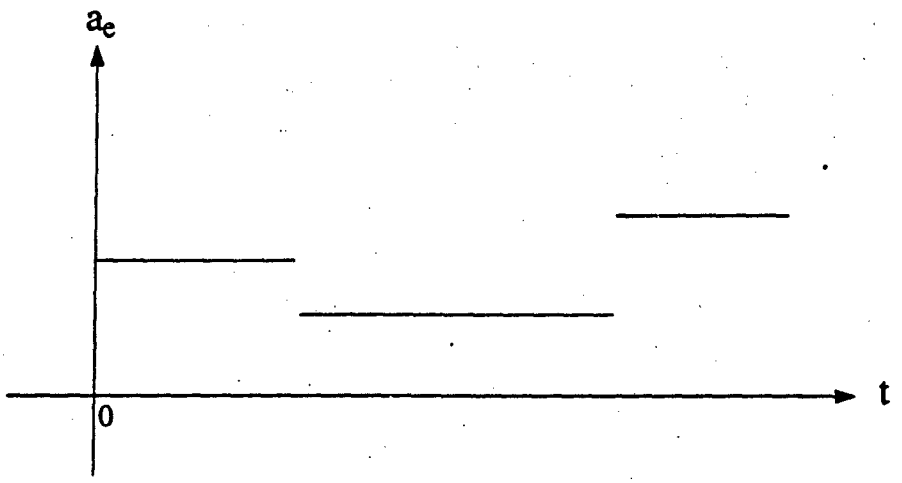


Figure 3.7: A Typical Function for a_e

In what follows, we assume that $a_e(t)$ is an independent variable. We shall show that the control strategy $a_p(t)$ for the pursuer is actually a function of $a_e(t)$. From equation (3.47), one can readily see that $a_p(t)$ is constant over the interval (t_0, t_f). The sign of $a_p(t)$ depends on the sign of $y(t_f)$. Therefore, the explicit expression for $y(t)$ is necessary.

From eq. (3.33), we know

$$v(t) = v(t_0) + \int_{t_0}^{t} a_p(\tau)d\tau - \int_{t_0}^{t} a_e(\tau)d\tau, \tag{3.48}$$

which yields from eq. (3.47)

$$v(t) = v(t_0) - a_{pm}\text{sgn}[y(t_f)](t - t_0) - \int_{t_0}^{t} a_e(\tau)d\tau. \tag{3.49}$$

Letting

$$\tilde{a}_e(t) = \int_{t_0}^{t} a_e(\tau)d\tau,$$

we have

$$v(t) = v(t_0) - a_{pm}\text{sgn}[y(t_f)](t - t_0) - \tilde{a}_e(t). \tag{3.50}$$

It follows from eqs. (3.34) and (3.50) that

$$\begin{aligned} y(t) &= \int_{t_0}^{t} v(\tau)d\tau \\ &= \int_{t_0}^{t} [v(t_0) - a_{pm}\text{sgn}[y(t_f)](\tau - t_0) - \tilde{a}_e(\tau)]d\tau \\ &= v(t_0)(t - t_0) - \frac{a_{pm}}{2}\text{sgn}[y(t_f)](t - t_0)^2 - \int_{t_0}^{t} \tilde{a}_e(\tau)d\tau. \end{aligned} \tag{3.51}$$

Let

$$\bar{a}_e(t) = \int_{t_0}^{t} \tilde{a}_e(\tau)d\tau,$$

which is equivalent to

$$\begin{aligned} \bar{a}_e(t) &= \int_{t_0}^{t} \tilde{a}_e(\tau)d\tau \\ &= \int_{t_0}^{t}\int_{t_0}^{\tau} a_e(\rho)d\rho d\tau. \end{aligned} \tag{3.52}$$

It follows from eqs. (3.51) and (3.52) that y(t) depends on the time function $a_e(t)$ ($t_0 \leq t \leq t_f$), which further implies from eq. (3.47) that $a_p(t)$ ($t_0 \leq t \leq t_f$) depends on the function $a_e(t)$. This observation further justifies the statement of phase 3 in the general configuration in Section 3.1.

Next, let us consider the optimization problem

$$\min_{a_p} J(t_0, a_p, \hat{a}_e), \tag{3.53}$$

$$\text{subject to} \quad \dot{v} = a_p(t) - \hat{a}_e, \qquad v(t_0) = v_0, \tag{3.54}$$

$$\dot{y} = v, \qquad y(t_0) = y_0. \tag{3.55}$$

Its solution is denoted by $a_p^*(t)$, which is a functional of $\hat{a}_e(t)$. Please note that in this optimization problem $\hat{a}_e(t)$ is the estimate of $a_e(t)$.

We further consider the optimization problem

$$\max_{a_e} J(t_0, a_p^*, a_e) \tag{3.56}$$

$$\text{subject to} \quad \dot{v} = a_p^*(t) - a_e(t), \qquad v(t_0) = v_0, \tag{3.57}$$

$$\dot{y} = v, \qquad y(t_0) = y_0. \tag{3.58}$$

If we can denote $\text{sgn}[y(t_f)]$ by $S_{a_e}(t)$ which apparently means a functional of $a_e(t)$, it is interesting to see that

$$a_p^*(t) = -a_{pm} S_{a_e}(t), \tag{3.59}$$

and thus the system equations for the optimization problem (3.56), (3.57) and (3.58) becomes

$$\begin{aligned} \dot{v} &= a_p^*(t) - a_e(t) \\ &= -a_{pm} S_{a_e}(t) - a_e(t) \\ &= f(a_e(t)), \end{aligned} \tag{3.60}$$

and

$$\dot{y} = v(t). \tag{3.61}$$

Therefore, we know from eq. (3.60) that the optimization problem (3.56), (3.57) and (3.58) now becomes a nonlinear optimization problem even though the original one is linear.

Solving the above nonlinear optimization problem for an arbitrary function $a_e(t)$ is usually very difficult. In our case, we only consider the following function

$$a_e(t) = -a_e \text{sgn}[y(t_f)], \tag{3.62}$$

where a_e is assumed to have an arbitrary value between zero and a_{em}.

Substituting eq. (3.62) into eq. (3.52), we have

$$\begin{aligned} \tilde{a}_e(t) &= -\int_{t_0}^{t}\int_{t_0}^{\tau} a_e \text{sgn}[y(t_f)] d\rho d\tau \\ &= -a_e \text{sgn}[y(t_f)] \frac{1}{2}(t-t_0)^2, \end{aligned} \tag{3.63}$$

which, together with eq. (3.51), gives

$$y(t) = v(t_0)(t-t_0) - \frac{1}{2}a_{pm}\text{sgn}[y(t_f)](t-t_0)^2 + a_e \text{sgn}[y(t_f)]\frac{1}{2}(t_f-t_0)^2. \tag{3.64}$$

By comparing eq. (3.45) with eq. (3.64) , we can clearly see the reason why we have chosen this special form (3.62). The only difference between eq. (3.45) and eq. (3.64) is that the maximal value of $a_e(t)$ is used in eq. (3.45). An immediate benefit from this difference is that the formula of how to determine the sign of y(t) has the same form as that of the last section. Thus, by replacing a_{em} by a_e, we can use the eq. (3.45) to determine the sign of $y(t_f)$. Although we have such a nice representation when the special form (3.62) is used, we do not have to confine ourselves to it. In fact, any type of integrable function may be used for $a_e(t)$.

3.5.3. Neural Identifier

From previous subsections, it is known that $a_p(t)$ is a functional of $a_e(t)$. Thus, identification of a_e is essential for our work when a_e is unknown. In this subsection, the identification is accomplished by a Neural Identifier which is realized by a trained neural network.

In our problem, a neural identifier is a generic mapping between two domains. The first one is the Environmental Information Domain (EID) which is usually accessible. For our problem, the other domain is the one-dimensional space **R** if the system which we are considering is a single-input control system. The environmental information from EID is usually the numerical values of observable state variables or curves of state variables. The mapping is so defined that for each piece of information from EID there is one and only one point in **R** which uniquely determines that piece of information. The function of the Neural Identifier is thus to determine the point in **R** which yields that piece of information. The way to determine the point in **R** yielding the environmental information is essentially the method of training the network. We shall discuss the Phase-plane training method, which will be used in our simulations.

Phase-plane Method

The function of the Neural Identifier is to estimate the strategy for the evader, $a_e(t)$, based on the environmental information. In view of the fact that a player performing identification cannot make any control decision, we may assume that $a_p = 0$ (or any other real number) during the Identification Process. Then, we have the following simplified system equations

$$\dot{v} = -a_e, \tag{3.65}$$

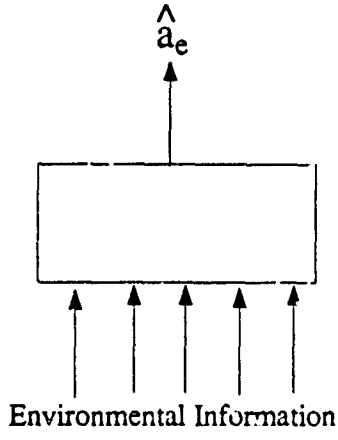


Figure 3.8: General Diagram for Neural Identifier

$$\dot{y} = v, \tag{3.66}$$

which yields

$$v(t) = v(t_0) - a_e(t - t_0), \tag{3.67}$$

and

$$\begin{aligned} y(t) &= y(t_0) + \int_{t_0}^{t} v(\tau)d\tau \\ &= y(t_0) + \int_{t_0}^{t} (v(t_0) - a_e\tau + a_e t_0)d\tau \\ &= y(t_0) + v(t_0)(t - t_0) - \frac{a_e}{2}(t - t_0)^2. \end{aligned} \tag{3.68}$$

Substituting eq. (3.67) into eq. (3.68) gives

$$y(t) = y(t_0) + v(t_0)(-\frac{v(t) - v(t_0)}{a_e}) - \frac{a_e}{2}\frac{(v(t) - v(t_0))^2}{a_e^2}. \tag{3.69}$$

Denoting

$$\begin{aligned} \Delta y(t_0) &= y(t) - y(t_0), \\ \Delta v(t_0) &= v(t) - v(t_0), \end{aligned}$$

we have

$$\Delta y(t_0) = -\frac{v(t_0)}{a_e}\Delta v(t_0) - \frac{1}{2a_e}\Delta^2 v(t_0), \tag{3.70}$$

which yields

$$a_e = -\frac{\Delta v(t_0)}{2\Delta y(t_0)}(\Delta v(t_0) + 2v(t_0)). \tag{3.71}$$

Equation (3.71) represents the relation for identifying the enemy's strategy.

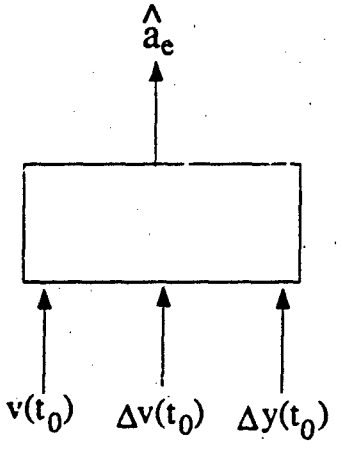


Figure 3.9: Phase-plane Method for Identification

As mentioned before, typical types of environmental information are the numerical values of the observable state variables or the curves generated by them. A trajectory of phase plane for positive values of a_e is shown in Figure 3.10. Every time a pursuer performs the identification using the Neural Identifier only the increments of state variables y(t) and v(t) at the time when he begins the identification and the value of v(t) at that time are fed into the network. The output of the network will be the estimate of a_e.

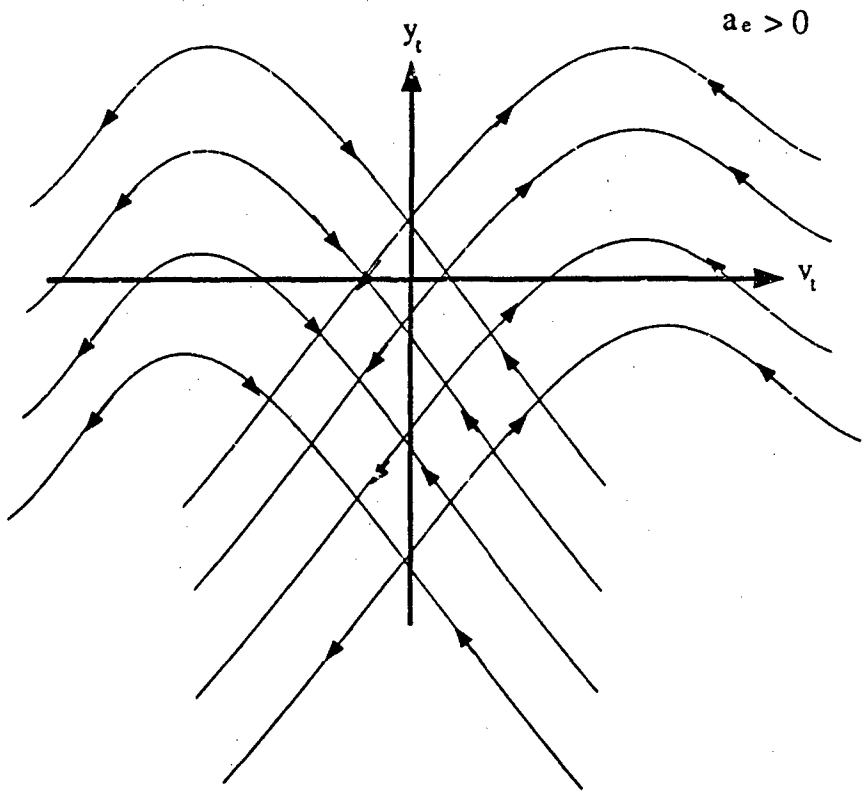


Figure 3.10: Trajectories In the Phase Plane

3.5.4. Training the Network

A diagram of the Neural Identifier is depicted in Figure 3.11. As mentioned above, we will use the Phase-plane method to generate the training set for the Neural Identifier. For our convenience, we rewrite the formula as follows

$$\hat{a}_e = -\frac{\Delta\bar{v}_{t_0}}{2\Delta\bar{y}_{t_0}}(\Delta\bar{v}_{t_0} + 2\bar{v}_{t_0}), \tag{3.72}$$

where the quantities $\Delta\bar{v}_{t_0}, \bar{v}_{t_0}$ and $\Delta\bar{y}_{t_0}$ are used in place of $\Delta v_{t_0}, v_{t_0}$ and Δy_{t_0} for the reason below.

The reason why we would use $\bar{v}_{t_0}, \Delta\bar{v}_{t_0}$ and $\Delta\bar{y}_{t_0}$ in place of v_{t_0}, Δv_{t_0} and Δy_{t_0} for identifying the strategy a_e is that the information coming from any practical

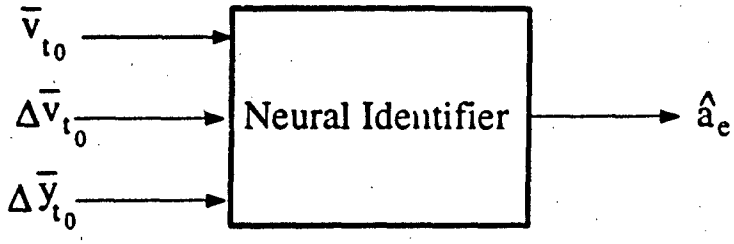


Figure 3.11: Block Diagram for Neural Identifier

system is always corrupted by some type of noise although the noise might be small. This substitution is illustrated in Figure 3.12 where the observable state variables are converted by some device, called device "I", into some variables suitable for the purpose of identification.

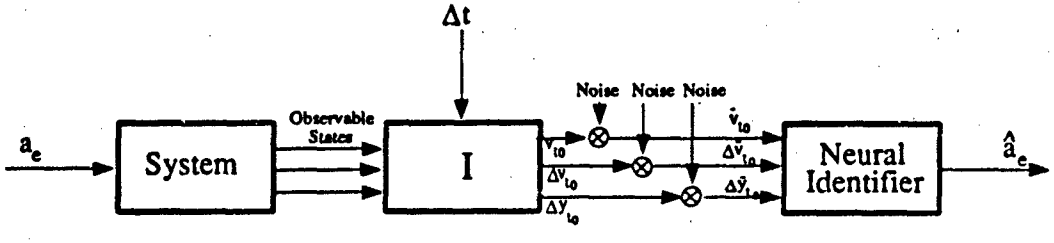


Figure 3.12: Structure for Generating the Training Set

In the ideal case in which the noise is zero, i.e. $\bar{v}_{t_0} = v_{t_0}$, $\Delta\bar{v}_{t_0} = \Delta v_{t_0}$, $\Delta\bar{y}_{t_0} = \Delta y_{t_0}$, we shall have from eq. (3.72) that

$$\begin{aligned}
\hat{a}_e &= -\frac{\Delta\bar{v}_{t_0}}{2\Delta\bar{y}_{t_0}}(\Delta\bar{v}_{t_0} + 2\bar{v}_{t_0}) \\
&= -\frac{\Delta v_{t_0}}{2\Delta y_{t_0}}(\Delta v_{t_0} + 2v_{t_0}) \\
&= -\frac{\Delta v_{t_0}}{2}(\Delta v_{t_0} + 2v_{t_0})(-\frac{2a_e}{\Delta v_{t_0}(2\Delta v_{t_0} + \Delta v_{t_0})}) \\
&= a_e. \qquad (3.73)
\end{aligned}$$

Thus, $\hat{a}_e$ becomes a_e in the ideal case. The above discussion implies that the Neural Identifier functions as an inverse system of the original system. Therefore,

it follows that we need to use the Phase-plane Method to generate the training set for the Neural Identifier with noise-corrupted inputs. The necessity of this assumption will be shown in the following discussion. We know that there will always be some errors, no matter how small they are, between the ideal mapping of equation (3.71) and the one approximated by the Neural Identifier. These errors can be viewed as some sort of noise added to the system in the assumption that the Neural Identifier is an ideal mapping of equation (3.71). Thus, we may assume that the Neural Identifier is an ideal mapping for the equation (3.71), and the errors between the ideal mapping of the equation (3.71) and the actual identifier is due to inputing imperfect information (corrupted by noise). This idea provides a basis for the method of generating the training set for the Neural Identifier. We simulated the noises by generating some random numbers with a given variation. The random numbers are assigned to v_{t_0}, and thus $\bar{v}_{t_0}$ is obtained. Note that $\Delta\bar{v}_{t_0} = \Delta v_{t_0} = 1$ since $a_e = 1$ and $a_p = 0$ during the Identification Process and $\Delta\bar{y}_{t_0}$ is the output of the system driven by a_e (see Figure 3.13). The values of $\bar{v}_{t_0}, \Delta\bar{v}_{t_0}$ and $\Delta\bar{y}_{t_0}$ are put in each input line of the training set and the value computed using formula (3.71) is the desired output of the Neural Identifier, corresponding to those particular $\bar{v}_{t_0}, \Delta\bar{v}_{t_0}$ and $\Delta\bar{y}_{t_0}$. Repeating these input lines and desired output lines forms a training set for the Neural Identifier. Please note that, instead of trying to approximate a mapping of eq. (3.71) for arbitrary inputs, we approximate the mapping with some specific domains which are the outputs of the original system driven by a_e.

The structure and parameters of the Neural Identifier are summarized as follows:

Type of neural network: Hetero-Association,

Control strategy: Backpropagation,

Learning rule: Delta-rule,

Transfer function: Sigmoid,

Scale: 1.0,

Summation: Sum,

Learning rate: Coe1 = 0.9, Coe2 = 0.6.

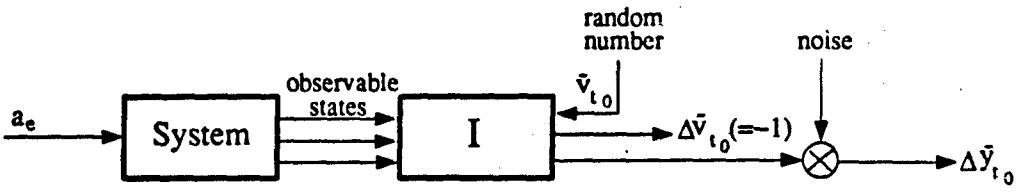


Figure 3.13: Generating Data for Neural Identifier

3.5.5. Simulation Results

The simulation works are done in a Sun 4/260 workstation under the environment of UNIX. All programs are written in C and the outputs are numerical values. The parameters of the system used in the simulation are summarized in Table 3.1. In our simulations, the value of a_e is fixed and the pursuer is to identify the value a_e. As shown in Figure 3.14, the simulation is carried out in several different periods (that is, period 1, period 2, etc.). In each period, the index is incremented from one to the maximum value *state_I_limit*. The goal for the pursuer is to compute control values based on the estimate of a_e such that the value of y at the end point of each period is minimized. In each individual period, the first interval (denoted by I in Figure 3.14) is for the pursuer to obtain the estimate of a_e and, in our simulation, $a_p = 0$. The maximum number of periods in entire simulation is the

greatest integer less than *global_time_limit*. Therefore, our simulation is virtually different from that in the original work.

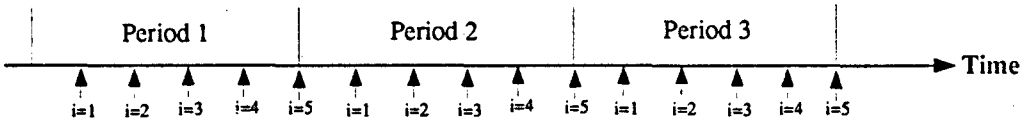


Figure 3.14: Simulation Periods

The flow chart of the program is shown in Figure 3.16.

Table 3.1: Parameters Used in Simulation

a_e	1.0
stage_I_limit	5
global_time_limit	5
a_{pm}	2.0

For comparison of the ideal case where there are no noises with those corrupted by noises of different variations, we have done the following experiments. They are:

Case [i] No neural networks, no noises, (simu_back.c, see Figure 3.15 (a));

Case [ii] Noise with variation of 144 for v_{t_0}, (simu3.c, see Figure 3.15 (b));

Case [iii] Noise with variation of 144 (uniform distribution [-6 , 6], for v_{t_0} and noise with variation of 0.04 (uniform distribution [-0.1 , 0.1] for $\Delta\bar{y}_{t_0}$ (simu4.c, see Figure 3.15 (c));

Case [iv] Constant (- 5.0) disturbance in v_{t_0} and constant (+ 5.0) disturbance in $\Delta\bar{y}_{t_0}$(simu3_d.c, see Figure 3.15 (d));

Case [v] Comparison of the following cases (see Figure 3.18):

(a) no disturbance (solid line),

(b) the case [iv] (dash line),

(c) same as case [iv] except that -10.0 for v_{t_0} and +10.0 for $\Delta\bar{y}_{t_0}$ (dashdot line),

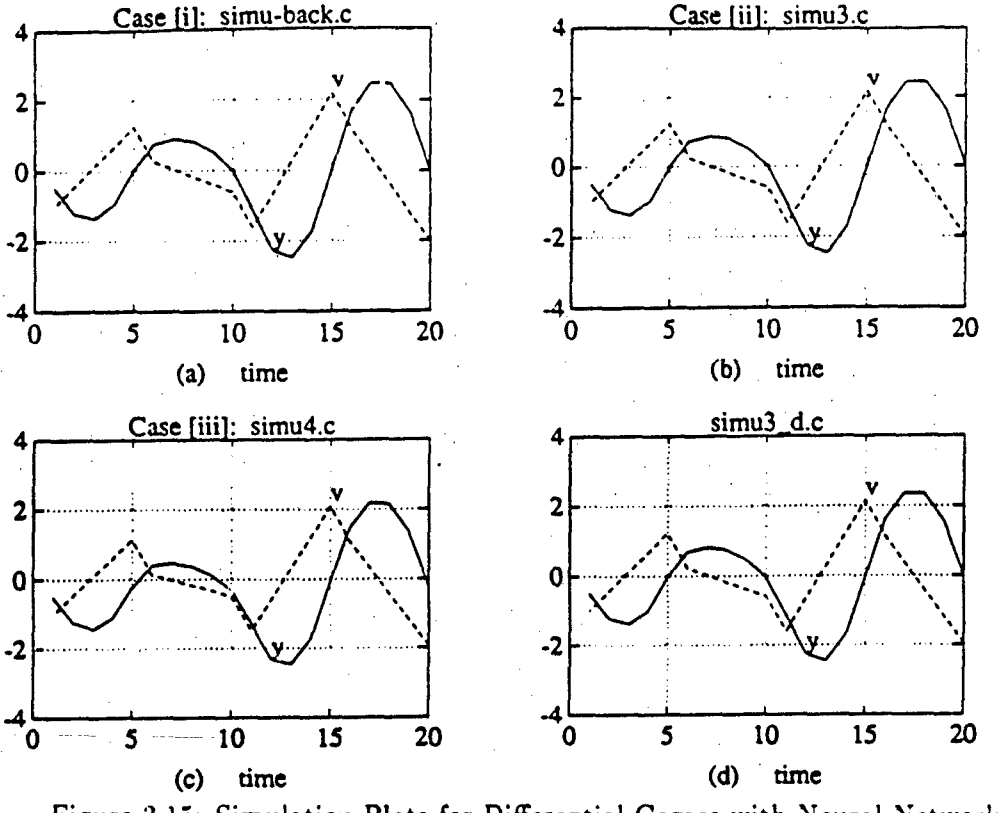


Figure 3.15: Simulation Plots for Differential Games with Neural Networks

The output data for these different cases are shown in Table 3.2. In Table 3.2, $\hat{a}_e$ denotes the estimate of a_e and a_p^* is the optimal control value for the pursuer. Using the parameters in Table 3.1, we know that at steps 0, 5, 10, 15, 20 the pursuer gets the estimate of a_e. Then, in the first period (step 1 through step 5), in the second period 9step 6 through step 10), in the third period (step 11 through step 15), and in the fourth period 9step 16 through step 20), the pursuer computes the optimal control value a_p^* fro each of these periods. The values of the

states y and v are also gievn in Table 3.2. Shown in Figure 3.15 are the curves plotted using Mat-lab software. The codes for the five cases are similar except for minor differences.

Table 3.2: Output Data for Simulation

Time_step	Simu_back				Simu3				Simu4				Simu3_d			
	$\hat{a}_e$	a_p^*	y	v	$\hat{a}_e$	a_p^*	y	v	$\hat{a}_e$	a_p^*	y	v	$\hat{a}_e$	a_p^*	y	v
1	1	1.562	-0.5	-1.0	0.999	1.562	-0.5	-1.0	0.971	1.534	-0.5	-1.0	0.995	1.558	-0.5	-1.0
2	1	1.562	-1.219	-0.438	0.999	1.562	-1.219	-0.438	0.971	1.534	-1.233	-0.466	0.995	1.558	-1.221	-0.442
3	1	1.562	-1.375	0.125	0.999	1.562	-1.376	0.124	0.971	1.534	1.433	0.067	0.995	1.558	-1.385	0.115
4	1	1.562	-0.968	0.688	0.999	1.562	-0.972	0.685	0.971	1.534	-1.099	0.601	0.995	1.558	-0.99	0.673
5	1	1.562	0	1.25	0.999	1.562	-0.006	1.247	0.971	1.534	-0.231	1.134	0.995	1.558	-0.039	1.231
6	1	0.781	0.75	0.25	0.999	0.783	0.642	0.247	0.962	0.845	0.404	0.135	0.995	0.793	0.692	0.231
7	1	0.781	0.59	0.031	0.999	0.783	0.88	0.03	0.962	0.845	0.461	-0.020	0.995	0.793	0.819	0.0241
8	1	0.781	0.813	-0.188	0.999	0.783	0.502	-0.1872	0.962	0.845	0.364	-0.175	0.995	0.793	0.74	-0.182
9	1	0.781	0.516	-0.406	0.999	0.783	0.507	-0.404	0.962	0.845	0.111	-0.33	0.995	0.793	0.454	-0.389
10	1	0.781	0	-0.625	0.999	0.783	-0.006	-0.621	0.962	0.845	-0.296	-0.485	0.995	0.793	-0.03797	-0.596
11	1	1.953	-1.125	-1.625	0.999	1.951	-1.126	-1.621	0.988	1.89	-1.281	-1.485	0.995	1.935	-1.134	-1.596
12	1	1.953	-2.273	-0.672	0.999	1.951	-2.272	-0.67	0.988	1.89	-2.321	-0.594	0.995	1.935	-2.262	-0.661
13	1	1.953	-2.469	0.281	0.999	1.951	-2.467	0.280	0.988	1.89	-2.47	0.296	0.995	1.935	-2.455	0.274
14	1	1.953	-1.711	1.234	0.999	1.951	-1.712	1.231	0.988	1.89	-1.728	1.187	0.995	1.935	-1.715	1.208
15	1	1.953	0	2.188	0.999	1.951	-0.006	2.181	0.988	1.89	-0.096	2.077	0.995	1.935	-0.039	2.143
16	1	0.195	1.688	1.188	0.999	0.199	1.676	1.181	0.971	0.248	1.481	1.077	0.995	0.223	1.604	1.143
17	1	0.195	2.473	0.383	0.999	0.199	2.456	0.380	0.971	0.248	2.192	0.325	0.995	0.223	2.359	0.366
18	1	0.195	2.453	-0.422	0.999	0.199	2.437	-0.420	0.971	0.248	2.131	-0.427	0.995	0.223	2.337	-0.41
19	1	0.195	1.629	-1.227	0.999	0.199	1.616	-1.221	0.971	0.248	1.327	-1.18	0.995	0.223	1.538	-1.187
20	1	0.195	0	-2.031	0.999	0.199	-0.005	-2.022	0.971	0.248	-0.229	-1.932	0.995	0.223	-0.037	-1.963

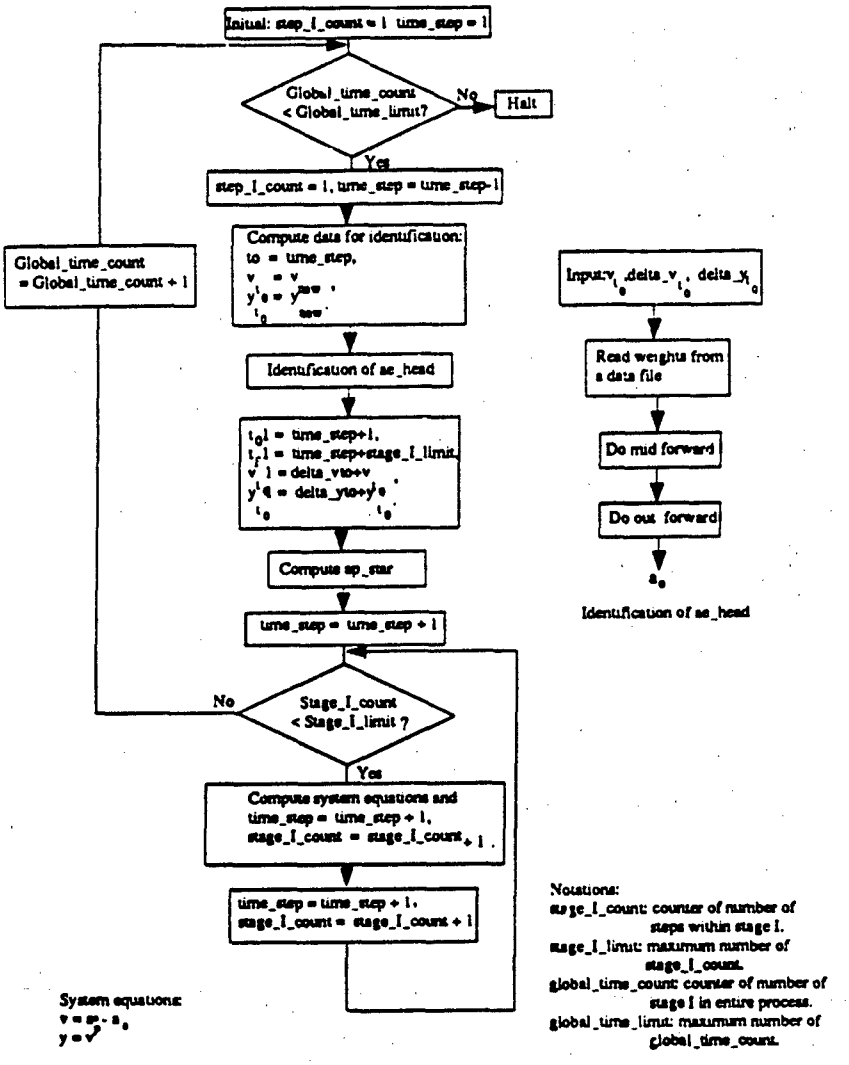


Figure 3.16: Flow Chart

3.5.6. Discussion and Conclusion

As mentioned before, the goal for the pursuer is to minimize the values of y at step 5 (for the first period), at step 10 (for the second period), at step 15 (for the third period), and at step 20 (for the fourth priod). For comparison, the simulation result for the same problem without using neural networks is also

given (see *Simu_back*). As shown in Figure 3.15, there always exists a jump in y between the beginning and the end of each "Identification Process" (please notice that the values of y(5) and y(6), the values of y(10) and y(11), and the values of y(15) and y(16), in each of the first four cases). This is because we set $a_e = 0$ during the "Identification Process", therefore, the system runs freely. One may argue that the magnitude of the jump in y in the "Identification Process" can be reduced if we keep a_p at the previous value, instead of zero. But it turns out not to be true (see Figure 3.17).

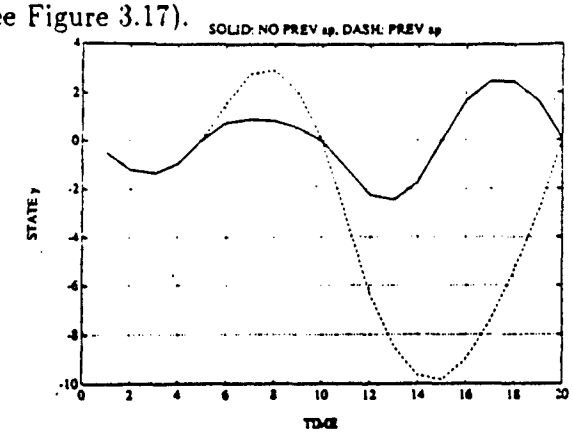


Figure 3.17: Programs Simu_back.c Without Previous a_p and That With Previous a_p

Different simulation results will be compared here (see Figure 3.15 and Figure 3.18). The result for program Simu3.c exhibits the best performance among the experiments done in the sense that the value $|y(t'_f)|$ has achieved the minimum (see y(5), y(10), y(15) and y(20) respectively from the table). The results are reasonable because in program Simu3 only the quantity $\Delta\bar{v}_{t_0}$ is disturbed by some noise while in program Simu4.c both quantities of $\Delta\bar{v}_{t_0}$ and $\Delta\bar{y}_{t_0}$ are disturbed by noises, and additional constant disturbances are added to v_{t_0} and $\Delta\bar{y}_{t_0}$ in program Simu3.d.c. The results are also explained partly by the accuracy of

the estimate of a_e, i.e. $\hat{a}_e$. Since the program Simu3.c has the highest accuracy, it has a better performance than the others.

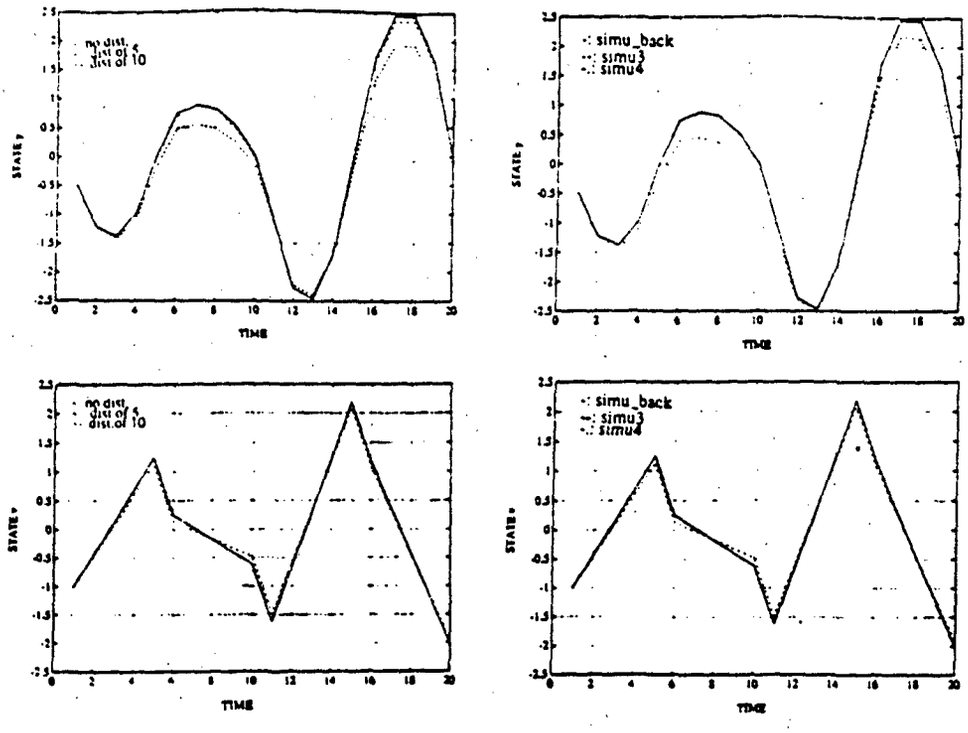


Figure 3.18: Comparison of Simulation Results

The simulation results show the feasibility of this approach. One may like to apply the algorithms presented in this work in some more complicated cases: highly nonlinear, corrupted by noises or even with uncertainty. What should be kept in mind is that both sides of the game can implement the same process. Therefore, in some sense, whoever has the better information about what his opponent is doing or a more accurate estimate of his opponent's strategy would eventually win the game. Using the principle outlined in the general configuration of this chapter, we can also implement the other algorithms presented in this chapter, e.g. "Two-player Algorithm", in the same or even more complicated systems. When carrying out this further research, we should always keep in mind

that the approach presented here is no longer the same as traditional approaches to differential game problems. It deals with a broader range of problems because it allows both sides of the players to take ANY strategy. For this reason, it can be expected that more properties exist about the optimal trajectories, controls and feasible sets, which may also be a direction for further research.

3.6. Learning Algorithm of Feedback Control

In the previous sections, the approach to differential games with neural networks, different paradigms and a case study example have been thoroughly discussed. The approach is based on the paradigm of semantic control[76]. Two neural networks, called Neural Identifier and Neural Controller, were used in each of these paradigms. The neural identifier identifies the control strategy of the opponent player based on the environmental information. The neural controller, on the other hand, gives the control strategy for the own player. The subsequent sections mainly discuss how to construct and to train the neural controller. A rigorous mathematical derivation for weight updating rule of the neural controller will be given.

In [7], a feedback control law in the class of L-layer neural networks is given to control the discrete-time system such that viability conditions are satisfied. The viability conditions are described by a subset K of the state space such that $d_K(x_{n+1}) = 0$ for all $n \geq 0$, where $d_K(x_{k+1})$ is the distance between the state x_{k+1} and the subset K. The strategy underlying the external algorithm is to apply the gradient method for the minimization of d_K. The network learns or will learn whenever the states lie outside of the domain K.

An extension to the differential game problems is considered. Based on the one-player paradigm described in Section 3.2, a differential game problem can be converted to an optimization problem provided that the estimate of the control strategy for the enemy has been obtained. A similar approach to that in [7] will be used for the differential game problems discussed in the previous sections. However, instead of minimizing d_K, minimizing an arbitrary cost function J is considered. A detailed formula for updating the weights for three-layer networks will be given.

3.6.1. The Updating Rule

Let us now consider a discrete-time version of the differential game problem, defined as follows

$$\max_{\psi_n} \min_{\phi_n} J_{n+1}(\mathbf{x_{n+1}}, \psi_n, \phi_n), \tag{3.74}$$

subject to the difference equation

$$\mathbf{x_{n+1}} = f(\mathbf{x_n}, \psi_n, \phi_n), \qquad \mathbf{x_0} \text{ is given}, \tag{3.75}$$

where $\phi_n = (\phi_{1,n}, \cdots, \phi_{l,n})$, $\psi_n = (\psi_{1,n}, \cdots, \psi_{m,n})$, $\mathbf{x_n} = (x_{1,n}, x_{2,n}, \cdots, x_{d,n})$, $\psi_n \in \Psi$, $\phi_n \in \Phi$, $\mathbf{x_n} \in$ X, X is the state space, Ψ and Φ are the feasible subspaces in the control space Z for the control strategies ψ_n and ϕ_n respectively, and f is a C^1 mapping from X $\times$ Z to X.

We can define a quantity $J_{n+1,\hat{\psi}_n}$, belonging to a subspace of the value space of J, as follows

$$J_{n+1,\hat{\psi}_n}(\mathbf{x_{n+1}}, \phi_n) = J_{n+1}(\mathbf{x_{n+1}}, \hat{\psi}_n, \phi_n), \tag{3.76}$$

where $\hat{\psi}_n$ is the estimate value for ψ_n.

Having obtained $J_{n+1,\hat{\psi}_n}$, we can state the original problem as follows

$$\min_{\phi_n} J_{n+1,\hat{\psi}_n}(\mathbf{x_{n+1}}, \phi_n) = \min_{\phi_n} J_{n+1}(\mathbf{x_{n+1}}, \hat{\psi}_n, \phi_n), \tag{3.77}$$

subject to

$$\mathbf{x_{n+1}} = f(\mathbf{x_n}, \hat{\psi}_n, \phi_n), \tag{3.78}$$

where $\phi_n \in \Phi$. We are now trying to find ϕ_n such that $J_{n+1,\hat{\psi}_n}(\mathbf{x_{n+1}}, \phi_n)$ is maximized. A feedback control law is obtained through an L-layer neural network

$$\phi_n = \Phi_L(W_1(n), W_2(n), \cdots, W_L(n), \mathbf{x_n}), \tag{3.79}$$

where $\Phi_L(W_1(n), W_2(n), \cdots, W_L(n), \mathbf{x_n})$ denotes the propagation of a signal x in a neural network, and $W_k(n)$ is the synaptic matrix associated with the network layers k-1 and k at step n. Let us denote the layer i by P^i and the number of nodes for layer P^i by n_i, i=0, ..., L. For a single input control system, the number of the output for the neural network should be one, that is, $n_L = 1$. Using the same notation, we can easily see that $W_i(n) \in R^{n_i \times n_{i-1}}$, i=1, ..., L. For compactness, denote the weight matrix by $W(n) = (W_1(n), W_2(n), ..., W_L(n))$. Each $W_\eta(n)$ can also be written as $W_\eta(n) = (w_{ij}^\eta(n))_{n_\eta \times n_{\eta-1}}$, η=1, ..., L. Moreover, let us define the output of the ith neuron in the jth layer by x_i^j. Thus, the output of the first neuron in the last layer should be denoted by x_1^L. The transfer function for the ith neuron in the jth layer is denoted by $g_{j-1}^i(.)$. Thus, if we consider a single-output network, the transfer function for the single-output neuron is $g_{L-1}^1(.)$. or for short, $g_{L-1}(.)$. For our convenience, we also use the compact form $g_L(.)$ to denote $(g_L^1(.), \cdots, g_L^{n_L}(.))$.

Having these notations, we can define a gradient learning rule, which is actually the back-propagation learning rule,

$$w_{ij}^\eta(n+1) = w_{ij}^\eta(n) - \alpha_{ij}^\eta \frac{\partial J_{n+1,\hat{\psi}_n}}{\partial w_{ij}^\eta(n)}, \tag{3.80}$$

where α_{ij}^{η} is the gradient step factor. For this standard formula, computational burden is mainly the calculation of $\frac{\partial J_{n+1,\dot{\psi}_n}}{\partial w_{ij}^{\eta}(n)}$ whose computational load will increase dramatically as L becomes larger.

For simplicity, we only consider the case in which the system (3.75) is a single-input system. In this case, $n_L = 1$. Note that

$$\frac{\partial J_{n+1,\dot{\psi}_n}}{\partial w_{ij}^{\eta}(n)} = \sum_{k=1}^{d} \frac{\partial J_{n+1,\dot{\psi}_n}}{\partial x_{k,(n+1)}} \frac{\partial f_k}{\partial w_{ij}^{\eta}(n)}. \tag{3.81}$$

and

$$\frac{\partial f_k}{\partial w_{ij}^{k}(n)} = \frac{\partial f_k}{\partial \phi_{1,n}} \frac{\partial \Phi_L}{\partial w_{ij}^{\eta}}. \tag{3.82}$$

Therefore, the key step is to compute $\frac{\partial \Phi_L}{\partial w_{ij}^{\eta}(n)}$ which contributes the most heavy computational load. We only consider the case for three layer and single output network since it is the most common case encountered.

At step n, the outputs of neurons in each layer are given by

$$\begin{aligned}
x_1^0 &= x_{1,n}, \\
&\vdots \\
x_d^0 &= x_{d,n}, \\
x_1^1 &= g_0^1(\sum_{k=1}^{d} w_{1k}^1(n) x_{k,n}), \\
&\vdots \\
x_{n_1}^1 &= g_0^{n_1}(\sum_{k=1}^{d} w_{n_1 k}^1(n) x_{k,n}), \\
x_1^2 &= g_1^1(\sum_{k=1}^{n_1} w_{1k}^2(n) x_k^1), \\
&\vdots \\
x_{n_2}^2 &= g_1^{n_2}(\sum_{k=1}^{n_1} w_{n_2 k}^2(n) x_k^1), \\
x_1^3 &= g_2^1(\sum_{k=1}^{n_2} w_{1k}^3(n) x_k^2).
\end{aligned}$$

For simplicity, we will ignore the dependence of $x_j^i, i = 0, ..., L, j = 1, ..., n_i$, on the integer n. In what follows, we will derive the formulas for computing quantities $\frac{\partial \Phi_L}{\partial w_{ij}^{\eta}(n)}$. Again, for simplicity, we drop the notion n in $w_{ij}^{\eta}(n)$ and write $w_{ij}^{\eta}(n)$ as w_{ij}^{η}.

For η=L, we have $\Phi_L = g_{L-1}(W_L X^{L-1}) = g_{L-1}(\sum_{i=1}^{n_{L-1}} w_{1i}^L x_i^{L-1})$, where g_{L-1} is the transfer function for the output layer, which is usually a sigmoidal function, and $X^{L-1} = (x_1^{L-1}, ..., x_{n_{L-1}}^{L-1})$. Elementary calculation yields

$$\frac{\partial \Phi_L}{\partial w_{1j}^L} = g'_{L-1}(\sum_{k=1}^{n_{L-1}} w_{1k}^L x_k^{L-1}) x_j^{L-1}. \tag{3.83}$$

For η=L-1, we similarly have

$$\begin{aligned} \Phi_L &= g_{L-1}(W_L X^{L-1}) \\ &= g_{L-1}(\sum_{i=1}^{n_{L-1}} w_{1i}^L x_i^{L-1}), \end{aligned} \tag{3.84}$$

and

$$\begin{aligned} X^{L-1} &= g_{L-2}(W_{L-1} X^{L-2}) \\ &= \begin{bmatrix} x_1^{L-1} \\ \vdots \\ x_{n_{L-1}}^{L-1} \end{bmatrix} \\ &= \begin{bmatrix} g_{L-2}^1(\sum_{i=1}^{n_{L-2}} w_{1i}^{L-1} x_i^{L-2}) \\ \vdots \\ g_{L-2}^{n_{L-1}}(\sum_{i=1}^{n_{L-2}} w_{n_{L-1}i}^{L-1} x_i^{L-2}) \end{bmatrix}. \end{aligned} \tag{3.85}$$

Thus,

$$\frac{\partial \Phi_L}{\partial w_{ij}^{L-1}} = g'_{L-1}(\sum_{k=1}^{n_{L-1}} w_{1k}^L x_k^{L-1})(w_{1i}^L \frac{\partial x_i^{L-1}}{\partial w_{ij}^{L-1}}), \tag{3.86}$$

where $\frac{\partial x_i^{L-1}}{\partial w_{ij}^{L-1}}$ can be written as

$$\frac{\partial x_i^{L-1}}{\partial w_{ij}^{L-1}} = g_{L-2}^{i'}(\sum_{k=1}^{n_{L-2}} w_{ik}^{L-1} x_k^{L-2}) x_j^{L-2}. \tag{3.87}$$

For η=L-2, we similarly have the following set of equations

$$\begin{aligned}\Phi_L &= g_{L-1}(\mathbf{W}X^{L-1}) \\ &= g_{L-1}(\sum_{i=1}^{n_{L-1}} w_{1i}^L x_i^{L-1}).\end{aligned} \tag{3.88}$$

$$\begin{aligned}X^{L-1} &= \begin{bmatrix} x_1^{L-1} \\ \vdots \\ x_{n_{L-1}}^{L-1} \end{bmatrix} \\ &= \begin{bmatrix} g_{L-2}^1(\sum_{i=1}^{n_{L-2}} w_{1i}^{L-1} x_i^{L-2}) \\ \vdots \\ g_{L-2}^{n_{L-1}}(\sum_{i=1}^{n_{L-2}} w_{n_{L-1}i}^{L-1} x_i^{L-2}) \end{bmatrix},\end{aligned} \tag{3.89}$$

$$\begin{aligned}X^{L-2} &= \begin{bmatrix} x_1^{L-2} \\ \vdots \\ x_{n_{L-2}}^{L-2} \end{bmatrix} \\ &= \begin{bmatrix} g_{L-3}^1(\sum_{i=1}^{n_{L-3}} w_{1i}^{L-2} x_i^{L-3}) \\ \vdots \\ g_{L-3}^{n_{L-2}}(\sum_{i=1}^{n_{L-3}} w_{n_{L-2}i}^{L-2} x_i^{L-3}) \end{bmatrix},\end{aligned} \tag{3.90}$$

$$\frac{\partial \phi_L}{\partial w_{ij}^{L-2}} = g'_{L-1}(\sum_{i=1}^{n_{L-1}} w_{1i}^L x_i^{L-1})(\sum_{k=1}^{n_{L-1}} w_{1k}^L \frac{\partial x_k^{L-1}}{\partial w_{ij}^{L-2}}), \tag{3.91}$$

$$\frac{\partial x_k^{L-1}}{\partial w_{ij}^{L-2}} = g_{L-2}^{k'}(\sum_{i=1}^{n_{L-2}} w_{ki}^{L-1} x_i^{L-2})(w_{ki}^{L-1} \frac{\partial x_i^{L-2}}{\partial w_{ij}^{L-2}}), \tag{3.92}$$

$$\frac{\partial x_i^{L-2}}{\partial w_{ij}^{L-2}} = g_{L-3}^{i'}(\sum_{p=1}^{n_{L-3}} w_{ip}^{L-2} x_p^{L-3}) x_j^{L-3}. \tag{3.93}$$

Based on the above formulas (3.83) - (3.93), we can compute the derivatives of Φ_L with respect to w_{ij}^{η} for each triple (η, i, j). Below, we consider the following special case.

Special Case:

A commonly found case for two-player differential game problems is that the cost function J is the function of the final state x_N. We have only considered the case of one control variable for both players. Thus, $J = J_1(\mathbf{x_N}, \psi_1, \cdots, \psi_N, \phi_1, \cdots, \phi_N)$ (refer to the pursuit-evasion game problem in Section 3.5). Both players try to maximize/minimize the cost function $J = J_1(\mathbf{x_N}, \psi_1, \cdots, \psi_N, \phi_1, \cdots, \phi_N)$ subject to a set of difference equations:

$$\max_{(\psi_1,\cdots,\psi_N)} \min_{(\phi_1,\cdots,\phi_N)} J = J_1(\mathbf{x_N}, \psi_1, \cdots, \psi_N, \phi_1, \cdots, \phi_N),$$

$$\mathbf{x_n} = f(\mathbf{x_{n-1}}, \phi_{n-1}, \psi_{n-1}), \qquad \text{n=1, 2, ... N,}$$

where $\phi_i \in \Phi$ and $\psi_i \in \Psi$, i = 1, 2, ..., N. In this case, the neural control problem for the case where $\hat{\psi}_i$ is known is formalized as follows:

$$\min_{(\phi_1,\cdots,\phi_N)} J_1(\mathbf{x_N}, \hat{\psi}_1, \cdots, \hat{\psi}_N, \phi_1, \cdots, \phi_N),$$

$$\mathbf{x_n} = f(\mathbf{x_{n-1}}, \phi_{n-1}, \hat{\psi}_{n-1}), \qquad \text{n=1, 2, ... N,}$$

and the control sequence $(\phi_1, \phi_2, \cdots, \phi_N)$ is given by $\phi_i = \Phi_L(\mathbf{W_1}, \cdots, \mathbf{W_L}, \mathbf{x_i}), i = 1, 2, \cdots, N$, and the updating rule takes the form

$$w_{ij}^{\eta}(N) = w_{ij}^{\eta}(N-1) - \alpha_{ij}^{\eta} \frac{\partial J_1}{\partial w_{ij}}(x_N, \hat{\psi}_1, \cdots, \hat{\psi}_N, \mathbf{W}), \qquad (3.94)$$

$$\eta = 1, 2, ..., L,$$

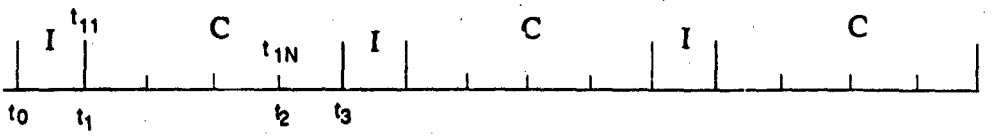
$$i=1, 2, ..., n_\eta,$$

$$j=1, 2, ..., n_{\eta-1},$$

which means that updating the weights happens at the (N-1)th step.

3.6.2. Implementation

In order to use above updating rule in real time, an on-line scheme has to be considered. A consideration is shown in Figure 3.19.



I: Identification Process

C: Control Process

Figure 3.19: Identification-Plus-Control Process

An identification-plus-control period is from t_0 to t_3, which has three subintervals: $[t_0, t_1]$(identification), $[t_1, t_2]$(updating weights), $[t_2, t_3]$(control). The identification part identifies the control strategy of the opponent based on the environmental information (see Section 3.5). This process has been thoroughly discussed in Section 3.5. Evaluating the weights of the network happens in the subinterval $[t_{11}, t_{1N}]$ or $[t_{11}, t_2]$. In each of the small subintervals $[t_{1i}, t_{1(i+1)}]$, i=1, 2, ..., N-1, the weights of each layer are updated. The process is repeated and ends at t_2. At t_2, actual control is applied to the system.

3.7. Applications of the Learning Algorithm To the Problem of Aircraft Control In The Presence of Windshear

The particular problem we are considering here is that of control of an aircraft encountering windshear after take-off. Much effort has gone into modeling and identifying windshear; e.g. [23, 100], but only some of it has been concerned with the design of controllers to enhance the chances for survival. Among these are the studies of Miele[63] and of Leitman[55], et al. A stabilized controller is proposed by Leitman in which no *a priori* bounding information is needed.

In the last section, we already derived the updating rule for the neural controller for the problem of a differential game with neural networks. Since the updating rule can be applied to many general control problems, we shall use the

results derived in the last section for our aircraft control problem. We shall present simulation results also. The simulation results are not only reasonable but also advantageous over the controller proposed by Leitman.

We use the following notations:

Notation

$D \stackrel{\text{def}}{=}$ drag force, *lb*;

$g \stackrel{\text{def}}{=}$ gravitational force per unit mass, $ft\ \sec^{-1}$;

$h \stackrel{\text{def}}{=}$ vertical coordinate of aircraft center of mass (altitude), *ft*;

$L \stackrel{\text{def}}{=}$ lift force, *lb*;

$m \stackrel{\text{def}}{=}$ aircraft mass, $lb\ ft^{-1}\ sec^2$;

$O \stackrel{\text{def}}{=}$ mass center of aircraft;

$S \stackrel{\text{def}}{=}$ reference surface, ft^2;

$t \stackrel{\text{def}}{=}$ time, sec;

$T \stackrel{\text{def}}{=}$ thrust force, *lb*;

$V \stackrel{\text{def}}{=}$ aircraft speed relative to wind based reference frame, $ft\ \sec^{-1}$;

$W_x \stackrel{\text{def}}{=}$ horizontal component of wind velocity, $ft\ \sec^{-1}$;

$W_h \stackrel{\text{def}}{=}$ vertical component of wind velocity, $ft\ \sec^{-1}$;

$x \stackrel{\text{def}}{=}$ horizontal coordinate of aircraft center of mass, *ft*;

$\alpha \stackrel{\text{def}}{=}$ relative angle of attack, *rad*;

$\gamma \stackrel{\text{def}}{=}$ relative path inclination, *rad*;

$\delta \stackrel{\text{def}}{=}$ thrust inclination, *rad*;

$\rho \stackrel{\text{def}}{=}$ air density, $lb\ ft^2\ \sec^2$.

3.7.1. The Problem

Following Miele's lead, we employ equations of motion for the center of mass of the aircraft, in which the kinematic variables are relative to the ground while the dynamic ones are taken relative to a moving but non-rotating frame translating with the wind velocity at the aircraft's center of mass. The kinematic equations are

$$\dot{x} = V\cos(\gamma) + W_x, \tag{3.95}$$

$$\dot{h} = V\sin(\gamma) + W_h. \tag{3.96}$$

The dynamical equations are [55]

$$m\dot{V} = T\cos(\alpha+\delta) - D - mg\sin\gamma - m(\dot{W}_x\cos\gamma + \dot{W}_h\sin\gamma), \tag{3.97}$$

$$mV\dot{\gamma} = T\sin(\alpha+\delta) - L - mg\cos\gamma - m(\dot{W}_x\sin\gamma - \dot{W}_h\cos\gamma), \tag{3.98}$$

where $T = T(V)$ is the thrust force, $D = D(h, V, \alpha)$ is the drag, $L = L(h, V, \alpha)$ is the lift, and $W_x = W_x(x, h)$ and $W_h = W_h(x, h)$ are the horizontal and vertical windshears, respectively. In these equations, $x(t), h(t), V(t), \gamma(t)$ are the state variables and the angle of attack $\alpha(t)$ is the control variable.

A discrete-time version of equations (3.97) and (3.98) is given by

$$\begin{aligned} V_{k+1} &= f_1(V_k, T_k, D_k, \gamma_k, \dot{W}_{xk}, \dot{W}_{hk}, \alpha_k) \\ &= V_k + \frac{T_k\cos(\alpha_k+\delta)\Delta t}{m} - \frac{D_k\Delta t}{m} - g\sin\gamma_k\Delta t - (\dot{W}_{xk}\cos\gamma_k + \dot{W}_{hk}\sin\gamma_k)\Delta t. \end{aligned} \tag{3.99}$$

$$\begin{aligned} \gamma_{k+1} &= f_2(V_k, T_k, L_k, \gamma_k, \dot{W}_{xk}, \dot{W}_{hk}, \alpha_k) \\ &= \gamma_k + \frac{T_k\sin(\alpha_k+\delta)\Delta t}{mV_k} - \frac{L_k\Delta t}{mV_k} - \frac{g\cos\gamma_k\Delta t}{mV_k} - \frac{(\dot{W}_{xk}\sin\gamma_k - \dot{W}_{hk}\cos\gamma_k)\Delta t}{V_k}. \end{aligned} \tag{3.100}$$

where the notations $\dot{W}_{xk}$, $\dot{W}_{hk}$ denote the variables $\dot{W}_x(t)$, $\dot{W}_h(t)$ at the time t = k. Our goal is to design a controller $\alpha = \alpha_k$ such that the quantity $[h_{k+1} - \hat{h}_r]^2$ is minimized, where h_{k+1} is a value calculated from V_{k+1} and α_{k+1} and $\hat{h}_r$ is a given value. The cost function is given by

$$J(k+1) = (V_{k+1} sin\gamma_{k+1} - \hat{h}_r)^2. \tag{3.101}$$

The neural controller is designed so that the weights update at each step to minimize $J(k+1)$.

3.7.2. Assumptions

The following assumptions are the same as Miele's[55]:

1) The rotational inertia of the aircraft and the sensor and actuator dynamics are neglected.

2) The aircraft mass is constant.

3) Air density is constant.

4) Flight is in the vertical plane.

5) Maximum thrust is used.

3.7.3. Bounded Quantities

In order to account for aircraft capabilities, it is assumed that there is a maximum attainable value of the relative angle of attack α; that is, $\alpha \in [0, \alpha_*]$, where $\alpha_* > 0$. The range of practical values of the relative aircraft speed, V, is also limited, that is,

$$\underline{V} \leq V \leq \bar{V}, \tag{3.102}$$

where $\underline{V} > 0$ and $\bar{V} > \underline{V}$ depend on the specific aircraft[55].

3.7.4. Force Terms

The thrust, drag and lift force terms can be approximated [55] by

$$T = A_0 + A_1 V + A_2 V^2, \tag{3.103}$$

$$D = \frac{1}{2} C_D \rho S V^2, \tag{3.104}$$

$$L = \frac{1}{2} C_L \rho S V^2, \tag{3.105}$$

where $C_D = B_0 + B_1 \alpha^2$, $C_L = C_0 + C_1 \alpha$. The coefficients A_0, A_1, A_2 depend on the altitude of the runway, the ambient temperature and the engine power setting. B_0, B_1, C_0, C_1, on the other hand, depend on the flap setting and the undercarriage position.

3.7.5. Windshear Model

In this work, we utilize the windshear model [55] described by the following equations

$$\begin{aligned} W_x &= -W_{x0} sin(2\pi t/T_0), \\ W_h &= -W_{h0}[1 - cos(2\pi t/T_0)]/2, \end{aligned} \tag{3.106}$$

where W_{x0} and W_{h0} are given constants, reflecting the windshear intensity and T_0 is the total flight time through downburst.

3.7.6. Controller Design

We employ a neural controller with one input layer of four neurons. The input variables to the network are $V, \dot{V}, \gamma, \dot{\gamma}$. The control output is given by $\alpha_k =$

$\phi(w_1(V - V(0)) + w_2\dot{V} + w_3(\gamma - \gamma(0)) + w_4\dot{\gamma})$, where the initial values $V(0)$ and $\gamma(0)$ will be given in the next subsection. The threshold function ϕ is the sigmoidal function $T(x) = A\frac{1}{1+e^{-gx}}$, where g is a design gain and A is saturation limit. In our study $A = \alpha_m$. As discussed before, the formula for updating weights is

$$w_i(k+1) = w_i(k) - \alpha_i \frac{\partial J(k+1)}{\partial w_i(k)}, \tag{3.107}$$

where $\frac{\partial J(k+1)}{\partial w_i(k)}$ is given by the following set of equations

$$\frac{\partial J(k+1)}{\partial w_i(k)} = \frac{\partial J(k+1)}{\partial \alpha_k}\frac{\partial \phi_k}{\partial w_i(k)}, \tag{3.108}$$

$$\frac{\partial J(k+1)}{\partial \alpha_k} = \frac{\partial J(k+1)}{\partial \gamma_{k+1}}\frac{df_2(.)}{d\alpha_k} + \frac{\partial J(k+1)}{\partial V_{k+1}}\frac{df_1(.)}{d\alpha_k}, \tag{3.109}$$

$$\frac{\partial J(k+1)}{\partial \gamma_{k+1}} = 2(V_{k+1}sin\gamma_{k+1} - \hat{h}_r)V_{k+1}cos\gamma_{k+1}, \tag{3.110}$$

$$\frac{\partial J(k+1)}{\partial V_{k+1}} = 2(V_{k+1}sin\gamma_{k+1} - \hat{h}_r)sin\gamma_{k+1}, \tag{3.111}$$

$$\frac{df_2(.)}{d\alpha_k} = \frac{\partial f_2(.)}{\partial \alpha_k} + \frac{\partial f_2(.)}{\partial L_k}\frac{\partial L_k}{\partial \alpha_k}, \tag{3.112}$$

$$\frac{\partial f_2(.)}{\partial \alpha_k} = \frac{T_k \Delta t}{mV_k}cos(\alpha_k + \delta), \tag{3.113}$$

$$\frac{\partial f_2(.)}{\partial L_k} = \frac{\Delta t}{mV_k}, \tag{3.114}$$

$$\frac{\partial L_k}{\partial \alpha_k} = \frac{1}{2}C_1\rho S V_k^2, \tag{3.115}$$

$$\frac{df_1(.)}{d\alpha_k} = \frac{\partial f_1(.)}{\partial \alpha_k} + \frac{\partial f_1(.)}{\partial D_k}\frac{\partial D_k}{\partial \alpha_k}, \tag{3.116}$$

$$\frac{\partial f_1(.)}{\partial \alpha_k} = -\frac{T_k \Delta t}{m}sin(\alpha_k + \delta), \tag{3.117}$$

$$\frac{\partial f_1(.)}{\partial D_k} = -\frac{\Delta t}{m}, \tag{3.118}$$

$$\frac{\partial D_k}{\partial \alpha_k} = \frac{1}{2}B_1\rho S V_k^2. \tag{3.119}$$

3.7.7. Numerical Data

As a specific model, we use one model for a Boeing-727 aircraft with JT8D-17 turbofan engines. We assume that the aircraft has become airborne from a runway located at sea-level. The data are identical to those of Miele

$$\begin{aligned}
\alpha_* &= 16^\circ, \\
C &= 3^\circ/sec, \\
A_0 &= 44564.0\ lb, \\
A_1 &= -23.98\ lbft^{-1}sec, \\
A_2 &= 0.01442\ lbft^{-1}sec, \\
\delta &= 2^\circ, \\
\rho &= 0.002203\ lbft^{-4}sec^2, \\
S &= 1560\ ft^2, \\
B_0 &= 0.0218747, \\
B_1 &= 0.6266795, \\
C_0 &= 0.2624993, \\
C_1 &= 5.3714832, \\
mg &= 180000\ lb. \\
\underline{V} &= 184\ ftsec^{-1}, \\
\tilde{V} &= 422\ ftsec^{-1}, \\
\Delta t &= 0.001\ sec, \\
\hat{h}_r &= 33.6807,
\end{aligned}$$

while the initial conditions are x(0)=0 ft, h(0)=50 ft, and V(0)=276.8 ft/sec, $\gamma(0) = 6.989°$.

3.7.8. Simulation Results

Numerical simulations were carried out for the case where the windshear intensity is $W_{x0}/W_{h0} = 50/30$. Simulation results show that the neural controller performs well in the presence of wind. From the windshear model, we know that in the first 30 seconds the horizontal wind blows against the aircraft. In the frist 20 seconds, the aircraft gains the altitude. As the wind becomes less strong, the lift of the aircraft decreases and the aircraft gradually slows its climbing rate and begins losing it altitude at the 20th second. To compensate this, the angle of attack increases correspondingly. This is shown in Figure 3.20. After the 30rd second, the horizontal wind blows in the direction of the aircraft which continues loosing its altitude even through the angle of attack increases to try to compensate the loss. As the wind becomes less stronger from the 45th second to the 60th second, the aircraft increases its altitude.

Under the same conditions, the neural controller works better than those of Leitemann's and Miele's in the sense that the control value reached the satured limit (in this case 16 degrees for the angle of attack) for only short period of time. The performance of the aircraft is almost the same as that of Leitman. That is because the angle of attack did not reach high enough from the 45th second to the 60th second to compensate the loss of the altitude of the aircraft. The reason is that the greatest descent algorithm has low convergence rate. To improve the performance, one should consider using an optimization method with

a higher convergence rate. In fact, this is one of our further research directions. For a windshear with stronger wind intensity, e.g. $W_{x0}/V_{h0} = 80/48$, we need to adjust the four weights accordingly. With this type of neural controllers, the performance depends on the sensitivity of the gradient of the cost function to the change of the measure error. With a suitable choice of learning rates $\alpha_1, \alpha_2, \alpha_3, \alpha_4$, the performance should be improved.

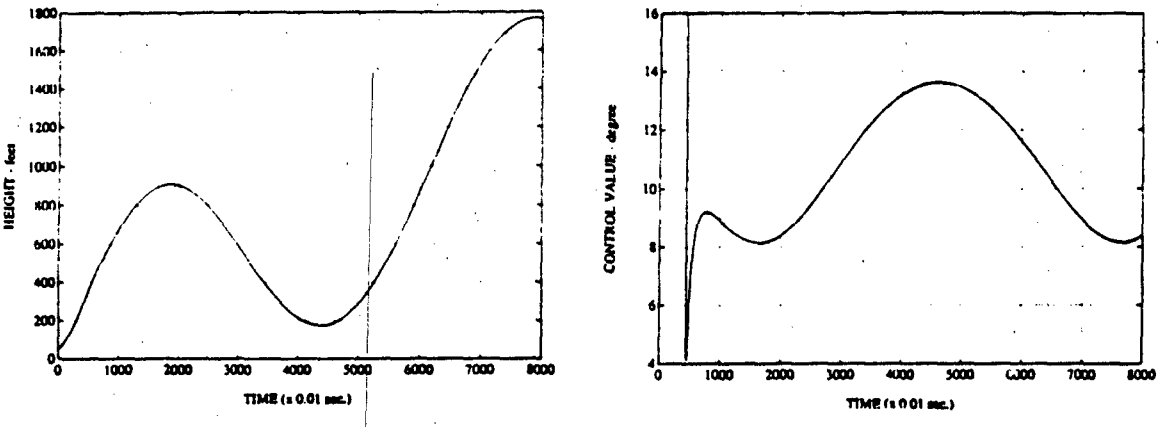


Figure 3.20: Simulation Results For the Aircraft Control Problem

4.
Optimal Control Problem in the Layered Defense Project

4.1. Introduction

In this chapter, an application of artificial intelligence methodology, more exactly, a rule-based expert system, in a class of pursuit-evasion game problems based on the Semantic Control Theory will be discussed. The object of this effort is to apply the principles of the semantic control theory to situation assessment in a layered defense system. A tactical decision aid has been developed to assist its user in the selection of heading, speed, and countermanuevers in the presence of a real or potential threat. The project, started in June, 1991, is a cooperative effort of the Center for Semantic Control, Washington University, and the Electronics and Space Corporation and has been proven to be successful.

A game problem of multi-pursuer, single-evader is the main one for this project. The following players exist: one evader, also termed Ownship, several pursuers, and a limited number of shadowing players engaged in the game situation. There are two types of pursuers: [i] primary pursuers which usually represent aggressive aircraft, equipped with air-to-air and all-aspect missiles; [ii] secondary pursuers which typically represent the offensive missiles launched by primary pursuers. Both primary pursuers and the evader have the capability of spawning shadowing players which represent passive objects to blind the opponents, such as flairs. Dynamics can/cannot be incorporated into the shadowing players, depending on the types of shadowing players used. The initial stage of the project will discuss the

situation with one evader, one/several secondary pursuers and a limited number of shadowing players for both the primary pursuer and evader.

The TDA has no control over the strategies of either primary pursuers or secondary pursuers, yet it has the capability of detecting and identifying their strategies and maneuvers, mainly by means of Contact Reports. The contact reports contain enough information about the pursuers and the evader, such as location(coordinates), speed, heading, bearing, etc., all of which can be processed to assess the situation. The TDA receives the Contact Reports every fixed period of time so that new information can be updated periodically. While the human will assume the role of the controller, the functions that the TDA will have to perform are [25] obtaining data from Contact Reports and updating the player information, using new information to assess the current game situation, etc. Processing new information from the contact reports includes updating the information about each of the players and storing the old information on each player to an instance of the class OLD. Once the process is finished, a role or the optimal control strategy will be used to govern the next movement of the evader. A detailed discussion is given below.

As shown in Figure 4.1, the structure of the TDA is organized hierarchically. Each player in the TDA is called an object. Typical examples of objects are the primary pursuers, the secondary pursuers, and the evader which are also instances of some appropriate class. Each object belongs to one class which is organized hierarchically. Each class is a subclass of its parent class, and all classes are the subclass of the root class called "ROOT". Each object has its attributes, stored in memory called "Slots", which are either inherited from its parents or locally resident. Associated with each object are the rules, methods, and functions which

can be used to interact with objects. The information about each player of the game, capabilities for each player, and game situation are all stored in the slots of each player. Table 4.1 summarizes the slots for the primary pursuers and the evader. An object-oriented formulation is selected because it allows incremental refinement of its situation assessment, knowledge of pursuer's capability, and evasion strategies.

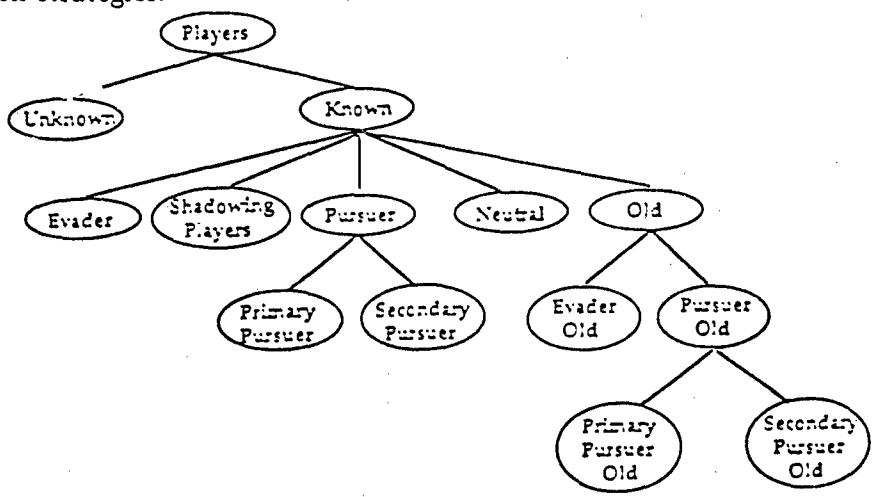


Figure 4.1: Object Hierarchy

Table 4.1: Table of Slots

Player	Known	Evader	Pursuers	Shadowing Players
Position Speed Range Bearing Heading Contact ID Classification Event Confidence	Lifetime MaxSpeed Perception	Home Destination Risk Level Max Safe Distance Panic Distance Number of Shadowing Players	Effectiveness	Effectiveness Delay Duration Spatial Effectiveness

The mission of the evader is to depart from Home, while evading the pursuit of its offensive opponent, and to head to a fixed location called "Destination".

The role of the evader is to assess the game situation, detect the pursuer strategy if necessary and make a decision based on the updating information. It does not assume any offensive capabilities. With this assumption, the TDA presents suggestions of movement for the user's approval. The movement of the evader is governed by the rule fired from the rule-based system or the optimal control strategy. Whether a role or the optimal control strategy will be used depends on the range between two players. Assuming a fixed distance, we can have more than one rule to select, in which case a quantity called "utility" plays a key role. A utility is a positive real number scaling from zero to one, associated with each pursuer. The utility can be thought of as a measurement of threat to the evader. The pursuer with the highest utility value is said to have greatest threat to the evader and hence should be paid the greatest attention. The utility of the pursuer depends on five components - range, heading, elevation, speed and bearing - all of which has been obtained through processing the information from the contact reports. A detailed discussion of the utility calculation can be found in [25].

Recent work in the area of Semantic Control Theory [76] provides the means for our project. In this paradigm, a control problem is broken into three blocks, namely, Identifier, Goal Selector, and Adaptor [76]. In this project, the Goal Selector is designed for the following [i] To select the rule from the rule-based system based on the information provided by the System Identifier, which has preprocessed data from the Contact Reports, [ii] To activate the optimal control law based on the information of range, and [iii] To compute the heading for the next movement of the evader. The decision of the Goal Selector depends on several factors: range, heading, bearing, speed and elevation. When the pursuer is within a certain distance from the evader, the optimal control law is activated to achieve a fast response to the situation. In other cases, the Rule-Based system

plays a key role. There are four sets of rules [25]. They are: [i] the set of rules for the case in which no pursuers are within a specified range, [ii] the set of rules for the case in which one primary pursuer is within a specified range, [iii] the set of rules for the case in which one secondary pursuer is within a specified range, and [iv] the set of rules for the case in which both a primary pursuer and secondary pursuer are within a specified range. Each of these four rule sets can have several rules. The rule set appropriate to the situation is made active and used to make recommendations for safely moving the evader. Which rule will be fired depends on the value of utility associated with each pursuer which represents the measure of the safety for each player.

The function of TDA is to provide a movement set-point recommendation for the user. Therefore, a man-machine graphic interface is an essential part of the project. A human assumes control of the final decision. The Adaptor consists of four different graphic displays: Action Panel, Control Panel, Local View and Global View (see Figure 4.2). The Action Panel shown in Figure 4.2(a) displays the setpoint recommendation for the user, heading, speed and location of each player and the current rule being used. There are several buttons available for the user to decide either to use the recommendation of the TDA or to enter his own command. The Action Panel is updated in real-time so that the user can have a view of the on-going game situation. The Control Panel (see Figure 4.2(b)), on the other hand, is in control of the behavior of each player, monitors the game situation, and has the authority to change it dramatically. The Control Panel initializes the game, modifies the attributes of each player in the game, and runs/stops the game. The Control Panel is actually the first display shown to the user when the simulation begins. Two coordinate frames are used in the project. They are the inertial frame, which provides a global view for the game, and the

frame centered at the speed direction of the evader, who provides a view locally. Local and Global views (see Figure 4.2) provide the basic displays in the initial stage of the project.

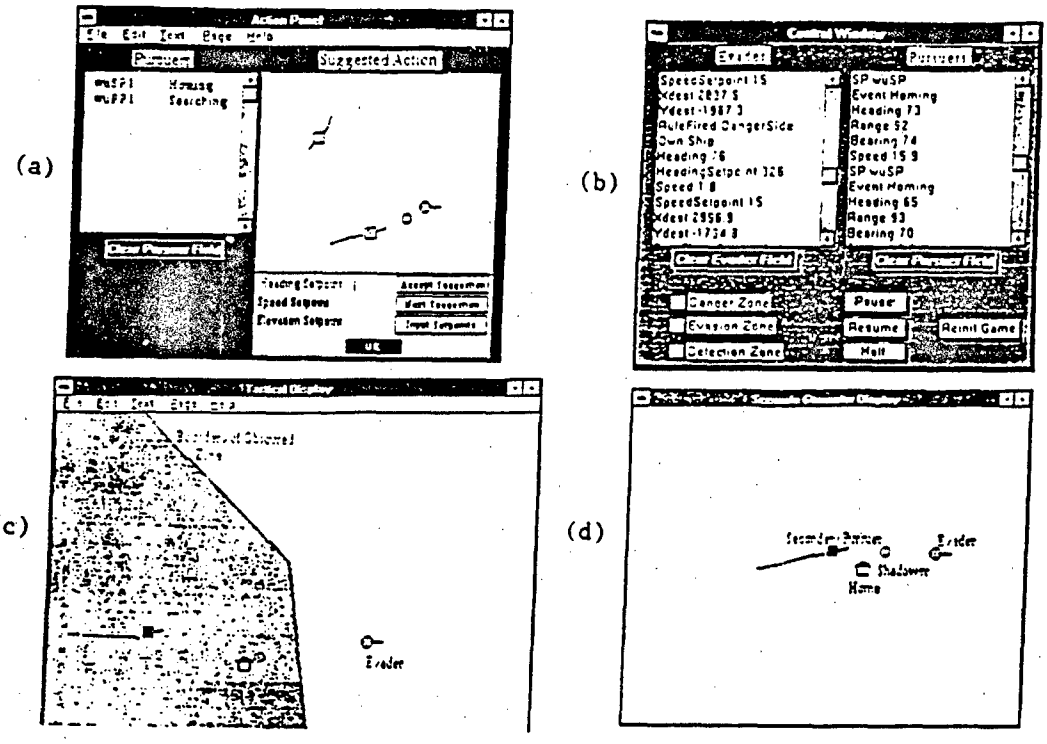


Figure 4.2: Display Panels

The TDA is currently implemented on a DELL SYSTEM 333 D^{TM} Personal Computer, using KAPPA-PC^{TM} and Toolbook^{TM} under the Windows envirnment. For a complete decription of the project and its operation, please refer to [25].

In sections that follow, we shall discuss the optimal control problem that arises from the project. The background for this problem is as follows. The Layered Defense Project's aim is to study, analyze and solve a class of pursuit-evasion problems and to develop a tactical decision aid, which, in the presence of a real-time or potential threat, aids its user in the selection of heading, speed,

and countermaneuvers. An assumption at initial stage of this project is that there are two players in the game situation: one pursuer and one evader. The control strategy for the pursuer is generated by the so-called Scenario Generator, which also provided the Contact Report to TDA. The evader has the capability of identifying the control strategy of the pursuer but has no control over it. The role of the evader is [i] to access the game situation, to identify the strategy of pursuer, and [ii] to take corresponding actions governed by the rule fired from the rule-based system or by the control value of the optimal control law, depending on the range R between the pursuer and him. If R is greater than some given value R_0, the evader may want to continue doing what he has been doing. While R decreases to some given value R_1 with $R_1 < R_0$, which usually represents the situation that the pursuer is getting closer to the evader, the value of utility for the evader is high enough to make a rule fired from the rule-based systems. Which rule will be fired depends on the utility associated with the rule [25]. However, if the rule fired for governing the next movement cannot improve the situation and R further decreases to some given value R_2, where $R_2 < R_1 < R_0$, an optimal control strategy is employed to yield an accurate, fast response to the situation. The assumption that two players are engaged in the game is reasonable since only one aggressive player shows the highest potential threat to the ownship and hence will be paid much more attention than others, if they exit, when the optimal control law is used.

With the line-of-sight model, our problem can be described as follows: given σ_2, a control strategy for the pursuer, find an optimal control for the evader, i.e., $\sigma_{1\text{optimal}}$, where $|\sigma_1| \leq 1, |\sigma_{1\text{optimal}}| \leq 1$, such that the range between two players reaches some prespecified value in minimum time.

4.2. Optimal Control Problem

4.2.1. Line of Sight Coordinates

A line-of-sight coordinate model has been used to study the pursuit-evasion game with fixed role determination [17, 18, 19, 41, 83, 92]. Using the line of sight as common reference, we have three state variables. These state variables are the range $0 \leq R \leq \infty$ and the two off-boresight angles $-\pi \leq \phi_1 < \pi, -\pi \leq \phi_2 < \pi$ [18, 19].

The equations of motion in the general line of sight coordinates are

$$\dot{R} = -(cos\phi_1 + cos\phi_2), \tag{4.1}$$

$$\dot{\phi}_1 = (sin\phi_1 + sin\phi_2)/R + \sigma_1, \tag{4.2}$$

$$\dot{\phi}_2 = (sin\phi_1 + sin\phi_2)/R + \sigma_2, \tag{4.3}$$

where σ_1 and σ_2 are the respective control variables (turning rates) constrained by

$$|\sigma_i| \leq 1, \quad i = 1, 2. \tag{4.4}$$

The geometry of the engagement is depicted in Figure 4.3.

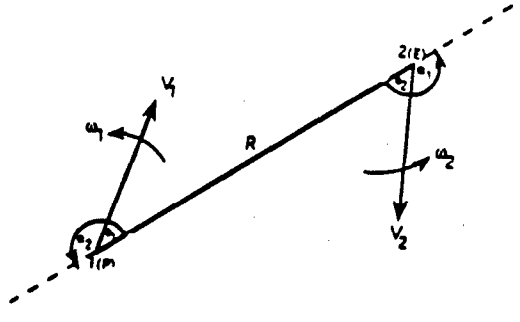


Figure 4.3: Geometry of the Engagement

A major element in the differential game formulation is the definition of the terminal surface (target set). Such a terminal surface represents the firing envelope of the pursuer aircraft weapon system. The firing envelope of an air-to-air missile for each player i (i = 1, 2) is a subspace defined by an implicit equation of 3 variables

$$F(R, \phi_i, \alpha_i) = 0, \tag{4.5}$$

where R is the range (the magnitude of the line of sight vector), and ϕ_i and α_i have the relation

$$\phi_i + \alpha_i = \pi. \tag{4.6}$$

In [18], Davidovitz *et al* specified the terminal surface of the game, a special form of equation (4.5), as

$$\begin{aligned} R(t_f) &\triangleq R_f \\ &= \beta, \end{aligned} \tag{4.7}$$

where t_f is the final time of the game and β is some prespecified positive value. With this terminal surface, one can evaluate the necessary conditions, which are used to obtain the optimal control strategies. Such control strategies can be evaluated, using retrointegration, by calculating the so-called optimal terminal strategy at the terminal surface. The target set eq. (4.7) has been successfully used for the pursuit-evasion game analysis.

On the other hand, for an air combat duel between similar aggressive fighter aircrafts, both equipped with the same type of guided missiles, different target sets are used to represent the effective firing envelope of an all-aspect fire-and-forget

air-to-air missile

$$-\beta \leq \phi_i \leq \beta, \qquad i = 1,2, \tag{4.8}$$

$$\underline{R}_i(\phi_j) \leq R < \bar{R}_i(\phi_j), \qquad j = 1,2,\ j \neq i, \tag{4.9}$$

where β is the off-boresight limit for missile firing, while $\underline{R}$ and $\bar{R}$ are the minimum and maximum normalized firing ranges, respectively, given by

$$\underline{R}_i(\phi_j) = a_i + b_i cos\phi_j, \tag{4.10}$$

$$\bar{R}_i(\phi_j) = (\bar{R}_0)_i - |\phi_j + sin\phi_j|, \tag{4.11}$$

where a_i, b_i and $(\bar{R}_0)_i$ are the normalised parameters (see [19]).

For our problem, a terminal surface similar to that in [18] is used, i. e. $R(t_f) = \beta$, where β is a prespecified positive value. In [18], β is specified to be less than the initial distance $R(t_0)$ since β represents the distance for capture. In our case, however, β is a positive value greater than $R(t_0)$ since our problem is that representing escape.

In general, the solution consists of the decomposition of the game space into four regions: the respective winning zone of the two opponents, the draw zone, and the region where the game terminates by a mutual kill. Davidovitz *et al* [19] presented a qualitative study, the first of its kind, of an air combat between two similar aircrafts equipped with modern air-to-air missiles, which is modeled as a two-target differential game. Their results of the study also reveal several yet unknown elements to be expected in later air combat.

For our problem, since the initial stage of the project only discusses two players (pursuer and evader or termed as "Ownship"), the line-of-sight coordinate model is ideal for our quantitative analysis. However, unlike [19], where two aggressive

aircrafts with all-aspect fire-and-forget air-to-air missiles are considered, we have a limited number of players engaged in the game: pursuer and evader. To better model the real-time pursuer-evader situation, we modify the classical line-of-sight model in [17] to incorporate the speeds of two players

$$\dot{R} = -(v_p cos\phi_1 + v_e cos\phi_2), \tag{4.12}$$

$$\dot{\phi_1} = (v_p sin\phi_1 + v_e sin\phi_2)/R + \sigma_1, \tag{4.13}$$

$$\dot{\phi_2} = (v_p sin\phi_1 + v_e sin\phi_2)/R + \sigma_2, \tag{4.14}$$

where the state variables R, ϕ_1, ϕ_2 have the same definitions as before, v_p represents the speed of the pursuer and v_e for that of evader. Again, the respective controls (turning rates) are constrained by

$$|\sigma_i| \leq 1.$$

4.2.2. Optimal Control Law

The problem can be stated as

$$\min_{\sigma_1} J = \min_{\sigma_1} \int_{t_0}^{t_f} dt, \tag{4.15}$$

subject to

$$\dot{R} = -(v_p cos\phi_1 + v_e cos\phi_2), \tag{4.16}$$

$$\dot{\phi_1} = (v_p sin\phi_1 + v_e sin\phi_2)/R + \sigma_1, \tag{4.17}$$

$$\dot{\phi_2} = (v_p sin\phi_1 + v_e sin\phi_2)/R + \hat{\sigma}_2, \tag{4.18}$$

where $(R(t_0), \phi_1(t_0), \phi_2(t_0)) = (R_0, \phi_{10}, \phi_{20})$ and $(R_0, \phi_{10}, \phi_{20})$ are given and the control for the pursuer has been replaced by its estimate value $\hat{\sigma}_2$.

Consider the Hamiltonian given by

$$\begin{aligned} H &= \lambda_0 + \lambda_R(-(v_p cos\phi_1 + v_e cos\phi_2)) \\ &\quad + \frac{\lambda_1 + \lambda_2}{R}(v_p sin\phi_1 + v_e sin\phi_2) \\ &\quad + (\lambda_1\sigma_1 + \lambda_2\hat{\sigma}_2), \end{aligned} \tag{4.19}$$

where $\lambda_0, \lambda_R, \lambda_1$ and λ_2 are the respective components of the gradient vector satisfying the adjoint equations

$$\dot{\lambda}_R = (v_p sin\phi_1 + v_e sin\phi_2)(\frac{\lambda_1 + \lambda_2}{R^2}), \tag{4.20}$$

$$\dot{\lambda}_1 = (-\lambda_R sin\phi_1 - \frac{\lambda_1 + \lambda_2}{R} cos\phi_1)v_p, \tag{4.21}$$

$$\dot{\lambda}_2 = (-\lambda_R sin\phi_2 - \frac{\lambda_1 + \lambda_2}{R} cos\phi_2)v_e, \tag{4.22}$$

$$\dot{\lambda}_0 = 0. \tag{4.23}$$

The optimal control is obtained by

$$\min_{\sigma_1} H, \tag{4.24}$$

which yields

$$\sigma_1^*(t) = -\text{sgn}\lambda_1(t). \tag{4.25}$$

Next, we shall state a lemma which will be used later.

Lemma 4.1 *The following equality holds*

$$\lambda_R^2 + (\frac{\lambda_1 + \lambda_2}{R})^2 = C, \tag{4.26}$$

where C is a constant.

Proof: Differentiating the right hand side of eq. (4.26), we have

$$
\begin{aligned}
&2\lambda_R\dot{\lambda}_R + 2(\frac{\lambda_1+\lambda_2}{R})\frac{(\dot{\lambda}_1+\dot{\lambda}_2)R-(\lambda_1+\lambda_2)\dot{R}}{R^2} \\
&\quad = 2[\lambda_R\frac{\lambda_1+\lambda_2}{R^2}(v_p sin\phi_1 + v_e sin\phi_2) + \frac{\lambda_1+\lambda_2}{R^2} \\
&\quad\quad \times((-\lambda_R sin\phi_1 v_p - \frac{\lambda_1+\lambda_2}{R}cos\phi_1 v_p)R \\
&\quad\quad +(-\lambda_R sin\phi_2 v_e - \frac{\lambda_1+\lambda_2}{R}cos\phi_2 v_e)R \\
&\quad\quad +(\lambda_1+\lambda_2)(v_p cos\phi_1 + v_e cos\phi_2))/R] \\
&\quad = 2\frac{\lambda_1+\lambda_2}{R^2}[\lambda_R v_p sin\phi_1 + \lambda_R v_e sin\phi_2 - \lambda_R v_p sin\phi_1 \\
&\quad\quad -\frac{\lambda_1+\lambda_2}{R}v_p cos\phi_1 - \lambda_R v_e sin\phi_2 - \frac{\lambda_1+\lambda_2}{R}v_e cos\phi_2 \\
&\quad\quad +\frac{\lambda_1+\lambda_2}{R}(v_p cos\phi_1 + v_e cos\phi_2)] \\
&\quad = 0. \qquad (4.27)
\end{aligned}
$$

Thus, eq. (4.26) follows.

Q.E.D.

REMARKS:

The constant C could be zero or nonzero. If C is zero, $\lambda_R(t) = 0$ for all $t \geq 0$. In particular, $\lambda_R(0) = 0$. Thus, we may choose $\lambda_R(0) \neq 0$ such that $C \neq 0$. For $C \neq 0$, we can normalize the formula (4.26) such that

$$\lambda_R^2 + (\frac{\lambda_1+\lambda_2}{R})^2 = 1. \qquad (4.28)$$

Using eq. (4.17), eq. (4.20) can be written as

$$
\begin{aligned}
\dot{\lambda}_R &= (v_p sin\phi_1 + v_e sin\phi_2)(\frac{\lambda_1+\lambda_2}{R^2}) \\
&= (\dot{\phi}_1 - \sigma_1)R(\frac{\lambda_1+\lambda_2}{R^2}) \\
&= (\dot{\phi}_1 - \sigma_1)(\frac{\lambda_1+\lambda_2}{R}) \\
&= (\dot{\phi}_1 - \sigma_1)(1-\lambda_R^2)^{\frac{1}{2}}. \qquad (4.29)
\end{aligned}
$$

Likewise,

$$\dot{\lambda}_R = (1-\lambda_R^2)^{\frac{1}{2}}(\dot{\phi}_2 - \sigma_2). \tag{4.30}$$

Define $\theta_1(t)$ for $0 \leq t \leq t_1$ as

$$\theta_1(t) = -\phi_{10} - \int_0^t \sigma_1(\tau)d\tau, \tag{4.31}$$

where $0 \leq t \leq t_1$ and $\phi_{10} = \phi_1(0)$, or

$$\dot{\theta}_1 = -\sigma_1(t). \tag{4.32}$$

Therefore,

$$\frac{\dot{\lambda}_R}{\sqrt{1-\lambda_R^2}} = \dot{\phi}_1 + \dot{\theta}_1, \tag{4.33}$$

and thus,

$$arcsin\lambda_R = \phi_1 + \theta_1 + C_0, \tag{4.34}$$

where C_0 is an arbitrary constant. Eq. (4.34) becomes

$$\lambda_R = sin(\phi_1 + \theta_1 + C_0). \tag{4.35}$$

Now for t=0, we have

$$\begin{aligned} \lambda_R(0) &= sin(\phi_1(0) + \theta_1(0) + C_0) \\ &= sin(\phi_{10} - \phi_{10} + C_0) \\ &= sinC_0. \end{aligned} \tag{4.36}$$

Thus,

$$\lambda_{R0} = sinC_0. \tag{4.37}$$

Now,

$$\sqrt{1-\lambda_R^2} = cos(\phi_1+\theta_1+C_0). \tag{4.38}$$

Therefore,

$$\begin{aligned}
\dot{\lambda}_1 &= -v_p(\lambda_R sin\phi_1 + \frac{\lambda_1+\lambda_2}{R}cos\phi_1) \\
&= -v_p(sin(\phi_1+\theta_1+C_0)sin\phi_1 + cos(\phi_1+\theta_1+C_0)cos\phi_1) \\
&= -v_p cos(\phi_1+\theta_1+C_0-\phi_1) \\
&= -v_p cos(\theta_1+C_0) \\
&= -v_p cos(C_0-\phi_{10}-\int_0^t \sigma_1(\tau)d\tau).
\end{aligned} \tag{4.39}$$

Without loss of generality, we may assume that $\sigma_1(\tau) = \sigma_{10}^* =$ constant for $0 \leq \tau < t_1$, where t_1 is the first switching time greater than zero.

Then,

$$\lambda_1(t) = \lambda_1(0) + \frac{v_p}{\sigma_{10}^*}sin(-\phi_{10}-\sigma_{10}^* t + C_0), \tag{4.40}$$

where

$$\sigma_{10}^* = -\text{sgn}[\lambda_{10}]. \tag{4.41}$$

The value of t_1 can be obtained by letting $\lambda_1(t_1) = 0$. In our case,

$$t_{1k} = \frac{(C_0-\phi_{10}) + sin^{-1}(\frac{\lambda_1(0)\sigma_{10}^*}{v_p})}{\sigma_{10}^*}, \qquad k = 0, \pm 1, \cdots,$$

and

$$t_1 = \min_{t_{1k}>0} t_{1k}. \tag{4.42}$$

The subsequent switching time can be calculated analogeously.

In general, suppose that $\{t_k; k = 1, 2, \cdots\}$ is a sequence of switching times and that

$$\phi_{1k} = \phi_1(t_k), \tag{4.43}$$

$$\lambda_{Rk} = \lambda_R(t_k). \tag{4.44}$$

$$\lambda_{1k} = \lambda_1(t_k), \tag{4.45}$$

$$\sigma_{1k}^* = \text{sgn}[\lambda_{1k}]. \tag{4.46}$$

Then, on $(t_k, t_{k+1}]$, we have

$$\dot{\lambda}_1(t) = -v_p cos(\theta_{k+1}(t) + C_k), \tag{4.47}$$

$$\lambda_R(t) = sin(\phi_1(t) + \theta_{k+1}(t) + C_k), \tag{4.48}$$

$$\theta_{k+1}(t) = -\phi_1(t_k) - \int_{t_k}^{t} \sigma_1^*(\tau) d\tau, \tag{4.49}$$

$$\sigma_1^*(t) = -\text{sgn}[\lambda_1(t)]. \tag{4.50}$$

Notice that on $(t_{k-1}, t_k]$, we have

$$\lambda_R(t) = sin(\phi_1(t) + \theta_k(t) + C_{k-1}), \tag{4.51}$$

and

$$\theta_k(t) = -\phi_1(t_{k-1}) - \int_{t_{k-1}}^{t} \sigma_1^*(\tau) d\tau. \tag{4.52}$$

Again, on $(t_{k-1}, t_k]$, $\sigma_1^*(\tau) = \sigma_{1(k-1)}^* = \text{constant}$, thus we have

$$\theta_k(t) = -\phi_1(t_{k-1}) - \sigma_{1(k-1)}^*(t - t_{k-1}). \tag{4.53}$$

Thus,

$$\begin{aligned} \lambda_{Rk} &= \lambda_R(t_k) \\ &= sin(\phi_1(t_k) + \theta_k(t_k) + C_{k-1}) \end{aligned} \tag{4.54}$$

But

$$\begin{aligned}\bar{\lambda}_{Rk} &= \lim_{t \to t_k^+} \lambda_R(t) \\ &= \lim_{t \to t_k^+} sin(\phi_1(t) + \theta_{k+1}(t) + C_k) \\ &= sin(\phi_1(t_k) - \phi_1(t_k) + C_k) \\ &= sinC_k \end{aligned} \tag{4.55}$$

Letting $\lambda_{Rk} = \bar{\lambda}_{Rk}$ yields

$$sin(C_k) = sin(\phi_1(t_k) + \theta_k(t_k) + C_{k-1}), \tag{4.56}$$

which is used to determine C_k.

Eqs. (4.43) - (4.50) together give an evaluation of an extremal bang-bang trajectory based on the choice of $[\lambda_{R0}, \lambda_{10}]$.

REMARKS:

Dependence of the control σ_1 on $\hat{\sigma_2}$ is implicit. The corresponding control law can be evaluated without retrograde integration. Thus, the computation time-savings is significant. Since the control law is a function of initial values for costates λ_R and λ_1, the final time is hence a function of the initial values of λ_R and λ_1. Various non-derivative optimization techniques can be employed to find the optimal values for λ_1 and λ_R. The so-called Box's algorithm will be used for our problem because of its well-known inequality constraints, and nonlinear objective function.

4.3. Optimization Technique

4.3.1. Box's Complex Algorithm

The Problem

This algorithm is to find the maximum of a multivariable, nonlinear function subject to nonlinear inequality constraints

$$\max F(x_1, x_2, \cdots, x_N) \tag{4.57}$$

$$\text{subject to} \qquad G_k \le x_k \le H_k, \qquad k = 1, 2, \cdots, N,$$

$$f_i(x_1, x_2, \cdots, x_N) \le x_i \le F_i(x_1, x_2, \cdots, x_N), \qquad i = N{+}1, \cdots, M,$$

where the functions $f_i(x_1, x_2, \cdots, x_N), F_i(x_1, x_2, \cdots, x_N)$ are dependent functions of the explicit independent variables $x_1, x_2, \cdots, x_N$. The upper and lower constraints H_k and G_k are either constants or functions of the independent variables.

Method

The procedure is based on the "complex" method of M. J. Box. This method is a sequential search technique which has proven effective in solving problems with nonlinear objective functions subject to nonlinear inequality constraints. No derivatives are required. The procedure should tend to find the global maximum due to the fact that the initial set of points are randomly scattered throughout the feasible region. If linear constaints are present or equality constraints are involved, other methods should prove to be more efficient. The algorithm proceeds as follows:

1. An original "complex" of $K \ge N + 1$ points is generated consisting of a feasible starting point and K-1 additional points generated from random

numbers and constraints for each of the independent variables

$$\begin{aligned} x_{i,j} &= G_i + r_{i,j}(H_i - G_i), \qquad (4.58) \\ i &= 1, 2, \cdots, N, \\ j &= 1, 2, \cdots, K-1, \end{aligned}$$

where $r_{i,j}$ are random numbers between 0 and 1.

2. The selected points must satisfy both the explicit and implicit constraints. If at any time the explicit constraints are violated, the point is moved a small distance δ inside the violated limit. If an implicit constriant is violated, the point is moved one half of the distance to the centroid of the remaining points

$$x_{i,j}(new) = (x_{i,j}(old) + \bar{x}_{i,c})/2, \qquad i = 1, 2, \cdots, N, \qquad (4.59)$$

where the coordinates of the centroid of the remaining points, $x_{i,c}$ are defined by

$$\bar{x}_{i,c} = \frac{1}{K-1}[\sum_{k=1}^{K} x_{i,k} - x_{i,j}(old)], \qquad i = 1, 2, \cdots, N. \qquad (4.60)$$

This process is repeated as necessary until all the implicit constraints are satisfied.

3. The objective function is evaluated at each point. The new point is located at a distance α times as far from the centroid of the remaining points as the distance of the rejected point on the line joining the rejected point and the centroid

$$x_{i,j}(new) = \alpha(\bar{x}_{i,c} - x_{i,j}(old)) + \bar{x}_{i,c}, \qquad i = 1, 2, \cdots, N. \qquad (4.61)$$

4. If the new point repeats in giving the lowest function values on consecutive trials, it is moved one half the distance to the centroid of the remaining points.

5. The new point is checked against the constraints and is adjusted as before if the constraints are violated.

6. Convergence is assumed when the objective function values at each point are within β units for γ consecutive iterations. An iteration is defined as the calculations required to select a new point which satisfies the constraints and does not repeat in yielding the lowest function value.

4.3.2. Implementation

For our problem, Box's complex algorithm is used to find the minimum of a multivariable nonlinear function subject to a set of nonlinear equality constraints

$$\min J(\lambda_{10}, \lambda_{R0}), \tag{4.62}$$

subject to

$$\begin{aligned}
-9.0 \leq \quad & \lambda_{10} \quad \leq 9.0. \\
-1.0 \leq \quad & \lambda_{R0} \quad \leq 1.0. \\
\dot{R} \quad &= \quad -(v_p cos\phi_1 + v_e cos\phi_2), \\
\dot{\phi_1} \quad &= \quad (v_p sin\phi_1 + v_e sin\phi_2)/R + \sigma_1, \\
\dot{\phi_2} \quad &= \quad (v_p sin\phi_1 + v_e sin\phi_2)/R + \sigma_2, \\
\sigma_1^*(t) \quad &= \quad -\mathrm{sgn}[\lambda_1(t)], \qquad t \in (t_k, t_{k+1}] \\
\lambda_1(t) \quad &= \quad \lambda_1(t_k) + \frac{v}{\sigma_1^*} sin(-\phi_{1k} - \sigma_1^*(t - t_k) + C_k), \qquad t \in (t_k, t_{k+1}]
\end{aligned}$$

$$
\begin{aligned}
\phi_{1k} &= \phi_1(t_k), \qquad t \in (t_k, t_{k+1}] \\
\sigma^*_{1k} &= -\mathrm{sgn}[\lambda_1(t_k)], \qquad t \in (t_k, t_{k+1}] \\
sinC_k &= sin(\phi_1(t_k) + \theta_k(t_k) + C_{k-1}), \\
C_0 &= sin^{-1}(\lambda_{R0}),
\end{aligned}
$$

where $J(\lambda_{10}, \lambda_{R0}) \triangleq t_f - t_0$, t_0 is the initial time value and t_f is the terminal time at which the distance between two players reaches a given value. Therefore, in our case, we have N=2, M=N=2.

For our problem, the following parameters have been used

$$
\begin{aligned}
G1 &= -9.0, \\
H1 &= 9.0, \\
G2 &= -1.0, \\
H2 &= 1.0, \\
K &= 4, \\
\beta &= 0.001, \\
\alpha &= 1.3, \\
\gamma &= 3, \\
\delta &= 0.01.
\end{aligned}
$$

We have two independent variables: $\lambda_{10}, \lambda_{R0}$ and we do not have constraint functions of explicit independent variables. In our implementation, after we randomly generate a complex of K points, we compare the values at each point of the complex with that at the centroid of all points. If the value at the centroid is the highest, we reselect the complex of starting points until the highest value does not occur at the centroid of all points. Figures 4.6 and 4.7 show the simulation results of our problem.

In Figures 4.6 and 4.7, lmbdaR and lmbda1 represent the values of λ_R and λ_{10}. The solid line represnts the trajectory of the evader and the dashed line represents the trajectory of the pursuer. From these plots, we can see clearly how the values of λ_R and λ_{10} affect the trajectories of both players.

The parameters used in our simulation are:

$$\begin{aligned}
\text{The speed of evader: } v_e &= 20.0 \text{ (units)}, \\
\text{The speed of pursuer: } v_p &= 25.0 \text{ (units)}, \\
\text{Initial distance: } R_0 &= 100.0, \\
\text{Final distance: } R_f &= 200.0, \\
\text{Initial off-boresight angle: } \phi_1 &= 0.52, \\
\text{Initial off-boresight angle: } \phi_2 &= 57.33, \\
\text{Fixed control for the pursuer: } \sigma_2 &= 0.55.
\end{aligned}$$

The goal of the optimization technique is to minimize the time such that R increases to a given value R_f.

From above, we know that although the optimal control law is obtained assuming that the speeds for both players are the same, the control law also works fine for the case with different speeds.

In actual implementation, the process of control evaluation is described as follows. As shown in Figure 4.4, $[t_1, t_4]$ is one sample period. In this sample period, $[t_1, t_2]$ is the subinterval for identifying σ_2. Once $\hat{\sigma}_2$ is obtained, an optimization process using Box's complex algorithm is carried out for the evader to obtain a sequence of optimal control values $\{u(t_3), u(t_6), ...\}$. Once the optimization process is completed, a control value from the sequence is issued to govern the next movement. This whole process in $[t_1, t_2]$ repeats for subsequent time intervals (see

Figure 4.4). However, if the $\hat{\sigma}_2$ is assumed fixed for entire encounter, a slightly different scheme (see Figure 4.5) will be used, in which only one identification process and one optimization process are needed and control values are issued following the optimization process.

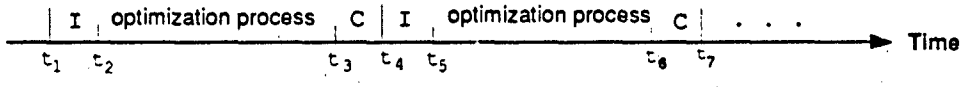


Figure 4.4: Implementation Scheme 1

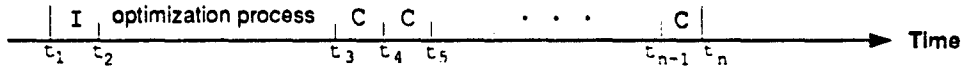


Figure 4.5: Implementation Scheme 2

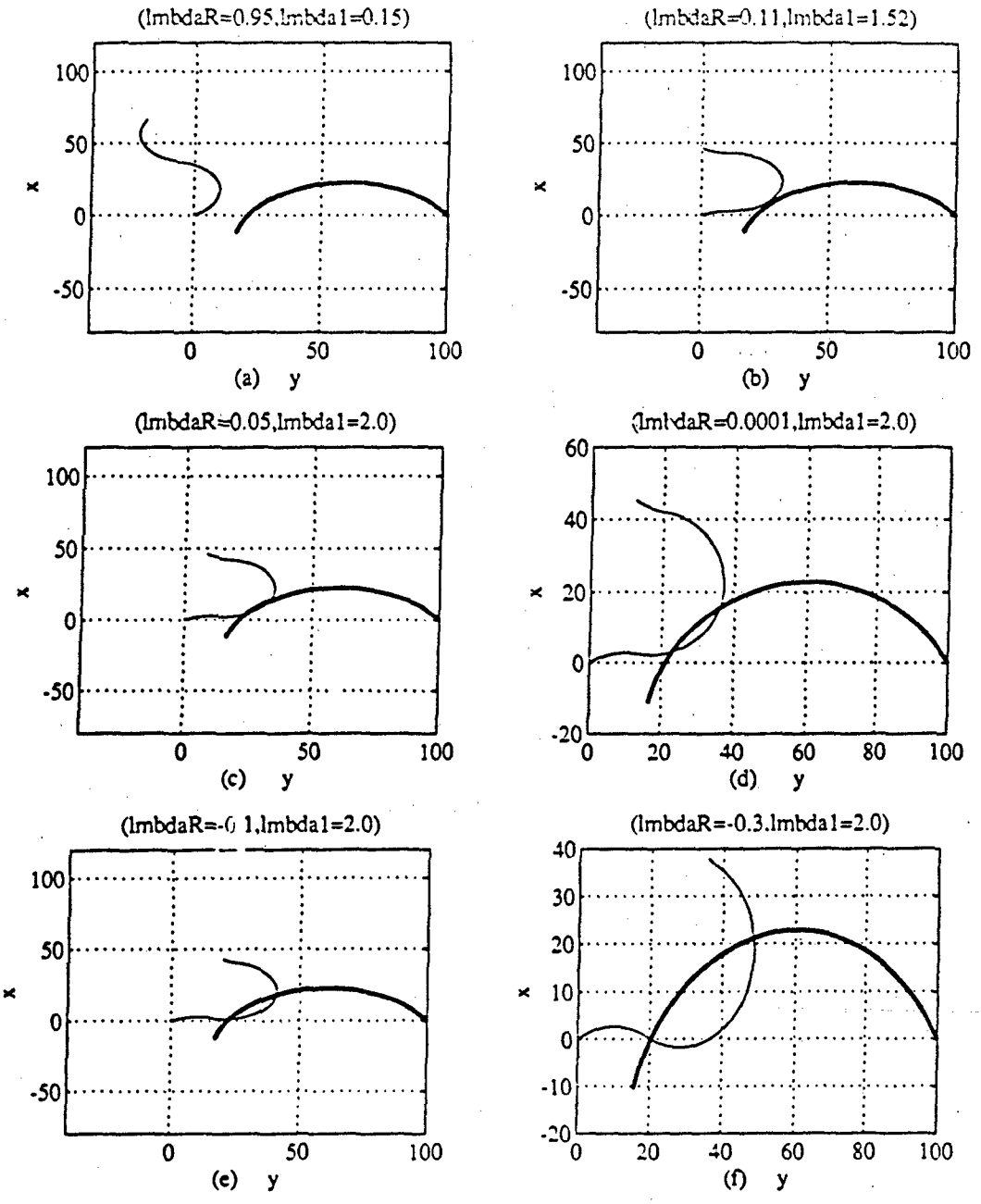


Figure 4.6: Simulation Results (a)

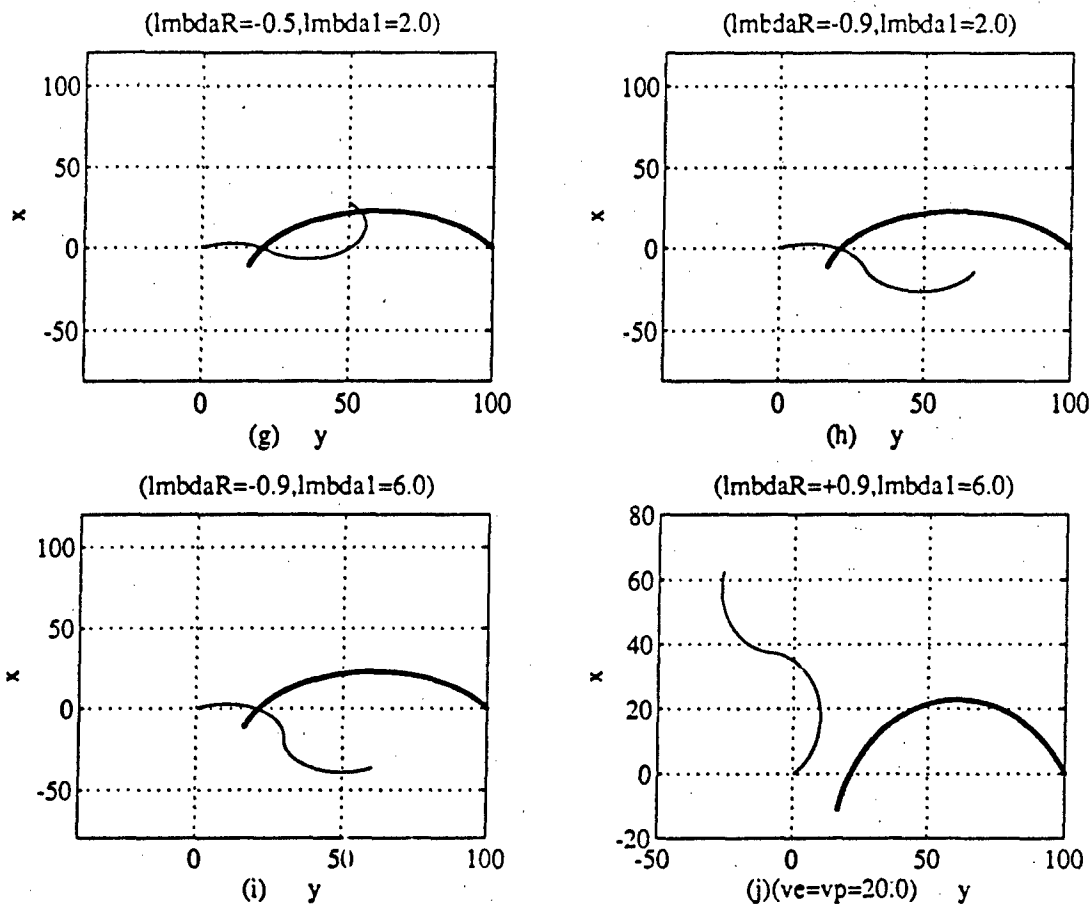


Figure 4.7: Simulation Results (b)

5. Conclusion

It is said that the community of Automatic Control is in an evolutionary phase, which is not a revolution. This seems to be true. But although we have experienced the astonishing revolutionary phase of this field in the early sixties, we should not overlook the great progress we have made and remarkable accomplishments we have achieved since then. We should not forget that each single step in this long march was made by the joint efforts of each individual in our society.

It is the author's hope that this dissertation will be one of those small contributions towards the progress of modern technologies in the field of artificial intelligence methodologies in control systems. The second chapter of this dissertation mathematically formulated the control systems inside the neural networks. Introducing a small feedback loop INSIDE each neuron, instead of a feedback connection in the network, we presented the discretized version of recurrent neural networks. Using these types of neural networks, we showed how to use the internal states directly to construct a feedback control law. What is more important, a network of this type is itself a system and not an unknown "Black Box", and thus its input-output performance can be studied just as is the case for a classical control system. Therefore, many conventional synthesis methods can be directly borrowed to design a controller.

The third chapter of this dissertation discusses the issues of applying neural network techniques to classical differential game problems. To model the real-time game situation more realistically, a configuration, based on the stages in real-life

conflicts, is proposed. Based on the paradigm of semantic control, the configuration can be used further to derive two paradigms of differential games with neural networks. To demonstrate the effectiveness of the method, we carried out a simulation experiment and studied a pursuit-evasion game problem. The same principle, paradigm and structure have the potential of being applicable to an entire class of pursuit-evasion game problems. We also studied the external learning algorithm for a neural controller, which may be used in one of the paradigms discussed previously. To test the algorithm, a real-time aircraft control problem in the presence of windshear has been studied.

The fourth chapter of this dissertation has discussed the Layered Defense Project. The project, which was initiated in June, 1991, is a real-time pursuit-evasion game problem with one evader and multi-pursuers. Based on line-of-sight coordinates this dissertation has discussed and solved the optimal control problem arising from the project. Box's algorithm has been used to find the optimal values for the costates.

Here, the following areas seem to be of sufficient interest to indicate further research and investigation:

1. to consider a robust, multi-purpose, real-time controller for various types of applications and problems, a product of a merger of the advanced techniques in the areas of artificial intelligence and control systems,

2. to more intelligently and massively incorporate parallel computer architecture into a control system, e. g. using neural networks,

3. to incorporate the graphical interface into a workstation-based control system,

4. to establish and fully utilize a data-base for knowledge representation and knowledge process,

5. to more intelligently apply neural network techniques for control systems, which is a promising area since the learning capacity, robustness, and memory capacity of neural networks provide revolutionary tools and mechanisms for the future of control systems.

To summarize, we have the following contributions of this dissertation to the area of artificial intelligence methodologies in aerospace and other control systems:

1. Although several applications are observed in using recurrent neural networks for control systems [73, 74, 93], there is no one in the past few years who has studied the issues of controllability/observability, linearizability via change of coordinates for such type of neural networks for control systems. This dissertation has covered these interesting topics and the results are satisfactory. For the first time, the so-called Separation Principle of Learning and Control is proposed. The significance of this study lies in the thoughts of exploring the intelligence/learning capacity and parallel architectures of neural networks for the purpose of control. This study has shown the promise for future research in this direction.

2. Motivated by the works in [77, 95], we have developed a new approach to differential games with neural networks. The approach which is based on the semantic control theory is more realistic to the real-life conflicts and has the potential of being applicable to an entire class of pursuit-evasion game problems. The study is significant for the community of differential games. In our study, the assumption that both players act optimally at all times

is no longer valid. This is particularly true for air-combat problems since during the real combat modeling an entire encounter in a fixed mathematical equation is essentially not realistic, and both players (assuming only two players in the encounter) always act according to their opponent's perceived action and their goals. Therefore, the configuration based on this idea is much closer to the real-time situation and can be used to develop more advanced algorithms.

3. The aircraft control in the presence of windshear is an interesting problem. The learning algorithm, which is originally developed for the neural controller in Chapter 3, is applied to the control of aircraft encounting windshear. Explicit formulas for evaluating weights in a neural controller have been given. The approach offers the advantages such as being easy to implement in practice, being applicable in several different windshear models without any change of control law.

4. Chapter 4 of this dissertation has discussed another aspect of artificial intelligence techniques for control systems: rule-based expert system applied in a class of pursuit-evasion game problems. Line-of-sight coordinates have been used by several authors such as Shinar [82, 81], in study of pursuit-evasion game problems. Based on the line-of-sight coordinates this dissertation has discussed and solved the optimal control problem arising from the project. There are three main differences between their approaches and ours. First, their solution requires retrointegration of costate equations, which is usually very time-consuming. The derived optimal control solution for our problem has an explicit formula which can be implemented in time-forward fashion. Thus, time-saving in implementing the solution is significant since it

does not require retrointegration which is normally performed in computing optimal control solutions. Second, the Box's complex algorithm has been incorporated into our optimal control problems. This particular aspect is interesting. Third, in our approach, the pursuer's strategy is assumed to be known and fixed during the evaluation of optimal control while in Shinar's work the control strategies for both pursuer and evader are evaluated simultaneously. This observation suggests the potential application of our results in Chapter 3 to this particular project. From above, we can see that our approach does offer several advantages over previous work.

7. Bibliography

[1] J. Alam, L. Berke, and P. L. N. Murthy. Use of Artificial Neural Networks in Nonlinear Material Modeling. *Proceedings of Artificial Neural Networks In Engineering, St. Louis, MO*, pages 607–612, November 1991.

[2] J. S. Albus. A New Approach to Manipulator Controller: The Cerebellar Model Articulation Controller (CMAC). *Transaction of ASME*, pages 220–227, September 1975.

[3] Panos J. Antsaklis. Neural Networks in Control Systems. *IEEE Control Systems Magazine*, 10(3–23), April 1990.

[4] K. J. Aström. Theory and Applications of Adaptive Control - A Survey. *Automatica*, 19:471–481, 1983.

[5] K. J. Aström. Toward Intelligent Control. *IEEE Control Systems Magazine*, pages 60–64, April 1989.

[6] K. J. Aström, J. J. Anton, and K. E. Arzen. Expert Control. *Automatica*, 22(3):277–286, 1986.

[7] J. P. Aubin. *Mathematical Methods in Artificial Intelligence*. December 1991.

[8] Andrew G. Barto, Richard S. Sutton, and Charles W. Anderson. Neuronlike Adaptive Elements That Can Solve Difficult Learning Control Problems. *IEEE Trans. on Systems, Man and Cybernetics*, SMC-13(5):834–846, October 1983.

[9] Tamar Basar and G. J. Olsder. *Dynamic Noncooperative Game Theory.* Academic Press, London/New York, 1982.

[10] M.E. Bennett. Real-time Continuous AI Systems. *IEEE Proceedings*, 134(4):272 - 227, July 1987.

[11] Jan T. Bialasiewicz and Tuan T. Ho. Neural Adaptive Identification And Control. *1st Conference On Artificial Neural Networks In Engineering*, pages 519-524, November 1991.

[12] Arthur E. Bryson and Jr. Yu-chi Ho. Applied Optimal Control - Optimization, Estimation and Control. *John Wiley & Sons, Inc.*, 1975.

[13] George Burgin. Using Cerebellar Arithmetic Computers. *AI Expert*, 7(6):32-41, June 1992.

[14] Fu-Chuang Chen. Back-propagation Neural Networks For Nonlinear Self-tuning Adaptive Control. *IEEE Control Systems Magazine*, 10(6), 1990.

[15] S. Chen, S. A. Billings, and P. M. Grant. Non-linear System Identification Using Neural Networks. *International Journal of Control*, 51(6):1191-1214, 1990.

[16] G. Cybenko. Approximation By Superposition of a Sigmoidal Function. *Mathematics of Controls, Signals and Systems*, 2:303-314, 1989.

[17] A. Davidovitz and J. Shinar. Eccentric Two-target Model for Qualitative Air Combat Game Analysis. *J. of Guidance, Control and Dynamics*, 8(3):325-331, 1985.

[18] A. Davidovitz and J. Shinar. Pursuit-evasion Game Analysis In a Line Of Sight Coordinate System. *The 27th Israel Annual Conference On Aviation and Astronautics*, Tel-Aviv-Haifa, February 1985.

[19] A. Davidovitz and J. Shinar. Two-target Game Model Of An Aircraft With Fire-and-forget All Aspects Missiles. *Journal of Optimization Theory and Applications*, Vol. 63(No. 2):133 - 165, 1989.

[20] Athanasios Sideris Demetri Psaltis and Alan A. Yamamura. A Hierarchical Model For Voluntary Movement and Its Application to Robotics. *Proc. IEEE 1st Int. Conf. Neural Networks(San Diego, CA)*, pages IV551–IV558, June 1987.

[21] Athanasios Sideris Demetri Psaltis and Alan A. Yamamura. A Multilayered Neural Network Controller. *IEEE Control Systems Magazine*, pages 17–21, April 1988.

[22] D.C. Fraser. Aircraft Control Systems - A Projection to the Year 2000. *IEEE Control Systems Magazine*, pages 11-13, Feb. 1985.

[23] W. Frost and D.W. Camp. Wind Shear Modeling for Aircraft Hazard Definition. *FAA Report FAA-RD-77*, 1977.

[24] K. S. Fu. Learning Control Systems and Intelligent Control Systems: An Interaction of Artificial Intelligence and Automatic Control. *IEEE Transaction On Automatic Control*, AC-16(1):70–72, 1971.

[25] Asdrubal Garcia-Ortiz, John Wootton, Ervin Y. Rodin, S. Massoud Amin, Michael Meusey, Chao Ruan, Alan Y. Wu, Prasanta De, and James Revetta.

Applications of Semantic Control to a Class of Pursuer- Evader Problems. *To appear in Mathematical and Computer Modeling*, 1992.

[26] A. A. Gddenberg, B. Benhabib, and K. G.. Fenton. A Complete Generalized Solution To The Inverse Kinematic of Robots. *IEEE J. Robotics and Automation*, 1:14–20, 1985.

[27] P.R. Gorman and T.J. Sejnowski. Analysis of Hidden Units in a Layered Network Trained to Classify Targets. *Neural Networks*, 1:75–89, 1988.

[28] Girish Govind and P. A. Ramamoorthy. Multilayered Neural Networks for Arbitrary Approximation: An Explanation and Simulations. *Proceedings of Artificial Neural Networks In Engineering, St. Louis. MO*, pages 589–594. November 1991.

[29] J. W. Grizzle. Feedback Linearization of Discrete-time Systems. *Analysis and optimization of Systems, Lecture Notes on Control and Information Science*, 83:273–281, 1986.

[30] A. Guez and Z. Ahmad. Accelerated Convergence In The Inverse Kinematics Via Multilayer Feedforward Networks. *Int. Joint Conf. Neural Networks, Washington D.C.*, pages II341 – II349, June 1989.

[31] Aloke Guha. A Hybrid Neural Network Approach to Adaptive Control And Optimization. *1st Conference On Artificial Neural Networks In Engineering*, pages 505–510, November 1991.

[32] R. Hecht-Nielsen. Theory of Backpropagation Neural Networks. *Proc. IJCNN, Washington, D.C.*, pages 593–608, June 1989.

[33] John Hopfield. 'Neural' Computations of Decisions in Optimization Problems. *Biological Cybernetics*, 52:141–152, 1985.

[34] John Hopfield and David Tank. Computing with Neural Circuits: A Model. *Science*, (233), 1986.

[35] K. Hornik, M. Stinchcombe, and H. White. Multilayer Feedforward Neural Networks Are Universal Approximation. *Neural Networks*, 2(5):359–366. 1989.

[36] D. A. Hoskins, J. N. Hwang, and J. Vagners. Iterative Inversion of Neural Networks and Its Applications to Adaptive Control. *IEEE Trans. on Neural Networks*, 3(2):292–301, March 1992.

[37] D.M. Hosmer. A Pilot's View of Intelligent Systems. *Aerospace America*, pages 32–33, 1989.

[38] A. Isidori. *Nonlinear Control Systems.* 1989.

[39] Rufus Issacs. Games of pursuit. *Rand Rep.*, # P-257, 1957.

[40] Rufus Issacs. *Differential Games.* John Wiley and Sons. Inc., 1965.

[41] Bernt Jarmark. A Missile Duel Between Two Aircraft. *J. of Guidance.* 8(4):508–513, 1985.

[42] M. I. Jordan. Generic Constraints On Underspecified Target Trajectories. *Int. Joint Conf. Neural Networks (iJCNN), Washington, DC*, pages I217–I225, June 1989.

[43] Ed. K. S. Sutton. Special Issue on Reinforcement Learning. *Machine Learning*, 8(314), 1992.

[44] T. Kailath. *Linear System.* Prentice-Hall, Eaglewood Cliffs, NJ, 1980.

[45] A. Karakasoglu, S. I. Sudharsanan, and M. K. Sundareshan. Neural Network-based Identification And Adaptive Control of Nonlinear Systems: A Novel Dynamical Network Architecture And Training Policy. *Proc. the 30th CDC, Brighton, England*, pages 180–181, December 1991.

[46] M. Kawato, Y. Uno, and M. Isobe. A Hierarchical Model For Voluntary Movement and Its Application To Robotics. *Proc. IEEE 1st Int. Conf. Neural Networks (IJCNN) Washington, DC*, pages II341–II349, June 1989.

[47] T. Kohonen. An Introduction to Neural Computing. *Neural Networks*, 1(1):3–16, 1988.

[48] S. B. Kooi and K. Khorasani. Control of Wood Chip Refining Using Neural Networks. *Proceedings of Artificial Neural Networks In Engineering, St. Louis, MO*, pages 567–571, November 1991.

[49] Gordon Kraft. Neural Networks and Optimal Control. *Workshop on Aerospace Applications of Neurocontrol*, 1990.

[50] L. Kraft and D. Campagna. A Comparison Between CMAC Neural Network Control and Two Traditional Adaptive Control Systems. *IEEE Control Systems Magazine*, 10(3), April 1990.

[51] Ganesh V. Kudav, Dilip K. Singh, and Sleiman A. Abdallah. Modeling of Linear and Nonlinear Thermal System. *Proceedings of Artificial Neural Networks In Engineering, St. Louis, MO*, pages 575–579, November 1991.

[52] Stephen H. Lane, David A. Handelman, and Jack J. Gelfand. Theory and Development of Higher-Order CMAC Neural Networks. *IEEE Control Systems Magazine*, 12(2):23–30, April 1992.

[53] H. G. Lee, A. Arapostathis, and S. I. Marcus. On the Linearization of Discrete-time Systems. *Int. J. Contr.*, 45:1803–1822, 1987.

[54] H. G. Lee and S. I. Marcus. Approximate and Local Linearizability of Nonlinear Discrete-time Systems. *Int. J. Contr.*, 44:1103–1124, 1986.

[55] George Leitmann and Sandeep Pandey. Aircraft Control For Flight In An Uncertain Environment: Takeoff In Windshear. *Conference On Modeling and Control of Uncertain Systems*, September 1990.

[56] Y. V. Lirov. Artificial Intelligence Methods in Decision and Control Systems. *D.Sc. Dissertation, Washington University*, August 1987.

[57] C. D. Meier and R. O. Stenerson. Recognition Networks for Tactical Air Combat Maneuvers. *Proceedings of the 4th Annual AAAIC Conference, Dayton, Ohio*, October 19[illegible].

[58] J. M. Mendell and R. W. McLaren. *Reinforcement Learning Control and Pattern Recognition Systems in Adaptive, Learning and Pattern Recognition Systems*. New York: Academic, 1970.

[59] A. Meystel. Intelligent Mobile Autonomous System. *Technical Report, Drexel University, Philadelphia, PA 19104*, 1987.

[60] A. Meystel. Theoretical Foundations of Planning and Navigation. *International Journal of Intelligent Systems*, 2(2), 1987.

[61] A. Meystel. Intelligent Control in Robotics. *J. of Robotic Systems*, 5(4). 1988.

[62] A. Meystel. Representation of Decriptive Knowledge For Nested Hierarchical Controllers. *Proc. of the 27th Conf. on Decision and Control, Austin, TX*, pages 1805–1811, December 1988.

[63] A. Miele, T. Wang, C.Y. Tzeng, and W.W. Melvin. Optimization and Guidance of Abort Landing Trajectories in a Windshear. *AIAA Guidance, Navigation and Control Conference*, 1987.

[64] R.K. Miller and T.C. Walker. AI Applications in Engineering. *California, Fairmont Press*, 1988.

[65] W. T. Miller. Real-time Application of Neural Networks For Sensor-based Control of Robots With Vision. *IEEE Trans. Syst. Man, Cybern.*, 19(4):825–831, 1989.

[66] W. T. Miller, F. A. Glanz, and L. G. Kraft III. Application of a General Learning Algorithm to the Control of Robotic Manipulations. *Intl. J. Robotics Res.*, 6(2):84–98, 1987.

[67] W. T. Miller, F. A. Glanz, and L. G. Kraft III. CMAC: An Associative Neural Network Alternative to Backpropagation. *Proceedings of the IEEE*. 78(10), October 1990.

[68] K. S. Narendra. Adaptive Control Using Neural Networks. *Neural Networks For Control, W. T. Miller, III, R. S. Sutton, and P. J. Werbos, Eds. Cambridge, MA: MIT Press*, 1991.

[69] Kumpati S. Narrendra and Kannan Parthasarathy. Identification and Control of Dynamical Systems Using Neural Networks. *IEEE Trans. on Neural Networks*, 1(1):4–27, March 1990.

[70] D. Nguyen and B. Widrow. The Truck Backer-upper: An Example Of Self-learning in Neural Networks. *Int. Joint Conf. Neural Networks (IJCNN) Washington, DC*, pages II357–II363, June 1989.

[71] S. Miyahara N.H. Farhat and K.S. Lee. Optimal Analog of Two-Dimensional Neural Networks and Their Applications in Recognition of Radar Targets. *American Institute of Physics*, pages 146–152, 1986.

[72] H. Nijmeijer and A. J. Vanderschaft. *Nonlinear Dynamical Control Systems*. Springer-Verlag, 1990.

[73] Michael Nikolaou and Vijakumar Hanagaudi. Input-Output Exact Linearization of Nonlinear Dynamical Systems Modelled by Recurrent Neural Networks. *Proceedings of Artificial Neural Networks In Engineering, St. Louis, MO*, pages 575–579, November 1991.

[74] Michael Nikolaou, Yong You, and Haralabos Sarimveis. Process Modeling With Recurrent Neural Networks. *Proceedings of Artificial Neural Networks In Engineering, St. Louis, MO*, pages 595–600, November 1991.

[75] Si-Zhao Qin, Hong-Te Su, and Thomas J. McAvoy. Comparison of Four Neural Net Learning Methods for Dynamic System Identification. *IEEE Trans. on Neural Networks*, 3(1):122–130, January 1992.

[76] E. Y. Rodin. Semantic Control. *Applied Mathematics*, 1(1), 1988.

[77] E. Y. Rodin, Y. Lirov, S. Mittnik, B. McElhaney, and L. Wilbun. Artificial Intelligence in Air Combat Games. *Computer and Mathematics with Applications*, 13:261 – 274, 1987.

[78] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning Internal Representations by Error Propagation. *Parallel Distributed Processing: Explorations In the Microstructure of Cognition, Vol.1, Foundations., Cambridge, MA: Bradfort Books/MIT Press*, 1986.

[79] G. N. Saridis. On the Theory of Intelligent Machines: A Survey. *Proc. of the 27th Conf. on Decision and Control, Austin, Texas*, pages 1799–1804, December 1988.

[80] R. Shaw. Fighter Combat Tactics and Maneuvering. *U.S. Naval Institute, Annapolis, MD*, 1985.

[81] J. Shinar. Evaluation of Sub-optimal Pursuit Evasion Game Strategies for Air Combat Analysis. *Proceedings of AIAA Atmospheric Flight Mechanics Conference*, 11:23 – 29, 1984.

[82] J. Shinar. Analysis of Dynamic Conflicts by Techniques of Artificial Intelligence. *Technical Report*, 1990.

[83] J. Shinar, K. H. Well, and B. Jermark. Near-optimal Feedback Control for Three-dimensional Interceptions. *preprint of ICAS Paper*, (86-5.1.3), The 15th ICAS Congress, London, UK(1986).

[84] D. F. Specht. Probabilistic Neural Networks. *Neural Networks* 3:109–118, 1990.

[85] D. F. Specht. A General Regression Neural Network. *IEEE Trans. on Neural Networks*, 2(6):568–576, November 1991.

[86] Janusz Starzyk and Xiaoming Wu. Approximation Using Linear Fitting Neural Network. *Proceedings of Artificial Neural Networks In Engineering, St. Louis, MO*, pages 619–624, November 1991.

[87] H. E. Stephanou. Knowledge Based Control Systems. *IEEE Workshop on Intelligent Control, RPI, Troy, New York*, pages 116–, 1986.

[88] M. Stinchcombe and H. White. Universal Approximation Using Feedforward Networks With Non-sigmoidal Hidden Layer Activation Functions. *Proceedings of the International Joint Conference on Neural Networks, Washington, D.C. (I:613-I:617), New York, IEEE Press*, 1989.

[89] Janos Sztipanovits, Gabor Karsai, Prashant Wakins, Nobuji Miyasaka, and Koji Okuda. Neural Adaptive Control With Piecewise Linear Approaximation. *1st Conference On Artificial Neural Networks In Engineering*, pages 525–530, November 1991.

[90] David Tank and John Hopfield. Simple 'Neural' Optimization Networks: an A/D converter, signal decision circuit, and a linear programming circuit. *IEEE Transaction on Circuits and Systems*, 33(5):533–541, 1986.

[91] Guilermo Tinetti and Ghassan Adbelnour. Performance of Intelligent and Classical Controllers In an Inverted Pendulum Application. *Proceedings of Artificial Neural Networks In Engineering, St. Louis, MO*, pages 631–636, November 1991.

[92] Thomas L. Vincent, Douglas J. Sticht, and Willy Y. Peng. Aircraft Missile Avoidance. *Operations Research*, 24(3), May-June 1976.

[93] W. Thomas Miller, III, Richard S. Sutton, and Paul J. Werbos, editors. *Neural Networks for Control.* The MIT Press, 1990.

[94] C. J. C. H. Watkins. Learning With Delayed Rewards. *Ph.D Thesis, Cambridge University, Psychology Dept.*, 1989.

[95] Roark David Weil. Artificial Intelligence Methods in Utilizing Low Dimensional Methods of Differential Games. *Doctoral Dissertation*, Washington University, 1990.

[96] P. J. Werbos. Generalization of Backpropagation With Application To a Recurrent Gas Market Model. *Neural Networks*, 1(4):339–356, 1988.

[97] Paul J. Werbos. Neural Networks for Control and Systems Identification. *Proceedings of the 28th Conference on Decision and Control*, December 1989.

[98] H. White. Connectionist Nonparametric Regression: Multilayer Feedforward Networks Can Learn Arbitrary Mappings. *Neural Networks*, 3:535–550, 1990.

[99] N. Wiener. On Control and Communication In the Animal and the Machine. *Cybernetics*, 1948.

[100] S. Zhu and B. Etkin. Fluid Dynamic Model of a Downburst. *UTLAS Report no. 271.*