A comparison of numerical methods in non-linear stability analysis

This item was submitted to Loughborough University's Institutional Repository by the/an author.

Additional Information:

• A Doctoral Thesis. Submitted in partial fulfilment of the requirements for the award of Doctor of Philosophy at Loughborough University.

Metadata Record: [https://dspace.lboro.ac.uk/2134/27902](https://dspace.lboro.ac.uk/2134/27902)

Publisher: © James R. Hewit

Rights: This work is made available according to the conditions of the Creative Commons Attribution-NonCommercial-NoDerivatives 2.5 Generic (CC BY-NC-ND 2.5) licence. Full details of this licence are available at: [http://creativecommons.org/licenses/by-nc-nd/2.5/](http://creativecommons.org/licenses/by-nc-nd/2.5/)

Please cite the published version.
This item was submitted to Loughborough University as a PhD thesis by the author and is made available in the Institutional Repository (https://dspace.lboro.ac.uk/) under the following Creative Commons Licence conditions.

For the full text of this licence, please go to:
http://creativecommons.org/licenses/by-nc-nd/2.5/
<table>
<thead>
<tr>
<th>AUTHOR/FILING TITLE</th>
<th>HEWIT, J</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACCESSION/COPY NO.</td>
<td>OUI0482102</td>
</tr>
<tr>
<td>VOL. NO.</td>
<td></td>
</tr>
<tr>
<td>CLASS MARK</td>
<td>t</td>
</tr>
<tr>
<td>- 7 MAY 1991</td>
<td>3 JUL 1992</td>
</tr>
<tr>
<td>27 JUN 1997</td>
<td></td>
</tr>
</tbody>
</table>
A COMPARISON OF NUMERICAL METHODS
IN
NON-LINEAR STABILITY ANALYSIS

by

JAMES R. HEWIT

Submitted for the degree of Doctor of Philosophy to
Loughborough University of Technology, 1st July, 1968.
SUMMARY

Many methods have been proposed for the stability analysis of systems described by non-linear differential equations. Of these methods, some are 'ad hoc', being applicable only to those systems for which they were developed. Others are more or less general.

The increasing availability of digital computing facilities has made it desirable to develop algorithms, where possible, to allow the numerical implementation of these general methods.

This report describes the development of such algorithms for a variety of methods, and a comparison of their power and applicability in different situations.

The work falls into four main sections. After an introductory section, in which are described the various concepts, definitions and theorems to be used, the next section deals with methods based upon the direct method of Liapounov. These methods are for the construction of Liapounov functions, the plotting of stability boundaries, and the optimisation of the construction procedures. Several extensions and developments, believed to be new, are described.

The third section deals with methods which are not based upon the method of Liapounov, and again several novel extensions are described. In particular, the tracking function method, originally a graphical procedure applicable to autonomous systems only,
is shown to provide a numerical algorithm which greatly increases its effectiveness, while extensions to linear non-stationary and non-linear forced systems are discussed.

Finally, the concluding section brings together the main results emerging from the comparison of the methods, this comparison being obtained from application of the various methods to some standard systems.

The digital computer routines are given both in block diagram form and, in the Appendices, explicitly, coded in FORTRAN 4.
## CONTENTS

### Chapter 1. INTRODUCTION

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1.</td>
<td>Elementary Concepts of Stability</td>
<td>1</td>
</tr>
<tr>
<td>1.2.</td>
<td>Some Definitions</td>
<td>2</td>
</tr>
<tr>
<td>1.3.</td>
<td>Definitions of Stability</td>
<td>5</td>
</tr>
<tr>
<td>1.4.</td>
<td>The Direct Method of Liapounov</td>
<td>6</td>
</tr>
<tr>
<td>1.5.</td>
<td>The Main Theorems</td>
<td>7</td>
</tr>
<tr>
<td>1.6.</td>
<td>Non-Liapounov Methods</td>
<td>11</td>
</tr>
<tr>
<td>1.7.</td>
<td>The Systems Analyzed</td>
<td>11</td>
</tr>
<tr>
<td>1.8.</td>
<td>Motivation</td>
<td>14</td>
</tr>
</tbody>
</table>

### Chapter 2. THE METHOD OF KRASSOVSKI

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1.</td>
<td>Introduction</td>
<td>17</td>
</tr>
<tr>
<td>2.2.</td>
<td>The Computing Algorithm</td>
<td>19</td>
</tr>
<tr>
<td>2.3.</td>
<td>Examples</td>
<td>21</td>
</tr>
<tr>
<td>2.4.</td>
<td>Comments</td>
<td>22</td>
</tr>
</tbody>
</table>

### Chapter 3. THE METHOD OF SZEGÖ

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1.</td>
<td>Introduction</td>
<td>25</td>
</tr>
<tr>
<td>3.2.</td>
<td>An Example</td>
<td>28</td>
</tr>
<tr>
<td>3.3.</td>
<td>A Counter-example</td>
<td>30</td>
</tr>
<tr>
<td>3.4.</td>
<td>The Computing Algorithm</td>
<td>31</td>
</tr>
<tr>
<td>3.5.</td>
<td>The System Equations in Companion Form</td>
<td>37</td>
</tr>
<tr>
<td>3.6.</td>
<td>Validity in the Linear Case</td>
<td>39</td>
</tr>
<tr>
<td>3.7.</td>
<td>Examples</td>
<td>40</td>
</tr>
<tr>
<td>3.8.</td>
<td>Comments</td>
<td>42</td>
</tr>
</tbody>
</table>

### Chapter 4. THE METHOD OF INGWERSON

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1.</td>
<td>Introduction</td>
<td>44</td>
</tr>
<tr>
<td>4.2.</td>
<td>The Method</td>
<td>45</td>
</tr>
<tr>
<td>4.3.</td>
<td>An Example</td>
<td>49</td>
</tr>
<tr>
<td>4.4.</td>
<td>The Computing Algorithm</td>
<td>50</td>
</tr>
<tr>
<td>4.5.</td>
<td>Examples</td>
<td>57</td>
</tr>
<tr>
<td>4.6.</td>
<td>Comments</td>
<td>58</td>
</tr>
</tbody>
</table>
## Chapter 5. THE METHOD OF ZUBOV

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>Introduction</td>
<td>59</td>
</tr>
<tr>
<td>5.2</td>
<td>An Example</td>
<td>62</td>
</tr>
<tr>
<td>5.3</td>
<td>Approximations to the Boundary of the DOA</td>
<td>62</td>
</tr>
<tr>
<td>5.4</td>
<td>The Computing Algorithm</td>
<td>64</td>
</tr>
<tr>
<td>5.5</td>
<td>Examples</td>
<td>71</td>
</tr>
<tr>
<td>5.6</td>
<td>Attempted Extension to Systems of Arbitrary Order</td>
<td>72</td>
</tr>
<tr>
<td>5.7</td>
<td>Comments</td>
<td>78</td>
</tr>
</tbody>
</table>

## Chapter 6. THE METHOD OF RODDEN FOR PLOTTING THE RAS.

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1</td>
<td>Introduction</td>
<td>81</td>
</tr>
<tr>
<td>6.2</td>
<td>Location of the $V = 0$ locus</td>
<td>82</td>
</tr>
<tr>
<td>6.3</td>
<td>Location of the Point of Tangency</td>
<td>84</td>
</tr>
<tr>
<td>6.4</td>
<td>Tracing the Boundary of the RAS</td>
<td>87</td>
</tr>
<tr>
<td>6.5</td>
<td>Two Pathological Configurations</td>
<td>91</td>
</tr>
</tbody>
</table>

## Chapter 7. OPTIMISATION OF THE CONSTRUCTION PROCEDURES

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.1</td>
<td>Introduction</td>
<td>93</td>
</tr>
<tr>
<td>7.2</td>
<td>The Size of the RAS as a Function of Certain Coefficients</td>
<td>93</td>
</tr>
<tr>
<td>7.3</td>
<td>Optimisation by Direct Search</td>
<td>96</td>
</tr>
<tr>
<td>7.4</td>
<td>Examples</td>
<td>98</td>
</tr>
<tr>
<td>7.5</td>
<td>Sequential Optimisation of the Zubov Procedure</td>
<td>100</td>
</tr>
<tr>
<td>7.6</td>
<td>Examples of Sequential Optimisation</td>
<td>101</td>
</tr>
<tr>
<td>7.7</td>
<td>Comments</td>
<td>102</td>
</tr>
</tbody>
</table>

## Chapter 8. THE METHOD OF WEISSENBERGER

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.1</td>
<td>Introduction</td>
<td>103</td>
</tr>
<tr>
<td>8.2</td>
<td>The Method</td>
<td>104</td>
</tr>
<tr>
<td>8.3</td>
<td>Weissemberger's Conjecture</td>
<td>107</td>
</tr>
<tr>
<td>8.4</td>
<td>Examples</td>
<td>109</td>
</tr>
<tr>
<td>8.5</td>
<td>Comments</td>
<td>111</td>
</tr>
</tbody>
</table>
### Chapter 9. THE TRACKING FUNCTION METHOD

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.1</td>
<td>Practical Stability</td>
<td>112</td>
</tr>
<tr>
<td>9.2</td>
<td>Tracking Functions</td>
<td>114</td>
</tr>
<tr>
<td>9.3</td>
<td>Disadvantages of the Graphical Method</td>
<td>116</td>
</tr>
<tr>
<td>9.4</td>
<td>The Computing Algorithm</td>
<td>117</td>
</tr>
<tr>
<td>9.5</td>
<td>Salient Points Concerning the Algorithm</td>
<td>120</td>
</tr>
<tr>
<td>9.6</td>
<td>Examples</td>
<td>122</td>
</tr>
<tr>
<td>9.7</td>
<td>Practical Stability in Polar Coordinates</td>
<td>126</td>
</tr>
<tr>
<td>9.8</td>
<td>Examples</td>
<td>128</td>
</tr>
<tr>
<td>9.9</td>
<td>Extension to Linear Non-Stationary Systems</td>
<td>129</td>
</tr>
<tr>
<td>9.10</td>
<td>Computing Algorithm for Non-Stationary Systems</td>
<td>136</td>
</tr>
<tr>
<td>9.11</td>
<td>Extension to Systems with Bounded Forcing</td>
<td>138</td>
</tr>
</tbody>
</table>

### Chapter 10. THE METHOD OF LUUS AND LAPI DUS

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.1</td>
<td>Introduction</td>
<td>144</td>
</tr>
<tr>
<td>10.2</td>
<td>The Method</td>
<td>144</td>
</tr>
<tr>
<td>10.3</td>
<td>An Example</td>
<td>148</td>
</tr>
<tr>
<td>10.4</td>
<td>Comparison with the Tracking Function Method</td>
<td>150</td>
</tr>
<tr>
<td>10.5</td>
<td>The Computing Algorithm</td>
<td>151</td>
</tr>
<tr>
<td>10.6</td>
<td>Examples</td>
<td>155</td>
</tr>
</tbody>
</table>

### Chapter 11. THE METHOD OF THE ALTERNATING EXTREME RADIUS PATH

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.1</td>
<td>Introduction</td>
<td>157</td>
</tr>
<tr>
<td>11.2</td>
<td>Construction of the A.E.R.P.</td>
<td>157</td>
</tr>
<tr>
<td>11.3</td>
<td>Illustration of the Ambiguity in the Choice of Radius Function</td>
<td>160</td>
</tr>
<tr>
<td>11.4</td>
<td>The Computing Algorithm</td>
<td>162</td>
</tr>
<tr>
<td>11.5</td>
<td>Examples</td>
<td>164</td>
</tr>
<tr>
<td>11.6</td>
<td>Comments</td>
<td>166</td>
</tr>
</tbody>
</table>

### Chapter 12. CONCLUSIONS

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ACKNOWLEDGEMENTS.</td>
<td>174</td>
</tr>
<tr>
<td></td>
<td>REFERENCES</td>
<td>175</td>
</tr>
<tr>
<td></td>
<td>APPENDICES</td>
<td>179</td>
</tr>
<tr>
<td></td>
<td>FIGURES.</td>
<td>240</td>
</tr>
</tbody>
</table>
1. **INTRODUCTION**

1.1. **Elementary Concepts of Stability.**

The notion of stability is probably introduced first in the study of elementary mechanics where, in considering the position of some mechanical device, it is required to assess the effect that some small displacement will have.

It is usual to class such systems into three categories namely stable systems, unstable systems and neutrally stable systems. A much quoted example of a stable system is a smooth sphere resting in a concave basin, while a smooth sphere at the apex of a convex surface illustrates instability. Neutral stability is illustrated by a smooth sphere resting on a smooth horizontal surface.

That such concepts are not even always in accord with intuition is seen from the example of a right circular cylinder standing vertically on a smooth horizontal surface. No matter how small the diameter of the base, this system is classed as stable since it is always possible in theory to postulate a small enough disturbance that the cylinder returns to its vertical position.

However, these elementary concepts do provide the basis for the more rigorous concepts required for modern stability analysis and give good insight into the essentially 'local'
nature of some of the criteria.

1.2. Some Definitions

We shall henceforth be concerned with systems whose motion is described by the set of \( n \) first order differential equations:

\[
\frac{dx_i}{dt} = x_i = f_i(x_1, x_2, \ldots, x_n, t) \quad 1.2.1
\]

\( i = 1, 2, \ldots, n \)

or in vector notation

\[
\dot{x} = f(x, t) \quad 1.2.2
\]

where \( x = (x_1, x_2, \ldots, x_n)^T \) is the state vector while the individual elements \( x_i \) are the state variables. The vector \( x \) may be viewed as defining a point in the \( n \)-dimensional state space \( X \) or the \( (n+1) \)-dimensional motion space \( X \times T \) where \( T \) is the range of \( t \).

If the right-hand side of 1.2.2 is continuous and such as to ensure the existence and uniqueness of solutions continuously dependent upon the initial conditions \( (x_0, t_0) \) then we say the function \( f(x, t) \) is of class \( E \) or \( f \in E \).

If \( f \) is of class \( E \), the function
\[ \phi(t, x_0, t_0) \]

denotes the well defined solution of 1.2.2 and obeys

\[ \phi(t_0, x_0, t_0) = x_0 \]

\[ \phi(t, x_0, t_0) = \int \{ \phi(t, x_0, t_0), t \} \]

This solution defines a motion in the motion space. The projection of the motion on the state space \( X \) is called a trajectory and the time \( t \) appears as a curve parameter.

A constant solution

\[ \phi(t, x_0, t_0) \equiv x_0 \]

is called an equilibrium, and if in a neighbourhood of \( x_0 \) no other equilibrium exists then \( x_0 \) is an isolated equilibrium. It will be assumed hereafter that all equilibria are isolated. It will further be assumed that all equilibria are situated at the origin of the state space. This can always be achieved by a simple linear transformation.

If in 1.2.2 the vector \( f(x, t) \) is independent of \( t \) i.e. \( f(x, t) \equiv f(x) \) then the system is autonomous. If \( f(x, t) \) can be written as \( A(t)x \) where \( A(t) \) is a matrix the system is linear.

For the autonomous system

\[ \dot{x} = f(x) \]
if the function \( f(x) \) possesses a convergent power series expansion about the origin the system may be written as

\[
\dot{x} = Ax + f_2(x)
\]

where \( A \) is a constant matrix and \( f_2(x) \) is a polynomial with terms of at least second degree.

The system \( \dot{x} = Ax \) is called the linear parts of (or the first approximation to) the system 1.2.7. If the eigenvalues of \( A \) all have negative non-zero real parts then \( A \) is called a stability matrix.

The following definitions apply

Definition 1.2.1

A scalar function \( V(x) \) is positive (negative) definite if in some neighbourhood of the origin \( V(x) > 0 \) (\( < 0 \)) and \( V(0) = 0 \). If \( V(x) \geq 0 \) (\( \leq 0 \)) and \( V(0) = 0 \) it is positive (negative) semidefinite.

Definition 1.2.2

A scalar function \( V(x,t) \) is positive (negative) definite if there exists a positive definite function \( \mathcal{W}(x) \) such that in some neighbourhood of the origin

\[
V(x,t) > \mathcal{W}(x) \quad \left\{ < -\mathcal{W}(x) \right\}
\]

and in addition \( V(0,t) = 0, \; t \geq t_0 \).
The function $V(\mathbf{x},t)$ is positive (negative) semi-definite if in some neighbourhood of the origin

$$V(\mathbf{x},t) \geq 0 \quad (\leq 0)$$

and in addition $V(\mathbf{0},t) = 0$, $t \geq t_0$.

Definition 1.2.3

A scalar function $V(\mathbf{x},t)$ is decrescent if the relation

$$\lim_{|\mathbf{x}| \to 0} V(\mathbf{x},t) = 0$$

holds uniformly in $t$.

A scalar function $V(x)$ is decrescent if $V(\mathbf{0}) = 0$ and $V(x)$ is continuous at $x = \mathbf{0}$.

1.3. Definitions of Stability.

The following definitions apply to the stability of the isolated equilibrium $\mathbf{x} = \mathbf{0}$ of the system

$$\dot{x} = f(x,t) \quad f(\mathbf{0},t) = 0 \quad 1.3.1$$

$f \in E$

Definition 1.3.1. (Stability)

The equilibrium of the system 1.3.1 is stable if for any $\varepsilon > 0$ there exists a $\delta > 0$ such that $|x_0| < \delta$ implies

$$|\phi(t,x_0,t_0)| < \varepsilon, \quad t \geq t_0.$$
Definition 1.3.2. (Asymptotic Stability)

The equilibrium of system 1.3.1 is asymptotically stable if it is stable and in addition there exists a $\lambda > 0$ such that $|x_o| < \lambda$ implies $\lim_{t \to \infty} \phi(t, x_o, t_o) = 0$.

Definition 1.3.3. (Instability)

The equilibrium of system 1.3.1 is unstable if it is not stable.

A multitude of other definitions have been proposed\textsuperscript{3,4,5} but for the purposes of this work the above will suffice.

We have the following theorem

Theorem 1.3.1.

For the autonomous system $\dot{x} = f(x)$ assume that $f(x)$ may be written as $A x + f_2(x)$ where $A$ is a stability matrix and $f_2(x)$ is a polynomial with terms of at least second degree. Then the system $\dot{x} = f(x)$ is asymptotically stable.

1.4. The Direct Method of Liapounov

If the solutions $\phi(t, x_o, t_o)$ to the system 1.3.1 were available, the analysis of stability would be merely a matter of inspection. In general, however, solution of 1.3.1 by analytic means is impossible and while numerical integration could be employed, the dependence of the solut-
ions on the initial conditions would necessitate a large
number of integrations with resultant increase in computing
time.

The direct method of Liapounov offers an approach by
which the stability of the equilibrium may be assessed
without recourse to the solutions of the describing equations.
This is achieved by consideration of certain scalar functions
defined in the motion space and possessing certain character-
istics. These functions are the so-called Liapounov
functions.

1.5. The Main Theorems.

The following theorems, which are presented without
proof, provide the basis of the analysis of stability by the
direct method.

Theorem 1.5.1. (Stability)

The equilibrium \( x = 0 \) of the system

\[
\dot{x} = f(x,t) \quad f(0,t) = 0 \quad \forall \in E \quad 1.5.1
\]

is stable provided there exists a positive definite scalar
function \( V(x,t) \) whose time derivative \( \dot{V}(x,t) \) by virtue of
1.5.1 is at least negative semidefinite.

(for proof see ref. 2.)
Theorem 1.3.2. (Asymptotic Stability)

The equilibrium of the system 1.5.1 is asymptotically stable if there exists a positive definite scalar function $V(x,t)$ such that

a. $V(x,t)$ is negative definite.
b. $V(x,t)$ is decreasent.

(For autonomous systems the condition a may be replaced by the weaker conditions

a$_1$. $V(x)$ is negative semi-definite.
a$_2$. $V(x)$ does not vanish identically on any non-trivial trajectory.

(for proof see ref. 2.)

The above theorems are concerned with stability and asymptotic stability in the sense of definitions 1.3.1 and 1.3.2 and are of a local character insofar as it is not possible a-priori to determine the magnitude of $S$. For linear systems stability extends to the entire state space. In other words, if the equilibrium is asymptotically stable (say) then the response to any initial conditions will eventually converge to the equilibrium. For non-linear systems, however, this is not generally true. The following theorem provides a means of assessing the extent of asymptotic stability for non-linear systems.
Theorem 1.5.3. (The Extent of Asymptotic Stability)\(^6\)

The response of the system 1.5.1 to initial conditions lying in some region \( D \) containing the equilibrium converges to the equilibrium asymptotically if there exists a scalar function \( V(x,t) \), positive definite in \( D \) and such that

1. \( \dot{V}(x,t) \) by virtue of 1.5.1 is at least negative semi-definite in \( D \).
2. \( V(0,t) = 0 \) and \( \dot{V}(x,t) \neq 0 \) on any other trajectory of the system in \( D \).
3. \( \nabla V(x,t) \neq 0 \) except at \( x = 0 \) in \( D \).
4. One of the curves \( V(x,t) = \) constant defines the boundary of \( D \).

Such a region \( D \) is called a region of asymptotic stability (RAS). If it is the entire region of initial conditions for which the responses approach the equilibrium asymptotically it is called the domain of attraction (DOA). If the domain of attraction is the whole space the system is asymptotically stable in the large.\(^2\)

Theorem 1.5.4. (Instability)\(^2\)

Let the system 1.5.1 be autonomous. Then the equilibrium is unstable if there exists a positive definite scalar function \( V(x) \) with continuous first partial derivatives and such that its time derivative \( \dot{V}(x) \) by virtue of 1.5.1 can assume positive values arbitrarily near the equilibrium.
The theorem of greatest use is theorem 1.5.3 since it provides the means of assessing not only stability but also the extent of stability.

The two main tasks in the application of Iiapounov's direct method are

a. The generation of a suitable Iiapounov function \( V(x, t) \) if it exists and

b. The determination of the boundary of the region of stability indicated by this Iiapounov function.

In chapters 2, 3, 4 and 5 are presented several methods for solving the first of these problems and in chapter 6 a---method for solving the second.

The following theorem provides a method for generating a Iiapounov function for a linear system.

**Theorem 1.5.5:**

For the linear system \( \dot{x} = Ax \) where \( A \) is a stability matrix the function \( V(x) = x^T B x \) is a Iiapounov function where \( B \) is the solution of the Iiapounov matrix equation

\[
A^T B + BA = -C
\]

where \( C \) is an arbitrary positive definite symmetric matrix.

Although the direct method of Liapounov has assumed a dominant role in modern stability analysis, many other methods have been proposed.

In chapters 9, 10, and 11 three such methods are described. The method of Ius and Lapidus\(^7\) and the method of the alternating extreme radius path\(^8\) make use of some simplifying approximations and because of this, as is shown, the results must be interpreted with care. The tracking function method\(^9\) involves no approximation but is concerned with a weaker concept of stability, namely practical stability (defined in chapter 9).

1.7. The Systems Analysed.

Each of the methods discussed hereafter is applied to some or all of a number of second order systems. To avoid the necessity of repeatedly writing the system equations these equations are given here and are designated in the text by the appropriate number. The equations of other systems which are used only occasionally are given where the necessity arises.
System 1.7.1. The van der Pol equation.

\[
\begin{align*}
\dot{x} &= y \\
\dot{y} &= \epsilon (1-x^2)y - x \quad \epsilon > 0
\end{align*}
\]

This system possesses an unstable equilibrium at the origin and a stable limit cycle of radius \( \sim 2 \) whose shape is dependent upon the value of \( \epsilon \).

System 1.7.1.A. The system 1.7.1 with time reversed.

This system is asymptotically stable. The DOA is the interior of the limit cycle of system 1.7.1.

System 1.7.2. An example from Rodden.\(^{10}\)

\[
\begin{align*}
\dot{x} &= y \\
\dot{y} &= -x - y + x^3
\end{align*}
\]

This system is asymptotically stable. The DOA is unbounded and shown in fig. 3.4.

System 1.7.3. An example from Hahn.\(^2\)

\[
\begin{align*}
\dot{x} &= -x + 2x^2y \\
\dot{y} &= -y
\end{align*}
\]

This system is asymptotically stable. The DOA is the region defined by \( xy < 1 \).
System 1.7.4. A Chemical Reactor

\[
\begin{align*}
\dot{c} &= (-.875c-.8737)f(\Theta) - c - 1.1124 \\
\dot{\Theta} &= (.8743c-.8735)f(\Theta) - (5.57\Theta - .6853)g(\Theta) \\
&\quad - 1.18\Theta - .4269
\end{align*}
\]

where \( f(\Theta) = \exp\left\{-35.925 \left( \frac{1}{\Theta+1.754} \right)\right\} 10^9 \)

\( g(\Theta) = \exp(3.908\Theta) \)

where \( c \) and \( \Theta \) are respectively the normalised concentration and temperature. This system has an unstable equilibrium at the origin and a stable limit cycle (see fig. 11.4.).

System 1.7.4.A. The above system with time reversed.

This system is asymptotically stable. The DOA is the interior of the limit cycle of system 1.7.4.

System 1.7.5. A Surge Tank system.

\[
\begin{align*}
\dot{x} &= y \\
\dot{y} &= -x\left\{1 - \beta \left(\frac{2+x}{1+x}\right)^2\right\} - \frac{\alpha^2}{\beta} y^2 - \\
&\quad \frac{-\beta \gamma}{\alpha(1+x)^2} \left\{ \frac{2\alpha^2}{\beta} - 1 + \frac{2\alpha^2}{\beta} \right\}
\end{align*}
\]

where \( x \) is the normalised displacement of the water level in the surge tank from its equilibrium position (see fig. 11.6.).
This system has an equilibrium at the origin the stability of which depends on the values of $\alpha$ and $\beta$ which in turn depend on the physical dimensions of the system.

1.8. Motivation.

The motivation for the investigation described herein was the realisation that whereas the theoretical content of modern stability analysis has increased rapidly, the actual implementation of the various methods has remained a rather neglected field. Thus, in the direct method of Liapounov, while a very great many theorems have been proposed to cover a wide variety of situations each requires the establishment of the existence of a suitable Liapounov function. For systems of even moderate complexity the construction of such a function can be extremely tedious and has been a matter of art and experience rather than a completely formal procedure.

With the increasingly easy availability of digital computing facilities it seemed worthwhile to pursue an investigation with two main objectives (a) to program those methods which were known to provide computing algorithms and to compare their efficiency in different situations and (b) to seek new and hopefully more powerful algorithms.

At the outset those methods which were known to provide algorithms were the methods of Zubov\textsuperscript{12,13,14} and Ingwerson\textsuperscript{6}.
During the course of the investigation Weissenberger published a paper describing a numerical method for generating optimum Liapounov functions. In addition to these three methods, algorithms, believed to be new, have been developed from the methods of Krassovski and Szego. A new method for optimising some of these procedures has been developed and shown to be more effective.

The method of Weissenberger is shown to be a special case of this optimisation procedure, while Weissenberger's conjecture for the construction of high degree optimum Liapounov functions is shown to be invalid.

The method of Rodden for tracing the boundary of the RAS is described and several modifications introduced to increase the speed of computation. The simple configurations in the state space described by Rodden are shown to be not the most general case and more complicated configurations are discussed which offer considerable problems particularly to the optimisation procedures.

Three non-Liapounov methods are described.

The tracking function method is shown to be amenable to machine computation. The algorithm which has been developed is optimal in the sense that the best tracking functions are selected and others ignored. This digital routine converts a tedious and inaccurate graphical procedure into a fast and efficient numerical procedure.
A novel extension of the tracking function method to the polar plane (plotted in rectangular Cartesian coordinates) is described. This extension is believed to be more powerful in certain cases.

The tracking function method is extended both in its graphical formulation and the numerical procedure to non-autonomous systems. Results have been obtained which are better than those obtained by previous methods.

An algorithm is described for the method of Iuus and Lapidus which disposes of the tedium of applying the method in its original formulation.

The alternating extreme radius path method is described and shown to provide an effective algorithm. An example is given which illustrates the care with which results must be interpreted.

All of the methods are compared by application to some standard systems.

At this point it must be emphasised that the number of results obtained has been subject to limitations in computing time available as is inevitable. It is believed that the results which have been obtained are fairly representative and permit a reasonable assessment of the various methods to be made. However it is possible that by using different systems in the analysis different opinions could be obtained. The conclusions should not, therefore be considered definitive.
2. THE METHOD OF KRASSOVSKI

2.1. Introduction

Krassovski proposed the following theorem concerning the stability of the system

\[ \dot{x} = f(x), \quad f(0) = 0, \quad x \in E \]  \hspace{1cm} 2.1.1

Theorem 2.1.1.

The function

\[ V(x) = f^T J(x) \]  \hspace{1cm} 2.1.2

is a Liapounov function sufficient to prove the asymptotic stability in the large of the system 2.1.1 if the symmetric matrix \( C(x) \) given by

\[ C(x) = -\left\{ f^T(x)B + B J(x) \right\} \]  \hspace{1cm} 2.1.3

is positive definite for all \( x \). In 2.1.3 the matrix \( J(x) \) is the Jacobian defined by

\[ J(x) \equiv \left\{ J_{ij} \right\} = \left\{ \frac{\partial f_i}{\partial x_j} \right\} \]  \hspace{1cm} 2.1.4

This condition will not in general be satisfied. However consider the linear autonomous system

\[ \dot{x} = Ax \]  \hspace{1cm} 2.1.5

17.
and let us assume that the system is asymptotically stable i.e. \( A \) is a stability matrix. For this system the Jacobian \( J \) is simply \( A \) and equation 2.1.3 is equivalent to the Lyapunov equation

\[
A^T B + BA = -C
\]

and the function \( V(x) = x^T B x \) is a Lyapunov function from theorem 1.5.5. Application of theorem 1.3.1 shows that this function is also a Lyapunov function for the non-linear system 2.1.1 provided the system has the linear parts of equation 2.1.5. It can be shown (see section 2.4) that the function defined by equation 2.1.2 is also a Lyapunov function provided \( B \) is the solution of equation 2.1.6 with positive definite symmetric \( C \). This is the Lyapunov function which we shall construct in polynomial form. The computing algorithm is extremely simple and straightforward. It is described below for second order systems, extension to systems of higher order being obvious.
2.2 The Computing Algorithm

Let us write the system equations as

\[ \begin{align*}
  x &= \sum_{i=1}^{\infty} \sum_{j=1}^{\infty} p_{ij} x^{i-1} y^{j-1} = f(x,y) \quad 2.2.1 \\
  y &= \sum_{i=1}^{\infty} \sum_{j=1}^{\infty} q_{ij} x^{i-1} y^{j-1} = g(x,y)
\end{align*} \]

and let us denote the positive definite symmetric matrix \( C \) and the symmetric matrix \( B \) by

\[ C = \begin{bmatrix} c_{11} & c_{12} \\ c_{12} & c_{22} \end{bmatrix} \quad 2.2.2 \]

\[ B = \begin{bmatrix} b_{11} & b_{12} \\ b_{12} & b_{22} \end{bmatrix} \quad 2.2.3 \]

It can be shown that the elements of \( B \) from 2.1.6 are given by

\[ b_{11} = \left( -c_{11} p_{11} q_{12} - c_{11} q_{12} + c_{11} q_{12} \right) / \Delta \quad 2.2.4 \]

\[ b_{12} = \left( -2c_{12} p_{11} q_{12} + c_{22} p_{11} q_{11} + c_{22} q_{11} \right) / \Delta \quad 2.2.5 \]

\[ b_{22} = \left( -c_{22} p_{11} q_{12} - c_{22} p_{11} q_{12} + 2c_{22} p_{11} q_{12} + \Delta \right) / \Delta \quad 2.2.6 \]
where
\[ \Delta = (q_{12} p_{11} - q_{11} p_{12})(p_{11} + q_{12}) \]  

2.2.7

Having thus obtained the elements of \( R \) the Liapounov function
\[ V(x,y) = f^T(x,y) R f(x,y) \]  

2.2.8

is easily obtained thus
\[ V(x,y) = b_{11} f^2(x,y) + 2b_{12} f(x,y) g(x,y) + b_{22} g^2(x,y) \]  

2.2.9

\[ = b_{11} \left\{ \sum_{l=1}^{m} \sum_{j=1}^{l} p_{ij} x^{i-j+1} y^{j-1} \right\}^2 \]

2.2.10

\[ + 2b_{12} \left\{ \sum_{l=1}^{m} \sum_{j=1}^{l} p_{ij} x^{i-j+1} y^{j-1} \right\} \times \left\{ \sum_{l=1}^{m} \sum_{j=1}^{l} q_{ij} x^{i-j+1} y^{j-1} \right\} \]

\[ + b_{22} \left\{ \sum_{l=1}^{m} \sum_{j=1}^{l} q_{ij} x^{i-j+1} y^{j-1} \right\}^2 \]

2.2.11

If the Liapounov function is required in the standard form
\[ V(x,y) = \sum_{l=2}^{m} \sum_{j=1}^{l} a_{ij} x^{i-j+1} y^{j-1} \]  

2.2.12

20.
then the coefficients $a_{ij}$ are obtained by the following
successive additions to the initial values $a_{ij} = 0$, all $i, j$

$$a_{i+k,j+s-1} = a_{i+k,j+s-1} + b_{ij}p_{ij}g_{ks} +$$

$$+ 2b_{ij}q_{ks} + b_{22}g_{ij}q_{ks}$$

$$i = 1, 2, \ldots, \infty$$

$$j = 1, 2, \ldots, (i+1)$$

$$k = 1, 2, \ldots, \infty$$

$$s = 1, 2, \ldots, (k+1)$$

As the algorithm is self evident from 2.2.13 no flow
diagram is given. The program designated SUBROUTINE INGIVER
is described in Appendix A.1. The reason for adopting the
name INGIVER which is also the name of the routine for the
method of Ingwerson is that the specification is the same for
each. Both routines are called by the main program of
Appendix A.4.

2.3. Examples

2.3.1. The system 1.7.2.

Figs. 2.1 and 2.2 show the RAS obtained for this system
using the matrices

$$\mathbf{C} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \quad \text{and} \quad \mathbf{C} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

respectively.
The different degrees of Liapounov function, obtained by truncation of the polynomial 2.2.12 give different RAS. It is seen that the higher degree functions indicate regions of inferior size than the quadratic regions. In other words the use of Krassovski's method gives poorer results than are obtained merely by linearising the system equations.

2.3.2. The System 1.7.3.

Figs. 2.3 and 2.4 show the RAS for this system using the same C matrices as in the above example. Again it is clear that the method results in inferior Liapounov functions.

2.3.3. The System 1.7.1.1. (ε = 0.1)

Fig 2.5 shows the RAS for this system. Again the effect of applying Krassovski's method is detrimental.

2.4. Comments

In a recent paper Mauror and Garlid state that the function

\[ V = \int^{\infty}_0 x^T P x \] 2.4.1

is a Liapounov function for the system 2.1.1 for any positive definite symmetric \( P \). We show here that this is not generally true and that an additional condition on \( P \) is required namely that \( P \) be the solution of the Liapounov matrix equation 2.1.6. We require the following lemma.
Lemma 2.4.1.

The matrix defined by \( A^T R A \) where \( A \) is non-singular is positive definite if and only if the matrix \( B \) is positive definite.

Now consider the linear parts of the system namely 2.1.5 bearing in mind that \( A \) is a stability matrix and cannot therefore be singular. Let \( B \) be some positive definite matrix and consider the function \( V \) defined in 2.2.8. For this linear system \( V \) is just

\[
V = (\dot{x})^T B \dot{x} \tag{2.4.2}
\]

\[
= x^T A^T B \dot{x} \tag{2.4.3}
\]

which by the lemma 2.4.1 is positive definite. In order to qualify as a Liapounov function however the time derivative \( \dot{V} \) must be negative (semi-) definite. Differentiating we have

\[
\dot{V} = x^T A^T B \dot{x} + x^T A^T B \dot{x} \tag{2.4.4}
\]

\[
= x^T A^T B \dot{x} + x^T A^T B \dot{x} \tag{2.4.5}
\]

\[
= x^T A^T (A^T B + B A) \dot{x} \tag{2.4.6}
\]

By lemma 2.4.1 this will be negative (semi-) definite if the function \( (A^T B + BA) \) is negative (semi-) definite which is equivalent to the requirement that \( B \) be the solution of 2.1.6.
As an illustration consider the system

\[ \begin{array}{c}
\dot{x} = y \\
\dot{y} = -x - y
\end{array} \]  \hspace{1cm} 2.4.7

which is asymptotically stable. Using the positive definite symmetric matrix

\[ B = \begin{bmatrix} 1.0 & 0.9 \\ 0.9 & 1.0 \end{bmatrix} \]  \hspace{1cm} 2.4.8

there results the function

\[ V = x^2 + 0.2xy + 0.2y^2 \]  \hspace{1cm} 2.4.9

which is positive definite. Its time derivative is

\[ \dot{V} = -0.2x^2 + 1.4xy - 0.2y^2 \]  \hspace{1cm} 2.4.10

which is sign indefinite. \( V \) therefore is not a Lyapunov function.
3. THE METHOD OF SZEGO

3.1. Introduction

The method of Szego\textsuperscript{24,25} is a procedure for the construction of Liapounov functions. It is described below for second order systems although extension to systems of higher order appears feasible.

The method is restricted to a particular class of systems namely systems describable by the pair of first order differential equations

\[ \begin{align*}
\dot{x} &= f(x,y) & f(0,0) = 0 \\
\dot{y} &= g(x,y) & g(0,0) = 0
\end{align*} \quad 3.1.1
\]

with the additional condition that \( f(x,y) \) and \( g(x,y) \) are linear in either \( x \) or \( y \). Without loss of generality we assume them to be linear in \( y \).

We construct a function \( V(x,y) \) given by

\[ V(x,y) = \alpha(x) \cdot x^2 + \beta(x) \cdot xy + y^2 \quad 3.1.2 \]

The time derivative \( \dot{V} \) of this function by virtue of 3.1.1 is given by

\[ \dot{V} = \frac{\partial V}{\partial x} f + \frac{\partial V}{\partial y} g \quad 3.1.3 \]
\[ V = \left\{ 2\alpha(x) + x\frac{d\alpha(x)}{dx} \right\} f(x, y) \]
\[ + \left\{ \beta(x) + x\frac{d\beta(x)}{dx} \right\} y f(x, y) \]
\[ + \left\{ \beta(x) + 2y \right\} g(x, y) \]  

3.1.4

Since \( 2\alpha(x) \) and \( \frac{d\alpha(x)}{dx} \cdot x \) are of the same degree, the first bracket in 3.1.4 may be replaced by a new function
\[ \alpha'(x) \equiv 2\alpha(x) + \frac{d\alpha(x)}{dx} \cdot x \]  

3.1.5

and similarly, the second bracket may be replaced by
\[ \beta'(x) \equiv \beta(x) + \frac{d\beta(x)}{dx} \cdot x \]  

3.1.6

giving
\[ V = \left\{ \alpha'(x) \cdot x + \beta'(x) \cdot y \right\} f(x, y) \]
\[ + \left\{ \beta(x) \cdot x + 2y \right\} g(x, y) \]  

3.1.7

Szegő now employs the device of generating a function \( \mathcal{J}(x, y) \) of the same form as \( V \) by identifying the functions \( \alpha'(x) \) and \( \beta'(x) \) with \( \alpha(x) \) and \( \beta(x) \) respectively in 3.1.7 leaving
\[ \mathcal{J} = \left\{ \alpha(x) \cdot x + \beta(x) \cdot y \right\} f(x, y) \]
\[ + \left\{ \beta(x) \cdot x + 2y \right\} g(x, y) \]  

3.1.8
Because of the linearity of $f(x,y)$ and $g(x,y)$ in $y$ this function is quadratic in $y$ and may be written as

$$\mathcal{Y} = A(x)y^2 + B(x)y + C(x)$$ \hspace{1cm} 3.1.9

if the two $x$-dependent roots of this quadratic are constrained to coincide by suitable choice of the functions $\alpha(x)$ and $\beta(x)$ the sign of $\mathcal{Y}$ will not change along any line in the state plane parallel to the $y$-axis. The resulting function $\mathcal{Y}$ will then hopefully possess sign semi-definiteness in some neighbourhood of the origin.

Using the form of $\mathcal{Y}$ so obtained, the forms of the polynomials $\alpha(x)$ and $\beta(x)$ are established and thus the form of $V$ from 3.1.2.

The entire process is re-applied to this form and its unknown coefficients evaluated to provide the actual function $V$ which is hopefully a Liapounov function.

The requirement that the roots of 3.1.9 coincide may be accomplished by the condition

$$B^0(x) = 4A(x)C(x)$$ \hspace{1cm} 3.1.10

As this condition may be satisfied in a variety of ways no unique Liapounov function is generated. It may also occur that 3.1.10 cannot be satisfied at all. However in many cases the method does produce a Liapounov function.

The method is illustrated by the following example.
3.2. **An Example**

Consider the system

\[
\begin{align*}
\dot{x} &= y \\
\dot{y} &= -y - x^3
\end{align*}
\]

From 3.1.8 we have, dropping arguments for convenience

\[
\nabla = (\alpha x + \beta y)y + (\beta x + 2y)(-y - x^3)
\]

Putting this in the form of 3.1.9 we have

\[
\nabla = (\beta - 2)y^2 + (\alpha x - \beta x - 2x^3)y \\
+ (-\beta x^4)
\]

\[
= A(x)y^2 + B(x)y + C(x)
\]

To achieve condition 3.1.10 let us set \(A(x)\) and \(B(x)\) equal to zero identically, then

\[
\beta = 2 \\
\alpha = 2 + 2x^2
\]

and, since at this stage we are interested only in the forms of these functions we have

\[
\beta \text{ has the form } c_1 \\
\alpha \text{ has the form } c_2 + c_3x^2
\]
where $c_1$, $c_2$ and $c_3$ are constants to be determined. Substituting in 3.1.2 we have

$$V(x,y) = c_2 x^2 + c_3 x^4 + c_1 xy + y^2 \quad 3.2.8$$

and the derivative $\dot{V}$ is, by virtue of 3.2.1,

$$\dot{V}(x,y) = (c_1 - 2)y^2 \rightleftharpoons (3.2.9)$$

$$+ (2c_2 x + 4c_3 x^3 - c_1 x - 2x^3)y$$

$$+ (-c_1 x^4)$$

which is again of the form 3.2.4. Letting $\lambda(x)$ and $B(x)$ vanish identically we have

$$c_1 = 2 \quad c_2 = 1 \quad c_3 = \frac{1}{2} \quad 3.2.10$$

The function $V$ is then

$$V(x,y) = x^2 + 2xy + y^2 + \frac{x^4}{2} \quad 3.2.11$$

and

$$\dot{V}(x) = -2x^4 \quad 3.2.12$$

The function $V$ satisfies the conditions of theorem 1.5.3 in the entire state plane and is thus a Liapounov function proving the system to be asymptotically stable in the large.
3.3. A Counter-Example

To illustrate the invalidity of the procedure in some cases consider the system 1.7.3. The equations are

\[\begin{align*}
\dot{x} &= -x + 2x^2 y \\
\dot{y} &= -y
\end{align*}\] 3.3.1

\[\Lambda(x) y^2 + B(x) y + C(x)\] 3.3.4

To satisfy 3.1.10 it is sufficient to set \(B(x)\) and either \(\Lambda(x)\) or \(C(x)\) identically zero. However in this case no polynomials \(\alpha(x)\) and \(\beta(x)\) may be found which achieve this (except \(\alpha(x) = /\beta(x) = 0\), in which case \(V = y^2\) and is not positive definite). It may yet be possible to satisfy 3.1.10 without the vanishing of \(\Lambda(x) B(x)\) or \(C(x)\). For this we require

\[B^2(x) = 4\Lambda(x) \cdot C(x)\] 3.3.5

or

\[4x^6 \alpha^2 + 4/\beta^2 x^2 - 8\alpha/\beta x^4 = -3\alpha/\beta x^4 + 8\alpha x^2\] 3.3.6

or

\[x^4 \alpha^2 + \beta^2 - 2\alpha = 0\] 3.3.7
Now let
\[ \alpha = \sum_{k=0}^{\infty} a_k x^k \]
and
\[ \beta = \sum_{k=0}^{\infty} b_k x^k \]

we have
\[ x^4 \sum_{k=0}^{\infty} \sum_{j=0}^{\infty} a_k a_j x^{k+j} + \sum_{k=0}^{\infty} \sum_{j=0}^{\infty} b_k b_j x^{k+j} - \sum_{k=0}^{\infty} 2a_k x^k = 0 \]

If the coefficients of like powers are collected and equated to zero there results a set of simultaneous nonlinear differential equations for the coefficients \( a_k, b_k \) and no solution is possible without recourse to some ad-hoc computing routine, and in fact it may be that no solution exists.

Thus for this system the method breaks down. The failure of the method in this case is even more surprising in view of the fact that any positive definite function whatsoever is a Liapounov function for this system. The reason for the failure is discussed later in section 3.7.

3.4. The Computing Algorithm

The method of Szego provides a relatively simple computing algorithm provided an additional weak restriction is placed on the system equations in order to eliminate those
systems such as the counter-example of section 3.3 for which
the method fails.

Let the linear parts of the system 3.1.1 be

\[
\begin{align*}
x &= p_1 x + p_1'y \\
y &= q_1 x + q_1'y
\end{align*}
\]

Then in addition to the linearity of \( f \) and \( g \) in \( y \) let
us further demand the satisfaction of the conditions

1. \( p_1 q'_1 - p'_1 q_1 > 0 \)
2. \( p_1 + q'_1 < 0 \) \hspace{1cm} 3.4.2
3. \( q'_1 \neq 0 \)

Conditions 1 and 2 are the conditions for the system to
be asymptotically stable. Condition 3 as we shall see is the
condition necessary to exclude the systems for which the
method fails. (Note that it may be possible to satisfy
condition 3 by interchanging the roles of \( x \) and \( y \).)

We now proceed to construct the function \( V \) of equation
3.1.2. Let us postulate general polynomial forms for \( \alpha(x) \)
and \( \beta(x) \) thus

\[
\begin{align*}
\alpha(x) &= \sum_{j=1}^{a_j} x^{j-1} \\
\beta(x) &= \sum_{j=1}^{b_j} x^{j-1}
\end{align*}
\]

3.4.3

32.
By the assumptions on \( f \) and \( g \) we may write the system equations as

\[
\begin{align*}
\dot{x} &= \sum_{k=1}^{m_f} p_k x^k + y \sum_{k=1}^{m_f} p_k x^{k-1} \\
\dot{y} &= \sum_{k=1}^{m_f} q_k x^k + y \sum_{k=1}^{m_f} q_k x^{k-1}
\end{align*}
\]  

3.4.4

Substituting from 3.4.3 and 3.4.4 into 3.1.2 and 3.1.3 we have

\[
\dot{v} = \Lambda(x) y^2 + B(x) y + C(x) 
\]  

3.4.5

where

\[
\begin{align*}
\Lambda(x) &= \sum_{k=1}^{m_f} \sum_{j=1}^{m_f} p_k b_{j} x^{j-2} + \\
&\quad + \sum_{k=1}^{m_f} \sum_{j=1}^{m_f} p_k b_{j} (j-1) x^{j-2} + \\
&\quad + \sum_{k=1}^{m_f} 2q_k x^{k-1} \\
B(x) &= \sum_{k=1}^{m_f} \sum_{j=1}^{m_f} 2p_k a_{j} x^{j-1} + \\
&\quad + \sum_{k=1}^{m_f} \sum_{j=1}^{m_f} p_k a_{j} (j-1) x^{j-1} + \\
&\quad + \sum_{k=1}^{m_f} \sum_{j=1}^{m_f} q_k b_{j} x^{j-1} + \\
&\quad + \sum_{k=1}^{m_f} \sum_{j=1}^{m_f} p_k b_{j} x^{j-1} + \sum_{k=1}^{m_f} 2q_k x^{k} + \\
&\quad + \sum_{k=1}^{m_f} \sum_{j=1}^{m_f} p_k b_{j} (j-1) x^{k-j-1}
\end{align*}
\]  

3.4.6 3.4.7
and \[ C(x) = \sum_{k=0}^{m} \sum_{j=0}^{l} 2p_k a_j x^{k+j} \]
\[ + \sum_{k=1}^{m} \sum_{j=1}^{l} p_k a_j (j-1) x^{k+j} \]
\[ + \sum_{k=1}^{m} \sum_{j=1}^{l} q_k b_j x^{k+j} \]

3.4.8

For \( A(x) \) to vanish identically we collect terms of like power and equate the coefficients to zero. There results the following set of simultaneous equations:

Term

- **const.**: \( p_1' b_1 + 2q_1' = 0 \)  
  \[ 3.4.9 \]

- **\( x \)**: \( p_2' b_1 + 2p_1' b_2 + 2q_2' = 0 \)
  \[ \ldots \ldots \ldots \ldots \]  
  \[ \ldots \ldots \ldots \ldots \text{etc.} \]

or

\[ b_1 = \frac{-2q_1'}{p_1'} \]

3.4.10

and

\[ b_n = \left\{ \frac{-2q_n' - \sum_{k=1}^{n-1} p_n' x^{n-k+1} b_k}{(n-1) m f} \right\} \frac{1}{n p_1'} \]

3.4.11

\[ n = 2, 3, \ldots \]

Similarly setting \( B(x) \) identically equal to zero we have
\[
\begin{align*}
\text{Term} \\
\begin{align*}
x & : 2p_1^1 a_1 + \\
x^2 & : 2p_2^1 a_1 + 2p_1^1 a_2 + \\
x^3 & : 2p_3^1 a_1 + 3p_2^1 a_2 + 4p_1^1 a_3 + \\
& \ldots \ldots \ldots \\
& + p_1^1 b_1 + \\
& + p_2^1 b_1 + 2p_1^1 b_2 + \\
& + p_3^1 b_1 + 2p_2^1 b_2 + 3p_1^1 b_3 + \\
& \ldots \ldots \ldots \\
& + q_1^1 b_1 + 2q_1 = 0 \\
& + q_2^1 b_1 + q_1^1 b_2 + 2q_2 = 0 \\
& + q_3^1 b_1 + q_2^1 b_2 + q_1^1 b_3 + 2q_3 = 0 \\
& \ldots \ldots \ldots \\
& \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldot}
the algorithm for computing the coefficients $a_k, b_k$ of the Liapounov function $V$ of 3.1.2. Thus

$$V = \sum_{j=1}^{\infty} a_j x^{j+1} + \sum_{j=1}^{\infty} b_j x^j y + y^2 \quad 3.4.15$$

If $V$ is required in the standard form

$$V = \sum_{k=2}^{K+1} \sum_{j=1}^{j-l} a_{kj} x^{k-j+1} y^{j-1} \quad 3.4.16$$

we have

$$a_{k1} = a_{k-1} \quad k = 2, 3, \ldots$$

$$a_{k2} = b_{k-1} \quad k = 2, 3, \ldots \quad 3.4.17$$

$$a_{k3} = 1$$

$$a_{kj} = 0 \quad \text{otherwise}$$

The time derivative $\dot{V}$ is given by

$$\dot{C}(x) = \sum_{k=1}^{n} \sum_{j=1}^{j-k} q_k b_j x^{k+j} + \sum_{k=1}^{n} \sum_{j=1}^{j-k} p_k a_j (j-1)x^{k+j} \quad 3.4.18$$

which has the general form

$$\dot{V} = \sum_{k=2}^{n} d_k x^k \quad 3.4.19$$

and the coefficients $d_k$ are obtained from the coefficients $a_k$ and $b_k$ thus

$$d_n = \sum_{k=1}^{n-1} q_k b_{n-k} + \sum_{k=1}^{n-1} p_k a_{n-k} (n-k+1) \quad 3.4.20$$

$$n = 2, 3, \ldots$$
Figs. 3.1.(a) and 3.1.(b) show the flow diagram for the method while the program is given in appendix 4.2.

3.5. The System Equations in Companion Form

A system is said to be in companion form if the equations may be written as

\[ \begin{align*}
\dot{x}_1 &= x_2 \\
\dot{x}_2 &= x_3 \\
& \vdots \\
\dot{x}_{n-1} &= x_n \\
\dot{x}_n &= f(x_1, x_2, \ldots, x_n, t)
\end{align*} \]

For a system in companion form the equations 3.4.4 become

\[ \begin{align*}
\dot{x} &= y \\
y &= \sum_{k=1}^{mf} a_k x^k + y \sum_{k=1}^{mf} b_k x^{k-1}
\end{align*} \]

Applying the method of Szego to this system the algorithm for \( a_k \) and \( b_k \) becomes

\[ b_n = -2q_n/n, \quad n = 1, 2, \ldots, mf \]

and
\[ a_n = \begin{cases} -2q_n - \sum_{K=1}^{n} b_k a_{n-k+1} \\ \frac{(n+1)}{(n-K+1) mf} \end{cases} \quad 3.5.4 \]

\[ n = 1, 2, \ldots, (2mf-1) \]

The reason for making a point of this special case is that the parameter \( n \) in 3.5.3 and 3.5.4 has a terminal value and \((2mf-1)\) namely \( mf \), whereas in 3.4.14 \( n \) may increase indefinitely. In other words if the system is in companion form the algorithm always terminates for finite \( mf \) whereas for systems not in companion form this is not guaranteed. The reason for this is obvious from a comparison of eqns. 3.4.14 and 3.5.4.

In the former the value of \( a_n \) depends upon the values of the previously determined \( a_k \) \((k < n)\) whereas in the latter the value of \( a_n \) depends only upon the coefficients of the system equations. This means that for systems not in companion form it may be impossible to find a Liapounov function of finite degree which satisfies the strong conditions imposed by the method of Szego. For example it was found to be impossible to construct a finite degree Liapounov function by this method for the system 1.7.4.1. and as is shown in section 3.8 it is not permissible to truncate Liapounov functions obtained by the method of Szego.

The flow diagram for the method for systems in companion form is shown in fig. 3.2 and the program in appendix 3.3.
3.6. Validity in the Linear Case

Any formal method for the construction of Liapounov functions for non-linear systems should be applicable to linear systems. The methods of Ingwerson and Zubov can be shown to possess this characteristic (they are in fact equivalent for linear systems). It is shown here that the method of Szego also has this property.

Consider the linear system 3.4.1 bearing in mind the conditions 3.4.2. From 3.4.10 we have

\[ b_1 = -2q'_1/p'_1 \]  \hspace{1cm} 3.6.1

and from 3.4.13

\[ a_1 = \left\{ \frac{2q'_1(p'_1 + q'_1)/p'_1 - 2q'_1}{2p'_1} \right\} \]  \hspace{1cm} 3.6.2

then \( V \) is given by

\[ V = a_1 x^2 + b_1 xy + y^2 \]  \hspace{1cm} 3.6.3

For this to be a Liapounov function it must be positive definite i.e. we must have

\[ a_1 > 0 \]  \hspace{1cm} 3.6.4

\[ a_1 - b_1^2/4 > 0 \]  \hspace{1cm} 3.6.4

Re-arranging 3.6.2 we have
\[ a_1 = \left\{ \frac{(p_1 q_1' - q_1 p_1') + (q_1')^2}{(p_1')^2} \right\} \quad 3.6.5 \]

which is positive by condition 1 (3.4.2), and

\[ a_1 - \frac{b_1^2}{4} = \left\{ \frac{p_1 q_1' - q_1 p_1'}{(p_1')^2} \right\} \quad 3.6.6 \]

which again is positive by condition 1. It remains to show that \( \bar{V} \) is negative. By 3.4.5 we have

\[ \dot{\bar{V}} = (2p_1 a_1 + q_1 b_1) x^2 \quad 3.6.7 \]

and by 3.6.1 and 3.6.2 we have

\[ \dot{\bar{V}} = 2(q_1 p_1' - q_1 p_1')(p_1 + q_1) x^2 \quad 3.6.8 \]

which is negative by conditions 1 and 2.

The method is thus applicable to those systems to which the methods of Zubov and Ingwerson are applicable provided the additional condition 3 is satisfied.

3.7. Examples

3.7.1. The system 1.7.1.4. (\( \varepsilon = 1.0 \))

\[ \dot{x} = -y \]
\[ \dot{y} = x - y + x^2 y \quad 3.7.1 \]

The only non-zero coefficients are
\[ p_1' = -1 \quad q_1 = 1 \quad q_1' = -1 \quad q_3 = 1 \quad 3.7.2 \]

From 3.4.10, 3.4.11, 3.4.13 and 3.4.14 we have

\[
\begin{align*}
b_1 &= -2 \quad a_1 = 2 \\
b_3 &= 2/3 \quad a_3 = -2/3 \\
a_5 &= 1/9
\end{align*}
\]

3.7.3

The Liapounov function is thus

\[ V = 2x^2 - 2xy + y^2 - 2x^4/3 + 2x^3y/3 + x^6/9 \]

and

\[ \dot{V} = -2x^2(1 - x^2/3) \]

3.7.4

3.7.5

V satisfies the conditions of theorem 1.5.3 in a region about the origin. The RAS is shown in fig. 3.3. The actual domain of attraction is of similar shape but passing through \( x = 2 \) approx. The method thus gives a good approximation to the DOA.

3.7.2. The System 1.7.2.

Fig 3.4. shows the RAS for this system together with the actual DOA. In this case the approximation is not so good but is better than those obtained by the method of Krassovski (figs. 2.1 and 2.2).
3.8. Comments

The method of Szego is well suited to machine computation and possesses the advantage of being simpler to program than either the method of Lugwerson (chapter 4) or Zubov (chapter 5). Also, it is relatively fast since there is only a single set of simultaneous equations (unlike the method of Zubov in which there is a set for each homogeneous part of the Liapounov function) and these equations are in a form suitable for solution by simple recursion.

If the method of Rodden is employed to trace the boundary of the RAS then the method of Szego possesses an attractive feature. In the method of Rodden it is necessary to follow the locus \( V = 0 \) in the state plane. In the general case this locus is curvilinear and may only be traced by the use of a time consuming iterative technique (see chapter 6). For Liapounov functions generated by the method of Szego, however, the locus \( \dot{V} = 0 \) is a straight line and may be traced simply. For systems of order higher than second \( V = 0 \) defines flat instead of curved surfaces.

The main disadvantages of the method are that it is applicable only to a restricted class of systems and that the RAS are not generally so good as those obtained by the method of Zubov (chapter 5).

Another important feature of the method is that unlike
the other methods described, the Liapounov function may not in general be truncated. This is because the derivative is only semidefinite. If \( V \) is a Liapounov function of degree \( n > 2 \) with \( V \) negative definite then the function \( V' \) obtained by truncating \( V \) at degree \( n_1 \) (\( 2 < n_1 < n \)) is also a Liapounov function. That this is not true for the method of Szego is illustrated by the following example. Consider the system

\[
\begin{align*}
\dot{x} &= y \\
\dot{y} &= -x - y + xy
\end{align*}
\]

3.8.1

The Liapounov function by Szego's method is

\[
V = 2x^2 + 2xy + y^2 - x^3 + x^4/4 - x^2y
\]

3.8.2

and

\[
\dot{V} = -2x^2(1 - x/2)
\]

If we truncate \( V \) at degree 2 we obtain the function

\[
V' = 2x^2 + 2xy + y^2
\]

3.8.3

which is of cause still positive definite. The time derivative, however, is

\[
\dot{V}' = -2x^2(1 - y - y^2/x)
\]

which is sign indefinite. \( V' \) is not, therefore, a Liapounov function.
4. **The Method of Ingwerson**

4.1. **Introduction**

The method of Ingwerson is a method for the construction of Liapounov functions for autonomous systems. It is an extension to non-linear systems of the well known method for obtaining Liapounov functions for linear systems (see theorem 1.5.5).

If we have the linear system

\[ x = \Delta x \]  

where \( \Delta \) is a stability matrix, then

\[ V = x^T B x \]  

is a Liapounov function if \( B \) is the solution of the equation

\[ \Delta^T B + B \Delta = -C \]  

where \( C \) is an arbitrary positive definite symmetric matrix.

This function \( V \) will also be a Liapounov function for a non-linear system

\[ \dot{x} = f(x) \]  

provided its first approximation is just 4.1.1.

In general this Liapounov function which takes no account of the non-linearities in the system will not be very
effective, and the RAS will be only a poor approximation to the domain of attraction.

By the method of Ingwerson a Liapounov function of degree higher than second is constructed whose quadratic part is the function of 4.1.2.

Any method for the construction of high degree Liapounov functions ought to ensure that the functions constructed are in fact superior to those which could be obtained simply by linearisation of the system equations otherwise there would be no point in using the method. As is shown later the method of Ingwerson, like the method of Krassovski often results in Liapounov functions less effective than those obtained by linearisation.

4.2. The Method

Consider the system

\[ \dot{x} = f(x) \quad f(0) = 0 \quad x \in E \quad 4.2.1 \]

and assume that \( f \) has a convergent power series expansion about the origin i.e. the system may be written

\[ \dot{x} = Ax + f_2(x) \quad 4.2.2 \]

where \( f_2(x) \) has no linear terms. Let us further assume that \( A \) is a stability matrix.
Differentiating 4.2.1 with respect to time we have

\[ \ddot{x} = A(x) \dot{x} \quad 4.2.3 \]

where \( A(x) \) is the Jacobian matrix; i.e. the matrix of elements \( \frac{\partial f_i}{\partial x_j} \). (Note that \( \dot{A} \) and \( A(x) \) are distinct.)

Equations 4.1.1 and 4.2.3 have the same form so we proceed as for the linear case and set up the equation analogous to 4.1.3 thus

\[ A^T(x)B(x) + B(x)A(x) = -C \quad 4.2.4 \]

where we have indicated that now \( B \) is a function of \( x \). If the elements of this matrix \( B \) were the second partial derivatives of a scalar function \( V \) then \( V \) would be obtainable from \( B \) by integrating twice and would be a Liapounov function. However, \( B \) will not in general have this form. It is possible to obtain a matrix \( B^* \) from \( B \) such that \( B^* \) does satisfy the conditions necessary for it to be the matrix of the second partial derivatives of a scalar, and which gives a Liapounov function which according to Ingwerson 'often gives good results'.

This matrix \( B^* = B^*(x) \) must be symmetric and the elements \( b^*_{ij}(x) \) must satisfy

\[ \frac{\partial b^*_{ij}(x)}{\partial x_k} = \frac{\partial b^*_{ik}(x)}{\partial x_j} \quad 4.2.5 \]
To obtain $B^*(x)$ we first multiply each element of $B(x)$ by the lowest common multiple of the denominators so that each element assumes polynomial form. Then, in each element $b_{ij}(x)$ of this modified matrix, all state variables are allowed to vanish except $x_i$ and $x_j$. This leaves the required matrix $B^*(x)$ which, as a result of the above modifications, may be written as $B^*(x_i, x_j)$. A Lyapunov function may now be obtained by two integrations.

First a vector $T(x)$ is obtained from $B^*(x_i, x_j)$ by an integration of a special type. The elements $t_k$ of $T$ are given by

$$t_k(x) = \sum_{j=1}^{n} \int_{\chi_k \text{ constant}}^{x_j} b_{kj}(x_k, x_j) \, dx_j$$

for $k = 1, 2, \ldots, n$

Ingwerson\(^3^4\) has shown that this vector $T(x)$ satisfies the conditions necessary for it to be the gradient of a scalar function $V(x)$. This function $V(x)$ is obtained by line integration from $x = 0$ to $x = x$ along any path

$$V(x) = \int_{0}^{x} T(x) \, dx$$

This function $V(x)$ is the required Lyapunov function. Note that the matrix $C$ need only be semi-definite in some cases. However if a semi-definite $C$ is used it is
necessary to check that the function \( V \) is a Liapounov function. The method may also be used if the linear parts of the system are only stable instead of asymptotically stable but again it is necessary to check that a Liapounov function has been obtained. If \( A \) is a stability matrix and \( C \) positive definite then a Liapounov function is guaranteed.

4.3. An Example

Consider system 1.7.2

\[
\begin{align*}
\dot{x} &= y \\
\dot{y} &= -x - y + x^3
\end{align*}
\]

Let us choose the positive semi-definite matrix

\[
C = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}
\]

Solving 4.2.4 and clearing the denominators we have

\[
P(x) = \begin{bmatrix} 4-6x^2 & 2 \\ 2 & 2 \end{bmatrix}
\]

This is already in the form required for \( P^*(x_k, x_j) \) and we proceed to the integrations. We have

\[
P(x) = \begin{bmatrix} \int_{0}^{x} (4-6x^2) dx + \int_{0}^{y} 2dy \\
\int_{0}^{x} 2dx + \int_{0}^{y} 2dy \end{bmatrix}
\]
\[ T(x) = \begin{bmatrix} 4x-2x^3 + 2y \\ 2x + 2y \end{bmatrix} \]  

Then

\[ V(x) = \int_0^x (4x - 2x^3 - 2y) \, dx + \int_0^y (2x - 2y) \, dy \]  

\[ (y = 0) \quad (x = x) \]

\[ = 2x^2 - x^4/2 + 2xy + y^2 \]  

\[ \dot{V}(x) = 2x^2(x^2 - 1) \]

This is the same Lyapunov function as obtained by the method of Szego (see section 3.7 example 3.7.2). The RIS is shown in fig. 3.4.

4.4. The Computing Algorithm

Ingwerson has given the \( B(x) \) matrices for systems of up to fourth order for some simple \( C \) matrices and under the assumption that the system equations are in companion form (defined in section 3.5). For the more general case of systems of the form 4.2.1 the \( B(x) \) is given for the second order case only and for only two \( C \) matrices. Since the optimisation procedure described in chapter 7 requires a routine which can construct a Lyapunov function for arbitrary \( C \) such a routine was developed and is described below. This algorithm is applicable to second order systems only for reasons which will be apparent.
Consider the system
\[ \dot{x} = f(x, y) \quad f(0,0) = 0 \]
\[ \dot{y} = g(x, y) \quad g(0,0) = 0 \]

Let us write
\[ f(x, y) = \sum_{l=1}^{\infty} \sum_{j=1}^{\infty} p_{lj} x^{i-l+1} y^{j-l} \]
\[ g(x, y) = \sum_{l=1}^{\infty} \sum_{j=1}^{\infty} q_{ij} x^{i-j+1} y^{j-l} \]

Performing the differentiation of 4.2.3 we have
\[ \ddot{x} = a_{11}(x, y) \dot{x} + a_{12}(x, y) \dot{y} \]
\[ \ddot{y} = a_{21}(x, y) \dot{x} + a_{22}(x, y) \dot{y} \]

where
\[ a_{11}(x, y) = \sum_{l=1}^{\infty} \sum_{j=1}^{\infty} p_{lj} (i-l+1) x^{i-l+1} y^{j-l} \]
\[ a_{12}(x, y) = \sum_{l=1}^{\infty} \sum_{j=1}^{\infty} p_{lj} (j-l-1) x^{i-j+1} y^{j-l} \]
\[ a_{21}(x, y) = \sum_{l=1}^{\infty} \sum_{j=1}^{\infty} q_{ij} (i-j+1) x^{i-j+1} y^{j-l} \]
\[ a_{22}(x, y) = \sum_{l=1}^{\infty} \sum_{j=2}^{\infty} q_{ij} (j-1) x^{i-j+1} y^{j-l} \]

Let us denote the matrices \( \mathbb{B}(x) \) and \( \mathbb{C} \) by
\[ \mathbb{B}(x) = \begin{bmatrix} b_{11}(x, y) & b_{12}(x, y) \\ b_{12}(x, y) & b_{22}(x, y) \end{bmatrix} \]
and \[ c = \begin{bmatrix} c_{11} & c_{12} \\ c_{12} & c_{22} \end{bmatrix} \] 4.4.7

then it is easy to show that the elements \( b_{ij} \) of \( B \) are given by the following (assuming the denominators have been cleared)

\[
\begin{align*}
b_{11}(x,y) &= -c_{11}a_{11}a_{22} - c_{11}a_{22}^2 + c_{11}a_{12}a_{21} \\
&
+ 2c_{12}a_{21}a_{22} - c_{22}a_{21}^2 \\
&
+ 2c_{12}a_{11}a_{22} - 2c_{12}a_{11}a_{22} \\
&
+ c_{22}a_{21}a_{12} - c_{22}a_{12}^2 \\
&
+ 2c_{12}a_{11}a_{12} \\
&
\end{align*}
\]

Since, to obtain \( \bar{B} \) from \( B \), we allow variables to vanish, it is convenient to write the elements \( a_{ij} \) of 4.4.5 in the following form

\[
x = 0 \quad y = 0
\]

\[
a_{11} : \sum_i p_{1i}y^{i-1} \quad \sum_i p_{i1}(i)x^{i-1}
\]

\[
a_{12} : \sum_i p_{i1}(i)y^{i-1} \quad \sum_i p_{12}x^{i-1}
\]

\[
a_{21} : \sum_i q_{1i}y^{i-1} \quad \sum_i q_{i1}(i)x^{i-1}
\]

\[
a_{22} : \sum_i q_{i1}(i)y^{i-1} \quad \sum_i q_{12}x^{i-1}
\]

51
All summations in 4.4.11 are \( i = 1 \ldots mf \).

For \( b_{11}^*(x) \) we use the terms with \( y = 0 \) and from 4.4.8 we obtain

\[
b_{11}^*(x) = -c_{11} \sum_i \sum_j p_{11} b_{12}^i(1)x^{i+j-2} -c_{11} \sum_i \sum_j q_{12} b_{12}^i x^{i+j-2} + c_{11} \sum_i \sum_j p_{12} b_{12}^i(j)x^{i+j-2} + 2c_{12} \sum_i \sum_j q_{11} b_{12}^i(i)x^{i+j-2} -c_{22} \sum_i \sum_j p_{11} b_{12}^i(ij)x^{i+j-2}
\]

4.4.12

or

\[
b_{11}^*(x) = \sum_i \sum_j \left\{ -c_{11} p_{11} q_{12}^i(1) - c_{11} q_{12}^i q_{12}^j \right\} x^{i+j-2} + c_{11} p_{12} q_{12}^i(j) + 2c_{12} q_{11} q_{12}^i(1) - c_{22} q_{11} q_{12}^i(ij) \right\} x^{i+j-2}
\]

4.4.13

For \( b_{12}^*(x,y) \) we use the general forms of 4.4.5 and we obtain

\[
\text{Equation 4.4.12 all the subscript } b_{ij} \text{ should be replaced by } q_{ij}. \]

52
\[ b_{12}(x, y) = \]
\[
\sum_{l=1}^{m_f} \sum_{j=1}^{m_f} \sum_{k=1}^{k+1} \sum_{s=1}^{k+2} \sum_{i=1}^{c_1} q_{ks} (i-j+1)(k-s) y_{i+k-j-s} y_{s+1} y_{j+s} - 2 \]
\[
\sum_{l=1}^{m_f} \sum_{j=1}^{m_f} \sum_{k=1}^{k+1} \sum_{s=2}^{k+2} \sum_{i=1}^{c_1} q_{ks} (i-j+1)(s-1) y_{i+k-j-s+2} y_{s+1} y_{j+s} - 4 \]
\[
\sum_{l=1}^{m_f} \sum_{j=1}^{m_f} \sum_{k=1}^{k+1} \sum_{s=2}^{k+2} \sum_{i=1}^{c_1} q_{ks} (i-j+1)(s-1) y_{i+k-j-s+1} y_{s+1} y_{j+s} - 3 \]
\[
4.4.14 \]

and for \( b_{22}(y) \) we use the terms with \( x = 0 \) giving

\[ b_{22}(y) = \]
\[
-2 \sum_{l=1}^{m_f} \sum_{j=1}^{m_f} p_{i_1} p_{j_1} y_{i+j-2} \]
\[
-2 \sum_{l=1}^{m_f} \sum_{j=1}^{m_f} p_{i_1} q_{j_1} (j) y_{i+j-2} \]
\[
+ 2 \sum_{l=1}^{m_f} \sum_{j=1}^{m_f} p_{i_1} p_{j_1} (j) y_{i+j-2} \]
\[
+ \sum_{l=1}^{m_f} \sum_{j=1}^{m_f} p_{i_1} q_{j_1} (i) y_{i+j-2} \]
\[
-2 \sum_{l=1}^{m_f} \sum_{j=1}^{m_f} p_{i_1} q_{j_1} (i) y_{i+j-2} \]
\[
4.4.15 \]

or

\[ b_{22}(y) = \]
\[
\sum_{l=1}^{m_f} \sum_{j=1}^{m_f} \left\{ -2 p_{i_1} p_{j_1} y_{i+j-2} - 2 p_{i_1} q_{j_1} (j) \right\} \]
\[
+ 2 \sum_{l=1}^{m_f} \sum_{j=1}^{m_f} p_{i_1} p_{j_1} (j) y_{i+j-2} + \sum_{l=1}^{m_f} \sum_{j=1}^{m_f} p_{i_1} q_{j_1} (i) y_{i+j-2} \]
\[
-2 \sum_{l=1}^{m_f} \sum_{j=1}^{m_f} p_{i_1} q_{j_1} (i) y_{i+j-2} \]
\[
4.4.16 \]
We now perform the integration described in 4.2.6 and obtain

\[ t_1(x,y) = \]

\[ \sum_{i=1}^{m_f} \sum_{j=1}^{m_f} \left[ -c_{11} p_{i1} q_{j2}(i) - c_{12} q_{i2} q_{j2} + c_{11} p_{12} q_{j1}(j) + 2 c_{12} q_{i1} q_{j2}(i) - c_{22} q_{i1} q_{j1}(ij) \right] / (i + j - 1) \]

\[ \times (i + j - 1) \]

\[ + c_{22} \sum_{i=1}^{m_f} \sum_{j=1}^{m_f} \sum_{k=1}^{m_f} \sum_{s=1}^{K+1} \left\{ \frac{p_{ij} q_{ks}(i-j+l)(k-s+1)}{(j+s-1)} \right\} x^{i+k-j-s} y^{j+s-1} \]

\[ + c_{11} \sum_{i=1}^{m_f} \sum_{j=1}^{m_f} \sum_{k=1}^{m_f} \sum_{s=2}^{K+1} \left\{ \frac{p_{ij} q_{ks}(j-l)(s-1)}{(j+s-3)} \right\} x^{i+k-j-s+2} y^{j+s-3} \]

\[ -2 c_{12} \sum_{i=1}^{m_f} \sum_{j=1}^{m_f} \sum_{k=1}^{m_f} \sum_{s=2}^{K+1} \left\{ \frac{p_{ij} q_{ks}(i-j+l)(s-1)}{(j+s-2)} \right\} x^{i+k-j-s+1} y^{j+s-2} \]

4.4.17

Similarly for \( t_2(x,y) \) we obtain
\[ t_2(x, y) = \]
\[
\sum_{i=1}^{m} \sum_{j=1}^{m} \left[ \left\{ c_{22} p_{ij} q_{jj, j+1}(i) + 2 c_{12} q_{jj, j+1}(i) \right\} / (i+j-1) \right] + c_{22} p_{i, i+1} q_{jj}(i) + c_{11} p_{i, i+1} q_{jj}(i+j) \}
\]
\[ + c_{22} \sum_{l=1}^{m} \sum_{j=1}^{m} \sum_{k=1}^{K} \sum_{s=1}^{s} \left\{ p_{ij} q_{ks}(i-j+1) (k-s+1) \right\} x^{i+k-j-s+1} y^{j+s-2} - 2 c_{12} \sum_{l=1}^{m} \sum_{j=1}^{m} \sum_{k=1}^{K} \sum_{s=2}^{s} \left\{ p_{ij} q_{ks}(i-j+1) (k-s+1) \right\} x^{i+k-j-s+2} y^{j+s-3} \]
\[ 4.4.18 \]

Finally we must perform the line integration of eqn. 4.2.7 to obtain the Liapounov function \( V(x, y) \) thus
Equation 4.4.19 gives the algorithm for constructing the Liapounov function \( V(x, y) \). From the complexity of the expression it may be appreciated that extension to systems of higher order than second, though possible in principle, would present formidable difficulties.

As the algorithm is self evident from equation 4.4.19 no flow diagram is given. The actual program designated SUBROUTINE INGWER is given in appendix A.4.
4.5. Examples

4.5.1. The System 1.7.3.

Figs. 4.1 to 4.4 show the RIS for this system using various different C matrices. The highest degree of Liapounov which it is possible to construct is 4. Also shown are the quadratic RIS obtained by merely linearising the system equations. It is clear that in the majority of cases the Liapounov functions constructed by Ingwerson's method are inferior to the quadratic Liapounov functions. The regions are, however, an improvement on those obtained by the method of Krassovski (figs. 2.3 and 2.4).

4.5.2 The System 1.7.4.A.

Figs. 4.5 4.6 and 4.7 show the RIS for this system using different C matrices. Again in some cases the application of Ingwerson's method results in Liapounov functions less effective than are obtained by linearisation.

A glance at the scales of the figures shows how extremely small are the RIS compared with the actual DOA which has average radius .55 (see fig. 11.4.).

4.5.3. The System 1.7.1.A. (ε = 1.0)

Figs. 4.8 (a) (b) and (c) show the results of applying the method to this system. Yet again the method has an
adverse effect in some cases although in others the RAS is a good approximation to the DOA, comparable with that obtained by Szego's method.

4.6. Comments

As can be seen from the examples the use of the method of Ingwerson is not justified in many cases by the fact that the regions of asymptotic stability indicated are in fact smaller than those obtained by the much simpler device of linearising the system equations and applying theorem 1.5.5.

The reason for this would appear to be the vanishing of terms in obtaining $B^*$ from $B$ with resultant loss of characterisation of the non-linearities in the equations.

The method is applicable to discontinuous systems although not in its numerical formulation. It would appear that the method is a powerful tool in stability analysis when applied in its original non-numerical formulation being capable of handling a wide variety of systems of orders higher than second provided the system equations are of a reasonably simple nature. For systems of the complexity which demands numerical computation the method is not very effective.
5. **THE METHOD OF ZUBOV**

5.1. **Introduction**

The method of Zubov is a construction procedure for Liapounov functions for systems of the form

\[ \dot{x} = f(x), \quad f(0) = 0, \quad f \in \mathbb{R} \quad 5.1.1 \]

We impose the following conditions: (a) the function \( f \) possesses a convergent power series expansion about the origin and (b) the linear parts of the system are asymptotically stable.

According to theorem 1.5.1 if \( V \) is a Liapounov function for the system then the time derivative \( \dot{V} \) by virtue of 5.1.1 must be negative (semi-) definite. This condition may be imposed by equating the expression for the time derivative to a negative (semi-) definite form thus

\[ \dot{V}(x) = -\dot{\phi}(x) \left\{ 1 + \sum_k f^2(x) \right\} \left\{ 1 - V(x) \right\} \quad 5.1.2 \]

or thus

\[ \dot{V}(x) = -\dot{\phi}(x) \left\{ 1 + \sum_k f^2(x) \right\} \quad 5.1.3 \]

In equations 5.1.2 and 5.1.3 the function \( \dot{\phi}(x) \) is
a positive (semi-) definite form. (It is noted that for the numerical procedure \( \phi(x) \) must be strictly positive definite. In the non-numerical formulation of the method it may be possible to use a \( \phi \) which is only semi-definite to find a Liapounov function for a system for which condition (b) above does not hold i.e. a system which is stable but not asymptotically stable. In such a case however, the \( \phi \) must be chosen carefully and it is not true that any \( \phi \) will suffice. This is the reason for demanding condition (b); if (b) holds and \( \phi \) is positive definite a Liapounov function is guaranteed.)

The factor \( 1 + \sum_k f_k^2(x) \) in 5.1.2 and 5.1.3 is equivalent to defining a new independent variable in 5.1.1 so that the right hand sides are bounded and the solutions are extendible for \(-\infty < t < \infty\). This is automatically satisfied for our systems and the equations 5.1.2 and 5.1.3 may be replaced by the simpler equations

\[
\nabla V^T(x) f(x) = -\phi(x) \{1 - V(x)\} \tag{5.1.4}
\]

\[
= -\phi(x) \tag{5.1.5}
\]

These equations are known as the regular and modified Zubov equations respectively.

The basis of Zubov's method is contained in the following theorems which are presented without proof. The proofs can be found in refs. 12 and 13.
Theorem 5.1.1. (see ref. 12)

The function $V(x)$ satisfying 5.1.4 or 5.1.5 if it exists is a Liapounov function for the system 5.1.1 sufficient to prove asymptotic stability.

Theorem 5.1.2 (see ref. 12)

Let $D$ be the domain of attraction of the system 5.1.1. Then the function $V(x)$ which satisfies 5.1.4 also satisfies

$$0 \leq V(x_0) < 1 \quad 5.1.6$$

for all $x_0$ in $D$.

Theorem 5.1.3. (see ref 13)

If $V(x)$ satisfies 5.1.4 the curve defined by

$$V(x) = 1 \quad 5.1.7$$

is an integral curve of the system 5.1.1 and defines the boundary of the domain of attraction.

If $V(x)$ satisfies 5.1.5 then the curve defined by

$$V(x) = \infty \quad 5.1.8$$

is an integral curve of the system 5.1.1 and defines the boundary of the domain of attraction.
Consider the system 1.7.3. The equations are

\[
\begin{align*}
\dot{x} &= -x + 2x^2y \\
\dot{y} &= -y
\end{align*}
\]

and let us choose

\[
\phi(x, y) = 2x^2 + 2y^2
\]

Then equation 5.1.5 becomes

\[
\frac{\partial V}{\partial x}(-x + 2x^2y) + \frac{\partial V}{\partial y}(-y) = -2x^2 - 2y^2
\]

It may be verified simply that the function

\[
V = y^2 + \frac{x^2}{1 - xy}
\]

satisfies 5.2.3 and by the second part of theorem 5.1.3 the boundary of the domain of attraction is given by equating \(V\) to \(\infty\) i.e. by \(xy = 1\).

5.3. Approximations to the Boundary of the DOA

It will not in general be possible to find an analytic solution to either 5.1.4 and 5.1.5. However it is possible to obtain Liapunov functions which give approximations to the boundary of the DOA i.e. which define regions of asymptotic stability (RAS).
Let us suppose that the Liapounov function $V(x)$ had been obtained by solving 5.1.4 or 5.1.5 and let us also suppose that $V(x)$ were expanded in an infinite power series of form. Let $V'(x)$ be the function obtained by truncation of the expansion for $V(x)$ so that $V'(x)$ has degree $n$ (finite). Then the following theorem applies.

Theorem 5.3.1 (see ref. 13)

$V'(x)$ is a Liapounov function and the RIS $D'$ indicated by it is bounded by the curve

$$V'(x) = c$$

5.3.1

for some constant $c$, and $D' \subset D$.

It is the availability of such power series expansions that makes the method of Zubov suitable for numerical computation. The algorithm is described below for second order systems. Extension to higher order systems is, in principle, straightforward, however it is shown later that the difficulties encountered are formidable.
5.4. The Computing Algorithm

Let the system equations be written as

\[ x = \sum_{i=1}^{mf} \sum_{j=1}^{i+1} p_{ij} x^{i-j+1} y^{j-1} \]  
\[ y = \sum_{i=1}^{mf} \sum_{j=1}^{i+1} q_{ij} x^{i-j+1} y^{j-1} \]  

or

\[ x = f_1 + f_2 + \ldots + f_{mf} \]
\[ y = g_1 + g_2 + \ldots + g_{mf} \]

where the \( f_i \) and \( g_i \) are homogeneous polynomials of degree \( i \) in \( x \) and \( y \). Thus we may write

\[ f_r = \sum_{j=1}^{r+1} p_{rj} x^{r-j+1} y^{j-1} \]  
\[ g_r = \sum_{j=1}^{r+1} q_{rj} x^{r-j+1} y^{j-1} \]  

\( r = 1, 2, \ldots, mf \)

Let the function \( \phi(x, y) \) of 5.1.4 or 5.1.5 be written as

\[ \phi(x, y) = \sum_{i=2}^{mf} \sum_{j=1}^{i+1} r_{ij} x^{i-j+1} y^{j-1} \]  

or

\[ \phi(x, y) = \phi_2 + \phi_3 + \ldots + \phi_{mf} \]

where the \( \phi_i \) are homogeneous polynomials of degree \( i \) in \( x \) and \( y \) while \( \phi_2 \) is a positive definite quadratic form.
Thus we may write

\[ \phi_k(x,y) = \sum_{j=1}^{K+1} r_{kj} x^{k-j+1} y^{j-1} \quad 5.4.7 \]

\[ k = 2, 3, \ldots, m \phi \]

We shall construct a Liapounov function \( V(x,y) \) of the form

\[ V(x,y) = \sum_{i=2}^{m v} \sum_{j=1}^{l+1} a_{ij} x^{i-j+1} y^{j-1} \quad 5.4.8 \]

or

\[ V(x,y) = V_2 + V_3 + \ldots + V_{m v} \quad 5.4.9 \]

where the \( V_i \) are homogeneous polynomials of degree \( l \) in \( x \) and \( y \) so that we may write

\[ V_i = \sum_{j=1}^{l+1} a_{ij} x^{i-j+1} y^{j-1} \quad 5.4.10 \]

\[ i = 2, 3, \ldots, m v \]

then

\[ \frac{\partial V_i}{\partial x} = \sum_{j=1}^{l} a_{ij} (i-j+1) x^{i-j} y^{j-1} \quad 5.4.11 \]

\[ \frac{\partial V_i}{\partial y} = \sum_{j=2}^{l+1} a_{ij} (j-1) x^{i-j+1} y^{j-2} \quad 5.4.12 \]

\[ i = 2, 3, \ldots, m v \]

Equations 5.1.4 and 5.1.5, after substitution from 5.4.2 5.4.6 and 5.4.9, become
\[
\frac{\partial}{\partial x} (v_2 + v_3 + \ldots + v_\text{mv}) (f_1 + f_2 + \ldots + f_{\text{mf}}) + \frac{\partial}{\partial y} (v_2 + v_3 + \ldots + v_\text{mv}) (g_1 + g_2 + \ldots + g_{\text{mf}}) = \\
= - (\phi_2 + \phi_3 + \ldots + \phi_{\text{mf}}) (1 - v_2 - v_3 - \ldots - v_\text{mv}) \\
= - (\phi_2 + \phi_3 + \ldots + \phi_{\text{mf}})
\]

5.4.13 is the regular equation, 5.4.14 the modified. Equating terms of like degree (Zubov has shown that the equations are satisfied for all \(x, y\) in the DOF) we have

\text{degree 2} \quad \frac{\partial v_2}{\partial x} f_1 + \frac{\partial v_2}{\partial y} g_1 = -\phi_2 \quad 5.4.15

\text{degree 3} \quad \frac{\partial v_3}{\partial x} f_1 + \frac{\partial v_2}{\partial x} f_2 + \frac{\partial v_3}{\partial y} g_1 + \frac{\partial v_2}{\partial y} g_2 = -\phi_3 \quad 5.4.16

and so on. These equations provide a recursive method for determining the coefficients \(a_{ij}\) of the Liapounov function. Thus 5.4.15 gives the quadratic part \(V_2\). This is substituted in 5.4.16 from which \(V_3\) is then obtained. This process continues until the highest degree part \(V_\text{mv}\) has been obtained.

Rearranging the equations so that the unknown elements appear on the left hand sides we have
2 \[ \frac{\partial v_2}{\partial x} f_1 + \frac{\partial v_2}{\partial y} g_1 = -\phi_2 \] \hspace{1cm} 5.4.17

3 \[ \frac{\partial v_3}{\partial x} f_1 + \frac{\partial v_3}{\partial y} g_1 = \frac{\partial v_2}{\partial x} f_2 - \frac{\partial v_2}{\partial y} g_2 - \phi_3 \] \hspace{1cm} 5.4.18

\[ \frac{\partial v_n}{\partial x} f_1 + \frac{\partial v_n}{\partial y} g_1 = -\sum_{l=2}^{n-1} f_i \frac{\partial (v_{n-i+1})}{\partial x} \]

\[ -\sum_{l=2}^{n-1} g_i \frac{\partial (v_{n-i+1})}{\partial y} \]

\[ -\phi_n \]

\[ + \sum_{l=2}^{n-2} \phi_i v_{n-i} \] \hspace{1cm} 5.4.19

where the last term is present for the regular procedure and absent for the modified procedure. It is convenient to write

\[ \alpha_n \equiv \frac{\partial v_n}{\partial x} f_1 + \frac{\partial v_n}{\partial y} g_1 \] \hspace{1cm} 5.4.20

\[ \beta_n \equiv \sum_{l=2}^{n-1} \left\{ f_i \frac{\partial (v_{n-i+1})}{\partial x} + g_i \frac{\partial (v_{n-i+1})}{\partial y} \right\} \] \hspace{1cm} 5.4.21

\[ \gamma_n \equiv \sum_{l=2}^{n-2} \phi_i v_{n-i} \quad \delta_n \equiv \phi_n \] \hspace{1cm} 5.4.22
Then we have

degree

\[ \alpha_2 = -\delta_2 \]
\[ \alpha_3 = -\delta_3 - \beta_3 \]
\[ \alpha_n = -\delta_n - \beta_n + (\gamma_n) \quad 5.4.23 \]

where the term in brackets is present for the regular procedure.

It is now necessary to expand the expressions for \( \alpha_n \), \( \beta_n \), \( \gamma_n \) and \( \delta_n \) by virtue of the expressions 5.4.3, 5.4.4, 5.4.5, 5.4.6, 5.4.8, 5.4.11, and 5.4.12. It is easy to show that

\[ \alpha_n = \sum_{j=1}^{n-2} \sum_{k=1}^{2} a_{nj} p_{jk}(n-j+1)x^{2-k+n-j}y^{i+k-2} \]
\[ + \sum_{j=2}^{n+1} \sum_{k=1}^{2} a_{nj} q_{jk}(j-1)x^{3-k+n-j}y^{i+k-3} \quad 5.4.24 \]

\[ \beta_n = \sum_{i=2}^{n-1} \sum_{j=1}^{n-i+1} \sum_{k=1}^{p_{ij} n-i+1, k} (n-1-k+2)x^{n-k-j+2}y^{i+k-2} \]
\[ + \sum_{i=2}^{n-1} \sum_{j=1}^{n-u+2} \sum_{k=1}^{p_{ij} n-i+1, k} (k-1)x^{n-k-j+3}y^{i+k-3} \quad 5.4.25 \]

\[ \gamma_n = \sum_{i=2}^{n-2} \sum_{j=1}^{n-i+1} \sum_{k=1}^{r_{ij} n-i, k} x^{n-k+j+2}y^{i+k-2} \]
\[ \delta_n = \sum_{j=1}^{n+1} r_{nj} x^{n-j+1}y^{j-1} \quad 5.4.27 \]
The equations 5.4.23 in view of the expressions 5.4.24 to 5.4.27 represent \( n-1 \) vector equations. The equation for the \( k \)th degree requires the solution of \( k+1 \) simultaneous equations for the coefficients \( a_{kj} \), \( j = 1, 2, \ldots, k+1 \). If we write the equation for the \( k \)th degree as

\[
DB = 0
\]

where \( D \) is a \((k+1, k+1)\) matrix of elements \( d_{ij} \), \( B \) and \( C \) are \( k+1 \) vectors with elements \( b_1 \) and \( c_1 \) respectively, then it can be shown that the elements of \( D \) and \( C \) may be obtained by the following sequential conditional additions to initially zero values:

\[
\begin{align*}
d_{j+i-1,j} & = d_{j+i-1,j} + p_{1i}(k-j+1) \quad 5.4.29 \\
& \quad \{ j = 1, 2, \ldots, k \} \\
& \quad \{ i = 1, 2 \} \\

d_{j+i-2,j} & = d_{j+i-2,j} + q_{1i}(j-1) \quad 5.4.30 \\
& \quad \{ j = 1, 2, \ldots, k+1 \} \\
& \quad \{ i = 1, 2 \} \\

c_{j+r-1} & = c_{j+r-1} - p_{ij}a_{k-i+1,r}(k-i-r+2) \quad 5.4.31 \\
& \quad \{ i = 2, 3, \ldots, k-1 \} \quad (i \leq mf) \\
& \quad \{ j = 1, 2, \ldots, i+1 \} \\
& \quad \{ r = 1, 2, \ldots, k-i+1 \}
\end{align*}
\]
\[
\begin{align*}
    c_{j+r-2} &= c_{j+r-2} - q_{ij}a_{k-i+1,r}^{(r-1)} \\
    \{i &= 2,3,\ldots,k-1\} \quad (i \leq m_f) \\
    \{j &= 1,2,\ldots,i+1\} \\
    \{r &= 2,3,\ldots,k-1+i\}
\end{align*}
\]

\[
\begin{align*}
    c_j &= c_j - \frac{r_i}{k_i} \\
    j &= 1,2,\ldots,k+1 \quad (k \leq m_f)
\end{align*}
\]

\[
\begin{align*}
    c_{j+s-1} &= c_{j+s-1} + r_{ij}a_{k-1,s}^{i} \\
    \{i &= 2,3,\ldots,k-2\} \quad (i \leq m_f) \\
    \{j &= 1,2,\ldots,i+1\} \\
    \{s &= 1,2,\ldots,k-i+1\}
\end{align*}
\]

where the last set of equations is present for the regular procedure and absent for the modified.

The equation 5.4.28 is solved for the elements $b_i$ which are the coefficients of the $k^{th}$ degree part of the Liapunov function i.e. $b_i = a_{k,i}^{i} \quad i = 1,2,\ldots,k+1$.

The value of $k$ is increased by 1 and the process repeated until the Liapunov function of required degree is obtained.

The flow diagram is shown in fig. 5.1 and the actual program designated SUBROUTINE ZUBOV with the simultaneous equations routine SUBROUTINE GAEI is given in appendix A.5.
5.5. **Examples**

5.5.1. The System 1.7.3.

Figs. 5.1 and 5.2 show respectively the RAS using
\[ \phi = x^2 + y^2 \]
for the regular and modified procedures. The non uniform convergence of the method is obvious in that the 20th degree RAS is in fact inferior to the quadratic RAS in each case. The RAS are, however, considerably better than those obtained by the method of Krassovski (figs. 2.3 and 2.4) or Ingwerson (figs. 4.1 to 4.4).

5.5.2. The System 1.7.2.

Figs. 5.3 and 5.4 show respectively the RAS using
\[ \phi = x^2 + y^2 \]
for the modified and regular procedures. The regular procedure exhibits non-uniformity of convergence whereas the convergence of the modified procedure is uniform for this case. The 20th degree region for the modified procedure is much superior to the RAS of Krassovski (figs. 2.1 and 2.2) or Szego (fig. 3.4).

5.5.3 The System 1.7.5. (\( \alpha = 0.3, \beta = 0.1 \))

Figs 5.5 and 5.6 show the RAS for the modified and regular procedures respectively for \( \phi = x^2 + y^2 \). The two procedures are in this case about equally effective (note the difference in the scales of the figures).
5.5.4 The System 1.7.1.A. (ε = 0.1)

Fig. 5.7 shows the RAS for this system using the modified procedure and \( \phi = x^2 + y^2 \). The convergence here is uniform and the approximation to the DOA is good. The DOA is very nearly circular of radius 2 while the 20th degree RAS has radius 1.82.

5.5.5 The System 1.7.4.

The RAS for this system are shown in figs. 8.3.(a) and 8.3.(b) for two different \( \phi \)-functions. It can be seen that these RAS are very much better than those obtained by the method of Ingwerson (figs 4.5 4.6 and 4.7).

Many other examples of the method are given later in chapters 7 and 8 when the optimisation of the method is discussed.

5.6 Attempted Extension to Systems of Arbitrary Order

The comparison of the various construction procedure indicated that the method of Zubov gives consistently better results than the other methods. In view of this an attempt was made to extend the method to systems of arbitrarily high order. Although the attempt was unsuccessful the formulation is described here in the hope that it may prove useful to future researchers. In any event it indicates clearly
the formidable difficulties involved, and that these difficulties, which are not obvious in the theoretical formulation arise as notational and numerical problems.

No attempt has been made in what follows to make efficient use of computer storage and in the representations of the various polynomials which arise multidimensional arrays are used in which only a few of the elements are non-zero. This storage inefficiency is justified by the simplifications it permits and may later be eliminated after a successful routine has been devised.

A homogeneous polynomial of degree \( n \) of \( n \) variables can be written

\[
f_m(x_1, x_2, \ldots, x_n) = \sum_{m_1=1}^{m+1} \sum_{m_2=1}^{m+1} \cdots \sum_{m_n=1}^{m+1} a(m_1, m_2, \ldots, m_n) x_1^{m_1-1} x_2^{m_2-1} \cdots x_n^{m_n-1} \quad (\sum m_k = m+n)
\]

5.6.1

A non-homogeneous polynomial of degree \( m_0 \) can then be written

\[
f_{m_0}(x_1, x_2, \ldots, x_n) = \sum_{m=1}^{m_0} f_m(x_1, x_2, \ldots, x_n) \quad 5.6.2
\]

Our system of order \( n \) with right hand sides of degree \( m_r \) can then be written
\[ x_1 = \sum_{m=1}^{m_f} f_{m}^{1}(x_1,\ldots,x_n) = \sum_{m=1}^{m_f} f_{m}^{1}(x_1,\ldots,x_n) \]

\[ x_2 = \sum_{m=1}^{m_f} f_{m}^{2}(x_1,\ldots,x_n) = \sum_{m=1}^{m_f} f_{m}^{2}(x_1,\ldots,x_n) \]

\[ \ldots \ldots \ldots \]

\[ x_n = \sum_{m=1}^{m_f} f_{m}^{n}(x_1,\ldots,x_n) = \sum_{m=1}^{m_f} f_{m}^{n}(x_1,\ldots,x_n) \]

where

\[ f_{j}^{i} = \sum_{m=1}^{j+1} \sum_{m_{2}=1}^{j+1} \ldots \sum_{m_{n}=1}^{j+1} a_{i}^{(m_1,\ldots,m_n)} x_1^{m_1-1} x_2^{m_2-1} \ldots x_n^{m_n-1} \]

\[ (\sum_{k=1}^{j+n}) \]

Let us assume that the Liapounov function \( V \) can be written as

\[ V \equiv V(x_1,\ldots,x_n) = V_2 + V_3 + \ldots + V_{nv} \]

where \( V_i \) is a homogeneous polynomial of degree \( i \), so that we may write

\[ V_i = \sum_{m_1=1}^{l+1} \sum_{m_2=1}^{l+1} \ldots \sum_{m_n=1}^{l+1} p(m_1,\ldots,m_n) x_1^{m_1-1} x_2^{m_2-1} \ldots x_n^{m_n-1} \]

\[ (\sum_{k=1}^{i+n}) \]

\[ i = 2,3,\ldots,nv \]

74.
The method of Zubov requires the solution, for \( V \) of one of the two equations

\[
\frac{\partial V}{\partial x_1} f_1 + \frac{\partial V}{\partial x_2} f_2 + \ldots + \frac{\partial V}{\partial x_n} f_n
\]

\[= -\phi (1 - V_2 - V_3 - \ldots - V_{nv}) \quad 5.6.7
\]

\[= -\phi \quad 5.6.8
\]

the first of which is the regular procedure, the second the modified.

In these equations let us assume that can be written as

\[
\phi = \phi_2 + \phi_3 + \ldots + \phi_n\phi \quad 5.6.9
\]

where the \( \phi_i \) are homogeneous polynomials of degree \( i \), while \( \phi_2 \) is a positive definite quadratic form, so that

\[
\phi_i = \sum_{m_1=1}^{i+1} \sum_{m_2=1}^{i+1} \ldots \sum_{m_n=1}^{i+1} r(m_1, \ldots, m_n) x_1^{m_1-1} x_2^{m_2-1} \ldots x_n^{m_n-1}
\]

\[\left( \sum m_k \leq n+1 \right) \quad i = 2, 3, \ldots, n \quad 5.6.10
\]

Our equations are just those of 5.4.23 but written in the new notation. Expanding \( \alpha_k, \beta_k, \gamma_k \) and \( \delta_k \) we have
\[
\alpha_k = \sum_{j=1}^{n} \left( \frac{\partial v_k}{\partial x_j} \right) f_i \quad 5.6.11
\]

\[
\sum_{j=1}^{K+1} \sum_{j=1}^{K+1} \sum_{i=1}^{K+1} \sum_{i=1}^{K+1} \sum_{i=1}^{2} \sum_{i=1}^{2} \sum_{i=1}^{2} \left\{ \right. \\
\left. (\sum m_k = k+n) \right. \\
\left. (\sum q_k = n+1) \right. \\
= \left\{ \right. \\
\left. (\sum m_k = k+n) \quad (\sum q_k = n+1) \right. \\

(m_j - 1)p(m_1, \ldots, m_n) \cdot a^j(q_1, \ldots, q_n) x_1^{m_1+q_1-2} x_2^{m_2+q_2-2} \ldots x_j^{m_j+q_j-3} \ldots x_n^{m_n+q_n-2} \} \quad 5.6.13
\]
\[ \beta_k = \sum_{j=2}^{K-1} \sum_{l=1}^{n} \left( \frac{\partial V_i}{\partial x_1} \right)^i \phi_{k-j+1} \]

\[
= \sum_{j=2}^{K-1} \sum_{l=1}^{n} \sum_{m_1=1}^{j+1} \sum_{m_2=1}^{j+1} \cdots \sum_{m_{i-1}=1}^{j+1} \sum_{m_i=1}^{j+1} \sum_{q_{i-1}=1}^{K-j+2} \sum_{q_i=1}^{K-j+2} \cdots \sum_{q_n=1}^{K-j+2} \left\{ \left( \sum m_k = j+n \right) \left( \sum q_k = n+k-j+1 \right) \right. \\
\left. (m_1-1)p(m_1 \ldots m_n)q^i(q_1 \ldots q_n)x_1^{m_1+q_1-2}x_2^{m_2+q_2-2} \cdots x_i^{m_i+q_i-3}x_{i+1}^{m_{i+1}+q_{i+1}-2} \right\}
\]

\[ \gamma_k = \sum_{l=2}^{K-2} \phi_{l+1}^i V_{k-l} \]

\[
= \sum_{l=2}^{K-2} \sum_{m_1=1}^{l+i} \sum_{m_2=1}^{l+i} \cdots \sum_{m_{i-1}=1}^{l+i} \sum_{m_i=1}^{l+i} \sum_{q_{i-1}=1}^{K-i+1} \sum_{q_i=1}^{K-i+1} \cdots \sum_{q_n=1}^{K-i+1} \left\{ \left( \sum m_k = n+1 \right) \left( \sum q_k = k-i+n \right) \right. \\
\left. r(m_1 \ldots m_n)p(q_1 \ldots q_n)x_1^{m_1+q_1-2}x_2^{m_2+q_2-2} \cdots x_n^{m_n+q_n-2} \right\}
\]

The expression for \[ \delta_k \equiv \phi_k \] is given by 5.6.10.

These expressions for \[ \alpha_k, \beta_k, \gamma_k \] and \[ \delta_k \] together with the equations 5.4.23 provide the algorithm for obtaining the Liapunov function. The coefficients of the \( k \)th degree part of the Liapunov function are

\[ p(m_1, m_2, \ldots, m_n) \left( \sum m_k = k+n \right) \]
Each of the equations 5.4.23 represents a set of simultaneous equations. In the second order case described in section 5.4, these simultaneous equations appeared in a form suitable for solution. Unfortunately, this is not so for higher order cases.

The equations 5.4.23 for the higher order case are hyper-dimensional blocks of simultaneous equations.

For a sixth order system the number of equations is $6^6$ or 46,656. The magnitude of this number clearly shows the impossibility of treating high degree systems using this notation. Of course these numbers are inflated by the inefficient notation used, there being very few non-zero elements in the arrays. While it is possible in principle to arrange the storage of the coefficients so as to make solution feasible the accompanying complexity of the formulation is just as prohibitive as the dimensionality it seeks to eliminate.

The attempt was therefore abandoned at this stage.

5.7. Comments.

The major drawback of the method of Zubov lies in the non-uniformity of convergence of the R.S to the D.C. Thus a low degree Liapounov function may well be more effective than a high degree one.
This is also true of the methods of Ingwerson and Krassovski but is more unfortunate in the case of the Zubov method since it is known that as \( n \) approaches \( \infty \), the R\&S do converge to the DO\&. No such claim is made for the other methods.

Investigation into the reason for this non-uniformity of convergence would appear to be worthwhile and potentially fruitful.

The method of Zubov consumes more computing time than the other methods since the recursive formula for the construction requires the solution of a set of simultaneous equations for each stage inside the loop. However the superior Liapounov functions obtained compensate for this and in any case the time to construct a Liapounov function is negligible compared with the time required for the plotting of the boundary of the R\&S by Rodden's method (see chapter 6).

It is impossible to give an exact figure but typically the time required for the construction of a Liapounov function of 20\(^{th}\) degree is of the order of 90 seconds while the plotting of the boundary of the R\&S may take up to one hour depending upon accuracy requirements, step sizes etc. (All times stated are for an L.C.T 1905 computer).

Like the method of Ingwerson (chapter 4) the treatment of higher order systems appears to be infeasible at present. Some more compact notation is required than that used in
section 5.6 but even if this were available the requirement of converting the hyper-dimensional blocks of simultaneous equations to a single array suitable for solution would appear to present considerable difficulties.
6. THE METHOD OF RODDEN FOR PLOTTING THE RAS.

6.1. Introduction

The two main tasks in the application of the direct method of Liapounov to the stability analysis of non-linear systems are, first the construction of a Liapounov function, and then the establishment of the region of stability associated with this Liapounov function according to the dictates of theorem 1.5.3.

The first of these tasks may be accomplished by the methods described in the previous chapters.

This chapter is concerned with the second. In what follows we draw heavily from Rodden.\(^1\) However, several modifications have been introduced into the original scheme both to increase the speed of computation and to make the program suitable for incorporation into the optimisation procedures of chapters 7 and 8.

Consider fig. 6.1.(a) which shows the configuration in the state space for a Liapounov function \(V(x,y)\) applied to the second-order system

\[
\begin{align*}
\dot{x} &= f(x,y) \quad f(0,0) = 0 \quad (6.1.1) \\
\dot{y} &= g(x,y) \quad g(0,0) = 0
\end{align*}
\]

(Note that the same general principles apply to higher order systems: only the geometrical configuration is more difficult to visualise.)
Since, by definition, \( V \) is positive definite the curves \( V = \) constant are closed in a neighbourhood containing the origin. From the requirements of theorem 1.5.3 it is obvious that the largest region of stability is that bounded by the curve \( V = c_2 \) (note that \( c_1 < c_2 < c_3 \)). This curve is tangent to the locus \( \dot{V} = 0 \).

Originally it was necessary to plot the entire configuration in the state space in order to establish the region of stability. The method of Rodden provides a procedure by which this extremely tedious (and inaccurate) task may be obviated.

The method falls into three main parts. First the \( \dot{V} = 0 \) locus is located. Then this locus is traced until the point of tangency is reached. Finally the curve \( V = c_2 \) is plotted.

6.2. Location of the \( \dot{V} = 0 \) locus.

Starting at some point close to the origin, a spiral path is followed until a region in which \( \dot{V} > 0 \) is reached. Fig. 6.1.(b) shows how the spiral is constructed. At any point \( P(x, y) \) a step is taken along the tangent to the circle centred at the origin passing through \( P \). This defines a new point \( P'(x', y') \) on the spiral. The distance between \( P \) and \( P' \) is such as to cause the point \( P' \) to be \( s_1 \) further from the origin than \( P \). It can be
shown that the coordinates of $P'$ are given by

$$
x' = x - y \left\{ \frac{s_1 + 2s_1 \sqrt{x^2 + y^2}}{\sqrt{x^2 + y^2}} \right\} \sqrt{x^2 + y^2}
$$

$$
y' = y + x \left\{ \frac{s_1 + 2s_1 \sqrt{x^2 + y^2}}{\sqrt{x^2 + y^2}} \right\} \sqrt{x^2 + y^2}
$$

6.2.1

This spiral is followed until a region where $V > 0$ is entered.

As there is no way of knowing 'a-priori' the distance of the $V = 0$ locus from the origin, the step size $s_1$ is at first made extremely small. If a $V > 0$ region is not reached in a certain number of steps (50 in our case), $s_1$ is increased and the spiral restarted. Once a $V > 0$ region is reached in $n$ steps ($n < 50$) the step size is multiplied by $n/20$ and the spiral again restarted. In this way it has been found that the $V > 0$ region is entered in approximately 15 steps. (This is merely to standardise the location of the $V > 0$ region for different $V$ functions for the optimisation procedures of chapters 7 and 8.) In the actual program the step size $s_1$ is stored in address STEP.

Having entered the $V > 0$ region the $V = 0$ locus is now located. Consider fig. 6.1.(b). Point $P_1$ is the first point on the spiral for which $V > 0$ while $P_0$ is the preceding point. It is required to locate the $V = 0$ locus at $Q$. This is achieved as follows. The point $R$ midway

83.
between \( P_1 \) and \( P_0 \) is calculated. If \( R \) is on the same side of the \( \dot{V} = 0 \) locus as \( P_1 \) (\( P_0 \)) it replaces \( P_1 \) (\( P_0 \)). This bisection of the line joining \( P_1 \) and \( P_0 \) is continued until the distance between \( P_1 \) and \( P_0 \) is small enough and \( Q \) located to within some predetermined accuracy. In our case \( Q \) is considered to have been located when the distance between \( P_1 \) and \( P_0 \) becomes less than .01 times the radius to \( R \).

This part of the procedure is shown in flow diagram form in figs. 6.3.(a) and 6.3.(b). Terminal A indicates the entering of a \( \dot{V} > 0 \) region and termination of the spiral. Terminal B indicates that the region of negative \( \dot{V} \) is either non-existent or so small as to be negligible. Terminal C indicates the location of the \( \dot{V} = 0 \) locus at \( Q \) and the initiation of the search for tangency.

6.3. Location of the Point of Tangency.

Consider fig. 6.2. The \( \dot{V} = 0 \) locus has been located at \( Q \). It is required to find the tangency point \( T \). This is achieved by iterating along the \( \dot{V} = 0 \) locus. The direction from \( Q \) to \( T \) may be found by evaluation of the vector \( \mathbf{H} \) given by

\[
\mathbf{H} = \frac{\mathbf{VV}}{|\mathbf{VV}|} - \left( \frac{\mathbf{V}}{|\mathbf{VV}|} \cdot \frac{\mathbf{V}}{|\mathbf{VV}|} \right) \cdot \frac{\mathbf{VV}}{|\mathbf{VV}|}
\]

6.3.1.
This is the component of the normalised gradient of $V$ perpendicular to the gradient of $V$. The vector $H$ always points away from the tangency point (at least for the simple configuration of fig. 6.2.; more complicated configurations for which this is not true are discussed later).

Thus, a step of size $s_3$ is taken from $Q$ in the direction of $-H$ to the point $A_1$. This new point will not be on the $V = 0$ locus which must be relocated at $B$ by iterating along $V \dot{V}$ in steps of initial size $s_4$. This stepping and iterating is continued until the point $T$ is passed. This is indicated by the changing of the direction of the vector $H$, i.e. if $H_1$ and $H_2$ are two $H$ vectors computed on different sides of the tangency point $T$, the scalar product $H_1 \cdot H_2$ is negative. When this stage is reached the size of the step $s_3$ is decreased successively each time $T$ is passed until $s_3$ becomes so small that $T$ can be considered to have been located to within some predetermined accuracy. In the program given in appendix A.6, the process is terminated when $s_3$ becomes smaller than 0.01 times the radius to $T$.

Rodden describes the case where the gradient of $V$ vanishes at the tangency point. This is illustrated in fig. 6.4. In such a case the vector $H$ changes direction abruptly at $T$. Fig. 6.5. represents the 'direction' of the $H$ vector as a function of the distance along the $V = 0$
locus. (Note that fig. 3.4. shows a particular example of this
type. \( \nabla V \) vanishes at \( x = 1, y = -1 \) and \( x = -1, y = 1 \).)

Such cases presented a considerable problem to the
method in its original formulation. This problem arose
because Rodden used the size of \( \| \) as the criterion for
terminating the iteration along the \( \hat{\nabla} = 0 \) locus. It is
obvious that for the case represented in fig. 6.4. the
iteration will never terminate and the procedure will
oscillate about the tangency point.

To overcome this problem Rodden proposed a lengthy
and time consuming procedure. This involved noting the
rate of change of \( \| \) along the \( \hat{\nabla} = 0 \) locus and, when it
appeared that \( \| \) would go to zero in a few steps, moving to
a neighbouring locus \( \hat{\nabla} = \varepsilon \) where \( \varepsilon \) is a small negative
constant.

However this procedure is unnecessary. It is not
required to locate the point at which \( \| \) has magnitude smaller
than some value. What is required is to locate the point \( T \)
to within some tolerance. Thus it is necessary merely to
keep a record of the step size \( s_3 \) and to terminate the iter-
ation when \( s_3 \) becomes smaller than some predetermined value
irrespective of the magnitude of \( \| \).

Rodden also describes a somewhat complicated procedure
for modifying the step size \( s_3 \) as \( T \) is approached. This
involves computing the partial derivatives of \( \| \). It has
been found that this time-consuming procedure is not necessary and that division of $s_3$ by 3 each time $T$ is passed is quite sufficient.

The two modifications described above greatly increase the speed of computation. Without them the procedure would be too slow for incorporation in the optimisation procedures of chapters 7 and 8.

Figs. 6.3.(c) and 6.3.(d) show the flow diagram for this part of the procedure. (Note that in the program of appendix A.6 the values of $s_3$ and $s_4$ are stored in addresses $S$ and $SS$ respectively while the initial value of $s_4$ which is used for reference is stored in ESS).

Terminals D and E indicate respectively the successive location of the locus $V = 0$ and the location of the tangency point.

6.4. **Tracing the Boundary of the RAS**

Having located the point of tangency $T$ to within the specified tolerance the value of $V$ is computed. This gives the constant $c$ such that $V = c$ defines the boundary of the RAS.

The boundary $V = c$ is traced by a stepping and iterating procedure similar to that used in tracing the $V = 0$ locus. From some point $x_0$ on $V = c$ the next point is predicted by

87.
the truncated Taylor expansion

\[
X = X_0 + \left( \frac{dX}{ds} \right)s + \frac{1}{2} \left( \frac{d^2X}{ds^2} \right)s^2
\]

where the derivatives are calculated at \( X_0 \). In 6.4.1 \( s \) is the distance along \( V = c \).

We restrict the remainder of the discussion to second order systems.

From geometrical considerations we have

\[
\frac{dx}{ds} = \begin{bmatrix}
\frac{dx}{ds} \\
\frac{dy}{ds}
\end{bmatrix} = \begin{bmatrix}
\frac{\partial V}{\partial y} & \left( \frac{\partial V}{\partial x} \right)^2 + \left( \frac{\partial V}{\partial y} \right)^2
\end{bmatrix}^{-\frac{1}{2}}
\]

If \( R \) is the radius of curvature of the locus \( V = c \) we have

\[
\frac{1}{R} = - \left\{ \frac{d^2V}{ds^2} \right\} \bigg/ \left\{ \frac{\partial V}{\partial n} \right\}
\]

where \( n \) is the normal displacement. Further we have

\[
\frac{d^2V}{ds^2} = \frac{d^2V}{dx^2} \left\{ \frac{dx}{ds} \right\}^2 + 2 \frac{d^2V}{dx \, dy} \left\{ \frac{dx}{ds} \right\} \left\{ \frac{dy}{ds} \right\} + \frac{d^2V}{dy^2} \left\{ \frac{dy}{ds} \right\}^2
\]

\( \frac{\partial V}{\partial n} \) is just \(|V_V|\) and

\[
\frac{d^2X}{ds^2} = \frac{1}{R} \frac{V_V}{|V_V|}
\]
The new point obtained from 6.4.1 will, because of the truncation of the Taylor series, be displaced from the locus \( V = c \). An iteration procedure along \( \nabla V \) is instituted to relocate \( V = c \) to within some specified accuracy. This iteration initially has the step size \( s_5 = s/5 \), and \( s_5 \) is successively diminished each time \( V = c \) is crossed.

An important point to note is that neither fig. 6.1.(a) nor 6.4 represent the most general configuration in the state plane. Consider fig. 6.6 which displays the configuration for the system 1.7.3 with Liapounov function

\[
V(x,y) = 0.2222x^2 - 0.16667xy + 0.2222y^2 \\
- 0.02469x^4 + 0.25926x^3y - 0.14667x^2y^2 \\
+ 0.03704xy^3 - 0.02469y^4
\]

6.4.6

As can be seen there are four points of tangency and, unlike the configuration of fig. 6.1.(a), these points are not all located on the same \( V = \) constant locus. The actual RAS is bounded by the curve \( V = 0.165 \) as indicated by tangency point \( A \) (at which the gradient of \( V \) vanishes). If tangency point \( A' \) were to be located by the procedure described above, the spurious value 0.24 would be attributed to \( V \) and during the subsequent plotting of the \( V = 0.24 \) curve a region in which \( \dot{V} > 0 \) would be entered at \( C \). It is therefore necessary, at each point on the \( V = c \) curve, to evaluate \( \dot{V} \) and, if it becomes positive, to initiate a new
search for the correct point of tangency. Rodden does not appear to recognise this contingency for second order systems although he does for systems of higher order.

Another minor modification to the method is necessary for the optimisation procedures of chapters 7 and 8. Since it is not 'a priori' known how many steps will be required to complete exactly one circuit of the origin following the \( V = c \) locus, an automatic procedure must be introduced to terminate the trace. If \((x,y)\) is the tangency point, the straight line through the origin passing through \((x,y)\) has gradient \( m = y/x \). Let \((x_1,y_1)\) and \((x_2,y_2)\) be successive points on the boundary. Then, if \((y_1/x_1 - m)\) and \((y_2/x_2 - m)\) have different signs, the two points are on different sides of the line \( y = mx \) which passes through the tangency point. The procedure is thus terminated on the second occasion that this situation occurs. (The first time of course being when the trace crosses \( y = mx \) radially opposite the tangency point).

For the optimisation procedures of chapters 7 and 8 it is necessary to have some measure of the size of the region of stability. We take as this measure the average radius of the boundary. As each successive point is located on the curve \( V = \text{constant} \) the radius is computed and added to a store initially set zero, and a counter is increased by one to record the number of points. At termination of the trace the average radius is obtained.
by dividing the sum of the radii by the number of points. This part of the procedure is depicted in figs. 6.3.(e) and 6.3.(f) in flow diagram form.

The entire procedure, designated SUBROUTINES VALUE, is given in appendix A.6. This main subroutine calls various other subroutines thus:

- SUBROUTINE PXTANA to compute $V_x$ and $V_y$
- SUBROUTINE PXTAND to compute $V_{xx}$, $V_{xy}$ and $V_{yy}$
- SUBROUTINE PXTANB to compute $f$ and $g$
- SUBROUTINE PXTANC to compute $f_x$, $f_y$, $g_x$, and $g_y$

where subscripts denote partial derivatives. $f$ and $g$ are the right hand sides of the system equations thus:

\begin{align*}
\dot{x} &= f(x,y) \\
\dot{y} &= g(x,y)
\end{align*}

6.5. **Two Pathological Configurations.**

Fig. 6.7 shows the configuration in the state plane of system 1.7.3. for the Liapounov function

\[ V = 2x^2 - 1.5xy + 1.25y^2 - 0.25x^4 - 0.5x^3y \]
As can be seen there are four tangency points (the figure is radially symmetric). The RAS is bounded by \( V = 3.93 \) as indicated by tangency point \( A \). If however, tangency point \( A' \) were located by the program the method could fail depending upon which direction the curve \( V = 5 \) were traced. Thus if the trace were to adopt the direction \( A'B \) the method would succeed since a region of \( \dot{V} > 0 \) would be entered and the correct tangency (radially opposite \( A \)) would be located. If however the trace took the direction \( A'C \) the method would fail since the trace would continue along \( A'C \) without limit.

The only way to overcome this problem is to try both directions simultaneously but of course this doubles the computing time.

Fig. 6.8 shows another configuration which causes great difficulty and which is not discussed by Redden. (The diagram is not accurate and is not drawn to scale for clarity). The actual RAS is bounded by \( V = .15 \). If however, the \( \dot{V} = 0 \) locus were located at \( P \), the \( \mathbf{H} \) vector here points towards the RAS instead of away from it (see section 6.3.) The \( \dot{V} = 0 \) locus would therefore be traced in the wrong direction. This is due to the reversal of the gradient of \( V \) at point \( B \). No effective way round this particular problem is known at present. Fortunately, the occasions on which this type of configuration occur are rare.
7. **OPTIMISATION OF THE CONSTRUCTION PROCEDURES.**

7.1. **Introduction.**

Of the methods for constructing Liapounov functions described in the foregoing chapters, the methods of Krassovski, Ingwerson and Zubov possess an important and potentially powerful feature. Unlike the method of Szego these methods admit of optimisation such that the Liapounov function indicating the RAS of maximal size may be obtained.

The feature common to these three methods which facilitates optimisation is that each depends upon the initial choice of certain arbitrary coefficients. The Liapounov function constructed, and hence the size of the RAS, is a function of these coefficients.

It is described below how a direct search procedure may be applied to find that set of coefficients which provides the Liapounov function whose RAS is of maximum size.

7.2. **The Size of the RAS as a Function of Certain Coefficients.**

Consider the methods of Ingwerson (chapter 4) and Krassovski (chapter 2). To generate a Liapounov function by either of these methods requires, initially, the solution of the Liapounov matrix equation
\[ T \mathbf{A} \mathbf{B} + \mathbf{B} \mathbf{A} = -\mathbf{C} \tag{7.2.1} \]

for the elements \( b_{ij} \) of the matrix \( \mathbf{B} \). (In 7.2.1 \( \mathbf{A} \) is the matrix of the linear parts of the system and \( \mathbf{C} \) is an arbitrary positive definite symmetric matrix.)

For the second order case the matrix \( \mathbf{C} \), given by

\[
\mathbf{C} = \begin{bmatrix}
    c_{11} & c_{12} \\
    c_{12} & c_{22}
\end{bmatrix}
\tag{7.2.2}
\]

has three coefficients \( c_{11}, c_{12} \) and \( c_{22} \) which are arbitrary but subject to the conditions

\[
c_{11} > 0 \tag{7.2.3}
\]

\[
c_{11} c_{22} > c_{12}^2
\]

which ensure the positive definiteness of \( \mathbf{C} \).

Consider now the method of Zubov (chapter 5). For the generation of a Liapounov function \( V(x) \) the method requires the solution of one of the equations

\[
\nabla V(x)^T \mathbf{f}(x) = - \varphi(x) \left\{ 1 - V(x) \right\} \tag{7.2.4}
\]

or

\[
\nabla V(x)^T \mathbf{f}(x) = - \varphi(x) \tag{7.2.5}
\]

where \( \varphi(x) \) is a positive definite form but otherwise
arbitrary. For the second order case, and assuming \( \phi(x) \) to be a quadratic form we may write

\[
\phi(x) \equiv \phi(x,y) = c_{11}x^2 + 2c_{12}xy + c_{22}y^2
\]

Again therefore the Liapounov function is dependent upon three coefficients \( c_{11}, c_{12}, c_{22} \) which are chosen to satisfy the constraints.

It is noted here that the function \( \phi(x,y) \) is not necessarily quadratic but may be of arbitrary degree. A general function of degree \( n \) may be written

\[
\phi(x,y) = \sum_{i=2}^{n} \sum_{j=1}^{i+1} c_{ij}x^{i-j+1}y^{j-1}
\]

and the Liapounov function is then a function of the \((n + 1)(n + 2)/2\) coefficients \( c_{ij} \) which are now subject to the conditions

\[
c_{21} > 0
\]

\[
c_{21}c_{23} > c_{22}^2/4
\]

The optimisation procedures will however be described below for the case of three arbitrary coefficients always bearing in mind that the Zubov procedure may be extended to larger numbers of coefficients.
7.3. **Optimisation by Direct Search**

The object of constructing a Liapounov function is to obtain a RAS which is as good an approximation to the actual DOA as possible. In general the RAS will not be the entire DOA but it will of necessity be contained in it. A measure, therefore, of the efficiency of a particular Liapounov function is the area or the average radius of the RAS indicated by it.

If the area or average radius is denoted by $A$, dependence of $A$ on the coefficients $c_{11}$, $c_{12}$ and $c_{22}$ described earlier is denoted by writing

$$A = \Lambda(c_{11}, c_{12}, c_{22})$$

7.3.1

The three coefficients $c_{11}$, $c_{12}$ and $c_{22}$ may be viewed as defining a point in 3-dimensional space, and a direct search may be instituted in this space to find the parameter set $(c'_{11}, c'_{12}, c'_{22})$ such that

$$\Lambda(c'_{11}, c'_{12}, c'_{22})$$

7.3.2

is of maximum size.

The basic necessity is that $\Lambda(c_{11}, c_{12}, c_{22})$ should be able to be computed at any point which satisfies the constraints 7.2.3 or 7.2.8. This is achieved as follows; first the coefficients $c_{11}$, $c_{12}$ and $c_{22}$ are introduced into the
appropriate construction procedure, then the Liapounov function is introduced into the procedure for Rodden's method (chapter 6) and the boundary of the RIS is traced and its average radius obtained.

The direct search procedure used is the modified Simplex method of Nelder and Mead. The reason for choosing this method is that the partial derivatives $\partial A / \partial c_{ij}$ are not available so that gradient methods are inapplicable (unless the gradient is found by perturbation). Also, the modified Simplex method requires only one function evaluation at each point on the route to the extremum, a feature desirable when the function evaluation is a lengthy process. Of course monotonic convergence to the extremum is not achieved.

To handle the constraints a simple penalty function is introduced. If at any stage one of the vertices of the simplex enters a region in the space for which the constraints are not satisfied, the function constructed by the construction procedure would not be a Liapounov function and the method of Rodden would fail. At such a point therefore the policy adopted is to assign the value zero to $A$. Since, for a Liapounov function, $A > 0$ this policy ensures that any point which crosses a constraint has associated with it the worst function value. Then, by the modified Simplex policy, it will be this point which is reflected to bring it back into the permissible region.
As it was not obvious what criterion to use to terminate the search when the extremum had been located with sufficient accuracy, it was decided merely to incorporate a time check and to terminate the search after a certain time had elapsed. If it transpired that the extremum had not been located after this time the search could be restarted at the final point.

The simplified flow diagram for the method is shown in fig. 7.1 and the modified Simplex method is discussed briefly in appendix A.13. The actual program for the optimisation procedure is shown in appendix A.7. It is designated SUBROUTINE SILOPT and in the form shown it is suitable for optimisation of Zubov's method. By making minor alterations and changing the subroutines called by the program it is also suitable for optimising the methods of Krassovski and Ingwerson.

It is emphasised that in the optimisation of Zubov's method the program is able to deal with $\phi$-functions of degree higher than second.

7.4. Examples

7.4.1. The System 1.7.3 (a)

Figs. 7.2.(a) and 7.3 show the optimum RIS obtained for this system for 4th and 6th degree Liapounov functions constructed by Zubov's method (regular). Fig 7.4 shows the
optimum 4th degree Liapounov function obtained by Ingwerson's method. The improvement in the size of the RIS is clearly seen, and is more marked in the case of the Zubov procedure. The RIS of fig. 7.3 is, to the author's knowledge, better than any other hitherto obtained.

7.4.2. The System 1.7.4.\(a\)

Fig. 7.5 shows the optimum 4th degree Liapounov function for this system using the regular Zubov procedure. Figs. 7.6 and 7.7 show the optimum 4th and 6th degree Liapounov functions by Ingwerson's method. The improvement in area is particularly marked, although the method of Zubov still gives the better Liapounov functions.

7.4.3. The System 1.7.2 \(a\)

Fig. 7.8 shows the optimisation of the regular Zubov method for this system for 6th degree Liapounov functions. Again the improvement is considerable. Figs. 7.9 and 7.10 show the optimisation of Ingwerson's method for 4th and 6th degree Liapounov functions. In this case there is not much improvement.

7.4.4. The System 1.7.5. (\(\alpha = .2, \beta = .075\))

Fig. 7.11 shows the optimum 6th degree Liapounov function obtained by the modified Zubov procedure. The optimum RIS is 4 times greater in area than the initial guess.
7.4.5. The System 1.7.1.4.

Fig. 7.12 shows the optimum Liapounov function obtained by the method of Ingwerson for this system with $\varepsilon = 0.1$. The actual DOA is also shown. Although the improvement is not very great the optimum RAS is much closer to the DOA than is the region of the initial guess.

Fig. 7.13 shows the optimisation of Ingwerson's method for 6th degree Liapounov functions for this system with $\varepsilon = 1.0$. In this case the improvement is quite dramatic. The RAS for the initial guess is vanishingly small while the optimum RAS is very close to the actual DOA.

Fig. 7.13.(a) shows the optimisation of Krassovski's method for this system with $\varepsilon = 0.1$. The degree of Liapounov function is 6. The improvement is obvious.

7.5. Sequential Optimisation of the Zubov Procedure

As pointed out in section 7.3 it is possible to use $\phi$-functions of degree higher than second in the optimisation of Zubov's method. However if a $\phi$-function of degree $m\phi$ is used the space in which the search is conducted has dimension $(m\phi + 1)(m\phi + 2)/2 - 3$. This limits the use of the method to low degree $\phi$-functions. To overcome this problem it was decided to investigate the 'sequential' optimisation of the Zubov procedure. Suppose the degree of Liapounov function desired is of degree $m\phi$. Then we may adopt the following procedure. First, a search is made using a second
degree $\phi$-function (this requires a space of three dimensions. Having found the optimum second degree $\phi$, this is held and a search for the optimum 3rd degree part of $\phi$ is pursued. (This requires a space of 4 dimensions). Thus each homogeneous part of $\phi$ is optimised in sequence starting with degree 2. If $m\phi$ is the degree of the $\phi$-function, the procedure requires $(m\phi - 1)$ distinct searches in spaces of dimension $3, 4, \ldots, m\phi + 1$.

Unfortunately, this method was found to be ineffective due partly to the non-uniform convergence of the method of Zubov, and partly due to the prohibitively long computing time required.

The program, designated SUBROUTINE CYRIL is given in appendix A.8, while a simplified flow diagram is given in fig. 7.16.

7.6. Examples of Sequential Optimisation

7.6.1. The System 1.7.3.

In fig. 7.14, $A$ is the initial guess for 4th degree Liapunov function. $B$ represents the optimum RIS obtained using 2nd degree $\phi$. This optimum quadratic part was held and a search instituted for the optimum 3rd degree part. The result was $C$ which as can be seen is in fact inferior even to the initial guess.
7.6.2. The System 1.7.4.4.

Fig. 7.15 shows the method applied to this system. However, in this case, no optimisation was carried out on the quadratic part of \( \varphi \). This is held at \( c^2/2 + \Theta^2 \). The optimisation was carried out on the 3rd degree part of \( \varphi \) and there resulted the RIS B which is a reasonable improvement.

7.7. Comments

The optimisation procedures certainly result in Liapunov functions with enhanced properties. This is of course gained at the expense of computing time. It has been found that the number of steps required to locate the optimum with sufficient accuracy is of the order of 20 to 30. Since the tracing of the boundary for a particular 6th degree Liapunov function may be about 200 seconds, the total time may be about 1 hour. (i.e.T. 1905).

If the search encounters Liapunov functions with the pathological configurations of section 6.5, the method may fail. It has been the experience of the author that success is achieved in about 50% of the attempts.

The sequential optimisation looked promising but unfortunately, as discussed in section 7.5 this promise was not fulfilled.
8. **THE METHOD OF WEISSENBERGER**

8.1. **Introduction**

The method of Weissenberger is a procedure for the construction of optimum Liapounov functions. As such it bears considerable similarity to the optimisation methods described in chapter 7. Originally formulated as a method for dealing with relay control systems, the method is applicable to continuous systems. Unfortunately, as will be shown, the method is restricted to Liapounov functions of low degree. Weissenberger proposed a method for dealing with high degree Liapounov functions, based upon what we shall call Weissenberger's conjecture. It will be shown that Weissenberger's conjecture is in fact invalid and that the high degree Liapounov functions so constructed do not possess optimal character.

8.2. **The Method**

Consider the second order continuous system

\[
\begin{align*}
\dot{x} &= f(x,y) & f(0,0) &= 0 \\
\dot{y} &= g(x,y) & g(0,0) &= 0
\end{align*}
\]

8.2.1

and the Liapounov function
This Liapounov function, of degree $mv$, has

$$N_c = (mv + 1)(mv + 2)/2 - 3$$

8.2.3

coefficients $a_{ij}$. The size of the RIS indicated by this Liapounov function is a function of these coefficients. If $\Lambda$ is again the average radius of the RIS we may write

$$\Lambda \equiv \Lambda(a_{21}, a_{22}, a_{23}, a_{31}, \ldots, a_{mv, mv+1})$$

8.2.4

The coefficients $a_{ij}$ may be viewed as defining a point in the parameter space of dimension $N_c$ and in a manner similar to that described in chapter 7 a direct search may be made for the optimal set.

It is emphasised that the difference between this method and the method of chapter 7 is that here no construction procedure is involved. The coordinates of the space in which the search is made are the actual coefficients of the Liapounov function itself. This leads to a problem of dimensionality, in that to construct a Liapounov function of degree $mv$ the dimension of the space is $N_c$ (see 8.2.3) which rises rapidly with $mv$. Thus to construct a Liapounov function of $6^{th}$ degree would require a space of dimension 25. This is completely infeasible considering the time required for a single function evaluation (evaluation of $\Lambda$ by Rodden's
method).

Another aspect of the method lies in the constraints imposed. Not only must the coefficients $a_{ij}$ be such as to ensure the positive definiteness of $V$ but also the negative definiteness of $\dot{V}$. The first of these constraints is equivalent to the inequalities

$$a_{21} > 0$$

$$a_{21}a_{22} > \frac{a_{22}}{4} \quad 8.2.5$$

The second constraint may be handled by solving the Liapounov matrix equation

$$\bar{A}^T B + B \bar{A} = -C \quad 8.2.6$$

for $C$, where $\bar{A}$ is the matrix of the linear parts of the system 8.2.1 and $B$ is the matrix

$$B = \begin{bmatrix} a_{21} & \frac{a_{22}}{2} \\ \frac{a_{22}}{2} & a_{22} \end{bmatrix} \quad 8.2.7$$

if $C$ is given by

$$C = \begin{bmatrix} c_{11} & c_{12} \\ c_{12} & c_{22} \end{bmatrix} \quad 8.2.8$$
the second constraint is equivalent to the inequalities

\[ c_{11} > 0 \]

\[ c_{11} c_{22} > c_{12}^2 \]  \hspace{1cm} 8.2.9

As in chapter 7 a penalty function is introduced to handle these constraints.

The direct search procedure used by Weissenberger in his original presentation was the steepest ascent gradient method, the gradient being found by perturbation. However this procedure requires several function evaluations at each point on the way to the optimum. It was therefore decided to use the modified Simplex method of Melder and Neudt as was used in the methods of chapter 7.

The flow diagram for the method is shown in fig. 8.1 and the actual program is given in appendix A.9.

It is easy to see that the method of Weissenberger (for continuous systems) is just a special case of the optimisation of Zubov's method (chapter 7). Thus consider the optimisation of Zubov's method. The main equation, for second order systems and using the regular procedure is

\[ \frac{\partial v}{\partial x} f + \frac{\partial v}{\partial y} g = -\phi(1-V) \]  \hspace{1cm} 8.2.10

If the Liapounov function and the function \( \phi \) have the
same degree then there is a 1-1 relationship between them. Therefore to generate a Liapounov function it is immaterial whether the optimisation is carried out in the coefficient space of the Liapounov function or that of the $\phi$-function. Of course the former will be marginally faster since it requires no construction procedure. The program for Weissenberger's method could equally well, therefore, be that of appendix A.7 with the proviso that the degree of $\phi$ is the same as the degree of Liapounov function.

Before illustrating the method by examples we discuss the device proposed by Weissenberger to permit the construction of high degree Liapounov functions.

8.3. **Weissenberger's Conjecture**

As shown above the major drawback to the method of Weissenberger is the high dimensionality of the space in which the search is conducted. This effectively limits the method to Liapounov functions of degree $\leq 3$. To overcome this Weissenberger conjectured that higher degree Liapounov functions could be obtained by the following procedure and that such Liapounov functions would possess enhanced properties.

1. The optimum Liapounov function of 2nd degree is found by the procedure described in the previous chapter.
This we call $V_2^{\text{opt}}$.

(2). The time derivative of this function is obtained. If we denote this by $\dot{V}^{\text{opt}}$ we have

$$\dot{V}^{\text{opt}} = \frac{\partial V_2^{\text{opt}}}{\partial x} f + \frac{\partial V_2^{\text{opt}}}{\partial y} g$$

8.3.1

(3). The quadratic part $V_2^{\text{opt}}$ of $\dot{V}^{\text{opt}}$ is isolated and used as the function $-\phi$ in the Zubov equation (either 5.1.4 or 5.1.5).

(4). The Zubov equation is solved for a Liapounov function of arbitrary order.

The high degree Liapounov function so obtained has as its quadratic part the optimum quadratic Liapounov function $V_2^{\text{opt}}$. Weissenberger conjectured that it should itself enjoy enhanced properties.

That this conjecture is invalid is illustrated by the following examples which also illustrate the direct use of Weissenberger's method for generating low degree Liapounov functions indicating R.H.S of optimum size.
8.4. **Examples**

8.4.1. *The System 1.7.3*

Consider fig. 8.2. Contour A is the RAS for the initial guess

\[ V = 2x^2 + 2y^2 \]  

8.4.1

B is the optimum quadratic RAS obtained by the method of Weissanberger and corresponding to the Lyapunov function

\[ V = .77972x^2 + 1.5532xy + 1.3206y^2 \]  

8.4.2

The time derivative \( \dot{V} \) by virtue of the system equations is

\[ \dot{V} = -1.59944x^2 - 3.1064xy - 2.6412y^2 \]

\[ + 3.11888x^2y + 3.1064x^2y^2 \]  

8.4.3

Taking the quadratic part of \( \dot{V} \) and using it as \(-\Phi\) and solving the regular Zubov equation for 4th and 6th degree Lyapunov functions respectively there result the RAS of contours C and D. Far from possessing enhanced properties these regions are considerably inferior to the quadratic region B.

Contour E is the RAS for the 6th degree Lyapunov function obtained by the method of chapter 7. This region is a considerable improvement and although it is not much larger than contour B, it does indicate more clearly the shape of the actual DOA.
It is worth noting that the union of the R\&S of fig. 8.2 is itself a R\&S.

8.4.2. The System 1.7.1.4.

Fig. 8.3.(a) shows the R\&S for Liapounov functions of various degrees using the optimum \( \phi \)-function

\[ \phi = 0.0116c^2 + 0.1275c + 0.4437c^2 \]

which was obtained by the method of Weissenberg.

In comparison fig. 8.3.(b) shows the same degree Liapounov functions using the arbitrarily selected function

\[ \phi = c^2 + c^2 \]

The enlargement of the quadratic region \( A \) is obvious, but the higher degree regions of fig. 8.3.(a) are inferior to those of fig. 8.3.(b).

8.4.3. The System 1.7.1.4.

Fig. 8.4 shows the R\&S for this system. \( A \) is the R\&S for the initial guess

\[ V = x^2 - 0.5xy + 0.75y^2 \]

B is the optimum quadratic R\&S obtained by Weissenberg's method. The Liapounov function for this is

\[ V = 1.5066x^2 - 1.1111xy + .98734y^2 \]
This corresponds to the function
\[
\phi = 1.1111x^2 - 0.072545xy + 0.86363y^2
\]

Using this function in the regular Zubov equation and generating the 6th degree Liapounov function there results the RAS of contour C.

That this RAS is not the best that can be obtained is shown by the RAS of contour D which is superior to it. D was obtained by the method of chapter 7.

8.5. Comments

The method of Weissenberger is not particularly effective in application to continuous systems owing to the invalidity of Weissenberger's conjecture. The reason for this is the inherent non-uniformity of convergence of the method of Zubov.

The method was originally proposed as a method for the analysis of relay control systems and as such it is a powerful systematic procedure. We have not given any examples of the application to relay systems as the original paper of Weissenberger contains a considerable number.
9. THE TRACKING FUNCTION METHOD

9.1. Practical Stability

The tracking function method\(^9\) is used to investigate the practical stability of second order systems. Originally formulated as a graphical procedure applicable to autonomous systems only, the method admits of modification to facilitate machine computation\(^{17,18}\), and can in certain cases be extended to the analysis of forced systems and systems with time dependent parameters.\(^{20}\)

In many practical situations the concepts of stability in the sense of Liapunov are either too restrictive, classing as unstable, systems whose performance is quite satisfactory, or not restrictive enough, classing as stable, systems whose response transgresses those physical constraints to which the system is subject.

In such situations, the concept of 'practical stability' is more useful.

Definition 9.1.1. (Practical Stability)

Let \( \Omega_1 \) be a closed bounded region of the state space such that all expected initial conditions (or impulse disturbances) lie in \( \Omega_1 \).

Let \( \Omega_2 \) be the region of the state space comprising
all states which are acceptable.

Then, a system is *practically stable* provided the response to initial conditions in $\mathcal{N}_1$ cannot leave $\mathcal{N}_2$.

A more rigorous definition of practical stability may be found in ref. 3. For the purposes of the following discussion the above is sufficient.

That practical stability and stability in the sense of Lyapunov are distinct may be appreciated from the following examples.

Consider a chemical plant operating in a stable fashion except for small oscillations of one of the state variables (temperature, say) about the required operating point. According to the Lyapunov concept such a system would be unstable. However, if the oscillations were of sufficiently small amplitude it may be that the operation of the plant is entirely satisfactory. The plant would then be practically stable. (The system 1.7.4 is just such a system).

Alternatively, consider a position control servomechanism stable in the sense of Lyapunov, but underdamped. If the underdamping were such as to cause unacceptable overshoot the system would not be practically stable.

To analyse practical stability, therefore, it is necessary to specify 'a-priori' the regions $\mathcal{N}_1$ and $\mathcal{N}_2$ then to seek bounds on the response to initial conditions in $\mathcal{N}_1$ and to determine whether or not this response lies
for all time in $\Omega_{2}$.

Such bounds on the system response may be obtained by the consideration of several simple functions, the so-called tracking functions. In many respects the tracking function method replaces the search for a single, possibly complicated, Lyapunov function with the consideration of a large number of simple functions.

9.2. Tracking Functions

Consider the state plane of fig. 9.1 for the system

\[ \dot{x} = f(x,y), \quad f(0,0) = 0 \]
\[ \dot{y} = g(x,y), \quad g(0,0) = 0 \]

9.2.1

It is required to bound the system trajectory from the point P. The values of two arbitrary tracking functions $E_{1}(x,y)$ and $E_{2}(x,y)$ near P are shown. The loci $\dot{E}_{1}(x,y) = 0$ and $\dot{E}_{2}(x,y) = 0$ are drawn. These derivatives are obtained from

\[ \dot{E}_{i} = \frac{\partial E_{i}}{\partial x} f + \frac{\partial E_{i}}{\partial y} g \]

9.2.2

In fig. 9.1 we have assumed that at P, $\dot{E}_{1}$ is negative and $\dot{E}_{2}$ positive. Consideration of these signs together with the directions of increase of the $E_{i} = \text{constant}$ curves through P establishes regions (shown shaded) forbidden to
the trajectory from P, so that this trajectory is limited to an allowable sector (in this case the sector RPQ). An outer bound on the system response from P is therefore that arm of the allowable sector which takes the response farthest from the origin (in this case PQ).

This extreme bounding curve is extended until one of the $E_1 = 0$ loci is crossed and continued in this new region in accordance with the new configuration existing there.

If this bounding path at any time would, if produced, enter the previously defined $\Omega_1$ region, it is constrained from doing so and made to follow the boundary of $\Omega_1$ until such time as a more extreme path is indicated.

In the above, we have described the use of only two tracking functions. There is of course no limit (except increasing complexity) to the number of tracking functions used, and the larger the number the closer will be the bound obtained.

As the extreme bounding path is extended one of three situations will prevail;

(1). It may converge to the boundary of the $\Omega_1$ region. In this case the $\Omega_1$ and $\Omega_2$ regions are the same and the system is practically stable.

(2). It may form a closed curve C about the origin. Let $iC$ be the interior of C. Then the system is practically
stable if $10 < Q_2$

(3). None of the above may happen and either the extreme path diverges to infinity or, in (2) $10 \notin Q_2$. In this event, since the method gives sufficient but not necessary conditions for practical stability, nothing may be inferred, and the analysis should be pursued further with the use of a larger number of tracking functions, to obtain if possible, closer bounds.

9.3 Disadvantages of the Graphical Method

This otherwise elegant and powerful procedure suffers from three main drawbacks:

(1). The number of tracking functions required to elicit sufficient information may rise to impractical size. Thus, each $E_i = 0$ locus divides the state plane into at least two distinct regions. The use of a large number of tracking functions therefore results in an impossibly large number of regions.

(2). The process of plotting the $E_i = 0$ loci requires, in all but the simplest cases, the use of a digital computer, and is both time consuming and inaccurate.

(3). The preselected set of tracking functions may
not be the most effective for a particular problem or at a particular point in the state space.

The complexity of the graphical procedure may be appreciated from consideration of fig. 9.2 which shows the method applied to the chemical reactor (system 1.7.4). Here we have used the eight tracking functions

\[
\begin{align*}
E_1 &= x^2 + y^2 \\
E_2 &= .6x + y \\
E_3 &= x + y \\
E_4 &= 4x + y \\
E_5 &= x \\
E_6 &= x - y \\
E_7 &= y \\
E_8 &= x + 2y
\end{align*}
\]

The computing algorithm described below effectively overcomes these objections and is both simple and fast.

9.4. The Computing Algorithm

Consider the state plane of fig. 9.3 for the system 9.2.1 and the two linear tracking functions

\[
\begin{align*}
\dot{E}_1 &= x \quad \text{9.4.1} \\
\dot{E}_2 &= y
\end{align*}
\]

then

\[
\begin{align*}
\dot{E}_1 &= x = f(x,y) \quad \text{9.4.2} \\
\dot{E}_2 &= y = g(x,y)
\end{align*}
\]
At a point \( P \) the signs of \( E_1 \) and \( E_2 \) are computed from 9.4.2. At a point \( Q \) distant \( s \) from \( P \) on one side of the \( E_1 \) constant line through \( P \), the value of \( E_1 \) is computed. We denote this by \( E_1(Q_1) \).

Then, if the sign of \( E_1(Q_1) - E_1(P) \) is the same as the sign of \( E_1(P) \) the point \( Q_1 \) is accepted as lying on the allowable side of the line \( E_1 = \) constant through \( P \). If the signs are different the point \( Q_1 \) is rejected as lying on the forbidden side, and the point \( Q_1' \) is accepted, where \( Q_1' \) is distant \( s \) from \( P \) but on the other side of the \( E_1 \) constant line.

Similarly by comparing the signs of \( E_2(P) \) and \( E_2(Q_2) - E_2(P) \), either the point \( Q_2 \) or \( Q_2' \) is accepted as lying on the allowable side of the line \( E_2 = \) constant through \( P \).

In fig. 9.3, \( E_1 \) is assumed to be negative, \( E_2 \) is assumed to be positive and the directions of increase of the \( E \) values shown. The forbidden sides are shown shaded.

The allowable sector for the trajectory from \( P \) is, in this case, the \( 90^\circ \) sector \( Q_1Q_2Q' \).

As the next tracking function, the line through \( P \) bisecting the angle \( Q_1Q_2 \) is used. If \( P \) is the point \((x_P,y_P)\), \( Q_1 \) the point \((x_1,y_1)\) and \( Q_2 \) the point \((x_2,y_2)\) the equation of this new tracking function \( E_3 \) is

\[
E_3 = y - mx = \text{constant}.
\]

9.4.3
where
\[ m = \frac{y_1 + y_2 - 2y_p}{x_1 + x_2 - 2x_p} \]  \[ 9.4.4 \]

Then
\[ E'_3 = e - mf \]  \[ 9.4.5 \]

Applying the tracking function policy to this new tracking function, one of its sides may be established as forbidden. Either point \( Q_1 \) or point \( Q_2 \) will lie on this side and may be rejected and replaced by a new point \( T \) lying on the line \( E_3 = \text{constant} \) through \( P \), distant \( s \) from \( P \) and lying in the allowable sector. The coordinates of this new point (\( T \) in fig. 9.3) are given by
\[ x = \frac{t}{s}(1 + m^2)^{-1} \]  \[ 9.4.6 \]
\[ y = mx \]  \[ 9.4.7 \]

where we have assumed that \( P \) is the origin. (We in fact are using a moving coordinate system centred on \( P \) as explained later).

In 9.4.6 the sign to be used is determined by examining the signs of \( E_3(P) \) and \( E_3(T) - E_3(P) \) as above. If \( Q_1 \) lies on the forbidden side of \( E_3 \) the function \( E_3 \) becomes the new \( E_2 \) and \( T \) becomes the new \( Q_1 \). If \( Q_2 \) lies on the forbidden side the function \( E_3 \) becomes the new \( E_1 \) and \( T \) becomes the new \( Q_2 \).

In any event there remain two tracking functions \( E_1 \) and \( E_2 \) with points \( Q_1 \) and \( Q_2 \) and the allowable sector has
been narrowed to the 45° sector $Q_1PQ_2$ (or $Q_1PT$ in the case of fig. 9.3).

This bisection of the allowable sector and rejection of forbidden sides is continued until the angle of the allowable sector is deemed small enough. At this stage the point $P$ is moved from its present position to either $Q_1$ or $Q_2$ whichever takes the locus to the greater radius. In fig. 9.3 the locus would be extended to $T$.

The entire procedure is applied at this new point $P$ and so on until the system response has been bounded by the curve composed of short straight-line sections.

The previously defined $\Omega_1$ region contains all the initial states expected. Obviously, therefore, the $\Omega_2$ region contains $\Omega_1$ since all initial states are achievable if only at the initial instant. Thus, if the bounding locus at any stage enters the $\Omega_1$ region it is constrained to follow its boundary until a more extreme path is indicated.

The flow diagram for the method is shown in fig. 9.4. The actual program is not given since it is a special case of the program for the non-autonomous case which is given later.

9.5. Salient Points Concerning the Algorithm

(1). If the number of bisections employed is $b$, the number of tracking functions implicitly employed is $2^{b+1}$.
and the angle of the allowable sector reduces to \((90/2^b)\)°. However, at each point, the forbidden sides of only \((2 + b)\) tracking functions (x, y and the bisectors) are established, the forbidden sides of the others being superfluous. Thus the efficiency of the method, which is directly proportional to the number of tracking functions used, may be increased without proportionate increase in computing time.

(2). The method errs each time one of the \(E = 0\) loci is crossed. However by keeping the step size \(s\) as small as necessary this error may be made negligible. In any case, in the majority of situations, this error will be conservative. This is due to the fact that in the majority of cases the \(\mathcal{N}_2\) region will, because of the conservative estimate given by the tracking function method, be convex. In such cases the \(\mathcal{N}_2\) region indicated by the method will be greater than the \(\mathcal{N}_2\) which would be obtained graphically.

(3). Since it is advisable to keep the step size \(s\) as small as possible, the difference between the values of, say, \(E_1(\Omega_1)\) and \(E_1(P)\) may be, when compared to \(E_1(P)\), outside the computer significance. However, since we are interested in the difference between values rather than in absolute values, and since the tracking functions are linear, it is possible to transfer the point \(P\) to the origin at each stage. This is equivalent to using a coordinate frame.
centred on P and moving with it. This is accomplished by ignoring the constant in equation 9.4.3, as we have already done in equations 9.4.6 and 9.4.7.

(4). Although the tracking functions used are linear a large enough number can be employed to cause the $\Omega_2$ boundary to become almost continuous.

9.6. Examples

9.6.1. The System 1.7.4.

Fig. 9.5 shows the method applied to this system for 1 and 4 bisections. Also shown is the actual region of practical stability (the interior of the limit cycle). The improvement in using the computer method can be seen by comparing figs. 9.5 and 9.2. The region B of fig. 9.5 is a much closer bound than the region of fig. 9.2, although the latter required about 1 day to obtain (graphically) while the former required only 15 mins. (1.C.T. 1905).

Since no constraints have been placed upon the system response the system is practically stable. The response to initial conditions lying inside contours A or B can never leave these regions.

Fig. 9.6 shows the analysis of this system with a rectangular $\Omega_1$ region and using 1 bisection. As expected the region of practical stability is larger than those of fig. 9.5.
9.6.2. A Relay Control System

Consider the system shown in fig. 9.7. This system has been the subject of study by Weissenberger. The equations are

\[ \begin{align*}
\dot{x} &= y \\
\dot{y} &= x - \text{sgn}(2x + y)
\end{align*} \]

where

\[ \text{sgn}(z) \triangleq \frac{z}{|z|} \]

and is undefined for \( z = 0 \).

Fig. 9.8. shows the bound obtained using l bisection. Also shown are the \( E_i = 0 \) loci which would be required to apply the method graphically.

From the figure it may be seen that any trajectory originating inside the contour FGHI must eventually attain the region bounded by \( ABCD \). The system is therefore ultimately bounded for responses starting inside FGHI. Also shown for comparison is the RIS obtained by Weissenberger using the Liapunov function

\[ V = \frac{1}{2}x^2 + \frac{1}{2}y^2 + (2x + y)\text{sgn}(2x + y) \]

It is interesting to note that the tracking function method gives a larger region of stability (FGHI) than does the method of Liapunov, and gives a closer bound on the response to initial conditions \((0,0)\). It cannot of course
predict asymptotic stability.

This example illustrates that for stable systems the graphical procedure may be more powerful since the region of ultimate boundedness is contained in the region of initial conditions. For unstable systems, however, the computer method is more powerful since it is required to find as close a bound as possible to the system response and this requires the use of as many tracking functions as possible.

9.6.3. The System 1.7.1. \((\varepsilon = 0.1)\)

Contours A, B and C of fig. 9.11 show the \(\Omega_2\) regions obtained for this system using various numbers of bisections.

Curve F is the actual region of practical stability. Since no constraints have been placed on the system response practical stability is established.

9.6.4. A Negative Resistance Oscillator

Consider the system of fig. 9.9. Cunningham\textsuperscript{36} gives the following equations

\[
\begin{align*}
\dot{e}_r &= \frac{1}{C}(i_r - f(e_r)) \quad 9.6.2 \\
i_r &= \frac{1}{L}(-Ri_r - e_r + E)
\end{align*}
\]

Approximating the non-linear function of the negative-
resistance element by the three resistances \( r_1, r_2 \) and \( r_3 \) as shown in fig. 9.10 and using the values

\[
\begin{align*}
E &= 55V \\
r_1 &= 60\, \Omega \\
r_2 &= 50\, \Omega \\
r_3 &= 87.5\, \Omega \\
L &= 2,10^{-3} \, H \\
L &= 28.57143 \, \Omega \\
R &= 28.57143 \, \Omega \\
E &= 55V \\
C &= 10^{-6} \, F \\
\end{align*}
\]

and writing \( x = e_r \) and \( y = i_r \) we have

\[
\begin{align*}
x &= 10^6(y - f(x)) \\
y &= 5,10^2(-x - 28.57143y + 55)
\end{align*}
\]

where

\[
\begin{align*}
f(x) &= .02x & 0 < x < 30 \\
&= -.016x + 1.1 & 30 < x < 60 \\
&= .0114286x - .585716 & x > 60
\end{align*}
\]

The focus of the system is the point where the load-line crosses the non-linear characteristic. For the purposes of the tracking function method the focus must be transferred to the origin. This is accomplished by the transformation

\[
\begin{align*}
X &= x - 45 \\
Y &= y - .35
\end{align*}
\]
and the transformed system is

\[
\begin{align*}
\dot{x} &= 10^6(y - f(x)) \\
\dot{y} &= 5.10^2(-x - 28.57143y)
\end{align*}
\]

where

\[
\begin{align*}
f(x) &= \begin{cases} 
0.02x + 0.55 & x < -15 \\
-0.016x & -15 < x < 15 \\
0.011428x - 0.421429 & x > 15
\end{cases}
\]

Fig. 9.10 shows the regions bounding the response obtained for various numbers of bisections. The actual region of practical stability is the interior of the limit cycle which exists just inside the region obtained for 5 bisections. Again, each region is both the \( \mathcal{N}_1 \) and \( \mathcal{N}_2 \) region.

Notice that if the voltage were constrained to lie below 70V the analysis using 5 bisections would establish practical stability while that using 2 bisections would not. (see line d). This illustrates the sufficiency but not necessity of the conditions.

9.7. Practical Stability in Polar Coordinates

It has been found that in certain cases a simple extension of the method increases its effectiveness. Let the system equations be as given in 9.2.1. Let us transform the system to polar coordinates by the transformation
\[ x = r \cos \Theta \]
\[ y = r \sin \Theta \]

9.7.1

The system equations become

\[ \dot{r} = f(r \cos \Theta, r \sin \Theta) \cos \Theta + g(r \cos \Theta, r \sin \Theta) \sin \Theta \]
\[ \dot{\Theta} = \left\{ g(r \cos \Theta, r \sin \Theta) \cos \Theta - f(r \cos \Theta, r \sin \Theta) \sin \Theta \right\} / r \]

9.7.2

or, in general,

\[ \dot{r} = F(r, \Theta) \]
\[ \dot{\Theta} = G(r, \Theta) \]

9.7.3

Equations 9.7.3 are of the same form as equations 9.2.1 and the tracking function method may be applied in the \( r \Theta \) plane (plotted in rectangular Cartesian coordinates) either in the graphical procedure or in the numerical formulation.

The only modification necessary is that having established the allowable sector, the locus is extended not along the arm which takes it farthest from the origin but along that which gives the highest \( r \) value.

When, as this locus is extended, it reaches either \( \Theta = -2\pi \) or \( \Theta = 2\pi \) the point is moved back to \( \Theta = 0 \), and the process continued.

One desirable feature of this modification is that if the \( \mathcal{N}_1 \) region is circular in the \( xy \) plane, it plots into
a straight line parallel to the $\theta$-axis. If at any stage
the extreme path would cross this line it is constrained to
move along it till a more extreme path is indicated.

The extreme path in the $r\theta$ plane may then be re-plotted
in the $xy$ plane to present the more familiar region of
practical stability. If in the $r\theta$ plane linear tracking
functions are used, the extreme path in the $xy$ plane compr-
ises arcs of circles and spirals.

The program for the method in the $r\theta$ plane is given
in appendix A.10.

9.8. Examples


Fig. 9.11 shows the analysis of this system in the
$r\theta$ plane (curves D and E). Also shown are the curves obta-
ined using the tracking function method in the $xy$ plane.

The increased power of the method in the $r\theta$ plane is
apparent from the fact that the region of practical stability
obtained using the $r\theta$ plane method with only 2 bisections
(curve E) is a closer bound than that obtained using the $xy$
plane method and 5 bisections (curve C).

It would appear that the $r\theta$ plane method will be more
effective for those systems whose region of practical
stability is approximately circular.
9.8.2 The System 1.7.4

Fig. 9.12 shows the method applied to this system in the $r\theta$ plane using initial conditions of magnitude $\leq .5$. The number of bisections is 3. Fig. 9.13 shows the same result re-plotted in the $xy$ plane (or in this case the $c$ plane). Also shown (curve Z) is the region of practical stability for zero initial conditions.

Fig. 9.14 shows the $r\theta$ plane method applied to this system for zero initial conditions and various numbers of bisections. Fig. 9.15 shows these regions re-plotted in the $xy$ plane.

Comparison of figs. 9.15 and 9.5 shows that in this case the $xy$ plane method is more effective.

9.9. Extension to Linear Non-Stationary Systems

Consider the general second order non-stationary system

$$\ddot{x} + p(t)x + q(t)x = 0$$  \hspace{1cm} 9.9.2

where the time-dependent coefficients $p(t)$ and $q(t)$ satisfy the bounds

$$0 \leq m_1 \leq p(t) \leq m_2 \leq \infty$$  \hspace{1cm} 9.9.3

$$0 \leq m_1 \leq q(t) \leq m_2 \leq \infty$$
Starzinski \(^{37,2}\) obtained, by the use of a special Liapounov function, the inequalities

\[
\begin{align*}
    n_1 &> \sqrt{m_2} - \sqrt{m_1} \\
    n_2 &< \frac{m_2 + 2m_1m_2 + 5m_1}{\sqrt{m_2} - \sqrt{m_1}}  
\end{align*}
\]

as sufficient conditions for asymptotic stability.

The tracking function method is applicable to such systems, and it will be shown that the upper bound \(n_2\) of \(p(t)\) may be relaxed to infinity while the lower bound may be lowered slightly.

Consider the particular system

\[
x + p(t)x + 4(1 + 0.5\cos \omega t)x = 0
\]

then from 9.9.1 we have

\[
q(t) = 4(1 + 0.5\cos \omega t)
\]

so that by 9.9.3 we have

\[
n_1 = 2 \quad m_2 = 6
\]

Applying the inequalities 9.9.4 the bounds on \(p(t)\) sufficient for asymptotic stability are
1.035 \leq p(t) \leq 22.15 \quad 9.9.7

We now apply the tracking function method to this system. Let us write the system equations as

\begin{align*}
x &= y \\
y &= -q(t)x - p(t)y
\end{align*} \quad 9.9.8

and let us consider the general linear tracking function

\[ E(x, y) = Ax + y \quad 9.9.9 \]

then

\[ E(x, y) = \dot{Ax} + \dot{y} \quad 9.9.10 \]

\[ = (A - p(t))\dot{y} - q(t)x \quad 9.9.11 \]

For this to be positive we require

\[ x < \left\{ \frac{A - p(t)}{q(t)} \right\} y \quad 9.9.12 \]

Of course the right hand side of 9.9.12 is time varying. However, let us consider \( y \) to be positive (without sacrificing generality since the configuration of the \( E = 0 \) loci in the phase plane is, as we shall see, radially symmetric).

Then it is not difficult to see that, because of the bounds on \( p(t) \) and \( q(t) \), we may state

For positive \( y \), \( E \) is positive provided
\[ x < \left\{ \frac{\lambda - n_2}{m_1} \right\} y \quad \text{for} \quad (\lambda - n_2) < 0 \quad 9.9.13 \]
\[ x < \left\{ \frac{\lambda - n_2}{m_2} \right\} y \quad \text{for} \quad (\lambda - n_2) > 0 \quad 9.9.14 \]

Similarly we may state

For positive \( y \), \( E \) is negative provided

\[ x > \left\{ \frac{\lambda - n_1}{m_2} \right\} y \quad \text{for} \quad (\lambda - n_1) < 0 \quad 9.9.15 \]
\[ x > \left\{ \frac{\lambda - n_1}{m_1} \right\} y \quad \text{for} \quad (\lambda - n_1) > 0 \quad 9.9.16 \]

For the system of 9.9.5 \( m_1 = 2 \) and \( m_2 = 6 \). Let us assume an infinite upper bound on \( p(t) \) i.e.

\[ n_2 = \infty \quad 9.9.17 \]

Then the inequalities 9.9.13 and 9.9.14 are never satisfied (assuming finite \( \lambda \)). In words, the time derivatives of the tracking functions are never positive for positive \( y \).

Substituting the values of \( m_1 \) and \( m_2 \) in 9.9.15 and 9.9.16 we have

For positive \( y \), \( E \) is negative for

\[ x > \left\{ \frac{\lambda - n_1}{2} \right\} y \quad \text{for} \quad (\lambda - n_1) \text{ positive} \quad 9.9.18 \]
\[
x > \left\{ \frac{\lambda - n_1}{6} \right\} y \quad \text{for} \quad (\lambda - n_1) \text{ negative}
\]

9.9.19

It is required to find the lowest value of \( n_1 \) for which asymptotic stability is assured. To this end we take various values of \( n_1 \) and obtain, via the tracking function method, bounds on the system response. The different tracking functions are obtained by taking different values of \( \lambda \).

For example let us take \( n_1 = 1 \) and \( \lambda = 2 \) (which corresponds to the tracking function

\[
E = 2x + y
\]

9.9.20

Then conditions 9.9.18 and 9.9.19 become simply \( \dot{E} \) is negative for \( x > y/2 \) and is undefined elsewhere, for \( y \) positive.

For the values

\[
\lambda = 0, -\frac{1}{2}, -1, -2, \frac{1}{2}, 1, \text{ and } 2
\]

and \( n_1 = \frac{1}{2}, 1, 1\frac{1}{2}, 2, \text{ and } 3 \)

we obtain the following table of conditions for the \( \dot{E} \) to be negative for positive \( y \).
Figs. 9.16.(a) to 9.16.(d) show the configurations in the phase plane for the various values of \( n_1 \) (\( n_1 = \frac{1}{2} \) not shown). In fig 9.16.(a) the symbols \( ? \) on one side of each of the \( E = 0 \) loci indicate the regions of sign indeterminacy of the \( E \). The symbols \( \rightarrow \) show the allowable sectors. It should be noted that the tracking function \( E = x \) is employed in addition to those listed above. The sign of the derivative of this function is of course defined everywhere and is positive for \( y \) positive.

If \( r_0 \) is the initial radius of the extreme path and \( r_1 \) the radius after one encirclement of the origin, a
measure of the stability of the system is the ratio $r_1/r_0$. For asymptotic stability we must have

$$r_1/r_0 < 1 \quad 9.9.21$$

The values of $r_1/r_0$ are plotted in fig. 9.17 against $n_1$. This curve passes through the line $r_1/r_0$ at the value $n_1 = 1.65$. This establishes the following conditions as sufficient for asymptotic stability

$$1.65 \leq p(t) \leq \infty \quad 9.9.22$$

which compares with the Starzinski criterion

$$1.035 \leq p(t) \leq 22.15 \quad 9.9.23$$

Thus, although the lower bound obtained by the tracking function method is more restrictive than the bound obtained by Starzinski, the upper bound is relaxed to infinity.

In this analysis only 8 tracking functions were employed. The use of a larger number of tracking functions, though possible, is impractical owing to the resultant complexity of the graphical procedure. It is possible, however, to evolve a numerical procedure to deal with non-stationary systems and which allows the consideration of a large number of tracking functions. This is described below, and it is shown that the lower bound may be decreased to 1.03 at least.
9.10. Computing Algorithm for Non-Stationary Systems

Consider the system

\[ x = f(x,y,t) \]
\[ y = g(x,y,t) \]

Then the extreme bounding path from any point in the state plane may still be determined by the procedure described in section 9.4, provided \( t \) is considered constant. Since \( t \) is not constant, however, additional logic must be introduced and a condition must be fulfilled. This condition is that for any \( x \) and \( y \) the functions \( f \) and \( g \) are bounded functions of \( t \). If this condition is satisfied the tracking function method is applicable.

The only difference between the routine described here and that described in section 9.4 is that at any point \( P \) in the state plane (see fig. 9.3) it is necessary to find the allowable sector for four distinct contingencies namely

1. \( f \) assumes its maximum, \( g \) its minimum for all \( t \)
2. \( f \) its maximum \( g \) its maximum
3. \( f \) assumes its minimum, \( g \) its minimum
4. \( f \) its minimum \( g \) its maximum

Thus four allowable sectors are computed. The most extreme path in each of these sectors is established so that
at most four extreme paths from P are indicated (it may be that more than one of the above four contingencies indicate the same extreme path). Of these four extreme paths, one will be more extreme than the others. This path is then taken.

The program for the method is shown in appendix A.10.A under the name TFNONJUT.

Using this program with the value \( n_1 = 1.03 \) there resulted the extreme path of fig. 9.18 which indicates asymptotic stability. Our final conditions for asymptotic stability are

\[
1.03 \leq n_1 \leq \infty
\]

in which both bounds are better than those of Starzinski. In fig. 9.18 the number of bisections was 4, i.e. the number of tracking functions was 32. It is possible to obtain even better bounds using a greater number of tracking functions although the computing time may then be prohibitive.
9.11. Extension to Systems with Bounded Forcing Functions

The tracking function method may be extended to the analysis of systems with bounded forcing functions. In a recent paper Aggarwal described a method for obtaining bounds on the response of a restricted class of forced systems. The systems to which the method of Aggarwal is applicable are those whose behaviour is governed by equations of the form

\[ \begin{align*}
\dot{x} &= -f(x) \pm g(y) + p(t) \\
\dot{y} &= \pm h(x) - k(y) + q(t)
\end{align*} \]

where \( f(x), g(y), h(x) \) and \( k(y) \) are polynomials of odd degree with positive leading coefficients, and \( p(t) \) and \( q(t) \) are bounded functions.

For this class of systems the tracking function method gives closer bounds and in addition is applicable to a wider class of systems.

As for the case of linear non-stationary systems (see section 9.9) there arise regions of indeterminacy of the signs of the time derivatives of the tracking functions. For many systems, provided a sufficient number of tracking functions is employed, practical stability may still be inferred by utilising those functions whose signs are determinate at any particular point in the state plane.
Consider the system
\[
\begin{align*}
\dot{x} &= -x + 2x^2y, \\
\dot{y} &= -y + f(t)
\end{align*}
\]  
where \( f(t) \) is an arbitrary function bounded by
\[-\infty < -L \leq f(t) \leq L < \infty \]

Let us consider the two tracking functions
\[
\begin{align*}
E_1 &= x, \\
E_2 &= y
\end{align*}
\]
then, by virtue of 9.11.2 we have
\[
\begin{align*}
\dot{E}_1 &= -x + 2x^2y, \\
\dot{E}_2 &= -y + f(t)
\end{align*}
\]  
For \( \dot{E}_1 \) to be positive we require
\[y > \frac{1}{2x}\]
and for \( \dot{E}_1 \) to be negative we require
\[y < \frac{1}{2x}\]
Thus, the sign of \( \dot{E}_1 \) is defined everywhere.
For \( \dot{E}_2 \) to be positive we require
\[y < f(t)\]
and by the boundedness of \( f(t) \) this implies

\[ y < -L \]  \hspace{1cm} 9.11.9

Similarly for \( E_2 \) to be negative we require

\[ y > L \]  \hspace{1cm} 9.11.10

and \( E_2 \) is undetermined for

\[ -L < y < L \]  \hspace{1cm} 9.11.11

Referring to the state plane of fig. 9.19 for this system, it is evident that the rectangle ABCD is a region of practical stability for initial conditions lying inside it. This is obvious because at all points on the boundary all trajectories are directed inwards.

Further, if EFGH is any other rectangle containing the origin and such that E and F lie between A and B, and G and H lie between C and D, then any trajectories originating inside EFGH can never leave it.

In the limit as E and F approach the y-axis the region degenerates into the y-axis between \( y = L \) and \( y = -L \). In other words, for zero initial x the oscillation is contained in the y-axis. This is obvious from the equations 9.11.2 where if x is zero there can be no x variation subsequently.

For this, admittedly rather simple, system the tracking
function method, using only two tracking functions, has elicited the following information

(a). If the initial conditions $x_0, y_0$ are such that

$$|x_0| < \frac{1}{2}L \quad |y_0| < \frac{L}{41}$$

then the trajectories are bounded by $\triangle BCD$ for any forcing function $f(t)$ satisfying $9.11.3$.

(b). If $x_0 = 0$ and $|y_0| < \frac{L}{4}$ the trajectories are confined to the $y$-axis.

It is worth noting that this system is not a member of the class to which Aggarwal's method is applicable.

As a further, more realistic, example consider the system

$$\begin{align*}
\dot{x} &= -x^3 + y + \sin \omega t \\
\dot{y} &= -x - y + 2 \sin \omega t
\end{align*}$$

This system has been analysed by Aggarwal. It is shown below that the tracking function method, using only four linear tracking functions, provides closer bounds.

Consider the four tracking functions

$$\begin{align*}
E_1 &= x \\
E_2 &= y \\
E_3 &= x + y \\
E_4 &= x - y
\end{align*}$$
Then, by virtue of 9.11.13, we have

\[ E_1 = \dot{x} = -x^3 + y + \sin \omega t \quad 9.11.15 \]

The right hand side of 9.11.15 is time varying, but by the boundedness of the \( \sin \) function it is easily seen that

\[ E_1 \text{ is positive provided} \]

\[ y > x^3 + 1 \quad 9.11.16 \]

and negative if

\[ y < x^3 - 1 \quad 9.11.17 \]

and is indeterminate for

\[ x^3 - 1 < y < x^3 + 1 \quad 9.11.18 \]

In fig. 9.20 the region \( \mathcal{A}_1 \) is the region of sign indeterminacy of \( \dot{E}_1 \). The curves \( y = x^3 - 1 \) and \( y = x^3 + 1 \) are drawn and the sign of \( \dot{E}_1 \) is given on the appropriate sides of these lines.

Similarly, we have

\[ E_2 = \dot{x} - y + p \quad -2 < p < 2 \] 9.11.19
\[ E_3 = -x^3 - x + q \quad -3 < q < 3 \]
\[ E_4 = -x^3 + x + 2y + r \quad -3 < r < 3 \]

and we deduce
\[ \dot{E}_2 \]

positive  \( y < -x - 2 \)

negative  \( y > -x + 2 \)

\[ \dot{E}_3 \]

positive  \( x < \frac{-3}{\sqrt{3}} \)

negative  \( x > \frac{3}{\sqrt{3}} \)

\[ \dot{E}_4 \]

positive  \( y > \frac{(x^3 - x + 3)}{2} \)

negative  \( y < \frac{(x^3 - x - 3)}{2} \)

Fig. 9.20 shows the configuration in the state plane. The regions \( \Delta_1 \) are the regions of sign indeterminacy in \( E_1 \). Constructing the extreme locus by the tracking function policy results in the closed region PORSTUV. This is the \( \Omega_2 \) region for zero initial conditions. Any trajectory starting inside this region can never leave it, irrespective of the frequencies \( \Omega_1 \) and \( \Omega_2 \) of the forcing terms.

The bound obtained by Aggarwal is the rectangle JKLM. The tracking function method thus gives much closer bounds. For example, if practical stability demanded that \( y \) be always less than 3 in magnitude, the tracking function method would establish practical stability, while Aggarwal's method would not.
10. **THE METHOD OF IUVUS AND LAPIDUS**

10.1 **Introduction**

Consider a system describable by the equations

\[
\begin{align*}
\dot{x} &= f(x,y) + u_1(t) \\
\dot{y} &= g(x,y) + u_2(t)
\end{align*}
\]

In the general case it will not be possible to obtain the solution of these equations analytically. However, if only the stability of the solution is of interest, it may be possible to assess this via some simpler technique. The methods described in the previous chapters have, to a greater or less extent achieved this. In this chapter we describe the method of Iuus and Lapidus\(^7\) which provides another approach, simple to implement, but whose effectiveness is, as we shall see, open to some doubt.

The method is based upon an averaging technique which allows the setting up of an approximation to the system equations in a form which is readily interpreted from a stability viewpoint.

10.2. **The Method**

The system equations 10.1.1 are transformed to polar coord-
inates by the transformation
\[
\begin{align*}
\dot{x} &= r \cos \theta \\
\dot{y} &= r \sin \theta
\end{align*}
\]
10.2.1
giving
\[
\begin{align*}
\dot{r} &= \left\{ f(r \cos \theta, r \sin \theta) + u_1(t) \right\} \cos \theta \\
+ \left\{ g(r \cos \theta, r \sin \theta) + u_2(t) \right\} \sin \theta \quad 10.2.2 \\
\dot{\theta} &= \frac{1}{r} \left[ g(r \cos \theta, r \sin \theta) + u_2(t) \right] \cos \theta \\
- \left[ f(r \cos \theta, r \sin \theta) + u_1(t) \right] \sin \theta \quad 10.2.3
\end{align*}
\]

We now assume that, on the right hand side of 10.2.2 and 10.2.3, the radius \( r \) is constant \((= r_0)\) and that the angle \( \theta \) is a linear function of time \((= \Theta_0 + \lambda t)\).

Since we are interested only in the stability of the system, and since this is implicit in the behaviour of \( r \), we ignore equation 10.2.3. Equation 10.2.2 becomes
\[
\dot{r} = \left\{ f\left[r_0 \cos(\Theta_0 + \lambda t), r_0 \sin(\Theta_0 + \lambda t)\right] + u_1(t) \right\} \cos(\Theta_0 + \lambda t) \\
+ \left\{ g\left[r_0 \cos(\Theta_0 + \lambda t), r_0 \sin(\Theta_0 + \lambda t)\right] + u_2(t) \right\} \sin(\Theta_0 + \lambda t) \quad 10.2.4
\]

If the assumptions of constant radius and linear dependence of the angle on the time were correct, integration
of equation 10.2.4 between \( t = 0 \) and \( t = \frac{2 \pi}{A} \) should have the result 0 since the system would have returned to its original state. However, because of the invalidity of the assumptions, this integration will result in a finite value which is interpreted as a change in radius, thus

\[
r\left(\frac{2\pi}{A}\right) - r(0) = \int_{0}^{2\pi/A} \left\{ f_1 r_0 \cos(\theta + \omega t) r_0 \sin(\theta + \omega t) + v_1(t) \right\} \cos(\theta + \omega t) \, dt
\]

\[
+ \int_{0}^{2\pi/A} \left\{ g_2 r_0 \cos(\theta + \omega t) r_0 \sin(\theta + \omega t) + v_2(t) \right\} \sin(\theta + \omega t) \, dt
\]

If the above integrations are performed (since the integrands are now merely functions of \( t \)) and both sides divided by \( \frac{2\pi}{A} \) there results

\[
\frac{\left\{ r\left(\frac{2\pi}{A}\right) - r(0) \right\}}{\frac{2\pi}{A}} = F(r_0, \theta_0) \]

At this stage the assumption of constant \( r_0 \) is removed and \( r_0 \) is replaced by \( r \). In addition, it is assumed that the forcing functions (or their fundamental harmonics) are of that frequency which causes the greatest instability, namely the natural frequency of the system. The natural
frequency may be computed by the formula

\[ \omega_n = \sqrt{\frac{-\alpha + \beta}{\alpha}} \]

where \( a, b, c \) and \( d \) are the coefficients of the linear parts of the system 10.1.1 namely

\[
\begin{align*}
\dot{x} &= ax + by \\
\dot{y} &= cx + dy
\end{align*}
\]

Under these assumptions the function \( F \) of 10.2.6 becomes a function of \( r \) only and we may write it as \( F(r) \).

Now the left hand side of 10.2.6 may be viewed as a linear approximation to the time derivative of the radius \( r \). Replacing the left hand side by \( \frac{dr}{dt} \) then, we have

\[ \frac{dr}{dt} = F(r) \]

As will be seen later, in applying the above procedure the value of the constant \( \lambda \) in the expression for \( \Theta \) in 10.2.4 is immaterial as it cancels out in obtaining 10.2.6 from 10.2.5 so that it may be taken to be unity with no loss of generality (see following example).

It is the basic contention of the method that 10.2.9 contains sufficient information to assess the stability of the actual system 10.1.1. Since 10.2.9 is much simpler
than 10.1.1 it may be possible to integrate to obtain \( r \) as an explicit function of time. Alternatively, if \( \frac{dr}{dt} \) is plotted against \( r \), the behaviour of \( r \) may be inferred by inspection.

Thus, consider fig. 10.2 which shows a plot of \( \frac{dr}{dt} \) vs. \( r \) for a hypothetical system. It is obvious that the origin is unstable since a small displacement results in positive \( \frac{dr}{dt} \). The radius will increase until point \( A \) is reached. Point \( A \) obviously indicates a stable limit cycle. Point \( B \) represents an unstable limit cycle, while \( C \) represents a limit cycle stable on the inside, unstable on the outside.

10.3 An Example

Consider the system

\[
\begin{align*}
\dot{x} &= -x + 2x^2y \\
\dot{y} &= -y + f(t)
\end{align*}
\]

where \( f(t) \) is a sinusoidal function of amplitude \( L \) and of frequency and phase such as to cause the greatest degree of instability. Applying the transformation 10.2.1 we have

\[
\dot{r} = -r \cos^2\Theta + 2r^3\cos^3\Theta \sin\Theta - r \sin^2\Theta + f(t)\sin\Theta
\]

\[
= -r + 2r^3\cos^3\Theta \sin\Theta + f(t)\sin\Theta
\]
Letting $\theta = \theta_0 + \omega t$ and $r = r_0$ on the right hand side of 10.3.2 we have

$$
\dot{r} = -r_0 + 2r_0^3 \cos^3(\theta_0 + \omega t) \sin(\theta_0 + \omega t) \\
+ f(t) \sin(\theta_0 + \omega t) \tag{10.3.3}
$$

Now we have assumed an angular frequency $\omega$ so that the sinusoidal function $f(t)$ causing the greatest degree of instability will have this frequency, thus

$$
f(t) = L \sin(\theta_0 + \omega t) \tag{10.3.4}
$$

Substituting in 10.3.3 and integrating we have

$$
r\left(\frac{2\pi}{\omega}\right) - r(0) = \\
\int_0^{2\pi/\omega} \left[ -r_0 + 2r_0^3 \cos^3(\theta_0 + \omega t) \sin(\theta_0 + \omega t) \\
+ L \sin^2(\theta_0 + \omega t) \right] dt \tag{10.3.5}
$$

Now let $u = \theta_0 + \omega t$ then

$$
du = \omega dt \tag{10.3.6}
$$

so that the right hand side of 10.3.5 becomes

$$
\int_{\theta_0}^{\theta_0 + 2\pi} \left[ -r_0 + 2r_0^3 \cos^3 u \sin u + L \sin^2 u \right] \frac{du}{\omega} \tag{10.3.8}
$$
Dividing throughout by \( \frac{2\pi}{A} \) and identifying the left hand side with \( \frac{dr}{dt} \) we have

\[
\frac{dr}{dt} = -r + \frac{L}{2}
\]

where \( r_0 \) has been allowed to regain its variable nature.

Equation 10.3.11, the approximation to 10.3.1, gives, after integration,

\[
r(t) = \frac{L}{2}(1 - e^{-t})
\]

Thus, the system eventually performs oscillations of amplitude \( \frac{L}{2} \) about the origin.

The above example shows how the constants \( A \) and \( \Theta_0 \) do not appear in the final result.

10.4. **Comparison with the Tracking Function Method**

In section 9.11 the tracking function method was applied to this system.

Assuming zero initial conditions the tracking function method established that the trajectories were contained
entirely in the y-axis between \( y = L \) and \( y = -L \) and that for given initial conditions a suitable rectangle could be erected of width sufficient to contain \( x_0 \) and height \( 2L \) to bound the oscillations.

The method of Luus and Lapidus only specifies the average radius of oscillation and gives no indication of its shape. The value of \( \frac{L}{2} \) is equivalent to oscillation along the y-axis between \( y = L \) and \( y = -L \), which agrees with the result obtained by the tracking function method. However, if circular oscillation is assumed, then the method of Luus and Lapidus is misleading since a sufficiently high amplitude of forcing function would cause the oscillation to enter the region \( xy > 1 \) which is unstable.

10.5. The Computing Algorithm

It can be seen that for systems of reasonable complexity the manipulations involved in obtaining 10.2.9 from 10.1.1 could be extremely laborious. In their original paper, Luus and Lapidus never consider systems whose polynomial right hand sides are of degree \( > 3 \). However, it has been found that for some systems a 3\(^{rd}\) degree expansion is insufficient to provide the stability information. (For example in the case of the chemical reactor system 1.7.4 an expansion of less than 9\(^{th}\) degree fails to give meaningful results.)
We describe below a computer algorithm which allows the consideration of expansions of arbitrarily high degree.

Consider the system 10.1.1, neglecting the forcing terms. This is permissible since they play no part in the analysis, appearing only as additional constants in the final expansion for \( \frac{dx}{dt} \) (see for example equation 10.3.10.)

Expanding the right hand sides of 10.1.1 as polynomials about the origin our system is

\[
\begin{align*}
\dot{x} &= \sum_{l=1}^{m} \sum_{j=1}^{l} p_{ij} x^{i-j+1} y^{j-1} \\
\dot{y} &= \sum_{l=1}^{m} \sum_{j=1}^{l} q_{ij} x^{i-j+1} y^{j-1}
\end{align*}
\]

Applying the transformation 10.2.1 we have

\[
\begin{align*}
\dot{r} &= \sum_{l=1}^{m} \sum_{j=1}^{l} \left\{ p_{ij} r^{i} \cos^{i-j+1} \theta \sin^{j-1} \theta \\
&+ q_{ij} r^{i} \cos^{i-j+1} \theta \sin^{j} \theta \right\}
\end{align*}
\]

As shown earlier it is sufficient to take \( \Theta = 0 \) and \( \Theta = t \) so that, allowing \( r \) to be constant \( (= r_0) \) in 10.5.2 and integrating from \( t = 0 \) to \( t = 2\pi \) we have
\[
\frac{dr}{dt} = \frac{r(2\pi) - r(0)}{2\pi} = \\
= \frac{1}{2\pi} \sum_{l=1}^{m} \sum_{j=1}^{i} \int_0^{2\pi} (p_{ij}\cos^{i-j+2} t \sin j-1 t \\
+ q_{ij}\cos^{i-j+1} t \sin j t) dt \tag{10.5.4}
\]

Thus, in the expansion \( F(r) \) for \( \frac{dr}{dt} \) of equation 10.2.9, the coefficient of \( r^i \) is given by

\[
a_i = \frac{1}{2\pi} \sum_{l=1}^{m} \sum_{j=1}^{i} \int_0^{2\pi} (p_{ij}\cos^{i-j+2} t \sin j-1 t \\
+ q_{ij}\cos^{i-j+1} t \sin j t) dt \tag{10.5.5}
\]

for \( i = 1, 2, \ldots, mf \)

The integrals appearing in 10.5.5 are of the form

\[
I(m,n) = \int_0^{2\pi} \sin^m x \cos^n x \, dx \tag{10.5.6}
\]

and may be computed for any \( m \) and \( n \) by the well known formulae.

\[
I(m,n) = \prod_{i=1}^{m/2} (m - 2i + 1) \prod_{i=1}^{m/2} (n - 2i + 1) \\
\prod_{l=1}^{(m+n)/2} (m + n - 2i + 2) \tag{10.5.7}
\]

for \( m \) and \( n \) both even and non-zero.
\[
\begin{align*}
\lambda(m, n) &= \frac{m/2}{2\pi} \left( \prod_{i=1}^{m/2} (m - 2i + 1) \right) \quad \text{for } m \text{ even} \\
\lambda(0, m) &= \frac{m/2}{2\pi} \left( \prod_{i=1}^{m/2} (m - 2i + 2) \right)
\end{align*}
\]

for \( m \) even

\[
\lambda(0, 0) = 2\pi \quad \text{10.5.8}
\]

and \( \lambda(m, n) = 0 \quad \text{otherwise} \quad \text{10.5.9} \)

After this computation we are left with

\[
\frac{dr}{dt} = \sum_{i=1}^{nf} a_i r^i 
\]

Now it will in general be impossible to integrate this equation. However the required estimate of stability may, as stated earlier, be inferred from a plot of \( \frac{dr}{dt} \) vs. \( r \) for the range of \( r \) of interest.

Having obtained this plot the effect of the forcing functions may be added. The forcing functions merely raise the curve by an amount depending on the nature of the forcing functions.

Fig. 10.1 shows the routine for computing the \( \lambda(m, n) \). The actual program for this is designated SUBROUTINE SINCOS. The whole program called IUMP is shown in appendix A.11.
10.6. Examples

10.6.1. The System 1.7.1.

After application of the method the approximate equation becomes

\[ \frac{dr}{dt} = \left( \frac{r}{2} - \frac{r^3}{8} \right) \epsilon \] 10.6:1

Thus, the limit cycle is stable and has average radius 2 independent of the value of \( \epsilon \). This accords well with actuality. A sinusoidal forcing function of amplitude \( I \) moves the curve of \( \frac{1}{\epsilon} \frac{dr}{dt} \) up by the amount \( \frac{I}{2\epsilon} \) so that the limit cycle moves to a larger radius (see fig. 10.3). The presence of the factor \( \frac{1}{\epsilon} \) indicates that increasing \( \epsilon \) increases the stability of the limit cycle since the larger \( \epsilon \) is, the larger must be the amplitude of the forcing function required to move the limit cycle by a given amount.

10.6.2. The System 1.7.2.

The approximate equation is

\[ \frac{dr}{dt} = -\frac{r}{2} \] 10.6.2

which indicates asymptotic stability in the large. The system is in fact only conditionally asymptotic stability. The actual DOA is shown in fig. 3.4.
10.6.3. The System 1.7.4.

Application of the method to this system, expanding the right hand sides of the system equations to degree 20, gave the dependence of $\frac{dr}{dt}$ upon $r$ shown in the table of fig. 10.4. This is not plotted as a curve because of the widely varying values. However, it can be seen that the sign of $\frac{dr}{dt}$ is positive for $r < 2$ and changes abruptly at $r = 2$. This indicates a stable limit cycle of radius 2. The actual limit cycle is stable but of radius .55. The discrepancy is presumably due to the highly non-linear character of the system equations and the resulting gross invalidity of the approximations made in the analysis.

10.6.4. The System 1.7.5. ($\alpha = .2, \beta = .075$)

For this system a 5th degree expansion of the right hand sides is necessary to obtain meaningful results. If this is done and the method applied, there is predicted a limit cycle of radius .43, unstable on both sides. The actual limit cycle is unstable, but of radius .3 (see fig. 11.7 which shows the actual limit cycle for this system, together with the result of applying the A.E.R.P. method).
11. THE METHOD OF THE ALTERNATING EXTREME RADIUS PATH.

11.1. Introduction.

The object of the alternating extreme radius path (A.E.R.P.) method is to predict the existence of limit cycle behaviour in non-linear systems. In this object the method is similar to the method of Iwus and Lapidus (chapter 10). However the procedure adopted for the achievement of this object is different.

The method is restricted to second-order systems and no extension to higher orders seems feasible.

A locus, the alternating extreme radius path, is constructed in the phase plane according to a particular policy. If this locus finally closes upon itself to produce a closed contour about the origin it is contended that a limit cycle exists.

Originally formulated as a graphical technique the method is well suited to machine computation.

11.2. Construction of the A.E.R.P.

Consider the system

\[
\begin{align*}
\dot{x} &= f(x, y) & f(0, 0) &= 0 \\
\dot{y} &= g(x, y) & g(0, 0) &= 0
\end{align*}
\]

11.2.1
If $f$ and $g$ are expanded as polynomials about the origin the equations 11.2.1 become

\[
\begin{align*}
\dot{x} &= ax + by + f_2(x, y) \\
\dot{y} &= cx + dy + g_2(x, y)
\end{align*}
\]

where $f_2$ and $g_2$ are polynomials of degree $\geq 2$.

Further, let us assume that the linear parameters $a$, $b$, $c$, $d$ satisfy the following conditions

\[
ad - bc > 0 \quad 11.2.3
\]

\[
b/c < 0
\]

Then the authors of the method claim that sufficient information concerning stability may be obtained from the three loci

\[
\begin{align*}
\dot{x} &= f(x, y) = 0 \\
\dot{y} &= g(x, y) = 0 \\
\dot{R} &= \frac{d}{dt} \sqrt{x^2 + y^2} = 0
\end{align*}
\]

where $\mu^2 = -b/c \quad 11.2.7$

The A.Ž.R.P. is constructed as follows. At some point in the phase plane the signs of $\dot{x}$, $\dot{y}$ and $\dot{R}$ will define a sector allowable to the trajectory from $P$. (by a process
exactly similar to that used in the tracking function method (see section 9.2). From $P$ the $\Delta E R P$ is extended according to the following policy:

(1). If $\dot{R}$ is positive the locus is extended along the path of minimum radius (i.e. along the most stable arm of the allowable sector).

(2). If $\dot{R}$ is negative the locus is extended along the path of maximum radius (i.e. along the most unstable arm of the allowable sector).

If this locus closes upon itself to produce a closed cycle about the origin, limit cycle behaviour is inferred.

A point of confusion arises here concerning the insertion of the factor $K^2$ in the expression 11.2.6 for $\dot{R}$. Because of this factor $R$ is not the actual radius and the curves $R = \text{constant}$ are not circles but ellipses. The question arises; in the plotting of the $\Delta E R P$ should the path comprise arcs of circles or ellipses? The authors of the original work appear to use circles, however in that case will not necessarily remain in the allowable sector. Since, as the authors admit in the original work, there is no mathematical justification for the method (at present), there is consequently no rigorous method of answering the question posed above.

The following example illustrates the problem.
11.3 Illustration of the Ambiguity in the Choice of Radius Function

Consider the linear system

\[ \begin{align*}
\dot{x} &= y \\
\dot{y} &= -(x + y)/16
\end{align*} \]  

11.3.1

Thus, by 11.2.2 we have

\[ a = 0 \quad b = 1 \quad c = -1/16 \quad d = -1/16 \]

so that

\[ L^2 = -b/c = 16 \]  

11.3.2

The loci of interest are

\[ \begin{align*}
\dot{x} &= 0 \quad \text{or} \quad y = 0 \\
\dot{y} &= 0 \quad \text{or} \quad y = -x
\end{align*} \]  

11.3.3

11.3.4

while

\[ \dot{R} = \frac{1}{2} (x^2 + 16y^2) \frac{d}{dt} \left( x^2 + 16y^2 \right) \]

\[ = -y^2 (x^2 + 16y^2) \frac{d}{dt} \]

11.3.5

11.3.6

\[ \dot{R} \] is therefore negative or zero everywhere and the A.E.R.P. always takes the path of maximum radius. Fig. 11.1.(a) shows the configuration in the state plane. The symbols \( \Delta \) indicate the allowable sectors. The A.E.R.P. is the locus ABCDEFGH.

Now consider the same system after applying the linear transformation
The system equations become

\[
\begin{align*}
X &= x/4 & \text{ll.3.7} \\
X &= y/4 & \text{ll.3.8} \\
Y &= -X/4 - y/4
\end{align*}
\]

Then, \( a = 0 \quad b = 1/4 \quad c = -1/4 \quad d = -1/4 \) and

\[ m^2 = 1 \quad \text{ll.3.9} \]

Thus by this transformation the function \( R \) is the actual radius in the \( Xy \) plane.

Fig. 11.1.(b) shows the A.E.R.P. for this case (the locus PQRSTUW). This locus is re-plotted in the \( xy \) plane and there results the locus ABC'D'E'F'G'H'. This locus corresponds more closely to the trajectories of the system than does the original trajectory.

In this linear case both methods imply stability since both converge to the origin. However the locus produced after the transformation indicates a greater degree of instability. It appears quite possible that there exist non-linear systems for which the original method would give spurious results.

Hence, before applying the method to the system 11.2.2 it is first of all necessary to apply the transformation:

\[ X = \Delta x \quad \text{where} \quad \Delta = (-c/b)^{1/2} \]

so that the function \( R \) is the actual radius.
11.4. The Computing Algorithm

Although the method is in principle straightforward, for non-linear systems of reasonable complexity it is necessary to use a computer to plot the loci \( \dot{x} = 0, \dot{y} = 0 \) and \( \ddot{R} = 0 \). After this there remains the task of plotting the A.E.R.P. It appeared worthwhile to seek an algorithm to enable a computer to do the whole job. Such an algorithm is described below. This algorithm is both fast and accurate, attributes not possessed by the graphical procedure. The method is very similar to the tracking function method of section 9.4.

Consider fig. 11.2. At some point \( P (x,y) \) in the phase plane, \( f(x,y) \), \( g(x,y) \) and \( \ddot{R} \) are computed. A new point \( T \) distant \( s \) from \( P \) on the line \( x = \text{constant} \) through \( P \) is obtained. The coordinates \( (x_1, y_1) \) of this point are just

\[
\begin{align*}
  x_1 &= x \\
  y_1 &= y + s \cdot g / |g| 
\end{align*}
\]

Similarly a new point \( U \) on the line \( y = \text{constant} \) through \( P \) is obtained with coordinates \( (x_2, y_2) \) given by

\[
\begin{align*}
  x_2 &= x + s \cdot f / |f| \\
  y_2 &= y 
\end{align*}
\]

Now according to the sign of \( \ddot{R} \), one or both of these points will lie on the allowable side of the circle centred
at the origin and passing through P. Thus if $R$ is positive and

$$r_1^2 \equiv (x_1^2 + y_1^2) > (x^2 + y^2)$$

then $T$ is acceptable. Similarly it may be calculated whether or not $U$ is acceptable. If both are acceptable we are done. If either is not acceptable it is replaced by the point $W$ distant $s$ from $P$ along the circle centred at the origin passing through $P$. The direction from $P$ to $W$ is obvious since $W$ must lie in the sector $UP$ as the others are not allowable.

In any event, at this stage there are two points which define the allowable sector. In fig. 11.2 we have assumed that $x$ is positive, $y$ and $R$ are negative. The forbidden sides are shown shaded.

At this stage, depending on the sign of $R$ the A.E.R.P. is extended from $P$ to one or other of the two remaining points according to the A.E.R.P. policy. In fig 11.2 the allowable sector is $TPN$ and since $R$ is negative the most extreme path is adopted and the locus extended to $W$ which becomes the new $P$. The entire process is repeated here and so on.

The flow diagram for the method is shown in fig. 11.3 and the program is given in appendix A.12.
11.5. Examples

11.5.1. The System 1.7.4

Fig. 11.4 shows the A.E.R.P. method applied to the chemical reactor system. Also shown for comparison is the actual limit cycle. There is reasonably good agreement between the two boundaries.

11.5.2. The System 1.7.1

Fig. 11.5 shows the method applied to this system for $\varepsilon = 0.1$ and 1.0. For the case $\varepsilon = 0.1$ the A.E.R.P. is almost exactly in agreement with the actual limit cycle which is approximately circular of radius 2.0. For the case of $\varepsilon = 1.0$ the A.E.R.P. is less representative of the actual limit cycle which is more diamond shaped (see ref. 40.). In both cases the A.E.R.P. is in agreement with the method of Luus and Lapidus (chapter 10) which predicts limit cycle existence with radius 2.0 irrespective of the value of $\varepsilon$.

11.5.3. The System 1.7.5. ($\alpha = 0.2, \beta = 0.075$)

Fig. 11.6 shows the physical set-up of this surge tank system.

Fig. 11.7 shows the result of applying the A.E.R.P. method together with the actual limit cycle.
11.5.4. The System 1.7.2.

For this system, as will be seen, the numerical procedure provides less information than the graphical method. While the numerical procedure does indicate the fact that the system does not possess a limit cycle it is in addition possible to determine by the graphical method a region of stability.

Consider fig. 11.8 which shows the A.E.R.P.'s from various starting positions. None of these A.E.R.P.'s converges to a closed curve.

It is evident that any A.E.R.P. starting inside the region ABCDEF eventually attains the origin, while those starting outside diverge to infinity. The region ABCDEF may therefore be considered an estimate of the DOA of the system. The actual DOA is bounded by the curves GG'. The small shaded regions R are in fact unstable (i.e. the response to initial conditions in R diverges to infinity) but the A.E.R.P. method indicates that they are stable. While it is true that we are here asking more of the method than its authors claim for it, this example illustrates the care which must be exercised in interpreting the results obtained.

It is interesting to note that the A.E.R.P. method did indicate that the stability of this system was not 'in the large'. The method of Luus and Lapidus erroneously indicated asymptotic stability in the large (see chapter 10).
11.6. Comments

The A.E.R.P. method is eminently suitable for machine computation. Although there is not any rigorous mathematical justification for the method, the examples show that the method does appear to work insofar as a limit cycle is predicted if and only if a limit cycle does exist.

As shown by the example 11.5.4, when the method is used to estimate stability rather than merely the existence of limit cycle behaviour the results must be viewed with scepticism. It does not appear inconceivable that there exist systems for which the method would erroneously predict the existence or otherwise of limit cycle behaviour.
12. CONCLUSIONS

We attempt here to bring together the main conclusions resulting from the investigation, and which have been stated and discussed throughout the text particularly in the 'comments' sections.

For the construction of Liapounov functions the method of Zubov emerged clearly as the most powerful, at least as far as numerical calculation is concerned. The fact that the convergence of the RAS to the actual DOA is non-uniform in general is the major drawback of the method. However, as has been shown, the methods of Krassovski and Ingwerson also possess a similar feature in that there is no guarantee that a high degree Liapounov function will be better than a low degree one.

Investigation into the reason for this non-uniformity of convergence of the Zubov method would appear to be a worthwhile area for further research.

We have not obtained results for systems of order greater than second, for two reasons. First the formidable difficulties in involved in extending the methods is obvious from section 5.6, so that while in principle such extension is possible, in practice the time and effort required could not be justified. Second, it appears reasonable to suppose that the efficiency of a particular method can be gauged by its efficiency in dealing with second order systems. In other words if a method is efficient in dealing with
second order systems it seems plausible that this efficiency will extend to higher order systems.

As far as computing time is concerned, this ranges from approximately 2 seconds for a quadratic Liapounov function to approximately 2 minutes for a 20th degree function, for the Zubov procedure (I.C.T. 1905 computer). The other construction procedures are generally faster than this but not significantly so.

As shown in section 5.6, the major difficulty in extending the method of Zubov to higher order systems is a problem of notation. With the simple notation used, there arise hyper-dimensional blocks of simultaneous equations. It would appear useful therefore to investigate either a means of solving simultaneous equations in this form, or a more efficient notation.

The method of Weissenberger, though straightforward conceptually, suffers from a problem of dimensionality. As shown in section 8.3, Weissenberger's conjecture for overcoming this problem is invalid, due no doubt to the non-uniformity of convergence of the Zubov method. Thus, the only effective method for obtaining high degree optimum Liapounov functions is the approach developed and described in chapter 7. (Although it must be stressed that such Liapounov functions are not strictly optimum but only optimum for the particular construction procedure adopted.) This approach is applicable to the methods of Krasovski, Ingwerson and Zubov, and again it is the method of Zubov which provides the best Liapounov functions.
It is worthwhile to note here that in a recent survey paper Gurel and Lapidus (1968) state that, "In a paper to be published, ..... (Luocke and McGuire) have derived a computational method to determine the A matrix (our matrix B of Krassovski's method; equation 2.1.2) so as to yield the largest RAS for the two variable problem." This is precisely what has been achieved by the method developed in chapter 7, not only for the method of Krassovski, but also for the methods of Ingwerson and Zubov.

The method of Rodden for plotting the boundary of the RAS is described in chapter 6 is the only numerical method known at present for achieving this purpose. We have described several modifications to the method and also certain pathological configurations (figs. 6.7 and 6.8) which can arise and which pose considerable problems.

The computing time to plot the boundary depends upon the various parameters used in the procedure. In our case, typical times were of the order of 100 seconds for quadratic Liapounov function to 2000 seconds for 20th degree. As can be seen the times required for construction of the Liapounov functions are negligible compared with the times required for the plotting of the RAS.

Again, no extension to higher order systems has been developed. Rodden has described the analysis of a 3rd order system. However for orders > 2 the method would appear to be infeasible except for particularly simple systems and low degree Liapounov functions.
As a general comment on the direct method of Liapounov then, it is possible to conclude that it is not particularly well suited to numerical computation for the following reasons. 1. The system equations (at present) must be able to be represented by polynomials. 2. For high order systems the algorithms are extremely difficult to program and the RAS almost impossible to plot or to visualise. 3. The Liapounov functions give only a poor approximation to the DOA (see fig. 7.3 which shows the best RAS obtained for the system 1.7.1.A; this is only an infinitesimal portion of the actual DOA.)

Other methods for constructing Liapounov functions have been investigated but no algorithms have been obtained. These methods are the methods of Walker & Clark\(^{31}\), Infante & Clark\(^{30,32}\) and the variable gradient method\(^{33}\). The common feature of these methods preventing the formulation of a computing algorithm is that each requires the selection of a large number of coefficients - this selection to be made in the light of the analyst's experience and ingenuity rather than by a formal procedure. It would be worthwhile to continue research in this direction to try to find suitable algorithms for these methods.

The tracking function method (chapter 9) is often more germane and is certainly simpler to use. However it is limited (at present) to second order systems. It would be worthwhile to pursue extension to higher order systems - to which end a more analytic treatment of
the method may be fruitful.

A point of interest is the use of the tracking function method together with the direct method of Lyapunov, for locating the position of limit cycles. Consider fig. 7.15. Curve B is the best RAS obtained for the system 1.7.4A. Since any trajectory starting inside this region asymptotically approaches the origin, then, if the direction of time is reversed (t replaced by -t), any trajectory starting inside B will eventually leave B. This time reversal transforms system 1.7.4A to system 1.7.4. Fig. 9.5, curve B shows the best bound on the trajectories of system 1.7.4 obtained by the tracking function method. Thus any trajectory starting inside B can never leave the region bounded by B. Thus it can be seen that the limit cycle for system 1.7.4 must lie in the region between the curve B of fig. 9.5 and the curve B of fig. 7.15. Similarly it can be deduced that the limit cycle of system 1.7.1 (ε = .1) must lie between the curve B (fig. 9.11) and the 20th degree curve of fig. 5.7.

For this location of the position of a limit cycle, the closest bounds are required for the greatest precision so that the Lyapunov function for the stable system should be obtained by the optimisation procedure of chapter 7 while the bound on the unstable response should be obtained by the computing algorithm for the tracking function method as described in section 9.4.
The method of Luus and Lapidus (chapter 10) is straightforward and eminently suitable for machine computation. However the usefulness of the method is severely handicapped by the fact that there is no way of assessing the effects of the gross approximations involved. It is advised, therefore, that the method be used in conjunction with another, more rigorous method, perhaps to give an initial indication of the stability behaviour, rather than on its own.

This comment applies equally to the A.E.R.P. method (chapter 11). A possible area of further work is an investigation into the question of whether or not there is any justification for the policy adopted. Certainly in those cases analysed herein the method was successful in its aim of predicting limit cycle existence. However an example is given (fig. 11.8) for which the method is shown to be misleading if it is applied to the analysis of general stability behaviour.

In ref. 5, Kalman and Bertram offer the following comment.
"The abstract method (of Liapounov) becomes a concrete one whenever explicit expressions can be found for a Liapounov function. This is always possible in the linear case....... In the linear non-stationary case or non-linear cases, no straightforward methods are available for doing this. Further applied mathematical research should be directed toward developing efficient digital computer programs for finding Liapounov functions", while as recently as 1965, Dorf suggests that "automatic digital computer programs for testing the
Liapounov functions would be of great value. Furthermore if a program could be written which generates the Liapounov function the analyst would have a complete Liapounov stability test program."

It is hoped that the research described in the foregoing has, to some extent, contributed to the implementation of these proposals, and has indicated the numerical application of other non-Liapounov methods.
ACKNOWLEDGEMENTS

It is with pleasure and gratitude that I acknowledge the assistance and encouragement given by Professor C. Storoy, who supervised this research.

Thanks are also due to the staff of the Computer Centre, Loughborough University of Technology, for many services.
NOTE

In the references which follow, those numbered 15, 17, 18, 19, 20, 26, 27 and 28 were published as a result of the research described in this thesis.
REFERENCES.


APPENDICES.

In the following appendices the various programs developed and used in the investigation are given. The notes accompanying the programs are intended only to enable the interested reader to use the programs and give no information regarding the logic involved. This logic is described in the main text and is also given in flow diagram form.
Appendix A.1. Program for Krassovski's Method

The program, written in FORTRAN 4 is designated SUBROUTINE INGVIER. It computes the coefficients $a_{ij}$ of the Liapounov function $V$ given by equation 2.2.12 for the system 2.2.1.

The call statement is

\[ \text{CALL INGVIER(MF, P, Q, C)} \]

where

MF is the degree of the system equations,

$P$ and $Q$ are 2-dimensional arrays of the coefficients $p_{ij}$ and $q_{ij}$ of the system equations.

$C$ is a 1-dimensional array containing the 3 coefficients $c_{ij}$ of the positive definite matrix $C$ where $C$ is defined by equation 2.2.2. Thus $C(1)$ contains $c_{11}$, $C(2)$ contains $c_{12}$ and $C(3)$ contains $c_{22}$.

The output is the matrix $C$ in the form

\[
\begin{bmatrix}
 c_{11} & c_{12} & c_{22}
\end{bmatrix}
\]

and the matrix $B$ in the form

\[
\begin{bmatrix}
 b_{11} & b_{12} & b_{22}
\end{bmatrix}
\]

both in FORMAT(20X, 3E16.10).

The coefficients $a_{ij}$ of the Liapounov function are held in array $A$ which is not outputted but held in the COMMON area.
SUBROUTINE INGWER(MF,P,Q,B)
DIMENSION P(10,11),Q(10,11),A(20,21),B(3)
COMMON A
DO 1 K=1,20
DO 1 J=1,21
1 A(K,J)=0.
WRITE(2,4)B(1),B(2),B(3)
DET=(Q(1,2)*P(1,1)-Q(1,1)*P(1,2))*(P(1,1)*Q(1,2))
 T1=-B(1)*P(1,1)*Q(1,2)-B(1)*Q(1,2)*2+B(1)*P(1,2)*Q(1,1)+2.*Q(1,1)
1*Q(1,2)*B(2)-Q(1,1)+2*B(3)
 T2=-2.*B(2)*P(1,1)+Q(1,2)*B(3)+P(1,1)*Q(1,1)+B(1)*P(1,2)*Q(1,2)
 T3=-P(1,1)*2*B(3)-P(1,1)*Q(1,2)*B(3)+2.*B(1)*P(1,2)*B(2)+P(1,2)
1*Q(1,1)*R(3)+Q(1,2)*B(1)
B(1)=T1/DET
B(2)=T2/DET
B(3)=T3/DET
WRITE(2,4)B(1),B(2),B(3)
4 FORMAT(2X,3E17.10)
DO 2 I=1,MF
DO 2 J=1,I+1
DO 2 K=1,MF
DO 2 IS=I+1,K+1
2 A(I+K,J+IS-1)=A(I+K,J+IS-1)+B(1)*P(I,J)*P(K,IS)+2.*B(2)*P(I,J)*Q(K,1)
1,IS)+B(3)*Q(I,J)*Q(K,IS)
RETURN
END

END OF SEGMENT, LENGTH 568, NAME INGWER
Appendix A.2. Program for Szego's Method: System in General Form

This program, written in FORTRAN 4, computes the coefficients \( a_{ij} \) of the Liapounov function \( V \) of equation 3.4.16 for the system defined in 3.4.4. The call statement is

\[
\text{CALL SZEGO(MV,MF,P,PD,Q,QD,AA,D,A,B)}
\]

where \( MV \) is the degree of the Liapounov function.

MF is the degree of the system equations.

\( P, PD, Q \) and \( QD \) are 1-dimensional arrays containing the elements \( p_i, p'_{i}, q_i \) and \( q'_{i} \) of the system equations.

\( AA \) is a two dimensional array of the coefficients \( a_{ij} \) of the Liapounov function (see equation 3.4.16).

\( D \) is a 1-dimensional array of the coefficients \( d_{ij} \) of \( V \) (see equation 3.4.19).

\( A \) and \( B \) are one dimensional arrays of the coefficients \( a_j \) and \( b_j \) of the Liapounov function in the form of equation 3.4.15. \( A \) and \( B \) may be considered as working space.

The program gives no output.
SUBROUTINE SZE00(MV,MF,P,PD,Q,QD,AA,D,A,B)
DIMENSION P(19),Q(19),PD(19),QD(19),A(19),B(19),D(39),AA(20,21)
DO 1 K=1,19
A(K)=0.
1 B(K)=0.
DO 31 K=1,20
DO 31 J=1,21
31 A(K,J)=0.
B(1)=-2.*QD(1)/PD(1)
DO 5 N=2,MV-1
XN=N
DO 6 I=1,N-1
XI=I
IF(N-I+1-MF)7,7,6
7 B(N)=B(N)-P(N-I+1)*B(I)*XI
6 CONTINUE
IF(N-MF)8,8,5
8 B(N)=B(N)-2.*QD(N)
5 B(N)=B(N)/(XN*PD(1))
A(I)=(-P(1)*B(1)-QD(1)*B(1)+2.*Q(1))/(2.*PD(1))
DO 15 N=2,MV-1
XN=N
DO 16 I=1,N-1
XI=I
IF(N-I+1-MF)17,17,16
17 A(N)=A(N)-A(I)*P(N-I+1)*XI+QD(N-I+1)
16 CONTINUE
DO 18 I=1,N
XI=I
IF(N-I+1-MF)19,19,18
19 A(N)=A(N)-B(I)*P(N-I+1)*XI+QD(N-I+1)
18 CONTINUE
IF(N-MF)21,21,15
21 A(N)=A(N)-2.*Q(N)
15 A(N)=A(N)/(XN*PD(1))
DO 40 I=2,MV
AA(I,1)=A(I-1)
40 AA(I,2)=B(I-1)
AA(2,3)=1.
DO 140 K=1,38
140 D(K)=0.
MM=MV+MF-1
DO 100 N=2,MM
XN=N
DO 100 I=1,N-1
XI=I
IF(N-I-MV+1)101,101,100
101 IF(N-I-MF)102,102,100
102 \( D(N) = D(N) + Q(I) * (N-1) + P(I) * A(N-1) * (XN-XI+1) \)
100 CONTINUE
RETURN
END

END OF SEGMENT, LENGTH 714, NAME SZEGO
Appendix 3  Program for Szego's Method; Systems in Companion Form

This program, written in FORTRAN 4, computes the coefficients $a_i$ and $b_i$ of the Liapounov function of equation 3.4.15 for the system 3.5.2.

The required data is as follows:

$NQ$, the degree of the system equations ($m$ in 3.5.2) in FORMAT(13).

$P(i), Q(i)$, a list of the coefficients $q_i$ and $q_i$ of the system equations 3.5.2 in FORMAT(2E12.5).

The output is the Liapounov function $V$ in the form of equation 3.4.15 and $\dot{V}$ in the form of equation 3.4.19.
LIST(LP)
PROGRAM(M101)
TRACE
INPUT1=CRO
OUTPUT2, (MONITOR)=LP0
END
MASTERSZEGOSMETHODCOMPFORM
DIMENSION P(40), Q(40), A(79), B(79)
READ(1,1) NQ
1 FORMAT(I3)
DO 2 I=1, NQ
2 READ(1,3) P(I), Q(I)
3 FORMAT(2E12.5)
   NN=2*NQ+1
DO 4 I=1, NN
   A(I)=0.
4 B(I)=0.
   DO 5 I=1, NQ
      XI=I
5   B(I)=B(I)+2.*Q(I)/XI
   DO 6 IR=1, NN
      XIR=IR+1
   6    DO 7 I=1, IR
      IF(IR-I+1=NQ) 8, 8, 7
      A(IR)=A(IR)-B(I)*Q(IR-I+1)
7    CONTINUE
     IF(IR=NQ) 9, 9, 6
9    A(IR)=A(IR)-2.*P(IR)
6    A(IR)=A(IR)/XIR
8   WRITE(2,10) A(I), B(I)
10 FORMAT(2I2,E12.5), 14H 1.000000E 00)
   DO 11 I=2, NN
11  WRITE(2,12) A(I), B(I)
12  FORMAT(2I2,E12.5))
   DO 13 IR=1, NQ
      C=0.
   13    DO 14 I=1, IR
      C=C+B(I)*P(IR-I+1)
14   WRITE(2,15) C
15  FORMAT(40X,E12.5)
STOP01
FINISH
Appendix A.4. Program for Ingwerson's Method

This program, written in FORTRAN 4 and designated SUBROUTINE INGWER, computes the coefficients $a_{ij}$ of the Liapounov function in standard form (see equation 2.2.12) by Ingwerson's method, for the system of equations 4.4.2 and 4.4.3.

The call statement is

```
CALL INGWER(MF,P,Q,C)
```

where MF is the degree of the system equations.

P and Q contain the elements $p_{ij}$ and $q_{ij}$ of eqns. 4.4.2 and 4.4.3.

C is a 1-dim. array containing the elements $c_{11}, c_{12}$ and $c_{22}$ of the matrix $C$ of eqn. 4.4.7. Thus, $C(1)$, $C(2)$ and $C(3)$ contain respectively $c_{11}$, $c_{12}$ and $c_{22}$.

The coefficients $a_{ij}$ of $V$ are stored in COMMON in array P (in the subroutine). This is outputted.

Also shown is the main program which allows the generation of families of Liapounov functions of different degrees and for different $C$ matrices.
SUBROUTINE INGWER(MF,A,B,C)
DIMENSION A(10,11),B(10,11),P(20,21),C(3)
COMMON P
MFF=2*MF.
DO 1 K=1,MFF:
  KK=K+1
  DO 1 J=1, KK
  1 P(K,J)=0.
DO 2 I=1,MF
  XII=I
  DO 2 J=1, MF
  XJ=J
  2 P(I+J,1)=P(I+J,1)+(-C(1)*A(I,1)*B(J,2)*XI-C(1)*B(I,2)*B(J,2)+C(1)
  1*A(I,2)*B(J,1)*XJ+2.*C(2)*B(I,1)*B(J,2)*XI-C(3)*B(I,1)*B(J,1)*XI-
  2*XJ)/((XI+XJ-1.)*(XI+XJ))
  P(I+J,K+1)=P(I+J,K+1)+(-C(3)*A(I,K)+A(J)-C(3)*A(I,K)+B(J,K+1)
  1)*XJ+2.*C(2)*A(I,K)+A(J,K+1)*XI+C(3)*A(I,K)+1)*B(J,K)*XI-C(1)*A(I
  2,K+1)*A(J,K+1)*XI*XJ)/((XI+XJ-1.)*(XI+XJ)).
DO 3 K=1, I
  XK=K
  DO 3 L=1, J
    XL=I
    3 CONTINUE
II=I+1
  DO 4 K=2, II
    XK=K
    KK=K+1
    JJ=J+1
  4 CONTINUE
II=I+1
  DO 4 K=2, II
    XK=K
    DO 4 L=2, JJ
      XL=L
      4 CONTINUE
II=I+1
  DO 4 K=2, II
    XK=K
    DO 4 L=2, JJ
      XL=L
      4 CONTINUE
II=I+1
  IF(P(2,1))5,6,7
  5 CONTINUE
IF(P(2,3))5,6,7
9 P(K,J)=-P(K,J)
GO TO 7
8 WRITE(4, 10)
10 FORMAT('///, 18H L.F. NOT +VE DEF.,///)
7 DO 15 K=2, KK
     KK=K+1
     WRITE(4, 11)(P(K, J), J=1, KK)
11 FORMAT(1H , E12.5)
15 CONTINUE
RETURN
END

END OF SEGMENT, LENGTH 867, NAME INGWER
MASTER FAMILIES OF ING

DIMENSION P(10,11),Q(10,11),A(20,21),C(3)

COMMON A

READ(1,1)MF,NFAMIL

1 FORMAT(2I4)

WRITE(2,2)

2 FORMAT(24X,2H Families of Stability Boundaries/ 59X,3HFOR:,/26X,28hw

VARIOUS INGHETSON C MATRICES,/) WRITE(2,3)NFAMIL

3 FORMAT(6X,2H Number of Sets of Families; I4)

DO 5 K=1, MF

5 READ(1,4)(P(K,J),J=1,KK) NN=O

4 FORMAT(1H ,6E17.10)

DO 6 K=1, MF

KK=K+1

6 READ(1,4)(Q(K,J),J=1,KK)

7 READ(1,7)BOUND

7 FORMAT(14)

READ(1,8)C((1),C(2),C(3)

8 FORMAT(3E12.5)

WRITE(2,9)C((1),C(2),C(3)

9 FORMAT(3H Elements of C Matrix..... C11 = E12.5, //, 29X, 6HC12 =

1,E12.5, //, 29X, 6HC22 = E12.5, //)

CALL INGHETON(MF,P,Q,C)

DO 10 K=1, NBOUND

10 READ(1,11)MV

11 FORMAT(14)

WRITE(2,200)MV

200 FORMAT(14H ORDER OF L.F., I4)

CALL ITIME(ISEC)

WRITE(2,100)ISEC

100 FORMAT(5H TIME, I6, 3HSEC)

CALL VALUE(MV,1,AVRAD,1)

CALL ITIME(ISEC)

10 WRITE(2,100)ISEC

NN=NN+1

IF(NN-NFAMIL)17,18,18

18 STOP 01

END

END OF SEGMENT, LENGTH 227, NAME FAMILIESOFING
Appendix A.5. Program for Zubov’s Method.

This program, written in FORTRAN 4 and designated SUBROUTINE ZUBOV, computes the coefficients \( a_{ij} \) of the Liapounov function \( V \) (eqn. 5.4.10) for the system 5.4.1 by Zubov’s method.

The call statement is

```fortran
CALL ZUBOV(MV,MF,MPHI,P,Q,R,IND)
```

where \( MV, MF, \) and \( MPHI \) are respectively the degrees of \( V \) the system equations and \( \phi \). See eqns. 5.4.10, 5.4.1 and 5.4.5.

\( P \) and \( Q \) contain the coefficients \( p_{ij} \) and \( q_{ij} \) of the system equations (eqns. 5.4.1).

\( R \) contains the coefficients \( r_{ij} \) of the \( \phi \)-function (eqn. 5.4.5).

\( IND \) is set to 0 for the regular procedure and any other value for modified.

The coefficients \( a_{ij} \) of \( V \) are held in COMMON in array A.

There is no output.

Also shown is SUBROUTINE GJEL, a simultaneous equations routine, and the main program which permits the generation of families of Liapounov functions of various degrees and for different \( \phi \)-functions.
SUBROUTINE ZUBOV(MV, MF, MPHI, P, Q, R, IND)

DIMENSION RHS(11), SQ(11,11), P(10,11), Q(10,11), R(3,4), AA(11), A(10,11)

COMMON A

DO 1 M=2, MV
  MM=M+1
  XM=M
  DO 2 K=1, MM
  DO 3 J=1, MM
  SQ(J, K)=0.
  2 RHS(K)=0.
  DO 4 J=1, M
  XJ=J
  DO 4 K=1, 2
  SQ(J*K-1, J)=SQ(J*K-1, J)+P(1, K)*(XM-XJ+1,)
  4 DO 5 J=2, MM
  XJ=J
  DO 5 K=1, 2
  SQ(J*K-2, J)=SQ(J*K-2, J)+Q(1, K)*(XJ-1,)
  5 IF(M=MPHI)30,30,31
  DO 6 J=1, MM
  RHS(J)=RHS(J)-R(M, J)
  6 IF(M=2)8,7,8
  MMM=M-1
  DO 9 I=2, MMM
  IF(I=MF)17,17,16
  XJ=J
  DO 9 J=1, II
  NN=M-1
  DO 9 K=1, NN
  XK=K
  9 RHS(J*K-1)=RHS(J*K-1)-P(I, J)*A(M-I+1, K)*(XM-XI-XK+2.)
  DO 10 I=2, MMM
  IF(I=MF)19,19,20
  II=I+1
  DO 10 J=1, II
  NN=M+1
  DO 10 K=2, NN
  XK=K
  10 RHS(J*K-2)=RHS(J*K-2)-Q(I, J)*A(M-I+1, K)*(XK-1,)
  IF(IND)7,12,7
  12 IF(M=3)100,7,100
  MMM=M-2
  DO 13 I=2, MMM
  IF(I=MPHI)21,21,7
  II=I+1

192.
DO 13 J=1,II
MX=M+I*1
DO 13 K=1,MX
13 RHS(J*K-1)=RHS(J*K-1)+R(I,J)+A(M-1,K)
7 CALL GAEL(SQ,AA,RHS,MM)
DO 16 J=1,MM
16 A(M,J)=AA(J)
1 CONTINUE
RETURN
END

END OF SEGMENT, LENGTH 660, NAME ZUBOV
SUBROUTINE GAEL(A, VAR, B, N)
DIMENSION A(11,11), B(11), IPVOT(11), INDEX(11,2), PIVOT(11), VAR(11)
EQUVALENCE (IROW, IROW), (ICOL, JCOL)

17 DET=1.
DO J=1,N
17 IPVOT(J)=0
DO I=1,N
13 T=0.
DO J=1,N
13 IF(IPVOT(J)=1)13,9,13
DO K=1,N
13 IF(IPVOT(K)=1)83,23,83
43 IF(ABS(T)=ABS(A(J,K)))83,23,23
83 IROW=J
ICOL=K
T=A(J,K)
23 CONTINUE
9 CONTINUE
IPVOT(icol)=IPVOT(icol)*1
IF(IRROW-ICOL)<73,109,75
73 DET=-DET
DO L=1,N
12 T=A(IRROW,L)
A(IRROW,L)=A(icol,L)
12 A(ICOL,L)=T
33 T=B(IRROW)
B(IRROW)=B(ICOL)
B(ICOL)=T
109 INDEX(1,1)=IRROW
INDEX(1,2)=ICOL
PIVOT(I)=A(ICOL,ICOL)
DET=DET*PIVOT(I)
A(ICOL,ICOL)=1.
DO L=1,N
205 A(ICOL,L)=A(ICOL,L)/PIVOT(I)
52 B(ICOL)=B(ICOL)/PIVOT(I)
347 DO L=1,N
139 IF(L-ICOL)<21,135,21
21 T=A(L,ICOL)
A(L,ICOL)=0.
DO L=1,N
89 A(L,L)=A(L,L)-A(ICOL,L)*T
B(L)=B(L)-T*ICOL,T
135 CONTINUE
222 DO 3 I=1,N
3 L=L+1
4 IF(INDEX(L,1)<INDEX(L,2))19,3,19
19 JROW=INDEX(L,1)
JCOL=INDEX(L,2)
DO 549 K=1,N
T=A(K, JROW)
A(K, JROW)=A(K, JCOL)
A(K, JCOL)=T
549 CONTINUE
3 CONTINUE
DO 901 K=1,N
VAR(K)=B(K)
RETURN
END

END OF SEGMENT, LENGTH 577, NAME GAEL
MASTER FAMILIES OF ZUB

DIMENSION P(20,21),C(20,21),R(10,11),A(20,21)

COMMON/AREA/P,OR/AREA2/A

WRITE(2,100)

100 FORMAT(24X,24HfAMILIES OF STABILITY_BOUNDARIES/39X,3HFOR,/,26X,27

ARIOUS ZUBOV PHI FUNCTIONS,///)

READ(1,1)NFAMIL

C MF IS ORDER OF RHS OF SYSTEM EQUATIONS

C NFAMIL IS NUMBER OF FAMILIES

1 FORMAT(214)

WRITE(2,103)NFAMIL

101 FORMAT(6X,24H NUMBER OF SETS OF FAMILIES;14)

B=1.

IPRINT=1.

DO 2 K=1,MF

KK=K+1

READ(1,3)(P(K,J),J=1,KK)

3 FORMAT(1H,12,17,10)

C P IS ARRAY OF COEFFICIENTS OF RHS OF FIRST SYSTEM EQUATION

DO 2 CONTINUE

C Q IS ARRAY OF COEFFICIENTS OF RHS OF SECOND SYSTEM EQUATION

C Q=17 READ(1,8)NP=1,NBOUND,IND

8 FORMAT(314)

C NBOUND IS NUMBER OF BOUNDARIES PER PHI FUNCTION

C MPH1 IS ORDER OF PHI FUNCTION

C IND IS SET AT 0 FOR REGULAR ZUBOV CONSTRUCTION,1 FOR MODIFIED

WRITE(2,103)

103 FORMAT(6X,24H PHI FUNCTION)

DO 6 K=2,MPH1

KK=K+1

READ(1,3)(R(K,J),J=1,KK)

6 CONTINUE

C R IS ARRAY OF COEFFICIENTS OF PHI FUNCTION

WRITE(2,3)(R(K,J),J=1,KK)

CONTINUE

DO 7 K=1,NBOUND

READ(1,5)MV

7 CONTINUE

C MV IS ORDER OF INDIVIDUAL LIAPOUNOV FUNCTION IN GIVEN FAMILY

5 FORMAT(14)

WRITE(2,104)MV,IND

104 FORMAT(//6X,14,26H ORDER OF LIAPOUNOV FUNCTION,3X:5HIND =114)

CALL ZUBOV(MV,MF,MPHI,IND)

CALL VALUE(MV,M,AVRAD,IPRINT)

7 CONTINUE

NN=NN+1

196.
IF (NN-UFAMIC) .LT. 18, 18
10 STOP 01
END

END OF SEGMENT, LENGTH 245, NAME FAMILIESOFZUR
Appendix A.6. Program for Rodden's Method

This program, written in FORTRAN 4 and designated SUBROUTINE VALUE computes points on the RAS for a particular system and a particular Liapounov function. The call statement is

CALL VALUE(MV,B,AVRAD,IPRINT)

where MV is the degree of the Liapounov function.
B is set at 1.0 normally, when the average radius of the RAS is required. If B is set at -1.0 the negative of the average radius is computed (for the optimisation techniques of chapters 7 and 8 which are minimisation routines).
AVRAD is working space, and contains the average radius.
IPRINT is either set to 1 or 2. If 1, the program outputs a list each line of which is X Y VDOT V where X and Y are the coordinates of points on the boundary of the RAS and VDOT and V are the values of \( \dot{V} \) and \( V \) at these points. If IPRINT is set to 2 this print is suppressed.

Before this list is outputted various other values are outputted. These are of no consequence, being merely for checking purposes.

The program requires the following secondary subroutines

SUBROUTINE EXTANA(X,Y,MV,VX,VY) to compute \( V_x \) and \( V_y \)
SUBROUTINE EXTAND(X,Y,KV,VXX,VXY,VYY) to compute \( V_{xx} \), \( V_{xy} \), and \( V_{yy} \)
SUBROUTINE PXTANE(X,Y, FX, FY) to compute V
These are permanent routines. The user must supply the
particular routines
SUBROUTINE PXTANB(X,Y, F, G) to compute f and g
SUBROUTINE PXTANC(X,Y, FX, FY, GX, GY) to compute f_x, f_y, g_x, g_y.
In the above, subscripts denote partial derivatives, and
f and g are the right hand sides of the system equations thus
\[ \begin{align*}
  x &= f(x, y) \\
  y &= g(x, y)
\end{align*} \]
The coefficients of the Liapounov function are contained in
array A which is stored in COMMON.
We have shown the subroutine PXTANB and PXTANC for the sys-
tem 1.7.4.5.
SUBROUTINE VALUE(MV,B,AVRAD;PRINT)

DIMENSION A(6,7)
COMMON/AREA3/A
EXPF(X) = EXP(X)
ATANF(X) = ATAN(X)
SQRTF(X) = SQRT(X)
ABSF(X) = ABS(X)

2 FORMAT(1H1,9E12.9)
X1=0.000001
Y1=0.000001
STEP=0.0001
GGX=1.0E-40
AVRAD=0.
TOTAL=0.
JJJ=0

444 N=0
X=X1
Y=Y1
CALL PXTANA(X,Y,MV,VX,VY)
CALL PXTANB(X,Y,F,G)
VDOT=VX*F+VY*G
IF(VDOT)445,446,446

446 X1=X1/5,
Y1=Y1/5,
IF(ABSV1=GGX)529,529,444
RAD=SQRTF(X*X+Y*Y)
STAN=SQRTF(STEP**2+STEP*RAD)
GX=-Y/RAD
GY=X/RAD
X0=X
Y0=Y
X=X+GX*STAN
Y=Y+GY*STAN
VDOTO=VDOT
CALL PXTANA(X,Y,MV,VX,VY)
CALL PXTANB(X,Y,F,G)
VDOT=VX*F+VY*G
N=N+1
IF(VDOT)462,462,450
450 IF(N-10)451,451,452
451 XN=N
STEP=STEP*X**2/20.
JJJ=1
GO TO 444
462 IF(JJJ)445,463,445
463 IF(N-50)445,464,464
464 STEP=STEP+5.
GO TO 144

C

452 X2=(X+X0)/2
Y2=(Y+Y0)/2
DIST=SQR((X-X0)**2+(Y-Y0)**2)
RADIUS=SQR((X2**2+Y2**2)
IF(DIST=RADIUS/100.)706,706,703

703 CALL PXTAN1(X2,Y2, MV, VX, VY)
CALL PXTANB(X2,Y2,F,G)
VDOT=VX*F+VY*G
IF( VDOT )702,706,447

702 X0=X2
Y0=Y2
GO TO 452

447 X=X2
Y=Y2
GO TO 452

706 X=X2
Y=Y2

C

701 CALL PXTAN1(X,Y, MV, VX, VY)
CALL PXTANB(X,Y,F,G)
CALL PXTANC(X,Y,FX,FY,GX,GY)
CALL PXTAND (X,Y, MV, VX, VY, VXX, VXY, VYY)
VDOTX=VXX*F+VXX*FX+VXX*GX+VXY*GY+VXX*G
VDOTY=VY*F+VYY*FY+VYY*GY
ROOT=SQR((VX**2+VY**2)
BL=ROOT*VDOTX**2+VDOTY**2)
FAC=(VX*VDOTX*VY*VDOTY)/BL
HX=VX/ROOT=FAC*VDRX
HY=VY/ROOT=FAC*VDOTY
HMID=SQR((X**2+Y**2)/10.
ESS=S/5;
SS=ESS

708 DX=-S*HX/HMDD
D Y=-S*HY/HMDD

720 X=X+DX
Y=Y+DY

CALL PXTANA(X,Y, MV, VX, VY)
CALL PXTANB(X,Y, F,G)
VDOT=VX+F+VY*G
IF( VDOT )704,709,709

704 X8=1;
GO TO 710
799 X8=-1;
710 CALL PXTANA(X,Y,MV, VX, VY)
CALL PXTANB(X,Y,FX, FY)
CALL PXTANC(X,Y,FX, FY, GX, GY)
CALL PXTAND(X,Y,MV, VXX, VXV, VYY)
VDOTX=VXX*FX+VX*FY+VY*G+VY*G
VDOTY=VXY*FX+VXY*FY+VYY*G+VYY*G
ROOT=SQRTF(VDOTX**2+VDOTY**2)

705 VDOTO=VDOT:
X=X+VDOTX*S*S*X/R
Y=Y+VDOTY*S*X/R

713 CALL PXTANA(X,Y,MV, VX, VY)
CALL PXTANB(X,Y,FX, FY, G)
VDOT=VX+F+VY*G
IF(VDOT)*712,709,711

712 XB=1;
IF(SS=ESS/10.)709,709,714
XB=-1,
714 IF(VDOT*VDOTO)*715,705,705
715 SS=SS/3.
GO TO 705

C LOCATION OF VDOT = 0

709 CALL PXTANA(X,Y,MV, VX, VY)
CALL PXTANB(X,Y,FX, FY)
CALL PXTANC(X,Y,FX, FY, GX, GY)
CALL PXTAND(X,Y,MV, VXX, VXV, VYY)
VDOTX=VXX*FX+VX*FY+VY*G+VY*G
VDOTY=VXY*FX+VXY*FY+VYY*G+VYY*G
ROOT=SQRTF(VX**2+VY**2)
FAC=(FX*VDOTX+FY*VDOTY)/(ROOT*(VDOTX**2+VDOTY**2))
HX=FX/ROOT=FAC*VDOTX
HY=FY/ROOT=FAC*VDOTY
MMOD=SQRTF(X**2+Y**2)

721 RADIUS=SQRTF(X**2+Y**2)
IF(S-RADIUS/100.)*707,707,934

934 DX=S*HX/MMOD
DY=S*HY/MMOD
IF(DX*DX+DY*DY)*722,722,729

729 SS=ESS
GO TO 720

722 S=S/3.
ESS=ESS/3.
SS=ESS
GO TO 706

C LOCATION OF TANGENCY

707 AVRAD=0.
RADIV=0.
TOTE=0.
CALL PXTANE(X,Y,MV,UTANG)
WRITE(1,2)VTAN
ST=SQRT(X**2+Y**2)/10.
X=X*.95
Y=Y*.96
CALL PXTANA(X,Y,MV,VX,VY)
CALL PXTAND(X,Y,MJ,VX,VY)
VDOT=VX*F+VY*G
CALL PXTAN(V,X,Y,MV)
TOL=(VTANG-V)/3.
WRITE(*,106)MX,MY,VDOT,V
106 FORMAT(4E12.5)
ESST=ST/5.
SST=ESST
CONST=V
III=I
XMM=Y/X
951 XO0=X
YO0=Y
X0=X
YO=Y
CALL PXTANA(X,Y,MV,VX,VY)
CALL PXTAND(X,Y,MJ,VX,VY)
GROOT=SQRT(VX**2+VY**2)
BL=GROOT**4
TL=-VX*VY**2/V**2.*Y*VY-VY*VX*VY**2
X=X-VY*ST/GROOT**5*TL*VX*ST**2/BL
Y=Y+VY*ST/GROOT**5*TL*VY*ST**2/BL
CALL PXTANE(X,Y,MV,V)
EN=V-CONST
IF(EN)993,522,508
993 IF(EN+TOL)508,508,520
508 EQ=EN
CALL PXTANA(X,Y,MV,VX,VY)
GROOT=SQRT(VX**2+VY**2)
IF(EN)509,509,510
509 X=X+VX*ST/GROOT
Y=Y+VY*ST/GROOT
GO TO 521
510 X=X-VX*ST/GROOT
Y=Y-VY*ST/GROOT
521 CALL PXTANE(X,Y,MV,V)
EN=V-CONST
IF(EN)523,522,520
523 IF(EN+TOL)520,520,522
520 IF(EN+EQ)11,11,508
11 SST=ST/5.
GO TO 508
522 CALL PXTANA(X,Y,MV,VX,VY)
CALL PXTANB(X,Y,F,G)
VDOT=VX*F+VY*G
IF(VDOT)<530,530,452
530 GO TO(100,101),IPRINT
100 WRITE(4,2)X,Y,VDOT,Y
101 SST=ESST
XO=X
YO=Y
ER=100.*(V-CONST)/CONST
TOTER=TOTER+ER
AVRAD=AVRAD+SORTF(X*X+Y*Y)
RADIV=RADIV+1.
XMO=YOO/XOO
IF(ABS(XMO-XM))<0.0001)951,951,950
950 IF((Y-X*X*H)+Y(Y-U*H)+X*W))952,952,951
952 IF(I11)953,953,953
953 111=0
GO TO 951
954 TOTER=TOTER/RADIV
AVRAD=AVRAD/RADIV
529 WRITE(4,320)AVRAD,TOTER
320 FORMAT(17H AVERAGE RADIUS= ,E12,5,3X,16HAVERAGE ERROR = ,E12,5,7HP,1ERCENT)
RETURN
END

END OF SEGMENT, LENGTH 1508, NAME VALUE
C

SURROUNTE PXTANA(X,Y,NV, VX Vy)

DIMENSION A(6,7)

COMMON AREAS

VX=0.

VY=0.

DO 1 I=2, NV

XI=I

DO 2 J=1, I

XJ=J

2 VX=VX+4 A(I,J)*X(I-1)(J)*X(I,J)*X(I-1,J)*X(I,J-1)

II=I+1

DO 1 J=2, II

XJ=J

1 VY=VY+4 A(I,J)*X(I,J-1)*X(I,J)*X(I,J-1)*X(I,J)

RETURN

END OF SEGMENT, LENGTH 148, NAME PXTANA
SUBROUTINE PXTAND(X, Y, NV, VXX, VXY, VYY)
DIMENSION A(6,7)
COMMON/AREA3/A
VXX=0.
VXY=0.
VYY=0.
DO 1 I=2, MV
   XI=I
   II=I+1
   III=I-1
   DO 2 J=1, III
      XJ=J
   2 VXX=VXX+A(I,J)*(XI-XJ+1.)*(XI-XJ)*X**((I-J+1)*Y**((J-1)
      DO 3 J=2, I
         XJ=J
      3 VXY=VXY+A(I,J)*(XI-XJ+1.)*(XJ-1.)*X**((I-J+1)*Y**((J-2)
      DO 1 J=3, I
         XJ=J
      1 VYY=VYY+A(I,J)*(XJ-1.)*(XJ-2.)*X**((I-J+1)*Y**((J-3)
      RETURN
   END
END OF SEGMENT, LENGTH 220, NAME PXTAND
C
SUBROUTINE PXTANE(X,Y,MV,V)
DIMENSION A(6,7)
COMMON/AREA3/A
V=0,
DO I = 1, MV
II=I+1
DO 1 J=1,II
1 V=V+A(I,J)+V**((I-J+1)**(J-1))
RETURN
END

END OF SEGMENT, LENGTH 81, NAME PXTANE
SUBROUTINE EXTNB(X, Y, F, G)
AA=1.0E9
EEP=EXP(3.908*Y)
EP=EXP(-35.925/(Y+1.754))
F=.875*AA*X*EP+.8737*AA*EP+X-.8737*AA*EEP(-35.925/1.754)
G=-.8735*AA*EP+.8743*AA*X*EP+1.18*Y+5.57*Y*EEP+.6853*EEP+.8735*AA*
EEP(-35.925/1.754) -.6853
RETURN
END

SUBROUTINE EXTNB(X, Y, FX, FY, GX, GY)
EEP=EXP(3.908*Y)
EEP=EXP(-35.925/(Y+1.754))
AA=1.0E9
EEPY=3.908*EEP
EPY=35.925*EEP/(Y+1.754) #2
FY=.875*AA*X*EPY+.8737*AA*EPY
GX=-.8743*AA*EP
GY=-.8735*AA*EPY+.8743*AA*X*EPY+1.18*Y+5.57*Y*EEP+5.57*EEP+.6853*EEP
+1
+1
RETURN
END
Appendix A.7. Program for the Optimisation of Zubov's Method

This program, written in FORTRAN 4 and designated SUBROUTINE SIMOPT, uses the modified Simplex method of Nelder and Mead (see Appendix A.13) to optimise the method of Zubov as described in chapter 7. The call statement is

```
CALL SIMOPT(HV,LP,LPHILND,LP,BET,GAM,B,P,Q,X,ITRIAL)
```

where HV LP LPHILND B P and Q are defined in Appendices A 1.5 and A.6.

LP, BET and GAM are the parameters $\lambda, \beta$ and $\gamma$ of the simplex method (see appendix A.13). ITrial is the time in seconds at which the search is to terminate.

The 2-dim. array X holds the initial points of the simplex. Thus $X(3,J)$ holds the coordinates of the 2$^{rd}$ point ($J = 1,2,\ldots, l+1$ where $l$ is the dimension of the space).

The coefficients of the Liapounov function are stored in COMMON in array $\alpha$.

209
SUBROUTINE SIMOPT(MV, MF, MPH, IND, ALP, GAM, B, P, Q, X, ITRIAL)
DIMENSION P(10, 11), Q(10, 11), X(8, 7), R(3, 4), A(10, 11), F(8), XBAR(7)
PS(7), XPSS(7)
COMMON A, CALL ITIME(ISEC1)
WRITE(2, 555) ISEC1
555 FORMAT(8H TIME = , I6, 4HSECS)
IPRINT = 1
N = ((2 * MPH + 3) ** 2 - 25) / 8
NN = N + 1
N = N
DO 404 L = 1, NN
   = 2
   J = 1
   DO 401 K = 1, N
      R(I, J) = X(L, X)
      IF (J - I = 1) 403, 402, 403
  402: J = J + 1
      = 1 + 1
      GG 10, 401
  403: J = J + 1
   DO 401 K = 1, N
   CALL ZUBOV(MV, MF, MPH, P, Q, IND)
   CALL VALUE(MV, B, FF, IPRINT)
   F(L) = FF
   IPRINT = 2
   DO 404 CONTINUE
   404: CONTINUE
   NH = 1
   DO 8 K = 2, NN
       IF (F(NH) = F(K)) 7, 7, 8
    7 NH = K
   8 CONTINUE
   NL = 1
   DO 9 K = 2, NN
       IF (F(NL) = F(K)) 9, 10, 10
    9 NL = K
   10 CONTINUE
   NHD = 1
   IF (NH = 1) 15, 14, 15
   14 NHD = 2
   DO 11 K = 2, NN
       IF (K - NH) 12, 11, 12
    12 IF (F(NH) = F(K)) 13, 13, 11
    13 NHD = K
   11 CONTINUE
   WRITE(2, 150) NL, NHD, NH, F(NL), F(NH)
150 FORMAT(/1H, 3I3, 3E12.5)
   CALL ITIME(ISEC2)
ISEC2 = ISEC1
WRITE (2, 556) IONE
556 FORMAT (16H TIME ELAPSED = ,16,4HSECS)
IF (IGONE - ITRIAL) 167, 168, 168
C THIS SECTION FOR FINAL PRINT AFTER TIME LIMIT
168 J = 2
J = 1
IPRINT = 1
DO 172 K = 1, N
R(I, J) = X(NL, K)
171 J = 1
J = I + 1
60 YO. 172
170 J = J + 1
172 CONTINUE
DO 160 K = 2, MPH
KK = K + 1
WRITE (2, 159) (R(K, J), J = 1, KK)
159 FORMAT (13H PHI=FUNCTION, //,1H , 6E12.5)
160 CONTINUE
CALL ZUBOV (MV, MF, MPH, P, Q, R, IND)
CALL VALUE (MV, B, AVRAD, IPRINT)
RETURN
C 167 DO 3 K = 1, N
3 XPBAR(K) = 0.
DO 5 J = 1, N
DO 4 I = 1, NN
IF (I - NH) 6, 4, 6
4 XPBAR(J) = XPBAR(J) + X(I, J)
4 CONTINUE
XPBAR(J) = XPBAR(J) / XN
5 CONTINUE
WRITE (2, 151) (XPBAR(J), J = 1, N)
151 FORMAT (1H , 6E12.5)
DO 20 K = 1, N
20 YPS(K) = (1. + ALP) * XPBAR(K) - ALP * X(NH, K)
I = 2
J = 1
DO 501 K = 1, N
R(I, J) = XPS(K)
1F (J - I - 1) 503, 502, 503
502 J = 1
J = I + 1
60 IN: 501
503 J = J + 1
501 CONTINUE
RETURN
C
IF(R(2,1)*R(2,3),LE.R(2,2)**2/4,GO TO 610
CALL ZUBOV(MV, MF, MPHI, P, Q, R, IND)
CALL VALUE(MV,B,FS,IPRINT)
GO TO 612
610 WRITE(2,611)
611 FORMAT(24H CONSTRAINT TRANSGRESSED)
FS=(R(2,1)*R(2,3)-R(2,2)**2/4,)*B
612 WRITE(2,151)(XPS(J),J=1,N)
WRITE(2,152)FS
152 FORMAT(4H FS ,E12.5)
IF(FS-F(NL))21,21,22
21 DO 24 K=1,N
24 XPS(K)=(1.+GAM)*XPS(K)-GAM*XBAR(K)
I=2
J=1
DO 601 K=1,N
R(I,J)*XPS(K)
601 I=I+1
GO TO 601
602 J=J+1
CONTINUE
IF(R(2,1)*R(2,3),LE.R(2,2)**2/4,GO TO 710
CALL ZUBOV(MV, MF, MPHI, P, Q, R, IND)
CALL VALUE(MV,B,FS,IPRINT)
GO TO 712
710 WRITE(2,611)
711 WRITE(R(2,1)*R(2,3)-R(2,2)**2/4,)*B
712 WRITE(2,151)(XPS(J),J=1,N)
WRITE(2,153)FS
153 FORMAT(5H FS ,E12.5)
IF(FS-F(NL))23,23,29
23 DO 26 K=1,N
26 X(NH,J)=XPS(K)
F(NH)=FS
GO TO 33
22 IF(FS-F(NHD))25,25,27
25 DO 28 K=1,N
28 X(NH,J)=XPS(K)
F(NH)=FS
GO TO 33
27 IF(FS-F(NH))29,29,30
29 DO 101 K=1,N
101 X(NH,K)=XPS(K)
F(NH)=FS
30 DO 32 K=1,N
32 XPS(K)=(1.-BET)*XBAR(K)+BET*X(NH,K)
I=2
J=1
DO 701 K=1,N
R(I,J)=XPSS(K)
IF (I-1=1) 703, 702, 703
702 J=I+1
I=I+1
GO TO 701. 
703 J=J+1
701 CONTINUE:
IF (R(2,1)+R(2,3), LE, R(2,2)**2/4.) GO TO 810
CALL ZUBOV(VW,MF,MPHI,P,Q,R,IND)
CALL VALUE(VW,B,FSS,IPRINT)
GO TO 812.
810 WRITE(6,151) FSS
99 CONTINUE
WRITE(2,154) FSS
154 FORMAT(6H,FSS2,E12.5)
IF (FSS=F(NH)) 34,35,38
34 DO 36 K=1,N
36 X(NH,K)=XPSS(K)
 Finn=KSS
GO TO 33
35 DO 37 J=1,N
37 X(J,J)=(X(J,J)+X(NL,J))/2.
DO, 380 J=1,N
380 WRITE(6,151) X(J,K), K=1,N
38 FORMAT(6H,X(J,K),E12.5)
DO 800 L=1,N
800 WRITE(6,151) X(I,L), I=1,N
801 WRITE(6,151) R(I,J), J=1,N
99 CONTINUE
IF (J-1=1) 203, 202, 203
202 J=1
201 CONTINUE:
CALL ZUBOV(VW,MF,MPHI,P,Q,R,IND)
CALL VALUE(VW,B,FF,IPRINT)
P(I)=FF.
200 CONTINUE
33 CONTINUE
GO TO 40
55 RETURN
END
Appendix A.8. Program for Sequential Optimisation of Zubov's Method

This program, written in FORTRAN 4 and designated SUBROUTINE CYRIL applies the method of section 7.5 to the construction of optimum Liapounov functions.

The call statement is

CALL CYRIL(MV,MF,IND,ΛP,BET,B,MPHIMX)

where all arguments except the last are defined in appendix A.7.

MPHIMX is the degree of the Φ-function.

The main program must also supply other variables which are stored in COMMON.

ITRIAL is a 1-dim. array holding the times in seconds at which each stage of the optimisation is to terminate. Thus if the optimisation is carried out on 3rd degree Φ-functions MPHIMX is 3; the optimisation of the quadratic part of will terminate after ITRIAL(1) secs. and the optimisation of the 3rd degree part after ITRIAL(2) secs.

XX is a three-dimensional array containing the points of the simplex. The first subscript denotes the degree of the part of Φ being optimised; the second the point on the simplex; the third the coordinate of the point. Thus XX(3,2,1) holds the first coordinate of the second point on the simplex for the 3rd degree part of Φ.

P and Q are defined in appendix A.5.
26 X(NH,K)=XPSS(K)   F(NH)=FSS
   GO TO 33
29 IF(FS-F(NHD))25,25,27
28 X(NH,K)=XPSS(K)   F(NH)=FSS
   GO TO 33
27 IF(FS-F(NHD))29,29,30
25 DO 28 K=1,MMORE
26 X(NH,K)=XPSS(K)   F(NH)=FSS
   GO TO 33
24 IF(FS-F(NHD))34,34,35
23 DO 36 K=1,MMORE
22 X(NH,K)=XPSS(K)   F(NH)=FSS
   GO TO 33
21 DO 37 J=1,MMMORE
20 DO 37 I=1,MMORE
29 X(I,J)=X(I,J)+X(NL,I))/2.
10 DO 820 J=1,MMMORE
70 WRITE(2,821)(X(I,J),I=1,MMMORE)
80 F FORMAT(6H FS2 ,E12.5)
60 WRITE(2,154)FSS
50 IF(FSS=F(NH))34,34,35
40 DO 36 K=1,MMORE
39 X(NH,K)=XPSS(K)   F(NH)=FSS
   GO TO 33
38 DO 37 J=1,MMMORE
37 DO 37 I=1,MMORE
36 X(I,J)=X(I,J)+X(NL,I))/2.
35 DO 820 J=1,MMMORE
34 DO 820 I=1,MMMORE
33 CONTINUE
70 WRITE(2,519)X(NH,K)
60 IF(NH=1)70,70,71
50 IF(NS>2)70,70,71
40 READ(5,15)X(NH,K)
30 IF(NH=1)70,70,71
20 IF(NH=1)70,70,71
10 IF(NH=1)70,70,71
00 STOP

END OF SEGMENT, LENGTH 1258, NAME CYRIL
Appendix A.9. Program for Weissenberger's Method

This program, written in FORTRAN 4 and designated SUBROUTINE SIMOPT, applies Weissenberger's method to the construction of optimum Liapounov functions. The call statement is

CALL SIMOPT(N,X,B,ALP,BET,GIL,F11,F12,G11,G12)

where N is the dimension of the parameter space and is given by \((mv+1)(mv+2)/2 - 3\) where \(mv\) is the degree of the Liapounov function.

X holds the points of the simplex so that \(X(I,J)\) holds the \(J\)th coordinate of the \(I\)th point.

F11,F12,G11 and G12 are the coefficients of the linear parts of the system thus

\[
\begin{align*}
  x &= F11x + F12y \\
  y &= G11x + G12y
\end{align*}
\]

The other arguments are defined in Appendix A.7.
SUBROUTINE SIMOPT(N,X,B,ALP,BET,GAM,F11,F12,G11,G12,ITRIAL)
DIMENSION X(21,20),A(10,11),F(21),XBAR(20),XPS(20),XPSS(20)
CALL ITIME(ISEC1)
IPRINT=1
KK=0
KKK=0
NN=N+1
X=N
C PUTTING POINT COORDINATES IN LIAPOUNOV ARRAY AND OBTAINING VALUE
DO 404 L=1,NN
I=2
J=1
DO 401 K=1,N
A(I,J)=X(L,K)
402 J=1
I=I+1
GO TO 401
403 J=J+1
401 CONTINUE
XMV=(-3.*SORT(NX+P5.))/2.
XMV=XMV+1.
MV=XMV
R1=2.*A(2,1)+F11+A(2,2)*G11
R2=A(2,2)+F11+2.*A(2,1)*F12+A(2,2)*G12+2.*A(2,3)*G12
R3=A(2,2)+F12+2.*A(2,3)*G12
TEST=R3-R2**2/4.
IF (TEST) GT 710,710,711
710 F(L)=-TEST
GO TO 404
711 CALL VALUE(MV,A,B,FF,IPRINT)
F(L)=-FF
IPRINT=2
404 CONTINUE
40 NH=1
DO A K=2,NN
IF(F(NH)-F(K))7,7,8
7 NH=K
8 CONTINUE
NL=1
DO 9 K=2,NN
IF(F(NL)-F(K))9,10,10
9 NL=K
10 CONTINUE
NHD=1
IF(NH-1)15,14,15
14 NHD=2
15 DO 11 K=2,NN
IF(K-NH)12,11,12
12 IF(F(NHO)-F(K))13,13,11
13 NH=K
11 CONTINUE
WRITE(2,150)NL,NHD,NH,F(NL),F(NHO),F(NH)
150 FORMAT(3i3,3E12.5)
CALL ITIME(ISFC2)
IGONE=ISEC2-ISEC1
IF(IGONE=ITRIAL)167,167,168
C THIS SECTION FOR FINAL PRINT AFTER TIME LIMIT
168 I=2
J=1
DO 172 K=1,N
A(I,J)=X(NL,K)
IF(J-I-1)170,171,170
171 J=1
I=I+1
GO TO 172
170 J=J+1
172 CONTINUE
IPRINT=1
CALL VALUE(WV,A,B,AVRAD,IPRINT)
RETURN
C
167 DO 3 K=1,N
XPBAR(K)=0.
3 CONTINUE
DO 5 J=1,N
DO 4 I=1,NN
IF(I-NH)6,4,6
6 XPBAR(J)=XPBAR(J)+X(I,J)
4 CONTINUE
XPBAR(J)=XPBAR(J)/XN
5 CONTINUE
WRITE(2,151)(XPBAR(J),J=1,N)
151 FORMAT(1H,3E17.10)
DO 20 K=1,N
XPS(K)=(1.+ALP)*XPBAR(K)-ALP*X(NH,K)
20 CONTINUE
C PUTTING POINT COORDINATES IN LIAPOUNOV ARRAY AND OBTAINING VALUE
I=2
J=1
DO 501 K=1,N
A(I,J)=XPS(K)
IF(J-I-1)503,502,503
502 J=1
I=I+1
GO TO 501
503 J=J+1
501 CONTINUE
XMV=(-3.*SORT(8.*XN+25.))/2.
XMV=XMV*1
MV=XMV
RI=2.*A(2,1)*F11+A(2,2)*G11
R2=A(2,2)*F11+2.*A(2,1)*F12+A(2,2)*G12+2.*A(2,3)*G11
R3=A(2,2)*F12+2.*A(2,3)*G12
TEST=P1*R3-R2**2/4.
IF(TEST)720,720,721
720 FS=-TEST
GO TO 722
721 CALL VALUE(MV,A,B,FS,IPRINT)
722 WRITE(2,151)(XPS(J),J=1,N)
WRITE(2,152)FS
152 FORMAT(4H FS ,E17.10)
IF(FS=F(NL))21,21,22
21 DO 24 K=1,N
XPS(K)=(1.*GAM)*XPS(K)-GAM*XPBAR(K)
24 CONTINUE
C PUTTING POINT COORDINATES IN LIAPOUNOV ARRAY AND OBTAINING VALUE
I=2
J=1
DO 601 K=1,N
A(I,J)=XPS(K)
IF(J-1-1)603,602,603
602 J=1+1
GO TO 601
603 J=J+1
601 CONTINUE
XMV=(-3.*SORT(8.*XN+25.))/2.
XMV=XMV*1
MV=XMV
RI=2.*A(2,1)*F11+A(2,2)*G11
R2=A(2,2)*F11+2.*A(2,1)*F12+A(2,2)*G12+2.*A(2,3)*G11
R3=A(2,2)*F12+2.*A(2,3)*G12
TEST=P1*R3-R2**2/4.
IF(TEST)730,730,731
730 FS=-TEST
GO TO 732
731 CALL VALUE(MV,A,B,FS,IPRINT)
732 WRITE(2,151)(XPS(J),J=1,N)
WRITE(2,152)FS
153 FORMAT(5H FS ,E17.10)
IF(FS=F(NL))23,23,25
23 DO 26 K=1,N
X(NH,K)=XPS(K)
26 CONTINUE
F(NH)=FSS
GO TO 33
22 IF(FS-F(NHO))<25,25,27
25 DO 28 K=1,N
X(NH,K)=XPS(K)
28 CONTINUE
F(NH)=FS
GO TO 33
27 IF(FS-F(NH))<29,29,30
29 DO 101 K=1,N
X(NH,K)=XPS(K)
101 CONTINUE
F(NH)=FS
30 DO 32 K=1,N
XPSS(K)=(1.-BET)*XPBAR(K)+BFT*X(NH,K)
32 CONTINUE
C PUTTING POINT COORDINATES IN LAPOUNOV ARRAY AND OBTAINING VALUE
I=2
J=1
DO 701 K=1,N
A(I,J)=XPSS(K)
IF(J=1-1)703,702,703
702 J=J+1
I=I+1
GO TO 701
703 J=J+1
GO TO 701
701 CONTINUE
XMV=(-3.+SQRT(8.*XN+25,1)))/2.
XMV=XMV+.1
MV=XMV
R1=2.*A(2,1)*F11+A(2,1)*G11
R2=A(2,2)*F11+2.*A(2,1)*F12+A(2,2)*G12+2.*A(2,3)*G12
R3=A(2,2)*F12+2.*A(2,3)*G12
TEST=R1+R2+R3=2**2/4.
IF(TEST)740,740,741
740 FSS=-TEST
GO TO 742
741 CALL VALUE(MV,A,B,FSS,PRINT)
742 WRITE(2,151)(XPSS(J),J=1,N)
WRITE(2,154)FSS
154 FORMAT(6H FSS2 ,E17.10)
IF(FSS-F(NH))<35,35,35
34 DO 36 K=1,N
X(NH,K)=XPSS(K)
36 CONTINUE
F(NH)=FSS
GO TO 33
35 DO 37 J = 1, NN
36 DO 37 I = 1, N
37 CONTINUE
38 X(J, I) = (X(J, I) + X(NL, I)) / 2.
39 CONTINUE
33 WRITE(2, 48) (X(NH, K), K = 1, N)
40 FORMAT(5E12.5)
41 WRITE(2, 49) (NH)
42 FORMAT(E17.10)
43 GO TO 40
44 RETURN
45 END

END OF SEGMENT, LENGTH 1401, NAME SIMOPT
Appendix A.10. Program for the Tracking Function Method in the rθ plane

This program written in FORTRAN 4 computes the extreme bounding path according to the tracking function policy in the rθ plane (see section 9.7). The data required is

RADMIN
NPOINT NDIV NANG
XXL YYL
S SS

where RADMIN is the radius of the circular N_1 region (in the xy plane). NPOINT is the number of points on the extreme path before termination. NDIV is the number of steps between printouts; thus if NDIV is 10 every tenth point will be printed out. NANG is the number of bisections. XXL and YYL are the coordinates θ and r resp. of the starting point. S and SS are resp. the step size for plotting and the step size for finding the allowable sector. It is reasonable to let S = SS.

The program requires one subroutine, namely:

SUBROUTINE FDG(XXL,YYL,F,G) where F and G are the right hand sides of the system equations thus

\[ \dot{r} = f(r, \theta) \]
\[ \dot{\theta} = g(r, \theta) \]
We also show the program TFNONAUT which applies the tracking function method in the xy plane to the linear non-stationary systems described in section 9.10. The data is NPOINT NANG 3 XX1 YY1 where the first three of these arguments are defined above. XX1 and YY1 are the coordinates x and y of the initial point.

The program requires one subroutine

```
SUBROUTINE F1G(N,X,Y,F,G)
```

where N is working space. F and G are the right hand sides of the system equations. This subroutine starts with the statement GO TO (1,2,3,4)N. Each of the statements labelled 1,2,3 and 4 computes the right hand sides F and G for one of the 4 possibilities described in section 9.10.

This is better appreciated by the example given. This is the subroutine F1G for the case $m_1 = 2$ $m_2 = 6$ $n_1 = 1.03$ $n_2 = 10^{-40}$ (approximating $\infty$).

This program may also be used for the autonomous case merely by making the statements 1,2,3 and 4 compute the same functions, namely the right hand sides of the system equations.
LIS:(LP)
PROGRAM(M101)
TRACE
INPUT1=CRO
OUTPUT4,(MONITOR)=LPO
END
 Mas/ser tracking function radius
NN=0
NSTEP=0
READ(1,110) ADMIN
110 FORMAT(E12.5)
READ (1,50) NPOINT,NDIV,NANG
50 FORMAT(314).
READ (1,1) XX1,YY1
1 FORMAT(2F12.5)
READ(1,35)S,SS
33. FORMAT(2E12.5)
ULT=SS/S
21. CALL FAG(XX1,YY1,F,G)
N=0
X2=0.
Y3=S
IF(0)2,2,3
2 X3=S
GO TO 4
3 X3=S
4 IF(0)5,5,6
5 Y2=S
GO TO 7
6 Y2=S
7 GRAD=(Y2+Y3)/(X2*X3)
X4=S/SQRT(1.+GRAD**2)
Y4=GRAD*X4
IF(X2)8,9,8
9 IF(X4*F)10,10,11
10 X4=X4
Y4=Y4
GO TO 11
8 E2X4=Y4-Y2*X4/X2
E2DOT=G-Y2*F/X2
IF(E2X4+E2DOT)12,12,11
12 X4=X4
Y4=Y4
11 E4X2=Y2-Y4*X2/X4
E4DOT=G-Y4*F/X4
IF(E4X2+E4DOT)13,13,14
13 X2=X4
Y2=Y4
GO TO 15
14 X3=X4
Y3=Y4
15 N=N+1
IF(N=NANG)7,16,16
16 XX2=XX1+X2*ULT
YY2=YY1+Y2*ULT
XX3=XX1+X3*ULT
YY3=YY1+Y3*ULT

226.
IF (YY2-YY3) 17,17,18
17 XX4=XX3
   YY4=YY3
   GO TO 19
18 XX4=XX2
   YY4=YY2
19 IF (YY1=RAH)900,900,901
   XX1=XX4
   YY1=YY4
   GO TO 106
900 IF (F)904,904,902
902 XX5=XX1+SS
   GO TO 903
904 XX5=XX1-SS
903 YY5=YY1
   IF (YY4=YY5) 812,812,901
812 XX1=XX5
   YY1=YY5
106 NSTEP=NSTEP+1
   IF (XX1=-6.2831853) 817,816,818
816 XX1=XX1-6.2831853
   GO TO 821
817 IF (XX1+6.2831853) 820,821,821
820 XX1=XX1+6.2831853
821 X=YY1*COS(XX1)
   Y=YY1*SIN(XX1)
   IF (NSTEP-NDIV) 21,31,31
31 WRITE (4,52)x,Y,XX1,YY1
52 FORMAT (1H,4E12.5)
   NN=NN+1
   IF (NN-NPOINT) 501,501,502
501 NSTEP=0
   GO TO 21
502 STOP 01
END
MASTER TFNOVAUT
DIMENSION XX4(4), YY4(4), RR(4)

NSTEP=0
READ(1, 110) VPOINT, NANG, S, XX1, YY1
110 FORMAT(214, 3E12.5)
21 DO 100 M=1, 4
   CALL FAG(M, XX1, YY1, F, G)
   N=0
   X2=0
   Y3=0
   IF(F)2, 2, 3
      2 X3=5
      GO TO 4
   3 X3=5
   4 IF(G)5, 5, 6
      5 Y2=5
      GO TO 7
   6 Y2=5
   7 GRAD=(Y2+Y3)/(X2*X3)
      X4=S/SQRT(1. + GRAD**2)
      Y4=GRAD*X4
   8 GO TO 11
   9 IF(X2)8, 9, 10
   10 X4=F
      Y4=-X4
   11 GO TO 13
   12 X4=Y4
   13 X2=X4
      Y2=Y4
   14 GO TO 15
   15 N=N+1
   IF(N-NANG)7, 16, 16
   16 XX2=XX1+X2
      YY2=YY1+Y2
      XX3=XX1+X3
      YY3=YY1+Y3
      RR2=SQR((XX2**2+YY2**2)
      RR3=SQR((XX3**2+YY3**2)
      IF(RR2-RR3)17, 17, 16
   17 X3=X4
      Y3=Y4

228
SUBROUTINE FAG(X, Y, F, G)
F = Y
GO TO (1, 2, 3, 4) N
1 P = 1.03
   Q = 2
   GO TO 5
2 P = 1.03
   Q = 6
   GO TO 5
3 P = 1.0E+0
   Q = 2
   GO TO 5
4 P = 1.0E+0
   Q = 6
   G = -P*Y - Q*X
   RETURN
END

END OF SEGMENT, LENGTH 84, NAME FAG

This program, written in FORTRAN 4, gives the rate of change of radius as a function of the radius according to the method of Luus and Lapidus. The data to be supplied is

- HF
- P
- Q
- RAD1
- RAD2
- STEP

where HF is the degree of the system equations. P and Q are 2-dim. arrays containing the coefficients $p_{ij}$ and $q_{ij}$ of the system equations (see equations 2.2.1). RAD1 and RAD2 are the least and greatest radii of interest, while STEP is the amount by which the radius $r$ is increased from RAD1 to RAD2.

The output is a list of values of $r$ and $\frac{dr}{dt}$ as given, for example in fig. 10.4.

Also given is the program designated SUBROUTINE SINCOS which computes the value of the integral $I(m, q)$ as defined in section 10.5.
LIST(LP)
PROGRAM(M127)
INPUT1=CR0
OUTPUT2=LOP
END
MASTERLULAP
DIMENSION P(40,41),Q(40,41),R(40)
PI=3.14159
READ(1,1)MF
WRITE(2,1)M=
1 FORMAT(14)
   DO 2 K=1,MF
      KK=K+1
      READ(1,3)(P(K,J),J=1,KK)
      3 FORMAT(1H1,4E17.10)
   2. CONTINUE
      DO 4 K=1,MF
         KK=K+1
      READ(1,3)(Q(K,J),J=1,KK)
   4 CONTINUE
      READ(1,5)RAD1,RAD2,STEP
   5 FORMAT(3E12.5)
      DO 10 I=1,MF
         R(I)=0.
      II=I+1
      DO 10 J=I,II
         ISIN=J
         ICOS=J+2
         CALL SINCOS(ISIN,ICOS,PI,XINT)
         R(I)=R(I)+XINT*P(I,J)/(2.*PI)
      ISIN=J
      ICOS=J+1
      CALL SINCOS(ISIN,ICOS,PI,XINT)
      10 R(I)=R(I)+XINT*Q(I,J)/(2.*PI)
      RAD=RAD1
      WRITE(2,30)
   30 FORMAT(4X,9HRA0US(R),7X,5HDR/DT)
   20 DROT=0.
      DO 15 K=1,MF
         DROT=DROT+R(K)*RAD**K
      WRITE(2,16)RAD,DROT
   16 FORMAT(2X,E12.5,3X,E12.5)
      RAD=RAD+STEP
      IF(RAD>RAD2)20,20,19
   19 STOP 01
END
SUBROUTINE SIN COS (N, N, PI, XINT)

MN = N - 1
XN = N
MM = M - 1
XM = M
MN = N + N - 1
TOP = 1.
TOP2 = 1.
BOT = 1.

1 IF (M) 1, 2, 3
2 IF (N) 4, 5, 6
5 XINT = 2. * PI
RETURN

1 WRITE (2, 7)
7 FORMAT (24H ERROR POWER OF SINE = VE)
STOP 01

3 MH = M / 2
MW = MH + MH
IF (M = MW) 29, 8, 29
29 IF (N) 4, 9, 9
8 IF (N) 4, 11, 12

12 NH = N / 2
NW = NH + NH
IF (N = NW) 9, 13, 9
9 XINT = 0.
RETURN

6 NH = N / 2
NW = NH + NH
IF (N = NW) 9, 14, 9
14 DO 15 K = 1, NN, 2
XK = K
TOP = TOP * (XN - XK)
15 BOT = BOT * (XN - XK + 1.)
XINT = TOP * 2. * PI / BOT
RETURN

4 WRITE (2, 17)
17 FORMAT (23H ERROR POWER OF COS = VE)
STOP 01

11 DO 18 K = 1, MW, 2
XK = K
TOP = TOP * (XM - XK)
18 BOT = BOT * (XM - XK + 1.)
XINT = TOP * 2. * PI / BOT
RETURN

13 DO 20 K = 1, NN, 2
XK = K
20 TOP = TOP * (XN - XK)
DO 21 K = 1, MN, 2
XK = K
21 TOP2 = TOP2 * (XM - XK)
DO 22 K = 1, MN, 2
XK = K
22 BOT = BOT * (XM + XN - XK + 1.)
XINT = TOP * TOP2 * 2. * PI / BOT
RETURN
END
FINISH
Appendix A.13. Program for the A.E.R.P. method

This program, written in FORTRAN 4, computes the A.E.R.P. according to the policy described in chapter 11. The data to be supplied is

XIND NP0INT
XM X1 Y1 SDIV

where XIND indicates the direction the trajectories take around the origin. If the trajectories are clockwise XIND is set to 1 and if anticlockwise, to -1 (the direction of the trajectories may be inferred from the linear parts of the system). NP0INT is the number of points before termination. \( X_0^2 \) is the value of \( \frac{b}{c} \) (see section ) and under the assumption that the transformation of section has been applied, takes the value 1. X1 and Y1 are the coordinates of the initial point. SDIV is the number by which the radius is divided to give the step size. Thus if SDIV is 100, the step size is .01 times the radius.

The program requires SUBROUTINE FAG(X1,Y1,F,G) to compute the right hand sides of the system equations.

We have shown subroutine FAG for the system 1.7.4.
MASTER AERP

ALTERNATING EXTREME RADIUS PATH METHOD

DIMENSION X(2), Y(2), R(2)

N=0

READ(1,1)XIND, NPOINT

1 FORMAT(E12.5,14)

READ(1,2)XM, X1, Y1, SDIV

2 FORMAT(4E12.5)

RAD=SQRT(X1**2+Y1**2)

18 S=RAD/SDIV

CALL FAG(X1, Y1, F, G)

SGNF=F/ABS(F)

SGNG=G/ABS(G)

X(1)=X1*S*SGNF

Y(1)=Y1

Y(2)=Y1+S*SGNG

R(1)=SQRT(X(1)**2+Y(1)**2)

R(2)=SQRT(X(2)**2+Y(2)**2)

ROOT=X1+F+XM*Y1*G

IF(ROOT*(R(1)-RAD))3,4,4

3 R(1)=R(2)

X(1)=X(2)

Y(1)=Y(2)

GO TO 5

4 IF(ROOT*(R(2)-RAD))5,6,6

5 A=1.-S**2/(2.*RAD**2)

B=SQRT(4.*S**2*RAD**2-2.*S**4)/(2.*RAD**2)

IF(ABS(X1)-ABS(Y1))9,9,10

9 X(2)=X1*X1+XIND*Y1*B

Y(2)=(2.*RAD**2-2.*X(2)*X1-S**2)/(2.*Y1)

GO TO 11

10 Y(2)=A*Y1-XIND*X1*B

X(2)=(2.*RAD**2-2.*Y(2)*Y1-S**2)/(2.*X1)

11 R(2)=SQRT(X(2)**2+Y(2)**2)

6 IF(R(1)=R(2))14,14,15

14 AA=X(1)

BB=Y(1)

CC=R(1)

X(1)=X(2)

Y(1)=Y(2)

R(1)=R(2)

X(2)=AA

Y(2)=BB

R(2)=CC

15 IF(RDOT)20,20,21

20 X1=X(1)

Y1=Y(1)
RAD=R(1)
GO TO 16
21 X1=X(2)
   Y1=Y(2)
   RAD=R(2)
16 WRITE(2,17)X1,Y1
   N=N+1
17 FORMAT(1H,E12.5,3X,E12.5)
   IF(N=NPOINT)18,19,19
19 STOP 01
END

END OF SEGMENT, LENGTH 548, NAME AERP
SUBROUTINE FAG(XXI,YYI,F,G)

EP = EXP(-35.929/(YYI+1.754))
EEP = EXP(3.908*YYI)

AA = 10.**9

F = (-.875*AA*XXI*EEP-.8737*AA*EEP*XXI+1.1195)/3.416014
G = .8735*AA*EEP+.8743*AA*XXI*EEP-1.18*YYI-5.57*YYI*EEP-.6885*EEP-.433

RETURN
END

END OF SEGMENT, LENGTH 81, NAME FAG
Appendix A.13. The modified Simplex Method

The method searches for the minimum value of a function of \( n \) variables. The values of the function at the \( (n+1) \) points of a simplex are computed and compared. The vertex with the highest function value is replaced by another point. The actual policy adopted for this replacement may be found in ref. 35. It suffices here to define the nomenclature of the flow diagram of fig. A.14.1, from which the logic involved may be inferred.

The subscripts \( h, l, \) and \( h' \) refer respectively to the vertices with the highest, lowest and next to highest function values. Other subscripts refer to the vertices by number. Thus, \( P_i \) is the \( i \)th vertex, while \( P_h \) is the vertex with the highest function value. The function values are denoted by \( f \) with the appropriate subscript. \( P^* \) is the centroid of the vertices neglecting \( P_h \). \( P^* \) is a new point obtained by reflecting \( P_h \) in \( F \) so that

\[
P^* = (1 + \alpha)F - P_i
\]

\( P^{**} \) is a new point obtained by extending the line from \( F \) to \( P^* \) and is given by

\[
P^{**} = (1 + \gamma)P - F
\]

if \( P^* \) is a point with a new minimum i.e. a better point, or

\[
P^{**} = P_h + (1 - \sqrt{3})F
\]

if \( P^* \) is a worse point.
The actual values used in the program are

\[ \alpha = 1 \]
\[ \beta = .5 \]
\[ \gamma = 1 \]

which correspond to direct reflection, retracting to half way if the new point is worse, and stepping twice the distance if the new point is better.
Fig. 2.1. Kroossovski's method

System 1.7.2. \( C = \begin{bmatrix} 10 & 1 \\ 1 & 1 \end{bmatrix} \)

\( \Rightarrow B = \begin{bmatrix} 10 & 10 \\ 10 & 11 \end{bmatrix} \)

\[
\begin{array}{|c|c|}
\hline
\text{degree} & \text{Av. radius} \\
\hline
2 & 0.913 \\
4 & 0.321 \\
6 & 0.357 \\
\hline
\end{array}
\]

Fig. 2.1. Kroossovski's method

System 1.7.2. \( C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \)

\( \Rightarrow B = \begin{bmatrix} 3 & 1 \\ 1 & 2 \end{bmatrix} \)
Fig. 2.3. Krassovski's method

System 1.7.3. \[ C = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \]

\[ \Rightarrow B = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \]

Fig. 2.4. Krassovski's method

System 1.7.3. \[ C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \]

\[ \Rightarrow B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \]
<table>
<thead>
<tr>
<th>degree</th>
<th>Av. radius</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1.34</td>
</tr>
<tr>
<td>6</td>
<td>0.672</td>
</tr>
</tbody>
</table>

**Fig 2.5. Krassovski's Method**

System $1.71 A (\varepsilon = 0.1)$

$$C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\Rightarrow B = \begin{bmatrix} 20.1 & -1 \\ -1 & 20 \end{bmatrix}$$

Note actual RAS is approximately circular of radius 2.
Fig. 3.1(a).

Flow Diagram for Szego's Method [Pt.1]

System in general form
Eig. 3.1 (b).
Flow Diagram for Szego’s Method. [Pt. 2]
System in general form.
\[ \text{ENTER} \]
\[ a_i, b_i \rightarrow 0, \text{all } i \]
\[ \text{READ } N, p_i, q_i \]
\[ b_i = \frac{-2q_i}{i}, \text{all } i \]

\[ i = 1 \]
\[ j = 1 \]

\[ \text{SGN}[i+j+1-N] \]
\[ a_i = a_i - b_i q_i j+1 \]

\[ j = j+1 \]
\[ j : i < \]

\[ \text{STOP} \]

Flow Diagram for
Szegö's Method System in companion form.
Fig. 3.3.

Szegö's Method

System 1.7.1.A. (ε = 1.0)
Fig 4.1.

Ingwerson's Method

\[ C = \begin{bmatrix} 10 & 1 \\ 1 & 10 \end{bmatrix} \]

System 1.7.3.

<table>
<thead>
<tr>
<th>Degree</th>
<th>Av. radius</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1.22</td>
</tr>
<tr>
<td>4</td>
<td>0.95</td>
</tr>
</tbody>
</table>
Fig. 4.2.
Ingwerson's method.
\[ C = \begin{bmatrix} 2 & 1 \\ 1 & 20 \end{bmatrix} \]
System 17.3.

<table>
<thead>
<tr>
<th>Degree</th>
<th>Av. radius</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1.4</td>
</tr>
<tr>
<td>4</td>
<td>1.0</td>
</tr>
</tbody>
</table>
Fig. 4.3.
Ingleson's method
\[ C = \begin{bmatrix} 10 & 0 \\ 0 & 1 \end{bmatrix} \]
System 1.7.3.
Fig 4.4.
Ingwerson's Method

\[ C = \begin{bmatrix} 1 & 0 \\ 0 & 10 \end{bmatrix} \]

System 17.3.

<table>
<thead>
<tr>
<th>Degree</th>
<th>\text{Av. radius}</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1.255</td>
</tr>
<tr>
<td>4</td>
<td>1.298</td>
</tr>
</tbody>
</table>
Fig. 4.5. 
Ingraham's method

\[
C = \begin{bmatrix} 10 & 1 \\ 1 & 1 \end{bmatrix}
\]

System: 1.7.4.A.

<table>
<thead>
<tr>
<th>Degree</th>
<th>Radius (m)</th>
<th>Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>$633 \times 10^{-3}$</td>
<td>400</td>
</tr>
<tr>
<td>4</td>
<td>$186 \times 10^{-3}$</td>
<td>150</td>
</tr>
<tr>
<td>≥ 6</td>
<td>$178 \times 10^{-3}$</td>
<td>200</td>
</tr>
</tbody>
</table>
Fig 4.6.

Inwerson's method

$$C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

System 1.7.4. A.

<table>
<thead>
<tr>
<th>Degree</th>
<th>Av. radius</th>
<th>Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>$6.1 \times 10^2$</td>
<td>300</td>
</tr>
<tr>
<td>$\geq 4$</td>
<td>$9.4 \times 10^2$</td>
<td>1000</td>
</tr>
</tbody>
</table>
Fig. 4.7.

Inglawson's method

\[ C = \begin{bmatrix} 10 & 0 \\ 0 & 1 \end{bmatrix} \]

System 1.7.4 A.

<table>
<thead>
<tr>
<th>Degree</th>
<th>Avg. Radius (sec)</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1.83 \times 10^{-4}</td>
<td>90</td>
</tr>
<tr>
<td>4</td>
<td>1.85 \times 10^{-4}</td>
<td>150</td>
</tr>
<tr>
<td>\geq 6</td>
<td>1.57 \times 10^{-4}</td>
<td>200</td>
</tr>
</tbody>
</table>
Fig 4.8. Ingwerson's Method System 1.7.1.A. ($\epsilon = 1.0$)

\[ C = \begin{bmatrix} 10 & 0 \\ 0 & 1 \end{bmatrix} \]

\[ C = \begin{bmatrix} 10 & 1 \\ 0 & 1 \end{bmatrix} \]

\[ C = \begin{bmatrix} 10 & 1 \\ 0 & 1 \end{bmatrix} \]

<table>
<thead>
<tr>
<th>Degree</th>
<th>Av. radius</th>
<th>Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.063</td>
<td>100</td>
</tr>
<tr>
<td>4</td>
<td>1.59</td>
<td>240</td>
</tr>
<tr>
<td>6</td>
<td>0.045</td>
<td>440</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Degree</th>
<th>Av. radius</th>
<th>Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.036</td>
<td>100</td>
</tr>
<tr>
<td>4</td>
<td>1.63</td>
<td>200</td>
</tr>
<tr>
<td>6</td>
<td>0.054</td>
<td>300</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Degree</th>
<th>Av. radius</th>
<th>Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1.34</td>
<td>150</td>
</tr>
<tr>
<td>4</td>
<td>0.98</td>
<td>300</td>
</tr>
<tr>
<td>6</td>
<td>1.08</td>
<td>300</td>
</tr>
</tbody>
</table>
Fig. 5.1.
(Zubov regular procedure)
\[ \phi = x^2 + y^2 \]
System 1.7.3.

<table>
<thead>
<tr>
<th>Degree</th>
<th>Av. radius</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1.16</td>
</tr>
<tr>
<td>4</td>
<td>1.71</td>
</tr>
<tr>
<td>6</td>
<td>1.35</td>
</tr>
<tr>
<td>10</td>
<td>1.41</td>
</tr>
<tr>
<td>15</td>
<td>1.31</td>
</tr>
<tr>
<td>20</td>
<td>0.918</td>
</tr>
</tbody>
</table>

Fig. 5.2.
(Zubov modified procedure)
\[ \phi = x^2 + y^2 \]
System 1.7.3.

<table>
<thead>
<tr>
<th>Degree</th>
<th>Av. radius</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1.16 (same as above)</td>
</tr>
<tr>
<td>4</td>
<td>0.925</td>
</tr>
<tr>
<td>6</td>
<td>1.45</td>
</tr>
<tr>
<td>10</td>
<td>1.57</td>
</tr>
<tr>
<td>20</td>
<td>1.07</td>
</tr>
</tbody>
</table>
**Fig. 5.3.** (radially symmetric)
(Zubov modified procedure)
\[ \phi = x^2 + y^2 \quad \text{System 1.7.2.} \]

<table>
<thead>
<tr>
<th>Degree</th>
<th>Av. radius</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1.03</td>
</tr>
<tr>
<td>4</td>
<td>1.19</td>
</tr>
<tr>
<td>6</td>
<td>1.27</td>
</tr>
<tr>
<td>10</td>
<td>1.34</td>
</tr>
<tr>
<td>20</td>
<td>1.41</td>
</tr>
</tbody>
</table>

**Fig. 5.4.** (radially symmetric)
(Zubov regular procedure)
\[ \phi = x^2 + y^2 \quad \text{System 1.7.2.} \]

<table>
<thead>
<tr>
<th>Degree</th>
<th>Av. radius</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1.03</td>
</tr>
<tr>
<td>4</td>
<td>0.72</td>
</tr>
<tr>
<td>6</td>
<td>1.21</td>
</tr>
</tbody>
</table>
Fig 5.5. (Zubov modified procedure)
\[ \phi = x^2 + y^2 \] System 1.75. \( \lambda = 0.3, \beta = 0.1 \)

<table>
<thead>
<tr>
<th>Degree</th>
<th>Avg. rad.</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.129</td>
</tr>
<tr>
<td>15</td>
<td>0.289</td>
</tr>
<tr>
<td>4,10</td>
<td>0.365</td>
</tr>
<tr>
<td>6,10</td>
<td>0.396</td>
</tr>
</tbody>
</table>

Fig 5.6. (Zubov regular procedure)
\[ \phi = x^2 + y^2 \] System 1.75. \( \lambda = 0.3, \beta = 0.1 \)
Fig. 5.7.

(Zubov modified
procedure)

\[ \phi = x^2 + y^2 \]

System 1.7.1.A. \((\varepsilon = 0.1)\)

<table>
<thead>
<tr>
<th>Degree</th>
<th>(r)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1.34</td>
</tr>
<tr>
<td>4</td>
<td>1.58</td>
</tr>
<tr>
<td>6</td>
<td>1.62</td>
</tr>
<tr>
<td>10</td>
<td>1.76</td>
</tr>
<tr>
<td>20</td>
<td>1.82</td>
</tr>
</tbody>
</table>

[N.B. The actual domain of
attraction is approx. circular
of radius 2.]
Fig. 5.8.
Flow diagram for Zubov's method.
Fig 6.3(a) Flow diagram for Rodden's method (Pt.1)
(The Spiral path)
Fig. 6.3.(b) Flow diagram for Roddens method (Pt. 2)

(First location of $\dot{V} = 0$)
Fig 6.3(c)
Flow diagram for Roddens method (Pt.3)
(Successive location of $V = 0$)
Fig 6.3 (d)
Flow diagram for Roddens method (pt. 4)
(location of tangency)
Fig. 6.3(e) Flow diagram for Rodden's method (Pt. 5)
(Tripping of the stability boundary)
Flow diagram for Roddew's method (Pt. 6)
Check for V positive; calc of Av radius.
Fig. 6.4
$\nabla V$ vanishes at tangency

Fig. 6.5
Representation of the $H$ vector along $V=0$
Fig. 6.6. (radially symmetric).

Configuration for case where several tangency points exist.
Fig 6.7.
Configuration for system 1.7.3.
with Lyapunov function:
\[ V = 2x^2 - 1.5xy + 1.25y^2 \]
\[ -0.25x^4 + 0.5x^3y \]
Fig. 6.B.
System: \( \dot{x} = y \)
\( \dot{y} = -x + y - x^2 y \).
Lapunov Function: 
\[ 1.5x^2 - xy + y^2 + 1.8x^3 + \frac{2}{3}xy^2 + 1.111y^3 \]

\( \dot{V} < 0 \) for points inside the region.

Points marked on the graph:
- P
- H
- B

The region where \( \dot{V} > 0 \) is outside the region where \( \dot{V} < 0 \).
Fig. 7.1. Simplified Flow diagram for Optimisation Procedure.
Fig. 7.2 (a)

Optimum 4th degree L.F.
(Zubov regular procedure)
System 1.7.3.
A: Initial guess \( \phi = x^2 + y^2 \)
B: Optimum \( \phi = 0.375x^2 - 0.111xy + 0.816y^2 \)

Fig. 7.2 (b)

Average radius
Fig. 7.3(a)
Optimum 6th degree L.F.
(Zubov regular procedure)
System 1.7.3

A: Initial guess \( \phi = x^2 + y^2 \)

B: Optimum \( \phi = 0.20601x^2 + 0.13168xy + 0.048032y^2 \)

Fig. 7.3(b)
Average radius
2
1.14

Number of trials
5
10
15
20
Fig. 7.4:
Optimisation of
Logwerson's method.
System 1.7.3.
(Liapounov fns of
degree ≥ 4)

A: Initial $C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$
Guess

B: Optimum $C = \begin{bmatrix} 1.79 & -3 \\ -3 & 1.84 \end{bmatrix}$

Av. radius.
Fig. 7.5.
Optimum 4th degree L.F.
(Zubov regular procedure)

System 1.7.4 A.

A: Initial guess $0.75x^2 - 0.5xy + 0.6y^2$ [Av. radius .105]
B: Optimum $0.74815x^2 - 0.4691xy + 0.66733y^2$ [Av. radius .138]
Fig. 7.6. Optimisation of
Ingham's method
System 1.7.4 A.
(4th degree Liapounov fn.)

A. Initial guess \( C = \begin{bmatrix} 1.0 \\ 0.1 \end{bmatrix} \)
B. Optimum \( C = \begin{bmatrix} 1.87 \\ 13.8 \end{bmatrix} \)

Av. radius \( 0.0095 \)

Fig. 7.7. Optimisation of
Ingham's method
System 1.7.4 A.
(6th degree Liapounov fn.)

A. Initial guess \( C = \begin{bmatrix} 433 \\ 73 \\ 0.73 \\ 1000 \end{bmatrix} \)
B. Optimum \( C = \begin{bmatrix} 3.4 \\ -0.375 \\ 3.75 \end{bmatrix} \)

Av. radius \( 0.0426 \)

Av. radius \( 0.0235 \)
Fig. 7.8 (a)

Optimum 6th degree L.F.
(Zubov regular procedure)
System 1.7.2.

A: Initial guess \( \phi = x^2 + y^2 \)

B: Optimum \( \phi = 0.632x^2 + 0.007xy + 0.0132y^2 \)

Fig. 7.8 (b)

Average radius

H20

2.14
Fig 7.9.
Optimisation of
Ingwerson's method
System 1.7.2.
(4th degree Lyapounov)

A Initial guess $C = \begin{bmatrix} 3 & 0 \\ 0 & 2 \end{bmatrix}$
B Optimum $C = \begin{bmatrix} 2.11 & -1.11 \\ -1.11 & -0.7 \end{bmatrix}$

Av. radii
A: 0.791
B: 0.94

Fig 7.10.
Optimisation of
Ingwerson's method
System 1.7.2.
(6th degree Lyapounov fn.)

A Initial guess $C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$
B Optimum $C = \begin{bmatrix} 1.62 & -1.83 \\ -1.83 & 1.63 \end{bmatrix}$

Av. radii
A: 1.02
B: 1.124
Fig. 7.11.
Optimum 6th degree L.F.
(Zubov modified procedure)
System 1.7.5. (α = 0.2, β = 0.75)

A: Initial guess \( \phi = x^2 + y^2 \)
B: Optimum \( \phi = 0.0021x^2 + 1.13xy + 3.96y^2 \)

Av. radius: 1.1, 2.3
Fig. 7.12.
Optimisation of
Ingwerson's Method
\textbf{System} 1.7.1.A (ε = 1.1)
(4\textsuperscript{th} degree Lyapunov fn.)

\begin{itemize}
  \item A: Initial guess $C = \begin{bmatrix} 0.9 \\ 0.1 \end{bmatrix}$
  \item B: Optimum $C = \begin{bmatrix} 1.945 & 0.624 \\ 0.624 & 0.423 \end{bmatrix}$
  \item C: Actual stability boundary
\end{itemize}

Fig. 7.13.
Optimisation of
Ingwerson's Method
\textbf{System} 1.7.1.A (ε = 1.0)
(6\textsuperscript{th} degree Lyapunov fn.)

\begin{itemize}
  \item A: Initial guess $C = \begin{bmatrix} 10.1 \\ 11 \end{bmatrix}$
  \item B: Optimum $C = \begin{bmatrix} 1.8 & 1.52 \\ 1.52 & 0.13 \end{bmatrix}$
\end{itemize}
Fig. 7.13 (a)

Optimisation of Krossovski’s Method

System 1.7.1.A. (ε = 0.1)

(6th degree L.F.)

A: Initial guess  \( C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \)  \( \approx 0.672 \)

B: Optimum  \( C = \begin{bmatrix} \frac{1}{6} & -\frac{1}{3} \\ -\frac{1}{3} & 2\frac{2}{3} \end{bmatrix} \)  \( \approx 0.925 \)

Av. radius
Fig 7.14
Sequential Optimisation of Zubov's method (regular).
System 1.7.3.

4th degree linap. fns.
A: Initial guess $\phi = x^2 + y^2$ (Av. radius 0.729)
B: Opt. 2nd degree $\phi = \frac{1}{3} x^2 - \frac{2}{3} xy + \frac{1}{2} y^2$ (Av. radius 0.981)
C: Opt. 3rd degree $\phi = \frac{1}{3} x^2 - \frac{2}{3} xy + \frac{1}{2} y^2$

$-\frac{1}{4} x^3 + 0.04687 x^2 + 0.391 y^2$
$+ 0.9344 y^3$ (Av. radius 0.335)

Times: $A \rightarrow B$, 500 secs
$B \rightarrow C$, 1500 secs
Fig. 7.15.

Optimisation of 3\textsuperscript{rd} degree part of $\phi$ only.

(Zubov regular procedure) (6\textsuperscript{th} degree L.F.)

System: 1.1.4.1.

A: Initial guess $\phi = \frac{1}{2} c^2 + \theta^2$ \[Av. radius: 0.17\]

B: Optimum $\phi = \frac{1}{2} c^2 + \theta^2 + 0.4118 c^3 + 0.94457 c^2 \theta$

$+ 1.3707 c \theta^2 + 1.4348 \theta^3$ \[Av. radius: 0.22\]
Fig 7.16. Simplified Flow Diagram for Sequential Optimisation.
**Fig 8.1.**

Flow diagram for Weissenberger's method.
Fig 8.2. RAS for system 1.7.3.
(See example 8.4.1)
Zubov (reg.) construction
\[ \phi = 0.0116c^2 + 1.275c\theta + 4.437\theta^2 \]
(opt. 2nd degree \( \phi \) by Weisseningberg)

<table>
<thead>
<tr>
<th>Curve</th>
<th>Degree of L.F.</th>
<th>Average radius</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>2</td>
<td>0.139</td>
</tr>
<tr>
<td>B</td>
<td>4</td>
<td>0.145</td>
</tr>
<tr>
<td>C</td>
<td>6</td>
<td>0.168</td>
</tr>
<tr>
<td>D</td>
<td>10</td>
<td>0.182</td>
</tr>
<tr>
<td>E</td>
<td>20</td>
<td>0.060</td>
</tr>
</tbody>
</table>

System 17.4.4.
(see example 8.4.2.)
Fig 8.4.
RAS for system 1.7.1.A.
(see example 8.4.3)
Fig 9.1.

Tracking Functions.
Fig. 9.2.

Graphical Tracking Function

Method. System 1.7.4.

\[ \dot{E} = 0 \] loci

A \( x^2 + y^2 \)
B \( 6x + y \)
C \( x + y \)
D \( 4x + y \)
E \( x \)
F \( x - y \)
G \( y \)
H \( x + 2y \)
Fig. 9.3.

Obtaining the allowable sector.
Fig. 9.4: Tracking Function Method.
Fig 9.5.
Numerical tracking function method.
System 1.7.4.

A 1 bisection
B 4 bisections
C actual region of practical stability
Fig. 9.6

Tracking function method with \( R_1 \) region.

System 1.7.4. (1 bisection)
Fig. 9.7. Relay control system
Fig. 9.8 Tracking Function Method

Relay Control System

\[ E_1 = x \]
\[ E_2 = y \]
\[ E_3 = x - y \]
\[ E_4 = x + y \]
Fig 9.9. Negative Resistance Oscillator

Fig 9.10. Tracking Function Method
Applied to System of Fig 9.9.

a: 1 bisection
b: 2 bisections
c: 5 

: possible constraint
Fig. 9.11.

Tracking function method in $xy$ and $r\theta$ planes

System 1.7.1 ($\epsilon = .1$)

<table>
<thead>
<tr>
<th>Curve</th>
<th>Number of tracking fns</th>
<th>Plane</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>$x\bar{y}$</td>
</tr>
<tr>
<td>B</td>
<td>3</td>
<td>$r\theta$</td>
</tr>
<tr>
<td>C</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>Actual region of practical stability</td>
<td></td>
</tr>
</tbody>
</table>
Tracking Function Method (+θ plane)  
System 1.7.4: Circular Ω₂ region.  
(3 bisections)

Fig. 9.12

Fig. 9.13
Tracking Function Method (+θ plane)

System 1.7.4.
Fig. 3 Tracking Function Method
applied to a linear non-stationary system
(Equation 9.9.5.) \( n_i = 1 \)
\[ A = -2 \quad -1 \quad 0 \quad 1 \]

\[ A = -2 \quad -1 \quad 0 \quad 1 \]

\[ A = -2 \quad -1 \quad 0 \quad 1 \]

\[ n_1 = 1.5 \]

\[ n_1 = 2 \]

\[ n_1 = 3 \]

**Fig. 9.16 (b)**

**Fig. 9.16 (c)**

**Fig. 9.16 (d)**
Fig. 9.17.
Tracking function method
applied to system of eqn. 9.9.5.
Fig. 9.18.

Tracking function

Method (4 bisections)

System of Eqn. 9.95.

n₁ = 1.03
Fig 9.19  Tracking Function Method
Applied to System of Eqn. 9.11.2.
Fig 9.20. Tracking Function Method
Applied to System of Eqn. 9.11.13.
(radially symmetric)
Fig. 10.1. Flow diagram for the calculation of \( I(m, n) = \int_{0}^{2\pi} \sin^{m-1} \cos^{n-1} dx \) (SUBROUTINE SINCOS)
Fig 10.2. \( \frac{dr}{dt} \) vs. \( t \).

Fig 10.3. The Method of Luus and Kapidus System 1.7.1.
<table>
<thead>
<tr>
<th>( r )</th>
<th>( \frac{dr}{dt} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>( 1.26 \times 10^{-1} )</td>
</tr>
<tr>
<td>0.2</td>
<td>( 6.09 \times 10^{-1} )</td>
</tr>
<tr>
<td>0.3</td>
<td>1.97</td>
</tr>
<tr>
<td>0.4</td>
<td>5.06</td>
</tr>
<tr>
<td>0.5</td>
<td>( 1.13 \times 10 )</td>
</tr>
<tr>
<td>0.6</td>
<td>( 2.27 \times 10 )</td>
</tr>
<tr>
<td>0.7</td>
<td>( 4.25 \times 10 )</td>
</tr>
<tr>
<td>0.8</td>
<td>( 7.47 \times 10 )</td>
</tr>
<tr>
<td>0.9</td>
<td>( 1.25 \times 10^2 )</td>
</tr>
<tr>
<td>1.0</td>
<td>( 2.00 \times 10^2 )</td>
</tr>
<tr>
<td>1.1</td>
<td>( 3.07 \times 10^2 )</td>
</tr>
<tr>
<td>1.2</td>
<td>( 4.54 \times 10^2 )</td>
</tr>
<tr>
<td>1.3</td>
<td>( 6.51 \times 10^2 )</td>
</tr>
<tr>
<td>1.4</td>
<td>( 9.02 \times 10^2 )</td>
</tr>
<tr>
<td>1.5</td>
<td>( 1.21 \times 10^3 )</td>
</tr>
<tr>
<td>1.6</td>
<td>( 1.37 \times 10^3 )</td>
</tr>
<tr>
<td>1.7</td>
<td>( 1.95 \times 10^3 )</td>
</tr>
<tr>
<td>1.8</td>
<td>( 2.19 \times 10^3 )</td>
</tr>
<tr>
<td>1.9</td>
<td>( 1.05 \times 10^3 )</td>
</tr>
<tr>
<td>2.0</td>
<td>( -1.18 \times 10^4 )</td>
</tr>
<tr>
<td>2.1</td>
<td>( -1.13 \times 10^5 )</td>
</tr>
<tr>
<td>2.2</td>
<td>( -8.07 \times 10^5 )</td>
</tr>
<tr>
<td>2.3</td>
<td>( -5.12 \times 10^6 )</td>
</tr>
</tbody>
</table>

Fig. 10.4. The Method of Luus and Lapidus

System 1.7.4
Example to illustrate the ambiguity in the choice of radius function (see section 11.3)
Fig 11.2. Obtaining the A.E.R.P.
Fig 11.3.
Flow diagram for
the A.E.R.P. method
Fig. 114. A.E.R.P. Method.
System 1.74.
Fig. 11.5. A.E.R.P. method
System 1.7.1.
Fig 11.6. Surge Tank System.
Fig 11.7

A.E.R.P method

System 1.75  ($\alpha = 0.2$

$\beta = 0.075$)
Fig 11.8

A.E.R.P Method

System 1.7.2.
Fig. A.14.1. Flow diagram for Modified Simplex method.