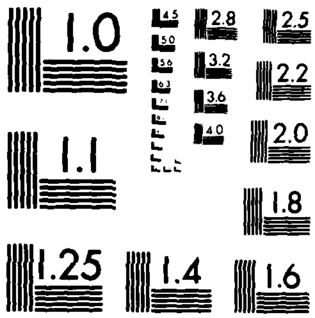
AD-A151 510 EQUILIBRIUM SOLUTIONS STABILITIES AND DYNAMICS OF LANCHESTER'S EQUATIONS W..(U) NAVAL POSTGRADUATE SCHOOL MONTEREY CA A B NING SEP 84 1/2

UNCLASSIFIED F/G 12/1 NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

(2)

NAVAL POSTGRADUATE SCHOOL

Monterey, California

THESIS

EQUILIBRIUM SOLUTIONS, STABILITIES AND DYNAMICS OF LANCHESTER'S EQUATIONS WITH OPTIMIZATION OF INITIAL FORCE COMMITMENTS

by

Ang Bing Ning

September 1984

Thesis Advisor: Paul H. Moose

Approved for public release; distribution unlimited

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER	2. GOVT ACCESSION NO. AD A151 510
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle): Equilibrium Solutions, Stabilities and Dynamics of Lanchester's Equations with Optimization of Initial Force Commitments	5. TYPE OF REPORT & PERIOD COVERED: Master's thesis; September 1984
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): Ang Bing Ning	8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Naval Postgraduate School, Monterey, California 93943	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS: Naval Postgraduate School, Monterey, California 93943	12. REPORT DATE: September 1984
	13. NUMBER OF PAGES: 128
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15. SECURITY CLASS. (of this report): Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

Accession For		
NTIS GRA&I	X	
DTIC TAB	☐	
Unannounced	☐	
Justification		
By		
Distribution/		
Availability Codes		
Dist	Avail and/or Special	
A-1		

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Lanchester's Equations
Equilibrium Solutions
Stabilities
Domains of Attractions
Initial Force Commitments

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Generalised Lanchester-type differential equations are used to study combat processes. This system of non-linear equations has multiple equilibrium solutions which can be determined by a numerical technique called the Continuation Method. Useful properties pertaining to neighborhood stability are derived by considering the lowest-dimensional (1*1) problem. A new set of parameters based on the system asymptotes is defined and used to characterize stabilities. System dynamics are investigated

#20 - ABSTRACT - (CONTINUED)

using phase trajectories which are found to depend on the domains of attraction and stabilities of surrounding equilibria. The effect of varying initial force levels (X,Y) is studied by calculating an objective function which is the difference of the losses at the end of a multistage battle simulation. Based on the minimax theorem, a set of mixed strategies for (X,Y) can be found. For highly unstable warfare with large war resources, instability can be used to influence battle outcome.

Approved for public release; distribution unlimited.

Equilibrium Solutions, Stabilities and Dynamics of Lanchester's Equations with Optimization of Initial Force Commitments

by

Ang Bing Ning
Captain, Republic of Singapore Navy
B. SC. (Honors), University of Singapore, 1979

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
September 1984

Author: ______________________________
Ang Bing Ning

Approved by: ______________________________
Paul H. Moose, Thesis Advisor

____
John M. Wozencraft, Second Reader

____
Harriet B. Rigas, Chairman,
Department of Electrical and Computer Engineering

____
John N. Dyer
Dean of Science and Engineering

ABSTRACT

Generalized Lanchester-type differential equations are used to study combat processes. This system of non-linear equations has multiple equilibrium solutions which can be determined by a numerical technique called the Continuation Method. Useful properties pertaining to neighborhood stability are derived by considering the lowest-dimensional (1*1) problem. A new set of parameters based on the system asymptotes is defined and used to characterize stability. System dynamics are investigated using phase trajectories which are found to depend on the domains of attraction and stabilities of surrounding equilibria. The effect of varying initial force levels (X,Y) is studied by calculating an objective function which is the difference of the losses at the end of a multistage battle simulation. Based on the minimax theorem, a set of mixed strategies for (X,Y) can be found. For highly unstable warfare with large war resources, instability can be used to influence battle outcome.

Addition [illegible] theses, computer [illegible] military commanders, decision making, [illegible]

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENT

I would like to thank my thesis advisor, Professor Paul Moose for his patience in guiding and assisting me in this thesis. I also wish to thank Professor John Wozencraft who has done much more than just being a second reader in his effort to help me. For all the support, encouragement and patience in typing this manuscript, I am grateful to my wife.

I. INTRODUCTION

Since World War II, combat modeling, simulation and analysis have been the subjects of considerable research. The objectives of this research are to support defense decision making and doctrinal developments during peace and war time. During peacetime defense-planners are primarily concerned with weapon procurement, development, acquisition, organisation and structuring. During war time it is believed that a better understanding of the quantitative aspects of attrition can help commanders make better command and control decisions.

Combat processes involve complicated interactions between opposing forces. These interactions are often influenced by many external factors such as environment, troop quality and tactics. There are different types of combat models such as war games, simulations and analytical models. A fundamental requirement for a good model is that it must be of a fairly high degree of operational realism, since otherwise they would not be credible to military planners. On the other hand, excessively complicated models can make the mathematics too difficult to handle.

In this thesis, a generalised Lanchester [Ref. 1] model which contains area-fire, aimed-fire, self-attrition and replenishment coefficients is used. It consists of a system of 2N bilinear equations and belongs to the general category of analytical models. The model is rich enough to treat modern combined-arms operations involving heterogeneous forces. It is also possible to extend the model to analyse operations on two or more fronts.

Among the many important issues that could be analysed using this model, the problem of optimum force distribution had been studied by Wozencraft and Moose (1983). In their paper [Ref. 2], an objective function was chosen as the difference of the aggregate attrition rates. It was shown that the optimization problem is mathematically equivalent to a matrix game. Hence, the model has a saddle-point solution with corresponding optimum force distribution vectors x and y for Blue and Orange forces respectively.

In addition, the neighborhood stability of the model at the operating point (x^* and y^*) was also investigated. By defining two parameters, K1 and K2 which are obtained by considering small perturbations around the operating points, a great deal could be learned about stability.

Motivated by these results, much of the work done during the initial part of this thesis was directed at studying the effect of stability on battle outcome. The ultimate question is, how do we exploit the knowledge of stability of an operating point to influence battle outcome? Before this question can be answered, it appears that there is a need for a better understanding of the equilibrium points. Chapter III is devoted to finding and understanding the equilibrium solutions and their stability behavior. Like many other nonlinear system of equations, the Lanchester's model adopted here has multiple equilibria. Stability analysis [Ref. 3] of a non-linear system is usually done by methods which do not require prior knowledge of the equilibrium solutions. One example of such a method is the Liapunov method [Ref. 4]. If, by some realizable means, the equilibrium solutions can be found explicitly then there is no need to rely on these indirect methods which are often difficult to implement.

One of the reasons for resorting to the Liapunov method is the difficulty in obtaining equilibrium solutions of a non-linear system. Many numerical methods are unsuitable for reasons such as difficulty in obtaining good initial guesses, non-convergence, ill-conditioning and so forth. Fortunately, a powerful numerical technique called the Continuation Method can be applied for our purpose. This method not only finds all the solutions (i.e. it is exhaustive), it does not even require initial guesses.

In order to gain a firm grasp on the dynamics of the system surrounding the equilibria, it is helpful to temporarily focus attention on the homogeneous (1*1) system. In spite of its simplicity, the 1*1 system is not devoid of the essential characteristics of the N*N system. In fact, the 1*1 model is sufficiently sophisticated for certain analyses in which the opposing forces can be assumed to be homogeneous. As we proceed through Chapter IV, it will become clear that much insight into the stability and system dynamics could be gained by merely considering the 1*1 system. Part of the chapter is devoted to the derivations and interpretations of the relations between system asymptotes, locations of equilibrium points and stability. The dynamics of the system are studied using the idea of phase trajectories. These trajectories represent changes of force levels with time and they will be shown to depend not only on the stabilities of equilibrium points but also on the domains of attraction.

Chapter V concentrates on battle outcome which is one of the main issues facing a commander. It encompasses many issues such as, (1) Who will win and by what margin? (2) What is the length of battle? (3) How do initial deployments affect battle outcome? (4) Which parameters affect battle outcome most? But we will only address the two following subjects :

(a) The effect of stability on battle outcome;

(b) The effect of varying X and Y, the initial force levels.

The basic approach is to define a multistage battle with a predetermined condition for termination. The resultant payoff matrix can then be used to obtain the optimum set of mixed strategies. An example, which employs KOREAN WAR data, is presented for the purpose of illustrations and discussions.

The essence of the findings are:

1. Unstable operating conditions can be exploited to influence battle outcome, especially when total war resources are large. The effect on battle outcome is more pronounced for highly unstable warfare;
2. Initial force deployment can be optimized in accordance with a set of mixed strategies.

We conclude this introduction by stating two of the outstanding issues. The first question is the extent to which one can replace the N*N problem by the 1*1 problem. The motivation to find an equivalent 1*1 system stems from (1) our better understanding of the 1*1 system, (2) ease of presenting and visualizing two-dimensional pictures, and (3) savings in computational effort.

The second question concerns replenishment rates. In this thesis, the replenishment terms used in the model have been constant. It is therefore reasonable to ask, how to modify replenishment terms to reflect a higher degree of operational realism? In other words, are there more suitable time-dependent replenishment rates $\tilde{r}(t)$?

II. LANCHESTER'S EQUATION

A. BACKGROUND

Combat models have been studied as a form of decision aid for defense planning. A wide variety of defense planning problems, ranging from force structuring and weapon selection to rates of deployment in battles have been analysed using combat models. There are many different types of models. They can be loosely categorized as either war games, simulations or analytical models. Discussions on the nature, advantages and shortcomings of each can be found in [Ref. 5].

Our attention will be focused on a generalized Lanchester's [Ref. 5] model, which is an analytical model. It consists basically of a system of ordinary differential equations describing the mutual interactions between opposing combat forces. Although earlier works in Lanchester's model [Ref. 6] employed only a few terms in the equations, modern high speed computers enable more generalised, realistic and responsive versions to be used.

Consider a battlefield with opposing forces, Blue and Orange, denoted by $\{x_i\}$ and $\{y_i\}$ respectively. The subscripts i, j refer to the type of forces such as infantry, tanks, artillery, etc. A generalised version of Lanchester's model given by

$$\begin{aligned}
\dot{x}_i &= -x_i u_i - x_i \sum_j a_{ij} y_j - \sum_j b_{ij} y_j + r_i \\
\dot{y}_j &= -v_j y_j - y_j \sum_i x_i c_{ij} - \sum_i x_i d_{ij} + s_j
\end{aligned} \qquad \text{(eqn 2.1)}$$

$$i = 1,2,\ldots,I$$
$$j = 1,2,\ldots,J$$

where

u_i , v_j = self-attrition coefficients

a_{ij}, c_{ij} = area-fire attrition coefficients

b_{ij}, d_{ij} = aimed-fire attrition coefficients

r_i , s_j = replenishment coefficients

is adopted in this thesis.

Note that in general $I \neq J$, implying that the force compositions may be different for the two sides. It is also possible to extend the above formulation to a scenerio involving more than one battlefield.

In the next two sections, the highlights of the work done by Wozencraft and Moose (1983) are given. The work done in this thesis is a continuation and extention of their work. The detailed derivations of the results obtained by them can be found in [Ref. 2], and hence are not included here.

B. OPTIMUM FORCE DISTRIBUTION

The question of optimum force distribution arises in combined-arms operations. The problem is fundamentally this: Given aggregated forces X, Y, how should one distribute them among the different types x_i and y_j , $i = 1,2\ldots,I$, $j = 1,2\ldots,J$? Since loss rate is one of the fundamental concepts in combat modeling, it is reasonable to choose this measure as a starting point. The objective function was chosen to be

$$M \triangleq \sum_i (\dot{x}_i - r_i) - \sum_j (\dot{y}_j - s_j) \qquad \text{(eqn 2.2)}$$

For this choice of M, it was shown that there exists optimum force distribution (row and column) vectors x^* and y^* such that for any other vectors x and y

$$x\hat{A}y^* < M^* < x^*\hat{A}y \qquad \text{(eqn 2.3)}$$

where

$M^* = x^*\hat{A}y^*$

$\hat{A}$ = matrix determined by attrition coefficients and the aggregate force levels X and Y

The resemblance of this result to the Minimax theorem [Ref. 7] in matrix games is very striking. Indeed, this result holds precisely because M can be written in a form mathematically equivalent to a matrix game. Consequently, it is not surprising that one can solve for the optimum vector x^* and y^* by means of a Linear Program. An interactive program to solve a 2*2 program is given in Appendix A.

C. NEIGHBORHOOD STABILITY

Equilibrium conditions can be achieved if the replenishment rates are chosen to make

$$\dot{x}_i = \dot{y}_j = 0$$
$$i = 1,2\ldots,I$$
$$j = 1,2\ldots,J$$

at $x = x^*$ and $y = y^*$. Following the usual approach in the analysis of nonlinear system stability, equation 2.1 can then be transformed into a system of linear equations.

$$\begin{bmatrix} \delta\dot{x} \\ \delta\dot{y} \end{bmatrix} = -\bar{C} \begin{bmatrix} \delta x \\ \delta y \end{bmatrix} \quad ; \quad \bar{C} = \begin{bmatrix} \hat{A} & \hat{B} \\ \hat{D} & \hat{C} \end{bmatrix}$$

$\bar{C}$ is called the conflict matrix and its elements are determined by the attrition coefficients and the optimum vectors x^* and y^*. For the system of equations 2.1, $\hat{A}$ and $\hat{C}$ are diagonal matrices. It was shown that two parameters k_1, k_2 partially characterize the stability of the system. k_1 and k_2 turn out to be the column sums of the left and right side of the matrix

$$\tilde{C} \triangleq \left[\begin{array}{c|c} -\hat{A} & \hat{B} \\ \hline \hat{D} & -\hat{C} \end{array}\right]$$

Denoting the elements of the submatrices $\hat{A}$, $\hat{B}$, $\hat{C}$, $\hat{D}$, by $\hat{a}_{ij}$, $\hat{b}_{lj}$, $\hat{c}_{jj}$ and $\hat{d}_{li}$ respectively, k_1 and k_2 can be written as

$$k_1 = -\hat{a}_{ii} + \sum_{l} \hat{d}_{li}$$

$$k_2 = -\hat{c}_{jj} + \sum_{l} \hat{b}_{lj}$$

independent of the columns i, j. Furthermore, it was shown that the following relation holds

$$\delta\dot{X} - k_1 \delta X = \delta\dot{Y} - k_2 \delta Y$$

where

$$\delta X \triangleq \sum_i \delta x_i \quad ; \quad \delta Y \triangleq \sum_j \delta y_j$$

It was found that the equilibrium point (x^*, y^*) is stable if k_1 and k_2 are negative. If k_1 and k_2 are positive, then the system is 'unstable'.[1] Furthermore, values of k_1 and k_2 and hence the stability of the operating point was found to be affected by the aggregate X and Y.

---

[1] More generally, it can be shown that $k_1 < \lambda_0 < k_2$, where λ_0 is the maximum eigenvalue of $-C$.

III. MULTIDIMENSIONAL (N*N) SYSTEM

A. NATURE OF N*N PROBLEM

The interesting results highlighted in the last chapter provided motivation to extend the body of knowledge. A study of the effect on stability of battle outcome seems to have important potentials for applications. Should a commander strive to establish a stable operating point, and if so, under what conditions? Also, what is the optimum initial level of forces he should deploy and how many should he maintain in reserve? To answer these questions, more knowledge about the nature of these equilibria and their stability behavior is required.

The next section outlines the kind of problems we would expect to see and their potential complexity. It is followed by a section on finding the equilibrium solutions.

1. Existence of Multiple Equilibria

An N*N system is in equilibrium if the replenishment rates r_i, s_j are such that there is no change in the force levels ($\dot{x}_i = \dot{y}_j = 0$). The system of equations becomes

$$
\begin{aligned}
0 &= -x_i u_i - x_i \sum_j a_{ij} y_j - \sum_j b_{ij} y_j + r_i \\
0 &= -v_j y_j - y_j \sum_i x_i c_{ij} - \sum_i x_i d_{ij} + s_j
\end{aligned}
\qquad \text{(eqn 3.1)}
$$

$$i,j = 1,2,\ldots,N$$

where, for simplicity, i and j are each assumed to have N types of forces.

A 2N-tuple vector, $\bar{z} \triangleq (\bar{x}, \bar{y})$ which satisfies equation 3.1 is an equilibrium solution. Like many nonlinear systems of equations, equations 3.1 have more than one equilibrium point. Geometrically, these equilibrium points are at the intersections of a set of hypersurfaces in the 2N-dimensional space. To help in visualizing the geometry, we can look at an example using a 1*1 system as shown in figure 3.1. In this case the hypersurfaces simply reduce to hyperbolic curves.

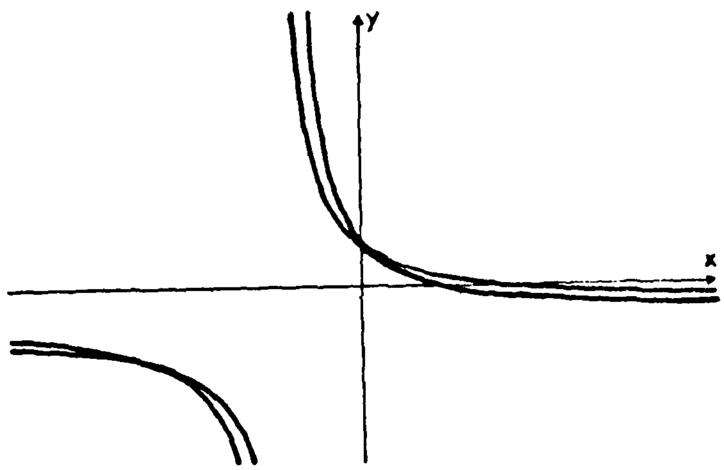


Figure 3.1 Equilibrium Points at Hyperbolic Intersections.

The existence of multiple equilibria makes the analysis of the N*N problem very interesting but difficult. In chapter IV, some illustrations on how the locations of these equilibria affect phase trajectories will be presented.

A few other interesting questions arise spontaneously. For instance, h́ . many of these equilibria are there

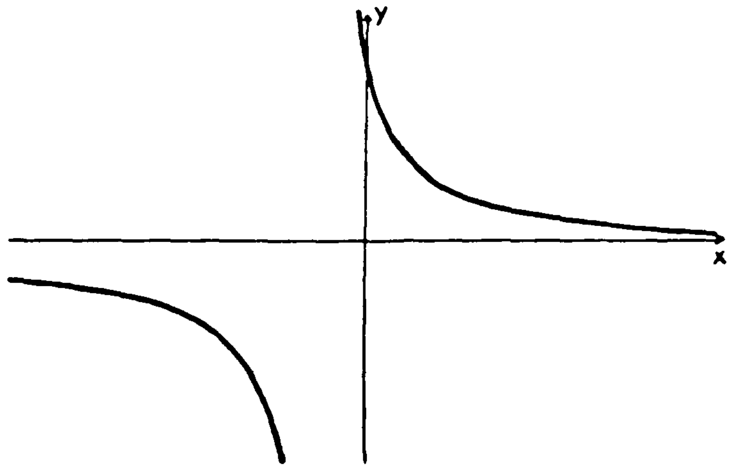


Figure 3.2   Infinite Number of Equilibria.

in an N*N problem? The answer to this question is not immediately obvious just by looking at equation 3.1; however, it emerges quite naturally when the Continuation Method is considered in Section IIIB. It will be seen then that an N*N system has, in general, N_k equilibrium points where

$$N_k = \sum_{i=0}^{N} \binom{N}{i}^2 = \binom{2N}{N}$$

Two exceptions, or degenerate cases, have been observed, namely: (1) when some or all of the hypersurfaces merge there are an infinite number of equilibrium points, (see Figure 3.2), (2) when some or all of the hypersurfaces intersect in such a manner that repeated equilibria are formed, the number of distinct equilibria is less than N_k. Figure 3.3 illustrates such a degeneracy.

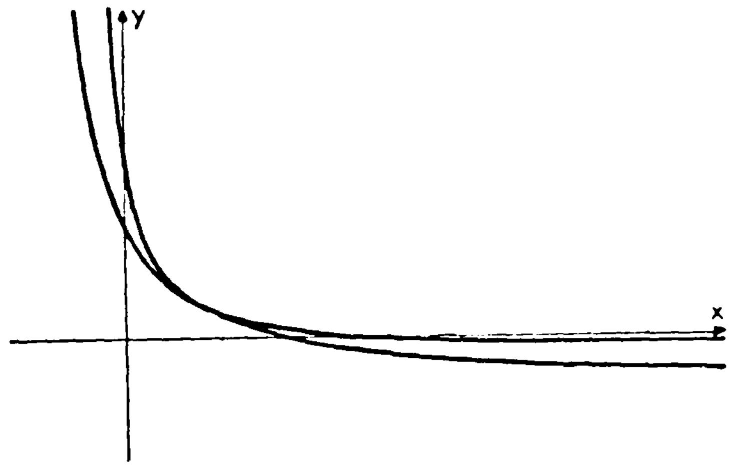


Figure 3.3 Repeated Equilibria.

2. Stability and Domains of Attraction

Each equilibrium point in an N*N system may or may not be stable depending on whether or not its equilibrium point can be maintained. The property of neighborhood stability is important because it has a strong influence on the phase trajectories. Generally, if an operating point is stable (the maximum eigenvalue is negative), then any perturbation away from that point results in the system returning to the same point. Conversely, perturbations about an unstable point results in divergence from that point.

The notion of domains of attraction is also critical when determining phase trajectories. Any operating point within this domain or region will be "attracted" toward a stable equilibrium point. In short, a domain of attraction is a volume in the 2N-dimensional space surrounding a stable equilibrium point. Figure 3.4 shows a typical domain in which some of the trajectories are shown converging to a stable equilibrium point.

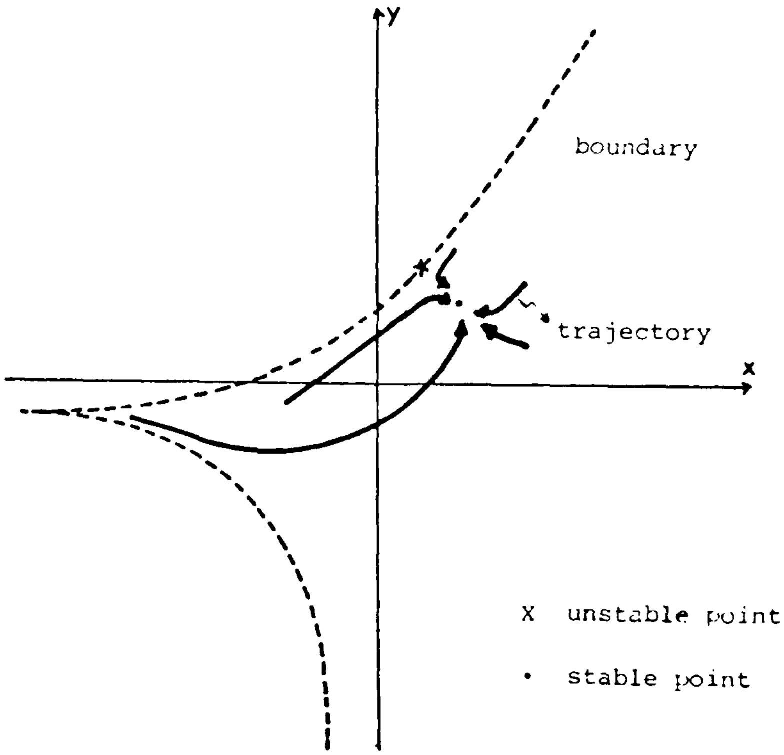


Figure 3.4 Domains of Attraction.

Domains of attraction are separated by boundaries which are invariant curves in 1*1 problems and invariant hypersurfaces in N*N problems. A boundary surface may be considered as an infinite number of invariant curves placed side by side. A boundary curve is the locus of points that approach an unstable point from both sides. The boundary line can be obtained by backward integration (i.e. using negative time in equation 2.1) starting just on either side of an unstable point. The rationale behind this method is that to approach an unstable equilibrium, a point must remain <u>exactly</u> on the boundary. If this is not the case, then the point will be attracted into the domains and move toward a stable point or infinity. By performing a backward integration, we are actually retracing the path taken by a point which previously approached the unstable equilibrium point. This method requires knowledge of the unstable equilibria, but this is made feasible because the Continuation Methods can be used to find all equilibrium solutions.

B. FINDING THE EQUILIBRIUM SOLUTIONS

To obtain a set of equilibrium solutions, one has to solve equation 3.1, which can be written using a more compact notation as

$$F(\bar{z}) = \bar{0} \qquad \text{(eqn 3.2)}$$

where

F(.) represents the right-hand side of equation 3.1

$\bar{z} = (\bar{x},\bar{y})$

$\bar{0}$ = zero vector

It is well known that numerical techniques for solving nonlinear equations are not always successful. Since equation 3.2 describes a bilinear system, one should expect to face similiar difficulties when attempting to solve it numerically.

Most numerical methods for root finding generally require that a fairly good initial guess ($\bar{z}_0$) be known so that some convergent iteration process

$$\bar{z}_{n+1} = g(\bar{z}_n)$$

brings the approximated root closer and closer to $\bar{z}^*$ the desired equilibrium solution or root. In practice, the following difficulties are often encountered :

(1) The convergence condition of the algorithm must be ensured;

(2) Finding an initial guess that is sufficiently close to the correct solution is difficult, especially for higher dimensions;

(3) Even if a good initial guess has been obtained, the numerical process may still be plagued by ill-conditioning, saddle points, etc.;

(4) Not all the solutions are guaranteed to be found.

1. Continuation Method

Fortunately, the above problems are avoided if a numerical method called the Continuation Method [Ref. 8] is used. This technique, which is sometimes called The Imbedding Method, has been successfully applied in many fields. It introduces an artifical guide which will channel the iterates toward a specific solution. Such a guiding principle is actually a knowledge of the existence of a suitable curve connecting an initial point with the desired solution.

Continuation Method has significant advantages over other numerical techniques. Most importantly, a good initial guess is not necessary and all the solutions can be obtained.

a. Basic Theory

Given the problem $F(\bar{z}) = \bar{0}$ to solve, the first step is to embed it into a homotopy or a parameterized set of problems, $H(\bar{z},t)$. The requirements on $H(\bar{z},t)$ are :

(1) $H(\bar{z},1) = F_1(z) = \bar{0}$ is the original problem

(2) $H(\bar{z},0) = F_0(z) = \bar{0}$ has a trivial or easily computed solution

For example, a homotopy could be :

$$H(\bar{z},t) = tF(\bar{z}) + (1-t)F_0(\bar{z}) \quad , \; t\in[0,1] \qquad \text{(eqn.3.3)}$$

Using the above parameterization, the simple problem of $F_0(\bar{z}) = \bar{0}$ is deformed into the desired one, $F_1(\bar{z}) = \bar{0}$. This is done by calculating the solution to the deformed problem at each stage of the deformation. The existence of a continuous curve such that $H(\bar{z}(t), t)$ is a solution to $H(.,.) = 0$ for all $t \in [0, 1]$ is assumed.

b. Implementation

To actually carry out the above continuation process one usually differentiates $H(.,.)$ to form

$$\dot{H}(\bar{z}(t),t) = 0 \qquad \text{(eqn 3.4)}$$

Using equation 3.4, $\dot{\bar{z}}$ can be written as a function of $\bar{z}$ and t as given in equation 3.5. The function, $h(.,.)$ is preferably a linear function that can be integrated numerically.

$$\dot{\bar{z}} = h(\bar{z},t) \qquad \text{(eqn 3.5)}$$

Together with the initial condition $\bar{z}(0) = \bar{z}_0$, equation 3.5 is actually an initial value problem which can be integrated numerically. The solution at t=1 is then the solution to the original problem $F(\bar{z}) = \bar{0}$.

2. Algorithm to Obtain 2*2 Equilibrium Problem

A 2*2 Lanchester problem is first formulated into a Continuation process. It is followed by a discussion on how the accuracy of the method can be improved. The last part of this subsection includes a note on the number of equilibrium points in an N*N problem.

a. Formulation

For the 2*2 problem, $F(\bar{z}) = \bar{0}$ is explictly

$$-z_1(u_1+a_{11}z_3+a_{12}z_4) + r_1 - b_{11}z_3 - b_{12}z_4 = 0$$
$$-z_2(u_2+a_{21}z_3+a_{22}z_4) + r_2 - b_{21}z_3 - b_{22}z_4 = 0$$
$$-z_3(u_3+c_{11}z_1+c_{21}z_2) + r_3 - d_{11}z_1 - d_{21}z_2 = 0$$
$$-z_4(u_4+c_{12}z_1+c_{22}z_2) + r_4 - d_{12}z_1 - d_{22}z_2 = 0$$

The homotopy is formed by writing

$$\begin{aligned} H_1(\bar{z}) &= H_1(\bar{z},0) + t(r_1-b_{11}z_3-b_{12}z_4) = 0 \\ H_2(\bar{z}) &= H_2(\bar{z},0) + t(r_2-b_{21}z_3-b_{22}z_4) = 0 \\ H_3(\bar{z}) &= H_3(\bar{z},0) + t(r_3-d_{11}z_1-d_{21}z_2) = 0 \\ H_4(\bar{z}) &= H_4(\bar{z},0) + t(r_4-d_{12}z_1-d_{22}z_2) = 0 \end{aligned} \qquad \text{(eqn 3.6)}$$

where

$$H_1(\bar{z},0) = -z_1(u_1+a_{11}z_3+a_{12}z_4) = 0$$
$$H_2(\bar{z},0) = -z_2(u_2+a_{21}z_3+a_{22}z_4) = 0$$
$$H_3(\bar{z},0) = -z_3(u_3+c_{11}z_1+c_{21}z_2) = 0$$
$$H_4(\bar{z},0) = -z_4(u_4+c_{12}z_1+c_{22}z_2) = 0$$

Next, we differentiate equation 3.6 with respect to t and put it in a matrix form

$$\underline{A}\dot{\bar{z}} = \underline{B} \qquad \text{(eqn 3.7)}$$

where

$$\underline{A} = \begin{bmatrix} u_1+a_{11}z_3+a_{12}z_4 & 0 & a_{11}z_1+b_{11}t & a_{12}z_1+b_{12}t \\ 0 & u_2+a_{21}z_3+a_{22}z_4 & a_{21}z_2+b_{21}t & a_{22}z_2+b_{22}t \\ c_{11}z_3+d_{11}t & c_{21}z_3+d_{21}t & u_3+c_{11}z_1+c_{21}z_2 & 0 \\ c_{12}z_4+d_{12}t & c_{22}z_4+d_{22}t & 0 & u_4+c_{12}z_1+c_{22}z_2 \end{bmatrix}$$

$$\bar{z} = [z_1, z_2, z_3, z_4]^T$$

$$\underline{B} = \begin{bmatrix} r_1-b_{11}z_3-b_{12}z_4 \\ r_2-b_{21}z_3-b_{22}z_4 \\ r_3-d_{11}z_1-d_{21}z_2 \\ r_4-d_{12}z_1-d_{22}z_2 \end{bmatrix}$$

Equation 3.7 can now be integrated numerically using one of the readily available integration routines.

We have assumed that the trivial solution to $H(\bar{z}, 0) = \bar{0}$ has been previously found.

b. Improving Accuracy

Numerical integration of equation 3.7 inevitably produces some errors at each iteration. Since the

Continuation method relies on following curves to arrive at the desired solution, it is esssential that each iterate remains close to the actual curve. It is necessary to include a way to correct the approximated position by means of a corrector step. The combination of integration and correction is often called a "predictor-corrector step"

This process of prediction-correction is shown in Figure 3.5 where each integration error has been exaggerated for illustrative purposes. The algorithm to be presented later employs an IMSL routine called ZSCNT for the predictor step. Other forms of curve following routine can also be found in the literature, and are briefly mentioned in [Ref. 8].

c. Trivial Solution

The trivial system $H(\bar{z},0) = \bar{0}$ was chosen to be

$$
\begin{aligned}
H_1(\bar{z},0) &= -z_1(u_1+a_{11}z_3+a_{12}z_4) = 0 \\
H_2(\bar{z},0) &= -z_2(u_2+a_{21}z_3+a_{22}z_4) = 0 \\
H_3(\bar{z},0) &= -z_3(u_3+c_{11}z_1+c_{21}z_2) = 0 \\
H_4(\bar{z},0) &= -z_4(u_4+c_{12}z_1+c_{22}z_2) = 0
\end{aligned}
\qquad \text{(eqn 3.8)}
$$

In non-degenerate cases, there are six solutions corresponding to equation 3.8. The result is derived in Appendix B which also deduces the number of trivial solutions for an N*N problem to be

$$N_k = \sum_{i=0}^{N} \binom{N}{i}^2 \qquad \text{(eqn 3.9)}$$

The method of obtaining the trivial solutions is given in Appendix B. Using a combinatorial identity, N_k can be written as

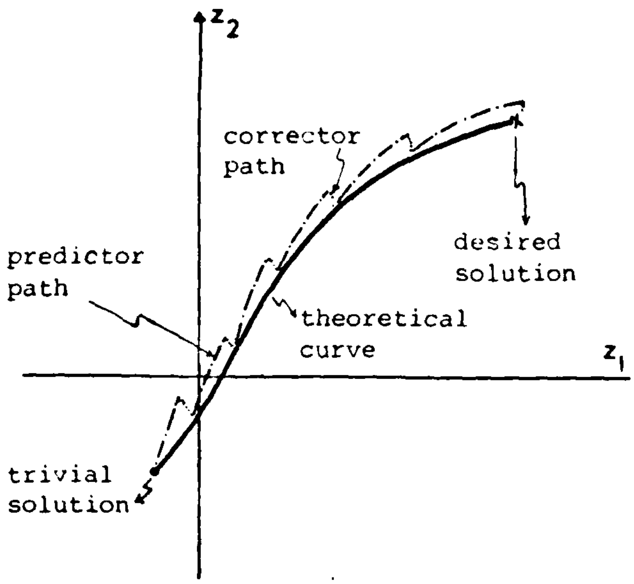


Figure 3.5 Integration Path using Predictor-Corrector.

$$N_k = \binom{2N}{N}$$

Each continuation process starts from a trivial solution $\tilde{z}_0$ and follows a specific curve until it reaches the equilibrium point. Consequently, the number of equilibrium points will also be N_k. As mentioned in section IIIA, the two exceptions are situations involving infinitely-many and repeated equilibria. Situations involving degeneracy are discussed in Appendix B.

d. Algorithm

(1) <u>Singularity Treatment</u>. In Continuation Method algorithms [Ref. 8], it is sometimes necessary to give special treatment to cases in which the curves being

followed by the integration routine pass through a singularity. Experimentally, it had been observed that in our problem, the singularity took on the form shown in Figure 3.6. Corrective measures were necessary to ensure that upon crossing the singularity, the large magnitude was preserved but the sign was changed; otherwise the curve might terminate at an equilibrium point which was not the intended one.

In the algorithm, the presence of the singularity is detected by monitoring the rate of change of the individual component z_i. Once identified, this fast-changing and large-magnitude component (z_F) is monitored at each step t where $0 = t_0 < \ldots < \ldots t_k < t_{k+1} < \ldots < t_{end} = 1$. When z_F is found __not__ to cross the singularity and end up at approximately $-z_F$, the algorithm attempts to correct this irregularity by artificially making $z_F = -z_F$ before the next predictor step commences.

(2) __Flowchart.__ The flowchart for the algorithm is given in figure 3.7. Only the major steps have been shown. The program listing is given in Appendix C.

3. Example and Results

Consider as an example a 2*2 problem with the following attrition coefficients

$$A = \begin{bmatrix} 1.0 & 0.3 \\ 0.6 & 0.9 \end{bmatrix} \quad C = \begin{bmatrix} 1.2 & 1.0 \\ 1.1 & 0.6 \end{bmatrix}$$

$$C = \begin{bmatrix} .15 & 0.1 \\ .15 & 0.3 \end{bmatrix} \quad D = \begin{bmatrix} 0.0 & 0.0 \\ 0.0 & 0.0 \end{bmatrix}$$

$$U = \begin{bmatrix} 0.3 & 0.3 \end{bmatrix} \quad V = \begin{bmatrix} 0.1 \\ 0.2 \end{bmatrix}$$

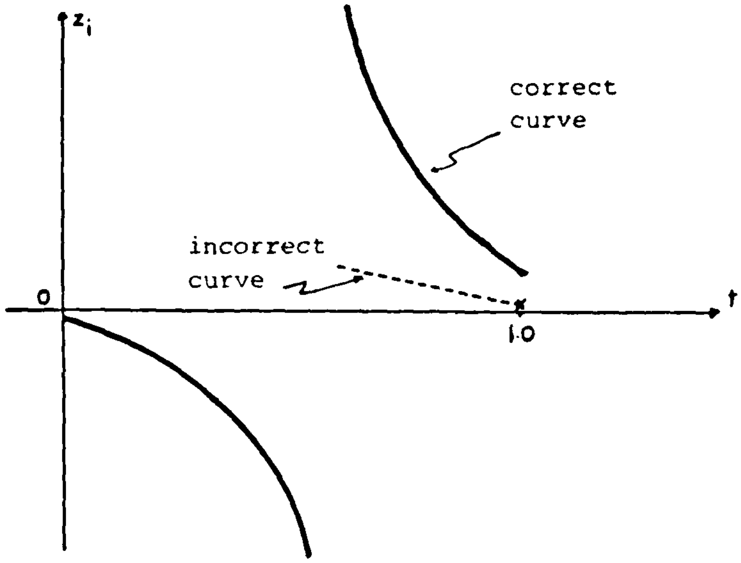


Figure 3.6 Curve Passing through Singularity.

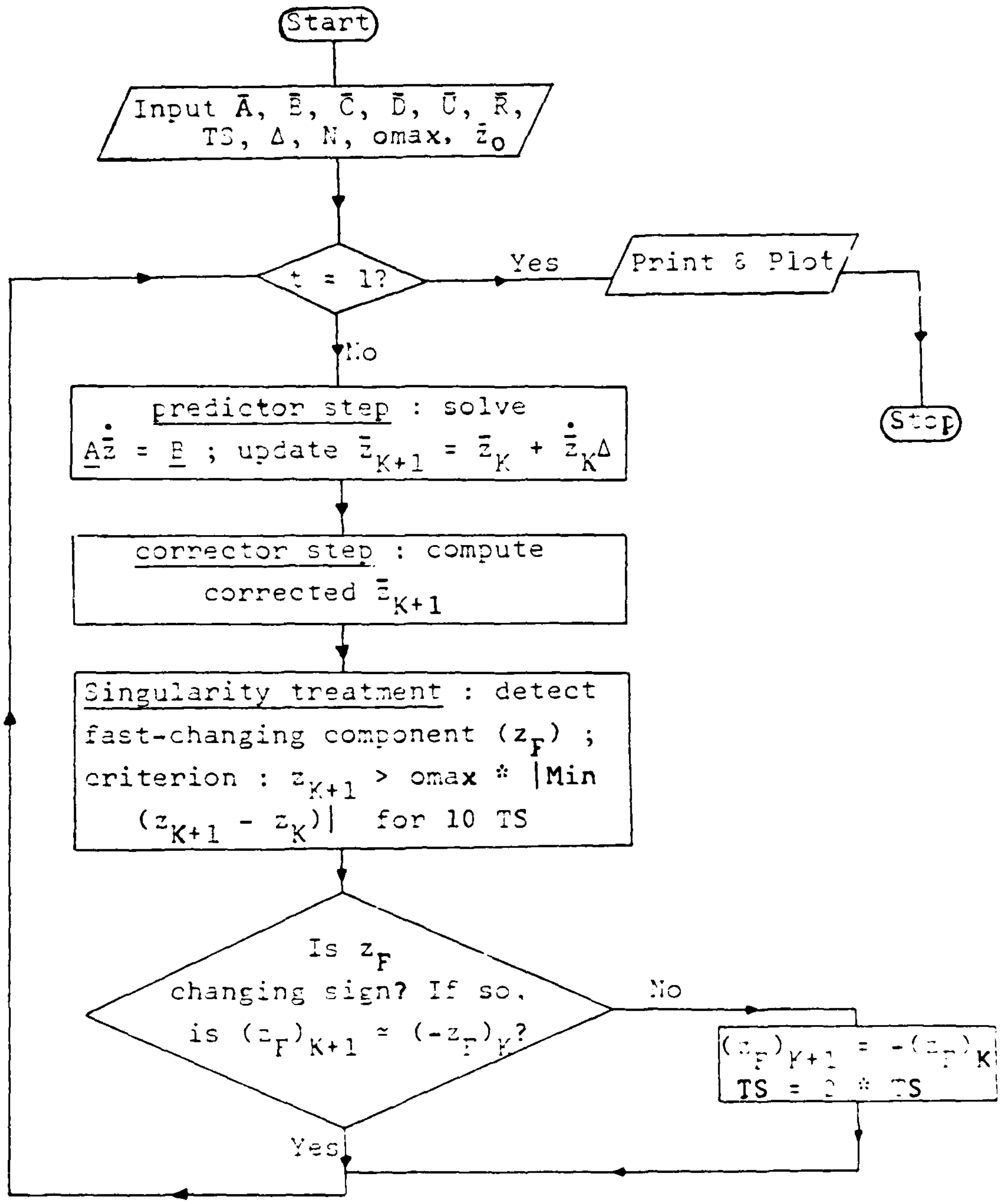
The trivial solutions are first computed and serve as one of the inputs to the program. The program obtains the values of $\bar{z}(t)$ and plots each component ($z_i(t)$) versus t. In Figure 3.8, the plots for t close to zero show one set of curves for $z_i(t)$ starting from their respective trivial solutions. The curves of $z_i(t)$ versus t for all the six sets of equilibrium solutions are shown in Figure 3.9.

A few interesting features of the continuation process are worth noting. For example

- Each trivial solution leads to different equilibrium solution and the integration path is different for each component.

- All the curves are smooth ; one of the four curves may pass through a singularity. (see Figure 3.9 (c) and (d)).

Table I summarizes the computed equilibrium solutions. They are tabulated in the same order as the plots in



Legend : TS = partition $0<t<1$; Δ = integration step;
N = number of equations ; omax = a constant;
$\bar{z}_o$ = trivial solution.

Figure 3.7 Flowchart for Algorithm to find 2*2 Equilibria

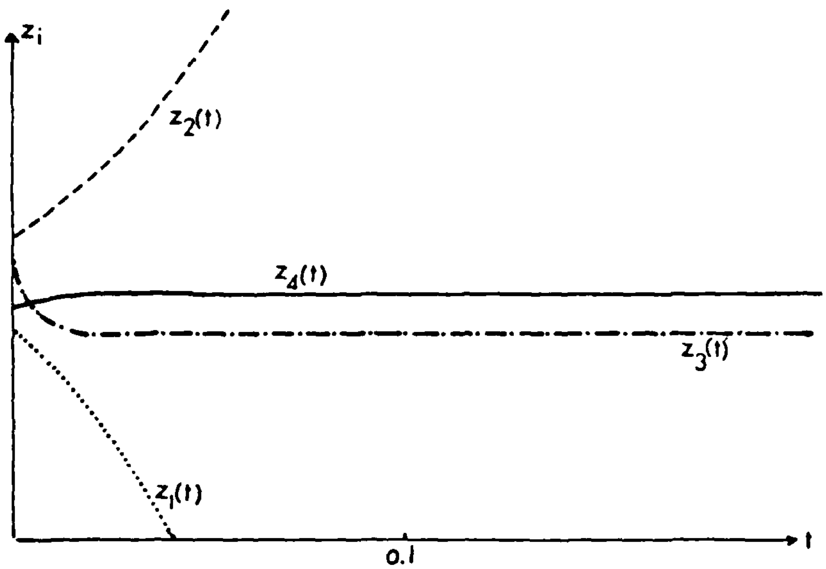


Figure 3.8 $z_i(t)$ Versus t for Values of t Close to Zero.

Figure 3.9. To estimate the accuracies of the results, we defined error as

$$\text{ERROR} = \sqrt{\tilde{F}_1^2 + \tilde{F}_2^2 + \tilde{F}_3^2 + \tilde{F}_4^2}$$

where

$$\tilde{F}_i = F_i(\tilde{z}) \ , \ \tilde{z} \text{ is the } \underline{\text{computed}} \text{ equilibrium solution to } F(\bar{z}) = \bar{0}$$

Decreasing the integration step size in the predictor and corrector routines may reduce the errors by a small amount; but the increase in computational effort may not be justifiable. Conversely, it may be desirable to cut down computing time. Currently, the algorithm performs one corrector step for each predictor step. If two or more

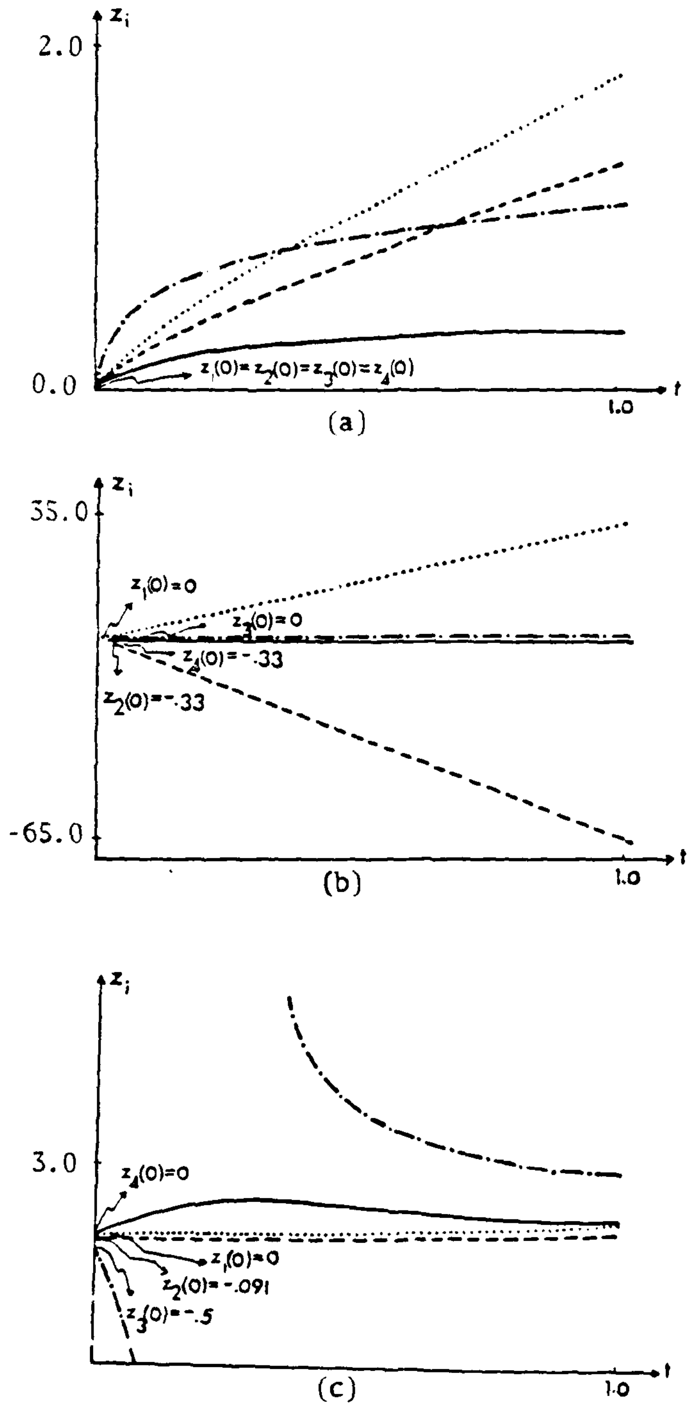


Figure 3.9 Plots of $z_i(t)$ Versus t during Continuation Process

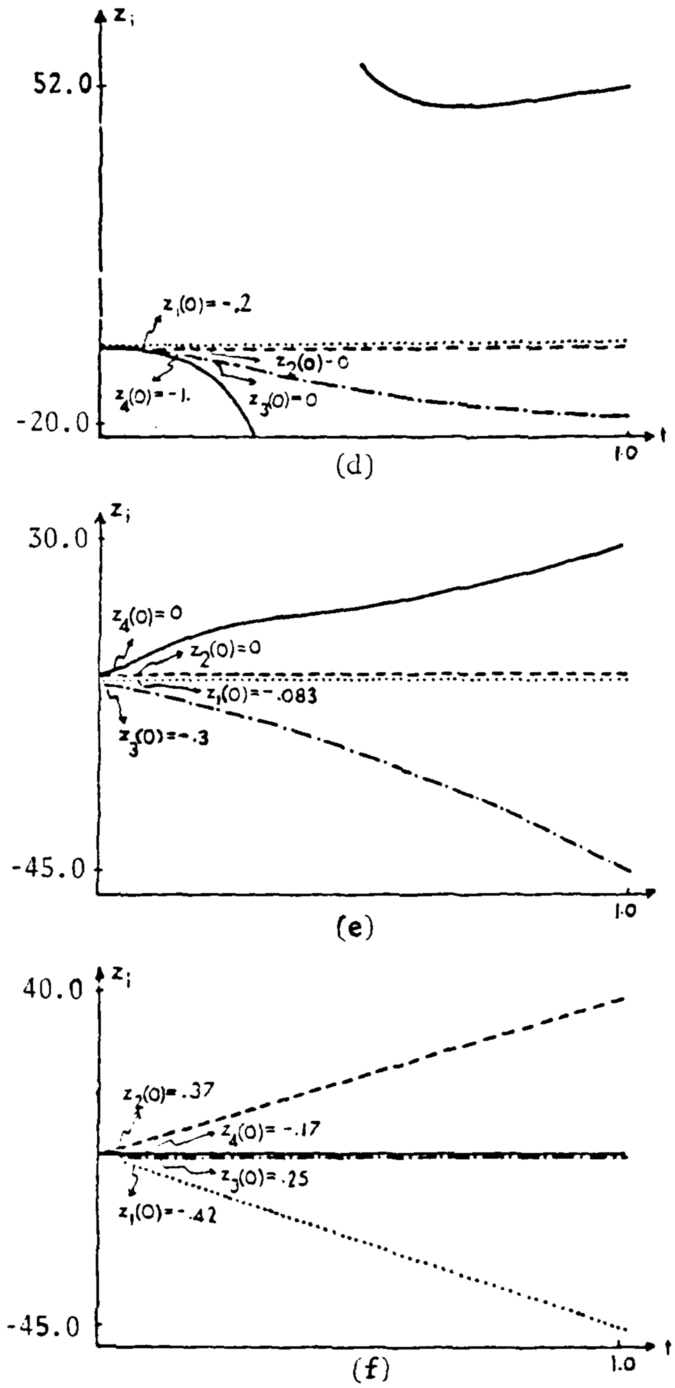


Figure 3.9 (contd.)

TABLE I

Computed Equilibria for X = 1.0, Y = 4.0

Trivial Solution	Computed Equilibrium Solution, $\bar{z}$	Error
(0, 0, 0, 0)	(1.7755, 1.2906, 1.0833, 0.3512	$0.2*10^{-4}$
(0, -0.33, 0, -0.33)	(34.8989, -64.2951, -0.1350, -0.2776)	$0.6*10^{-6}$
(0, -0.091, -0.5, 0)	(0.6154, 0.3846, 3.0769, 0.9231)	$0.1*10^{-5}$
(-0.2, 0, 0, -1.0)	($0.46*10^{-4}$, -0.3030, -16.6400, 52.9820)	$0.22*10^{-5}$
(-0.083, 0, -0.3, 0)	(-0.1857, 0.0325, -44.5297, 28.6147)	$0.56*10^{-6}$
(-0.42, 0.37, 0.25, -0.17)	(-45.2284, 39.1604, -0.3497, -0.04485)	$0.23*10^{-6}$

predictor steps are done for each corrector step, some computational effort[2] can be saved.

---

[2]Saving in computational effort will be more significant when solving higher dimensional systems.

IV. PROPERTIES OF THE 1*1 SYSTEM

The 1*1 problem is the simplest case in our model. It is nevertheless important for us to investigate and understand its properties. Despite its relative simplicity, it is by no means uninteresting. There exists many situations which can be realistically and easily modeled by the 1*1 system. For example, when the opposing forces can be considered as homogeneous, it is convenient to use the 1*1 model for analysis. It is also useful for the analyses at the strategic level when the forces and parameters can be aggregated. In many instances, it seems to provide insight on how to approach the N*N problem, which is much more difficult to visualize. In fact, as the understanding of the 1*1 system increases, there is a strong urge to try to represent the N*N problem by an equivalent 1*1 problem. The equivalent representation is not only attractive in terms of its simplicity but also its economy in computational efforts.

The next section will focus on the relation between system asymptotes and stability of the equilibria. By formulating the problem quantitatively, we are able to arrive at some useful properties. In Section IVB, the system dynamics i.e. the changes in the force levels are analysed by considering the phase trajectories.

A. SYSTEM ASYMPTOTES AND EQUILIBRIUM POINTS

For the 1*1 problem, the system reduces to

$$\begin{aligned} \dot{x} &= -x(u + ay) + r - by \\ \dot{y} &= -y(v + cx) + s - dx \end{aligned} \qquad \text{(eqn 4.1)}$$

An equilibrium condition exists if r and s are chosen such that $\dot{x}$ and $\dot{y}$ are both zero. In general, there will be two equilibrium points corresponding to two locations where the two hyperbolas intersect. The hyperbolas are described by

$$x = \frac{r - by}{u + ay} \qquad \text{(eqn 4.2)}$$

$$y = \frac{s - dx}{v + cx}$$

From equation 4.2, one can easily deduce the four asymptotes (two vertical and two horizontal) associated with the hyperbolas. Figure 4.1 shows a typical set of four asymptotes. They always cross in the third quadrant of the x-y plane and <u>do not</u> depend on the replenishment coefficients. The relative displacements between the two horizontal (and also vertical) asymptotes depend only on the <u>ratios</u> of attrition coefficients and not on the coefficients themselves. It turns out that these properties of the system asymptotes help to simplify the analysis considerably.

1. <u>Stability Criteria</u>

Considering small perturbations about an equilibrium (x_e, y_e) and linearizing the equations, we have

$$\begin{bmatrix} \delta\dot{x} \\ \delta\dot{y} \end{bmatrix} = - \begin{bmatrix} (u + ay_e) & (b + ax_e) \\ (d + cy_e) & (v + cx_e) \end{bmatrix} \begin{bmatrix} \delta x \\ \delta y \end{bmatrix}$$

The characteristic polynomial is simply

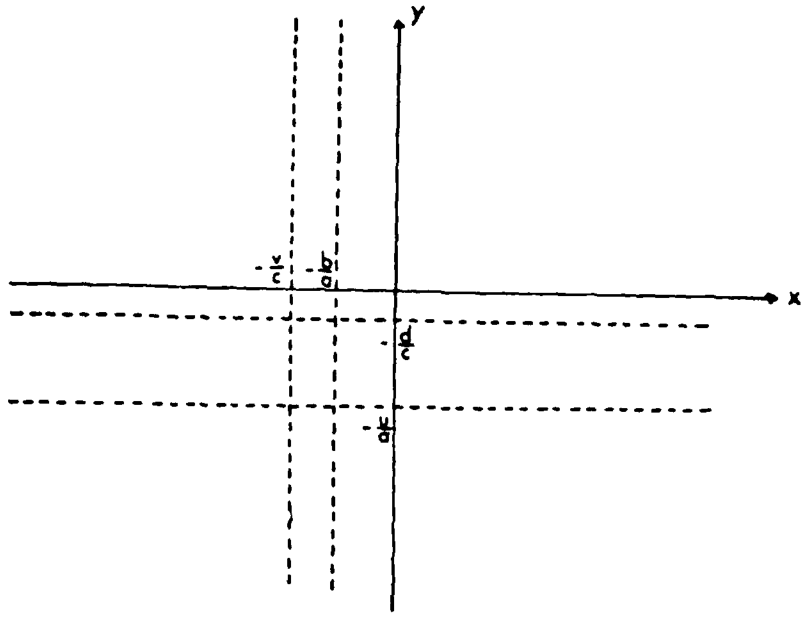


Figure 4.1 System Asymptotes.

$$D(s) = \mathrm{Det}\left[s\tilde{I} - \tilde{C}\right] \qquad \text{(eqn 4.5)}$$

where

I = identity matrix

C = the 2*2 matrix in equation 4.3

Hence,

$$\begin{aligned} D(s) &= \left[s + (u+ay_e)\right]\left[s + (v+cx_e)\right] - \left[(b+ax_e)(d+cy_e)\right] \\ &= s^2 + \left[(u+ay_e) + (v+cx_e)\right]s + \left[(u+ay_e)(v+cx_e) \right. \\ &\quad \left. - (b+ax_e)(d+cy_e)\right] \end{aligned}$$

The conditions for (x_e , y_e) to be a stable equilibrium, i.e. for the roots of $D(s)$ to be in the Left Half Plane (LHP) are given by

$$(u + ay_e) + (v + cx_e) > 0$$
$$(u + ay_e)(v + cx_e) - (b + ax_e)(d + cy_e) > 0 \qquad \text{(eqn 4.4)}$$

2. Stability and Asymptotes

Five different ways in which the hyperbolas can intercept have been identified and their stabilities accounted for. These five cases are shown in Figure 4.2 and each case will be elaborated upon subsequently.

a. Definitions and Formulation

One of the most intriguing facets of the 1*1 problem is the connection between the asymptotes and the stability of the resulting equilibria. We begin the quantitative treatment by first defining the following ratios:

$$\eta_1 \triangleq \frac{u}{a} \quad , \quad \eta_2 \triangleq \frac{d}{c}$$
$$\mu_1 \triangleq \frac{b}{a} \quad , \quad \mu_2 \triangleq \frac{v}{c}$$

The four asymptotes are $x = -\mu_1$, $x = -\mu_2$, $y = -\eta_1$ and $y = -\eta_2$. If we let the first equilibrium point be (x_{e1} , y_{e1}) and substitute the corresponding r and s into the equation 4.2, we have

$$\eta_1(x - x_{e1}) + (xy - x_{e1}y_{e1}) + \mu_1(y - y_{e1}) = 0$$
$$\eta_2(x - x_{e1}) + (xy - x_{e1}y_{e1}) + \mu_2(y - y_{e1}) = 0 \qquad \text{(eqn 4.5)}$$

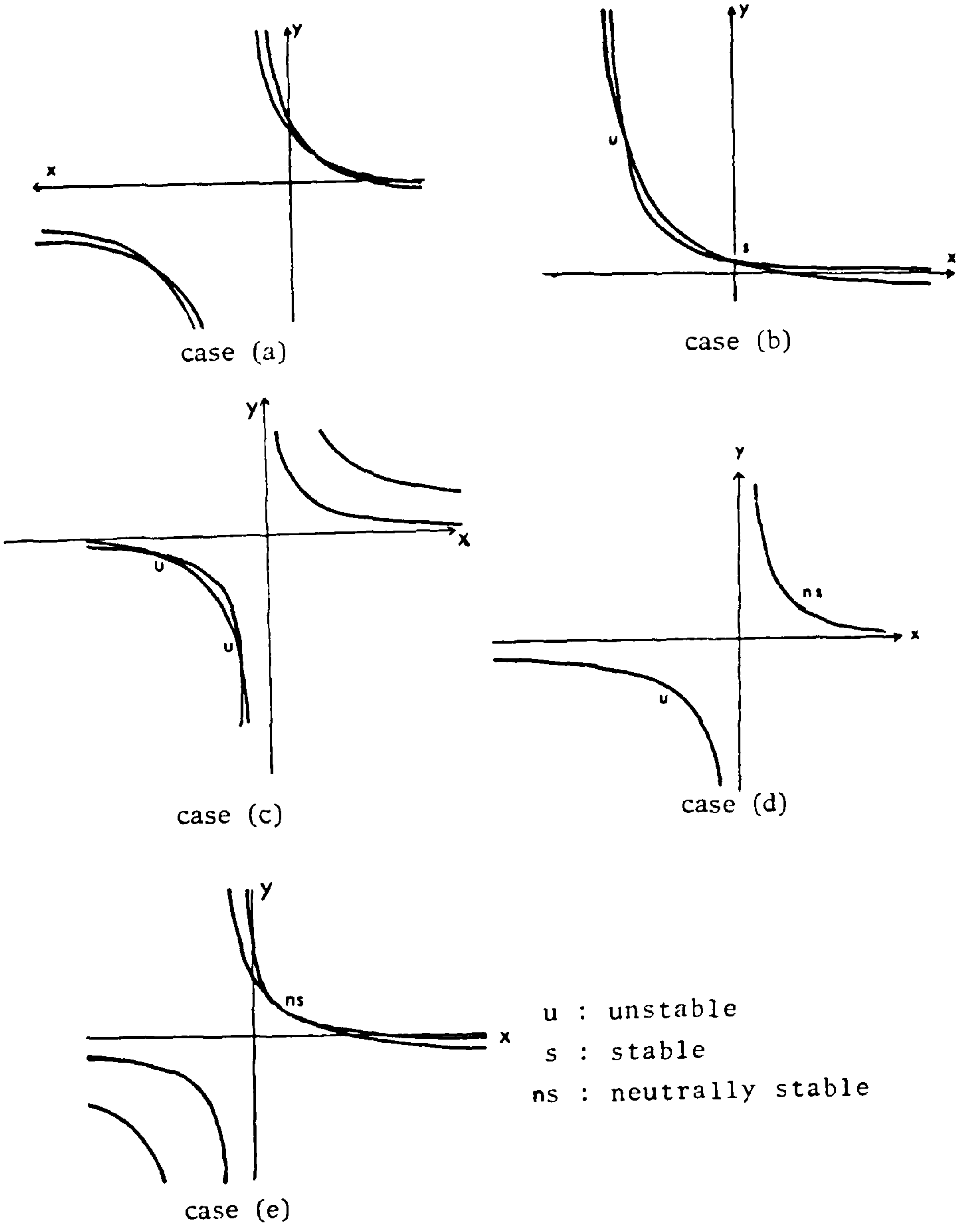


Figure 4.2 Types of Equilibria

Next, the distances between the asymptotes are defined as

$$\varepsilon_x \triangleq \mu_2 - \mu_1$$

$$\varepsilon_y \triangleq \eta_2 - \eta_1$$

It is not difficult to see that ε_x and ε_y will decide where the hyperbolas intersect. For instance, when $\varepsilon_x > 0$ and $\varepsilon_y > 0$, there may be two equilibria in the first quadrant[3] (See case (b) of Figure 4.2). In general, the second equilibrium point (X_{e2}, Y_{e2}) can be found by eliminating y or x from equation 4.5 and comparing coefficients with $(y - y_{e1})(y - y_{e2})$ and $(x - x_{e1})(x - x_{e2})$. The final expressions are

$$x_{e2} = \frac{\varepsilon_x}{\varepsilon_y}(y_{e1} + \eta_1) - \mu_1$$

$$y_{e2} = \frac{\varepsilon_y}{\varepsilon_x}(x_{e1} + \mu_1) - \eta_1 \qquad \text{(eqn 4.6)}$$

For constant x_{e1}, x_{e2}, y_{e1}, and y_{e2}, equation 4.6 can be written to represent two straight lines in ε_x, ε_y plane. The equations of these two lines are

$$\varepsilon_y = \varepsilon_x \frac{(y_{e2} + \eta_1)}{(x_{e1} + \mu_1)}$$

$$\varepsilon_y = \varepsilon_x \frac{(y_{e1} + \eta_1)}{(x_{e2} + \mu_1)} \qquad \text{(eqn 4.7)}$$

---

[3] In our context, the quadrants are defined by the asymptotes and not by the x, y axes.

b. Types of Equilibria and their Stability

To derive the different types of equilibria and their associated stabilities, we make a transition from the x, y plane into the ε_x, ε_y plane. Briefly, the basic approach is to fix one equilibrium point (x_{e1}, y_{e1}) on the first quadrant hyperbolas and consider the regions in the ε_x, ε_y plane when we have the other point in various places of the x, y plane. The other essential step is to express the stability criteria (equation 4.4) in terms of ε_x, ε_y, μ_1, η_1, x_{e1}, y_{e1}. A summary of the results which are derived in Appendix D is given below :

(1) When both equilibrium points are on the first quadrant hyperbolas (case (b) in Figure 4.2), one will be stable and the other unstable;

(2) When one equilibrium point is on the first quadrant and the other on the third, both can be unstable or one will be stable and the other unstable (case (a) in Figure 4.2);

(3) When both equilibrium points are on the third quadrant hyperbolas, both are unstable (case (c) in Figure 4.2);

(4) When there are infinite number of equilibria as in case (d) in Figure 4.2, $\varepsilon_x = \varepsilon_y = 0$ and the two sets of hyperbolas merge. Equilibria lying on the first quadrant hyperbola are neutrally stable (one eigenvalue equals zero) and those on the other hyperbola are unstable;

(5) When there are repeated equilibria as in case (e) in Figure 4.2, they are neutrally stable if the hyperbolas touch in the first quadrant ; otherwise they are unstable.

Most of the above results are embedded within Figure 4.3 which is reproduced from Appendix D for convenience. Evidently, both the coordinates of the equilibrium points (x_e, y_e) and the location in the ε_x, ε_y plane determine the stabilities. The ε_x, ε_y plane has been subdivided into a few regions each with distinct stability characteristics.

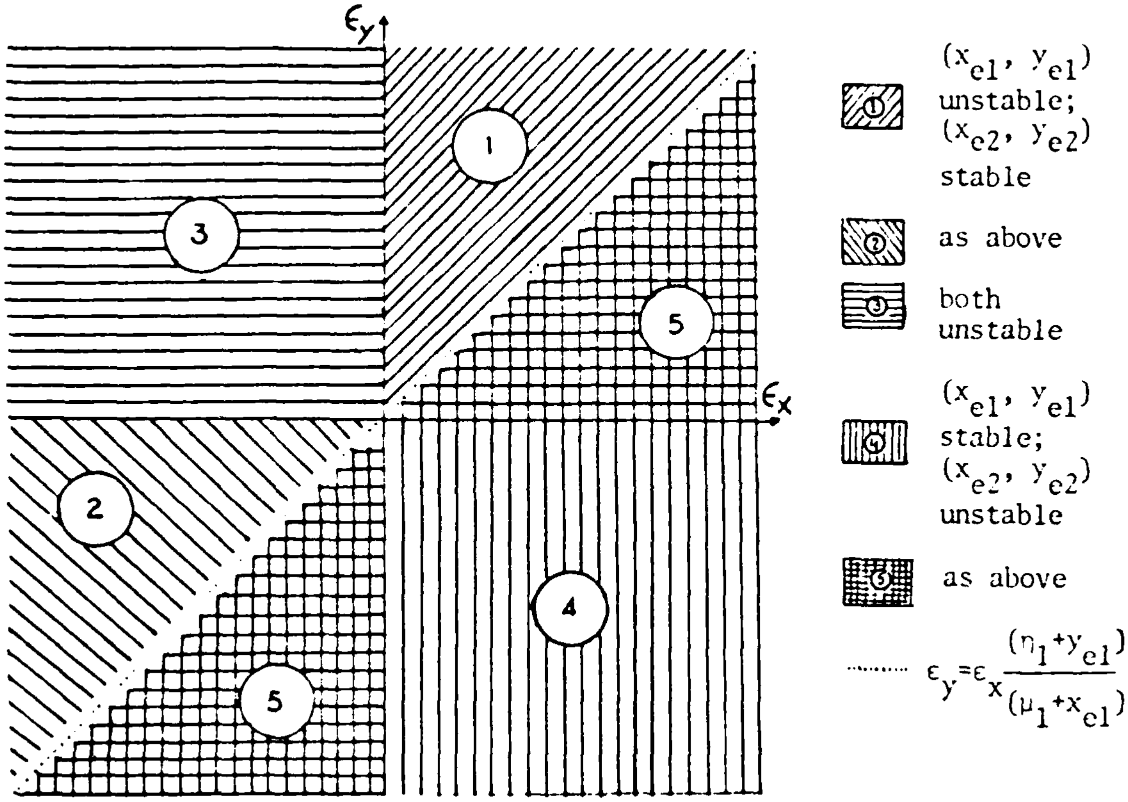


Figure 4.3 The ε_x, ε_y Plane.

The case of infinitely many equilibria corresponds to the origin of ε_x, ε_y plane ($\mu_1 = \mu_2$, $\eta_1 = \eta_2$). The only way for two sets of hyperbolas to merge is for their respective asymptotes to merge. This case is a degenerate instance of repeated equilibria (case (e) in Figure 4.2), which is shown in Appendix D to correspond to operating points on the line $\varepsilon_y = \varepsilon_x (Y + \eta_1)/(X + \mu_1)$ as illustrated in Figure 4.3.

As a corollary, we note that there cannot be two stable equilibrium points in the 1*1 problem. This deduction can be made by referring to Figure 4.3. There is no region in the ε_x, ε_y plane which allows for this case. At most, there can be two <u>neutrally</u> stable equilibria which are repeated. Numerous attempts have been made to obtain two stable equilibria in the 2*2 problem, but in vain. Whether it is also true for 2*2 or higher dimensional problems that only one equilibrium may be stable is still a matter of conjecture.

In Appendix E, the relations between the regions on the ε_x, ε_y plane and their associated stabilities are verified. Some representative points on the ε_x, ε_y plane are chosen and their stabilities checked.

B. SYSTEM DYNAMICS

The dynamics of a 1*1 system are characterised by its phase trajectories, which are curves on the x-y plane describing the history of the system as the time, t, changes. These trajectories can be conveniently obtained by integrating equation 4.1 numerically.

Needless to say, being able to predict the trajectories is important, for it means that we know how our model of a battle progresses. Once the factors influencing the course of a battle are known, appropriate command decisions can be introduced to ensure favorable battle outcome. In Chapter V, we will see how many of the results obtained in this section can be used to rationalize and predict battle outcome.

Some typical trajectories corresponding to the different types of equilibria are described in the next subsection. Besides the stability which influences trajectories, it was briefly mentioned in Chapter III that domains of attraction also affect the trajectories. In the subsection that follows, we will show specific examples of the way to determine the domains by finding their exact boundaries.

1. Trajectories

Two methods of establishing the trajectories from a given initial condition will be described. The brute-force method which has been mentioned uses numerical integration. The other method which often provides better insight, is more graphical. The graphical method is based on a few very simple rules to predict the gross behavior of a trajectory. Some of these rules are listed below :

(1) A stable point "attracts"; unstable point "repels";

(2) Points on either side of a boundary move into their respective domains;

(3) For large (x, y), trajectories are governed by the Lanchester "linear law";

(4) Points near the hyperbolas can be easily analyzed by noting the signs of $\dot{x}$ and $\dot{y}$.

As an example of using the graphical method to determine trajectories, consider a region around an unstable equilibrium point on the first quadrant hyperbola. The whole picture of the phase trajectories (sometimes called phase-plane portrait [Ref. 9]) can be put together in a logical fashion by using those simple rules. Since this equilibrium point is unstable, trajectories will be expected to diverge from it. As an unstable equilibrium point, it will have a boundary line passing through it. Initial conditions start from each side give rise to different trajectories. Next, we determine the signs of $\dot{x}$, $\dot{y}$ on both sides of each hyperbola as indicated in Figure 4.4 where only one intersection is shown.

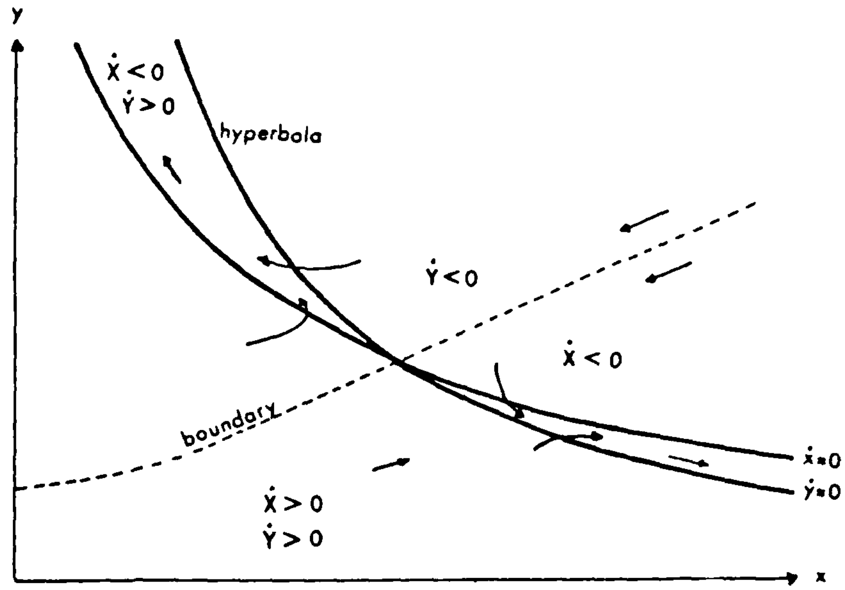


Figure 4.4 Analytical method of predicting trajectories.

Note how predictable these trajectories are. If, for some reasons, the exact trajectories are required, we can resort to the brute-force method. The methods are obviously complementary in nature. The advantages of the brute-force method are accuracy and simplicity. In Figure 4.5, a typical computer plot consisting of ten trajectories is shown. The program which produces the plot is included in Appendix F.

Referring to Figure 4.5, the trajectories cross the hyperbolas and move asymptotically along a common curve

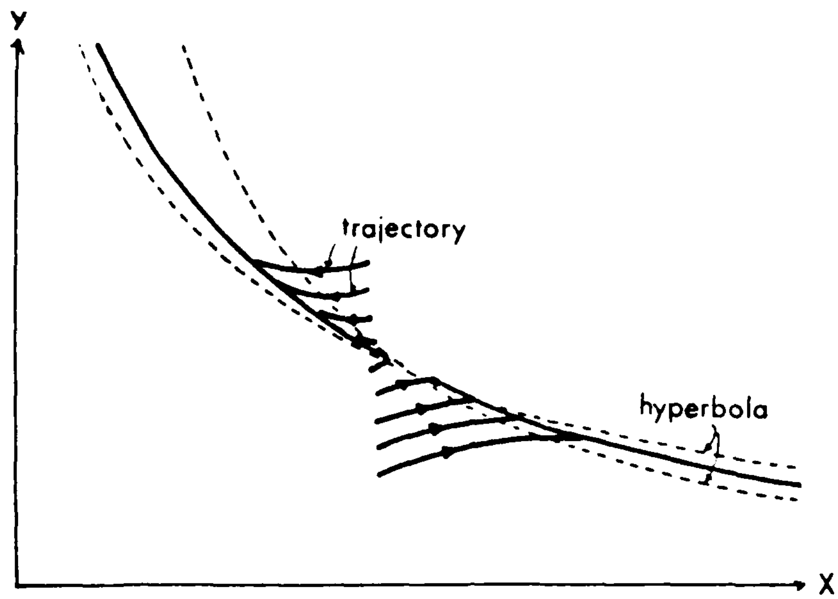


Figure 4.5 Computer Plot of Trajectories.

lying between the hyperbolas. This same property is exhibited by other cases. Even the special case with no hyperbolic intersection has been found to behave similarly as can seen in Figure 4.6.

Our ability to determine the trajectories and present them vividly is partly due to fact that two-dimensional pictures can be easily drawn and visualized. For dimensions higher than the third, it is impossible to visualize trajectories; however, the notion of trajectories can be conceptually extended to n-dimensional space. Thus, it seems likely that in the higher dimensional systems, trajectories cross hypersurfaces and move along a common asymptotic curve analogous to that in the 1*1 system. Further studies are required before this behavior can be confirmed.

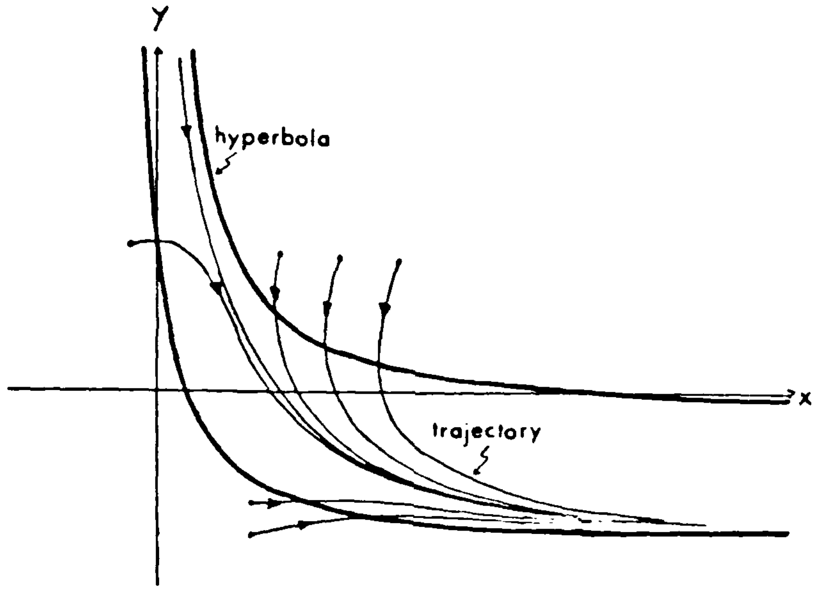


Figure 4.6 Trajectories when Hyperbolas do not Intersect.

2. Boundaries of Domains of Attraction

In Chapter III, the idea of the domains of attraction was briefly discussed. In an n-dimensional space, such a domain is a region or volume in which all initial points come under similiar influence. When domains exist, there will be boundary surfaces which can be thought of as collections of invariant curves passing through unstable equilibria.

For a 1*1 problem, domains and boundaries are not at all abstract. In the last subsection, they have been shown to affect trajectories. Recall that in Chapter III, we mentioned a simple and yet effective way of finding the boundary curves and establishing the domains in the x-y plane. Examples on the use of backward integration to obtain boundary curves are now presented.

a. Boundary Curve through an Unstable Point

Starting from an unstable point, we apply small perturbations in both directions perpendicular to an eigenvector associated with the most positive eigenvalue and integrate backward in time (in the computer program, this is easily done by employing negative time steps for integration). The result is a smooth, invariant curve which is **exactly** the boundary or the so-called separatrix like the one shown in Figure 4.7.

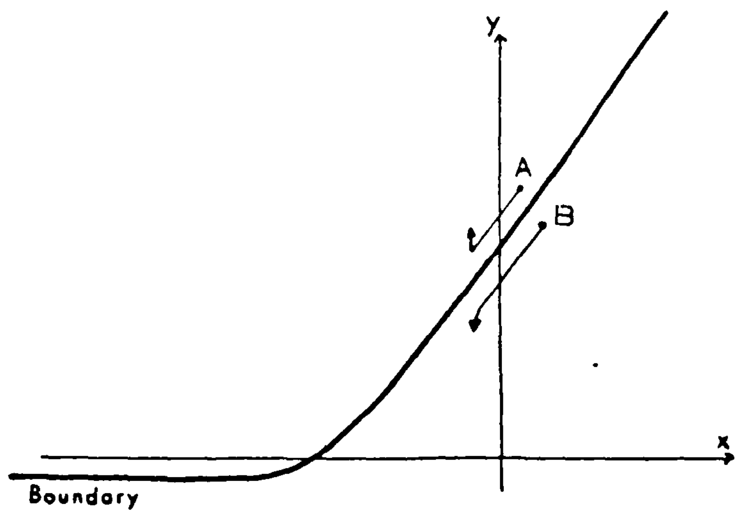


Figure 4.7 Boundary Curve through an Unstable Point.

To verify that the curve is indeed the boundary, two initial points are chosen just off the curve (e.g A, B in Figure 4.7). If we forward integrate from these two points, they move into different domains as indicated in the same diagram. Appendix G contains a Fortran program that does the backward integration and plots the boundary curve.

3. Boundary Curve between Two Hyperbolas

Boundary curves do not necessarily pass through unstable points. Backward integration methods can also be used if a boundary exists but there is no unstable equilibrium point to serve as the starting point of integration. This is best illustrated by considering the case of both equilibria on the first quadrant hyperbolas. In this case, there is no equilibrium point in the third quadrant; nevertheless a boundary does exist between the third-quadrant hyperbolas. The existence of the boundary is visible by simply considering the signs of $\dot{x}$ and $\dot{y}$ on both sides of the hyperbolas. In figure 4.8, the signs of $\dot{x}$ and $\dot{y}$ and also the directions of some typical trajectories are depicted.

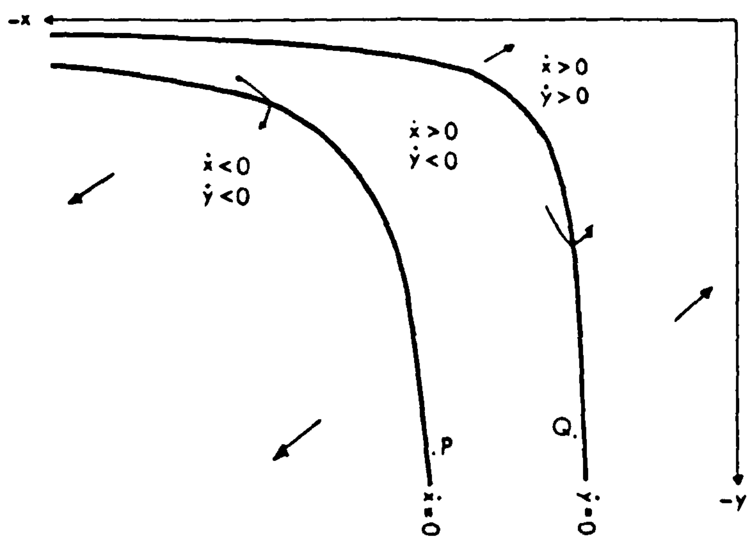


Figure 4.8 Existence of Boundary Between Two Hyperbolas.

To obtain the exact boundary, choose a point close to a hyperbola and on lower part of the hyperbolas (e.g. point P or Q in Figure 4.8) and integrate backward. The result is a boundary curve as shown in Figure 4.9.

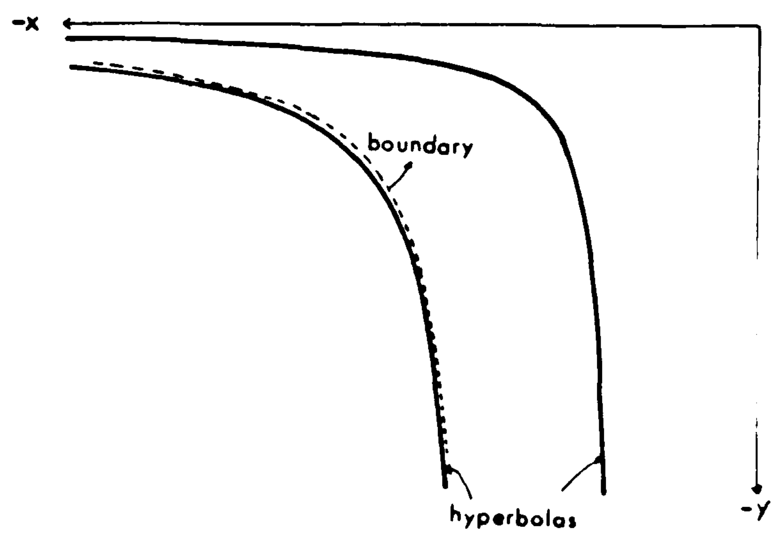


Figure 4.9 Exact Boundary Curve Between Two Hyperbolas.

4. Summary of the 1*1 Problem

We have seen the close relation between system asymptotes and stabilities. Through the use of newly defined variables ε_x and ε_y, the stability of different types of equilibria has been derived. Five cases have been identified, and they correspond to the types of intersections on the x-y plane. For example, if both the equilibria are found on the third quadrant hyperbolas, then we know that they will be unstable.

Two methods of establishing the trajectories have been described in this chapter. These two methods complement each other and the choice depends on our requirements. The dynamics of the system are characterized by the trajectories, which as we have seen are very predictable. These trajectories are influenced by the stabilities of equilibria and domains of attraction which are separated by boundary curves. A simple way of plotting the boundary curves has also been presented along with specific examples.

The results derived in this chapter will be applied in the next chapter. The knowledge of the system dynamics and how they are affected by stability and other parameters will enable us to analyze changes in force levels as the battle progresses.

V. STRATEGY FOR INITIAL FORCE COMMITMENT

In the last two chapters, emphasis has been placed on establishing the mathematical framework of the system dynamics and stability. In this chapter, we examine some model operational problems that are related to stability and dynamic considerations.

One of the major command decisions that has to be made during a build-up period of a war pertains to initial force commitment. A good strategy calls for a balance between initial deployment and reserves. In practice, a multitude of factors have to be considered before deciding on a particular commitment. The approach in this chapter provides us with a set of mixed strategies but does not consider intangiable factors like world politics, national economy, survival factor and so on.

Stability has been shown to effect trajectories which in turn effect battle outcome. Recall from Chapter IV that there are some trajectories which represent speedy and complete annihilation of one force; hence it seems reasonable that the side that is tipped to win the battle will want to operate on an unstable trajectory. But to what extent can one exploit the stability behavior of the system to influence battle outcome? Obviously there will be practical limitations; an important one of these is total available resources.

A. PROBLEM STATEMENT AND APPROACH

The problem statement is as follows :

> Given total defense resources Q_x , Q_y for x and y respectively, what is the optimum set of strategies for initial force commitment, X and Y?

We begin by treating this as a 1*1 problem at the strategic level. The dynamics of the problem are thus governed by equation 4.1. Both sides are assumed to operate initially at equilibrium with constant replenishment rates given by

$$r = X(u + aY) + bY \qquad \text{(eqn 5.1)}$$
$$s = Y(v + cX) + dX$$

Since both sides have limited defense resources Q_x, Q_y, the replenishment rates versus time may be as shown in Figure 5.1, where $Q_x = rT_x$ and $Q_y = sT_y$.

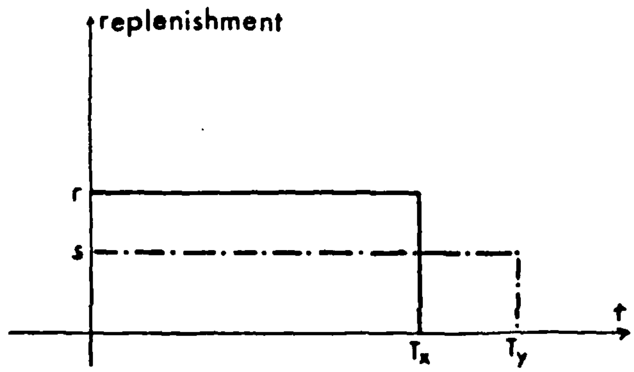


Figure 5.1 Replenishment Versus Time.

The next step is to select some suitable form of payoff function which is to be optimized for a certain choice of X and Y. The payoff function (from X to Y) has been chosen to be

$$A(X,Y) = L_y - L_x$$

where L_x, L_y = Total losses for x, y at battle termination

As each side runs out of resources at different times, the simulation is conducted in stages. The total losses are determined by simulating the dynamics of the system until one of the force levels drops to ten percent of its total resources, Q.

If X and Y are assumed to be chosen from a finite set of values, then for each pair (X,Y), one A(X,Y) can be obtained. A payoff matrix can be formed and the problem can be treated as a two-person game. Based on the minimax theorem, there exists a set of optimal mixed-strategies and one convenient way of finding them is through the use of Linear Programming.

It is perhaps worth-noting that the approach is computation- oriented. It has been made feasible by the availability of high-speed computers and efficient software for numerical computations.

B. MULTISTAGE BATTLE

Using the above approach, the entire battle can be divided into three stages, namely

(1) Both r and s are nonzero

(2) One of the r or s equals zero

(3) Both r and s are zero

1. Stage 1

This stage will be the period from outbreak of war to the time (T_1) when one side runs out of resources. It is also possible that $x < 0.1Q_x$ or $y < 0.1Q_y$ before T_1 is reached, in which case the battle is over. In general, this period T_1 can be written mathematically as

$$T_1 = \text{Min}\ \{T_x, T_y\}$$

During this stage, the dynamics of the system is given by the familiar 1*1 system

$$\dot{x} = -x(u + ay) - by + r$$
$$\dot{y} = -y(v + cx) - dx + s \qquad \text{(eqn 5.2)}$$

When this 1*1 system is integrated, just as in Chapter IV, the resulting trajectories behave similiarly. However, there is a major difference. Now, we no longer have unlimited defense resources, and this stage will not last forever. It implies that, unless Q_x or Q_y is extremely large*, trajectories which reflect quick annihilation of one of the forces are rare. In general, T_x and T_y are given by

$$T_x = \frac{Q_x - X}{r}$$
$$T_y = \frac{Q_y - Y}{s}$$

If one of the force levels drops to less than ten percent of Q_x or Q_y, the battle is arbitrarily considered over and the losses are calculated as in Figure 5.2. The finish time (FINTIM) is simply t, the time when $x < 0.1Q_x$ or $y < 0.1Q_y$.

2. Stage 2

Since either x or y can run out of reserves first, the dynamics of stage 2 are governed by either equation 5.3 or 5.4 respectively.

---

*Q_x or Q_y may be very large if x or y is backed by a superpower who is fully committed to provide military aid.

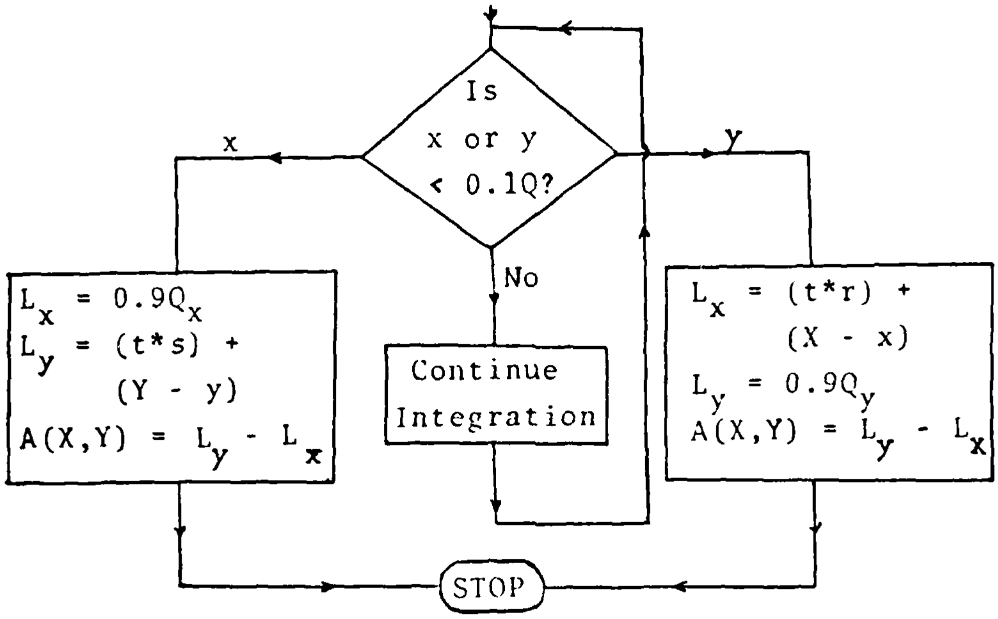


Figure 5.2 Losses at Stage 1.

$$\begin{aligned} \dot{x} &= -x(u + ay) - by \\ \dot{y} &= -y(v + cx) - dx + s \end{aligned} \qquad \text{(eqn 5.3)}$$

$$\begin{aligned} \dot{x} &= -x(u + ay) - by + r \\ \dot{y} &= -y(v + cx) - dx \end{aligned} \qquad \text{(eqn 5.4)}$$

Unless the battle ends earlier, this period will last for T_2 which is given by

$$T_2 = \mathrm{Max}\,\{T_x, T_y\} - T_1$$

During this period, the trajectory will be different from that in stage 1. This is because when $r = 0$ or $s = 0$, one of the hyperbolas is shifted so as to cross the origin and we have different equilibrium points. The trajectory will now be influenced by the new equilibrium point. This is illustrated in Figure 5.3.

Calculations of the losses are more complex than in stage 1 since there are now two cases to deal with i.e. $r = 0$ or $s = 0$. The procedure is shown in Figure 5.4.

3. Stage 3

If the battle enters stage 3 without either $x < 0.1Q_x$ or $y < 0.1Q_y$ then the dynamics will be dominated by attritions since $r = s = 0$. Equation 5.5 is now used for integration.

$$\dot{x} = -x(u + ay) - by \qquad \text{(eqn 5.5)}$$
$$\dot{y} = -y(v + cx) - dx$$

Again, the trajectory will have to change because now both hyperbolas pass through the origin. This is illustrated in Figure 5.5 where we show how the intersection at stage 2 has changed. Losses and FINTIM are calculated in accordance with the procedure in Figure 5.6.

C. MIXED STRATEGIES

The range 0 to Q[5] for both X and Y can be subdivided into m force levels. There are m*m pairs of X and Y and corresponding number of payoffs, A(X,Y). We thus have an m*m payoff matrix having elements A(X,Y). Figure 5.7 gives a pictorial representation of this two-person game.

---

[5] In the actual program, one may wish to restrict the range of X and Y to interval (0.2Q - 0.75Q) to reflect practical limitations in initial force deployment.

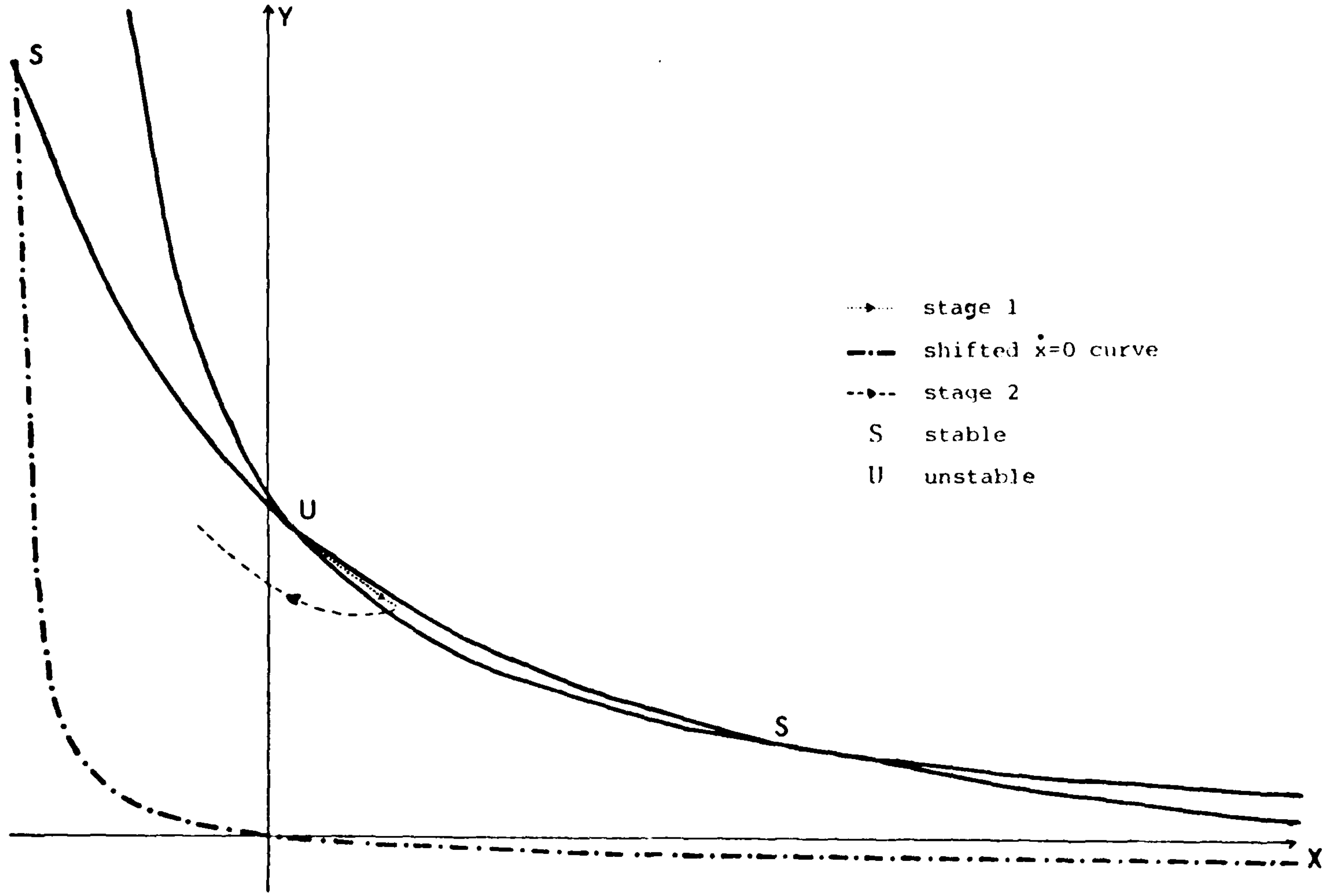


Figure 5.3 Trajectories and Hyperbolic Intersection During Stage 2.

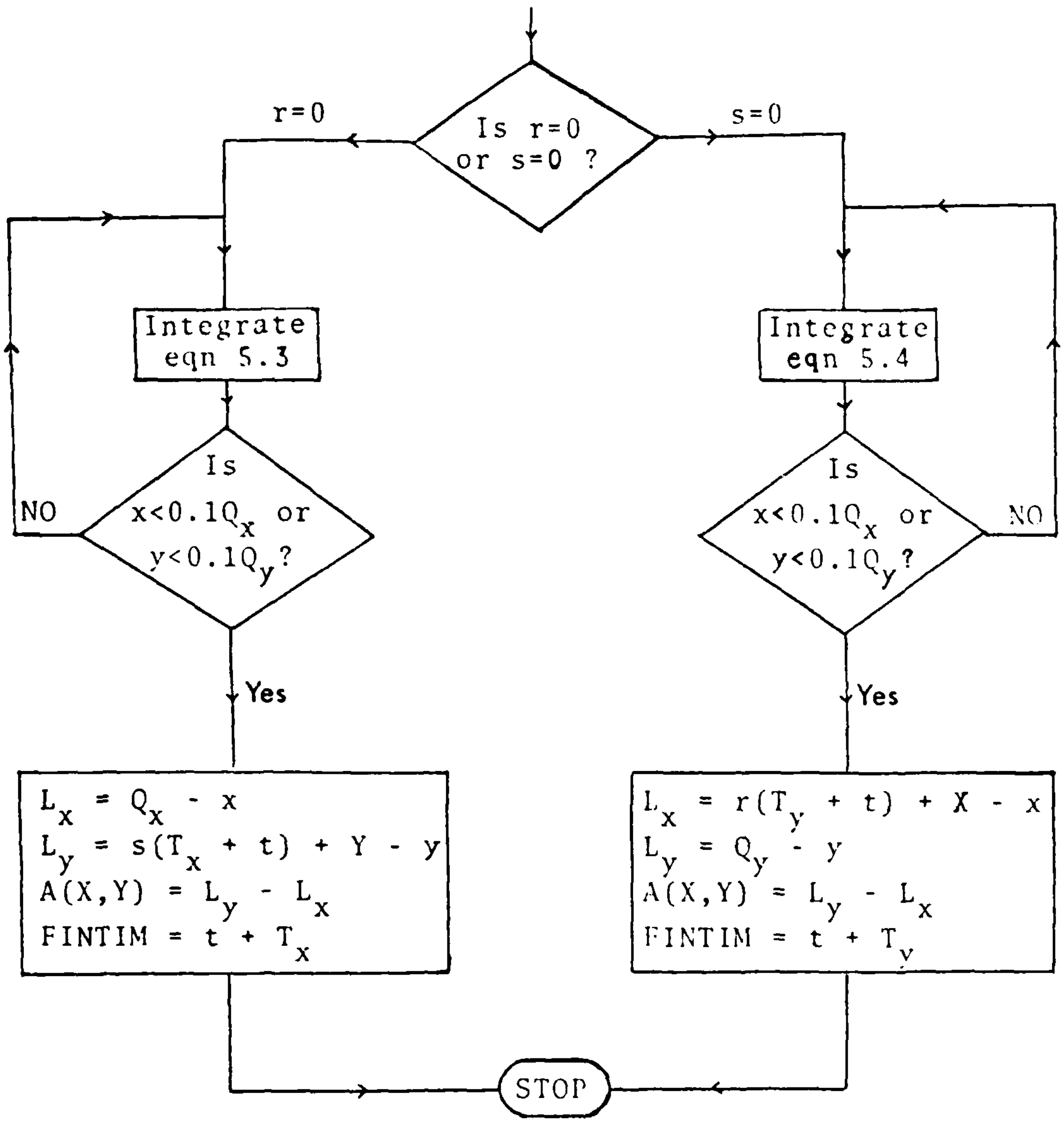


Figure 5.4 Losses at Stage 2.

In the last section, the procedure for computing $A(X,Y)$ has been described. A simple program can be written to compute each element of the payoff matrix. One such program is given in Appendix H.

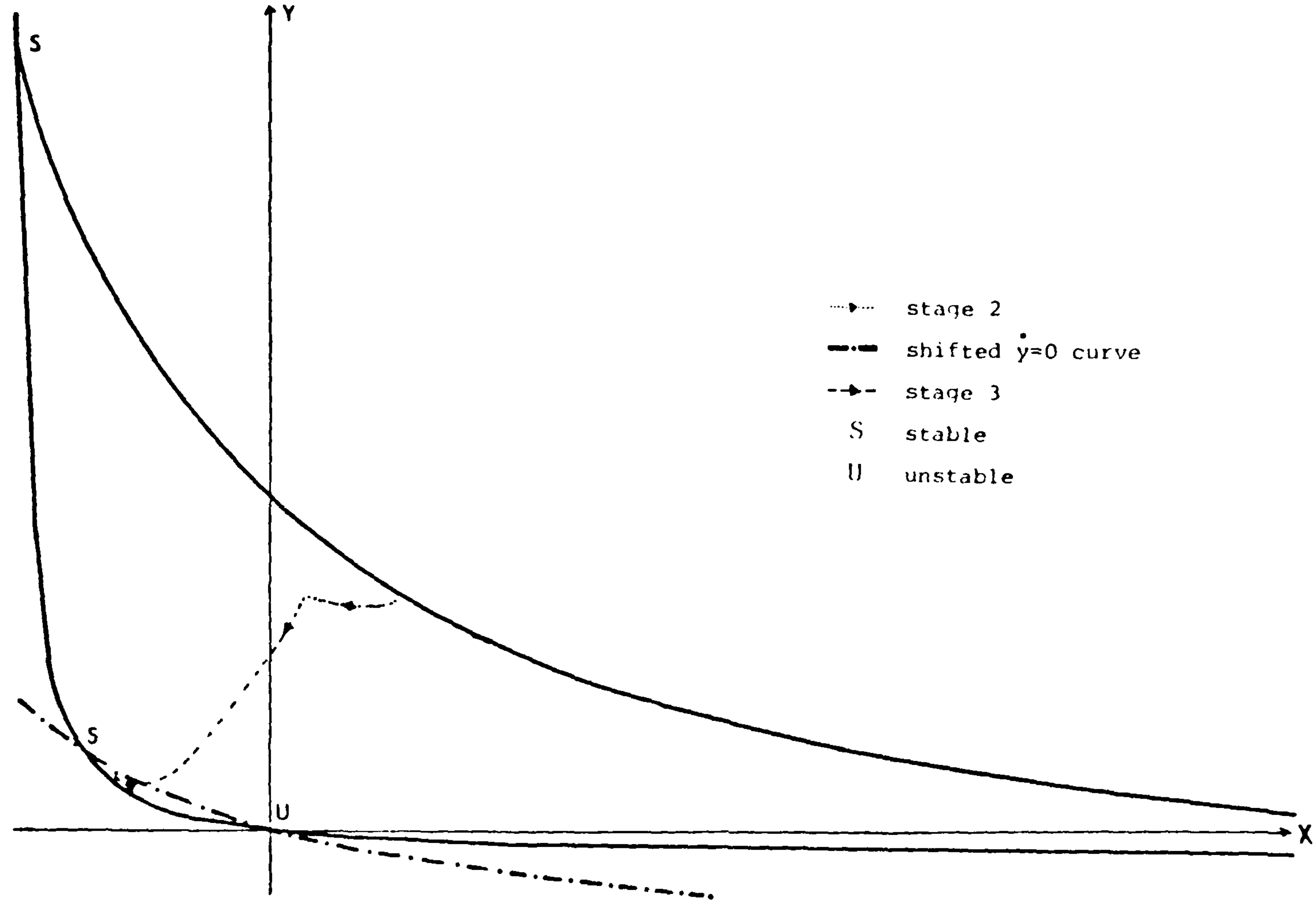


Figure 5.5 Trajectories and Hyperbolic Intersection During Stage 3.

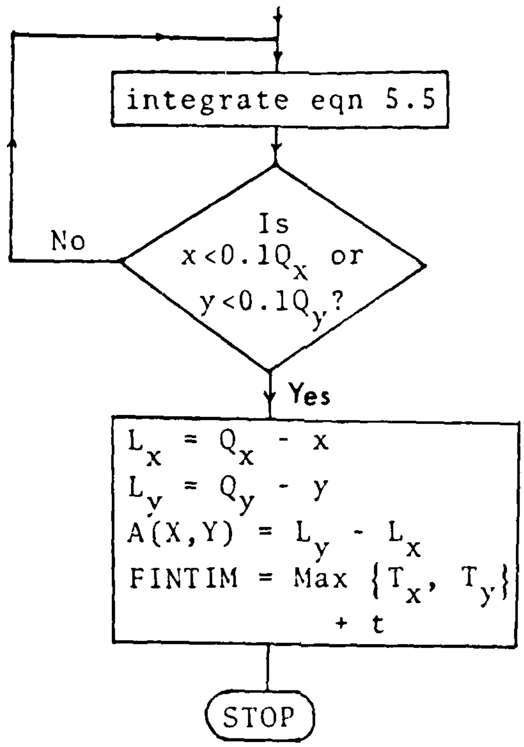


Figure 5.6 Losses at Stage 3.

There are a few ways of presenting and interpreting the payoff matrix. A normal practice is to present it in tabular form and consider only pure strategy. Alternatively, a plot of A (X, Y) as a function of X and Y could be obtained. When using pure strategies, it has been observed that the game does not always have a saddle point [Ref. 10] and it would be better to use mixed strategies.

In mixed strategies, x and y may play all their strategies in accordance with a certain set of probabilities. Although in our situation, x and y can only play once, the same concept of mixed strategies is still useful. If we let p_i and q_j be the probabilities by which x and y select their ith and jth pure strategies respectively, then

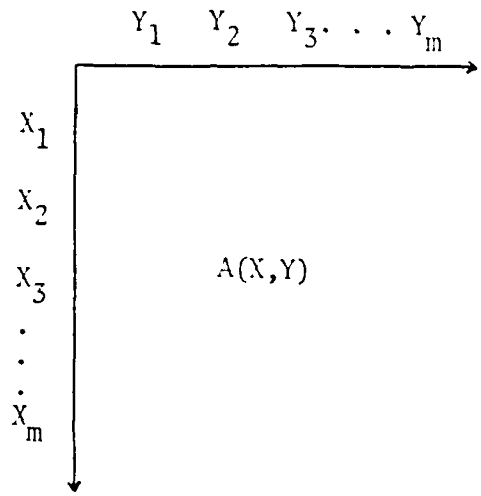


Figure 5.7 Payoff Matrix.

$$\sum_i p_i = \sum_j q_j = 1 \qquad\qquad p_i > 0,\ q_j > 0$$

In addition the (i,j)th entry of the payoff matrix be denoted by a_{ij}, the probabilities can be represented by the matrix below

Y

		p_1	p_2	...	p_m
	q_1	a_{11}	a_{12}	.	a_{1m}
	.	.	.	.	.
X	.	.	.	.	.
	.	.	.	.	.
	q_m	a_{m1}	a_{m2}	...	a_{mm}

The optimal mixed strategy is based on the minimax criterion. Mathematically, x and y select p_i and q_j which will yield U and V as given equation 5.6 and equation 5.7 respectively.

$$u = \max_{p_i} \left\{ \min \left[\sum_{i=1}^{m} a_{i1} p_i \, , \, \sum_{i=1}^{m} a_{i2} p_i , \ldots , \sum_{i=1}^{m} a_{im} p_i \right] \right\} \qquad \text{(eqn 5.6)}$$

$$v = \min_{q_j} \left\{ \max \left[\sum_{j=1}^{m} a_{1j} q_j \, , \, \sum_{j=1}^{m} a_{2j} q_j , \ldots , \sum_{j=1}^{m} a_{mj} q_j \right] \right\} \qquad \text{(eqn 5.7)}$$

Appendix I describes how the problem of solving for the optimal values of p_i and q_j can be put into linear programming form. The program given in Appendix H also computes this optimum set of solution in addition to obtaining the payoff matrix.

The concept of mixed strategies is quite intuitive if a game is to be played repeatedly. But since we are using it to provide us with an optimum set of probabilities of selecting the pure strategies, some interpretation is required. Although the optimum mixed strategies have been obtained, a pure strategy still has to be selected and used. However, it is important that the selection process should be random[6] according to the optimized probabilities obtained.

One simple but valid statistical procedure [Ref. 11] to select a pure strategy from a set of mixed strategies is to first plot the probability distribution function. A random number generator is then used to generate a number between zero and one. The corresponding value of the strategy could then be selected. This procedure is shown in Figure 5.8.

---

[6]The selection process must be random otherwise the opponent can select a strategy to improve his outcome.

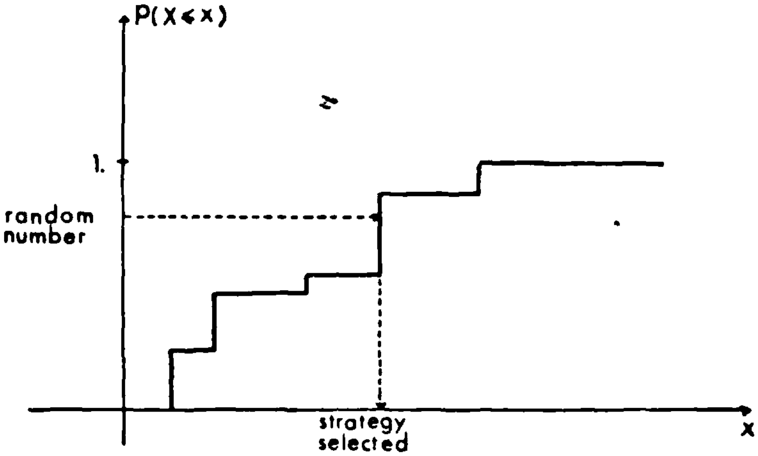


Figure 5.8 Obtaining Pure Strategy from Mixed Strategies.

D. EXAMPLE USING KOREAN WAR DATA

One of the main objectives of using actual historical data in a model is for validation. It is important that the results obtained using the model should at least be consistent with actual events. The Korean War has been chosen because there was a clear-cut victor during the initial phase of the war. We consider the period when only North Korea and Republic of Korea (South Korea) were involved.

Before the entire simulation can be carried out, the actual force strategies, fighting ability, weapon state, etc, have to be transformed into familiar quantities and parameters such as Q_x, Q_y, X, Y, a, b, c, d,..., and so on. This transformation, together with some background data on the Korean War are given in Appendix J.

1. Results and Discussions

First we examine the resultant trajectories during the three stages of the battle which are shown in Figure 5.9. The simulation uses the X and Y which correspond to the actual initial deployment by both North and South Korea respectively. Clearly we see that the victor is x, as it was in history. The result of the simulation also shows the three stages explained in the last section. Note that the trajectories for the first and second stages are curtailed because both sides run out of war reserves. The implication is that in practice, the kind of trajectories leading to large and rapid changes in force levels are rather rare.

However, the effect of instability on battle outcome is borne out by experimenting with the directions of perturbations. Consider the case in which x (North Korea) fixes the initial force and y (South Korea) varying the initial force levels around the equilibrium point. In Figure 5.9, these perturbed points are denoted by points A to D spanning across the boundary separating the domains of attraction. From our understanding of the stability and system dynamics each perturbation will give rise to different trajectory and payoff at the end of the simulation. Clearly, y will want to operate at the perturbed points A or B rather C or D since the former will result in the trajectory for stage one to be in a decreasing x direction. Table II shows the variation in the payoff as the perturbation point changes. When the perturbed points are at A or B, the payoffs to x are less then those for points C or D. Thus we have seen how an unstable system can be used to inflict heavier losses on the opponent. The more unstable a system gets, the more significant will be the effects of initial perturbation which are manifested by initial victory and element of surprise. Since some systems with large aimed-fire coefficients tend to be highly unstable, we can expect this effect to be most pronounced in battles involving high-technology and highly-lethal weapons.

The payoff matrix and optimal probabilities p_i^* and q_j^* are shown in Table III. The results suggest that the

TABLE II

Effect of Different Perturbations on the Payoff

Location in Figure 5.9	Co-ordinates of Perturbed Point	Payoff to X
A	(6.7, 3.05)	2.39
B	(6.7, 3.025)	2.41
C	(6.7, 2.975)	2.45
D	(6.7, 2.95)	2.47

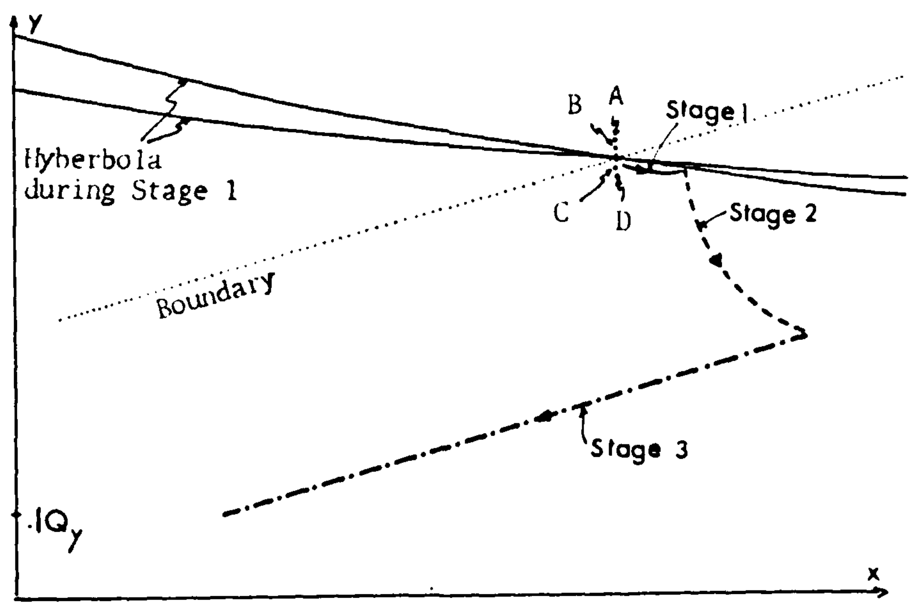


Figure 5.9   Trajectory for Korean War.

North Koreans should use large initial deployment. In the actual war, North Korea actually deployed almost all of its regular force and within a few days captured Seoul, the capital of South Korea. The payoff matrix also shows that no matter which strategy is chosen by South Korea, it is bound to suffer much more losses than North Korea. Again, this is in agreement with history since it is an accepted fact that without US intervention, there would be no South Korea today.

So far in the example, we have always considered the situation in which the equilibrium point (X,Y) determines the replenishment rates as given in equation 5.1. It is interesting to investigate the effect on the payoff when the initial operating point is at some other location other than (X,Y). Let the new initial point be at (X_1,Y_1) and consider a case where X_1 is kept equal to X and only Y_1 is varied. (X,Y) has been chosen to be (6.7,3.0). In Figure 5.10, three trajectories corresponding to Y_1 at 2.5, 4.0, 5.0 are shown together with the hyperbolic intersection during stage one. Basically, the trajectories correspond to the three stages of simulation as before and x is still the victor. However, both the payoff (Ly-Lx) and finish time are slightly different from operating at (X,Y). Table IV shows that y inflicts more losses on x when operating at Y_1 above the boundary curve rather than at (X,Y), but in doing so y is defeated faster. Thus depending on his mission, a commander can choose to lengthen the battle or inflict more casualties on his opponent by choosing a suitable operating point which may be other than an equilibrium point.

TABLE III

Payoff Matrix and Mixed Strategies : Korean War

X	Y	2.5	2.7	2.9	3.2	3.4	3.6	3.8	4.0	4.2	4.4	4.7	4.8	5.0	5.3	5.5
	P_x^* \ P_y^*	0.0	0.0	0.0	0.0	0.0	0.53	0.30	0.17	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2.5	0.0	2.07	2.04	2.03	2.01	1.99	2.00	2.00	1.97	1.97	1.97	1.98	1.97	1.98	1.97	1.99
2.8	0.0	2.11	2.10	2.07	2.05	2.03	2.02	2.02	2.02	2.02	2.03	2.00	2.02	2.00	2.02	2.01
3.1	0.0	2.15	2.12	2.12	2.10	2.08	2.07	2.07	2.07	2.04	2.05	2.06	2.05	2.06	2.05	2.03
3.4	0.0	2.18	2.14	2.15	2.13	2.12	2.11	2.11	2.08	2.09	2.10	2.07	2.09	2.11	2.19	2.22
3.7	0.0	2.22	2.19	2.17	2.15	2.14	2.14	2.14	2.10	2.11	2.17	2.20	2.26	2.12	2.18	2.20
4.0	0.0	2.23	2.20	2.18	2.16	2.15	2.15	2.15	2.21	2.23	2.29	2.14	2.19	2.20	2.25	2.27
4.3	0.0	2.25	2.27	2.21	2.19	2.19	2.23	2.24	2.28	2.33	2.16	2.19	2.20	2.23	2.24	2.28
4.6	0.0	2.27	2.25	2.23	2.25	2.27	2.26	2.30	2.32	2.31	2.34	2.18	2.16	2.18	2.17	2.20
4.9	0.0	2.29	2.27	2.28	2.29	2.30	2.27	2.28	2.29	2.27	2.28	2.29	2.27	2.28	2.26	2.28
5.2	0.0	2.31	2.30	2.28	2.27	2.26	2.25	2.20	2.39	2.39	2.35	2.36	2.32	2.33	2.30	2.30
5.5	0.0	2.31	2.27	2.39	2.36	2.34	2.32	2.29	2.23	2.21	2.38	2.37	2.36	2.32	2.31	2.26
5.8	0.0	2.44	2.39	2.34	2.29	2.38	2.34	2.30	2.27	2.42	2.40	2.33	2.31	2.29	2.22	2.20
6.1	0.19	2.40	2.44	2.37	2.30	2.42	2.37	2.32	2.22	2.40	2.32	2.28	2.24	2.45	2.38	2.36
6.4	0.34	2.42	2.32	2.40	2.31	2.42	2.35	2.23	2.40	2.34	2.24	2.42	2.37	2.32	2.24	2.48
6.7	0.47	2.46	2.33	2.40	2.49	2.39	2.29	2.39	2.31	2.45	2.38	2.28	2.51	2.42	2.36	2.27

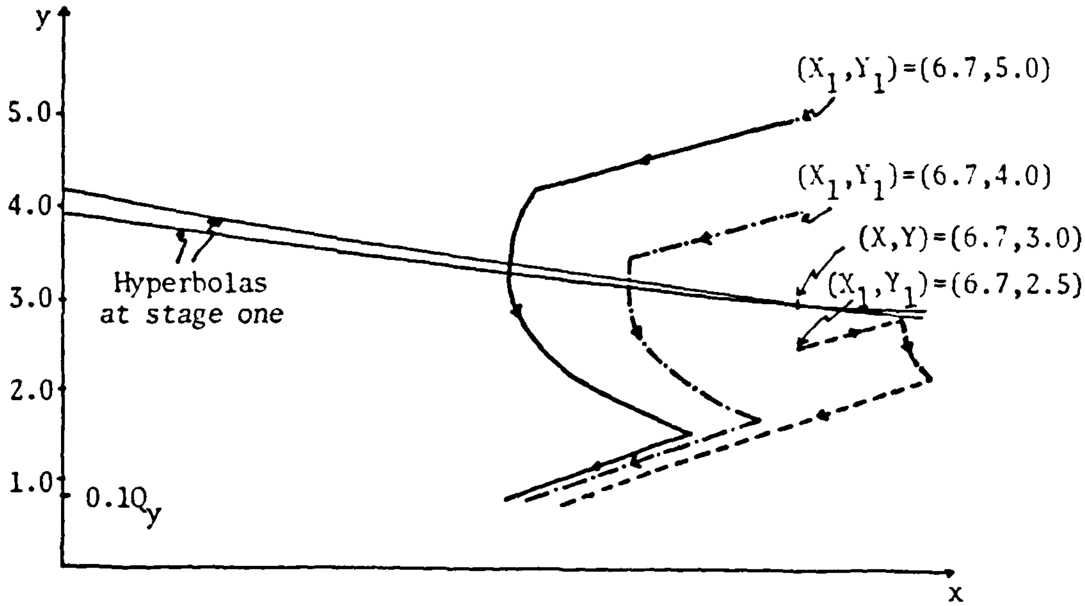


Figure 5.10 Initial Operating Points at Non-equilibrium Points.

TABLE IV

Effect of Operating at Non-equilibrium Points

Y_1	(Ly-Lx)	Finish Time (FINTIM)	Remarks
2.5	2.475	0.323	Below boundary
3.0	2.435	0.313	At equilibrium
4.0	2.353	0.293	Above boundary
5.0	2.305	0.283	Above boundary

VI. CONCLUSIONS AND RECOMMENDATIONS

A. CONCLUSIONS

This thesis has covered a number of subjects which are based on the generalized Lanchester Model. The first part of the results has to do with finding the equilibrium points in the N*N system. The Continuation Methods have been found to be suitable for this purpose. The advantages of the Continuation Methods over numerical techniques are numerous and important to our understanding of the non-linear set of equations. The method finds <u>all</u> the equilibrium solutions accurately and <u>does</u> <u>not</u> need good initial guesses. An example to compute the equilibrium solutions of a 2*2 system is presented along with a way to treat singularity problem.

The derivations and interpretations of the relations between stability and system parameters form the next major portion of the thesis. By considering the simpler 1*1 problem, a few interesting conclusions have been reached, namely

(1) Both the system asymptotes and equilibrium points are intrinsic to a system in equilibrium. The locations of the equilibrium points on the x-y and ε_x, ε_y planes completely characterize their stabilities; ε_x and ε_y are the differences in the system asymptotes;

(2) The dynamics of a system are characterized by the phase trajectories which represent the ways a battle progresses. Besides stability, the domains of attraction also influence the trajectories. The boundary curves which separate these domains can be ascertained by graphical or backward integration.

The last portion of the thesis integrates the concept of equilibrium stability and system dynamics. It relates these theoretical concept to operational problems. Two operational issues are addressed namely, (1) the effect of varying X and Y, the initial force deployment on battle outcome, (2) the exploitation of stability to influence battle outcome. A methodology which combines multistage battle simulation with two-person game has been employed and the conclusions are

(1) Initial force deployment, X and Y can be optimized by finding a set of mixed strategies. A suitable pure strategy can then be selected from the mixed strategies;

(2) Instability can and should be used to shape the course of battle and its outcome. This is particularly true in highly unstable warfare which is normally associated with large aimed-fire attritions. Unless defense resources are extremely large, it is not possible to completely reverse the outcome of a lopsided-battle where one side is much stronger than the other.

As far as military commanders are concerned, the above conclusions suggest two things. Firstly, depending on the relative strengths, it is not necessarily true that deploying the largest possible force will bring victory, reduce loss or even buy time. There is an optimum way of deploying available forces. Secondly, if a war involves large aimed-fire attritions due to weapons like aircraft, missiles, tanks, artillery, naval bombardment, etc., then initial victory which could perhaps be achieved through a preemptive strike certainly affects battle outcome.

B. RECOMMENDATIONS

1. Transformation of N*N Problem into the 1*1 Problem

It has been mentioned that the simplicity of the 1*1 problem can be attributed to the simpler mathematics involved and our ability to draw and visualize two-dimensional pictures. Despite its simplicity, it does share many of the properties with the N*N problem. Considering these factors, it seems logical that an attempt should be made to find the 1*1 equivalent to the N*N system. Another reason is that there is much to be gained in terms of savings in computational effort by going to the 1*1 equivalent.

Of course, the "equivalent" system will not be expected to be identical to the N*N problem in every aspect. One can only hope that it is equivalent in some sense, for example

(a) Preservation of stability characteristics and dynamics;

(b) Preservation of mixed strategies.

One way of transforming the N*N system into the equivalent 1*1 system is to equate losses in both systems. The equivalent system parameters are obtained by using relations such as

$$aX_{eq}Y_{eq} = \sum_i \sum_j a_{ij} x_i y_j$$

where

x_i, y_j = coordinates of the equilibrium point in the N*N problem

$$X_{eq} = \sum_i x_i$$

$$Y_{eq} = \sum_j y_j$$

The results of the preliminary studies suggest that this method of transformation can preserve some stability characteristics. The possibility of using the equivalent system to obtain the mixed strategies should not be dismissed until further studies have been conducted.

2. Time Variable Replenishment Coefficients

The replenishment rates used in the thesis have always been assumed to be constant. In actual wars, constant replenishment rates may not be used by either side; at times, it may not even be possible to do so. It would therefore be interesting to study the cases which involve time-varying replenishment rates $\tilde{r}(t)$. The choice of $\tilde{r}(t)$, for example periodic, non-periodic, ramp, etc. will depend on how well it represents practical replenishment rates. Whether a mathematical tool can be found to cope with the added complexity also has to be considered.

```
C                         APPENDIX A
C              SOLVING FOR OPTIMUM VECTORS    AND
C
C
C
C**********************************************************************
C                 THIS PROGRAM COMPUTES THE OPTIMUM FORCE
C           DISTRIBUTION,X* GIVEN X,Y,AND THE ATTRITION
C           COEFFICIENTS.PARAMETERS K1,K2 AND CONFLICT
C           MATRIX ARE ALSO OBTAINED.
C                 THIS IS AN INTERACTIVE PROGRAM.BEFORE EXECUTION
C           ,CHECK THE VALUES OF ATTRITION COEFFICIENT
C           MATRICES AA,BB,CC,D AND (U1,U2,V1,V2)
C           IN THE FILE LPD FORTRAN.THEN ENSURE YOU HAVE
C           ANOTHER FILE CALLED CCLIB EXEC.TO EXECUTE THE
C           PROGRAM,ENTER "COLIB LPD" AND FOLLOW INSTRUCTIONS.
C           ON INPUT OF X AND Y WILL CREATE THE A MATRIX
C           AND RUN THE LINEAR PROGRAM ZX4LP.
C**********************************************************************
C
C       A(M1+M2+2,N+M1+2)
C       B(M1+M2)
C       C(N)
C       PSOL(N)
C       DSOL(M1+M2)
C       RW(IA*(M1+M2+4)+2*(N+M1))
C       IW(2*(N+M1) +3)
        REAL A(4,7),B(2),C(3),S,PSOL(3),DSOL(2),RW(34),
       &IER,X,Y,M,PHI,BB(2,3),D(2,3)
       &,X1,X2,Y1,Y2,Y3,K1,K2,CC(2,3),AA(2,3),CON(5,5)
        INTEGER Q,IW(13),M1,M2,IA,N,R
C**********************************************************************
C           VARIABLE DEFINITIONS
C**********************************************************************
C
C           AA,BB,CC,D,U1,U2,V1,V2=ATTRTION COEFFICIENTS
C           PSOL,DSOL=PRIMARY AND DUAL SOLUTION
C           M1=NUMBER OF EQUALITY CONSTRAINTS
C           M2=NUMBER OF INEQUALITY CONSTRAINTS
C           RW,IW=PARAMETERS USED BY IMSL ROUTINE ZX4LP
C           IER=ERROR CODE FROM ZX4LP
C           N=NUMBER OF UNKNOWNS
C           A=MATRIX DEFINED IN EQUATION A.3 OF PAPER
C             "LANCHESTER EQUATIONS AND GAME THEORY"
C             BY P.H.MOUSE ANDJ.M.WOZENCRAFT
C           C=VECTOR CONTAINING COEEFICIENTS OF
C           OF OBJECTIVE FUNCTION IN LP
C           B=VECTOR CONTAINING THE RIGHTHAND SIDES
C           OF THE CONSTRAINTS.
C           IA=M1+M2+2;IS THE ROW DIMENSION OF A MATRIX
C           Q=COLUMN DIMENSION OF A
        M1=2
        M2=0
        N=3
        IA=M1+M2+2
        Q=N+M1+2
        R=M1+M2
C
        U1 =   .30
        U2 =   .30
        V1 =   .10
        V2 =   .0545
        V3 =   2.0
C
        AA(1,1)  =   1.0
        AA(1,2)  =  .3
        AA(1,3)  =   0.0
```

```
      AA(2,1)  =   .60
      AA(2,2)  =   .900
      AA(2,3)  =   0.0
C
      BB(1,1)  =  0.15
      BB(1,2)  =   .10
      BB(1,3)  =   0.0
      BB(2,1)  =  0.15
      BB(2,2)  =   .3
      BB(2,3)  =   0.0
C
      CC(1,1)  =   1.2
      CC(1,2)  =   1.0
      CC(1,3)  =   15.
      CC(2,1)  =   1.1
      CC(2,2)  =   0.6
      CC(2,3)  =   15.0
C
      D(1,1)  =  .00
      D(1,2)  =  .00
      D(1,3)  =  0.00
      D(2,1)  =  .00
      D(2,2)  =  .00
      D(2,3)  =  0.0
C
C*****************************************************************
C          REPETITION LOOP STARTS HERE
C*****************************************************************
C
7000  CONTINUE
C
C          ENTER X AND Y
C
          WRITE(6,296)
296       FORMAT(1X,'ENTER X AND Y ONE AT A TIME')
          READ(5,*) X,Y
C
C
          B1= BB(1,1)+BB(2,1)
          B2= BB(1,2)+BB(2,2)
          B3= BB(1,3)+BB(2,3)
          D1= D(1,1)+D(1,2)+D(1,3)
          D2= D(2,1)+D(2,2)+D(2,3)
C
          A(1,1)= CC(1,1) - AA(1,1) + (D1-U1)/Y - (B1-V1)
     &    /X+15.
          A(1,2)= CC(1,2) - AA(1,2) + (D1-U1)/Y - (B2-V2)
     &    /X+15.
          A(1,3)= CC(1,3) - AA(1,3) + (D1-U1)/Y - (B3-V3)
     &    /X+15.
          A(2,1)= CC(2,1) - AA(2,1) + (D2-U2)/Y - (B1-V1)
     &    /X+15.
          A(2,2)= CC(2,2) - AA(2,2) + (D2-U2)/Y - (B2-V2)
     &    /X+15.
          A(2,3)= CC(2,3) - AA(2,3) + (D2-U2)/Y - (B3-V3)
C
          WRITE(14,999) ((I,J,A(I,J),J=1,2),I=1,2)
999       FORMAT('0',3X,'A(',I3,I3,')=',F10.3)
C
          B(1)=1.0
          B(2)=1.0
1000      CONTINUE
C
          C(1)=1.0
          C(2)=1.0
          C(3)=1.0
C
```

```
      WRITE(14,130)
130   FORMAT(1X,'COEFFICIENTS OF CONSTRAINTS (A)')
      DO 300 I=1,R
         WRITE(14,200)   (A(I,K),K=1,N)
C           THIS NUMBER = N+M1+2
200      FORMAT(1X,7F15.6)
300   CONTINUE
C
      CALL ZX4LP(A,IA,B,C,N,M1,M2,S,PSOL,DSOL,RW,
     &   IW,IER)
C
C*************************************************************
C     COMPUTE Y1,Y2,Y3,X1,X2,M
C*************************************************************
C
      PHI=S
      Y1=PSOL(1)*Y/PHI
      Y2=PSOL(2)*Y/PHI
      Y3=PSOL(3)*Y/PHI
      X1=DSOL(1)*X/PHI
      X2=DSOL(2)*X/PHI
      M=X*Y/(PHI-15)
      K1=(  (B1-V1)*Y1 + (B2-V2)*Y2 + (B3-V3)*Y3 + M )/X
      K2=(  (D1-U1)*X1 + (D2-U2)*X2 - M )/Y
C
C*************************************************************
C     COMPUTE CONFLICT MATRIX
C*************************************************************
C
      CON(1,1)= U1 + AA(1,1)*Y1 + AA(1,2)*Y2 + AA(1,3)
     &   *Y3
      CON(1,2)= 0.0
      CON(1,3)= AA(1,1)*X1  + BB(1,1)
      CON(1,4)= AA(1,2)*X1  + BB(1,2)
      CON(1,5)= AA(1,3)*X1  + BB(1,3)
      CON(2,1)= 0.0
      CON(2,2)= U2  + AA(2,1)*Y1 + AA(2,2)*Y2 + AA(2,3)
     &   *Y3
      CON(2,3)= AA(2,1)*X2  + BB(2,1)
      CON(2,4)= AA(2,2)*X2  + BB(2,2)
      CON(2,5)= AA(2,3)*X2  + BB(2,3)
      CON(3,1)= CC(1,1)*Y1  + D(1,1)
      CON(3,2)= CC(2,1)*Y1  + D(2,1)
      CON(3,3)= V1  + CC(1,1)*X1 + CC(2,1)*X2
      CON(3,4)= 0.0
      CON(3,5)= 0.0
      CON(4,1)= CC(1,2)*Y2  + D(1,2)
      CON(4,2)= CC(2,2)*Y2  + D(2,2)
      CON(4,3)= 0.0
      CON(4,4)= V2  + CC(1,2)*X1 + CC(2,2)*X2
      CON(4,5)= 0.0
      CON(5,1)= CC(1,3)*Y3  + D(1,3)
      CON(5,2)= CC(2,3)*Y3  + D(2,3)
      CON(5,3)= 0.0
      CON(5,4)= 0.0
      CON(5,5)= V3  + CC(1,3)*X1 + CC(2,3)*X2
      DO 2100 L4=1,5
        DO 2110 L5= 1,5
          CON(L4,L5)= -1.0*CON(L4,L5)
2110    CONTINUE
2100  CONTINUE
C
C*************************************************************
C     PRINT RESULTS
C*************************************************************
C
      WRITE(14,150)
```

```
150       FORMAT(1X,'RIGHT HAND SIDES OF CONSTRAINTS (B)')
          WRITE(14,160)  (B(I),I=1,R)
C                                  THIS NUMBER =M1+M2
160       FCRMAT(1X,2F10.4)
          WRITE(14,170)
          WRITE(14,180)  (C(I),I=1,N)
170       FORMAT(1X,'COEFFICIENTS OF OBJECTIVE FUNCTION(C)')
C                                       THIS NUMBER =N
180       FORMAT(1X,3F10.4)
          WRITE(14,190)  N
          WRITE(14,191)  M1
          WRITE(14,192)  M2
190       FORMAT(1X,'NUMBER CF UNKNCWNS (N) = ',I20)
191       FCRMAT(1X,'NUMBER CF INEQUALITY CONSTRAINTS(M1)='
      &   ,I5)
192       FORMAT(1X,'NUMBER OF EQUALITY CCNSTRAINTS(M2)= '
      &   ,I7)
          WRITE(14,210)  S
210       FORMAT(1X,'VALUE CF OBJECTIVE FUNCTION(S) ='
      &   ,F15.6)
          WRITE(14,220)  (PSCL(I),I=1,N)
C                                         THIS NUMBER = N
220       FCRMAT(1X,'VALUE CF PRIMAL SOLUTION (PSOL) = '
      &   ,3F15.6)
          WRITE(14,230)  (DSCL(I),I=1,R)
C                                         THIS NUMBER = M1+M2
230       FCRMAT(1X,'VALUE CF DUAL SOLUTION (DSOL) = '
      &   ,2F15.6)
          WRITE(14,240)  IER
240       FCRMAT(1X,'IER = ',I5)
          WRITE(14,570)  X
          WRITE(14,580)  Y
          WRITE(14,530)  X1
          WRITE(14,540)  X2
          WRITE(14,500)  Y1
          WRITE(14,510)  Y2
          WRITE(14,520)  Y3
          WRITE(14,550)  M
          WRITE(14,660)  K1
          WRITE(14,670)  K2
570       FCRMAT(1X,'X  = ',F15.6)
580       FORMAT(1X,'Y  = ',F15.6)
500       FCRMAT(1X,'Y1 = ',F15.6)
510       FCRMAT(1X,'Y2 = ',F15.6)
520       FCRMAT(1X,'Y3 = ',F15.6)
530       FCRMAT(1X,'X1 = ',F15.6)
540       FORMAT(1X,'X2 = ',F15.6)
550       FCRMAT(1X,'M  = ',F15.6)
660       FCRMAT(1X,'K1 = ',F15.6)
670       FCRMAT(1X,'K2 = ',F15.6)
C
C*****************************************************************
C      PRINT PARAMETERS
C*****************************************************************
C
          WRITE(14,705)  U1
          WRITE(14,710)  U2
          WRITE(14,715)  V1
          WRITE(14,720)  V2
          WRITE(14,725)  V3
          WRITE(14,760)
          WRITE(14,750)  ((AA(I,J),J=1,3),I=1,2)
          WRITE(14,770)
          WRITE(14,750)  ((BB(I,J),J=1,3),I=1,2)
          WRITE(14,780)
          WRITE(14,750)  ((CC(I,J),J=1,3),I=1,2)
          WRITE(14,790)
```

```
      WRITE(14,750) ((D(I,J),J=1,3),I=1,2)
705   FORMAT(1X,'U1 = ',F15.6)
710   FORMAT(1X,'U2 = ',F15.6)
715   FORMAT(1X,'V1 = ',F15.6)
720   FORMAT(1X,'V2 = ',F15.6)
725   FORMAT(1X,'V3 = ',F15.6)
750   FORMAT(1X,3F15.5)
760   FORMAT(1X,'VALUE OF A MATRIX (AA)')
770   FORMAT(1X,'VALUE OF B MATRIX (BB)')
780   FORMAT(1X,'VALUE OF C MATRIX (CC)')
790   FORMAT(1X,'VALUE OF D MATRIX (D)')
C
C****************************************************************
C       PRINT CONFLICT MATRIX
C****************************************************************
C
      WRITE(14,673)
      WRITE(14,674) ((CON(L4,L5),L5=1,5),L4=1,5)
673   FORMAT(1X,'VALUE OF THE CONFLICT MATRIX (CON)')
674   FORMAT(1X,5F14.6)
C
C****************************************************************
C       PRINT MEANING OF IER
C****************************************************************
C
      WRITE(14,390)
      WRITE(14,400)
      WRITE(14,410)
      WRITE(14,420)
      WRITE(14,430)
      WRITE(14,440)
      WRITE(14,450)
      WRITE(14,460)
390   FORMAT(1X,' ')
400   FORMAT(1X,'IER=130 INDICATES M2>N')
410   FORMAT(1X,'IER=131 INDICATES EXCESSIVE ITERATIONS
     &')
420   FORMAT(1X,'IER=132 INDICATES REDUNDANCIES IN
     &CONSTRAINTS')
430   FORMAT(1X,'IER=134 INDICATES OBJECTIVE FUNCTION
     &UNBOUNDED')
440   FORMAT(1X,'IER=135 INDICATES CONSTRAINTS
     &INFEASIBLE')
450   FORMAT(1X,'IER=136 INDICATES PRIMARY OR DUAL
     &SOLUTIONS')
460   FORMAT(9X,'DO NOT SATISFY THE CONSTRAINTS')
C
C****************************************************************
C       REITERATION ROUTINE
C****************************************************************
C
      WRITE(6,470)
      WRITE(6,480)
470   FORMAT(1X,'TO CONTINUE ENTER 1')
480   FORMAT(1X,'TO STOP ENTER -1')
      READ(5,*) L7
      IF(ABS(L7+1.0).LT.1.0E-5) GOTO 9000
      WRITE(14,490)
      WRITE(14,491)
      WRITE(14,492)
490   FORMAT(1X,' ')
491   FORMAT(1X,'*********************************')
492   FORMAT(1X,' ')
      GOTO 7000
9000  CONTINUE
      STOP
```

APPENDIX B

NUMBER OF EQUILIBRIUM SOLUTIONS OF THE N*N PROBLEM

We first consider the 2*2 and 3*3 problem and extend the results to the N*N problem.

1. 2*2 Problem

In general, each trivial solution leads to a unique equilibrium solution in a continuation process and there will be equal number of equilibrium and trivial solutions. Section 4 discusses what happens when there is degeneracy. The trivial solutions are obtained by solving

$$\begin{aligned} z_1(u_1+a_{11}z_3+a_{12}z_4) &= 0 \\ z_2(u_2+a_{21}z_3+a_{22}z_4) &= 0 \\ z_3(u_3+c_{11}z_1+c_{21}z_2) &= 0 \\ z_4(u_4+c_{12}z_1+c_{22}z_2) &= 0 \end{aligned} \qquad \text{(eqn B.1)}$$

At first glance, there would seem to be 2^4 trivial solutions corresponding to the number of ways of making the lefthand sides of equation B.1 zero. Each lefthand sides can be made zero by either making z_i or the terms in parenthesis equal to zero. But closer examination reveals only six allowed cases, in non-degenerate cases, corresponding to $(1 + 2^2 + 1) = 6$ solutions. Table V shows how these cases arise.

TABLE V

Trivial Solution for 2*2 Probem

Case	z_i which are made zero	Remarks
1	z_1, z_2, z_3, z_4	
2	z_1, z_3	$2^2 = 4$ cases
3	z_1, z_4	
4	z_2, z_3	
5	z_2, z_4	
6	--	all terms in parenthesis equal to 0
7	$z = 0$	degenerate case; see section 4.

2. 3*3 problem

Here, the trivial solutions are obtained from

$$z_1(u_1+a_{11}z_4+a_{12}z_5+a_{13}z_6) = 0$$
$$z_2(u_2+a_{21}z_4+a_{22}z_5+a_{23}z_6) = 0$$
$$z_3(u_3+a_{31}z_4+a_{32}z_5+a_{33}z_6) = 0$$
$$z_4(u_4+c_{11}z_1+c_{21}z_2+c_{31}z_3) = 0$$
$$z_5(u_5+c_{12}z_1+c_{22}z_2+c_{32}z_3) = 0$$
$$z_6(u_6+c_{13}z_1+c_{23}z_2+c_{33}z_3) = 0$$

Again, there are some cases which are not allowed because of inconsistencies in non-degenerate cases. Table VI shows the different cases.

In this case, the total number of allowed cases is

$$[\ 1 + \binom{3}{2}^2 + \binom{3}{2}^2 + 1\ ] = 20$$

TABLE VI

Trivial Solution for 3*3 Problem

Case	z_i which are made zero	Remarks
1	$z_1,\ z_2 \ldots,\ x_6$	
2	$z_1,\ z_4$	
3	$z_1,\ z_5$	
4	$z_1,\ z_6$	
5	$z_2,\ z_4$	$\binom{3}{1}^2$ = 9 cases
6	$z_2,\ z_5$	
7	$z_2,\ z_6$	
8	$z_3,\ z_4$	
9	$z_3,\ z_5$	
10	$z_3,\ z_6$	
11	$z_1,\ z_2\ ;\ z_4,\ z_5$	
12	$z_1,\ z_2\ ;\ z_4,\ z_6$	
13	$z_1,\ z_2\ ;\ z_5,\ z_6$	
14	$z_2,\ z_3\ ;\ z_4,\ z_5$	
15	$z_1,\ z_3\ ;\ z_4,\ z_6$	$\binom{3}{2}^2$ = 9 cases
16	$z_1,\ z_3\ ;\ z_5,\ z_6$	
17	$z_2,\ z_3\ ;\ z_4,\ z_6$	
18	$z_2,\ z_3\ ;\ z_4,\ z_6$	
19	$z_2,\ z_3\ ;\ z_5,\ z_6$	
20	—	all items in parenthesis equals to 0.
21	$z_2,\ z_3\ ;\ z_4$	degenerate case; see section 4.
22	z_1	degenerate case; see section 4.

3. Extension to N*N Problem

Now we know the type of allowed cases, it is easy to recognise the pattern of results. The pattern looks like a Pascal triangle with each element squared.

N*N		Number of Equilibria (N_K)
1*1	$1^2 + 1^2$	= 2
2*2	$1^2 + 2^2 + 1^2$	= 6
3*3	$1^2 + 3^2 + 3^2 + 1^2$	= 20
4*4	$1^2 + 4^2 + 6^2 + 4^2 + 1^2$	= 70
5*5	$1^2 + 5^2 + 10^2 + 10^2 + 5^2 + 1^2$	= 252

For N*N in general,

$$N_k = \sum_{i=0}^{N} \binom{N}{i}^2$$

N_k can also be written as

$$N_k = \binom{2N}{N}$$

The proof of the identity,

$$\sum_{i=0}^{N} \binom{N}{i}^2 = \binom{2N}{N}$$

can be found in [Ref. 12].

4. Degeneracies

For the 2*2 problem, a degeneracy can arise when z_1 (or any other z_i) is made equal to zero. The other three lefthand sides in equation B.1 are made equal to zero making the terms in the parentheses equal to zero.

In general, such a case does not correspond to a new trivial solution because there is an inconsistency when $z_2 = -u_3/c_{21}$ and $z_2 = -u_4/c_{22}$. However, it seems that by making $u_3/c_{21} = u_4/c_{22}$, the inconsistency no longer exists and there will be an infinite number of trivial solution as long as z_3, z_4 are chosen to satisfy $u_2 + a_{21}z_3 + a_{22}z_4 = 0$. It turns out that the number of equilibrium solutions still remains at a maximum of six (disregarding the case with infinite number of equilibria). Similarly for the 3*3 case, even if there is degeneracy, the number of equilibria will not exceed 20.

That the above is true can be shown by considering the 1*1 problem. In this case, the two non-degenerate cases correspond to (a) $x = y = 0$ (b) $v + cx = 0$ and $u + ay = 0$. A degeneracy can occur if $v = 0$ or $u = 0$ in which case there seems to be an infinite number of trivial solutions lying on the y or x axis respectively and hence an infinite number of equilibria. But when the actual hyperbolas are plotted, there are only two intersections and hence two equilibrium points. Furthermore, when the Continuation method is used to find the equilibria, there are only two equilibria irrespective of whether the trivial solutions are chosen to be degenerate or not.

```
C
C
C                   APPENDIX C
C         PROGRAM LISTING FOR SOLVING 2*2 SYSTEM
C             USING CONTINUATION METHOD
C
C
C
C*****************************************************************
C       THIS PROGRAM SOLVES A 2*2 SYSTEM FOR THEIR
C       EQUILIBRIUM PCINTS USING CONTINUATION METHODS.SEE
C       SECTICN IIIB FOR THECRY AND IMPLEMENTATICNS.
C*****************************************************************
C
        REAL * 8 A,X,CEL,T,WKAREA,FNORM,WK,U,R,B,C,D,EP,AP,
      &XT,DX,OMAX,XFTEMP,TLEFT,TS,F1,F2,F3,F4,ERR
        REAL * 4 XA,XE,XC,XD,TA
        INTEGER I,J,K,N,M,MM,COUNT,IA,IDGT,IER,NSIG,IE,IREP
      &,CON,KK,NRECON,SIGN
      &,FAST,QCOUNT,RUN,ERFLAG,REFLAG,CHFLAG,CTR,CP1,JJ
        CIMENSION WKAREA(100),WK(115),BP(4),AP(4,4),
      &XA(1005),XE(1C05),XC(1005)
      &,XD(1005),TA(1005),DX(4)
        EXTERNAL FCN
        COMMON /REPLEN/ R(4)
        COMMON /PARAM/ A(2,2),B(2,2),C(2,2),D(2,2),U(4)
        COMMON /TREPAR/ XT(4),X(4),T(4),TS,FAST,CON,COUNT,N
      &,OMAX,QCOUNT,RUN,XFTEMP,SUM,SIGN,IER
        DATA DEL/1.0D-16/,
      &MM/1/,M/1/,NSIG/5/,ITMAX/200/,IDGT/0/,IA/4/,
      &ERFLAG/0/,REFLAG/0/
C
C*****************************************************************
C       VARIABLE CEFINATIONS
C*****************************************************************
C
C       AP=SEE EQUATION 3-7 FCR THE MEANING.
C       EP=AS ABCVE
C       X=THE RCCT OF THE 2*2 SYSTEM TO BE EVALUATED BY THE
C           CONTINUATION PROCESS.
C       T=PARAMETER DEFINING THE HUMOTOPY;HAS VALUE BETWEEN
C       C ANC 1.C
C       CEL=SMALL TIME INTERVAL USED TO APPROXIMATE THE
C             PARTIAL TIME DERIVATIVE.
C       AS,BS,CS,DS=THE ORIGINAL A,B,C,D PARAMETERS IN THE
C                     LANCHESTER EQUATION
C       U,R=CCRRESPCND TO THE SELF ATTRITION & REPLENISH-
C             MENT CCEEF. IN THE LANCHESTER EQUATION.
C       TS=PARTITION INTERVAL FRCM T=0 TC T=1 IN THE
C       CONTINUATION PROCESS.
C       COUNT=COUNTER FOR THE # OF TIME STEPS ADVANCEC IN
C               THE CONTINUATION PROCESS.
C       CON=THE CCNDITION FOR REACHING T=1
C       FCN=A SUBROUTINE USEC BY THE IMSL ROUTINE ZSCNT.
C       WK,NSIG,ITMAX,IER,FNCRM=PARAMETERS IN THE ROUTINE
C                                 ZSCNT.
C       IE,IA=PARAMETERS IN THE ROUTINE LEQT1F.
C       CTR=COUNTER FOR PLOTTING THE CURVES
C       NRECON=REPETITION CCUNTER.
C
C*****************************************************************
C
C           INPUT ATTRITION CCFFICIENTS,COUNTERS AND FLAGS.
C
C*****************************************************************
C
        A(1,1)=1.0D0
```

```
      A(1,2)=0.3D0
      A(2,1)=0.6D0
      A(2,2)=0.9D0
      B(1,1)=0.15D0
      B(1,2)=0.1D0
      B(2,1)=0.15D0
      B(2,2)=0.3D0
      C(1,1)=1.20D0
      C(1,2)=1.0D0
      C(2,1)=1.10D0
      C(2,2)=0.6D0
      D(1,1)=0.0D0
      D(1,2)=0.0D0
      D(2,1)=0.00D0
      D(2,2)=0.0D0
      U(1)=0.3D0
      U(2)=0.3D0
      U(3)=0.1D0
      U(4)=0.2D0
      N=4
      T(1)=0.0D0
      T(2)=0.0D0
      T(3)=0.0D0
      T(4)=0.0D0
      COUNT=1
      CCOUNT=0
      RON=1
      IER=0
      IREP=1
      NRECON=0
C
C*****************************************************************
C     CALCULATE R VECTOR USING ONE POSITIVE EQUILIBRIUM
C     SOLUTION OR MODIFY R VECTOR AS APPROPRIATE.
C*****************************************************************
C
      X(1)=0.6153BD0
      X(2)=0.3846D0
      X(3)=3.0769D0
      X(4)=0.9231D0
      R(1)=X(1)*(U(1)+A(1,1)*X(3)+A(1,2)*X(4))+B(1,1)
     &*X(3)+B(1,2)*X(4)
      R(2)=X(2)*(U(2)+A(2,1)*X(3)+A(2,2)*X(4))+B(2,1)
     &*X(3)+B(2,2)*X(4)
      R(3)=X(3)*(U(3)+C(1,1)*X(1)+C(2,1)*X(2))+D(1,1)
     &*X(1)+D(2,1)*X(2)
      R(4)=X(4)*(U(4)+C(1,2)*X(1)+C(2,2)*X(2))+D(1,2)
     &*X(1)+D(2,2)*X(2)
      WRITE(1,9998)(I,X(I),I,R(I),I=1,4)
9998  FORMAT('0',3X,'X(',I3,')=',F12.4,4X,'R(',I3,')=',
     &F12.4)
C
C*****************************************************************
C
C     WHILE IREP (THE CODE FOR REPEATINT ) IS 1,THE CONT-
C     INUATION PROCESS WILL BE REPEATED.EACH REPE TITION
C     WILL INCREASE TS IN MULTIPLES OF 0.005.
C
C*****************************************************************
C
7     IF(.NOT.(IREP.EQ.1)) GO TO 8
          NRECON=NRECON+1
C
C*****************************************************************
C     SET TRIVIAL SOLUTION
C*****************************************************************
C
```

```
      X(1)=-C.10C00D0/1.20000
      X(2)=-C.000000/1.10000
      X(3)=-C.3000/1.00000
      X(4)=-C.C000/0.30000
      T(1)=C.0C0
      T(2)=C.CD0
      T(3)=0.000
      T(4)=C.000
      CHFLAG=0
      QCOUNT=0
      RON=1
      ERFLAG=C
      REFLAG=C
      COUNT=1
      IER=0
      CTR=1
      TS=DFLCAT(NRECON)*0.00500D
      CON=IDINT((1.0D0-T(1))/TS)+2
C
C*********************************************************************
C     WHILE T<1 OR CORRECTOR STEP DOES NOT RETURN WITH
C     ERROR FLAG IER
C*********************************************************************
C
5     IF(.NOT.((IER.NE.129).AND.(IER.NE.130).AND.
     &  (IER.NE.131).AND.(COUNT.LE.CON))) GO TO 6
C
C     TREAT THE SINGULAR CASE IF REFLAG OR ERFLAG
C     IS SET.
C
         CALL TREAT(REFLAG,ERFLAG,CHFLAG)
C
C*********************************************************************
C     PREDICTOR STEP:
C     SET UP THE SET OF DIFFERENTIAL EQUATION AND
C     CALL LEQT1F TO SOLVE THE SYSTEM AX=B
C*********************************************************************
C
         DO 101 I=1,MM
            BP(1)=(R(1)-B(1,1)*X(3)-B(1,2)*X(4))*DEL
            BP(2)=(R(2)-B(2,1)*X(3)-B(2,2)*X(4))*DEL
            BP(3)=(R(3)-D(1,1)*X(1)-D(2,1)*X(2))*DEL
            BP(4)=(R(4)-D(1,2)*X(1)-D(2,2)*X(2))*DEL
            AP(1,1)=U(1)+A(1,1)*X(3)+A(1,2)*X(4)
            AP(1,2)=0.0D0
            AP(1,3)=A(1,1)*X(1)+B(1,1)*T(1)
            AP(1,4)=A(1,2)*X(1)+B(1,2)*T(1)
            AP(2,1)=0.0D0
            AP(2,2)=U(2)+A(2,1)*X(3)+A(2,2)*X(4)
            AP(2,3)=A(2,1)*X(2)+B(2,1)*T(2)
            AP(2,4)=A(2,2)*X(2)+B(2,2)*T(2)
            AP(3,1)=C(1,1)*X(3)+D(1,1)*T(3)
            AP(3,2)=C(2,1)*X(3)+D(2,1)*T(3)
            AP(3,3)=U(3)+C(1,1)*X(1)+C(2,1)*X(2)
            AP(3,4)=0.0D0
            AP(4,1)=C(1,2)*X(4)+D(1,2)*T(4)
            AP(4,2)=C(2,2)*X(4)+D(2,2)*T(4)
            AP(4,3)=0.0D0
            AP(4,4)=U(4)+C(1,2)*X(1)+C(2,2)*X(2)
            CALL LEQT1F(AP,M,N,IA,BP,IDGT,WKAREA,IE)
            DO 102 K=1,N
               X(K)=BP(K)+X(K)
 102        CONTINUE
101      CONTINUE
C
C*********************************************************************
C     CORRECTOR STEP:
```

```
C*************************************************************
C
            CALL ZSCNT (FCN,NSIG,N,ITMAX,T,X,FNCRM,WK,IER)
            UMAX=25C.0D0*DMIN1(DX(1),DX(2),DX(3),DX(4))
C
C       COMPUTE DIFFERENCE BETWEEN PREVIOUS AND CURRENT X
C
            DO 104 K=1,N
               DX(K)=DABS(XT(K)-X(K))
104         CONTINUE
C
C       CALL SUBROUTINE TO DETECT FAST CHANGING COMPONENT
C
            CALL DETECT(DX,REFLAG,ERFLAG)
C
C
C*************************************************************
C       COMPUTE ERROR IF T IS NEAR 1.0
C*************************************************************
C
            IF(.NOT.(DABS(T(1)-1.0D0).LE.0.0005D0)) GO TO
     &      1031
               F1=-X(1)*(U(1)+A(1,1)*X(3)+A(1,2)*X(4))
     &         +R(1)-B(1,1)*X(3)-B(1,2)*X(4)
               F2=-X(2)*(U(2)+A(2,1)*X(3)+A(2,2)*X(4))
     &         +R(2)-B(2,1)*X(3)-B(2,2)*X(4)
               F3=-X(3)*(U(3)+C(1,1)*X(1)+C(2,1)*X(2))
     &         +R(3)-D(1,1)*X(1)-D(2,1)*X(2)
               F4=-X(4)*(U(4)+C(1,2)*X(1)+C(2,2)*X(2))
     &         +R(4)-D(1,2)*X(1)-D(2,2)*X(2)
               ERR=DSQRT(F1**2+F2**2+F3**2+F4**2)
               WRITE(1,997)T(1),ERR
1031        CONTINUE
C
C*************************************************************
C       OUTPUT VALUE OF X VECTOR WHEN T IS NEAR 1.0
C*************************************************************
C
            IF(.NOT.(T(1).GE.(0.99D0)))GO TO 865
               WRITE(1,955)T(1)
               WRITE(1,999)(K,X(K),K=1,N)
865         CONTINUE
C
C*************************************************************
C       CHANGE DOUBLE TO SINGLE PRECISION FOR PLOTTING
C*************************************************************
C
            TA(CTR)=SNGL(T(1))
            XA(CTR)=SNGL(X(1))
            XB(CTR)=SNGL(X(2))
            XC(CTR)=SNGL(X(3))
            XD(CTR)=SNGL(X(4))
            COUNT=CCUNT+1
            CTR=CTR+1
            GO TO 5
6        CONTINUE
C
C*************************************************************
C       COMMENCE PLOTTING ROUTINE
C*************************************************************
C
         CALL TEK618
C        CALL COMPRS
         CALL PAGE(14.0,10.5)
         CALL NOBRDR
         CALL BLOWUP(0.4)
         CALL AREA2D(12.0,8.0)
```

```
      CALL XNAME('TIME IN SECS$',100)
      CALL YNAME('                X(I) $',100)
      CALL XNONUM
      CALL YNONUM
C
C     INPUT HEADING IF REQUIRED
C
C     CALL HEADIN('INPUT HEADING HERE****************$'
C    ,100,1.,4)
C     CALL HEADIN('************AND HERE****************$'
C    ,100,1.,4)
C     CALL HEADIN('**********AND HERE*******$',100,1.,4)
C     CALL HEADIN('**********AND HERE***************$'
C    ,100,1.,4)
      CALL GRAF(0.0,'SCALE',1.00,-56.0,'SCALE',35.0)
      CP1=CTR-1
      CALL CURVE(TA,XA,CP1,-1)
      CALL CURVE(TA,XB,CP1,-1)
      CALL CURVE(TA,XC,CP1,-1)
      CALL CURVE(TA,XD,CP1,1)
      CALL ENDPL(0)
      WRITE(6,9910)
 9910 FORMAT('0',3X,'ENTER 1 TO CONTINUE',3X,'OTHER NO.
     & TO STOP')
      READ(5,*)IREP
      GO TO 7
 8    CONTINUE
      CALL CONEFL
      STOP
955   FORMAT('0',3X,'T=',D24.12)
997   FORMAT('0',3X,'T=',D24.14,3X,'ERROR=',D24.6)
999   FORMAT('0',3X,'X(',I3,')=',D24.14/)
      END
C*************************************************************
C
C     SUBPROGRAM FCN REQUIRED BY THE ROUTINE ZSCNT
C
C*************************************************************
      SUBROUTINE FCN(X,F,N,T)
      INTEGER N,I,J,NN
      REAL * 8 X,F,R,U,A,B,C,D,T
      DIMENSION X(4),F(4),T(4)
      COMMON /REFLEN/ R(4)
      COMMON /PARAM/ A(2,2),B(2,2),C(2,2),D(2,2),U(4)
      F(1)=-X(1)*(U(1)+A(1,1)*X(3)+A(1,2)*X(4))+(R(1)
     &-B(1,1)*X(3)-B(1,2)*X(4))*T(1)
      F(2)=-X(2)*(U(2)+A(2,1)*X(3)+A(2,2)*X(4))+(R(2)
     &-B(2,1)*X(3)-B(2,2)*X(4))*T(2)
      F(3)=-X(3)*(U(3)+C(1,1)*X(1)+C(2,1)*X(2))+(R(3)
     &-D(1,1)*X(1)-D(2,1)*X(2))*T(3)
      F(4)=-X(4)*(U(4)+C(1,2)*X(1)+C(2,2)*X(2))+(R(4)
     &-D(1,2)*X(1)-D(2,2)*X(2))*T(4)
      RETURN
      END
C
C*************************************************************
C
C         SUBROUTINE TO TREAT SINGULAR CASE.WHEN EITHER
C         THE "REVERSE" FLAG,REFLAG OR THE ERROR FLAG,
C         ERFLAG IS SET,THE PARTITION INTERVAL,TS WILL BE
C         INCREASED.IN ADDITION,IF REFLAG IS SET,THE FAST
C         CHANGING COMPONENT WHICH HAS BEEN FOUND NOT TO
C         FLIP CORRECTLY WILL HAVE ITS SIGN CHANGED.
C
C*************************************************************
C
          SUBROUTINE TREAT(REFLAG,ERFLAG,CHFLAG)
```

```
      REAL * 8 XT,X,T,TS,OMAX,XFTEMP,SUM
       INTEGER K,N,COUNT,CON,FAST,QCOUNT,RON,SIGN,IER
      COMMON /TREPAR/ XT(4),X(4),T(4),TS,FAST,CON,COUNT,N
     &,OMAX,QCOUNT,RON,XFTEMP,SUM,SIGN,IER
       DO 105 K=1,N
          IF(.NOT.(((REFLAG.NE.1).AND.(ERFLAG.NE.1))
     &.OR.(CHFLAG.EQ.1))) GO TO 10
              T(K)=T(K)+TS
              XT(K)=X(K)
C
C     IF NEED TO SET X(FAST)=-XT(FAST) & ERFLAG IS SET
C
10        IF(.NOT.((REFLAG.EQ.1).AND.(ERFLAG.EQ.1)))
     &    GO TO 11
            X(FAST)=-XT(FAST)
            XT(FAST)=X(FAST)
            IF(.NOT.(CHFLAG.EQ.0)) GO TO 1011
                T(K)=T(K)+TS
                T(2)=T(1)
                T(3)=T(2)
                T(4)=T(3)
                TS=2.0D0*TS
                CHFLAG=1
C               T(K)=T(K)+TS
                CON=IDINT((1.0D0-T(1))/TS)+3
                COUNT=1
1011        CONTINUE
            REFLAG=0
            ERFLAG=0
          GO TO 20
C
C     IF ONLY ERFLAG IS SET
C
11        IF(.NOT.(ERFLAG.EQ.1)) GO TO 12
            IF(.NOT.(CHFLAG.EQ.0)) GO TO 1012
                T(K)=T(K)+TS
                T(2)=T(1)
                T(3)=T(2)
                T(4)=T(3)
                TS=2.0D0*TS
                CHFLAG=1
C               T(K)=T(K)+TS
                CON=IDINT((1.0D0-T(1))/TS)+3
                COUNT=1
1012        CONTINUE
            REFLAG=0
            ERFLAG=0
C
C     IF ONLY REFLAG IS SET
C
          GO TO 20
12        IF(.NOT.(REFLAG.EQ.1)) GO TO 20
                X(FAST)=-XT(FAST)
                IF(.NOT.(CHFLAG.EQ.0)) GO TO 1013
                    T(K)=T(K)+TS
                    T(2)=T(1)
                    T(3)=T(2)
                    T(4)=T(3)
                    TS=2.0D0*TS
                    CHFLAG=1
C                   T(K)=T(K)+TS
                    CON=IDINT((1.0D0-T(1))/TS)+3
                    COUNT=1
1013            CONTINUE
                REFLAG=0
                ERFLAG=0
20        CONTINUE
```

```
109       CONTINUE
          RETURN
          END
C
C*********************************************************************
C
C         SUBROUTINE TO DETECT FAST CHANGING COMPONENT OF
C         X.UPON RETURNINGTC MAIN PROGRAM,THE FLAGS ERFLAG
C         AND REFLAGWILL BE SET IF ANY SUCH COMPONENTIS
C         DETECTEC
C
C*********************************************************************
C
          SUBROUTINE DETECT(DX,REFLAG,ERFLAG)
          REAL * 8 OMAX,XFTEMP,SUM,XT,X,DX(4),TS,T
          INTEGER K,QCOUNT,FAST,RON,SIGN,REFLAG,IER,ERFLAG
     &    ,N,COUNT,CON
          COMMON /TREPAR/ XT(4),X(4),T(4),TS,FAST,CON,COUNT
     &    ,N,OMAX,QCOUNT,RON,XFTEMP,SUM,SIGN,IER
C
C       IF DX(K) > OMAX FOR 10 TS,THATCOMPONENT IS CHANGING
C       FAST.
C
          DO 105 K=1,N
               IF(.NOT.(DX(K).GE.OMAX)) GO TO 1014
                  IF(.NOT.(QCOUNT.GE.10)) GO TO 1015
                     FAST=K
                     XFTEMP=X(FAST)
                     RON=2
                     QCOUNT=0
1015              CONTINUE
                  QCOUNT=QCOUNT+1
1014           CONTINUE
105       CONTINUE
C
C       IF X(FAST)=-XT(FAST) WHEN FLIPPING,SET REFLAG
C
          IF(.NOT.(RON.NE.1)) GO TO 1016
             SUM=XT(FAST)+X(FAST)
             IF(.NOT.((DABS(X(FAST)).LE.(0.3D0*DABS
     &       (XT(FAST)))).OR.(IER.GT.0))) GO TO 1017
                IF(.NOT.((DABS(X(FAST)).LE.(0.3D0*DABS
     &          (XT(FAST)))).AND.(IER.GT.0))) GO TO 30
                   SIGN=IDINT(XT(FAST)/X(FAST))
                   IF(.NOT.(SIGN.LT.0))GO TO 1021
                      REFLAG=1
1021               CONTINUE
                   IER=0
                   ERFLAG=1
                GO TO 40
30              IF(.NOT.(DABS(X(FAST)).LE.(0.3D0*DABS
     &          (XT(FAST)))))GO TO 31
                   SIGN=IDINT(XT(FAST)/X(FAST))
                   IF(.NOT.(SIGN.LT.0))GO TO 1022
                      REFLAG=1
1022               CONTINUE
                GO TO 40
31              IF(.NOT.(IER.GT.0)) GO TO 40
                   IER=0
                   ERFLAG=1
40              CONTINUE
1017         CONTINUE
1016      CONTINUE
C
C       IF NO COMPONENT IS CHANGING FAST,BUT THERE IS ERROR
C       FROM ZSCNT
C
```

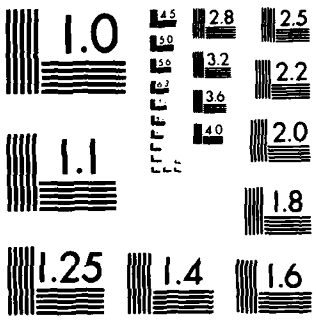
AD-A151 510 EQUILIBRIUM SOLUTIONS STABILITIES AND DYNAMICS OF LANCHESTER'S EQUATIONS W..(U) NAVAL POSTGRADUATE SCHOOL MONTEREY CA A B NING SEP 84 2/2

UNCLASSIFIED F/G 12/1 NL

END

FILMED

DTIC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

```
      IF(.NOT.((RUN.EQ.1).AND.(IER.GT.0))) GO TO 1020
         IER=0
         EFFLAG=1
1020  CONTINUE
      RETURN
      END
```

APPENDIX D

DERIVATIONS OF THE RELATIONS BETWEEN ε_x, ε_y AND STABILITY

1. Neutral Stability

In this case, $\varepsilon_x = \varepsilon_y = 0$ and equation 4.5 reduces to

$$\eta_1(x-x_{e1}) + (xy-x_{e1}y_{e1}) + \mu_1(y-y_{e1}) = 0$$

It follows that both sets of hyperbolas merge into one and all the points on the common hyperbola are equilibrium points. The lefthand side of the second condition in equation 4.4 can be manipulated as follows

$$\begin{aligned}
&(u+ay_e)(v+cx_e) - (b+ax_e)(d+cy_e) \\
&= ac\left[(\eta_1+y_e)(\mu_2+x_e) - (\mu_1+x_e)(\eta_2+y_e)\right] \\
&= ac\left[x_e(\eta_1-\eta_2) + y_e(\mu_2-\mu_1) + \eta_1\mu_2 - \eta_2\mu_1\right] \\
&= 0
\end{aligned}$$

This result implies that the constant term of the characteristics polynomial D(s), is zero ; hence one of the eigenvalues, s_1 equals to zero. Factoring out the characteristic polynomial, we get the other eigenvalue as

$$\begin{aligned}
S_2 &= -\left[(u+ay_e) + (v+cx_e)\right] \\
&= -\left[a(\eta_1+y_e) + c(\mu_2+x_e)\right] \qquad \text{(eqn D.1)}
\end{aligned}$$

For points on the first quadrant hyperbola, $x_e > -\mu_1$ and $y_e > -\eta_1$; therefore $s_2 < 0$ and neutral stability exists. Conversely, $s_2 > 0$ on the third quadrant hyperbola which is therefore unstable. The results are summarised as follows :

(1) When $\varepsilon_x = \varepsilon_y = 0$, infinitely-many equilibria exist as points on the two hyperbolas on which $\dot{x} = \dot{y} = 0$;

(2) The first quadrant hyperbola is neutrally stable;

(3) The third quadrant hyperbola is unstable.

2. Intersections in First and Third Quadrant

The proofs for the following results are given in this section :

(1) when both equilibrium points are on the first quadrant hyperbolas, one is stable and the other unstable

(2) when one equilibrium point is on the first quadrant hyperbola and the other on the third, both can be unstable or one will be stable and the other unstable.

The straight lines given by equation 4.7 are plotted on the ε_x, ε_y plane as shown in Figure D.1. It also shows the corresponding regions on the ε_x, ε_y plane as x_{e2} and y_{e2} vary.

Let (x_{e1}, y_{e1}) be the first equilibrium point on the first quadrant hyperbola. The first stability criterion in equation 4.4 is automatically satisfied since

$$(u+ay_e) + (v+cx_e)$$

$$= a(\eta_1+y_e) + c(\mu_2+x_e)$$

$$> 0 \qquad\qquad \text{for } x_{e2} > -\mu_2 \text{ and } y_{e2} > -\eta_1$$

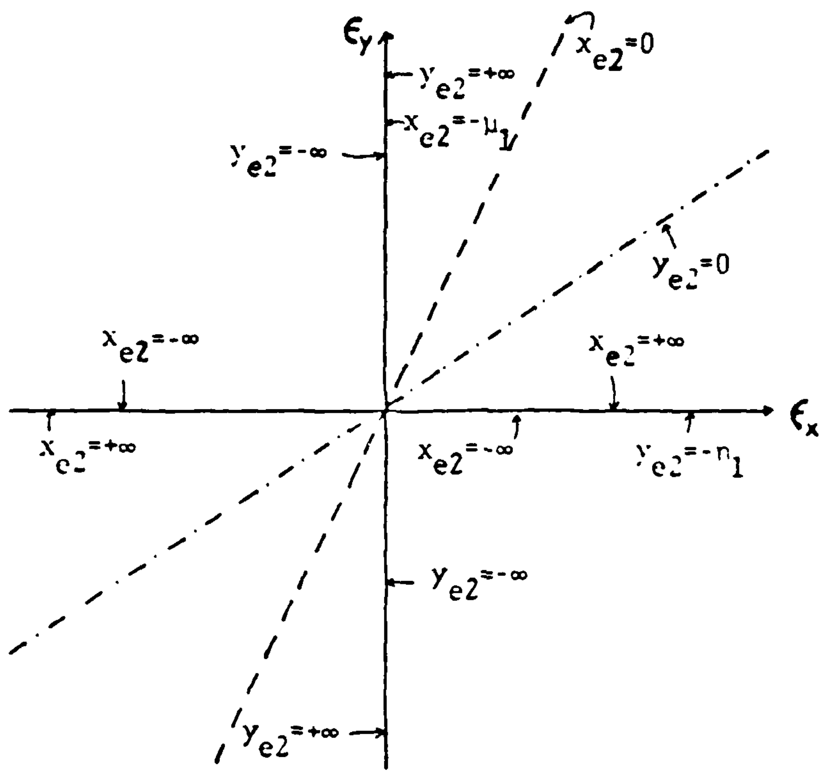


Figure D.1 Effect of Varying x_{e2} and y_{e2} on ε_x, ε_y Plane.

Consequently, the stability of (x_{e1}, y_{e1}) is solely determined by the second condition in equation 4.4. The second condition can be rewritten in the following manner :

$$
\begin{aligned}
&(u+ay_e)(v+cx_e) - (b+ax_e)(d+cy_e) \\
&= ac\left[x_e(\eta_1-\eta_2) + y_e(\mu_2-\mu_1) + \mu_1(\eta_1-\eta_2) + \eta_1(\mu_2-\mu_1)\right] \\
&= ac\left[-\varepsilon_y(x_e+\mu_1) + \varepsilon_x(\eta_1+y_e)\right] \\
&> 0
\end{aligned}
$$

or

$$\varepsilon_y \overset{s}{\underset{u}{\lessgtr}} \varepsilon_x \frac{(\eta_1+y_e)}{(\mu_1+x_e)} \qquad \text{(eqn D.2)}$$

For (x_{e1}, y_{e1}), equation D.2 represents the boundary line of stability and this line has a slope between $\eta_1/(x_{e1} + \mu_1)$ and $(y_{e1} + \eta_1)/\mu_1$. It is shown in Figure D.2.

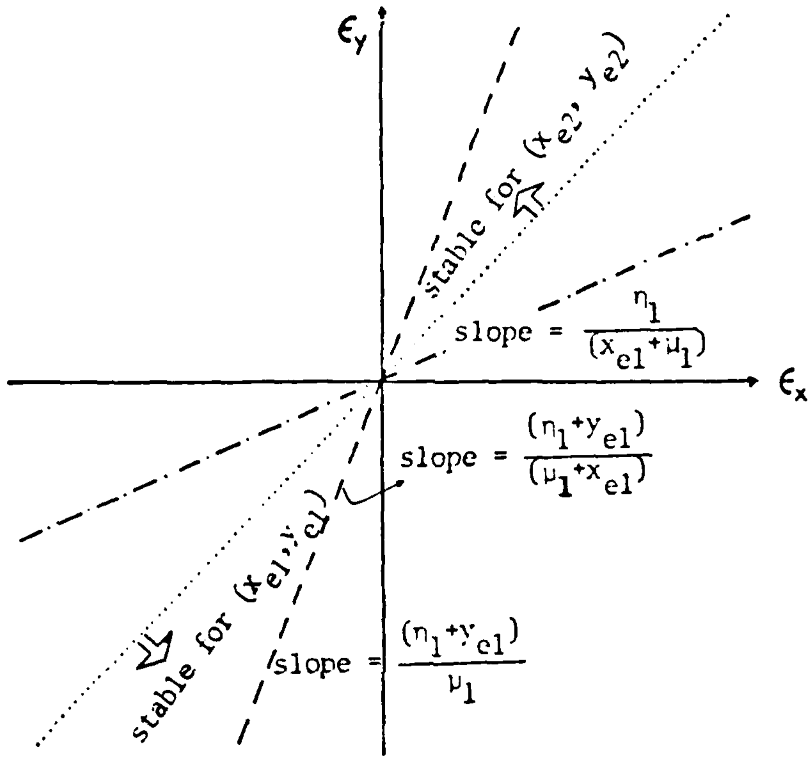


Figure D.2 Boundary line of Stability on ε_x, ε_y Plane.

To investigate the stability of (x_{e2}, y_{e2}), we substitute them into equation D.2 and obtain the condition for stability as

$$\varepsilon_y \overset{u}{\underset{s}{\lessgtr}} \varepsilon_x \frac{(\eta_1+y_{e1})}{(\mu_1+x_{e1})} \qquad \text{(eqn D.3)}$$

Equations D.2 and D.3 are only different in the directions of inequality. This means that on one side of the line $\varepsilon_y = \varepsilon_x(\eta_1 + y_{e1})/(\mu_1 + x_{e1})$, one equilibrium point is stable and the other is unstable. On the other side of the line, the opposite conditions exists.

Note that (x_{e2}, y_{e2}) may or may not satisfy the first condition of equation 4.4. This condition is

$$a(\eta_1 + y_{e2}) + c(\mu_1 + \varepsilon_x + x_{e2}) > 0 \qquad \text{(eqn D.4)}$$

From Figure D.1, equation D.4 and earlier results the following deductions can be made

(1) In the first quadrant of the ε_x, ε_y plane, $\varepsilon_x > 0$, $x_{e2} > -\mu_1$, $y_{e2} > -\eta_1$; hence equation D.4 is satisfied. The region labelled  in Figure D.3 is stable for (x_{e2}, y_{e2}) but unstable for (x_{e1}, y_{e1});

(2) In the third quadrant of the ε_x, ε_y plane but between the two lines where $x_{e2} > 0$ and $y_{e2} > -\eta_1$, equation D.4 is again satisfied. The region labelled  is also stable for (X_{e2}, Y_{e2}) but unstable for (x_{e1}, y_{e1});

(3) In the second quadrant of the $\varepsilon_x, \varepsilon_y$ plane, $x_{e2} < -\mu_1$, $y_{e2} < -\eta_1$ and $\varepsilon_x < 0$, equation D.4 is <u>not</u> satisfied ; so both (x_{e1}, y_{e1}) and (x_{e2}, y_{e2}) are unstable. The region is labelled  in Figure D.3;

(4) In the fourth quadrant of the ε_x, ε_y plane, equation D.3 is not satisfied ; so (x_{e2}, y_{e2}) is unstable but (x_{e1}, y_{e1}) is stable. This region is labelled  in Figure D.3;

(5) In the region labelled , equation D.3 is not satisfied ; so (x_{e2}, y_{e2}) is unstable but (x_{e1}, y_{e1}) is stable.

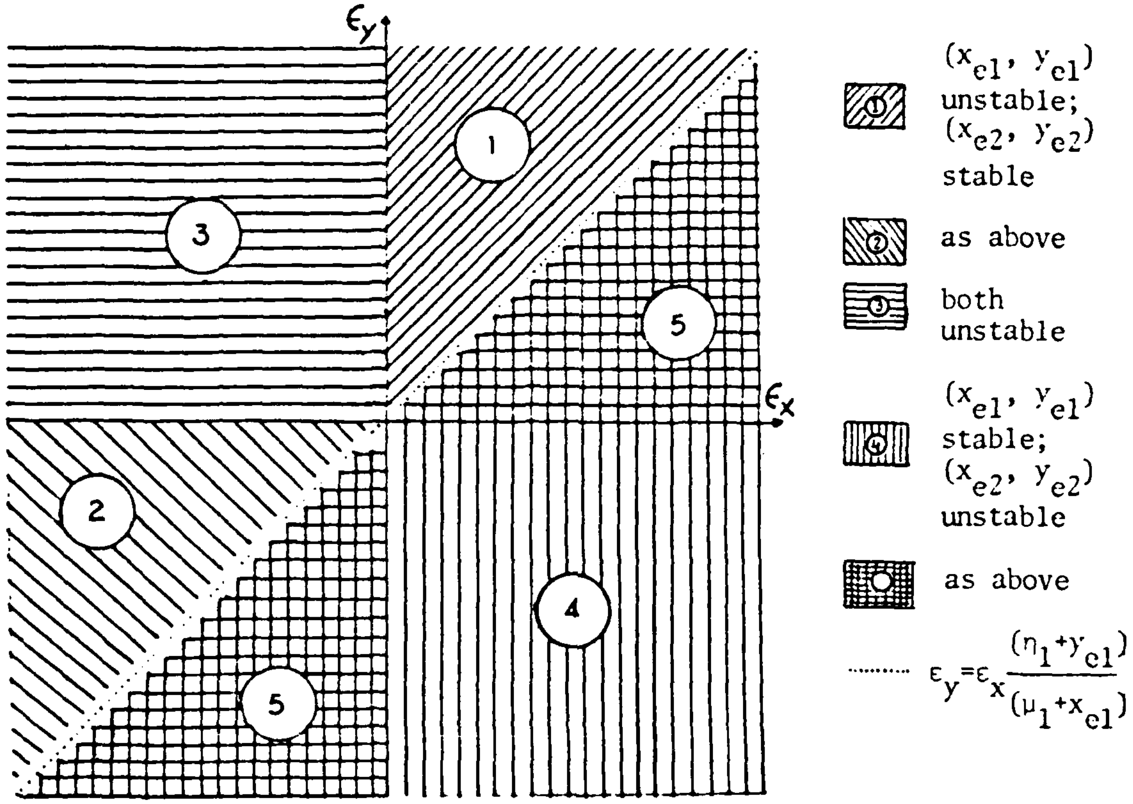


Figure D.3 Regions in ε_x, ε_y Plane.

By carefully noting the signs of (x_{e2}, y_{e2}) and the various regions in Figure D.3, all the previous deductions can be combined ; we conclude that :

(1) When both equilibrium points are on the first quadrant hyperbola, one is stable and the other unstable;

(2) When one equilibrium point is on the first quadrant hyperbola and the other on the third, both can be unstable or one will be stable and the other unstable.

3. Two Equilibria in the Third Quadrant

The simplest way to show that both equilibrium points in the third quadrant are unstable is to refer to Figure D.4. Clearly, we must have $x < -v/c$ and $y < -u/a$ for both equilibrium points. In that case, the first stability criterion in equation 4.4 is not satisfied since

$$(u+ay_e) + (v+cx_e) < 0$$

Therefore, both equilibria are unstable.

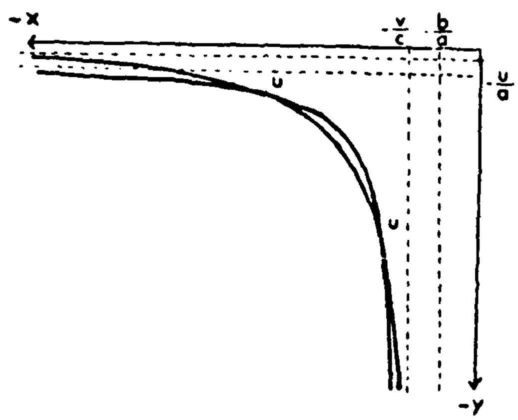


Figure D.4 Two equilibria in the Third Quadrant.

4. Repeated Equilibria

This case corresponds to operating exactly on the $\epsilon_y = \epsilon_x(y_{e1} + \eta_1)/(x_{e1} + \mu_1)$ on the ϵ_x, ϵ_y plane. The proof is obtained by substituting $x_{e2} = x_{e1}$, $y_{e2} = y_{e1}$ into equation 4.6 and solving for ϵ_y.

$$y_{e2} = y_{e1} = \frac{\varepsilon_y}{\varepsilon_x}(x_{e1}+\mu_1) - \eta_1$$

(eqn D.5)

$$x_{e2} = x_{e1} = \frac{\varepsilon_x}{\varepsilon_y}(y_{e1}+\eta_1) - \mu_1$$

Rewriting equation D.5, we have

$$\varepsilon_x y_{e1} = \varepsilon_y x_{e1} + \mu_1 \varepsilon_y - \eta_1 \varepsilon_x$$

$$\varepsilon_y x_{e1} = \varepsilon_x y_{e1} + \eta_1 \varepsilon_x - \mu_1 \varepsilon_y$$

subtracting one from the other,

$$2\varepsilon_x y_{e1} = 2\varepsilon_y x_{e1} + 2\mu_1 \varepsilon_y - 2\eta_1 \varepsilon_x$$

$$\varepsilon_y = \varepsilon_x \frac{(y_{e1}+\eta_1)}{(x_{e1}+\mu_1)} \qquad \text{(eqn D.6)}$$

Thus we have shown that the case of repeated equilibria corresponds to points on the straight line indicated in Figure D.3.

Equation D.6 can be also obtained by setting the second stability criterion equal to zero (see equation D.2). That is equilavent to saying one of the eigenvalues, s_1 is equal to zero. The sign of s_2 is determined by considering the first stability criterion. Following the same argument as in neutrally stable case, we can prove that repeated equilibria on the first quadrant hyperbolas are neutrally stable. On the other hand, repeated equilibria on the third quadrant hyperbolas are unstable.

APPENDIX E

RELATION BETWEEN STABILITY AND ε_x, ε_y PLANE: VERIFICATIONS

Some representative points on the ε_x, ε_y plane are chosen and the corresponding stablities of the equilibria calculated. The points chosen are marked F, G, H, M, N, P, Q, T and W in Figure E.1.

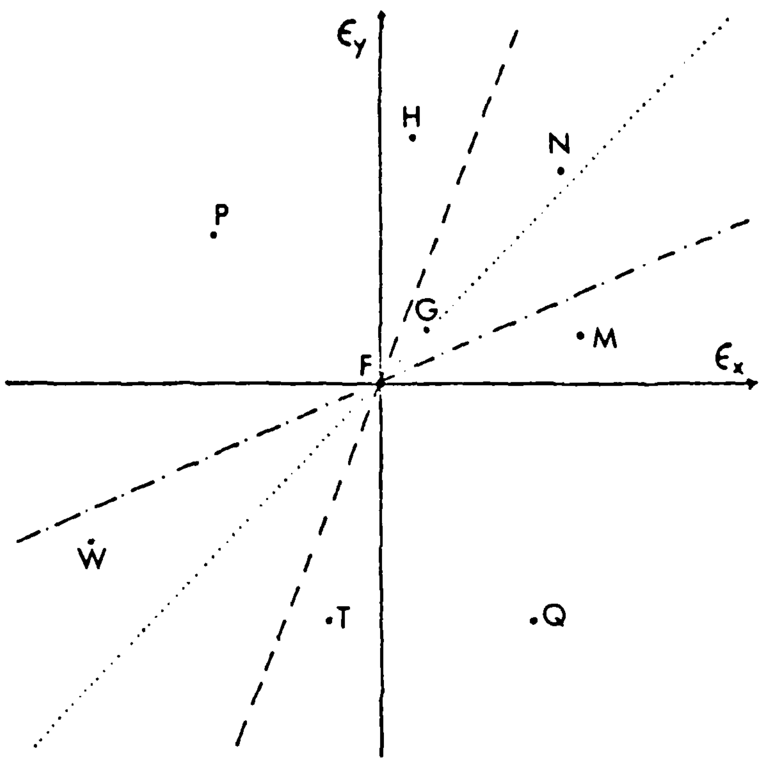


Figure E.1 Experimental Verifications.

1. Procedure

Having chosen the points on the ε_x, ε_y plane, we have to work backward to obtain a, b, c, d, u and v. Suitable (x_{e1}, y_{e1}) are then chosen, followed by a calculation of r, s, x_{e2} and y_{e2}. (x_{e2}, y_{e2}) can be obtained directly from equation 4.6. From all these parameters, the eigenvalues can be calculated and results compared with theory.

2. Results

The results are tabulated in Table VII. They agree with the theoretical results given in Figure E.1.

TABLE VII

Results of Experimental Verification

Point on the ε_x, ε_y plane	x_{e1}	y_{e1}	Maximum eigenvalue for (x_{e1}, y_{e1})	x_{e2}	y_{e2}	Maximum eigenvalue for (x_{e2}, y_{e2})	ε_x	ε_y	Remarks
M	1.0	1.0	-0.04	2.4	-0.4	0.039	0.2	-1.2	
N	1.5	1.0	0.11	1.27	1.19	-0.01	0.4	0.34	
H	1.3	0.8	0.12	-0.13	4.13	-0.13	0.3	0.7	
P	1.0	3.0	0.38	-11.0	-9.0	4.93	-1.0	1.0	
W	1.2	3.0	0.17	9.23	0.05	-0.07	-0.3	-0.1	
T	2.0	1.0	-0.16	-0.08	6.2	0.09	-0.2	-0.5	
Q	1.0	1.0	-0.28	-2.4	-4.7	3.25	0.3	-0.5	
G	1.0	1.0	0.0	1.0	1.0	0.0	1.0	1.0	Repeat equilibria
F	-	-	-	-	-	-	-	-	see Figure 4.2(d)

```
C
C                       APPENDIX F
C          PROGRAM TO PLOT TRAJECTORIES IN 1*1 SYSTEMS
C
C
C
C
C*********************************************************************
C                    PROGRAM I1PL FORTRAN
C       THIS PROGRAM PLOTS THE TRAJECTORIES OF A 1*1 SYSTEM.
C       IT CALLS AN IMSL ROUTINE,DVERK FOR INTEGRATION.
C       BEFORE EXECUTION,DETERMINE THE SCALE OF THE
C       PLOT AND NUMBER OF CURVES REQUIRED.
C       THE VALUE , "STEP" IS DETERMINED BY THE RANGE OVER
C       WHICH WE WANT TO SET THE INITIAL POINTS.CHECK A,B,
C       C,D,AND U(2).FIX EITHER X(1) OR X(2) AND VARY THE
C       OTHER;THE ONE FIXED HAS TO BE RESET AFTER EACH
C       CURVE HAS BEEN DRAWN.IT IS RESET IN THE LAST
C       ASSIGNMENT STATEMENTIN THE DO LOOP.TO EXECUTE,
C       ENTER "I1PL".THE EXEC FILE,"I1PL EXEC"
C       MUST BE IN THE DISK.IT MUST BE EXECUTED ON A
C       TERMINAL ATTACHED TO THE TEKTRONIX 618.
C*********************************************************************
C
C
C*********************************************************************
C       VARIABLE DEFINITIONS
C*********************************************************************
C       X=VECTOR OF LENGTH 2 CONTAINING THE UNSTABLE
C       EQUILIBRIUM POINT.
C       A,B,C,D,U,R=ATTRITION COEFFIENTS.
C       CAPX,CAPY,CAP1X,CAP1Y=ARRAYS USED TO STORE X"S
C       FOR PLOTTING PURPOSES.
C       DEL= INTEGRATION STEP SIZE.
C       IND,CC,W,TOL,TEND=PARAMETRS REQUIRED BY IMSL ROUTINE
C       DVERK
C       IER=ERROR MESSAGE NUMBER FROM DVERK
C       DIR=CONSTANT FACTOR DETERMINING THE DIRECTION
C       OF PERTURBATIONS FROM THE UNSTABLE POINT
C       FCN1=EXTERNAL FUNCTION REQUIRED BY DVERK
C       NSTEP=NUMBER OF CURVES TO PLOT;ALSO # OF STARTING
C       STEP=INTERVAL BETWEEN DIFFERENTINITIAL POINTS.
C       INTEGRATION POINTS.
C       K,KM1=COUNTERS FOR PLOTTING ROUTINE
C*********************************************************************
C
        INTEGER N,IND,NW,IER,K,NPOINT,KM1,KK,JJ,K1,K2,KM2
        INTEGER N,IND,NW,IER,K,NPOINT,KM1,KK,J,NSTEP
        REAL * 4 X(2),CC(24),W(2,9),T,TOL,TEND,DEL,R(2),A,B,
       &C,D,U(2),CAPX(1000),CAPY(1000),XA(1000),XB(1000),STEP
        EXTERNAL FCN1
        COMMON R,A,B,C,D,U
        DATA NW/2/,N/2/,T/0.0/,TOL/0.0010/,IND/1/,IER/0/
C
C       ENTER ATTRITION COEFFICIENTS HERE
C
        A=0.6000
        B=0.3000
        C=1.0000
        D=5.00
        U(1)=0.60
        U(2)=1.00000
C
C       ENTER ONE EQUILIBRIUM POINT HERE FOR THE CALCULATION
C       OF R VECTOR
C
        X(1)=5.00000
```

```
      X(2)=4.0CCC0
      R(1)=X(1)*(U(1)+A*X(2))+B*X(2)
      R(2)=X(2)*(U(2)+C*X(1))+D*X(1)
C
C     ENTER R DIRECTLY HERE IF THEY ARE KNOWN;REMOVE
C     THE COMMENT CHARACTERS.
C
C     R(1)=*****
C     R(2)=*****
C
C*****************************************************************
C     SET PLOTTING PARAMETERS
C*****************************************************************
C
      CALL TEK618
C     CALL COMPRS
C     CALL VRSTEC(0,0,0)
      CALL PAGE(14.7,11.5)
      CALL NOBRDR
      CALL CROSS
      CALL BLOWUP(0.5)
      CALL AREA2D(13.7,9.50)
      CALL XNAME('TOTAL X FORCES$',100)
      CALL YNAME('                 TOTAL Y FORCES$',100)
      CALL FRAME
C
C     INSERT HEADINGS WITHIN QUOTES;REMOVE COMMENT
C     CHARACTER
C
C     CALL HEADIN('**********************************$'
C    &,100,1.,4)
C     CALL HEADIN('********************************$'
C    &,100,1.,4)
C     CALL HEADIN('**********************$',100
C    &,100,1.,4)
C     CALL HEADIN('********************************$'
C    &,100,1.,4)
      CALL GRAF(-3.000,'SCALE',10.0,-6.00,'SCALE',1C.00)
C
C
C     INSERT INITIAL CONDITIONS FOR INTEGRATION
C
C
      STEP=0.5CCC
      NSTEP=10
      X(1)=1.0CCC
      DO 100 J=1,NSTEP
         X(2)=-5.00C+FLOAT(J)*STEP
         K=1
         DEL=0.01C
         NPOINT=300
5        IF(.NOT.((IER.LE.0).AND.(IND.GE.0).AND.(K.LE.
     &   NPOINT))) GO TO 6
            TEND=FLOAT(K)*DEL
            CALL DVERK(N,FCN1,T,X,TEND,TOL,IND,CC,NW,W,IER)
C           XA(K)=X(1)
C           XB(K)=X(2)
            WRITE(6,999)IER
999         FORMAT('0',3X,'IER=',I3)
            CAPX(K)=X(1)
            CAPY(K)=X(2)
            K=K+1
            GO TO 5
6        CONTINUE
         KM1=K-1
         CALL CURVE(CAPX,CAPY,KM1,3)
C
```

```
C         RESET INTEGRATION PARAMETERS AFTER EACH CURVE.
C
          T=0.0
          IER=0
          IND=1
          X(1)=1.0000
100     CONTINUE
        CALL ENDFL(0)
        CALL DONEPL
        STOP
        END
C
C***********************************************************************
C       SUBROUTINE FCN1 REQUIRED BY OVERK
C***********************************************************************
C
        SUBROUTINE FCN1(N,T,X,XPRI)
          INTEGER N,I,J
          REAL * 4 X(N),XPRI(N),T,R(2),A,B,C,D,U(2)
          COMMON R,A,B,C,D,U
          XPRI(1)=-X(1)*(U(1)+A*X(2))+R(1)-B*X(2)
          XPRI(2)=-X(2)*(U(2)+C*X(1))+R(2)-D*X(1)
          RETURN
        END
```

```
C
C                        APPENDIX G
C             PROGRAM TO PLOT BOUNDARY CURVE
C
C
C
C*****************************************************************
C                    PROGRAM BK1 FORTRAN
C       THIS PROGRAM DETERMINES THE BOUNDARY SEPERATING THE
C       DOMAINS OF ATTRACTION IN A 1*1 PROBLEM.BACKWARD
C       INTEGRATION (NEGATIVE TIME) IS USED STARTING
C       FROM AN UNSTABLE EQUILIBRIUM POINT.
C       BEFORE EXECUTION CHECK A,B,C,D,U(2),AND X(2).
C       TO EXECUTE ENTER BK1.THE EXEC FILE,"BK1 EXEC"
C       MUST BE IN THE DISK.IT MUST BE EXECUTED ON A
C       TERMINAL ATTACHED TO THE TEKTRONIX 618.
C*****************************************************************
C
C
C*****************************************************************
C       VARIABLE DEFINITIONS
C*****************************************************************
C       X=VECTOR OF LENGTH 2 CONTAINING THE UNSTABLE
C       EQUILIBRIUM POINT.
C       A,B,C,D,U,R=ATTRITION COEEFIENTS.
C       CAPX,CAPY,CAP1X,CAP1Y=ARRAYS USED TO STORE X"S
C       FOR PLOTTING PURPOSES.
C       DEL= INTEGRATION STEP SIZE.
C       IND,CC,W,TOL,TEND=PARAMETRS REQUIRED BY IMSL ROUTINE
C       DVERK
C       IER= ERROR MESSAGE NUMBER FROM DVERK
C       DIR=CONSTANT FACTOR DETERMINING THE DIRECTION
C       OF PERTURBATIONS FROM THE UNSTABLE POINT
C       FCN1=EXTERNAL FUNCTION REQUIRED BY DVERK
C       E=CONSTANTS EQUAL 1.0;FOR SETTING DIRECTION OF
C       OF PERTURBATION TOGETHER WITH DIR
C       K,K1,K2=COUNTERS FOR PLOTTING ROUTINE
C*****************************************************************
C
        INTEGER N,IND,NW,IER,K,NPOINT,KM1,KK,JJ,K1,K2,KM2
        REAL * 4 X(2),CC(24),W(2,9),T,TOL,TEND,DEL,E(2),R,A,B
       &,C,D,U,CAPX(1000),CAPY(1000),
       &CAP1X(1000),CAP1Y(1000),DIR
        EXTERNAL FCN1
        COMMON R(2)
        COMMON /PARA/ A,B,C,D,U(2)
        DATA NW/2/,N/2/,T/0.0/,TOL/0.0010/,IND/1/,IER/0/
       &,E/+1.000,+1.000/
        A=0.7
        B=0.4
        C=1.0
        D=0.6
        U(1)=0.15
        U(2)=0.2
C
C*****************************************************************
C       CALCULATE R,S VECTOR USING DATA FROM LP PROGRAM
C*****************************************************************
C
        X(1)=6.70
        X(2)=3.00
        R(1)=X(1)*(U(1)+A*X(2))+B*X(2)
        R(2)=X(2)*(U(2)+C*X(1))+D*X(1)
        WRITE(6,9998)(I,X(I),I,R(I),I=1,N)
9998    FORMAT('0',3X,'X(',I3,')=',F12.4,4X,'R(',I3,')=',F12.4
C
C*****************************************************************
```

```
C       INSERT INITIAL CONDITIONS FOR INTEGRATION
C*****************************************************************
C
        K1=1
        K2=1
        DO 100 JJ=1,2
C
C           FOR JJ=1 & 2,BACKWARD INTEGRATE ALONG BRANCH # 1,2
C           RESPECTIVELY
C
            IF(.NOT.(JJ.EQ.1))GO TO 50
               DIR=0.03
               X(1)=X(1)+DIR*E(1)
               X(2)=X(2)+DIR*E(2)
               GO TO 60
50          CONTINUE
               DIR=-0.03
               X(1)=6.70+DIR*E(1)
               X(2)=3.000+DIR*E(2)
60          CONTINUE
            K=1
            T=0.0
            DEL=0.100
            IF(JJ.EQ.2) DEL=0.100
            NPOINT=200
            IND=1
C
C*****************************************************************
C
C           WHILE NO ERROR AND NUMBER OF INTEGRATION STEPS
C           LESS THAN NPOINT
C
C*****************************************************************
C
5           IF(.NOT.((IER.LE.0).AND.(IND.GE.0).AND.(K.LE.
     &      NPOINT))) GO TO 6
               TEND=-(FLOAT(K)*DEL)
C
C              CALL INTEGRATION ROUTINE
C
               CALL DVERK(N,FCN1,T,X,TEND,TOL,IND,CC,NW,W,IER)
C
C              PRINT ERROR MESSAGE IN ANY
C
               WRITE(6,999)IER
999            FORMAT('0',3X,'IER=',I3)
C
C              STORE X(1),X(2) FOR THE TWO BRANCHES OF THE
C              BOUNDARY IN TWO ARRAYS
C
               IF(.NOT.(JJ.EQ.1))GO TO 501
                  CAPX(K)=X(1)
                  CAPY(K)=X(2)
                   K1=K1+1
                   GO TO 601
501            CONTINUE
                  CAP1X(K)=X(1)
                  CAP1Y(K)=X(2)
                  K2=K2+1
601            CONTINUE
               K=K+1
               GO TO 5
6           CONTINUE
100     CONTINUE
C
C       SET NUMBER OF PLOTTED POINTS TO ONE LESS THAN
C       THE NUMBER OF DATA POINTS
```

```
C
      KM1=K1-1
      KM2=K2-1
C
C****************************************************************
C     COMMENCE PLOTTING THE BOUNDARY
C****************************************************************
C
      CALL TEK618
C     CALL COMPRS
C     CALL VRSTEC(0,0,0)
      CALL PAGE(14.7,11.5)
      CALL NOBRDR
      CALL CROSS
      CALL BLOWUP(0.55)
      CALL AREA2D(13.7,9.0)
      CALL XNAME('                    TOTAL X FORCES$',100)
      CALL YNAME('                   TOTAL Y FORCES$',100)
C
C     INSERT HEADINGS WITHIN QUOTES;REMOVE COMMENT CHARACTER
C
C     CALL HEADIN('*********************************$'
C    &,100,1.,4)
C     CALL HEADIN('*********************************$'
C    &,100,1.,4)
C     CALL HEADIN('**********************$',100
C    &,100,1.,4)
C     CALL HEADIN('*********************************$'
C    &,100,1.,4)
      CALL GRAF(6.000,'SCALE',7.000,0.400,'SCALE',4.000)
      CALL CURVE(CAPX,CAPY,KM1,2)
      CALL CURVE(CAP1X,CAP1Y,KM2,2)
      CALL ENDPL(0)
      CALL DONEPL
      STOP
      END
C
C****************************************************************
C     SUBPROGRAM CALL BY INTEGRATION ROUTINE,DVERK
C****************************************************************
C
      SUBROUTINE FCN1(N,T,X,XPRI)
         INTEGER N,I,J
         REAL * 4 X(N),XPRI(N),T,R,A,B,C,D,U
         COMMON R(2)
         COMMON /PARA/ A,B,C,D,U(2)
         XPRI(1)=-X(1)*(U(1)+A*X(2))+R(1)-B*X(2)
         XPRI(2)=-X(2)*(U(2)+C*X(1))+R(2)-D*X(1)
         RETURN
      END
```

```
C                           APPENDIX H
C               PROGRAM TO OBTAIN PAYOFF MATRIX AND
C                   OPTIMUM MIXED STRATEGIES
C
C
C
C*****************************************************************
C           THIS PROGRAM COMPUTES THE PAYOFF MATRIX AND
C           OPTIMUM MIXED STRATEGIES FOR X AND Y.THE ALGORITHM
C           IS BASED ON THE THEORY GIVEN IN CHAPTER 5.
C           THE PAYOFF FUNCTIONS ARE COMPUTED FOR 60*60 X,Y
C           BUT ONLY 15*15 SAMPLED MATRIX IS PRINTED.
C           THE OUTPUT ALSO INCLUDES MATRIX FOR LX,LY,AND
C           FINTIM.
C           BY CHANGING ALAMDA TO LESS THAN 1,THE
C           OBJECTIVE FUNCTION CAN BE MADE TO EMPHASIZE
C           MORE OF FINTIM IN ACCORDANCE WITH:
C              A(X,Y)=ALAMDA*(LY-LX)+(1-ALAMDA)*FINTIM
C           BEFORE EXECUTION,CHECK ATTRITION COEFFIENTS,
C           QX,QY,ALAMDA,X1S,X2S,STEP1,STEP2,CARDS
C           LABELLED C1,C2 AND SCALE IN THE PLOTTING
C           ROUTINE.PERTURBATIONS ARE GIVEN BY CARDS
C           LABELLED C3 AND C4.
C           TO EXECUTE,YOU NEED THE EXEC FILE "LOSCP EXEC"
C           AND A TERMINAL CONNECTED WITH A TEK618.ENTER
C           "LOSQF" WHEN READY.
C*****************************************************************
C
C*****************************************************************
C           VARIABLE DEFINITIONS
C*****************************************************************
C           X(2)=ARRAY CONTAINING X AND Y
C           X1S,X2S=VARIBLES USED TO SET THE PARTITIONS
C                   OF X AND Y VALUES TO CALCULATE
C                   A(X,Y).THERE ARE 60 INTERVALS FOR X AND
C                   Y IN THE RANGE 0.20 AND 0.750.
C           STEP1,STEP2=TO OBTAIN THE CORRECT STEP INTERVAL
C                       AS DESCRIBED ABOVE
C           XPRIN1,XPRIN2=VARIABLES USED TO SAFEKEEP INITIAL
C                         VALUES OF X AND Y
C           TX,TY,R,S=AS DEFINED IN CHAPTER 5
C           RX,RY=(QX-X),(QY-Y) RESPECTIVELY
C           XLOSS,YLOSS=LX,LY IN CHAP. 5
C           CC,NW,W,TOL,TEND,DEL,IND=PARAMETERS REQUIRED BY
C                                    IMSL ROUTINE,DVERK.
C           IEC=ERROR CODE FROM DVERK.
C           ZMAT=MATRIX USED FOR PLOTTING SURFACE A(X,Y).
C           RATLOS=PAYOFF MATRIX
C           FINTIM=FINISH TIME MATRIX
C           RATAVE=15*15 PAYOFF MATRIX FOR OUTPUT
C           XLOSPR=LX MATRIX FOR OUTPUT
C           YLOSPR=LY MATRIX FOR OUTPUT
C           FTPR=FINTIM MATRIX FOR OUTPUT
C           ROWMIN=ROW MINIMUM IN PAYOFF MATRIX
C           COLMAX=COLUMN MAXIMUM IN PAYOFF MATRIX
C           AXMIN=MAXIMUM OF THE THE ROW MINIMUM
C           XINMAX=MINIMUM OF THE COLUMN MAXIMUM
C           RHS=ARRAY CONTAINING RHS OF CONSTRAINTS
C           OBCOEF=ARRAY CONTAINING COEFFICIENTS OF OBJECTIVE
C                  FUNCTION IN LP.
C           ALAMDA=NUMBER BETWEEN 0 AND 1 TO DETERMINE
C                  RELATIVE EMPHASIS OF LOSS AND FINTIM
C                  IN A(X,Y)
C           SMMIN=SMALLEST VALUE OF ROW MINIMUM
C           PAY=15*15 MATRIX USED TO SAFEKEEP RATAVE AND
C               COMPUTE THE OPTIMIZED STAREGIES
```

```
C          NS1,NS2=DIMENSION OF PAYOFF MATRIX,EQUALS 60.
C          NOGO=FLAG TO INDICATE END OF INTEGRATION.
C          JJ,KK=INDICES FOR MATRICES
C          COUNT=COUNTER FOR INTEGRATION
C          KKM1=NUMBER OF POINTS PLOTTED
C          FCN1,2,3,4=SUBROUTINES REQUIRED BY DVERK;THEY
C                        CONTAIN 1*1 EQUATIONS FOR THE VARIOUS
C                        STAGES FOR THE BATTLE
C          T=TIME
C          XA,YA,ZA=ARRAYS USED FOR STORING AND PLOTTING
C                   XPRIN1,XPRIN2,RATLOS AS DEFINED ABOVE
C          ZCOUNT=COUNTER FOR XA,YA
C          JC=COUNTER FOR ZA
C          TIME1=T1 IN CHAP.5
C          T1=T2 IN CHAP.5
C          NPLOT=NUMBER OF PLOTS REQUIRED
C*****************************************************************
           REAL X1S,X2S,X(2),R,S,RX,RY,TX,TY,CX,CY,A,C,U,V,
      &    W(2,9),T,TOL,TEND,DEL,XA(3650),YA(3650),RATIO
      &    ,RATLOS(60,60),XPRIN1,XPRIN2,XT1,XT2,ROTRAT
      &    ,ZMAT(60,60),ROWMIN(60),COLMAX(60),AXMIN,XINMAX
           REAL FINAL,FINTIM(60,60),TIME1,RATAVE(20,20)
      &    ,SMMIN,PAY(17,32),TEMPO,RHS(15),OBCOEF(15),ALAMDA
      &    ,XLOS(60,60),YLOS(60,60),FTPR(20,20),
      &    YLOSPR(20,20),CC(24),T1,STEP1,STEP2,YLOSS,XLOSS
           REAL YLOXL,ZA(3650),XLOSPR(20,20)
           INTEGER IND,NW,IEO,K,COUNT,NSTEP,NS1,NS2,J,NOGO
      &    ,ZCOUNT,NX,NY,NPOINT,JJ,KK,JK,JC,KKM1
           COMMON /PARAM/ A,C,U,V,S,R,B,D
           EXTERNAL FCN1
           EXTERNAL FCN2
           EXTERNAL FCN3
           EXTERNAL FCN4
C
C*****************************************************************
C          SET PARAMETERS AND COUNTERS
C*****************************************************************
C
           A=0.70
           C=1.00
           U=0.15
           V=0.200
           B=0.4000
           D=0.600
           ALAMDA=1.00
           FINAL=0.10
           NW=2
           N=2
           T=0.0
           TOL=0.001
           IND=1
           DEL=0.01
           COUNT=1
           QX=10.00
           QY=7.00
           X1S=3.0
           X2S=3.00
           JC=0
           NOGO=0
           NS1=60
           NS2=60
           NS=NS1-1
           ZCOUNT=0
           NPOINT=NS1*NS2
C
C*****************************************************************
C          START COMPUTING 60*60 PAYOFF FUNCTIONS
```

```
C**************************************************************
C
          DC 100 J=1,NS1
             STEP1=(FLOAT(J-NS1))/39.33
             DO 101 K=1,NS2
                JC=JC+1
                T=0.0
                COUNT=1
                IND=1
                STEP2=(FLCAT(K-NS2))/54.428
C    CARD C1:
                X(1)=X1S*(2.333+STEP1)
C    CARD C2:
                X(2)=X2S*(1.917+STEP2)
                XPRIN1=X(1)
                XPRIN2=X(2)
                XA(JC)=XPRIN1
                YA(JC)=XPRIN2
                R=X(1)*(U+A*X(2))+B*X(2)
                S=X(2)*(V+C*X(1))+D*X(1)
                RX=QX-X(1)
                RY=QY-X(2)
C    CARD C3:
                X(1)=XPRIN1-0.01
C    CARD C4:
                X(2)=XPRIN2-0.05
                TX=RX/R
                TY=RY/S
C
C**************************************************************
C         STAGE 1
C**************************************************************
C
                TEND=0.0
                TIME1=AMIN1(TX,TY)
5000            IF(.NOT.((TEND.LE.TIME1).AND.(NOGO.EC.0)))
     &          GO TO 6000
                   TEND=FLOAT(CCUNT)*DEL
                   CALL DVERK(N,FCN4,T,X,TEND,TOL,IND,CC
     &             ,NW,W,IED)
                   IF(.NOT.((X(1).LE.(FINAL*CX)).OR.(X(2)
     &             .LE.(FINAL*CY)))) GO TO 1028
                      IF(.NOT.(X(1).LE.(FINAL*QX)))GC TO 52
                         XLOSS=(1.-FINAL)*QX
                         YLOSS=T*S+XPRIN2-X(2)
                      GO TO 62
52                    CONTINUE
                         XLOSS=T*R+XPRIN1-X(1)
                         YLCSS=(1.-FINAL)*QY
62                    CONTINUE
                      NOGO=1
                      XLOS(J,K)=XLOSS
                      YLOS(J,K)=YLOSS
                      RATLOS(J,K)=YLOSS-XLOSS
                      FINTIM(J,K)=T
                      IF(.NOT.(RATLOS(J,K).LT.0.0))GO TO 153
                         SIGN=1.0
                         GO TO 163
153                     CONTINUE
                         SIGN=-1.0
163                     CONTINUE
                      RATLOS(J,K)=ALAMDA*RATLOS(J,K)+(1.
     &                -ALAMCA)*SIGN*FINTIM(J,K)
1028               CONTINUE
                   COUNT=COUNT+1
                   IED=0
                   GO TO 5000
```

```
6000          CONTINUE
C
C             CALCULATE T1
C
              IF(.NOT.(TX.LT.TY))GO TO 50
                 T1=TY-TX
                 GO TO 60
50            CONTINUE
                 T1=TX-TY
60            CONTINUE
C
C**********************************************************************
C             STAGE 2
C**********************************************************************
C
              TEND=0.0
              T=0.0
              COUNT=1
              IND=1
5             IF(.NOT.((TEND.LE.T1).AND.(NOGO.EQ.0)))
     &        GO TO 6
                 TEND=FLOAT(COUNT)*DEL
                 IF(.NOT.(TX.LT.TY))GO TO 51
                    CALL DVERK(N,FCN1,T,X,TEND,TOL,IND
     &              ,CC,NW,W,IED)
                    IF(.NOT.((X(1).LE.(FINAL*CX)).OR.
     &              (X(2).LE.(FINAL*QY)))) GO TO 1011
                       XLOSS=CX-X(1)
                       YLOSS=(T*S)+XPRIN2-X(2)+(TX*S)
                       RATLOS(J,K)=YLOSS-XLOSS
                       XLOS(J,K)=XLOSS
                       YLOS(J,K)=YLOSS
                       FINTIM(J,K)=TX+T
                       IF(.NOT.(RATLOS(J,K).LT.0.0))GO TO
     &                 154
                          SIGN=1.0
                          GO TO 164
154                    CONTINUE
                          SIGN=-1.0
164                    CONTINUE
                       RATLOS(J,K)=ALAMDA*RATLOS(J,K)+
     &                 (1.-ALAMDA)*SIGN*FINTIM(J,K)
                       NOGO=1
1011                   CONTINUE
                       GO TO 61
51               CONTINUE
                    CALL DVERK(N,FCN2,T,X,TEND,TOL,IND
     &              ,CC,NW,W,IED)
                    IF(.NOT.((X(1).LE.(FINAL*CX)).OR.
     &              (X(2).LE.(FINAL*QY)))) GO TO 1013
                       XLOSS=(T*R)+XPRIN1-X(1)+(TY*R)
                       YLOSS=CY-X(2)
                       RATLOS(J,K)=YLOSS-XLOSS
                       XLOS(J,K)=XLOSS
                       YLOS(J,K)=YLOSS
                       FINTIM(J,K)=TY+T
                       IF(.NOT.(RATLOS(J,K).LT.0.0))GO
     &                 TO 155
                          SIGN=1.0
                          GO TO 165
155                    CONTINUE
                          SIGN=-1.0
165                    CONTINUE
                       RATLOS(J,K)=ALAMDA*RATLOS(J,K)+
     &                 (1.-ALAMDA)*SIGN*FINTIM(J,K)
                       NOGO=1
1013                CONTINUE
```

```
61                      CONTINUE
                        COUNT=COUNT+1
                        IED=0
                        GO TO 5
6                   CONTINUE
C
C**********************************************************************
C          STAGE 3
C**********************************************************************
C
                    T=0.0
                    COUNT=1
                    IND=1
500                 IF(.NOT.(NOGO.EQ.0))GO TO 600
1005                    CONTINUE
                            TEND=FLOAT(COUNT)*DEL
                            CALL DVERK(N,FCN3,T,X,TEND,TOL,IND
     &                      ,CC,NW,W,IED)
                            IF(.NOT.((X(1).LE.(FINAL*CX)).OR.
     &                      (X(2).LE.(FINAL*QY)))) GO TO 1014
                                XLOSS=CX-X(1)
                                YLOSS=CY-X(2)
                                RATLOS(J,K)=YLOSS-XLOSS
                                XLOS(J,K)=XLOSS
                                YLOS(J,K)=YLOSS
                                FINTIM(J,K)=AMAX1(TX,TY)+T
                                IF(.NOT.(RATLOS(J,K).LT.0.0))GO
     &                          TO 156
                                    SIGN=1.0
                                    GO TO 166
156                               CONTINUE
                                    SIGN=-1.0
166                             CONTINUE
                                RATLOS(J,K)=ALAMDA*RATLOS(J,K)+
     &                          (1.-ALAMDA)*SIGN*FINTIM(J,K)
                                NOGO=1
1014                        CONTINUE
                            COUNT=COUNT+1
                            IED=0
                        IF(.NOT.((X(1).LE.(FINAL*CX)).OR.
     &                  (X(2).LE.(FINAL*QY)))) GO TO 1005
                        GO TO 500
600                 CONTINUE
C
C**********************************************************************
C          END OF BATTLE
C**********************************************************************
C
                XA(ZCOUNT)=XPRIN1
                YA(ZCOUNT)=XPRIN2
                NOGO=0
                ZA(JC)=RATLOS(J,K)
101             CONTINUE
100         CONTINUE
C
C**********************************************************************
C          COMPUTE ROW MINIMUMAND SMALLEST ROWMIN
C**********************************************************************
C
            DO 301 KK=1,NS1
                ROWMIN(KK)=RATLOS(KK,1)
                DO 302   JJ=2,NS2
                    IF(.NOT.(RATLOS(KK,JJ).LT.ROWMIN(KK)))
     &              GO TO 1017
                        ROWMIN(KK)=RATLOS(KK,JJ)
1017                CONTINUE
302             CONTINUE
```

```
301       CONTINUE
          DO 305    KK=2,NS1
             IF(.NOT.(ROWMIN(KK).LT.SMMIN))GO TO 1019
                SMMIN=ROWMIN(KK)
C               INSER=KK
1019         CONTINUE
305       CONTINUE
C
C*********************************************************************
C         SAMPLE 15*15 MATRICES FROM 60*60 MATRIX
C*********************************************************************
C
           DO 307 JJ=1,15
             J=4*JJ
             DO 308 KK=1,15
                K=4*KK
                RATAVE(JJ,KK)=RATLOS(J,K)
                PAY(JJ,KK)=RATAVE(JJ,KK)
                XLOSPR(JJ,KK)=XLOS(J,K)
                YLOSPR(JJ,KK)=YLOS(J,K)
                FTPR(JJ,KK)=FINTIM(J,K)
308          CONTINUE
307       CONTINUE
C
C*********************************************************************
C         OUTPUT MATRICES
C*********************************************************************
C
          WRITE(1,980)((RATAVE(JJ,KK),KK=1,15),JJ=1,15)
          WRITE(1,9814)
          WRITE(1,9811)((XLOSPR(JJ,KK),KK=1,15),JJ=1,15)
          WRITE(1,9814)
          WRITE(1,9812)((YLOSPR(JJ,KK),KK=1,15),JJ=1,15)
          WRITE(1,9814)
          WRITE(1,9813)((FTPR(JJ,KK),KK=1,15),JJ=1,15)
          WRITE(1,9814)
980       FORMAT('1',15F7.3//)
9811      FORMAT('1',15F7.3//)
9812      FORMAT('1',15F7.3//)
9813      FORMAT('1',15F7.3//)
9814      FORMAT('0','--------------')
C
C*********************************************************************
C         CALCULATE OP SOLUTION
C         ADD ABS(SMMIN) TO ALL -PAY(J,K)TO MAKE GAME VALUE
C         POSITIVE
C*********************************************************************
C
          SMMIN=ABS(SMMIN)+2.
          DO 403 JJ=1,15
             RHS(JJ)=1.
             OBCOEF(JJ)=1.
             DO 404 KK=1,15
                PAY(JJ,KK)=(PAY(JJ,KK)+SMMIN)
404          CONTINUE
403       CONTINUE
C
C         CALL SUBROUTINE LP TO COMPUTE OPTIMIZED
C         STRATEGIES
C
          CALL LP(PAY,RHS,OBCOEF,SMMIN)
C
C*********************************************************************
C         PLOTTING ROUTINE
C*********************************************************************
C
          CALL TEK618
```

```
      NPLOT=1
      DO 102 L=1,NPLOT
         CALL PAGE(8.50,11.0)
         CALL NOBRDR
         CALL BLOWUP(1.0)
         CALL AREA2D(5.50,7.00)
         READ(5,*)XVU
         WRITE(6,906)XVU
         READ(5,*)YVU
         WRITE(6,907)YVU
         READ(5,*)ZVU
         WRITE(6,908)ZVU
C        CALL FRAME
         CALL SCMPLX
         CALL XAXANG(45.0)
         CALL YAXANG(45.0)
         CALL ZAXANG(45.0)
         CALL X3NAME('X-DEPLOYMENT$',100)
         CALL Y3NAME('Y-DEPLOYMENT$',100)
         CALL Z3NAME('PAYOFF    TO     X$',100)
C
C        INSERT HEADING IN ******* ,REMOVE "C"
C
C        CALL HEADIN('*************$',100,1.0,4)
C        CALL HEADIN('*************$',100,1.0,4)
C        CALL HEADIN('*************$',100,1.0,4)
C        CALL HEADIN('*************$',100,1.0,4)
C        CALL MESSAG('*************$',100,1.2,
C     &   7.0)
         CALL VOLM3C(1.,1.,1.)
         CALL VUABS(XVU,YVU,ZVU)
         CALL GRAF3D(2.50,'SCALE',7.000,2.50,'SCALE',
     &   5.75,-5.0,'SCALE',5.000)
         CALL BOX3D
         CALL RASPLN(0.)
         CALL SURMAT(RATLOS,1,NS1,1,NS2,0)
         CALL ENDPL(0)
 102    CONTINUE
      CALL DONEPL
 906    FORMAT('0',3X,'XVU=')
 907    FORMAT('0',3X,'YVU=')
 908    FORMAT('0',3X,'ZVU=')
C     WRITE(1,988)INSER,INSEC
C988    FORMAT('0',3X,'INSER=',I4,'INSEC=',I4)
      WRITE(1,987)A,B,C,D
 987  FORMAT('0',2X,'A,B,C,D=',4F13.3)
      WRITE(1,986)U,V,ALAMDA
 986  FORMAT('0',2X,'U,V=',2F13.3,2X,'LAMDA=',F13.2)
C307 CONTINUE
      STOP
      END
C
C**********************************************************
C       SUBROUTINE FCN1
C**********************************************************
C
      SUBROUTINE FCN1(N,T,X,XPRIME)
      INTEGER N
      REAL X(N),XPRIME(N),T
      COMMON /PARAM/ A,C,U,V,S,R,B,D
      XPRIME(1)=-X(1)*(U+A*X(2))-B*X(2)
      XPRIME(2)=-X(2)*(V+C*X(1))+S-D*X(1)
      RETURN
      END
C
C**********************************************************
C       SUBROUTINE FCN2
```

```
C*************************************************************
C
        SUBROUTINE FCN2(N,T,X,XPRIME)
        INTEGER N
        REAL X(N),XPRIME(N),T
        COMMON /PARAM/ A,C,U,V,S,R,B,D
        XPRIME(1)=-X(1)*(U+A*X(2))+R-B*X(2)
        XPRIME(2)=-X(2)*(V+C*X(1))-D*X(1)
        RETURN
        END
C
C*************************************************************
C       SUBROUTINE FCN3
C*************************************************************
C
        SUBROUTINE FCN3(N,T,X,XPRIME)
        INTEGER N
        REAL X(N),XPRIME(N),T
        COMMON /PARAM/ A,C,U,V,S,R,B,D
        XPRIME(1)=-X(1)*(U+A*X(2))-B*X(2)
        XPRIME(2)=-X(2)*(V+C*X(1))-D*X(1)
        RETURN
        END
C
C*************************************************************
C       SUBROUTINE FCN4
C*************************************************************
C
        SUBROUTINE FCN4(N,T,X,XPRIME)
        INTEGER N
        REAL X(N),XPRIME(N),T
        COMMON /PARAM/ A,C,U,V,S,R,B,D
        XPRIME(1)=-X(1)*(U+A*X(2))+R-B*X(2)
        XPRIME(2)=-X(2)*(V+C*X(1))+S-D*X(1)
        RETURN
        END
C
C*************************************************************
C       SUBROUTINE LP
C*************************************************************
C
        SUBROUTINE LP(ALP,BLP,CLP,SMMIN)
        INTEGER IALP,NLP,M1,M2,IW(63),IELP
        REAL ALP(17,32),SLP,PSOL(15),DSOL(15),RW(383),
     &  SMMIN,VAL,BLP(15),CLP(15)
        NLP=15
        M1=15
        M2=0
        IALP=17
        CALL ZX4LP(ALP,IALP,BLP,CLP,NLP,M1,M2,SLP,PSOL
     &  ,DSOL,RW,IW,IELP)
        VAL=(1./SLP)-SMMIN
        WRITE(1,729)VAL
729     FORMAT('0','VALUE OF GAME=',F14.3)
        DO 405 JJ=1,15
           PSOL(JJ)=PSOL(JJ)/SLP
           DSOL(JJ)=DSOL(JJ)/SLP
           WRITE(1,730)JJ,PSOL(JJ),JJ,DSOL(JJ)
730        FORMAT('0',2X,'PROB OF.STR. # ',I3,'OF Y='
     &     ,F12.3,'PROB.OF STR. #',I3,'OF X=',F12.3)
405     CONTINUE
        RETURN
        END
```

APPENDIX I

TRANSFORMING MIXED STRATEGY PROBLEM INTO LINEAR PROGRAMMING

The payoff function has been defined to be

$$A(X,Y) = L_y - L_x$$

Y selects his optimum mixed strategies which yeild

$$V = \min_{q_j} \left\{ \max \left[\sum_{j=1}^{m} a_{1j} q_j , \sum_{j=1}^{m} a_{2j} q_j , \ldots , \sum_{j=1}^{m} a_{mj} q_j \right] \right\}$$

where

a_{ij} = payoff to x when x adopts ith strategy and y adopts jth strategy

q_j = probability that y selects jth strategy

Subject to :

$$\sum_j q_j = 1 \ , \quad q_j > 0 \ , \quad j = 1,2,\ldots,m$$

Let $v_0 \triangleq \max \left[\sum_{j=1}^{m} a_{1j} q_j , \sum_{j=1}^{m} a_{2j} q_j , \ldots , \sum_{j=1}^{m} a_{mj} q_j \right]$

Then the original problem becomes

minimize v_0

Subject to :

$$\sum_j a_{1j} q_j < v_0$$
$$\sum_j a_{2j} q_j < v_0$$
$$\vdots$$
$$\sum_j a_{mj} q_j < v_0$$
$$\sum_j q_j = 1$$
$$q_j > 0 \ , \quad j = 1,2,\ldots,m$$

Dividing the constants by v_0 (>0), we have

$$\frac{1}{v_0}\sum_{j=1}^{m} a_{1j} q_j < 1$$
$$\frac{1}{v_0}\sum_{j=1}^{m} a_{2j} q_j < 1$$
$$\vdots$$
$$\frac{1}{v_0}\sum_{j=1}^{m} a_{mj} q_j < 1$$
$$\frac{1}{v_0}\sum_{j=1}^{m} q_j = 1$$

Let $Q_j = \dfrac{q_j}{v_0}$ and since

$$\min v_0 = \max \frac{1}{v_0} = \max\,[Q_1 + Q_2 + \ldots + Q_m],$$

the problem can be written as :

$$\max Q_0 = [Q_1 + Q_2 + \ldots + Q_m]$$

Subject to :

$$\sum_j a_{1j} Q_j > 1$$

$$\sum_j a_{2j} Q_j > 1$$

$$\vdots$$

$$\sum_j a_{mj} Q_j > 1$$

$$Q_j > 0 \ , \ j = 1,2,\ldots,m$$

Since $Q_0 = \dfrac{1}{v_0}$ and $Q_j = \dfrac{q_j}{v_0}$

$$\Longrightarrow \quad q_j = Q_j v_0 = \frac{Q_j}{Q_0}$$

After solving the LP problem, the optimum strategies for y is given by $q_j^* = Q_j^* v_0$. Some constants, $K = |\min(a_{ij})|$ could have been added to a_{ij} to ensure $v_0 > 0$. If this is done, K has to be subtracted from the optimum value obtained by the LP, that is

$$v^* = Q_0 - K$$

where v^* = the value of the game.

APPENDIX J

BACKGROUND DATA ON KOREAN WAR

1. PERIOD CONSIDERED : 25 June 1950 to 7 July 1950 (No American ground involvement as yet)

2. TYPE OF ENGAGEMENT : Predominantly land combat

3. GENERAL STATE OF READINESS :

NORTH KOREA : Well prepared by 1950; arms build-up and training of troops since 1945; many military leaders and combat personnel were war veterans fighting in China

REPUBLIC OF KOREA : By 1950; a small defense force began to take shape through American aid; training only started around 1948.

4. SOURCE OF DATA : a) Appleman, R.E., <u>United States Army in the Korean War</u>, Department of the Army, 1961.

b) Montross, Lynn, <u>U.S. Marine Operations in Korea</u>, U.S. Marine Corps., 1954.

5. RELATIVE STRENGTH :

	NORTH KOREA	REPUBLIC OF KOREA
a. Total strength	= Q_x = 135,000 men	Q_y = 95,000 men
B. Tanks :	150	nil
Artillery pieces :	1,600	700
c. Aircraft		
(i) fighters	- 40	no combat aircraft
(ii) attack bombers	- 70	(22 trainer, 4
(iii) reconnaissance	- 10	auxiliary; no pilot)

6. CORRESPONDING PARAMETERS USED IN MODEL :

a = 0.7	c = 1.0
b = 0.4	d = 0.6
u = 0.15	v = 0.2

1 unit = 13,500 men

LIST OF REFERENCES

1. Dolansky, L., Present State of the Lanchester Theory of Combat, Operations Research Society of America, 12, 344-358, 1964.

2. Wozencraft, J.M.and Moose, P.H., Lanchaster Theory and Matrix Games, Proc. of the sixth MIT/ONR Workshop on C systems, December 1983.

3. Siljak, D.D., Non-linear Systems, John Wiley & Sons, 1969.

4. Ogata, K., Modern Control Engineering, Prentice-Hall, 1970.

5. Taylor, James G., Lanchester Models of Warfare, Operations Research Society of America, 1983.

6. Lanchester, F.W., Aircraft in Warfare : The Dawn of Fourth Arm No. V., The principle of Concentration , Engineering 90, 422-423, 1914.

7. Strang, L., Linear Algebra and Its Applications Academic Press, 1976, pages 319-325

8. Richter, Stephen L., Continuation Methods : Theory and Applications, IEEE Trans. on system, Man AND Cybernetics , vol. SMV-13, No. 4, July/August 1983.

9. Ohihiro, H., Non Linear Oscillations in Physical System, Mcgraw Hill, 1964.

10. Taha, H.A., Operations Research : An Introduction, MacMillan, 1971.

11. Frederick, S.H. and Gerald, J.L., Operations Research, Holden-Day, 1974.

12. Liu, C.L., Introduction to Combinatorial Mathematics, Mcgraw-Hill, 1968.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Library, Code 0142 Naval Postgraduate School Monterey, California 93943	2
2.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22314	2
3.	Professor Paul H. Moose, Code 62ME Department of Electrical and Computer Engineering, Naval Postgraduate School Monterey, California 93943	3
4.	Professor John M. Wozencraft, Code 62WE Department of Electrical and Computer Engineering, Naval Postgraduate School Monterey, California 93943	3
5.	Department Chairman, Code 62 Department of Electrical and Computer Engineering, Naval Postgraduate School Monterey, California 93943	1
6.	Dr. Percy Studt Outsdre (TWF/OM) The Pentagon Washington D.C. 20350	1
7.	Dr. William R. Verry P.O. Box 5520 Kirtland AFB N.M 87185	1
8.	Cpt Ang Bing Ning Block 5, Normanton Park 11-107, Singapore 0511 Republic of Singapore	1
9.	Staff Officer, Shore Training Naval Training Department No. 2, Ritchie Road Singapore 1024 Republic of Singapore	1

END

FILMED

4-85

DTIC