Application of interval analysis to problems of linear control systems

Edward Phillip Oppenheimer
Iowa State University

Follow this and additional works at: https://lib.dr.iastate.edu/rtd
Part of the Electrical and Electronics Commons

Recommended Citation
https://lib.dr.iastate.edu/rtd/5163
INFORMATION TO USERS

This material was produced from a microfilm copy of the original document. While the most advanced technological means to photograph and reproduce this document have been used, the quality is heavily dependent upon the quality of the original submitted.

The following explanation of techniques is provided to help you understand markings or patterns which may appear on this reproduction.

1. The sign or “target” for pages apparently lacking from the document photographed is “Missing Page(s)”. If it was possible to obtain the missing page(s) or section, they are spliced into the film along with adjacent pages. This may have necessitated cutting thru an image and duplicating adjacent pages to insure you complete continuity.

2. When an image on the film is obliterated with a large round black mark, it is an indication that the photographer suspected that the copy may have moved during exposure and thus cause a blurred image. You will find a good image of the page in the adjacent frame.

3. When a map, drawing or chart, etc., was part of the material being photographed the photographer followed a definite method in “sectioning” the material. It is customary to begin photoing at the upper left hand corner of a large sheet and to continue photoing from left to right in equal sections with a small overlap. If necessary, sectioning is continued again — beginning below the first row and continuing on until complete.

4. The majority of users indicate that the textual content is of greatest value, however, a somewhat higher quality reproduction could be made from “photographs” if essential to the understanding of the dissertation. Silver prints of “photographs” may be ordered at additional charge by writing the Order Department, giving the catalog number, title, author and specific pages you wish reproduced.

5. PLEASE NOTE: Some pages may have indistinct print. Filmed as received.

Xerox University Microfilms
300 North Zeeb Road
Ann Arbor, Michigan 48106
OPPENHEIMER, Edward Phillip, 1932-
APPLICATION OF INTERVAL ANALYSIS TO PROBLEMS
OF LINEAR CONTROL SYSTEMS.

Iowa State University, Ph.D., 1974
Engineering, electrical

Xerox University Microfilms, Ann Arbor, Michigan 48106

THIS DISSERTATION HAS BEEN MICROFILMED EXACTLY AS RECEIVED.
Application of interval analysis to problems of linear control systems

by

Edward Phillip Oppenheimer

A Dissertation Submitted to the Graduate Faculty in Partial Fulfillment of The Requirements for the Degree of DOCTOR OF PHILOSOPHY

Major: Electrical Engineering

Approved:

Signature was redacted for privacy.

In Charge of Major Work

Signature was redacted for privacy.

For the Major Department

Signature was redacted for privacy.

For the Graduate College

Iowa State University
Ames, Iowa

1974
TABLE OF CONTENTS

CHAPTER I. INTRODUCTION  1
   Motivation of the Research  1
   Dissertation Objectives  2
   Class of Problems Considered  3
   Dissertation Outline  3

CHAPTER II. MATHEMATICAL RESULTS  8
   Interval Analysis  8
      Interval arithmetic and topological considerations  8
      Continuous interval functions and a convergence result  11
      An interval exponential computation technique  35
      Continuous interval matrix functions and the convergence result  42
      The interval matrix exponential computation technique  50
      Conservativeness, bounding interval arithmetic and IBM floating-point computation considerations  59
      The Householder matrix norms and computation of the real fundamental matrix  69
      Nested centered form computations for the interval fundamental matrix  79

The Kalman Covariance Equation and Two Delayed State Inertial Navigation Solution Methods  92

The Optimal Linear Regulator Design with Minimum Plant Variation Sensitivity  98
CHAPTER III. NUMERICAL BOUNDING INTERVAL ARITHMETIC IMPLEMENTATIONS
The Washington State - Mathematics Research Center (University of Wisconsin) Package
The Fast Assembler Routines

CHAPTER IV. NUMERICAL INTERVAL METHODS FOR INITIAL-VALUE PROBLEMS IN ORDINARY DIFFERENTIAL EQUATIONS INVOLVING A PARAMETER
Autonomous Systems of Nonlinear First-Order Equations
Autonomous Systems of Linear First-Order Equations

CHAPTER V. NUMERICAL INTERVAL RESULTS
Linear Interval Integration Algorithm Examples
The parameterized RLC circuit—a second-order problem
The linearized instrument servomechanism with load inertial parameter—a fourth-order problem
The minimum plant sensitivity optimal linear regulator design—a fifth-order problem
Bounding Interval Arithmetic—Subdistributivity Considerations
Computation order dependence in an undriven observable random walk process covariance equation—an observation
Two methods for the delayed-state inertial navigation problem solution—a comparison

CHAPTER VI. CONCLUSIONS

LITERATURE CITED

ACKNOWLEDGMENTS
**LIST OF FIGURES**

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Isometric representation of $J$ and $J_1$</td>
<td>15</td>
</tr>
<tr>
<td>2.2a</td>
<td>Isometric representation of interval addition</td>
<td>15</td>
</tr>
<tr>
<td>2.2b</td>
<td>Isometric representation of interval subtraction</td>
<td>16</td>
</tr>
<tr>
<td>2.2c</td>
<td>Isometric representation of interval multiplication</td>
<td>16</td>
</tr>
<tr>
<td>2.3</td>
<td>IBM/360 single-precision floating-point number representation for $-5.25$</td>
<td>65</td>
</tr>
<tr>
<td>4.1</td>
<td>Flowchart for the $\bar{\theta}(t_1,I)$ computation of (4.4)</td>
<td>132</td>
</tr>
<tr>
<td>5.1</td>
<td>Solution response for the second-order example (calculated in double-precision) for three damping ratio values</td>
<td>172</td>
</tr>
<tr>
<td>5.2</td>
<td>Interval bound solutions for the second-order example ($\delta = [0.1,0.3]$)</td>
<td>176</td>
</tr>
<tr>
<td>5.3</td>
<td>Nonlinear positional motor-generator instrument servomechanism</td>
<td>178</td>
</tr>
<tr>
<td>5.4</td>
<td>Interval bound solution endpoints for the fourth-order example ($J_L = [2700.0,3300.0]$)</td>
<td>183</td>
</tr>
<tr>
<td>5.5</td>
<td>Interval bound solutions for the fifth-order example ($a_{11} = [-2.0,-1.8]$, $a_{22} = [-1.0,-1.5]$)</td>
<td>193</td>
</tr>
<tr>
<td>5.6</td>
<td>Single-step covariance trace guard digit interval arithmetic endpoint error results for three computational techniques initialized with R<em>8 covariance solution. Error = (R</em>4 guard digit endpoint result - R<em>8 solution result)/(R</em>8 solution result)</td>
<td>209</td>
</tr>
<tr>
<td>5.7</td>
<td>Errors for the bounding INTERVAL arithmetic and R<em>4 results: 4th component of $K$ (error = R</em>4 result - R*8 result)</td>
<td>217</td>
</tr>
</tbody>
</table>
5.8 Errors for the bounding INTTRVAL arithmetic and R*4 results: 6th component of $K_k$ (error = R*4 result - R*8 result) 218

5.9 Errors for the bounding INTTRVAL arithmetic and R*4 results: scalar $Q_k$ (error = R*4 result - R*8 result) 219

A-1 Abbreviated assembler flowchart for COMPLEX function ADD(A,B) with multiple entry point COMPLEX function SUB(A,B) 237

A-2 Abbreviated assembler flowchart for COMPLEX function MUL(A,B) with multiple entry point COMPLEX function DIV(A,B) 240

A-3 Abbreviated MINMAX assembler flowchart insert for COMPLEX function MUL(DIV) 243

B-1 Abbreviated flowchart for the algorithm operation indicating the required user subroutine calls A-E 252

B-2 Abbreviated flowchart for subroutine DRIVER operation 253

B-3 Abbreviated flowchart for subroutine PHCOMP operation 256
LIST OF TABLES

Table                                      Page

2.1 Summary of results for various interval arithmetic representations of \( f(x) = x - x^2 \),
\( x \in J = \left[ \frac{1}{2} - r, \frac{1}{2} + r \right] \) 62

2.2 Numerical results for the computation of \( J \)
in the two examples (2.61) and (2.62),
with various matrix norms 77

5.1 Interval and R*4 (hexadecimal) results for the trace of the covariance equation, \( \text{Tr}(P_k) \),
computed three ways 206

5.2 Miscellaneous comparison data for the two bounding interval arithmetic implementations of the two computational methods 216
CHAPTER I. INTRODUCTION

Motivation of the Research

In his preface to the advanced seminar proceedings entitled "Error in Digital Computation" (1), editor Louis B. Rall pointedly remarks that the speed of modern electronic digital computers has relegated the complexities of calculation to a secondary position, thereby allowing the problem of error estimation to reach the forefront.

To anyone involved in the error analysis of a specific numerical technique, the difficulties encountered are frequently evidenced by either a lack of information on the subject or an intricate analysis within a spectrum that contains such techniques as "orders of magnitude", functional analysis and statistical methods, each usually pessimistic in result. Further compounding the problem, numerical errors variously result from local intermediate calculation "round-off"\(^1\) and its accumulation, inexact initial data, discretization by the algorithm and the use of experimentally inaccurate quantities in the original mathematical model, any one of which may be so deleterious to the numerical solution convergence that it renders

\(^1\)In the case of the IBM System/360 Model 65 machine, "round-off" error is a misnomer since truncation occurs rather than rounding.
plausible appearing results completely meaningless.

It is therefore an attractive approach to use the speed of the computer to advantage and turn over the numerical error bounding to the machine. The topic of interval analysis is a discipline suited for this purpose.

In his book, *Interval Analysis* (2), Moore applies interval techniques to initial-value problems of nonlinear ordinary differential equations to determine numerical error inclusive bounds for solution trajectories. Since this class of problems is frequently encountered in automatic control theory, it becomes appealing to extend the technique to determine trajectory bounds which also include the effects of a perturbation parameter in the differential equations.

**Dissertation Objectives**

The objective of this dissertation is to investigate the application of interval analysis to the numerical integration of initial-value problems of ordinary differential equations involving a perturbation parameter, calculating interval bounding solutions or envelopes which contain the set of all solutions associated with the parameter and the effects of algorithmic numerical error.

A subordinate objective is to empirically investigate the subdistributivity properties of interval arithmetic on
a few specific numerical algorithms consociate to automatic control theory.

Class of Problems Considered

The investigation of numerical interval integration embraces the class of initial-value problems of autonomous systems of linear first-order ordinary differential equations having coefficients that involve a perturbation parameter. To a less successful degree, the research preliminarily considers analogous systems which are nonlinear but rational.

The empirical investigation of interval arithmetic subdistributivity properties appraises contrasting algorithmic forms for two Kalman filter estimation problems.

Dissertation Outline

Chapter II presents a large part of the mathematical development necessary for the numerical and analytic discussion of the subsequent chapters.

The necessary particulars of exact interval arithmetic and a brief topological discussion are included first. Then a sequence of continuous interval function propositions is developed in the metric space topology, concluding in an interval function convergence result which embodies a fundamental concept to be used in the linear interval
integration algorithm. A computation technique is developed for the interval exponential and analogously, the sequence of continuous matrix interval function propositions, concluding interval matrix function convergence result and the interval matrix exponential computation technique are presented. Next, a discussion of techniques used to reduce the conservativeness of the interval arithmetic, the need for a bounding interval arithmetic and an introduction to the IBM floating-point computations are presented. The Householder matrix norms and their use in computation of a real fundamental matrix are then presented. The section concludes with the development of the computation technique for the interval fundamental matrix which embodies the nested centered form interval computation methods, use of the Householder matrix norms to bound certain metric quantities arising in the technique and the convergence result.

In the next section the Kalman filter equations are briefly presented for an observable, undriven random walk process and a delayed state inertial navigation problem.

Concluding the chapter, the optimal linear regulator design for minimum plant variation cost functional sensitivity is given.

Chapter III is devoted to implementing the bounding interval arithmetic. Routines developed by the Mathematics
Research Center at the University of Wisconsin and modified for the IBM computer at Washington State University are discussed. Then, assembler language functions are designed to subrogate the Washington State package, minimizing the inconvenience of excessive execution time.

Chapter IV briefly discusses the pessimistic results of early efforts to extend the interval integration techniques of Moore to initial-value problems of autonomous systems of first-order nonlinear ordinary differential equations involving a perturbation parameter.

The investigation is then more successfully constrained to analogous linear systems, utilizing the problem linearity and applying the concepts of nested centered form bounding interval computation methods, the interval matrix exponential and the "interval" Householder matrix norms. A discussion is included to connect the analytic results of Chapter II and the actual implementation. Various aspects of the resulting algorithm are discussed.

The first part of Chapter V presents results obtained when applying the linear interval integration algorithm to various parameterized examples.

First, trajectory bounds are computed for a unit step driven second-order RLC circuit where the damping ratio is the perturbation parameter.

Next, a fourth-order linearized instrument servomechanism
example is examined when the load inertia is the perturbation parameter and the system is driven with a unit step displacement.

Finally, a fifth-order minimum plant variation sensitivity optimal linear regulator design is examined where the initial condition is the unit vector and the design is based on variation in two plant coefficients. The problem is computed for the minimum sensitivity criterion and again under the same initial condition and plant coefficient variations when the minimum sensitivity criterion is not observed, yielding a marked contrast in the optimal regulator design performance which strongly enhances minimum sensitivity considerations.

The second part of Chapter V is devoted to empirical observation of the interval arithmetic subdistributivity property when applied to contrasting forms of certain Kalman filter algorithms.

Using truncated double-precision covariance equation results as initial-values, the bounding interval arithmetic covariance equation trace errors are stepwise compared for three methods of computation for an observable, undriven random walk process, an inherently unstable numerical problem.

Following this, the accumulated bounding interval arithmetic solution errors are computed for two forms of
a delayed state inertial navigation problem.

Concluding remarks and recommendations for future research are given in Chapter VI.

Appendix A contains the documented assembler language bounding interval arithmetic function flowcharts and listings discussed in Chapter III.

Appendix B consists of flowcharts and the complete "user" annotated Fortran IV language linear interval integration algorithm devised in Chapter IV.

Appendix C contains selected results from functional analysis which are used in Chapter II.
CHAPTER II. MATHEMATICAL RESULTS

Throughout the dissertation it will be the practice to define notation and symbols as they initially appear. The notation \{x|P(x)\} is used for "the set of all x such that the proposition P(x) is true". The symbols ∈, ⊆, ∪ and ∩ are used in the usual set theory sense.

Interval Analysis

**Interval arithmetic and topological considerations**

Let

\[ \mathcal{I} = \{\text{set of all finite closed intervals } [a,b] | a,b \in \mathbb{R}^1, a \leq b\} \]

where

\[ \mathbb{R}^1 \equiv \text{the real line}. \]

The interval arithmetic operations are defined by

\[ I*J \equiv \{x*y | x \in I, y \in J\}, I,J \in \mathcal{I}, \]

where the symbol * indicates one of the arithmetic operations\(^2\) +, −, • and /, except that \( I/J \) is not defined if 0 ∈ J.

---

\(^1\) The interval \( I = [a,a] \) is called a "degenerate" interval.

\(^2\) The same symbol will be used for both interval and real arithmetic operations and the usual practice of omitting the symbol • will be followed.
Equivalently, in terms of the corresponding real interval endpoints

\[[a,b] + [c,d] \equiv [a+c, b+d],\]
\[[a,b] - [c,d] \equiv [a-d, b-c],\]
\[[a,b][c,d] \equiv \min(ac,ad,bc,bd), \max(ac,ad,bc,bd)]\text{ and}
\[[a,b]/[c,d] \equiv [a,b][1/d,1/c], \text{ provided } 0 \notin [c,d].\]

(2.4)

Attempts to determine a prevalent abstract mathematical structure for interval arithmetic are not entirely productive.

It follows from definition (2.3) that interval addition and multiplication are each associative and commutative operations and that the degenerate intervals \([0,0]\) and \([1,1]\) are their respective identity elements. However, it is obvious from definition (2.3) that if an interval has distinct endpoints, its inverse with respect to addition or multiplication does not exist since the result of either operation cannot be a degenerate identity element.

In the abstract algebraic sense then, \(\mathbb{I},+\) and \(\mathbb{I},\cdot\) are commutative semigroups with identities\(^1\) (3, p. 68) but fail as groups and therefore as Abelian groups.\(^2\) Therefore

\(^1\)More exactly, \(\mathbb{I},+\) is a commutative cancellation semigroup with an identity.

\(^2\)Also, both fail as quasi-groups and therefore as loops.
\{\mathcal{I},+,*\} \text{ fails to be a ring (3, p. 73).}

Using definition (2.4) it is easily demonstrated that multiplication is not necessarily distributive with respect to addition\(^1\) and therefore \(\{\mathcal{I},+,*\}\) fails as a ring for a second reason. However, from set theory it always follows from definition (2.3) that if \(I,J,K \in \mathcal{I}\), then

\[ I(\overline{J+K}) \subset \overline{IJ+IK} \text{ ,} \tag{2.5} \]

which is called the "subdistributivity" property\(^2\) of interval arithmetic. It also follows that if \(I,J,K, L \in \mathcal{I}\), \(I \subset K\) and \(J \subset L\), then

\[ I*J \subset K*L \text{ ,} \tag{2.6} \]

provided in the case of division that \(0 \notin L\). This property of interval arithmetic is called "monotonic inclusion".

When related to the functional analysis setting (4, p. 10), since \(\{\mathcal{I},+\}\) is not an Abelian group, interval arithmetic does not yield the structure of a real linear vector space.

An interesting effort to relate interval arithmetic

\(^1\)For example, \([1,2][1,2]+[-2,-1]) = [1,2][-1,1] = [-2,2]\), while \([1,2][1,2]+[1,3][-2,-1]) = [1,4]+[-4,-1] = [-3,3].

\(^2\)This property is frequently employed in efforts to reduce the conservativeness of the interval arithmetic results.
to an abstract structure is presented by Sunaga (5). Using the set inclusion relation, it is possible to partially order (6, p. 131) and by defining binary interval operations of "meet" and "join" (conceptually paralleling the inf and sup respectively of real analysis) it is possible to satisfy the structure of a lattice in the sense of Birkhoff (7, p. 6). However, the results of this memoir appear to be largely aesthetic in nature.

Continuous interval functions and a convergence result

The workable abstract mathematical structure which will be employed in this dissertation is that of the metric spaces. The subsequent development is constructed in this setting and concludes in a convergence result for certain continuous interval functions. This result embodies the "scalar" case analog of a philosophy upon which the linear interval integration algorithm will be founded, convergence of the interval exponential function which is defined by an infinite interval series (paralleling the fundamental matrix of linear ordinary differential equations) and convergence in union over the subdivisions of the interval variable of the function.

To the best of the author's knowledge the theoretical functional analysis development presented here as a generalization and extension of the concepts introduced in (2) (without the burden of the advanced topological
concepts—see Equation 2.10 and the footnote), does not appear in the literature.

Let \( \{X, D_\infty\} \) be the complete metric space where \( X \equiv \mathbb{R}^2 \), the 2-dimensional Euclidean space and \( D_\infty \) is the infinity metric (6, pp. 37, 113, 124). The elements of \( X \) are represented by the vector and 2-tuple notation \( x \equiv (x_1, x_2) \) and \( y \equiv (y_1, y_2) \). The metric \( D_\infty \) is defined by

\[
D_\infty(x, y) \equiv \max(|x_1 - y_1|, |x_2 - y_2|) .
\]

**Proposition 2.1:** Let \( X_1 \) be the subset of \( X \) defined by \( X_1 \equiv \{x \in X | x_1 \leq x_2\} \). Then \( \{X_1, D_\infty\} \) is a complete metric subspace of \( \{X, D_\infty\} \).

**Proof:** \( X_1 \) is the half plane above and including the line \( x_1 = x_2 \). Let the local neighborhoods of a point \( x_0 \in X \) be the open spheres defined by

\[
S(x_0, r) \equiv \{x \in X | D_\infty(x, x_0) < r\} , \ r \in \mathbb{R}^1 , \ r > 0 .
\]

The complement of the subset \( X_1 \), denoted \( X_1^c \), is open in \( X \) in the topology induced by the metric space \( \{X, D_\infty\} \), since for every point \( x \in X_1^c \), there exists an \( \varepsilon > 0 \) such that \( S(x, \varepsilon) \subset X_1^c \). Therefore \( X_1 \), the complement of \( X_1^c \), is closed in this topology and hence \( \{X_1, D_\infty\} \) is a subspace of \( \{X, D_\infty\} \) and it is complete (8a, pp. 57, 77, 86, 116).
Proposition 2.2: The metric space \( (\mathcal{J}, \rho) \), where \( \mathcal{J} \) is defined in (2.1) and for each \( I, J \in \mathcal{J}, I = [a, b], J = [c, d] \), the metric \( \rho \) is defined by

\[
\rho(I, J) = \max(|a-c|, |b-d|),
\]

(2.7)
is complete.

Proof: It is obvious that \( \rho \) satisfies the properties of a metric. Let \( f \) be the mapping \( f: \mathcal{J} \to X_1 \) defined by \( f([a, b]) = (a, b) \). Then for each \( I, J \in \mathcal{J} \), \( D_\infty(f(I), f(J)) = \rho(I, J) \) and \( f \) is an isometry. Since \( (\mathcal{J}, \rho) \) is isometric to the complete metric space \( (X_1, D_\infty) \), it is complete (8a, p. 117).

Proposition 2.3: Let \( X_2 \) be the subset of \( X_1 \) defined by

\[
X_2 = \{ x \in X_1 | a \leq x_1 \leq x_2 \leq b, a, b \in \mathbb{R}^1 \}.
\]

Then \( X_2, D_\infty \) is a complete and compact metric subspace of \( (X_1, D_\infty) \) and \( (X, D_\infty) \).

Proof: Since \( X_2 \subset X_1 \subset X \equiv \mathbb{R}^2 \) and \( X_2 \) is closed and bounded in \( X \) and \( X_1 \) in the topology induced by \( (X, D_\infty) \), it is compact by the Borel theorem (6, p. 66). But since \( (X, D_\infty) \) is complete and \( X_2 \) is a closed set of \( X \), \( (X_2, D_\infty) \) is also complete (4, p. 77).

Proposition 2.4: Let \( I \equiv [a, b] \in \mathcal{J} \) and

\[
\mathcal{J}_I \equiv \{ J \in \mathcal{J} | J \subset I \}.
\]

Then \( (\mathcal{J}_I, \rho) \) is a complete and compact metric subspace of \( (\mathcal{J}, \rho) \).
Proof: Obviously $\mathcal{J}_I \subset \mathcal{J}$ and therefore $\{\mathcal{J}_I, \rho\}$ is a subspace of $\{\mathcal{J}, \rho\}$. But using the isometry $f$ defined in Proposition 2.2, $\{\mathcal{J}_I, \rho\}$ is also a complete and compact metric space since it is isometric to the complete and compact metric space $\{X^2, D_\infty\}$ of Proposition 2.3 (8a, p. 146).

Since the isometry $f$ of Proposition 2.2 provides a convenient geometric interpretation for the sets $\mathcal{J}$ and $\mathcal{J}_I$ in the plane $X \equiv \mathbb{R}^2$, this conveyance will be used to illustrate the various concepts involved and the notation for the elements of $\mathcal{J}$ will be used directly on the plane. The symbol $\sim$ will occasionally be used to emphasize an isometric correspondence. Figure 2.1 gives the isometric representation of $\mathcal{J}$ and $\mathcal{J}_I$ where $I = [a, b] \in \mathcal{J}$.

The real line $\mathbb{R}^1$, when considered as the metric space $\{\mathbb{R}^1, d\}$, where $d(a, b) \equiv |a - b|$, $a, b \in \mathbb{R}^1$, is isometrically embedded in $\{\mathcal{J}, \rho\}$ under the mapping $g: \mathbb{R}^1 \rightarrow \mathcal{J}$ defined by $g(a) \equiv [a, a]$, an isometry because $\rho(g(a), g(b)) = d(a, b)$. In Figure 2.1 the line $x_1 = x_2$ is the isometric image of the real line under $fg$ and the degenerate intervals of $\mathcal{J}$ under $f$.

Figure 2.2a and b respectively illustrate interval addition and subtraction as geometric "vector additions" in $\{X^1, D_\infty\}$. The negative of the element $J \in \mathcal{J}$ provides an insight into the fact that, with respect to the operation
Figure 2.1. Isometric representation of $\mathcal{J}$ and $\mathcal{J}_I$

Figure 2.2a. Isometric representation of interval addition
I-J=I+(-J)  

Note: -JεJ is the "reflection" of J "about the line" 
x₁=−x₂ in X₁ in the sense that if J=[a,b], then -J=[−b,−a].

Figure 2.2b. Isometric representation of interval subtraction

\(X₁(\sim J)\)

\(I=[-2,3]\) \(J=[-1,3/2]\)

\{t\cdot I|t\in J, t\geq 0\}^*

\{t\cdot I|t\in J, t<0\}^*  

*Note that tI<IJ for every tεJ.

Figure 2.2c. Isometric representation of interval multiplication
of interval addition, the inverse of an element of $J$
exists if and only if the element is "on the line" $x_1 = x_2$
(that is, if the element is one of the degenerate intervals
of $J$).

Figure 2.2c gives the geometric interpretation of the
interval multiplication $IJ$ (8b, p. 29) as the unique ele­
ment of $J$ such that $tI \subseteq IJ$ for every $t \in J$ and $sI \neq IJ$ if
$s \notin J$. For brevity in notation in the interval multipli­
cation, $tI$, the degenerate interval $[t, t]$ has been indicated
simply by $t$.

Let $\mathcal{F} \equiv \{f | f: J \rightarrow \mathcal{I}, f \text{ continuous on } J \}$.  

If $f, g \in \mathcal{F}$, let

$$
\mu(f, g) \equiv \sup_{J \in J} \{\rho(f(J), g(J))\}.
$$

(2.9)

Then $\mu$ is a metric on $\mathcal{F}$ (assume $f, g, h \in \mathcal{F}$) since,

(i) by the symmetry of $\rho$,

$$
\mu(f, g) = \mu(g, f),
$$

(ii) since $\rho(J, K) = 0$ if and only if $J = K$,

$$
\mu(f, g) = 0 \text{ if and only if } f(J) = g(J) \text{ for every } J \in J \text{ and then } f \equiv g \text{ and }
$$

(iii) by the triangle inequality for $\rho$,
\[ \sup_{J \in \mathcal{J}_I} \{ \rho(f(J), g(J)) \} \leq \sup_{J \in \mathcal{J}_I} \{ \rho(f(J), h(J)) + \rho(h(J), g(J)) \} \]
\[ \leq \sup_{J \in \mathcal{J}_I} \{ \rho(f(J), h(J)) \} + \sup_{K \in \mathcal{J}_I} \{ \rho(h(K), g(K)) \}, \]

thus \( \mu(f, g) \leq \mu(f, h) + \mu(h, g) \) which is the triangle inequality for \( \mu \).

Therefore \( \{ \mathcal{S}, \mu \} \) is a metric space of continuous interval functions.

**Proposition 2.5:** The metric space \( \{ \mathcal{S}, \mu \} \) is complete.

**Proof:** Let \( \{ f_n \} \) be an arbitrary Cauchy sequence in \( \{ \mathcal{S}, \mu \} \). This means that for any \( \varepsilon > 0 \), there exists an \( N_0(\varepsilon) \) such that if \( n, m \geq N_0(\varepsilon) \), this implies that

\[ \rho(f_n(J), f_m(J)) \leq \mu(f_n, f_m) < \varepsilon \text{ for every } J \in \mathcal{J}_I. \]

In particular then, for fixed \( J_0 \in \mathcal{J}_I \), \( \{ f_n(J_0) \} \) is a Cauchy sequence in \( \{ \mathcal{J}, \rho \} \). Since \( \{ \mathcal{J}, \rho \} \) is complete (Proposition 2.2), there exists an element \( f(J_0) \in \mathcal{J} \) such that

\[ \rho(f_n(J_0), f(J_0)) \to 0 \text{ as } n \to \infty. \]

Applying the same argument for each \( J \in \mathcal{J}_I \), obtain the function definition \( f \equiv f(J) \), \( J \in \mathcal{J}_I \).

Since for \( n, m \geq N_0(\varepsilon) \), \( \rho(f_n(J), f_m(J)) \to 0 \text{ for each } J \in \mathcal{J}_I \).
$J \notin \mathcal{J}_I$, letting \( n \to \infty \), it follows that,

$$\rho(f_n(J), f(J)) \leq \varepsilon \text{ for each } J \in \mathcal{J}_I.$$  

Since \( N(\varepsilon) \) is independent of \( J \), it is therefore true that the sequence \( \{f_n\} \) converges uniformly to \( f \) on \( \mathcal{J}_I \).

By the triangle inequality, for any \( J, J_0 \in \mathcal{J}_I \) and for any \( n \),

$$\rho(f(J), f(J_0)) \leq \rho(f(J), f_n(J)) + \rho(f_n(J), f_n(J_0)) + \rho(f_n(J_0), f(J_0)).$$

By the uniform convergence of the sequence \( \{f_n\} \), the first and third terms can be made \( < \frac{\varepsilon}{3} \) by choosing \( n \) sufficiently large. But since each \( f_n \) is continuous on \( \mathcal{J}_I \) and \( \mathcal{J}_I \) is compact, each \( f_n \) is also uniformly continuous on \( \mathcal{J}_I \) (p. 121) and therefore there is a \( \delta = \delta(\varepsilon) \) such that if \( \rho(J, J_0) < \delta(\varepsilon) \), then \( \rho(f_n(J), f_n(J_0)) < \frac{\varepsilon}{3} \). Hence \( \rho(f(J), f(J_0)) < \varepsilon \). This means that \( f \) is continuous at \( J_0 \) and since \( J_0 \) is arbitrary, the argument is finished, namely \( f \in \{\mathcal{F}, \mu\} \).

Thus \( \{\mathcal{F}, \mu\} \) is complete.

Since \( \mathcal{J}_I \) and \( \mathcal{J} \) include the degenerate intervals, embedded in \( \mathcal{F} \) are the continuous real functions. Consequently, embedded in \( \{\mathcal{F}, \mu\} \) is the metric space of continuous real functions \( C[a, b] \) with sup metric \( d \), \( f, g \in C[a, b] \),

$$d(f, g) \equiv \sup_{t \in [a, b]} |f(t) - g(t)|.$$
It should also be repeated, since each \( f \in \mathcal{F}_I \) is continuous, \( \mathcal{I} = (I, \rho) \) and \( \mathcal{I} = (J, \rho) \) are metric spaces and \( \mathcal{I} \) is compact, then each \( f \in \mathcal{F}_I \) is uniformly continuous on \( \mathcal{I} \) (6, p. 121).

For any \( f \in \mathcal{F}_I \), \( J \in \mathcal{I} \), define the real values \( f^L(\cdot) \) and \( f^R(\cdot) \) by

\[
f(J) \equiv [f^L(J), f^R(J)].
\]  
\[ (2.10) \]

For any \( f \in \mathcal{F}_I \), \( J \in \mathcal{I} \), define the "united extension of \( f \)," \( \tilde{f}(J) \equiv \bigcup_{x \in J} f([x,x]) \).  
\[ (2.11) \]

In the following pages it will be the practice to omit the terminology "united extension of \( f \)" and simply assume the correspondence \( f, \tilde{f} \) and \( \{f_n\}, \{\tilde{f}_n\} \). The following proposition yields significance to this definition in terms of the metric space adopted here.

---

\(^1\)This terminology is used by Moore (2, p. 18). He refers the reader to a definitive article on fixed point theorems for multivalued functions (9, p. 552). Although the article has its foundations in the advanced topological concepts of Peano spaces (see compact Hausdorff spaces in reference 10), it still does not precisely include the case here. As will be seen, this matter takes on meaning at a considerably less abstract level of discussion.
Proposition 2.6: For arbitrary $f \in \mathcal{F}_\mu$ and $J \in \mathcal{J}$,

$$
\bar{f}(J) = \bigcup_{x \in J} f([x,x]) = [p(J), q(J)] \in \mathcal{J},
$$

(2.12)

where

$$
p(J) = \inf_{x \in J} f^L([x,x]),
$$

and

$$
q(J) = \sup_{x \in J} f^R([x,x]).
$$

Proof: Clearly

$$
\bigcup_{x \in J} f([x,x]) \subset [p(J), q(J)],
$$

since for every $x \in J$, by (2.10) it is true that $p(J) \leq f^L([x,x]) \leq f^R([x,x]) \leq q(J)$. Since $f$ is continuous on $\mathcal{J}$ and $\mathcal{J}$ is compact, $J \in \mathcal{J}$ and $S = \{[x,x] | x \in J\} \subset \mathcal{J}$, the values

$$
\inf_{t \in S} f^L(t) = p(J) \quad \text{and} \quad \sup_{t \in S} f^R(t) = q(J)
$$

are each attained (8a, p. 146) and $[p(J), q(J)] \in \mathcal{J}$. Thus it is true that

$$
\bigcup_{x \in J} f([x,x]) \supset \{p(J), q(J)\}, \text{ the two points.}
$$

It is then sufficient to show that every interior point of $[p(J), q(J)]$ is contained in $\bigcup_{x \in J} f([x,x])$. But this must be
true since \( f \) is continuous on \( J_I \) and the subset \( S \) is connected (note \( \bigcup_{t \in S} \bigcup_{x \in J} f([x,x]) \supseteq [p(J),q(J)] \)). Thus
\[
\bigcup_{x \in J} f([x,x]) \supseteq [p(J),q(J)] ,
\]
and since the set inclusion relation holds both ways, equality is obtained.

**Proposition 2.7:** For arbitrary \( f \in \{\mathcal{F},\mu\} \), \( \bar{f} \in \{\mathcal{F},\mu\} \).

**Proof:** For continuity of \( \bar{f} \) on \( J_I \), it is necessary and sufficient to show that for each \( J \in J_I \), each sequence \( \{J_n\} \subseteq J_I \), \( J_n \rightarrow J \), it always follows that \( \{\bar{f}(J_n)\} \rightarrow \bar{f}(J) \).

Suppose that for some \( J \in J_I \), \( \{J_n\} \subseteq J_I \), with \( J_n \rightarrow J \), it occurs that \( \{\bar{f}(J_n)\} \neq \bar{f}(J) \). By Proposition 2.6, this means that \( \{[p(J_n),q(J_n)]\} \neq [p(J),q(J)] \). The sets \( S \equiv \{[x,x] | x \in J\} \) and \( S_n \equiv \{[x,x] | x \in J_n\}, n=1,2,... \) are each individually connected subsets of the compact set \( J_I \). But \( f \) is continuous on \( J_I \) and also on its subsets and by the compactness of \( J_I \), the inf and sup of \( f^L([x,x]) \) and \( f^R([x,x]) \) respectively are attained for each set \( S \) and \( S_n \).
Thus \( \{[p(J_n),q(J_n)]\} \neq [p(J),q(J)] \) must mean that there are some points \([y,y]\) which do not belong to both \( S \) and the \( S_n \)'s, for an infinite number of the \( n \)'s. But since
\[
\bigcup_{t \in S} t = J \quad \text{and} \quad \bigcup_{t \in S_n} t = J_n', \quad n=1,2,...,
\]
this implies that \(\{J^*_n\} \not\subset J\), which is a contradiction.
Hence \(\{f(J^*_n)\} \rightarrow f(J)\) and this is true for every \(J \in \mathcal{J}_1\),
\(\{J^*_n\} \in \mathcal{J}_1\), whenever \(\{J^*_n\} \rightarrow J\), thus \(f\) is continuous on \(\mathcal{J}_1\)
and belongs to \(\{\mathcal{J}, \mu\}\). Since \(f\) is arbitrary, the above
is true for every \(f\).

An interval function will be called a "rational"
interval function if it is defined and can be expressed
as a rational interval arithmetic expression in the in-
terval variable and a finite set of constant coefficient
intervals.

**Proposition 2.8:** The rational interval functions belong
to \(\{\mathcal{J}, \mu\}\).

**Proof:** Let \(Y = \mathcal{J}_1 \otimes \mathcal{J}_2\) where \(\mathcal{J}_1 \equiv \mathcal{J}_2 \equiv \mathcal{J}\) (that is,
the Cartesian product space \(\mathcal{J} \otimes \mathcal{J}\)). Denote the elements
of \(Y\) by the 2-tuple \((I,J)\), \(I \in \mathcal{J}_1\), \(J \in \mathcal{J}_2\). Then for \((I,J),
(K,L) \in Y\), induce the metric,

\[\rho_Y((I,J),(K,L)) \equiv \max\{\rho(I,K),\rho(J,L)\}\]

where \(\rho\) is defined as before on \(\mathcal{J}\). The completeness of
\(\{Y, \rho_Y\}\) as a metric space follows directly from the com-
pleteness of \(\{\mathcal{J}, \rho\}\).

It is obvious that the interval arithmetic operations
defined by (2.3) are mappings \(h:Y \rightarrow \mathcal{J}\) defined by
\(h((I,J)) \equiv I\times J\) and for arbitrary \((I,J) \in Y\), if
{(I_n, J_n)} \rightarrow (I, J), \text{ then } \{I_n \ast J_n\} \rightarrow I \ast J \text{ in } \{J, \rho\}. \text{ Therefore the interval arithmetic operations are continuous.}

Since the rational interval functions may at most consist of a finite number of interval arithmetic operations, by repeated use of the continuity of the composition of the continuous interval arithmetic operations (8a, p. 88), the rational interval functions are continuous and hence belong to \{\mathcal{F}, \mu\}.

**Proposition 2.9:** For any rational interval function \( f \in \{\mathcal{F}, \mu\} \) and arbitrary \( J \in \mathcal{J}_I \),

\[
f(J) \supset \bar{f}(J) \equiv \bigcup_{x \in J} f([x, x]) .
\]

**Proof:** The result is inherently obvious from the monotonic inclusion property (2.6) of the interval arithmetic operations and the definition of the rational interval function. Since a finite number of these operations is involved and since for every \( x \in J, [x, x] \subset J, [x, x] \in \mathcal{J}_I \), then \( f(J) \supset f([x, x]) \).

\[\text{If } \ast \text{ indicates division and } 0 \text{ belongs to any of the intervals in the set } \{J, J_n, n=1, 2, \ldots\} \subset \mathcal{J}, \text{ the operation is not defined and will therefore be considered as an exception. Since it is assumed that the rational interval function is defined, this precludes the occurrence of an exception.} \]
To prove that equality may not be achieved, it is sufficient to demonstrate that the set inclusion may not go the other direction.

Let 
\[ J = [-\frac{1}{2}, 1] , \quad f(J) = J^2 = J \cdot J \, . \]

Then
\[
f(J) = f([-\frac{1}{2}, 1]) = [-\frac{1}{2}, 1] \not\subseteq f(J) = \bar{f}([-\frac{1}{2}, 1]) = [0, 1] \, .
\]

**Corollary 2.9:** For any rational interval function \( f \in \mathcal{F}_\mathcal{I} \) and arbitrary \( J \in \mathcal{I}_\mathcal{I} \), if \( K \in \mathcal{I}_\mathcal{I} \) and \( J \supset K \), then \( f(J) \supset f(K) \supset \bar{f}(K) \).

**Proof:** By Proposition 2.9, \( f(K) \supset \bar{f}(K) \). Arguing the definition of the rational interval function and the monotonic inclusion property of the interval arithmetic (2.6), \( f(J) \supset f(K) \). Equality may be precluded by demonstrating that for \( J = [-\frac{1}{2}, 1] \supset K = [-\frac{1}{4}, 1] \) and \( f(J) = J^2 = J \cdot J \),
\[
f(J) = f([-\frac{1}{2}, 1]) = [-\frac{1}{2}, 1] \not\subseteq f(K) = f([-\frac{1}{4}, 1]) = [-\frac{1}{4}, 1] .
\]

**Proposition 2.10:** Let \( \{f_n\} \to f \) be an arbitrary Cauchy sequence in \( \mathcal{F}_\mathcal{I} \). Then \( \{\bar{f}_n\} \) is a Cauchy sequence in \( \mathcal{F}_\mathcal{I} \) converging uniformly to \( \bar{f} \in \mathcal{F}_\mathcal{I} \).
Proof: Since the sequence \( \{f_n\} \) is a Cauchy sequence, given \( \varepsilon > 0 \), there exists \( N(\varepsilon) \) such that for all \( n, m \geq N(\varepsilon) \),
\[
\mu(f_n, f_m) < \varepsilon.
\]

But
\[
\mu(f_n, f_m) = \sup_{J \in \mathcal{J}_I} \{ \rho(f_n(J), f_m(J)) \} < \varepsilon.
\]

Hence for any \( J \in \mathcal{J}_I \), whenever \( n, m \geq N(\varepsilon) \),
\[
\rho(f_n(J), f_m(J)) < \varepsilon.
\]

Now each \( \overline{f}_n \) is a continuous function in \( \{\mathcal{F}, \mu\} \) (Proposition 2.7). For each \( J \in \mathcal{J}_I \) and for each \( n \),
\[
\overline{f}_n(J) = \bigcup_{x \in J} f_n([x, x]) = [p_n(J), q_n(J)]
\]
\[
\equiv [\inf_{x \in J} f_n^L([x, x]), \sup_{x \in J} f_n^R([x, x])],
\]
where
\[
f_n([x, x]) = [f_n^L([x, x]), f_n^R([x, x])].
\]

Consider then for all \( n, m \geq N(\varepsilon) \) and for any \( J \in \mathcal{J}_I \),
\[
\rho(\overline{f}_n(J), \overline{f}_m(J)) = \max\{ |\inf_{x \in J} f_n^L([x, x]) - \inf_{x \in J} f_m^L([x, x])| , \sup_{x \in J} f_n^R([x, x]) - \sup_{x \in J} f_m^R([x, x])| \}.
\]

Since \( \{[x, x] | x \in J\} \subset \mathcal{J}_I \), then for every element \( [x, x] \) of this set (by virtue of \( \mu(f_n, f_m) < \varepsilon \)), it is true that
\[
\rho(f_n([x,x]), f_m([x,x])) = \max\{|f_n^L([x,x]) - f_m^L([x,x])|, |f_n^R([x,x]) - f_m^R([x,x])|\} < \varepsilon.
\]

Thus, with respect to the set \{[x,x]|x \in \mathcal{J}\} and for any \(J \in \mathcal{J}\), the pairs of real numbers \{\(f_n^L([x,x]), f_m^L([x,x])\)\} and \{\(f_n^R([x,x]), f_m^R([x,x])\)\} are individually never separated by a distance greater than \(\varepsilon\). Thus it must be that \(\rho(\bar{f}_n(J), \bar{f}_m(J)) < \varepsilon\) for any \(J \in \mathcal{J}\) whenever \(n, m \geq N(\varepsilon)\).

Therefore \{\(\bar{f}_n\)\} is a Cauchy sequence in \(\mathcal{J}, \mu\) and since the space is complete, the sequence converges to an element of the space. (Note that this convergence is true pointwise for every \(J \in \mathcal{J}\) and that the sequence also converges uniformly.) Denote this element \(\bar{g}\) and suppose that \(\bar{g} \neq \bar{f}\).

Then for at least one \(\varepsilon_1 > 0\), one \(J \in \mathcal{J}\) and any \(N_1\), for an infinite number of \(n \geq N_1\) \(\rho(\bar{f}_n(J), \bar{f}(J)) > \varepsilon_1\) and this occurs as \(\{f_n\} \to \bar{f}\) uniformly on \(\mathcal{J}\).

But for \(\varepsilon_1 > 0\), there exists an \(N_2(\varepsilon_1)\) such that if \(n, m \geq N_2(\varepsilon_1)\), \(\mu(f_n, f_m) < \varepsilon_1\). Thus, letting \(m \to \infty\), \(\mu(f_n, f) < \varepsilon_1\). But this implies that for each \(J \in \mathcal{J}\), \(\rho(\bar{f}_n(J), \bar{f}(J)) < \varepsilon_1\) and this is a contradiction. Hence it must be that \(\bar{g}\) is arbitrarily close to \(\bar{f}\) and therefore \(\bar{g} \equiv \bar{f}\).
Proposition 2.11: Let \( \{f_n\} \to f \) be an arbitrary Cauchy sequence of rational interval functions in \( \{F, \mu\} \). Then for each \( J \in J_I \), \( f(J) \supset \bar{f}(J) \).

Proof: By Proposition 2.10, \( \{\bar{f}_n\} \) is a Cauchy sequence in \( \{F, \mu\} \) which converges uniformly to \( \bar{f} \in \{F, \mu\} \). By Proposition 2.9, for each \( J \in J_I \) and for each \( n \), \( f_n(J) \supset \bar{f}_n(J) \).

For any \( J \in J_I \), in the notation of (2.10) and Proposition 2.6, let

\[
\begin{align*}
  f_n(J) &\equiv [f^L_n(J), f^R_n(J)], & f(J) &\equiv [f^L(J), f^R(J)] , \\
  \bar{f}_n(J) &\equiv [p_n(J), q_n(J)], & \bar{f}(J) &\equiv [p(J), q(J)] .
\end{align*}
\]

By Proposition 2.9 then, for every \( n \),

\[
f^L_n(J) \leq p_n(J) \leq q_n(J) \leq f^R_n(J) .
\]

Suppose \( f(J) \not\supset \bar{f}(J) \). Then either \( p(J) < f^L(J) \), \( f^R(J) < q(J) \) or both occur. Suppose \( p(J) < f^L(J) \). Then in fact there is an \( \varepsilon > 0 \) such that \( p(J) + 3\varepsilon = f^L(J) \).

But the sequences \( \{p_n(J)\} \) and \( \{f^L_n(J)\} \) are each convergent sequences in the reals since the sequences \( \{f_n\} \) and \( \{f_n\} \) are Cauchy sequences in \( \{F, \mu\} \) and for each \( n \),

\( f^L_n(J) \leq p_n(J) \), thus the contrary assumption above cannot occur and \( f(J) \leq p(J) \). Similarly \( q(J) \leq f^R(J) \) and both of these relations hold for each \( J \in J_I \).

Thus \( f(J) \supset \bar{f}(J) \) for every \( J \in J_I \).
Corollary 2.11: If \( \{f_n\} \to f \) is an arbitrary Cauchy sequence of rational interval functions in \( \{S, \mu\} \), then for each \( J \in J_I \), if \( K \in J_I \) and \( J \supset K \),

\[
 f(J) \supset f(K) \supset \tilde{f}(K) .
\]

Proof: The proof is obvious. \( f(K) \supset \tilde{f}(K) \) is the result of Proposition 2.11. Employing Corollary 2.9, argue the balance of the corollary in a manner identical to the proof of Proposition 2.11.

Proposition 2.12: Let \( f \in \{S, \mu\} \) have the property that for any \( J \in J_I \), \( f(J) \supset \tilde{f}(J) \). Let \( J = [d_1, d_{n+1}] \) and \( J_i \equiv [d_i, d_{i+1}] \in J_I \), \( d_i \leq d_{i+1}, \ i=1, \ldots, n \). Then,

\[
 \bigcup_{i=1}^n f(J_i) \supset \tilde{f}(J) \quad \text{and} \quad \bigcup_{i=1}^n f(J_i) \in J .
\]

Proof: By assumption,

\[
 \bigcup_{i=1}^n f(J_i) \supset \bigcup_{i=1}^n \tilde{f}(J_i) = \bigcup_{i=1}^n (\bigcup_{x \in J_i} f([x, x])) .
\]

But \( f \in \{S, \mu\} \) and therefore

\[
 \bigcup_{i=1}^n (\bigcup_{x \in J_i} f([x, x])) = \bigcup_{x \in J} f([x, x]) = \tilde{f}(J) .
\]

Also, since \( f(J_i) \in J \) and \( f(J_i) \supset \tilde{f}(J_i) \in J \), \( i=1, \ldots, n \) and
\[
\bar{f}(J) = \bigcup_{i=1}^{n} \bar{f}(J_i) \in \mathcal{F},
\]
then obviously
\[
\bigcup_{i=1}^{n} f(J_i) \in \mathcal{F}.
\]

**Proposition 2.13:** Let \( \{f_k\} \to f \) be any Cauchy sequence of rational interval functions in \( \mathcal{F}, \mu \). For any \( J \in \mathcal{I}_1 \), let
\[
J = [d_1, d_{n+1}] \quad \text{and} \quad J_i = [d_i, d_{i+1}] \in \mathcal{I}_1, \quad d_i \leq d_{i+1}, \quad i=1, \ldots, n.
\]
Then
\[
f_k(J) \supset \bigcup_{i=1}^{n} f_k(J_i) \supset \bar{f}_k(J), \quad k=1,2,\ldots
\]
and this relation converges to the relation
\[
f(J) \supset \bigcup_{i=1}^{n} f(J_i) \supset \bar{f}(J).
\]

**Proof:** By Corollary 2.9, for each \( i=1, \ldots, n \), since
\[
J \supset J_i,
\]

\[
f_k(J) \supset f_k(J_i) \supset \bar{f}_k(J_i), \quad k=1,2,\ldots
\]

By Proposition 2.12 then,
\[
f_k(J) \supset \bigcup_{i=1}^{n} f_k(J_i) \supset \bar{f}_k(J), \quad k=1,2,\ldots
\]
and this is true for any \( J \in \mathcal{I}_1 \) and any \( n \) in the assumed
method of partitioning of \(J\).

By Corollary 2.11, for each \(i=1, \ldots, n\), since \(J \supset J_i\),
\[
f(J) \supset f(J_i) \supset \bar{f}(J_i).
\]

By Proposition 2.12 again,
\[
f(J) \supset \bigcup_{i=1}^{n} f(J_i) \supset \bar{f}(J).
\]

**Theorem 2.14:** Let \(\{f_k\} \to f\) be any Cauchy sequence of rational interval functions in \(F\). For any \(J \in \mathcal{J}\), let \(J = [c, d]\) and

\[
J_i^n = \left[\frac{(n-i+1)c + (i-1)d}{n}, \frac{(n-i)c + id}{n}\right], \quad i=1, \ldots, n \quad \text{and} \quad n=1,2, \ldots.
\]

Then
\[
f(J) \supset \lim_{n \to \infty} \bigcup_{i=1}^{n} f(J_i^n) = \bar{f}(J).
\]

**Proof:** Since for any \(J \in \mathcal{J}\) and for any \(n\), Proposition 2.13 provides that
\[
f(J) \supset \bigcup_{i=1}^{n} f(J_i^n) \supset \bar{f}(J),
\]

it is obvious that
\[
f(J) \supset \lim_{n \to \infty} \bigcup_{i=1}^{n} f(J_i^n) \supset \bar{f}(J).
\]

For arbitrary \(\varepsilon > 0\), denote the closed \(\varepsilon\)-sphere in \(\mathcal{J}\)
by \( S(\bar{f}(J), \varepsilon) \equiv \{ K \in J | \rho(f(J), K) \leq \varepsilon \} \).

Let

\[
\bar{f}_\varepsilon(J) \equiv \{ K \in S(\bar{f}(J), \varepsilon) | t \in K \text{ for every } t \in S(\bar{f}(J), \varepsilon) \}.
\]

This is a unique element in \( J \), since \( S(\bar{f}(J), \varepsilon) \) is a closed and bounded collection of closed intervals in \( J \) and as such, each of its members must be a subset of some maximal closed interval which belongs to the collection, namely \( \bar{f}_\varepsilon(J) \).

Thus \( \bar{f}_\varepsilon(J) \) has been selected so that

\[
\rho(\bar{f}_\varepsilon(J), \bar{f}(J)) \equiv \varepsilon, \quad \bar{f}_\varepsilon(J) \supset \bar{f}(J) \quad \text{and} \quad \bar{f}_\varepsilon(J) \supset t
\]

for every \( t \in S(\bar{f}(J), \varepsilon) \).

But it is now possible to claim that there is a number \( N(\varepsilon) \) such that if \( n \geq N(\varepsilon) \), then

\[
\bar{f}_\varepsilon(J) \supset \bigcup_{i=1}^{n} f(J^n_i) \supset \bar{f}(J) \quad \text{and} \quad \rho\left( \bigcup_{i=1}^{n} f(J^n_i) ; \bar{f}(J) \right) < \varepsilon.
\]

For by the uniform continuity of \( f \) on \( J \), it is possible to select \( N(\varepsilon) \) so that if \( n \geq N(\varepsilon) \), \( J \) has been partitioned into \( J^n_i \)'s such that for each \( i \) and for every \( x \in J^n_i \),

\[
\rho(J^n_i, [x, x]) \leq \frac{d - \varepsilon}{N(\varepsilon)} < \delta(\varepsilon)
\]

and this implies that for each \( i \) and every \( x \in J^n_i \),
\[ \rho(f(J^n_i), f([x,x])) < \varepsilon \].

But the immediate consequence of this produces the second part of the claim,

\[ \rho\left( \bigcup_{i=1}^{n} f(J^n_i), \bar{f}(J) \right) < \varepsilon. \]

Since

\[ \rho(\bar{f}_\varepsilon(J), \bar{f}(J)) = \varepsilon, \quad \bar{f}_\varepsilon(J) \supset \bar{f}(J), \]

\[ \rho\left( \bigcup_{i=1}^{n} f(J^n_i), \bar{f}(J) \right) < \varepsilon, \quad \bigcup_{i=1}^{n} f(J^n_i) \supset \bar{f}(J) \text{ and } \bar{f}_\varepsilon(J) \supset t \]

for every \( t \in S(\bar{f}(J), \varepsilon) \), complete the claim obtaining

\[ \bar{f}_\varepsilon(J) \supset \bigcup_{i=1}^{n} f(J^n_i) \supset \bar{f}(J). \]

Thus it is always possible to select \( n \) sufficiently large that no matter how small the positive number \( \varepsilon \),

\[ \bigcup_{i=1}^{n} f(J^n_i) \text{ is within } \varepsilon \text{ of } \bar{f}(J) \]

and this is possible for each \( J \in \mathcal{J} \).

In other words, Theorem 2.14 implies that by using a sufficiently large number in the specified partition of \( J \) and computing the union of the interval function over the partition subintervals, it is possible to approximate the exact range of the interval function for \( x \in J, \bar{f}(J) \), as closely as desired.
While the fundamental or state transition matrix for initial-value problems in linear first-order systems of ordinary differential equations is defined by the matrix exponential, in the case where the system constant coefficient matrix linearly depends on a perturbation parameter (in an interval), except in a very few special cases (11, p. 25; 12, p. 174) it is impossible to express the perturbed matrix exponential in a form suitable for the analytic or numerical determination of the closed interval range of values for each of its elements for all values of the perturbation parameter. The interval analysis techniques can be applied toward accomplishing this goal and subsequently a method embracing the convergence philosophy of Theorem 2.14 for the matrix case will be developed for this purpose.

Once such an "interval fundamental matrix" is available, it is not difficult to recognize that post interval multiplication by the interval vector initial condition in the homogeneous problem will yield an interval vector bounding solution or envelope which contains the set of all solutions associated with the parameter variation. While such an interval bounding solution method would be unduly conservative in the matrix case, this does not occur at the beginning "scalar" problem level of discussion here and in the interest of introductory simplicity such considerations
will be postponed.

In the "scalar" case, the partial sums in the interval infinite series representation of the interval exponential function certainly define a sequence of rational interval functions. Assuming for the moment that this sequence is Cauchy, then in the exact interval arithmetic Theorem 2.14 provides a philosophy for evaluating an interval result which contains the actual closed interval range of values for this "fundamental matrix" for all values of the perturbation parameter and does so as closely as desired. The interval product of this result and the interval initial condition then produces an interval bounding solution or envelope which contains the set of all solutions associated with the parameter variation and similarly does so with the same degree of accuracy.

An interval exponential computation technique

The following development presents a technique by which the interval exponential function may be approximated by an "augmented" truncated series representation that set theoretically includes the infinite series result, with prescribed relative error bounds for the interval result endpoints.

Let \( \{f_n\} \) be the sequence of rational interval functions defined by
\[ f_n(J) = \sum_{j=0}^{n} \frac{J^j}{j!}, \quad J = [c, d] \in \mathcal{J}_1. \]  

(2.13)

Now, for any \( n \) and \( k \),

\[ \rho \left( f_n(J), f_{n+k}(J) \right) \leq \rho \left( f_n(J), f_{n+1}(J) \right) + \ldots \]

\[ + \rho \left( f_{n+k-1}(J), f_{n+k}(J) \right). \]

It is similarly obvious that

\[ \rho \left( \emptyset, \frac{J^{n+1}}{(n+1)!} \right) + \ldots + \rho \left( \emptyset, \frac{J^{n+k}}{(n+k)!} \right). \]

Let \( \emptyset \equiv [0,0] \). From the definitions of the metric \( \rho \) (2.7) and the interval arithmetic operation of addition (2.3) then,

\[ \rho \left( f_n(J), f_{n+k}(J) \right) \leq \rho \left( \emptyset, \frac{J^{n+1}}{(n+1)!} \right) + \ldots + \rho \left( \emptyset, \frac{J^{n+k}}{(n+k)!} \right). \]

For convenience, define \( |J| \equiv \rho (\emptyset, J) \equiv \max(|c|, |d|) \). This is called the "magnitude" of the interval \( J \) (2, p. 7).

Thus

\[ \rho \left( f_n(J), f_{n+k}(J) \right) \leq \frac{|J|^{n+1}}{(n+1)!} + \ldots + \frac{|J|^{n+k}}{(n+k)!} \]

\[ = \frac{|J|^{n+1}}{(n+1)!} \left( 1 + \frac{|J|}{n+2} + \ldots + \frac{|J|^{k-1}}{(n+k) \cdot \ldots \cdot (n+2)} \right) \]

\[ \quad \text{[For simplicity in notation it will be assumed here that} \]

\[ j! \text{ denotes the degenerate interval } [j!, j!] \] and that the zeroth power of any interval is the degenerate interval \([1,1]\). \]
\[ \frac{|J|^{n+1}}{(n+1)!} \left( 1 + \frac{|J|}{n+2} + \cdots + \left( \frac{|J|}{n+2} \right)^{k-1} \right) \]
\[ = \frac{|J|^{n+1}}{(n+1)!} \cdot \frac{1 - \left( \frac{|J|}{n+2} \right)^{k}}{1 - \left( \frac{|J|}{n+2} \right)} \leq \frac{|J|^{n+1}}{(n+1)!} \cdot \frac{1}{1 - \left( \frac{|J|}{n+2} \right)} , \]

where it is assumed that \( n \) is sufficiently large that
\[ \frac{|J|}{n+2} < 1 . \]

Thus
\[ \rho(f_n(J), f_{n+k}(J)) \leq \frac{|J|^{n+1}}{(n+1)!} \cdot \frac{1}{1 - \left( \frac{|J|}{n+2} \right)} . \]

But this result can be made independent of \( J \) in the sense that \( J \in J_I \), \( J_I \) is compact and for every \( J \in J_I \), \( |J| \leq |I| \). Thus for any \( \varepsilon > 0 \) it is possible to select \( N \) sufficiently large that for every \( J \in J_I \) and for all \( n, m \geq N \),
\[ \rho(f_n(J), f_m(J)) \leq \frac{|I|^{N+1}}{(N+1)!} \cdot \frac{1}{1 - \left( \frac{|I|}{N+2} \right)} < \varepsilon . \]

\footnote{This can be seen in the following way. Note that from Stirling's formula (13, p. 384), that
\[ \frac{n!}{(\frac{n}{e})^n \sqrt{2\pi n}} \]
is a monotonically nonincreasing positive sequence with limit equal to one. Thus
\[ (\frac{n}{e})^n \sqrt{2\pi n} \leq n! . \]
Therefore}
Therefore \( \{f_n\} \) is Cauchy in \( \{F, \mu\} \) and converges to an element of \( F \). Denote this function

\[
f(J) \equiv \sum_{j=0}^{\infty} \frac{J^j}{j!} \equiv e^J.
\]

Note that \( \{f_n\} \) is a Cauchy sequence of rational interval functions and by Proposition 2.11,

\[
f(J) \supset f(J) \equiv \bigcup_{x \in J} f([x,x]) = \bigcup_{x \in J} e^x,
\]

since \( f([x,x]) \equiv e^x \).

For the present, assume that the interval arithmetic operations can be exactly computed in the reals (that is, Equations 2.4 are calculated on an "infinite-decimal" machine) and that it is desired to determine an interval result which will contain the value of the interval function

\[
f(J) \equiv \sum_{j=0}^{\infty} \frac{J^j}{j!} \equiv e^J.
\]

\[
\lim_{N \to \infty} \frac{|I|^{N+1}}{(N+1)!} = \lim_{N \to \infty} \frac{|I|^N}{N!} \leq \lim_{N \to \infty} \frac{|I|^N}{(\frac{N}{e})^N \sqrt{2\pi N}}
\]

\[
= \lim_{N \to \infty} \frac{1}{N} \left( \frac{N}{e|I|} \right)^N \sqrt{2\pi N} = 0.
\]

So it is obvious that by selecting \( N \) sufficiently large it is possible to insure that

\[
\frac{|I|^{N+1}}{(N+1)!} \frac{1}{1-\frac{|I|}{N+2}} < \epsilon.
\]
within some predetermined relative error.

Let the interval function be defined by

$$f(J) \equiv e^J = [f^L(J), f^R(J)]$$

and let the computable truncated interval series be defined by

$$f_n(J) \equiv \sum_{i=0}^{n} \frac{J^i}{i!} = [f^L_n(J), f^R_n(J)] .$$

Let the remainder of the interval series be defined by

$$r_n(J) \equiv \sum_{i=n+1}^{\infty} \frac{J^i}{i!} = [r^L_n(J), r^R_n(J)] .$$

It has been shown previously that there exists an upper bound on the metric measure of how closely the truncated series approximates the actual function,

$$\rho(f_n(J), f(J)) = \rho(\emptyset, r_n(J)) \equiv |r_n(J)| \leq \frac{|J|^{n+1}}{(n+1)!} \cdot \frac{1}{1 - \left|\frac{J}{n+2}\right|} , \quad (2.14)$$

provided $\left|\frac{J}{n+2}\right| < 1$. Since the truncated series is computable, assume by appropriate programming techniques that it is possible to satisfy the relation

$$|r_n(J)| \equiv \max(|r^L_n(J)|, |r^R_n(J)|) \leq \delta \leq 10^{-p}$$

$$\cdot \min\{|f^L_n(J)|, |f^R_n(J)|\} , \quad (2.15)$$
where \( P \) is a positive integer and then

\[
0 < |r_n^L(j)| \leq 0 < |r_n^L(j)| < 10^{-P} |f_n^L(j)| .
\] (2.16)

Since

\[
f(j) = f_n(j) + r_n(j) ,
\]

\[
f_n^L(j) = f_n^L(j) + r_n^L(j) \geq f_n^L(j) - |r_n^L(j)| .
\]

Combining the above two inequalities,

\[
f_n^L(j) \geq f_n^L(j) - \delta \geq f_n^L(j) - 10^{-P} |f_n^L(j)| .
\]

Taking the negative of this inequality and adding \( f_n^L(j) \),

\[
0 \leq f_n^L(j) - (f_n^L(j) - \delta) \leq f_n^L(j) - (f_n^L(j) - 10^{-P} |f_n^L(j)| ) .
\]

Now \( f_n^L(j) - (f_n^L(j) - \delta) \) is the error in the approximation endpoint \( (f_n^L(j) - \delta) \) and it is bounded above, since

\[
0 \leq |f_n^L(j) - (f_n^L(j) - \delta)| \leq |f_n^L(j) - f_n^L(j)| + 10^{-P} |f_n^L(j)| .
\]

Assuming

\[
|f_n^L(j)| \geq |f_n^L(j) - f_n^L(j)| - |f_n^L(j)| > 0 ,
\]

then

\[
\frac{|f_n^L(j) - (f_n^L(j) - \delta)|}{|f_n^L(j)|} \leq \frac{|f_n^L(j) - f_n^L(j)| + 10^{-P} |f_n^L(j)|}{|f_n^L(j) - f_n^L(j)| - |f_n^L(j)|} .
\]
\[ \frac{|f^L(J) - f^L_n(J)|}{|f^L_n(J)|} = \frac{10^{-P}}{|1 - \frac{|f^L(J) - f^L_n(J)|}{|f^L_n(J)|}} + \frac{10^{-P}}{|1 - \frac{|f^L(J) - f^L_n(J)|}{|f^L_n(J)|}} . \]

But \( |f^L(J) - f^L_n(J)| \equiv |f^L_n(J)| \leq 10^{-P}|f^L_n(J)| \) by assumption (2.16) and therefore
\[ \frac{|f^L(J) - (f^L_n(J) - \xi)|}{|f^L(J)|} \leq 2\left( \frac{10^{-P}}{1 - 10^{-P}} \right) . \] (2.17)

It is similarly determined that
\[ \frac{|f^R_n(J) + \xi - f^R(J)|}{|f^R(J)|} \leq 2\left( \frac{10^{-P}}{1 - 10^{-P}} \right) . \] (2.18)

Thus by selecting \( n \) sufficiently large that the conditions
\[ \frac{|J|}{n+2} < 1 \quad \text{and} \quad \frac{|J|^{n+1}}{(n+1)!} \cdot \frac{1}{1 - \frac{|J|}{n+2}} \equiv \xi \leq 10^{-P} \cdot \min\{|f^L_n(J)|, |f^R_n(J)|\} \]

are satisfied, denoting
\[ \varepsilon \equiv 2\left( \frac{10^{-P}}{1 - 10^{-P}} \right) , \]

then
In other words by including a sufficient number of terms in the truncated computable series (2.13) so that the algorithmic inequality (2.15) is satisfied, it is possible to augment this truncated series result so that the augmented interval result bounds the actual interval exponential function and does so within the specified relative endpoint error bounds (2.17) and (2.18).

It should be remarked at this point that nothing has been said indicating how well $e^J$ approximates the corresponding united extension, $\bigcup_{x \in J} e^x$. Proposition 2.13 and Theorem 2.14 indicate the direction which will be followed in improving this approximation and the corresponding augmented result approximating $e^J$.

Continuous interval matrix functions and the convergence result

Since the primary interest is in the interval matrix exponential functions, a substantial portion of the previous

---

1Recall that $0 \leq f^L(J) - (f^L_n(J) - \mathcal{Y})$ and $0 \leq (f^R_n(J) + \mathcal{Y}) - f^R(J)$. Then (2.17) and (2.18) imply $f^L(J) - \varepsilon |f^L(J)| \leq f^L_n(J) - \mathcal{Y}$ and $f^R_n(J) + \mathcal{Y} \leq f^R(J) + \varepsilon |f^R(J)|$, which provides that $[1-\varepsilon, 1+\varepsilon] \cdot e^J \supset (f_n(J) + [-\mathcal{Y}, \mathcal{Y}])$. 

theory must now be extended toward that goal. (The ex-
tended propositions, corollaries and theorem which are re-
quired will be given the same corresponding number designa-
tion except that the letter M will be appended to indicate
the matrix case.)

Define

\[ J_n^2 \equiv \{ \text{set of all } n \times n \text{ matrices where each of its}
\]
\[ n^2 \text{ elements belong to } J \} . \]  

(2.19)

Denote elements of \( J_n^2 \) by matrix notation

\[ A \equiv (a_{ij}) \equiv ((a^{L}_{ij}, a^{R}_{ij})) , \quad i,j=1,2,...,n . \]

Let \( A,B,C \in J_n^2 \) and let \( u \) be a fixed positive real vector

\[ u = (u_1, u_2, \ldots, u_n) \in \mathbb{R}^n , \quad u_i > 0 , \quad i=1,\ldots,n . \]

Define

\[ \sigma(A,B) \equiv \max \left\{ \frac{1}{u_i} \sum_{j=1}^{n} u_j \rho(a_{ij}, b_{ij}) \right\} , \]  

(2.20)

where \( \rho \) is the metric of \( \{ J, \rho \} \). Then \( \sigma \) is a metric and
\( \{ J_n^2, \sigma \} \) is a metric space, since

(i) by the symmetry of \( \rho \), \( \sigma(A,B) = \sigma(B,A) \),

(ii) since \( \rho(a_{ij}, b_{ij}) = 0 \) if and only if \( a_{ij} = b_{ij} \),
\[ \sigma(A,B) = 0 \] if and only if \( A = B \) and

(iii) by the triangle axiom of \( \rho \),
\[ \sigma(A,B) \leq \sigma(A,C) + \sigma(C,B) . \]
More will be said about the fixed real vector $u$ in the next section.

**Proposition 2.2M:** The metric space $\mathcal{J}^n$, is complete.

**Proof:** Let $\{A_k\}$ be an arbitrary Cauchy sequence in $\mathcal{J}^n$. Since $u$ is a fixed positive vector, define the fixed constant

$$0 < Q \equiv \min_{i,j} \frac{u_i}{u_j} \leq 1.$$

For the Cauchy sequence $\{A_k\}$, given any $\varepsilon > 0$ and letting $\varepsilon_1 \equiv Q \cdot \varepsilon$, there exists an $N(\varepsilon_1)$ such that for all $k, \ell \geq N(\varepsilon_1)$,

$$\sigma(A_k, A_\ell) \equiv \max_{i,j} \left\{ \frac{1}{n} \sum_{j=1}^{n} u_j \rho(a_{ij_k}, a_{ij_\ell}) \right\} < \varepsilon_1.$$

But then for each $i,j$ and for all $k, \ell \geq N(\varepsilon_1)$,

$$\frac{u_j}{u_i} \rho(a_{ij_k}, a_{ij_\ell}) < \varepsilon_1 \quad \text{and} \quad \rho(a_{ij_k}, a_{ij_\ell}) < \varepsilon.$$

Since $\varepsilon > 0$ is arbitrary, for each $i,j$, $\{a_{ij_k}\}$ is a Cauchy sequence in $\mathcal{J}, \rho$ and since that metric space is complete,

$$\{a_{ij_k}\} \rightarrow a_{ij} \in \mathcal{J}.$$ 

Thus

$$\{A_k\} \rightarrow A \in \mathcal{J}^n.$$
and therefore \( \{ J^{n^2}, \sigma \} \) is complete since \( \{ A_k \} \) is an arbitrary Cauchy sequence in \( \{ J^{n^2}, \sigma \} \).

Since the subsequent interval matrix function propositions, corollaries and theorem can be proved in a manner similar to the previous corresponding "scalar" cases, the proofs of the remaining results will not be given here.

Let

\[
F^{n^2} = \{ F: J_I \rightarrow J^{n^2}, F \text{ continuous on } J_I \}.
\]

If \( F, G, H \in F^{n^2} \), let

\[
\xi(F, G) = \sup_{J \in J_I} \{ \sigma(F(J), G(J)) \}.
\]

If \( F, G, H \in F^{n^2} \), let

\[
\xi(F, G) = \sup_{J \in J_I} \{ \sigma(F(J), G(J)) \}.
\]

Then \( \xi \) is a metric, since

(i) by the symmetry of \( \sigma \),

\[
\xi(F, G) = \xi(G, F).
\]

(ii) since \( \sigma(F(J), G(J)) = 0 \) if and only if \( F(J) = G(J) \),

\[
\xi(F, G) = 0 \text{ if and only if } F = G \text{ and}
\]

(iii) \( \sup_{J \in J_I} \{ \sigma(F(J), G(J)) \} \leq \sup_{J \in J_I} \{ \sigma(F(J), H(J)) \}
\]

\[
+ \sigma(H(J), G(J))
\]

\[
\leq \sup_{J \in J_I} \{ \sigma(F(J), H(J)) \} + \sup_{K \in J_I} \{ \sigma(H(K), G(K)) \},
\]

that is, \( \xi(F, G) \leq \xi(F, H) + \xi(H, G) \).
Therefore \( \{ \mathcal{F}^n, \xi \} \) is the metric space of continuous interval matrix functions.

**Proposition 2.5M:** The metric space \( \{ \mathcal{F}^n, \xi \} \) is complete.

**Proposition 2.6M:** For arbitrary \( F \in \mathcal{F}^n \) and \( J \in \mathcal{J}_I \),

\[
F(J) \equiv \bigcup_{x \in J} F([x, x]) = ([p_{ij}(J), q_{ij}(J)]) \in \mathcal{F}^n,
\]

where for each \( i, j \) and

\[
F(J) \equiv ([f^L_{ij}(J), f^R_{ij}(J)]),
\]

\[
p_{ij}(J) \equiv \inf_{x \in J} f^L_{ij}([x, x])
\]

and

\[
q_{ij}(J) \equiv \sup_{x \in J} f^R_{ij}([x, x]) \quad (2.22)
\]

**Proposition 2.7M:** For arbitrary

\[
F \in \{ \mathcal{F}^n, \xi \}, \quad \overline{F} \in \{ \mathcal{F}^n, \xi \}.
\]

**Proposition 2.8M:** The rational interval matrix functions belong to

\[
\{ \mathcal{F}^n, \xi \}.
\]

**Proposition 2.9M:** For any rational interval matrix function \( F \in \{ \mathcal{F}^n, \xi \} \) and arbitrary \( J \in \mathcal{J}_I \),
\[ F(J) \supseteq \overline{F}(J) \equiv \bigcup_{x \in J} F([x,x]) \]

**Corollary 2.9M:** For any rational interval matrix function \( F \in \mathcal{I}^{n^2, \xi} \) and arbitrary \( J \in \mathcal{I}_x \), if \( K \in \mathcal{I}_x \) and \( J \supseteq K \), then
\[ F(J) \supseteq F(K) \supseteq \overline{F}(K) \]

**Proposition 2.10M:** Let \( \{ F_k \} \rightarrow F \) be an arbitrary Cauchy sequence in \( \{ \mathcal{I}^{n^2, \xi} \} \). Then \( \{ \overline{F}_k \} \) is a Cauchy sequence in \( \{ \mathcal{I}^{n^2, \xi} \} \) converging uniformly to \( F \in \mathcal{I}^{n^2, \xi} \).

**Proposition 2.11M:** Let \( \{ F_k \} \rightarrow F \) be an arbitrary Cauchy sequence of rational interval matrix functions in \( \{ \mathcal{I}^{n^2, \xi} \} \). Then for each \( J \in \mathcal{I}_x \), \( F(J) \supseteq \overline{F}(J) \).

**Corollary 2.11M:** If \( \{ F_k \} \rightarrow F \) is an arbitrary Cauchy sequence of rational interval matrix functions in \( \{ \mathcal{I}^{n^2, \xi} \} \), then for each \( J \in \mathcal{I}_x \), if \( K \in \mathcal{I}_x \) and \( J \supseteq K \),
\[ F(J) \supseteq F(K) \supseteq \overline{F}(K) \]

**Proposition 2.12M:** Let \( F \in \mathcal{I}^{n^2, \xi} \) have the property that for any \( J \in \mathcal{I}_x \), \( F(J) \supseteq \overline{F}(J) \). Let \( J \equiv [d_1, d_{m+1}] \) and
\[ J_i \equiv [d_i, d_{i+1}] \in \mathcal{I}_x, \quad d_i \leq d_{i+1}, \quad i = 1, \ldots, m. \] Then
\[ \bigcup_{i=1}^{m} F(J_i) \supseteq \overline{F}(J) \quad \text{and} \quad \bigcup_{i=1}^{m} F(J_i) \in \mathcal{I}^{n^2} \]
Proposition 2.13M: Let \( \{F_k\} \to F \) be any Cauchy sequence of rational interval matrix functions in \( \mathcal{F}^2 \). For any \( J \in \mathcal{J} \), let \( J \equiv [d_1,d_{m+1}] \) and \( J_i \equiv [d_i,d_{i+1}] \in \mathcal{J} \), \( d_i \leq d_{i+1} \), \( i=1,...,m \). Then

\[
F_k(J) \supset \bigcup_{i=1}^{m} F_k(J_i) \supset F_k(J) , \quad k=1,...
\]

and this relation converges to the relation

\[
F(J) \supset \bigcup_{i=1}^{m} F(J_i) \supset F(J) .
\]

Theorem 2.14M: Let \( \{F_k\} \to F \) be any Cauchy sequence of rational interval matrix functions in \( \mathcal{F}^2 \). For any \( J \in \mathcal{J} \), let \( J \equiv [c,d] \) and

\[
J_i^m \equiv \left[ \frac{(m-i+1)c + (i-1)d}{m} , \frac{(m-i)c + id}{m} \right] , \quad i=1,...,m
\]

and \( m=1,2,... \). Then

\[
F(J) \supset \lim_{m \to \infty} \bigcup_{i=1}^{m} F(J_i^m) = \overline{F}(J) .
\]

Since the partial sums in the interval infinite matrix series representation of the interval matrix exponential function define a sequence of rational interval matrix functions, if this sequence is Cauchy, Theorem 2.14M provides a convergence philosophy for evaluating an interval fundamental matrix which elementwise contains the actual
closed interval range of values for the perturbed fundamental matrix for all values of the perturbation parameter and does so as closely as desired.

While interval postmultiplication of this interval fundamental matrix by the interval vector initial condition in the homogeneous problem will certainly produce an interval vector solution bound or envelope for the set of all solutions associated with the parameter variation, it is not difficult to understand that this result will be unduly conservative.

This inconvenience may be recognized by observing that in forming the interval fundamental matrix, the union over the partition subintervals in Theorem 2.14M (or Proposition 2.13M) is elementwise independent and therefore the partition subinterval "signature" is lost within and between elements. Consequently, a less conservative interval vector bound solution for the set of all solutions associated with the perturbation parameter will be obtained if the union is performed before this "signature" is lost by taking the union over the interval vector solution bounds obtained for each partition subinterval. This in fact will be the technique that is used in the linear interval integration algorithm and since the interval vector initial condition is fixed, Theorem 2.14M provides the necessary convergence in this technique as well.
The interval matrix exponential computation technique

The following development extends the previous interval exponential function approximation technique to the matrix counterpart.

Let \( \{ F_k \} \) be the sequence of rational interval matrix functions defined by

\[
F_k(J) \equiv \left( (f_{ij}^l(J)), f_{ij}^R(J) \right) \equiv \left( \left[ f_{ij}^l(J), f_{ij}^R(J) \right] \right) \equiv \sum_{i=0}^{k} \frac{A_{ij}(J)}{i!},
\]

where

\[
A(J) \equiv \left( (a_{ij}(J)) \right) \equiv \left( \left[ a_{ij}^l(J), a_{ij}^R(J) \right] \right).
\]

(2.23)

\( A(J) \) is the rational interval matrix function defined by

\[
A(J) \equiv A_1 + JA_2, \quad J \in \mathcal{L}_I,
\]

where \( A_1, A_2 \in \mathcal{J} \mathbb{R}^2 \) are constant interval matrices.

Then, for any \( k \) and \( l \),

\[
\sigma(F_k(J), F_{k+l}(J)) \leq \sigma(F_k(J), F_{k+1}(J)) + \cdots + \sigma(F_{k+l-1}(J), F_{k+l}(J)).
\]

(2.24)

Let \( 0 \) denote the interval matrix where each element is a degenerate zero interval. From the definitions of the metric \( \sigma \) (2.20) and the interval arithmetic operation of addition (2.3) then,

\[
\sigma(F_k(J), F_{k+l}(J)) \leq \sigma(0, \frac{A_{k+1}(J)}{(k+1)!}) + \cdots + \sigma(0, \frac{A_{k+l}(J)}{(k+l)!})
\]

(2.24)
It is apparent that
\[ \sigma(0, \frac{A^m(J)}{m!}) = \frac{1}{m!} \sigma(0, A^m(J)) \] \[ \text{1} \]

But it is also possible to claim that
\[ \sigma(0, A^m(J)) \leq [\sigma(0, A(J))]^m \]

This may be obtained by first defining the real matrix
\[ |B| \] from the interval matrix \( B \in \mathbb{D}^{n \times n} \) by
\[ |B| \equiv \left( \left| b_{ij} \right| \right) \equiv \left( \left[ b_{ij}^L, b_{ij}^R \right] \right) \]
\[ \equiv \left( \max \left\{ \left| b_{ij}^L \right|, \left| b_{ij}^R \right| \right\} \right) \] \[ \text{(2.25)} \]

Let \( G \equiv \text{diag}(u) \), where \( u \) is the fixed positive real vector used in the \( \sigma \) metric definition (2.20). By the interval arithmetic (2.3) and by (2.25),
\[ |A^m(J)| \leq |A(J)|^m \] \[ \text{(2.26)} \]

Note that \( G^{-1}|A^m(J)|G \) and \( G^{-1}|A(J)|^mG \) are real \( n \times n \) matrices.

For the real \( n \times n \) matrix \( C = (c_{ij}) \), define the matrix norm (14, p. 99)
\[ \|C\|_\infty \equiv \max_i \sum_{j=1}^n |c_{ij}| \] \[ \text{(2.27)} \]

From (2.25), (2.26) and the definition of the metric \( \sigma \) (2.20),

\[ 1 \text{Recall that } m! \text{ represents the degenerate interval } [m!, m!]. \text{ Also assume that the zero-th power of any interval matrix is the degenerate interval identity matrix.} \]
\[ \sigma(O, A^m(J)) \equiv \sigma(O, |A^m(J)|) \leq \sigma(O, |A(J)|^m) \quad (2.28) \]

But
\[
\sigma(O, |A(J)|^m) \equiv \|G^{-1}|A(J)|^m_G\|_\infty \\
= \|G^{-1}|A(J)|_{(1)}G^{-1}|A(J)|_{(2)} \cdots \\
\leq (\|G^{-1}|A(J)|G\|_\infty)^m, \quad (2.29)
\]

by the properties of the matrix norm (14, p. 107). Also
\[
\|G^{-1}|A(J)|G\|_\infty \equiv \sigma(O, A(J)) \quad (2.30)
\]

Combining (2.28), (2.29) and (2.30),

\[ \sigma(O, A^m(J)) \leq [\sigma(O, A(J))]^m \quad (2.31) \]

which was the claim to be proved.

For convenience define
\[ \lambda \equiv \sigma(O, A(J)) \quad . \]

Then combining (2.24) and (2.31),

\[
\sigma(F_k(J), F_{k+\ell}(J)) \leq \frac{\lambda^{k+1}}{(k+1)!} + \cdots + \frac{\lambda^{k+\ell}}{(k+\ell)!} \\
= \frac{\lambda^{k+1}}{(k+1)!} (1 + \frac{\lambda}{k+2} + \cdots + \frac{\lambda^{\ell-1}}{(k+2) \cdots (k+\ell)}) \\
\leq \frac{\lambda^{k+1}}{(k+1)!} (1 + \frac{\lambda}{k+2} + \cdots + \frac{\lambda^{\ell-1}}{(k+2)})
\]
provided that \( \frac{\lambda}{k+2} < 1 \). But \( J \in J_I \) and \( |J| \leq |I| \), hence
\[ |A(J)| \leq |A(I)| \]
in the sense that for each \( i,j \),
\[ |a_{ij}(J)| \leq |a_{ij}(I)| \]. Then
\[ \lambda \equiv \sigma(0,A(J)) \leq \bar{\lambda} \equiv \sigma(0,A(I)) \]
and \( \bar{\lambda} \) is independent of \( J \). For arbitrary \( \varepsilon > 0 \) it is obvious then that there exists \( N \) such that
\[ \sigma(F_k(J), F_{k+\ell}(J)) \leq \frac{\lambda^{N+1}}{(N+1)!} \cdot \frac{1}{1 - \left(\frac{\lambda}{N+2}\right)} \equiv \bar{\varepsilon} < \varepsilon \]
for all \( J \in J_I \), for all \( k \geq N \) and for all \( \ell = 1,2,\ldots \).

Therefore by the definition of the metric \( \xi \) (2.21),
\[ \xi(F_k,F_{k+\ell}) \leq \bar{\varepsilon} < \varepsilon \]
for all \( k \geq N \), \( \ell = 1,2,\ldots \),
hence \( \{F_k\} \) is Cauchy in the complete metric space
\[ \{\mathcal{F}^n, \xi\}, \{F_k\} \to F \in \mathcal{F}^n, \xi \]
and
\[ F(J) \equiv \sum_{i=0}^{\infty} \frac{A^i(J)}{i!} \equiv e^{A(J)} \varepsilon \mathcal{F}^n \]
for all \( J \in J_I \).

Then by Proposition 2.11M, for any \( J \in J_I \),
\[ F(J) \equiv e^A(J) \supset \bar{F}(J) \equiv \bigcup_{x \in J} F([x,x]) \equiv \bigcup_{x \in J} e^A([x,x]) \]

Proceeding as in the "scalar" interval exponential case, assume that the interval arithmetic operations are calculated on the infinite-decimal machine.

Let the actual interval matrix exponential function be represented as

\[ F(J) \equiv ([f_{ij}(J)]) \equiv ([f_{ij}^L(J), f_{ij}^R(J)]) \equiv e^A(J) \quad (2.33) \]

and let the computable truncated interval matrix series \( F_k(J) \) be defined by (2.23). Define the remainder of the interval matrix infinite series by

\[ R_k(J) \equiv ([r_{ij,k}(J)]) \equiv ([r_{ij,k}^L(J), r_{ij,k}^R(J)]) \]

\[ \equiv \sum_{i=k+1}^{\infty} \frac{A^i(J)}{i!} \quad (2.34) \]

From the previous arguments then, there exists an upper bound on the metric measure \( \sigma \) of how closely the truncated series approximates the actual function,

\[ \sigma(F_k(J), F(J)) \equiv \sigma(O, R_k(J)) \leq \frac{\lambda^{k+1}}{(k+1)!} \cdot \frac{1}{1 - \left(\frac{\lambda}{k+2}\right)} \equiv \tilde{\sigma}, \]

(2.35)

provided \( \frac{\lambda}{k+2} < 1 \).
From the definition of the metric $\sigma$ (2.20), note that

\[
\frac{u_{ij}}{u_i} |r_{ij_k}^L(J)| \equiv \left(\frac{u_{ij}}{u_i}\right) \max \{|r_{ij_k}^L(J)|, |r_{ij_k}^R(J)|\}
\]

\[
\leq \sigma(o, R_k(J)) \leq \frac{u_i}{u_j} \mathcal{F}.
\] (2.36)

Assume then that for the computable truncated series it is possible by appropriate programming techniques to satisfy the relation

\[
|r_{ij_k}^L(J)| \leq \frac{u_{ij}}{u_j} \cdot \sigma(o, R_k(J)) \leq \frac{u_i}{u_j} \mathcal{F}
\]

\[
\leq 10^{-p} \min\{|f_{ij_k}^L(J)|, |f_{ij_k}^R(J)|\}, \quad (2.37)
\]

for every $i,j$ and then

\[
0 < |r_{ij_k}^L(J)| \leq \frac{u_{ij}}{u_j} \mathcal{F} \leq 10^{-p} |f_{ij_k}^L(J)|.
\] (2.38)

Also for each $i,j$

\[
f_{ij}(J) \equiv f_{ij_k}(J) + r_{ij_k}(J)
\]

and therefore

\[
f_{ij}^L(J) \equiv f_{ij_k}^L(J) + r_{ij_k}^L(J) \geq f_{ij_k}^L(J) - |r_{ij_k}^L(J)|.
\] (2.39)

Combining (2.38) and (2.39),

\[
f_{ij}^L(J) \geq f_{ij_k}^L(J) - \left(\frac{u_{ij}}{u_j}\right) \mathcal{F} \geq f_{ij_k}^L(J) - 10^{-p} |f_{ij_k}^L(J)|
\]
and taking the negative of this inequality and adding $f_{ij}^L(J)$,

$$0 \leq f_{ij}^L(J) - (f_{ijk}^L(J) - \frac{u_i}{u_j} J)$$

$$\leq f_{ij}^L(J) - (f_{ijk}^L(J) - 10^{-P} |f_{ijk}^L(J)|) . \quad (2.40)$$

Hence the error in the approximation endpoint

$$f_{ij}^L - \frac{u_i}{u_j} J$$

is bounded above by the inequality

$$0 \leq |f_{ij}^L(J) - (f_{ijk}^L(J) - \frac{u_i}{u_j} J)|$$

$$\leq |f_{ij}^L(J) - f_{ijk}^L(J)| + 10^{-P} |f_{ijk}^L(J)| . \quad (2.41)$$

Assuming

$$|f_{ij}^L(J)| \geq |f_{ij}^L(J) - f_{ijk}^L(J)| - |f_{ijk}^L(J)| > 0$$

and using (2.41), then

$$\frac{|f_{ij}^L(J) - (f_{ijk}^L(J) - \frac{u_i}{u_j} J)|}{|f_{ij}^L(J)|}$$

$$\leq \frac{|f_{ij}^L(J) - f_{ijk}^L(J)| + 10^{-P} |f_{ijk}^L(J)|}{||f_{ij}^L(J) - f_{ijk}^L(J)| - |f_{ijk}^L(J)|||}$$
\[ \left| f^L_{ij}(J) - f^L_{ij_k}(J) \right| = \frac{\left| f^L_{ij_k}(J) \right|}{\left| f^L_{ij}(J) - f^L_{ij_k}(J) \right|} + 10^{-p} \tag{2.42} \]

But by assumption (2.37) and then Equation (2.38),

\[ \left| f^L_{ij}(J) - f^L_{ij_k}(J) \right| \equiv \left| f^L_{ij_k}(J) \right| \leq 10^{-p} \left| f^L_{ij_k}(J) \right| \tag{2.43} \]

Combining (2.43) and (2.42)

\[ \frac{\left| f^L_{ij}(J) - (f^L_{ij_k}(J) - \frac{u_i}{u_j}s) \right|}{\left| f^L_{ij}(J) \right|} \leq 2 \left( \frac{10^{-p}}{1 - 10^{-p}} \right) \tag{2.44} \]

and this holds for each \( i, j \).

It is similarly determined that for each \( i, j \),

\[ \left| (f^R_{ij_k}(J) + \frac{u_i}{u_j}s) - f^R_{ij}(J) \right| \leq 2 \left( \frac{10^{-p}}{1 - 10^{-p}} \right) \tag{2.45} \]

As in the scalar case, by selecting \( k \) sufficiently large so that for all \( i, j \), \( \frac{\lambda}{k+2} < 1 \) and

\[ \frac{u_i}{u_j} \cdot \frac{\lambda^{k+1}}{(k+1)!} \cdot \frac{1}{1 - \frac{\lambda}{k+2}} \equiv \frac{u_i}{u_j}s \leq 10^{-p} \min\left\{ \left| f^L_{ij_k}(J) \right|, \left| f^R_{ij_k}(J) \right| \right\} \]
then for

$$\varepsilon = 2 \left( \frac{10^{-p}}{1 - 10^p} \right),$$

$$[1-\varepsilon,1+\varepsilon] \cdot e^{A(J)} \supset \sum_{i=0}^{k} \frac{A_i(J)}{i!} + Z \supset e^{A(J)} \supset \bigcup_{x \in J} e^{A([x,x])},$$

where

$$Z \equiv \left( \left[ -\frac{u_i}{u_j}, +\frac{u_i}{u_j} \right] \right).$$

In other words, by including a sufficient number of terms in the truncated computable series (2.23) so that the algorithmic inequality (2.37) is satisfied, it is possible to augment this truncated series so that the interval result bounds the actual interval matrix exponential function $e^{A(J)}$ and does so within the specified relative endpoint error bounds (2.44) and (2.45).

As in the scalar case, nothing has been said regarding how well $e^{A(J)}$ approximates the corresponding united extension

$$\bigcup_{x \in J} e^{A([x,x])}.$$

Proposition 2.13M and Theorem 2.14M indicate the direction which will be followed in improving this approximation and the corresponding augmented result approximating $e^{A(J)}$. 
Conservativeness, bounding interval arithmetic and IBM floating-point computation considerations

Before discussing machine bounding arithmetic, it is appropriate to consider the various representational techniques that may be employed to reduce the conservativeness of the rational interval function evaluations.

The subdistributivity property of interval arithmetic (2.5) points to one such technique. Consider the interval polynomial (a rational interval function) with interval coefficients \( A_i \in \mathcal{I} \), \( i=0,1,\ldots,n \) and interval variable \( J \in \mathcal{I} \).

The so-called "nested form" of the polynomial yields an interval result contained in and frequently narrower\(^1\) than that produced using the sum of powers,

\[
(\cdots (A_n J + A_{n-1}) J + \cdots + A_1) J + A_0 < A_n J^n + \cdots + A_1 J + A_0
\]

(2.46)

(This same property obviously holds for interval matrix polynomials.) The nested form of computations also is reasonable from the computer programmer's point of view of improving the speed of an algorithm by minimizing the number of calculations required to obtain a result or from the

---

\(^1\)Narrower is used in the sense that Moore (2, p. 7) defines the width of an interval as \( w(J) \equiv w([c,d]) \equiv d-c \geq 0 \).
numerical analyst's attempt to minimize the accumulation of local rounding errors (15, pp. 51, 302) by the same technique. This is in effect a reduction in the number of occurrences of the interval variable.\(^1\)

Another method of selecting a rational interval expression which may produce a less conservative interval result which contains the corresponding united extension interval result is called the "centered form" (2, p. 42).

Suppose it is desired to calculate the range of the rational function of a real variable \(f(x) = x-x^2\) for \(x \in J \equiv \left[\frac{1}{2} - r, \frac{1}{2} + r\right], \ r \geq 0\). If \(c\) denotes the center or midpoint of the interval \(J\), the real function representation desired for the centered form is \(f(x) = f(c) + g(x-c)\). Obviously, \(g(x-c) = -(x-\frac{1}{2})^2\) and the centered form of the interval arithmetic representation is \(\frac{1}{4} - (J - \frac{1}{2})^2\), where for simplicity of notation it will be the practice to denote

\(^1\)Moore (2, p. 27) gives the following example. The rational function of a real variable is

\[
\frac{x}{x-2} = \frac{x-2}{x-2} + \frac{2}{x-2} = 1 + \frac{2}{x-2} .
\]

Then the corresponding interval function results are

\[
[10,12] - 2 = [1,1\frac{1}{2}] , \ 1 + \frac{2}{[10,12]} = [1\frac{1}{5},1\frac{1}{4}]
\]

and

\[
\{\frac{x}{x-2} | x \in [10,12]\} = [1\frac{1}{5},1\frac{1}{4}]
\]

which also points out that in the special case where the interval variable occurs only once, the interval function and the united extension yield the same interval result.
the degenerate intervals by the reals.

Thus in effect, the centered form is an interval arithmetic representation in which the result is computed in terms of the value of the original function at the interval center plus an interval arithmetic computation which is a function of an interval variable that is symmetric and centered at zero.

Table 2.1 lists the results of this example for the various interval representations. The table shows that for small $r$ the mean-value form gives a result which is less conservative than the nested form but more conservative than the centered form.

Moore (2, p. 45) conjectures from the various centered form examples studied (letting $f_c$ denote the centered form representation of $f$) that
\[ w(f_c(J)) \leq w(f(J)) + O(w^2(J)) \quad , \tag{2.47} \]
where $O(h)$ is the usual "order of $h$" numerical analysis terminology.\(^1\)

The computational technique which will subsequently be employed combines the centered form representations,

\[^1\text{The symbol } O(h) \text{ is used to designate that the quantity } |O(h)/h| \text{ has a finite upper bound for all } h \text{ when } |h| \text{ is less than some positive constant.}\]
Table 2.1. Summary of results for various interval arithmetic representations of \( f(x) = x-x^2 \),
\( x \in J = [\frac{1}{2} - r, \frac{1}{2} + r] \)

<table>
<thead>
<tr>
<th>computational technique</th>
<th>representation</th>
<th>interval result</th>
</tr>
</thead>
<tbody>
<tr>
<td>united extension (exact range of values)</td>
<td>( \bar{f}(J) = { f(x) \mid x \in J } )</td>
<td>( \left[ \frac{1}{4} - r^2, \frac{1}{4} \right] )</td>
</tr>
<tr>
<td>centered form</td>
<td>( \frac{1}{4} - (J - \frac{1}{2})^2 \supseteq \bar{f}(J) )</td>
<td>( \left[ \frac{1}{4} - r^2, \frac{1}{4} + r^2 \right] )</td>
</tr>
<tr>
<td>nested form</td>
<td>( J(1-J) \supseteq \bar{f}(J) )</td>
<td>( (r \leq \frac{1}{2}), \left[ \left( \frac{1}{2} - r \right)^2, \left( \frac{1}{2} + r \right)^2 \right] )</td>
</tr>
<tr>
<td>sum of powers form</td>
<td>( J - J^2 \supseteq \bar{f}(J) )</td>
<td>( (r \leq \frac{1}{2}), \left[ \frac{1}{4} - 2r - r^2, \frac{1}{4} + 2r - r^2 \right] )</td>
</tr>
<tr>
<td>&quot;mean-value&quot; form</td>
<td>( \frac{1}{4} + (1 - 2(\frac{1}{2} + (J - \frac{1}{2})[0,1]))(J - \frac{1}{2}) \supseteq \bar{f}(J) )</td>
<td>( \left[ \frac{1}{4} - 2r^2, \frac{1}{4} + 2r^2 \right] )</td>
</tr>
</tbody>
</table>

\(^a\)This result is simply included for completeness of the example by Moore and since the mean-value form will not be used further, its derivation will not be included here (2, p. 47).
subdivision of the interval (see Propositions 2.13, 2.13M and Theorems 2.14, 2.14M) and the nested form computations (2.5 and 2.46).

The interval arithmetic operations (2.3) and (2.4) are accurately defined in so far as computation in the reals are concerned. Suppose however, that a hypothetical decimal computer can retain only one digit after each computation and that an interval result for the square of 0.899 is required. The machine representable interval \([0.8, 0.9]\) contains this number and the square of this interval is \([0.64, 0.81]\). But this would be represented on the one-digit machine as \([0.6, 0.8]\), assuming that truncation of the excess interval endpoint result digits occurs. Obviously the exact result, 0.808201, is not contained in the resulting machine interval.

This simplified analogy begins with part of the original philosophy leading to the topic of interval analysis (the inability of a finite word length machine to exactly represent the real numbers) and it additionally points to the requirement for a machine bounding arithmetic to successfully accomplish the numerical interval arithmetic operations.\(^1\)

\(^1\)This certainly is expected since the best that any finite word length computer can do is to represent a bounded subset of the rational real numbers.
While the implementations of numerical bounding interval arithmetic are discussed in Chapter III, they are largely machine dependent and therefore a preliminary discussion of the IBM System/360 floating-point arithmetic is appropriate here.  

The bit or two state magnetic device is the basic building block of the main core storage (memory) of a computer, where the states are denoted as 0 or 1. In the IBM System/360 machine, floating-point number storage can be of two sizes, 32 bits and 64 bits called short and long (or single- and double-precision). In both (Figure 2.3), bit position 0 contains the sign (a 1 being the negative sign) and bit positions 1-7 contain the characteristic (exponent). The binary point of the fraction is understood to immediately precede bit position 8 and the fraction occupies bit positions 8-31 in short floating-point numbers and 8-63 in long floating-point numbers. The floating-point number is then represented by its binary fraction times the "excess 64" binary characteristic power of 16, with the sign attached. (The range of floating-point number magnitudes is roughly $16^{-64}$ to $16^{+63}$ or

---

1 Floating-point arithmetic is almost exclusively used for scientific computations since it relieves the programmer of the problem of scaling his computations.
<table>
<thead>
<tr>
<th>Sign</th>
<th>Exponent</th>
<th>Fraction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>11000010101010000000000000000000</td>
<td>012345678...</td>
</tr>
<tr>
<td>Bit position</td>
<td>012345678...</td>
<td>...31</td>
</tr>
<tr>
<td>Hexadecimal digits</td>
<td>C15400000</td>
<td>-415400000</td>
</tr>
<tr>
<td>Hexadecimal digits (sign external)</td>
<td>-415400000</td>
<td>-415400000</td>
</tr>
</tbody>
</table>

Figure 2.3. IBM/360 single-precision floating-point number representation for -5.25
Thus the floating-point number represented by Figure 2.3 includes a characteristic of \(1000001 = 2^6 + 2^0 = 65\) and hence the excess 64 exponent is \(65 - 64 = +1\). The fraction is \(0.0101010...0 = 2^{-2} + 2^{-4} + 2^{-6}\). The sign bit is 1 and denotes a negative number so that the quantity represented is \(- (2^{-2} + 2^{-4} + 2^{-6}) \times 16^{+1} = -(2^2 + 2^0 + 2^{-2}) = -5.25\).

A byte consists of eight bits and so a single-precision floating-point number occupies four bytes (called a word) of storage. Each successive group of four bits, when interpreted as a whole binary number with its 16 values represented in increasing order as 0–9 and A–F, is called a hexadecimal digit. Thus each byte contains two hexadecimal digits while the single-precision floating-point number is represented by eight hexadecimal digits.

Equivalently then, the floating-point number represented by Figure 2.3 has a sign plus characteristic (left-most or high-order) byte which is represented by the hexadecimal digits C1 or with an external sign as -41. Thus the "hexadecimal excess 40" characteristic is \(41 - 40 = +01\) and thereby codes an exponent of \(+ (0 \times 16^1 + 1 \times 16^0) = +1\).¹

¹If for example, the hexadecimal characteristic with sign removed had been 2F (characteristic bit string 0101111.), the excess 40 hexadecimal characteristic would be 2F-40 = -11 which codes an exponent of \(- (1 \times 16^2 + 1 \times 16^0) = -17 \) and therefore a fraction multiplier of \(16^{-17}\).
The hexadecimal fraction is \( .540000 = 5 \times 16^{-1} + 4 \times 16^{-2} \).
The sign is negative so that the quantity represented is
\[-(5 \times 16^{-1} + 4 \times 16^{-2}) \times 16^1 = -(5 + 4 \times 16^{-1}) = -5.25.\]

Single- and double-precision floating-point arithmetic are performed in the arithmetic and logic subsystem of the computer which contains, among other things, four 64 bit floating-point registers. When a floating-point arithmetic statement such as \( C = A - B \) is encountered, the machine language produced causes the numbers \( A \) and \( B \) to be retrieved from storage and placed in two of the four registers (a single-precision "load" uses the leftmost or high-order 32 bits leaving the low-order 32 bits unchanged). Consider the following specific single-precision subtraction example where in hexadecimal notation \( A = 4220778A \) and \( B = 40112211 \). The two 32 bit (eight hexadecimal digit) numbers \( A \) and \( B \) are again moved, this time to two 36 bit (nine hexadecimal digit) registers (in the arithmetic hardware section), where the low-order four bits (one hexadecimal digit, called the "guard" digit) of each register are set to zero. Next, the fraction of the number with the smaller characteristic is shifted right the number of hexadecimal characters equal to the difference, thereby aligning the characteristics of the two numbers. Then subtraction of the fraction takes place and the result is normalized so that the first hexadecimal digit in the
result fraction is nonzero, with the characteristic appropriately corrected. The truncated high-order 32 bits (eight hexadecimal digits) of the result are returned to one of the two 64 bit floating-point registers and then to the general storage location of C. For the numbers above, the following takes place (the parenthesized digit is the guard digit):

<table>
<thead>
<tr>
<th>Alignment</th>
<th>A = 4220778A</th>
<th>(0)</th>
<th>none</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>B = 42001122</td>
<td>(1)</td>
<td>1</td>
</tr>
</tbody>
</table>

Subtraction yields \(42206667\) (F)

Normalization of result (not required this example) and return to storage \(C = 42206667\)

It is interesting to note that the effect of the guard digit has been to create a result in which the low-order digit would be in less error if it were rounded to an 8.\(^1\)

If \(C = B - A\) were to represent the right endpoint interval arithmetic, the single-precision representable least upper bound low-order digit of 8 would be demanded.

\(^1\)IBM installed the guard digit hardware modification on the System/360 Model 65 machine at Iowa State University in 1968.
Addition is performed in an identical manner. Multiplication and division are performed similarly, except that a guard digit is not required.\(^1\)

Double-precision floating-point arithmetic operations are described in the references (16, pp. 301-307).

This completes a preliminary floating-point arithmetic discussion necessary for the numerical bounding interval arithmetic implementations of Chapter III.

**The Householder matrix norms and computation of the real fundamental matrix**

For the present discussion, with the end result in mind of determining the sharpest possible bound on the error created by truncating the series representation for \(e^{At}\), where \(A\) is a real matrix and \(t\) is real, consider the bounded linear transformations on \(\mathbb{R}^n\) and the customarily associated real \(n \times n\) matrix arrays, \(A = ((a_{ij}))\).

The following norms on \(\mathbb{R}^n\) induce the corresponding matrix norms on \(A\) (14, p. 99):

\[
\begin{align*}
(a) \text{ if } & \|x\|_1 = \sum_{i=1}^{n} |x_i|, \text{ then } \|A\|_1 = \max_j \left\{ \sum_{i=1}^{n} |a_{ij}| \right\};
\end{align*}
\]

\(^1\)Multiplication returns a full 48 bit result fraction to the arithmetic floating-point register, where at most a four bit left shift will be required for post-normalization, while in division, since only a right shift can occur, only a 24 bit fraction is returned.
(b) if \( \|x\|_2 \equiv \left( \sum_{i=1}^{n} x_i^2 \right)^{1/2} \), then \( \|A\|_2 \equiv \lambda_{\text{max}}^{1/2} \),

where \( \lambda_{\text{max}} \) is the largest eigenvalue of \( A^T A \); and

(c) if \( \|x\|_\infty \equiv \max\{|x_i|\} \), then \( \|A\|_\infty \equiv \max\{\sum_{j=1}^{n} |a_{ij}|\} \).

\[ (2.48) \]

The properties of the vector norms on \( \mathbb{R}^n \) and the matrix norms on \( A \) are listed here for convenience (\( v \) and \( m \) indicate the vector and matrix correspondence):

1. \( (I_v) \) \( \|x\| = 0 \), if and only if \( x = 0 \); \( \|x\| \geq 0 \);

2. \( (II_v) \) \( \|\gamma x\| = |\gamma| \cdot \|x\| \) for any scalar \( \gamma \);

3. \( (III_v) \) \( \|x+y\| \leq \|x\| + \|y\| \);

4. \( (I_m) \) \( \|A\| = 0 \), if and only if \( A = ((0_{ij})) \); \( \|A\| \geq 0 \);

5. \( (II_m) \) \( \|\gamma A\| = |\gamma| \cdot \|A\| \) for any scalar \( \gamma \);

6. \( (III_m) \) \( \|A+B\| \leq \|A\| + \|B\| \); and

7. \( (IV_m) \) \( \|AB\| \leq \|A\| \cdot \|B\| \).

\[ (2.49) \]

A matrix norm is said to be "consistent" with a given vector norm when for every \( A \) and \( x \) it is true that

\( (V) \) \( \|Ax\| \leq \|A\| \cdot \|x\| \).

\[ (2.50) \]

A matrix norm is said to be "subordinate" to the vector
norm in the case where the norms are consistent and for every \( A \) there exists an \( x \neq 0 \) such that

\[
(VI) \|Ax\| \equiv \|A\| \|x\| \quad .
\]

(2.51)

The matrix norms \( \| \cdot \|_1, \| \cdot \|_2 \) and \( \| \cdot \|_\infty \) are subordinate to their corresponding vector norms (17, pp. 1, 2; 18, p. 5).

The subsequent two vector and matrix norms were developed by A. S. Householder (17, pp. 9-16; 18, pp. 9, 10).

Let

\[
\|x\|_u \equiv \|G^{-1}x\|_\infty \quad , \quad \text{where } G = \text{diag}(g_i), \ g_i > 0 \quad .
\]

(2.52)

Since the \( \| \cdot \|_\infty \) matrix norm is subordinate to the vector norm,

\[
\|A\|_u = \sup_{x \neq 0} \frac{\|Ax\|_u}{\|x\|_u} = \sup_{x \neq 0} \frac{\|G^{-1}Ax\|_\infty}{\|G^{-1}x\|_\infty}
\]

\[
= \max_{\|G^{-1}x\|_\infty = 1} \frac{\|G^{-1}Ag^{-1}x\|_\infty}{\|G^{-1}x\|_\infty} \equiv \|G^{-1}A\|_\infty
\]

\[
= \max_{i} \left\{ \frac{1}{g_i} \sum_{j=1}^{n} g_j |a_{ij}| \right\} \quad .
\]

(2.53)

Also let

\[
\|x\|_u' \equiv \|Hx\|_1 \quad , \quad H = \text{diag}(h_i) \quad , \quad h_i > 0 \quad .
\]

(2.54)
Since the $\| \cdot \|_1$ matrix norm is subordinate to the vector norm,

$$\|A\|_u' = \sup_{x \neq 0} \frac{\|Ax\|_u'}{\|x\|_u'} = \sup_{x \neq 0} \frac{\|HAx\|_1}{\|Hx\|_1} = \max_{\|Hx\|_1=1} \frac{\|HAH^{-1}Hx\|_1}{\|Hx\|_1}$$

$$= \|HAH^{-1}\|_1 \leq \max_{i=1}^n \frac{1}{h_i} \sum_{j=1}^n h_i |a_{ij}| \quad (2.55)$$

A real matrix $B \in (b_{ij})$, $b_{ij} \geq 0$, is called "irreducible" in the sense of Frobenius (19, p. 50) if it is not possible to split the index set \{1, ..., n\} into two nonvoid disjoint sets $\alpha$ and $\beta$ such that $b_{ij} = 0$ for all $i \in \alpha$, $j \in \beta$.

For any real matrix $A = (a_{ij})$, the matrix $|A| = (|a_{ij}|)$ is called the "abmatrix of $A$" (17, p. 9).

The Frobenius theorem on the spectral properties of irreducible nonnegative matrices (19, p. 53) states that for such a matrix there is a simple real positive characteristic value (eigenvalue) $r$ which is greater than or equal to the modulus of every other characteristic value and that the corresponding characteristic vector (eigenvector) $w$ has strictly positive components. It is interesting to note that if this irreducible matrix is $|A|$, the Frobenius theorem proof develops the characterization (19, p. 65) $r = \|A\|_u$ (2.53) with $G = \text{diag}(w_i)$.

The modulus of the largest characteristic value of a matrix is called the "spectral radius". For any real matrix
A, the infimum of the set of values of the matrix norms on A induced by the family of all vector norms on $\mathbb{R}^n$ is the spectral radius and in this sense then, for an irreducible matrix $|A|$, the Householder matrix norm with $G = \text{diag}(w_i)$ produces the optimally infimum value, the spectral radius of $|A|$ (20, p. 249). When $|A|$ is irreducible, the reference additionally characterizes the subfamily of vector norms and induced matrix norms on $A$ for which this Householder matrix norm yields an optimally infimum value.\(^1\)

It must be remarked that the class of nonnegative real matrices $|A|$ for which either $\|A\|_x$ or $\|A\|_u$, yields an infimum value is larger than the set of irreducible matrices and necessary and sufficient conditions characterizing this additional set of matrices are stated precisely in the references (19, p. 77; Theorem 6).

The consequence of the above remark is that $\|A\|_u$, may be defined while $\|A\|_u$ is not\(^2\) (or the converse). Therefore,\(^3\)

\(^{1}\)This family includes the $\| \cdot \|_\infty$ and $\| \cdot \|_1$ vector norms. An interesting comparison of various matrix norms is found in (21).

\(^{2}\)For example, $A = \begin{bmatrix} 0 & 1 & 0 \\ -1 & -0.4 & 1 \\ 0 & 0 & 0 \end{bmatrix}$ results in $g_3 = 0$ while $A^T$ yields $h_i > 0$, $i=1,2,3$ and thus $\|A\|_u = 0.2 + \sqrt{1.04}$, the spectral radius of $|A|$ and $|A^T|$.\(^4\)
the interval integration algorithm provides external control for selection of the $\| \cdot \|_y$ or $\| \cdot \|_{y'}$ Householder matrix norms.

It is apparent from (2.48) that for any $i,j$, $|a_{ij}| \leq \|A\|_y$ when $y$ denotes 1 or $\infty$. This relationship is also true in the case where $y$ denotes 2. This claim may be proved in the following manner.

The real symmetric matrix $B = A^T A$ is normal in the sense that $B^*B = BB^*$ (since $B^* = B^T = B^T = B$, where $B$ denotes the complex conjugate of $B$) and therefore has at least one characteristic value $\lambda_t$ such that

$$|\lambda_t| \geq \max_{i,j} |b_{ij}| \quad (22, p. 161).$$

But $A^T A$ is nonnegative definite and this means that all of its characteristic values are real and nonnegative (23, pp. 266-270). Now for any $i,j$

$$|b_{ii}| \equiv \sum_{k=1}^{n} |a_{ki}|^2 \geq a_{ji}^2$$

and therefore, using (2.48),

$$\|A\|_2 \geq |\lambda_t|^{1/2} \geq |a_{ij}|,$$

which was the claim to be proved. In the case of the two Householder matrix norms,
\[
\frac{g_j}{g_i} |a_{ij}| \leq \|A\| \quad \text{and} \quad \frac{h_i}{h_j} |a_{ij}| \leq \|A\|' \quad .
\] (2.56)

Turn now to the numerical computation of the real matrix exponential (24, p. 104-106; 25). Let

\[
\bar{a}(t) \equiv e^{At} \equiv \sum_{i=0}^{\infty} \frac{A^{i}t^{i}}{i!} , \quad t > 0 ,
\]

\[
M \equiv ((m_{ij})) \equiv \sum_{i=0}^{k} \frac{A^{i}t^{i}}{i!} \quad \text{and}
\]

\[
R \equiv ((r_{ij})) \equiv \sum_{i=k+1}^{\infty} \frac{A^{i}t^{i}}{i!} ,
\] (2.57)

where it is assumed that \(\|A\|\) and \(\|A\|'\) exist. In a manner similar to the numerical scheme described previously for computation of the interval matrix exponential, letting \(y\) denote 1, 2 or \(\infty\), for any \(i, j\)

\[
|r_{ij}| \leq \|R\|_{y} \leq \gamma_{y} \equiv \frac{\left(\|A\|_{y}t\right)^{k+1}}{(k+1)!} \cdot \frac{1}{1 - \frac{\|A\|_{y}t}{k+2}} ,
\] (2.58)

provided

\[
0 < \frac{\|A\|_{y}t}{k+2} < 1 \quad .
\]

In the case of the Householder matrix norms (\(\gamma\) denotes the same calculation with the norms replaced appropriately in Equation 2.58), for any \(i, j\)
\[
\frac{g_j}{g_i} |r_{ij}| \leq \mathcal{J}_u \quad \text{and} \quad \frac{h_i}{h_j} |r_{ij}| \leq \mathcal{J}_u'. \quad (2.59)
\]

Then, if for each \(i,j\)

\[
|r_{ij}| \leq \mathcal{J}(i, 2 \text{ or } \infty) \leq 10^{-p} |m_{ij}|
\]

\[
|r_{ij}| \leq \left( \frac{g_i}{g_j} \right) \mathcal{J}_u \leq 10^{-p} |m_{ij}| \quad \text{and}
\]

\[
|r_{ij}| \leq \left( \frac{h_i}{h_j} \right) \mathcal{J}_u' \leq 10^{-p} |m_{ij}| \quad (2.60)
\]

the error in any term of the truncated matrix series \(M\) approximating \(\mathcal{A}(t) = e^{At}\) is less than \(10^{-p}|m_{ij}|\).\(^1\)

Hopefully the effect of using the "optimally infimum" Householder matrix norm in the computation of \(\mathcal{J}\) is to obtain a sharper result for this bound, thereby minimizing the number of terms in the series that is required to yield an approximation which is accurate to within a given decimal.

Of course the "optimal" Householder matrix norms require computation of the maximal characteristic value and vector of \(|A|\), but this is available in the fast double-precision EISPAC routine mounted on disc and available from

\(^1\)This is possible since \(k\) may be increased until the relationship is satisfied (this was previously proved).
SYSLIB in the present university computer system.\footnote{The EISPAC routine is the result of combined efforts at the Applied Mathematics Division of Argonne National Laboratory (supported by the AEC) and was furnished to the Ames Laboratory at Iowa State University.}

Table 2.2 gives results for two examples,

\[
A = \begin{bmatrix}
0 & 1 & 0 \\
0 & 0 & 1 \\
-0.75 & -2.75 & -3 \\
\end{bmatrix}
\] (24, pp. 104-106) \hspace{1cm} (2.61)

and

\[
B = \begin{bmatrix}
-0.05 & -6.0 & 0 \\
-10^{-3} & -0.15 & 0 \\
1.0 & 0 & 0 \\
0 & 1.0 & 0 \\
\end{bmatrix}
\] (2.62)

\hspace{1cm} (26, p. 300; 27).

The abmatrix of the first example is irreducible, while in the second example (a minesweeper automatic steering problem) \(|B|\) is reducible.\footnote{Let \(a = \{1,2\}\) and \(\beta = \{3,4\}\). Then \(|b_{ij}| = 0\) for each \(i \in a\) and \(j \in \beta\). Hence the abmatrix is reducible. However the \(||B||_u\) norm is still defined.} (The spectral radii and \(||\cdot||_2\) and \(||\cdot||_u\) norms were computed using the double-precision EISPAC routine and the remaining computations were evaluated...}
Table 2.2. Numerical results for the computation of $\int$ in the two examples (2.61) and (2.62), with various matrix norms

| Spectral Radius of $A$ | $\|A\|\ (\cdot) t$ | $\int (\cdot)$ | $|r_{ij}| \leq$ |
|----------------------|----------------------|-----------------|----------------|
|                      | $\|A\|_1$ 6.500000  | 6.5 x $10^{-2}$ | 6.104436 x $10^{-8}$ | $\int_\infty$ |
|                      | $\|A\|_\infty$ 4.000000 | 4.0 x $10^{-2}$ | 7.524985 x $10^{-10}$ | $\int_1$ |
|                      | $\|A\|_2$ 4.253694 | 4.253694 x $10^{-2}$ | 1.312234 x $10^{-9}$ | $\int_2$ |
|                      | $\|A\|_u$ 3.780003 | 3.780003 x $10^{-2}$ | 4.512331 x $10^{-10}$ | $|W^* \int_u| = 6.447407 \times 10^{-9}$ |

| Spectral Radius of $B$ | $\|B\|\ (\cdot) t$ | $\int (\cdot)$ | $|r_{ij}| \leq$ |
|----------------------|----------------------|-----------------|----------------|
|                      | $\|B\|_\infty$ 14.0 | 6.363636 x $10^{-2}$ | 2.448501 x $10^{-17}$ | $\int_\infty$ |
|                      | $\|B\|_1$ 13.0 | 5.909091 x $10^{-2}$ | 5.139476 x $10^{-18}$ | $\int_1$ |
|                      | $\|B\|_2$ 13.03841 | 5.926550 x $10^{-2}$ | 5.468980 x $10^{-18}$ | $\int_2$ |
|                      | $\|B\|_u$ 0.1921954 | 3.494462 x $10^{-2}$ | 8.108984 x $10^{-23}$ | $\frac{907.58886}{1.588150} \times \int_u$ |
|                      |                      |                  | $= 4.634085 \times 10^{-20}$ | |

\(^a t = 4.0 \text{ vice } 0.1 \text{ in the case of } \|B\|_u\).
on a 12-digit Hewlett-Packard Model 9100B calculator.)

It will be noted in Table 2.2 for the \( A \) matrix example with \( k=8 \) and \( t=0.1 \), that the

\[
\max_{i,j} \{ g_{ij} \} \cdot \mathcal{S}_u \equiv (1.051973 \cdot 0.07362483) \cdot \mathcal{S}_u \equiv W \cdot \mathcal{S}_u \quad (2.63)
\]

bound is an order of magnitude sharper than the \( \mathcal{S}_\infty \) bound obtained in the reference (24, pp. 104-107) but more conservative than the \( \mathcal{S}_2 \) and \( \mathcal{S}_1 \) bounds. However, given

\[
W \equiv \max_{i,j} \{ g_{ij} \}
\]

since \( \|A\|_u < \|A\|_y \), where \( y \) denotes 1, 2 or \( \infty \), by selecting \( k \) sufficiently large, for each \( i,j \) it is always possible to satisfy

\[
|r_{ij}| \leq W \mathcal{S}_u < \mathcal{S}_y \quad .\quad (2.64)
\]

For the example of matrix \( B \), Table 2.2 points up the

\[
1\text{This may be understood by observing that for } k \text{ sufficiently large, it is always possible to satisfy}
\]

\[
\frac{\mathcal{S}_y}{\mathcal{S}_u} = \left( \frac{\|A\|_y}{\|A\|_u} \right)^{k+1} \cdot \left[ \frac{1}{1 - \frac{\|A\|_y t}{k+2}} \right] > W .
\]

In the case of

\[
\frac{\|A\|_1}{\|A\|_u} > 1 , \quad k \geq 47 \text{ yields } W \cdot \mathcal{S}_u < \mathcal{S}_1 .
\]
importance in selecting an optimal matrix norm in computing the bound $J$. With a fixed maximal number of terms in the truncated series approximating the fundamental matrix $\bar{\alpha}(t=4.0)$, the optimal matrix norm bound $J_u$ is several orders of magnitude sharper than the other results for the same series approximating $\bar{\alpha}(t=0.1)$. It should be noted however that as $t$ increases, some or all of the $|m_{ij}|$ elements may be decreasing and therefore

$$|r_{ij}| \leq W \ J_u \leq 10^{-p}|m_{ij}|$$

may be more difficult to satisfy for all $i, j$, even though $J_u$ is decreasing with increasing $k$.1

**Nested centered form computations for the interval fundamental matrix**

The purpose of the present section is to reformulate the interval matrix exponential computation technique implementing (1) the perturbation parameter interval partitioning philosophy of Theorem 2.14M and Proposition 2.13M, (2) the nested and centered form techniques for reducing the conservativeness of the interval arithmetic evaluations

---

1 This would certainly be true in the case where $\bar{\alpha}(t)$ represents the state transition matrix for a system where all the eigenvalues of $A$ have negative real parts (28, p. 81).
and (3) the optimal Householder matrix norm.

In this sense then, return to the computational scheme previously developed which begins with (2.23).

Suppose that \( A \equiv ([a^L_{ij}, a^R_{ij}]) \) is the computer representation of the input "interval" matrix and \( T \) is the degenerate interval \( T = [t, t] \).\(^1\) Using the bounding interval arithmetic with

\[
\hat{a}^L_{ij} \equiv [a^L_{ij}, a^L_{ij}] \quad \text{and} \quad \hat{a}^R_{ij} \equiv [a^R_{ij}, a^R_{ij}],
\]

compute the interval matrices

\[
G_1 \equiv \left( \frac{\hat{a}^L_{ij} + \hat{a}^R_{ij}}{2} \right) \quad \text{and} \quad G_2 \equiv \left( \frac{\hat{a}^R_{ij} - \hat{a}^L_{ij}}{2} \right).
\]

Then

\[
A_{\text{True}} \subseteq G_1 + \theta G_2, \quad \text{where} \quad \theta = [-1, 1]. \quad (2.66)
\]

Let the Cauchy sequence of rational interval matrix functions \( \{F_K(\theta)\} \) be defined by

---

\(^1\)As will subsequently be seen in the interval computation (2.65) for \( G_2 \), in order to accommodate a "signed" interval matrix element dependence on the perturbation parameter, \( a^R_{ij} < a^L_{ij} \) is allowed for the input "interval" matrix \( A \). Here, \( a^L_{ij}, a^R_{ij} \) and \( t \) are single-precision floating-point machine numbers.

\(^2\)\( A_{\text{True}} \) denotes the interval matrix

\[
A_{\text{True}} = ([\min(a^L_{ij}, a^R_{ij}), \max(a^L_{ij}, a^R_{ij})]).
\]
\[ F_K(\theta) = \sum_{i=0}^{K} \frac{(G_1 + \theta G_2)^i}{i!} \]  

(2.67)

For convenience in notation and programming, using the bounding interval arithmetic recursively compute the interval matrices for \( i=2, \ldots, K \) and \( p=0, \ldots, i \)

\[
\frac{G_{i+1}(i+1)}{2} + p = \left( \begin{array}{l} 1, p=0 \\ 1, p=0, \ldots, i \end{array} \right) \sum_{l=1}^{i} \frac{G_{i-l}+p-1}{2} G_{l} 
\]

(2.68)

Algebraically expanding (2.67), substituting (2.68) and rearranging, obtain

\[
I + (G_1 + \theta G_2)T \\
+ (G_3 + \theta G_4 + \theta^2 G_5)T^2 \\
+ \\
\vdots \\
+ \left( \frac{G_{K+1}(K+1)}{2} + \cdots + \theta^K G_{K+3} \right) T^K 
\]

(2.69a)

and then

\[
I + \sum_{l=1}^{K} \frac{G_{l}(l+1)}{2} T^l + \sum_{p=1}^{K} \left( \sum_{\ell=p}^{K} \frac{G_{\ell}(\ell+1)}{2} + p \right) T^l \theta^p . 
\]

(2.69b)

Now subdivide the perturbation parameter interval \( \theta = [-1,1] \) into \( M \) "equal" width subintervals (using single-
precision arithmetic),

\[
\phi_i \equiv \left[ \frac{2(i-1) - M}{M}, \frac{2i - M}{M} \right] = \left[ \phi_i^L, \phi_i^R \right], \quad i = 1, \ldots, M.
\]

(2.70)

Represent each subinterval in the centered interval arithmetic form (computing with single-precision arithmetic)

\[
\phi_{i_c} \equiv \left[ \frac{2i - M - 1}{M}, \frac{2i - M - 1}{M} \right] = [c_i, c_i]
\]

and

\[
\eta_i \equiv [-w_i, w_i], \quad \text{where } w_i = \max \{c_i - \phi_i^L, \phi_i^R - c_i\}.
\]

(2.71)

Note that

\[
\phi_i \subset \phi_{i_c} + \eta_i.
\]

(2.72)

Then (2.69b) becomes

\[
I + \sum_{\ell=1}^{K} G^{\ell(\ell+1)/2} T^\ell + \sum_{p=1}^{K} \sum_{\ell=p}^{K} G^{\ell(\ell+1)/2 + p} \cdot \sum_{j=0}^{p} C_{j}^{p} \phi_{i_c}^j \eta_i^j,
\]

(2.73)

where the binomial coefficients are defined by
Using bounding interval arithmetic computations, define the interval matrices

\[ B_{K+1} = (\cdots (G_{K+1} + G_{K-1}) + \cdots + G_1) + I \]

and

\[ B_{K+1-p} = (\cdots (G_{K+1} + G_{K-1}) + \cdots + G_{p+1}) + I \]

Using (2.75a and b), (2.73) may be rewritten as

\[ \binom{p}{j} \equiv \frac{p!}{j!(p-j)!} \quad \text{(2.74)} \]

Using (2.75a and b), (2.73) may be rewritten as

\[ B_{K+1} = \sum_{p=1}^{K} B_{K+1-p} \binom{p}{j} + \sum_{p=1}^{K} B_{K+1-p} \binom{p}{j} \binom{p-j}{j} \quad \text{(2.76)} \]

When the computations for the binomial coefficients are actually programmed, it is simpler and more accurate to obtain these interval representations (which incidentally are degenerate intervals here since the maximum coefficient that is required is \( C_{10}^{20} = 184,756 = 452D1B40 \)) by recursively completing Pascal's triangle (15, p. 53).
Rearranging the last term of (2.76) in powers of $\eta_i$,

\[
\sum_{j=1}^{K} \left( \sum_{p=j}^{K} B_{K+1-p} C_j^{P-j} \right) \eta_i^j .
\] (2.77)

Using the nested computations, define the interval matrices

\[
E_{K+1} \equiv (\ldots (B_1 \Theta_i + B_2 \Theta_i^c + \ldots + B_K \Theta_i^c + B_{K+1} \Theta_i^c)
\]

\[
\subset B_{K+1} + \sum_{p=1}^{K} B_{K+1-p} \Theta_i^c
\] (2.78a)

and

\[
E_{K+1-j} \equiv (\ldots (B_1 C_j \Theta_i^c + B_2 C_j \Theta_i^c + \ldots + B_{K-j} C_j^{K-1} + B_{K-j+1} C_j^{K}) \Theta_i^c
\]

\[
\subset \sum_{p=j}^{K} B_{K+1-p} C_j^{P-j} \Theta_i^c , \quad j=1, \ldots, K .
\] (2.78b)

Then using (2.78a and b), (2.76) may be rewritten in nested form as

\[
(\ldots (E_1 \eta_i + E_2 \eta_i + \ldots + E_K \eta_i + E_{K+1} \eta_i + E_{K+1} \eta_i \times
\]

\[
\subset E_{K+1} + \sum_{j=1}^{K} E_{K+1-j} \eta_i^j .
\] (2.79)

This is the centered form interval expression for $E_{K}(\Theta_i)$ in (2.67). $E_{K+1}$ is the nested expression for the interval
center result \( F_k(\theta_i) \) and 
\( (\cdots (E_1\eta_i + E_2)\eta_i + \cdots + E_k)\eta_i \) is 
the nested expression for the balance of the centered form, 
explicitly in terms of the "zero-symmetric" interval vari-
able \( \eta_i = \theta_i - \theta_{i_c} \) and indirectly in terms of the interval 
center variable \( \theta_{i_c} \) (see Equation 2.78b). Denote the 
centered interval form of (2.57) by 
\[
\tilde{F}_k(\theta_i + \eta_i) = (\cdots (E_1\eta_i + E_2)\eta_i + \cdots + E_k)\eta_i + E_k + 1 .
\]
(2.80)

From the definition of the united extension in Propositions 
2.6M and 2.9M and the relation (2.72), obviously 
\[
\bar{F}_k(\alpha_i) \subset \tilde{F}_k(\theta_i + \eta_i) .
\]
(2.81)

While it is not proved here, for all of the numerical 
initial-value examples of Chapter V, bounding interval 
arithmetic computations have verified that 
\[
\tilde{F}_k(\theta_i + \eta_i) \subset F_k(\theta_i) .
\]
(2.82)

---

1. Because of the method by which \( \eta_i \) and \( \theta_{i_c} \) were com-
puted (2.71), equality is not strictly true. However, in 
terms of exact interval arithmetic this is theoretically 
correct.

2. The subset inclusion relation of (2.82) is frequently 
remarkable in the reduction of the width of the interval re-
sults accomplished by the centered form. For example, in 
the fifth-order optimal regulator problem of Chapter V, with 
K=20, t=4.0, i=19 (M=25 subdivisions), a typical result (the 
(3,2) matrix element) was \([-0.2459952, 0.3559166] \supset\) 
\([0.5354307 \times 10^{-1}, 0.563951 \times 10^{-1}]\).
Letting the remainder term associated with \( R_K(\Theta_i) \) in (2.67) be denoted by

\[
R_K(\Theta_i) \equiv (\ell_{m_K}(\Theta_i)) \equiv \sum_{j=K+1}^{\infty} \frac{(G_1 + \Theta_i G_2)^j}{j!}
\] (2.83)

and letting the remainder term associated with \( \tilde{R}_K(\Theta_i + \eta_i) \) in (2.80) be denoted by

\[
\tilde{R}_K(\Theta_i + \eta_i) \equiv (\tilde{\ell}_{m_K}(\Theta_i + \eta_i)),
\] (2.84)\]

assume also that

\[
\tilde{R}_K(\Theta_i + \eta_i) \subset R_K(\Theta_i) \quad .
\] (2.85)\]

Then for each \( \ell \) and \( m \) (\(|\cdot|\) denotes the "magnitude" of an interval),

\[
|\tilde{\ell}_{m_K}(\Theta_i + \eta_i)| \leq |\ell_{m_K}(\Theta_i)| .
\] (2.86)\]

Assume that

\[
|G_1 + \Theta_i G_2| \equiv (|g_{m_1} + \Theta_i g_{m_2}|)
\]

allows the optimal Householder matrix norm \( \| \cdot \|_u \) (2.53). Then if

\[1\text{In view of the nature of the remainder terms involved for increasing } K \text{ and the less conservative interval results produced by the centered form (see Equation 2.82), this is not an unreasonable assumption.} \]
\[
\lambda_i \equiv \| \left( |G_1 + e_i G_2| \right) \|_u \equiv \| G^{-1} |G_1 + e_i G_2| G \|_\infty
\]

\[
<< \| \left( |G_1 + e_i G_2| \right) \|_y, 1
\]

(2.87)

where \( y \) indicates 1, 2 or \( \infty \) and \( G \equiv \text{diag}(g_p), g_p > 0, \)

\( p=1, \ldots, n \), applying (2.20), (2.35), (2.56) and (2.85),

for each \( \ell \) and \( m \)

\[
\frac{g_m^{\ell m} |r_{\ell m} (e_i \epsilon + \eta_i)|}{g_\ell^{\ell m}} \leq \frac{g_m^{\ell m} |r_{\ell m} (e_i)|}{g_\ell^{\ell m}} \leq \gamma y_i
\]

\[
= \frac{\lambda_i^{K+1}}{(K+1)!} \frac{1}{1 - \frac{\lambda_i}{K+2}} ,
\]

(2.88)

assuming \( \lambda_i / (K+2) < 1 \).

Turn now to the computational relationship which is

equivalent\(^1\) to (2.37) for the hexadecimal machine,

\(^1\)The "<<" inequality here is intended to indicate that

for any \( \ell \) and \( m \),

\[
\frac{g_\ell^{\ell m}}{g_m^{\ell m}} \gamma y_i < \gamma y_i
\]

where \( y \) indicates the 1, 2 or \( \infty \) matrix norm use in (2.88).

\(^2\)Relations (2.82), (2.85) and (2.88) are implicitly

used in obtaining (2.89).
where \((P+1)\) is one of the integers \(\{1, \ldots, 6\}\) which indicates the number of the single-precision hexadecimal fraction digit to which the final interval result endpoints are required to be accurate when approximating the infinite series.

In the actual implementation of the numerical scheme for the calculation of the matrix exponential, if (2.89) is satisfied for all \(l\) and \(m\) for the preselected integer \(P\) and the initial selection of \(K\), then the final interval matrix result

\[
\tilde{r}_K(\theta_{i_c} + \eta_i) + Z_i, \quad \text{where } Z_i = \left(\left[\left[\frac{G_m}{G_m}\right]u_i, \frac{G_m}{G_m}\right]u_i\right)
\]

(2.90)

will contain\(^1\) the centered form infinite series result for

\[
e^{(G_1 + (\theta_{i_c} + \eta_i) G_2) T}
\]

(2.91)

\(^1\)This assumes the neglecting of possible additional accumulation of bounding errors which would result in the continuing calculation.
and the relative interval endpoint error bounds are given by (2.44) and (2.45).\(^1\)

In general, (2.89) will not always be satisfied for each \( \ell \) and \( m \). The following discussion describes some of these occurrences and the programming techniques employed.

The most obvious case where (2.89) will never be satisfied occurs when the \( p \)-th row of the original \( A \) matrix (and consequently the same rows of the \( G_1 \) and \( G_2 \) matrix) consists of degenerate zero interval elements. Then

\[
\tilde{f}_{pm_k}(\theta_{i} + \eta_{i}) = \begin{cases} 
[0,0], & m \neq p \\
[1,1], & m = p
\end{cases}
\]  

(2.92)

In this case, a logical test will indicate interval result degeneracy and the corresponding setting of

\[ z_{pm_i} = [0,0] \]

will yield the correct result for

\[
\tilde{f}_{pm_k}(\theta_{i} + \eta_{i}) + z_{pm_i}.
\]

Initially, for the preselected algorithm "accuracy" (input integer \( P \)), it may happen that the estimated starting

\(^1\)However, the bounds here are computed with respect to the hexadecimal base \( 16 \).

\(^2\)A similar case occurs with respect to the \( p \)-th column.
value of $K$ is not sufficiently large to satisfy (2.89) for all $l$ and $m$. For this reason, an automatic increase in the value of $K$ must be programmed into the routine. However, it must be pointed out that computer storage limits place a final constraint on this technique.\textsuperscript{1}

If the minimum of the two values on the right-hand side of (2.89) does not satisfy the relation but the maximum does, this is termed a "single-fault" and computational experience has demonstrated that $K+1$ usually results in clearing the single-fault. If additionally the maximum does not satisfy the relation, this is termed a "double-fault" and it may be necessary to "run-up" $K$ beyond $K+1$.

Since situations may obviously occur where there is a great disparity between elements of

$$\tilde{F}_K(\theta_{iC} + \eta_i),$$

some predetermined input judgment should be programmed into the routine so that there will be a preset limit number for the two types of faults which are allowed. This "fault acceptance" will not contradict the set containment of (2.91) by (2.90) but the relative error bounds given by (2.44) and

\textsuperscript{1}In the algorithm a limiting value of $K=20$ for $5 \times 5$ interval matrices results in a 46 K byte array for $G_1, \cdots, G_{(20)}(23) = G_{230}.$
(2.45) will no longer be valid. This is in fact done in the algorithm.

This completes the development and discussion of the reformulated interval matrix exponential (interval fundamental matrix) computation technique.

To summarize, the technique uses the concepts of subdivision of the parameter interval and the centered form representations for the resulting parameter subintervals, the centered and nested form matrix interval arithmetic computations, bounding of the interval matrix function metric employing the optimally infimum Householder matrix norm (for irreducible nonnegative real matrices) and interval augmenting of the computable truncated interval matrix series for set containment of the interval matrix infinite series form of the interval matrix exponential with prescribed relative error bounds.

The linear interval integration technique described briefly following Theorem 2.14M and subsequently implemented in Chapter IV requires the computation of interval fundamental matrices for each partition subinterval. In this sense the above technique provides the necessary computation method.
The Kalman Covariance Equation and Two Delayed State Inertial Navigation Solution Methods

The Kalman filter considered in this section is a recursive algorithm which computes the optimally minimal error variance estimates of the states of a linear discrete time state variable stochastic process from a sequence of measurements which are linearly related to the states. The process is zero mean gaussian with additive process disturbances and measurement errors which are zero mean gaussian white noise sequences (29, p. 185) that are individually and jointly independent.

In the following description, dependence on the discrete time sequence variable $t^k$ will be denoted by the subscript $k$ ($x^k = x(t^k)$) and the state transition or fundamental matrix will be written with the single $k$ subscript,

$$\Phi_k \equiv \Phi(t^k, t^{k+1}) .$$

Statistical expectation will be denoted by $E[\cdot]$. The discrete time process dynamics and measurement equations are

$$x^{k+1} = \Phi_k x^k + h_k$$

and

$$y_k = M_k x^k + \Delta y_k ,$$

(2.93)

where
$x_k$ is the n vector zero mean gaussian process state,
$h_k$ is the n vector independent zero mean gaussian white noise sequence process disturbance,
$y_k$ is the s vector zero mean gaussian measurement,
$\Delta y_k$ is the s vector independent zero mean gaussian white noise sequence measurement error and
$M_k$ is the s x n measurement matrix.

The Kalman filter solution is then given by recursively computing the results of the sequence of five equations (29, pp. 176-177)

$$K_k = P_k^+ M_k^T (M_k P_k^+ M_k^T + V_k)^{-1} ,$$

$$\hat{x}_k = \hat{x}_k + K_k (y_k - M_k \hat{x}_k) ,$$

$$P_k = E[(\tilde{x}_k - x_k) (\tilde{x}_k - x_k)^T] = P_k^+ - K_k (M_k P_k^+ M_k^T + V_k) K_k^T ,$$

$$\hat{x}_{k+1} = \bar{A}_k \hat{x}_k$$

and

$$P_{k+1}^+ = E[(\hat{x}_{k+1} - x_{k+1}) (\hat{x}_{k+1} - x_{k+1})^T] = \bar{A}_k P_k \bar{A}_k^T + H_k ,$$

where
\( K_k \) is the nxn optimal weighting (gain) matrix,

\( \hat{x}_k \) is the a priori estimate of \( x_k \) (the estimate based on measurements \( y_0, \ldots, y_{k-1} \)),

\( \tilde{x}_k \) is the a posteriori estimate of \( x_k \) (the estimate based on measurements \( y_0, \ldots, y_k \)),

\( P_k^* \) is the nxn a priori error covariance matrix and

\( P_k \) is the nxn a posteriori error covariance matrix.

Other terms which arise in the algorithm are (1) the covariance matrix of the process disturbance

\[
E[h_k h_j^T] = H_k \delta_{kj} ,
\]

where \( H_k \) is a symmetric nonnegative definite nxn matrix and \( \delta_{kj} \) is the Kronecker delta and (2) the covariance matrix of the measurement errors

\[
E[\Delta y_k \Delta y_j^T] = V_k \delta_{kj} ,
\]

where \( V_k \) is a symmetric nonnegative definite sxn matrix.

Equations (2.96) and (2.98) may be written as a single equation which is called the covariance equation for the Kalman filter,

\[
P_{k+1}^* = \tilde{d}_k [P_k - P_k^* M_k^T (M_k P_k M_k^T + V_k)^{-1} M_k P_k^*] \tilde{d}_k^T + H_k .
\]

From an analysis point of view, the Kalman covariance
equation may be recursively computed without recourse to the system measurements and estimates of the states.

In his dissertation, Duven investigated numerical aspects of an inherently unstable equilibrium solution to the covariance equation in the case of an undriven and observable 2-dimensional random walk process (30, pp. 106, 114-122). In particular, using two numerical orderings of the computations involved when employing the algebraically equivalent form of (2.96),

\[ P_k = (I - K_k M_k) P_k^* (I - K_k M_k)^T + K_k V_k K_k^T, \]  

(2.102)

he demonstrated that because of single-precision truncation this method does not insure that the matrix \( P_{k+1}^* \) will be nonnegative definite.\(^1\)

Since the bounding interval arithmetic provides a method of empirically observing the computational effects of single-precision truncation, as a possible numerical investigation technique, single-step results for the two numerical orderings described above (nested and expanded computations) and the method employing (2.96) were calculated in the bounding interval arithmetic, using truncated double-precision solution results as the initial conditions.

\(^1\)Sorenson (31, p. 261) proposed the use of (2.102) since each term is quadratic and therefore nonnegative definite.
for each interval solution step. These results are described in Chapter V where the stepwise interval endpoint errors (with respect to the double-precision result) are examined for the interval forms of the three computation techniques.

In the application of the Kalman filter to a specific inertial navigation system where a satellite Doppler count is included as a measurement (32, pp. 6-7), the measurement equation of (2.93) becomes

\[ y_k = M_k x_k + N_k x_{k-1} + \Delta y_k \]  \hspace{2cm} (2.93a)

where \( N_k \) is the sxn measurement matrix for the delayed state \( x_{k-1} \). In this case, the modified Kalman filter equations are given by (33, p. 69)

\[ K_k = (P_k^* M_k + \delta_k-1 P_k-1 N_k^T) Q_k^{-1} \]  \hspace{2cm} (2.94a)

\[ \tilde{x}_k = \hat{x}_k + K_k (y_k - M_k \hat{x}_k - N_k \tilde{x}_{k-1}) \]  \hspace{2cm} (2.95a)

\[ P_k = P_k^* - K_k Q_k K_k^T \]  \hspace{2cm} (2.96a)

---

1 The mechanization of Doppler count amounts to an integration of measured satellite-vehicle relative velocity over a specific interval of time and thus introduces a change in relative position from time \( t_{k-1} \) to \( t_k \), thereby involving the delayed state \( x_{k-1} \).
Because of certain physical considerations in the integrated Doppler navigation system, Stuva (34, pp. 5-6) demonstrated that algebraically equivalent modified Kalman filter equations (34, pp. 22-23) which in this case may yield numerically improved results are

\[ Q_k = (M_k P_k^{-T} + V_k) + N_k P_{k-1} N_{k-1}^T + N_k P_{k-1} \delta_k T \ M_{k-1}^T \]

\[ + M_k \delta_k P_{k-1} N_{k-1}^T \]  \hspace{1cm} (2.103a)

Employing the bounding interval arithmetic it is possible to compare the accumulated single-precision truncation solution errors of the two methods as a possible empirical measure of the proposed accuracy improvement. This comparison is included in Chapter V where the interval endpoint and single-precision result errors (with respect to the double-precision solution) for the two methods are discussed.
The Optimal Linear Regulator Design with Minimum Plant Variation Sensitivity

The following result is an excerpt from research by Ciric (35, 36, 37) and the salient features are presented here. Essentially, the concepts involved are those of the optimal linear state-regulator problem where a feedback compensator is optimally designed to minimize a specified cost functional. The optimal compensator design additionally includes a first variation cost functional analysis which allows arbitrary parameters in the optimal compensator to be selected so that the design is least sensitive to variation in certain plant parameters.

Let the linear time-invariant state variable system be given by

\[ \dot{x} = Ax + bu \quad \text{and} \]
\[ y = Cx \quad , \]

where

- $A$ is an $n \times n$ matrix with nominal value $A_0$,
- $C$ is a constant $m \times n$ matrix ($m<n$),
- $b$ is a constant $n$ vector,
- $x$ is the $n$ vector state and
- $u$ is the scalar control input.
Assume that \((A,b)\) is completely controllable and \((A,C)\) is completely observable (38, pp. 499, 502).

Consider the linear optimal regulator problem where the control \(u\) is to minimize the cost functional defined by

\[
I \equiv \int_{0}^{\infty} \left\{ x^TQx + \sum_{i=0}^{p} \gamma_i [u(i)]^2 \right\} dt ,
\]

where

\[ Q \text{ is a constant symmetric positive semidefinite } nxn \]

matrix and the real scalars \(\gamma_i \geq 0, i=0,\ldots,p-1\)

and \(\gamma_p \equiv \gamma > 0\)

\[ u(i) \equiv \frac{d^i u}{dt^i} \text{ and} \]

\(p\) is the smallest integer such that the rank of

\[ [C^T, A^T_0 C^T, \ldots, (A^T_0)^p C^T] = n \]

Let \(u_1 \equiv u, \ u_1 \equiv u_2, \ldots, \ u_1 \equiv u_{p+1} \equiv u \) and

\[ u \equiv (u_1, \ldots, u_p)^T, \text{ the } p \text{ vector. Let } z \equiv (x^T, u^T)^T, \text{ the } n+p \]

vector, with \(z_0 \equiv z(0)\),

\[
\hat{A}_0 \equiv \begin{bmatrix}
A_0 & (B; 0_{nx(p-1)}) \\
0_{pxn} & B
\end{bmatrix}
\]

the \((n+p)x(n+p)\) matrix where
the pxp matrix and let

\[ \hat{b} = (0, \ldots, 0, 1, 1)^T, \]

the n+p vector. Let

\[ \hat{Q} = \begin{bmatrix} Q & 0_{nxp} \\ 0_{pxn} & \Gamma \end{bmatrix}, \]

the (n+p)x(n+p) constant symmetric positive semidefinite matrix, where \( \Gamma = \text{diag}(\gamma_i^0), i=0, \ldots, p-1 \). Then the linear optimal regulator problem may be reformulated as

\[ \dot{z} = \hat{A}z + \hat{b}U, \quad z_0 = z(0) \quad (2.104a) \]

and

\[ J = \int_0^\infty (z^T \hat{Q}z + \gamma u^2) dt. \quad (2.105a) \]

The optimal scalar control \( U \) is given by

\[ U = -k^Tz = -\sum_{i=1}^n k_i x_i - \sum_{i=1}^p k_{n+i} u_i, \quad (2.106) \]
where the \( n+p \) vector \( \hat{k} \) is obtained from

\[
\hat{k} = \frac{1}{\gamma} P_\infty b
\]  

(2.107)

and \( P_\infty \) is the steady state solution of the matrix Riccati differential equation

\[
-\dot{P} = PA_0 + A_0^T P + Q - \frac{1}{\gamma} P\hat{b}\hat{b}^T P, \quad P(0) = O_{(n+p)\times(n+p)}
\]

(2.108)

From the original problem (36, p. 578), the optimal negative feedback compensating transfer function matrix is given by

\[
G_c(s) = [G_1(s), \ldots, G_m(s)]
\]

where

\[1\] With the EISPAC eigenvector-eigenvalue routine and a matrix inversion algorithm, \( P_\infty \) may be found without recourse to integration. Define the \( 2(n+p)\times2(n+p) \) matrix

\[
M \equiv \begin{bmatrix}
\hat{A}_0^T & \hat{Q} \\
\frac{1}{\gamma}\hat{b}\hat{b}^T & -\hat{A}_0
\end{bmatrix}
\]

The matrices \( \hat{Q} \) and \( \frac{1}{\gamma}\hat{b}\hat{b}^T \) are positive semidefinite Hermitian. Since the steady state solution \( P_\infty \) is symmetric positive definite, its solution is given by \( P_\infty = DE^{-1} \), where the \( 2(n+p)\times(n+p) \) matrix \( D \) consists of the \( (n+p) \) eigenvectors which correspond to the unique \( (n+p) \) eigenvalues with positive real parts (39, p. 498).
\[ G_j(s) = \frac{-u(s)}{y_j(s)} = \sum_{i=0}^{p} \beta_i^j s^i / (s^p + \sum_{j=0}^{p-1} \alpha_j^i s^j), \quad j=1, \ldots, m \]

(2.109)

and thus (2.106) may also be written as

\[ u = - \sum_{i=0}^{p-1} \alpha_i u_{i+1} - \sum_{i=0}^{p} \sum_{j=1}^{m} \beta_i^j y_j(i), \quad (2.110) \]

where

\[ y_j(i) = c_j^T x(i), \]

c_j^T is the jth row of C and x(i) is the ith time derivative of x. Then

\[ \sum_{i=0}^{p} \sum_{j=1}^{m} \beta_i^j c_j^T A_i^x = \sum_{i=1}^{n} k_i x_i \]

(2.111)

and

\[ \sum_{i=1}^{p} \alpha_i-1 u_i + \sum_{i=1}^{p} \sum_{j=1}^{m} \beta_i^j c_j^T A_i^{k-1} u_k = \sum_{i=1}^{p} k_i+1 u_i \]

(2.112)

In matrix form, (2.111) may be rewritten as
This represents \( n \) equations in \( m(p+1) \) unknowns. If \( m(p+1) > n \), then \( m(p+1)-n = k \) of the \( \beta_1^j \)'s are arbitrary and the system is still optimal. If (2.113) does not have arbitrary \( \beta_1^j \)'s, they may be realized by increasing the value of \( p \). Although the compensated system is optimal, it may be very sensitive to plant parameter variations if the arbitrary \( \beta_1^j \)'s are not properly chosen. (The \( a_i \)'s are defined by Equation 2.112.)

Assume that \( l \) plant parameters in \( A \) may vary and that \( k \) arbitrary \( \beta_1^j \)'s are to be determined. Let the \( l \) elements of \( A \) be denoted by the \( l \) vector \( a \) with nominal value \( a_0 \). As a result of the first-order variational analysis (with respect to \( a \)) of the cost functional, the \( k \) arbitrary \( \beta_1^j \)'s are assigned values so that the optimal design is least sensitive to the plant parameter variations \( a \).

Let the \( k \) vector \( \hat{\beta} \) represent the arbitrary \( \beta_1^j \)'s and
the vector \( \tilde{\beta} \), the \( n \) dependent \( \beta_i^j \)'s. Then (2.113) may be partitioned and rewritten as

\[
[S\bar{S}] \begin{bmatrix} \tilde{\beta} \\ \hat{\beta} \end{bmatrix} = \begin{bmatrix} k_1 \\ \vdots \\ k_n \end{bmatrix} \equiv k \quad \text{or} \quad \tilde{\beta} = (\bar{S})^{-1}(k - \hat{S}\hat{\beta}) \, , \tag{2.114}
\]

where \( \bar{S} \) and \( S \) are respectively \( nxn \) and \( nxk \) matrices.

Now let \( A \) represent the plant with the assumed parameter variation \( a \). Define

\[
\hat{A} \equiv \begin{bmatrix} A & B^T 0 \\ 0 & B \end{bmatrix} 
\]

With \( \hat{k} \) (and therefore \( k \), see Equations 2.111 and 2.113) determined by (2.107), on a term by term comparison of (2.106), (2.110), (2.111) and (2.112) and employing (2.114), the control \( U \) may be written in the form

\[
U = -[\hat{k}^T + \hat{\gamma}^T(\hat{A} - \hat{A}_0) + \hat{\beta}^T K^T(\hat{A} - \hat{A}_0)]Z \, , \tag{2.115}
\]

where \( \hat{\gamma} \) is an \( n+p \) constant vector and \( \hat{k} \) is an \( (n+p) \times k \) constant matrix.\(^1\) Equation (2.115) then represents the

\(^1\)Simple analytic expressions for \( \hat{\gamma} \) and \( \hat{k} \) are not possible. However, there is considerably more organization to their determination than a causal perusal here would indicate and the reader is referred to the literature (37, p. 853).
plant parameter variation dependent optimal control $U$ with the arbitrary constant vector $\mathbf{\hat{p}}$.

The "sensitivity-optimal" value of $\mathbf{\hat{p}}$, $\mathbf{\hat{p}^*}$, is selected on the basis of the first-order variational design analysis and may be computed algebraically. Since 3-tuple tensor multiplication is sometimes involved, the tensor multiplicative notation will appear following the definitions whenever these occur (40, p. 119).

\begin{equation}
\mathbf{\hat{p}^*} \equiv - (D^T D)^{-1} D^T d
\end{equation}

\begin{equation}
D^T \equiv \hat{K}^T \left( \frac{\partial \hat{A}}{\partial a} \right)_{a_0} \mathbf{\hat{p}Mv},
\end{equation}

\begin{equation}
(D^T)_{ij} \equiv (\hat{K}^T)_{ik} \left( \frac{\partial \hat{A}}{\partial a} \right)_{klj} (\hat{p})_{lm} (\hat{M})_{mn} (v)_n,
\end{equation}

\begin{equation}
\frac{\partial \hat{A}}{\partial a} \equiv \frac{\partial (\hat{A})_{klj}}{\partial a_j}, \quad a_j \in a,
\end{equation}

\begin{equation}
\hat{p}^{-1} (\hat{A}_0 - \hat{A} \hat{K}^T) \hat{p} \equiv \Lambda \equiv \text{diag}(\lambda_i),
\end{equation}

\begin{equation}
\hat{M} \equiv \text{diag} \left( \sum_{i}^{n+p} \frac{v_j w_j}{\lambda_i + \lambda_j} \right).
\end{equation}

---

1Strictly speaking, the control $U$ is optimal only when the plant assumes its nominal value $\hat{A}_0$.

2Assume here that the $n+p$ $\lambda_i$'s are distinct.
In summary then, the optimal compensator (2.109) or equivalently the optimal control (2.115) solves the linear optimal regulator problem (2.104), (2.105) or equivalently (2.104a), (2.105a) for the nominal plant and if the "sensitivity-optimal" \( \hat{p}^* \) is computed by (2.116), the cost functional is least sensitive to plant parameter variations on a first-order variational design basis.

The actual compensator implementation (36, p. 586) has not specifically been explored here since any numerical modeling can be accomplished using (2.104a), (2.105a),
(2.115) and (2.116).

Using the linear interval integration algorithm developed in Chapter IV, the state variable responses for a design example incorporating the above results (36, p. 584) for specified plant parameter variations are compared when \( \hat{p} = \hat{p}^* \) and when \( \hat{p} \) is not selected to satisfy the minimum sensitivity criterion. This comparison is included in Chapter V.
CHAPTER III. NUMERICAL BOUNDING INTERVAL ARITHMETIC IMPLEMENTATIONS

The Washington State - Mathematics Research Center (University of Wisconsin) Package

In order to perform bounding interval calculations from a Fortran language program, it is necessary to have available the software routines to accomplish the task.

The Mathematics Research Center of the University of Wisconsin had constructed such a package (2, p. 14) and upon contacting the center it was ascertained that their routines were designed for a CDC-1108 computer and were not compatible with the IBM System/360 (41). However, the Washington State University Computation Center had converted the package for their IBM System/360 Model 67 machine and they kindly furnished a 9-track 800 BPI magnetic tape listing of the modified routines as well as the necessary documentation (42, 43, 44).

The Washington State package essentially consisted of two partitioned data sets. The first set had only one member which was a Fortran "precompiler" (called CLUDGE). Its purpose was to interpret (precompile) the original Fortran source program and produce a new source "deck", converting type INTRVAL statements to COMPLEX and replacing operations on type INTRVAL variables with subroutine calls and function
references. The second data set was a library of subroutines and function subprograms necessary to execute the precompiled source program (43, pp. C1-C7). The precompiler consisted of a main program, two block data subprograms, 54 subroutines and nine function subprograms (5148 Fortran statements) and three assembler routines (84 statements). The library consisted of seven assembler routines (848 statements) and 15 subroutines, one block data subprogram and seven function subprograms (1219 Fortran statements), with a total of 40 library aliases (that is, multiple entry points for different operation definitions).

In order to efficiently use the package\(^1\) it was compiled and placed in two object modules on the SYSLIB disk pack (requiring approximately 35 tracks of storage).

Although the precompiler and library of routines operated correctly, certain functional disadvantages became immediately apparent. The precompiler output source deck was frequently inefficiently constructed.

Each interval arithmetic operation was performed by

\(^1\)Upon receipt of the listing, it was necessary to reconstruct 40 lines of lost program listing (due to a tape fold), correct various functional errors in the diagnostics routine and generate an assembler language function DINT which was available in the Washington State "H-extended" compiler but was not available in the Iowa State University "H" compiler.
a subroutine call and an intermediate storage location was required for the result. Also, for each real arithmetic operation within the interval arithmetic subroutine, control was passed to a "best possible" floating-point arithmetic assembler routine, then to a bounding arithmetic assembler routine and finally back to the interval arithmetic subroutine. The final interval arithmetic subroutine control step was to pass control to a diagnostics routine, return and then store the results (assuming that no diagnostics were required).

Floating-point computations in the "best possible" floating-point real arithmetic routines were partially accomplished in the floating-point registers. The signs, exponents and fractions of the floating-point operands were separated and loaded into main storage locations and the fraction arithmetic operations were performed by long un-normalized additions or subtractions in the double word length floating-point registers in which the highest-order word was initially cleared. Since a single-precision fraction consists of six hexadecimal digits, the lowest-order two hexadecimal digits of the low-order word were also initially zero. As an example, fraction multiplication was accomplished by recursive hexadecimal additions and intermediate result right shifts as the addition index moved upward from the lowest-order hexadecimal digit in the multi-
plier. A residue indicating flag was set whenever result digits were right shifted out of the low-order word so that the final bounding decision was made correctly when control was passed to the bounding routine.

It should be apparent from the few particulars mentioned above that the "best possible" floating-point real arithmetic operations included in the interval arithmetic subroutines were indeed very time consuming since they attempted to software mechanize the floating-point arithmetic operations in the manner in which they were accomplished by the floating-point arithmetic machine hardware. Also time consuming was the multiplicity of subroutine calls generated for each interval arithmetic operation.

Initial testing of the precompiler with the 15 line Fortran factorial test program (43, p. C5) indicated that approximately 134K bytes of main core storage were necessary to execute the precompile step and that the final object code CPU execution time would be more than 100 times the ordinary single-precision CPU execution time.

Because of the excessive execution times necessary to operate the Washington State bounding interval arithmetic routines, it became necessary to generate a compromise

---

1 By programming directly in terms of COMPLEX type variables and using subroutine calls and function subprogram references, it was possible to omit the precompile step.
bounding interval arithmetic implementation and that is the subject of the next section.

The Fast Assembler Routines

To obviate the excessive CPU execution times necessary to operate the Washington State bounding interval arithmetic routines, thereby satisfying financial constraints, assembler language COMPLEX functions were devised as an alternative.

Two COMPLEX function routines were generated, ADD(A, B) and MUL(A, B), with respective multiple entry point function definitions SUB(A, B) and DIV(A, B). As contrasted with the Washington State routines, an intermediate storage variable is not required. Additionally, the COMPLEX functions themselves may be used as arguments within the COMPLEX functions, which reduces the Fortran programmer's task and allows the

---

1 When used in the Fortran programs, the function result definitions ADD, SUB, MUL and DIV as well as the arguments A and B are intervals represented in the Fortran COMPLEX convention with the real part corresponding to the single-precision interval left endpoint and the imaginary part corresponding to the single-precision interval right endpoint. The bounding interval arithmetic operations are obvious from the function names.

2 For example, the Fortran bounding interval arithmetic statement which corresponds to $A = (A/B + A) \cdot (A - A \cdot B)$ is written as the single expression

$$A = \text{MUL}(\text{ADD}(\text{DIV}(A, B), A), \text{SUB}(A, \text{MUL}(A, B)))$$

Using the Washington State routines, this statement would be
optimizing Fortran H compiler to function more efficiently, producing smaller program object storage as well as faster program execution.

A contrast in the CPU execution time was obtained in the case of the delayed state covariance solution method comparison of Chapter V. The H compiled "hand written" program employing the Washington State routines required 527.35 seconds of CPU execution time while the H compiled program using the assembler language COMPLEX functions used only 30.94 seconds of CPU execution time, a ratio of improvement of greater than 17:1.

The COMPLEX function ADD (with multiple entry SUB) required 52 assembler language statements and the COMPLEX function MUL (with multiple entry DIV) requires 138 assembler language statements. An interval division minimally "handwritten" as

CALL INTDIV(A,B,D1),
CALL INTADD(D1,A,D2),
CALL INTMUL(A,B,D1),
CALL INTSUB(A,D1,D3),
CALL INTMUL(D2,D3,D1) and
A=D1,

where the five subroutine call definitions are obvious (INTerval DIVision, etc.), D1, D2 and D3 are nonarray temporary (intermediate) COMPLEX storage locations and A=D1 is a simple COMPLEX arithmetic assignment. If the Washington State precompiler were used, temporary COMPLEX array storage locations (INTerval TEMporary) INTTEM(1),...,INTTEM(5) would be required and the last statement would be written with the interval store subroutine as, CALL INTSTR(INTTEM(5),A).
diagnostics trap is incorporated within the latter function.

The COMPLEX functions MUL and DIV produce bounding interval arithmetic results which are identical to those of the Washington State routines while the COMPLEX functions ADD and SUB provide a pseudo-bounding interval arithmetic result.

In order to discuss the operation of the assembler language bounding interval arithmetic implementations, it is necessary to further describe the single- and double-precision floating-point arithmetic operations on the IBM System/360 Model 65 computer (16, pp. 301-307). These operations are performed in the arithmetic and logic subsystem of the computer which contains a set of 16 general-purpose four byte (32 bit) registers (addressed here as R0,...,R15) and another set of four floating-point eight byte (64 bit) registers (addressed here as 0, 2, 4 and 6).

To describe the single-precision (short) floating-point addition instruction, assume that the eight hexadecimal digit addends have been loaded from main storage into the high-order 32 bits of floating-point registers 0 and 2 and that the machine instruction is to form their sum in 0. First, the two eight digit addends are moved to the high-order 32 bits of two nine digit (36 bit) addition hardware registers where the cleared low-order extra hexadecimal digit (four bits) is called the "guard" digit. Next, a
comparison of the addend exponents is made and the smaller exponent addend fraction is right shifted the number of digits equal to the difference (called the alignment step). Then the fractions are added, the larger exponent is inserted and the correct sign is assigned. If the addition produces a "carry" out of the high-order sum fraction digit, the sum fraction is right shifted one digit, a hexadecimal 1 is inserted in the high-order sum fraction digit and its exponent is increased by one. If the high-order fraction sum digit is zero, the sum fraction is left shifted one digit (a hexadecimal 0 is inserted in the low-order digit in this left shift process) and the sum exponent is correspondingly decreased by one. This "normalization" process is completed when the high-order fraction digit is nonzero (assuming the sum fraction is nonzero). It should be noted that in this case the guard digit becomes one of the sum fraction digits. Finally, the truncated high-order eight sum digits are moved to the high-order digits of floating-point register 0.

In the double-precision (long) floating-point addition instruction, the 16 digit addends 0 and 2 are moved to 17 digit hardware storage registers, the addition is performed and the truncated 16 digit sum is moved to floating-point register 0. Single- and double-precision floating-point subtraction instructions may be understood by analogy with
the respective addition instructions.

It is now possible to describe the operation of the pseudo-bounding single-precision interval arithmetic COMPLEX functions ADD and SUB.

Consider the interval addition

\[ [A(1), A(2)] + [B(1), B(2)] \equiv [A(1) + B(1), A(2) + B(2)] \]  

(3.1)

and specifically the right endpoint sum \( A(2) + B(2) \). Suppose that the subsequent addition instruction will add registers 2 and 6 with the sum placed in register 2.\(^1\)

First, the floating-point registers are totally cleared.\(^2\)

Then, the single-precision addends are loaded short from general storage into registers 2 and 6. Next, the long addition is performed and the 16 digit sum is placed in

\(^1\)Recall that the COMPLEX storage is used for the interval variables. The Fortran G and H compiler COMPLEX function linkage convention requires that the COMPLEX result be placed in floating-point registers 0 and 2 and for this reason, the interval right endpoint sum is placed in floating-point register 2.

\(^2\)This is initially done for all four floating-point registers. A double-precision zero is loaded from main storage into 0, then long, register to register load operations are performed, 0 to 2, 0 to 4 and 0 to 6. This requires 4.55 \( \mu \) seconds of execution time as opposed to loading each of the registers 0, 2, 4 and 6 directly from main storage which requires 5.60 \( \mu \) seconds of execution time (45, pp. 31-35). The remark is made to indicate one of many steps taken to minimize execution time.
register 2. The following five-step example illustrates the concepts involved thus far:

(a) eight digit general storage addends
\[ A(2) = 41A205FF \]
\[ B(2) = 31125729 \]

(b) cleared, then loaded short, floating-point registers
\[ 2 = 41A205FF00000000 \]
\[ 6 = 3112572900000000 \]

(c) long addition hardware registers (addends aligned)
\[ \sim 2 = 41A205FF00000000(0) \]
\[ \sim 6 = 4100000000001257(2) \]

\( \sim \) denotes the hardware register corresponding to the floating-point register.

\( \cdots \) denotes the hardware guard digit. The hexadecimal digit 9 is lost in the alignment of \( \sim 6 \).

(d) normalized hardware sum
\[ 41A205FF00001257(2) \]

(e) truncated 16 digit sum placed in register 2
\[ 2 = 41A205FF00001257 \]

The assembler functions ADD, SUB, MUL and DIV have access to a double-precision (64 bit) 16 hexadecimal digit named common variable called AUGMNT. Assume that this variable has been hexadecimally initialized in the Fortran
"calling" program to 00000000FFFFFFFFFFF.

Since the right endpoint in (e) above is positive, the sign and exponent byte of register 2 (digits 41) is moved to the high-order byte of AUGMNT. Then, the long addition sum of 2 and the modified variable AUGMNT is placed in 2. This is illustrated in step (f) below:

(f) "single-precision" bounding of the positive right interval endpoint sum

\[
2 = 41A205FF00001257
\]

(modified) AUGMNT = 41000000FFFFFFFF

(bounded) sum 2 = 41A2060000001256

Thus the high-order eight digits (single-precision word) of the result in floating-point register 2 represents the pseudo-bounding single-precision interval arithmetic sum right endpoint.

If the sum in (e) had been negative or zero, the augmenting step (f) would have been bypassed since then the high-order eight digits would automatically be an upper bound for the 16 digit right endpoint sum.

The left endpoint sum is obtained in a similar manner using floating-point registers 0 and 4, placing the sum in 0, where the augmenting step will be omitted if the sum corresponding to step (e) is positive or zero.

The assembler language function SUB (multiple entry point SUB in function ADD) operates with the same software
routine except that initially \(-B(1)\) is loaded short into 6 and \(-B(2)\) is loaded short into 4.

Thus the COMPLEX functions ADD and SUB are pseudo-bounding interval arithmetic implementations in the sense that if for example the alignment in step (c) above had produced a zero fraction in the hardware floating-point register \(~6\), the augmented single-precision interval right endpoint sum in floating-point register 2 of step (f) would remain 41A205FF00000000, since the augmenting step would not have "seen" the digits right shifted out of the hardware register during alignment. (There are of course many cases where these pseudo-results will be identical to those produced by the Washington State bounding interval addition and subtraction subroutines, INTADD and INTSUB.)

The named common variable AUGMNT of the bounding interval arithmetic COMPLEX functions ADD, SUB, MUL and DIV does provide the Fortran programmer with options which would not be available in the Washington State routines. By initializing AUGMNT to 0000000088888888, the functions would result in a rounding interval arithmetic. By initializing AUGMNT with a chain of hexadecimal F's or 8's of length one through eight beginning in digit position 9, the programmer could preselect the "level of conservativeness" with which the bounding or rounding single-precision interval
arithmetic was to operate. It would also be possible for the Fortran programmer to vary the "level of conservativeness" with which the bounding or rounding interval arithmetic was performed at various points within a program.

The assembler language listing of ADD(SUB) and an abbreviated flowchart are included in Appendix A.

Consider now the single-precision (short) floating-point multiply instruction where it will be assumed that the eight digit multiplicand and multiplier have been loaded from main storage into the high-order 32 bits of floating-point registers 0 and 2 respectively and that the machine instruction is to form the product in 0. The multiplication step involves, first an exponent addition and subtraction of the bias quantity (hexadecimal 40), a product sign determination and a multiplication of the six

---

1By selecting AUGMNT = 000000000000000000, the COMPLEX functions would produce (essentially) degenerate interval (real) single-precision floating-point arithmetic.

2It is possible to generate assembler language COMPLEX functions ADD, SUB, MUL and DIV which (when the object decks are placed immediately following the WATFIV source deck) will operate properly with the WATFIV compiler, as well as with the G and H compiler. This was in fact done, but the reader should be warned that because of the additional complexity necessary to accept both WATFIV and G and H compiler COMPLEX function subprogram argument conventions, only the simpler G and H assembler language programs are listed in the appendix and they will not operate correctly with the WATFIV compiler.
digit fractions. The full normalized product (two digit exponent and sign and 12 digit fraction product) is returned to floating-point register 0 from the multiplication hardware registers. Thus at least the low-order two digits of the 16 digit register 0 will be zero. It should be noted that the product post-normalization will involve at most a single digit left shift of the fraction and a corresponding reduction of the exponent by one.

The double-precision (long) floating-point multiply instruction results in a truncated 17 digit product (including the guard digit) within the hardware section, post-normalization as described above and return of the truncated 16 digit product to floating-point register 0.

It is now possible to describe the implementation of the single-precision bounding interval COMPLEX function MUL. The interval product is defined by

\[
[A(1), A(2)] [B(1), B(2)] \equiv \\
[\min\{A(1)B(1), A(1)B(2), A(2)B(1), A(2)B(2)\}, \\
\max\{A(1)B(1), A(1)B(2), A(2)B(1), A(2)B(2)\}].
\]  

First the four floating-point registers are cleared long. Then B(1) is loaded short into 0 and 0 is loaded long into a double-precision variable in main storage, TEMPI. Then B(2) is similarly loaded into the variable TEMP2. Next registers 0 and 2 are loaded short with A(1) and registers 4
and 6 are loaded short with $A(2)$. Long multiply instructions form the long floating-point products

\begin{align*}
0 &= 0 \cdot \text{TEMP1} = A(1)B(1), \\
2 &= 2 \cdot \text{TEMP2} = A(1)B(2), \\
4 &= 4 \cdot \text{TEMP1} = A(2)B(1) \text{ and} \\
6 &= 6 \cdot \text{TEMP2} = A(2)B(2).
\end{align*}

Then the minimum and maximum of the four products are stored long in registers 0 and 2 respectively. (See the MINIMAX assembler flowchart, Figure A-3, in the Appendix.) Following this, the augmenting process described for the ADD(SUB) COMPLEX function is performed, if necessary, and the high-order eight digits of registers 0 and 2 contain the bounded interval product. It is noteworthy that if $\text{AUGMNT} = 00000000\text{FFFFFF00}$ is the initialization by the Fortran program, the bounded interval product will in every case be identical to the product obtained by the Washington State routines.

The single-precision (short) floating-point divide instruction yields a divide hardware register quotient with a fraction consisting of six digits and a carry digit. The double-precision (long) floating-point divide instruction yields a divide hardware register quotient with a fraction consisting of 14 digits and a carry digit. In either case, if a nonzero carry digit results, it is necessary to right shift the fraction one digit, insert the quotient carry
digit in the high-order fraction digit and increase the exponent by one. Then the short (eight digit) or long (16 digit) quotient is returned to the floating-point register.

The operation of DIV (multiple entry point DIV of complex function MUL) is the same as that of MUL through the completion of loading B(1) in TEMPl and B(2) in TEMP2. At this point it is necessary to insure that the divisor interval does not contain points "too close" to zero. For this purpose, the assembler function MUL(DIV) has access to a single-precision variable called EPSDVR, hexadecimally initialized by the calling Fortran program. If

\[ B(2) < -\text{EPSDVR} \text{ or } \text{EPSDVR} < B(1), \]

operation continues. If not, an error trap is triggered and the error diagnostics message

"** DIVFLT * PGM TERMINATED **"

is printed in addition to the short (eight digit) hexadecimal representation of B(1) and B(2) and the execution of the Fortran program is terminated.

If the diagnostics trap is not triggered, the loading of A(1) and A(2) described for MUL takes place and the routine branches to the long divide instruction sequence.

\(^1\) That is, if \([B(1), B(2)] \bigcap [-\text{EPSDVR}, \text{EPSDVR}] \) is the void intersection, operation continues.
The interval quotient is defined by
\[
[A(1), A(2)] / [B(1), B(2)]
\]
\[
\equiv [A(1), A(2)] [1/B(2), 1/B(1)]
\]
\[
\equiv \left[ \min\{A(1)/B(2), A(1)/B(1), A(2)/B(2), A(2)/B(1)\} , \max\{A(1)/B(2), A(1)/B(1), A(2)/B(2), A(2)/B(1)\} \right] .
\]
(3.3)

Therefore, the long divide instruction sequence forms the long floating-point register quotients

\[
0 = 0/\text{TEMP2} = A(1)/B(2) ,
\]
\[
2 = 2/\text{TEMP1} = A(1)/B(1) ,
\]
\[
4 = 4/\text{TEMP2} = A(2)/B(2) \text{ and}
\]
\[
6 = 6/\text{TEMP1} = A(2)/B(1) .
\]

From this point forward the operation of DIV is the same as MUL. However, one last remark must be made concerning the augment (bounding) step. The long dividend and divisor each contain fractions with eight low-order zero digits. If each of the four division fraction quotients were computed to more than 15 digits, 14 digits and one carry digit, it could never occur that, even after a possible one digit right shift to incorporate a carry digit, at least one of the quotients will be of the form

\[
e_1 e_2 x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8 x_9 x_{10} x_{11} x_{12} x_{13} x_{14} x_{15} x_{16} x_{17} \cdots .
\]

Here, the e's represent the sign and exponent byte, x₁ is
nonzero, at least one of the digits \(x_{15}, x_{16}, \ldots\), is nonzero, but \(0_7\) through \(0_{14}\) are the zero digits. Thus any bounding augment step using \(AUGMNT = 00000000FFFFFFFFFF\) will produce short quotients in the floating-point registers 0 and 2 which in fact bound even the infinite decimal quotients. For this reason, the bounding interval quotients produced by DIV will in every case be the same as the quotients obtained by the Washington State routines.

The assembler language listing of MUL(DIV) and an abbreviated flowchart are included in Appendix A.

To summarize, the COMPLEX functions ADD and SUB produce pseudo Washington State bounding interval arithmetic operations while MUL and DIV produce exact Washington State bounding interval arithmetic operations (assuming AUGMNT is initialized appropriately).

In order to efficiently use the COMPLEX bounding interval arithmetic functions, the two assembler language routines were assembled and machine language object decks were produced. The COMPLEX functions were then available to the Fortran program by placing these decks immediately after the linkage editor job control card.
CHAPTER IV. NUMERICAL INTERVAL METHODS FOR
INITIAL-VALUE PROBLEMS IN ORDINARY
DIFFERENTIAL EQUATIONS INVOLVING A PARAMETER

Autonomous Systems of Nonlinear First-Order Equations

At the inception of the research for this dissertation, an attempt was made at a direct extension of Moore's interval methods for initial-value problems of nonlinear ordinary differential equations (2, pp. 100-118) to problems in which there is continuous dependence on a perturbation parameter. The results were rather disappointing. The class of problems initially considered was the set of autonomous systems of \( n \) first-order nonlinear differential equations with rational right-hand sides\(^1\) which have continuous partial derivatives on a given domain for all values of the perturbation parameter and therefore satisfy

\^[1]The restriction to autonomous rational right-hand sides includes a larger class of functions than it may appear at first glance. For example, Moore (2, p. 109) demonstrates that by introducing new variables, the differential equation

\[
\dot{x} = f(x) = \begin{bmatrix}
  t^2 + x_1^2 \cos x_2 \\
  e \\
  x_1 \ln x_2
\end{bmatrix}
\]

may be redefined as a larger autonomous system of differential equations with right-hand sides that are rational in seven variables. By introducing a single additional variable it is always possible to form an autonomous system of first-order differential equations from one which is heteronomous.
the standard theorems on uniqueness and existence of solutions as well as the continuity and differentiability of the solutions with respect to the parameter and the initial condition (46, pp. 159-181).

A Fortran program for the Kth-order interval method was written for the perturbation parameter problem, but only a heuristic description of the method, program and the resulting deficiencies will be discussed here. (The substance of this chapter is contained in the following section which is devoted to extending a modification of Moore's Kth-order interval method to the class of linear autonomous perturbation parameter problems.)

The Kth-order interval method for the perturbation parameter problem may be derived in the following manner. Let \( \phi(t,a) \) be the unique solution for the initial-value problem of the autonomous system of \( n \) first-order nonlinear differential equations

\[
\dot{x} = f(x,a), \quad a \in \mathbb{I},
\]

and

\[
x(0) = x_0, \quad (4.1)
\]

where \( f \) is continuous on a given domain \( \Gamma \) and \( x_0, t=0 \) and \( a \in \mathbb{I} \) are contained in \( \Gamma \). Assume also that \( f \) is continuously differentiable on \( \Gamma \) up to and including the \((K-1)\)st-order
mixed partials with respect to $x$. Consider the Taylor series expansion of the solution $\varphi(t, a)$,

$$\varphi(t, a) = \varphi^{(0)}(0, a) + \varphi^{(1)}(0, a)t + \ldots$$

$$+ \varphi^{(K-1)}(0, a) \frac{t^{K-1}}{(K-1)!} + \varphi^{(K)}(\xi, a) \frac{t^K}{K!},$$

where

$$\varphi^{(m)}(0, a) \equiv \left. \frac{d^m}{dt^m} \varphi(t, a) \right|_{t=0}, \quad m=0, 1, \ldots, K$$

and

$$0 < \xi < t.$$  

(4.2)

Let

$$\hat{f}^{(l)}(\varphi(0, a), a) \equiv \left. \frac{d^l}{dt^l} \hat{f}(\varphi(t, a), a) \right|_{t=0}, \quad l=0, 1, \ldots, K-1.$$

---

1 That is, the partial derivatives

$$\frac{\partial^j f_i}{\partial x_1^{n_1} \partial x_2^{n_2} \ldots \partial x_n^{n_n}}(x, a)$$

exist for all $x$ and $a$ in $\Gamma$, where

$$\sum_{k=1}^{n} n_k = j, \quad i=1, 2, \ldots, n \text{ and } j=1, 2, \ldots, K-1.$$

2 Note that for each $a \in \Gamma$,

$$\varphi^{(0)}(0, a) \equiv \varphi(0, a) \equiv x_0 \in \mathbb{R}^n.$$
Then (4.2) may be rewritten as

\[ \phi(t, a) = \phi(0, a) + \sum_{j=1}^{K-1} \phi(j-1)(\phi(0, a), a) \frac{t^j}{j!} \]

\[ + \phi(K-1)(\phi(0, a), a) \frac{t^K}{K!} . \]  

(4.3)

For the class of problems considered above, since \( f \) is rational in \( x \) and \( a \), the \( \phi(\ell) \), \( \ell = 0, 1, \ldots, K-1 \) are rational in \( x \) and \( a \). Then, as rational interval functions in the interval vector variable \( X \) and the interval variable \( I \),

if \( X_1 \subset X \) and \( I_1 \subset I \),

\[ \phi(\ell)(X_1, I_1) \subset \phi(\ell)(X, I) , \quad \ell = 0, 1, \ldots, K-1 \]

and in this sense the \( \phi(\ell) \)'s exhibit the property of monotonic inclusion.\(^1\)

The Kth-order interval method is to initially determine \( t_1 = h/2q_1 \)\(^2\) and an interval vector \( X_1 \) which are contained in \( \Gamma \), such that the interval result is computed

---

\(^1\) Since this section is meant to be largely heuristic, a complete mathematical setting will not be given here (2, p. 18). By analogy with (2.6) and Corollary 2.9, the interval vector case should be obvious.

\(^2\) \( h \) is the programmer's input choice of the maximum allowed algorithm time increment for the problem and \( q_1 \) is a positive integer which is determined by the algorithm. The algorithmic determination of \( q_1 \) will be discussed later.
by

\[ \tilde{\theta}(t_1, I) = \tilde{\theta}(0, I) + \sum_{j=1}^{K-1} f^{(j-1)}(\tilde{\theta}(0, I), I) \frac{t_1^j}{j!} + f^{(K-1)}(X_1, I) \frac{t_1^K}{K!}, \]  

where

\[ \{\tilde{\theta}(0, a) | a \in I\} \subset \tilde{\theta}(0, I) \subset X_1 \]

and

\[ \tilde{\theta}([0, t_1], I) \subset X_1 . \]

Then the set of solutions satisfies the relationship

\[ \{\tilde{\theta}(t_1, a) | a \in I\} \subset \bigcup_{a \in I} \tilde{\theta}(t_1, a) \subset \tilde{\theta}(t_1, I) . \]

If, for example, initially \( X_1 \) has been determined with

\( q_1 = 3 \), a hypothetical sequence of solution values which the algorithm could recursively compute by extending the concepts of (4.4) is

\[ \tilde{\theta}(t_1, I) = \tilde{\theta}(t_0 + \frac{h}{g}, I) = \tilde{\theta}(0 + \frac{h}{g}, I) = \text{function of } \{\tilde{\theta}(0, I), \frac{h}{g}, X_1\} . \]

---

1This is necessary so that for any \( \tilde{t}, 0 < \tilde{t} < t_1 \) and for each \( a \in I \),

\[ f^{(K-1)}(\tilde{\theta}(\tilde{t}, a), a) \subset f^{(K-1)}(\tilde{\theta}(\tilde{t}, I), I) \subset f^{(K-1)}(\tilde{\theta}([0, t_1], I), I) \subset f^{(K-1)}(X_1, I) . \]
\[ \tilde{\mathcal{G}}(t_2, I) = \tilde{\mathcal{G}}(t_1 + h, I) = \tilde{\mathcal{G}}(\frac{h}{2} + h, I) \]

= function of \( \{ \tilde{\mathcal{G}}(t_1, I), h, x_2 \} \),

\[ \tilde{\mathcal{G}}(t_3, I) = \tilde{\mathcal{G}}(t_2 + \frac{h}{2}, I) = \tilde{\mathcal{G}}(\frac{3h}{2} + \frac{h}{2}, I) \]

= function of \( \{ \tilde{\mathcal{G}}(t_2, I), \frac{h}{2}, x_3 \} \),

etc.,

where the \( x_i \)'s and the \( q_i \)'s satisfy

\[ \tilde{\mathcal{G}}(\left[t_{i-1}, t_{i-1} + \frac{h}{2} \right], I) \subset \tilde{x}_i \quad \text{and} \quad \tilde{\mathcal{G}}(t_{i-1}, I) \subset \tilde{x}_i \quad . \]

(4.5)

The method used to compute the \( x_i \)'s and the \( q_i \)'s so that (4.5) is satisfied may be understood by referring to the flow chart of Figure 4.1 for the computation of \( \tilde{\mathcal{G}}(t_i, I) \). Attempts to satisfy (4.5) are accomplished by alternately redefining \( x_i \) and replacing \( hh \) by \( hh/2 \). In the actual algorithm, the initialization of \( hh = h \) as indicated on the flow chart is not precisely correct but it is shown that way for the purpose of simplification. The actual initialization,

\[ hh = \frac{q_i}{2} \]

where
Figure 4.1. Flowchart for the $\tilde{\phi}(t_i, I)$ computation of (4.4)
is accomplished by a computational method devised by Moore (2, pp. 101-102) in an attempt to keep

\[
\max_{\hat{\xi}_\varepsilon[t_{i-1}, t_i]} \{w(\tilde{\varnothing}(\hat{\xi}, I))\} = \max_{\hat{\xi}_\varepsilon[t_{i-1}, t_i]} \{\max_j w(\tilde{\varnothing}_j(\hat{\xi}, I))\}
\]

approximately equal to

\[
\varepsilon \cdot \max_{\hat{\xi}_\varepsilon[t_{i-1}, t_i]} \{|\tilde{\varnothing}(\hat{\xi}, I)|\} = \varepsilon \cdot \max_{\hat{\xi}_\varepsilon[t_{i-1}, t_i]} \{\max_j |\tilde{\varnothing}_j(\hat{\xi}, I)|\}
\]

where \(\varepsilon\) is a small positive real number and \(w(\cdot)\) and \(|\cdot|\)
are respectively the "width" and "magnitude" interval operations defined previously.\(^2\) In any event, if \(\xi_h\) and \(X_i\)
are selected so that (4.5) is satisfied, the solutions computed by (4.4) satisfy the relationship

---

\(^1\)The integer MAXDK is an input to the algorithm which limits the number of times the "DOWNCRANK" (see flowchart) halving of \(\xi_h\) may be performed in attempts to satisfy (4.5). It should also be pointed out that because of the way \(X_i\) is initially computed, \(\tilde{\varnothing}(t_{i-1}, I) \subset X_i\) will always be satisfied and that part of (4.5) need not be tested in the algorithm.

\(^2\)This is at best a qualitative interval error controlling technique. In fact, its derivation is not necessarily valid for the class of problems considered by Moore (2, pp. 101-102) and for the class of perturbation parameter problems initially investigated here, it is of doubtful value.
\[ \{ \tilde{\phi}(t_i, a) \mid a \in I \} \subset \tilde{\phi}(t_i, I) \quad , \quad (4.6) \]

for each \( t_i \).

Without including further programming details, the algorithm finally devised sequentially introduced the differential equation right-hand sides as a recursive sequence of sets of three integers, identifying two operands and a rational binary arithmetic operation. Interval parameter coefficients were similarly identified and the problem was subdivided by parameter interval subdivision, with each interval solution value obtained by taking the "interval unions" over the \( j=1, \ldots, M \) subproblem solutions. The rational \( j \text{th} \) subproblem interval Taylor coefficients

\[ f^{(l)}(\tilde{\phi}(t_{i-1}, I_j), I_j) , \quad l=0,1,\ldots,K-2 \quad \text{and} \]
\[ f^{(K-1)}(x_i, I_j) \quad , \]

were programmed to be machine generated for each \( j=1, \ldots,M \) subproblem by recursively using Leibnitz rule operators coded for the specific rational binary arithmetic operation involved (2, pp. 107-118). One of the primary objections to the method was the large amount of storage required to generate the Taylor coefficients for all of the subproblems. In addition, a scaling or normalizing of the Taylor coefficients was performed in order to obtain the greatest possible computational accuracy (1, pp. 103-112). Using the
analogous subproblem variables, the computation of (4.4) for each subproblem was accomplished, except for the last term, in nested form.\(^1\)

The algorithm consisted of a main program and 10 subroutines comprising 640 Fortran statements, including a subroutine devoted to producing graphical output. Since the algorithm was generated early in the research, the interval arithmetic defined by (2.4) was programmed directly in single-precision floating-point Fortran without incorporating the bounding features described in Chapter III.

The Kth-order nonlinear interval algorithm was tested on the three state first-order system derived from the second order linear differential equation \( \ddot{x} + 2\delta \dot{x} + x = U \) with zero initial conditions, where the parameter \( \delta = [0.1, 0.3] \) and \( U \) is the unit step input. (This problem is discussed in Chapter V where the results of applying the linear interval integration algorithm of the next section are given.) The parameter \( \delta \) was subdivided into 25 subintervals and 20th-order Taylor series interval method results were obtained. Although the test problem was linear

\(^1\)It should be noted that since the class of problems includes right-hand sides which are autonomous, rational and nonlinear, the coded recursive machine generation of the Taylor coefficients is not amenable to the centered form interval calculations discussed previously.
and the bounding aspect of the interval arithmetic had not yet been incorporated into the investigations, the performance of the algorithm was disappointing. The interval solution "envelope" produced for the response $x(t_1)$ diverged shortly after the first overshoot with the upper and lower bounds for all practical purposes respectively increasing exponentially positively and negatively. As a consequence of subsequent comparison of the linear and nonlinear algorithm results for this problem, this divergent behavior was qualitatively attributed to three factors. First, it was not possible to use the frequently less conservative centered form of the interval arithmetic for the nonlinear algorithm. Secondly, although the subset inclusion relations (4.5) and (4.6) are valid, the "$\epsilon$-error technique" was not effective in controlling the conservativeness of these inclusion relations. Finally, the recursive technique of reinitializing each algorithm subproblem to the subproblem interval solution value of the previous step had the overall algorithmic effect of attempting to produce solution bounds to the set of subproblems

$$\dot{x} = f(x, b), \quad (4.7)$$

where for each $t_{i-1} \leq t \leq t_i$, $i=1, \ldots$, the real scalar $b$ and the real "initial condition" vector $x(t_{i-1})$ may take on any fixed values.
where

\[ b \in \mathbb{I} \quad \left\{ \begin{array}{l} X(t_{i-1}) = x_j(t_{i-1}) \\ j = 1, \ldots, M \end{array} \right. \quad (4.7a) \]

and

\[ x_j(t_{i-1}) = \begin{cases} x_0 & , i = 1 \\ \text{the set of all the } j\text{th subproblem} \\ \text{solution values for time } t_{i-1}, i \neq 1 \end{cases} \quad (4.7b) \]

As \( M \) increases without bound, analytically the set of subproblems (4.7, 4.7a and b) approaches the original problem (4.1). To a large extent, the linear problem technique of the next section diminishes the above objections.

Autonomous Systems of Linear First-Order Equations

The class of initial-value problems of ordinary differential equations which are considered in this section is the set of autonomous systems of \( n \) first-order linear differential equations

---

1The subdivision of the parameter interval \( I \) is described in Theorem 2.14.

2If the original system is nonhomogeneous, containing a given continuous vector forcing function, by introducing additional dependent variables the system may be reformulated as the larger autonomous system (4.8). For example, if the forcing function is the unit step, a system of \( n+1 \) equations
\[ \dot{x} = Ax \quad \text{and} \]
\[ x(0) = \text{initial condition} \quad , \]

where the constant plant or coefficient matrix \( A \) contains elements which depend linearly on a perturbation parameter \((13, \text{p. 174})\) and the equations satisfy the standard theorems on uniqueness and existence as well as continuity and differentiability of the solutions with respect to the parameter and initial condition \((46, \text{pp. 159}-181)\).

In order to facilitate matrix element linear dependence on a negative or decreasing value of the parameter, the algorithm developed in this section is designed to input the matrix elements \( a_{ij} = [a_{ij}^L, a_{ij}^R] \) as a set of two single-precision floating-point numbers, not necessarily "interval-ordered" with \( a_{ij}^L \leq a_{ij}^R \). Then the matrices

\[ G_1 \equiv ((g_{ij}^1)) \equiv (((a_{ij}^R + a_{ij}^L)/2.0)) \quad \text{and} \]
\[ G_2 \equiv ((g_{ij}^2)) \equiv (((a_{ij}^R - a_{ij}^L)/2.0)) \quad (4.8a) \]

are computed\(^1\) employing the bounding interval arithmetic.

---

\(^1\)Appendix B contains the completely annotated algorithm developed in this section. The matrix \( A \) is input and the "nearly" degenerate interval matrices \( G_1 \) and \( G_2 \) are computed by subroutine NORM.
functions developed in Chapter III and the perturbation parameter dependent differential equation may be formulated as
\[ 
\dot{x} = (G_1 + \theta \cdot G_2) x \quad \text{and} \quad x(0) = \text{initial condition,} \quad \text{where} \\
\theta \in [-1.0, +1.0] . \quad (4.9)
\]

In this manner the coefficient matrix elements \( g_{ij}^1 + \theta g_{ij}^2 \) accommodate the "signed" dependence on the perturbation parameter \( \theta \in [-1.0, +1.0] \) as a result of the sign of \( g_{ij}^2 \).

For any value of the perturbation parameter \( \theta \in [-1.0, +1.0] \) (assume for this discussion that \( G_1 \) and \( G_2 \) are real matrices and that \( x(0) = x_0 \) is a real vector), the analytic solution of (4.9) is given by (23, p. 356)
\[ 
x(t) = \Phi(\theta, t) x_0 , \quad (4.10)
\]
where \( \Phi(\theta, t) \) is the fundamental solution or state transition matrix defined by
\[ 
\Phi(\theta, t) \equiv e^{(G_1 + \theta G_2) t} . \quad (4.11)
\]

The interval solution technique of the algorithm is to subdivide the \([-1.0, +1.0]\) perturbation parameter interval into \( M \) equal width subintervals defined by (single-precision endpoint computations)
and to compute, in a manner analogous to (4.10), the interval solutions for each of the subproblems

\[ \hat{x} = (G_1 + \Theta G_2)x \quad \text{and} \]

\[ x(0) = \text{initial condition}, \quad \text{where} \]

\[ \Theta \in \Theta_i, \quad i=1, \ldots, M \quad (4.13) \]

In contrast then to the nonlinear solution method previously described where the time increment may be recursively halved in order to satisfy an error criterion (see Figure 4.1), a fixed time increment is employed in the linear algorithm yielding solution values for equally spaced intervals of time. The linear algorithm subproblem solution technique, unlike the nonlinear algorithm, employs the specific solution value of time and the linearity of the problem (see Equation 4.10) by computing subproblem "pseudo-fundamental" interval matrices which set theoretically include the set of all real fundamental matrices for each value of the parameter in the subinterval (see Equations 4.11 and 4.12) and which elementwise satisfy an "inclusion"

\[ 1 \text{The nonlinear algorithm depends on the computation and use of an interval vector bound for the set of all possible computed interval solution values for each value of time within the incremented time interval (see Equation 4.4).} \]
error bound. Finally the interval solution for (4.9) is obtained by taking the union\(^1\) over the subproblem interval solutions to (4.13), producing interval bounds or envelopes for the set of all solutions associated with the interval vector initial condition and the perturbation parameter, including the effects of algorithmic numerical truncation error.

The reasons for subdividing the perturbation parameter interval are closely linked to the recursive interval subproblem solution technique and Moore's conjecture (2.47) relating the conservativeness of the width of centered form calculations to an order of magnitude of the square of the width of the interval variable.

The subproblem interval solutions and subsequent interval solution union over the subproblems are computed by subroutine DRIVER while the subproblem pseudo-fundamental interval matrices are computed (actually in the centered interval form as will be discussed) by subroutine PHCOMP (see subroutine PHCOMP and the implementation of the calculations for Equation 2.90).

Before turning to a description of the actual

\(^1\) It should be recognized that the interval union is obtained by individually computing an interval union for each component of the solution vector over the corresponding subproblem interval solution components.
implementation of the algorithm, since the bounding inter-
val arithmetic computation of the subproblem pseudo-funda-
mental matrices is somewhat removed from the exact interval
arithmetic theory developed in Chapter II, it is appropriate
at this point to include a discussion of the assumptions
necessary to connect the two.

The centered form interval representation for each sub-
interval \( \Theta_i \) (4.12) is computed in the single-precision arith-
metic\(^1\) defined by (see subroutine PRCCMP)

\[
\Theta_i \equiv [c_i', c_i], \quad c_i \equiv \frac{-M + 2i - 1}{M} \quad \text{and}
\]

\[
\eta_i \equiv [-w_i, w_i], \quad w_i \equiv \max\{c_i - \left(\frac{-M + 2(i-1)}{M}\right), \left(\frac{-M + 2i}{M}\right) - c_i\}.
\]

(4.14)

Let the following symbols over \( \Theta \) denote:

( no symbol) - an exact interval arithmetic calculation;

( ~ ) - a bounding interval arithmetic calcula-
tion;

\(^1\)It was verified on the computer (using the single-pre-
cision floating-point implementation of the interval arith-
metic (2.4) rather than the bounding interval arithmetic) by
exhausting the cases \( M=1, \ldots, 25 \), that

\[
\bigcup_{i=1}^{M} \Theta_i = [-1.0, +1.0], \quad \bigcup_{i=1}^{M} (\Theta_i + \eta_i) = [-1.0, 1.0] \quad \text{and}
\]

\[
\Theta_i \subseteq \Theta_i' + \eta_i, \quad i=1, \ldots, M.
\]
Also let
\[ \bar{\alpha}_K(\theta_j + \eta_j, t_1) = \text{the nested centered form } K \text{ term truncated series calculation for Equation 2.80 (by Proposition 2.8M,} \]
\[ \bar{\alpha}_K \in \{ \mathcal{F}^{n_2}, \mathcal{E} \} \text{ and} \]
\[ (4.15a) \]
\[ \bar{\alpha}_K^+(\theta_j, t_1) = \text{the expanded form } K \text{ term truncated series calculation for Equation 2.67 (by Proposition 2.8M,} \]
\[ \bar{\alpha}_K^+ \in \{ \mathcal{F}^{n_2}, \mathcal{E} \} . \]
\[ (4.15b) \]

Then by Proposition 2.7M,
\[ \bar{\alpha}_K \text{ and } \bar{\alpha}_K^+ \in \{ \mathcal{F}^{n_2}, \mathcal{E} \} \]
and as a result of the property of the bounding interval arithmetic and Proposition 2.9M,

\[ \text{To be precisely correct in the nested centered form representation, } \bar{\alpha}_K(\theta_j + \eta, t_1) \neq \bar{\alpha}_K(\theta_m + \eta, t_1) \text{ if } m \neq \ell, \eta \in J \text{ and } I = [-1.0, 1.0]. \text{ That is, the rational function is explicitly defined by the degenerate intervals } \theta_j \text{ or } \theta_m. \text{ In this sense, the function space member notation } \bar{\alpha}_p \text{ should, to be precise, additionally indicate definition on the subdivision index, } j. \]
Following Theorem 2.14M it was proved that \{\tilde{\alpha}_K^+\} is Cauchy and therefore by Propositions 2.5M and 2.10M,

\[
\{\tilde{\alpha}_K^+\} \to \tilde{\alpha}^+ \in \{\mathcal{F}^2, \Xi\}
\]

and by Proposition 2.11M,

\[
\tilde{\alpha}^+(\theta_j, t_1) \supseteq \tilde{\alpha}_K^+(\theta_j, t_1)
\]

Based on the observations of the considerable reduction of the conservativeness of the nested centered form results for the example problems actually tested on the algorithm (see Equation 2.82) when computationally checked in the relationship

\[
\tilde{\alpha}_K(\theta_j + \eta_j, t_1) \supseteq \tilde{\alpha}_K(\theta_j, t_1)
\]

assume that
\[ a_k(\theta_j + \eta_j, t_i) \subseteq \tilde{a}_k^+(\theta_j, t_i) \]  \hspace{1cm} (4.20)

From preliminary analytic investigation of \( \{ \tilde{a}_k \} \) and
the remainder terms associated with \( \{ \tilde{a}_k^+ \} \), it seems reasonable
to assume that \( \{ \tilde{a}_k \} \) is Cauchy although proof of this as­
sumption remains an open question because of the complexity
resulting from the nesting (2.75a and b, 2.78a and b and
2.80) when analyzing \( \tilde{c}(\tilde{a}_k, \tilde{a}_{k+1}) \) by (2.21). The limit then
is an infinitely nested centered form which is cumbersome
to formulate and will simply be denoted by \( \tilde{a}(\theta_j + \eta_j, t_1) \).

The infinite series expanded form remainder given by
(2.83) is denoted \( R_k^+(\theta_j, t_1) \). While it is not possible to
formulate an analogously denoted \( R_k(\theta_j + \eta_j, t_1) \) in view of
the nesting involved in the centered form calculations for
\( \tilde{a}_k(\theta_j + \eta_j, t_1) \), if it is assumed that \( \{ \tilde{a}_k \} \) is Cauchy then
there does exist a "remainder" perturbation interval matrix
which will be so denoted and then

\[ \tilde{a}(\theta_j + \eta_j, t_1) = \tilde{a}_k(\theta_j + \eta_j, t_1) + R_k(\theta_j + \eta_j, t_1) \]  \hspace{1cm} (4.21)

In view of (4.19), (4.20) and the exact interval arithmetic
calculations involved in computing \( R_k^+(\theta_j, t_1) \), it seems
reasonable to also assume that

\( ^1 \)This appears reasonable in view of the larger number
of calculations involved in \( \tilde{a}_k \) compared to \( \tilde{a}_k^+ \) on which the
bounding property of the interval arithmetic would tend to
contradict the observed relationship (4.19).
\[ R_k(\Theta_j, \eta_j, t_1) \subseteq R_k^+(\Theta_j, t_1) \quad (4.22) \]

In this case then, by Propositions 2.5M and 2.10M

\[ \{\overline{a}_k\} \rightarrow a \in \mathcal{F}_{n^2}, \xi \} \text{ and } \{\overline{a}_k\} \rightarrow a \in \mathcal{F}_{n^2}, \xi \} \quad (4.23) \]

By Proposition 2.11M

\[ a(\Theta_j, \eta_j, t_1) \supseteq \overline{a}(\Theta_j, \eta_j, t_1) \quad (4.24) \]

On the basis of assumption (4.22) and therefore (2.88) and (2.89) hold. Of course, the computational implementation of relationship (2.89) would be written as

\[ |r_{Lm_K}(\Theta_j, \eta_j, t_1)| \leq |r_{Lm_K}^+(\Theta_j, t_1)| \]

\((.|.\) is the interval magnitude operation\) and therefore (2.88) and (2.89) hold. Of course, the computational implementation of relationship (2.89) would be written as

\[ |r_{Lm_K}(\Theta_j, \eta_j, t_1)| \leq \frac{\overline{g}_{L_{ij}}}{g_{m_j}} \cdot \sum u_j \leq 16^{-p} \cdot \min \{ |\tilde{\phi}_{Lm_K}(\Theta_j, \eta_j, t_1)|, |\tilde{\phi}_{Rm_K}(\Theta_j, \eta_j, t_1)| \} \quad (4.25) \]

where the center term is a double-precision calculation using double-precision computed arguments (denoted by \( \tilde{\phi} \)) with the result truncated to a single-precision number and the relationship checked in the single-precision arithmetic. It is conceivable that this computational test may pass
where the exact interval arithmetic test might fail, due to the bounding interval arithmetic effects in computing \( \hat{a}_k(\theta_j + \eta_j, t_1) \). It will be assumed that this does not occur.

Then the augmented centered form final result \( \hat{a}_k(\theta_j + \eta_j, t_1) \) computed by (2.90) satisfies the relationship

\[
[1-\varepsilon, 1+\varepsilon] \cdot \tilde{a}(\theta_j + \eta_j, t_1) \supset \hat{a}_k(\theta_j + \eta_j, t_1)
\]

\[
\equiv \tilde{a}(\theta_j + \eta_j, t_1) + Z_j \supset \tilde{a}(\theta_j + \eta_j, t_1) \supset \bar{a}(\theta_j + \eta_j, t_1),
\]

(4.26)

where

\[
Z_j \equiv \left( \begin{array}{c} \frac{\tilde{g}}{g_m} \cdot \gamma_{u_j} \vspace{1em} \frac{\tilde{g}}{g_m} \cdot \gamma_{u_j} \end{array} \right)^1
\]

(4.27)

and the absolute value of the interval endpoint errors of the computed elements of \( \hat{a}_k \) relative to the corresponding endpoints of the elements of the exact interval arithmetic result \( \tilde{a} \) are bounded by (see Equations 2.44 and 2.45)

---

1This formulation for the augmenting matrix \( Z_j \) assumes that the \( \| \cdot \|_U \) rather than the \( \| \cdot \|_{U^*} \) optimal Householder matrix norm (see Equations 2.53 and 2.54 and MODE=1 in subroutine NORM) has been used in computing the bound \( \gamma_{u_j} \) in (2.88) and that certain special cases have not been algorithmically trapped and the special augmenting processes placed in effect (see Equation 2.92 and the detailed listing of subroutine PHCOMP in the Appendix).
While Proposition 2.12M is directly applicable to \( \tilde{a}_K^+ \), the same type of result may be indirectly obtained in the case of \( \bar{a}_K \) by observing that \( \epsilon_j \cap \epsilon_{j+1} \neq \emptyset, \epsilon_\ell \subset \epsilon_{\ell'}^{l+1} \), \( l=j, j+1 \) and therefore \( (\epsilon_{j_c} + \eta_j) \cap (\epsilon_{j+1_c} + \eta_{j+1}) \neq \emptyset \) and by arguing the continuity of \( \bar{a}_K, \tilde{a}_K^+, \tilde{a}_K^+ \) and \( \bar{a}_K \in \{ F_{n^2}, X \} \). However, in the case of \( \bar{a}_K \) this would be written (see Equation 4.15a and the footnote)

\[
\bigcup_{j=1}^{M} \bar{a}_K(\epsilon_{j_c} + \eta_j, t_1) \supseteq \bigcup_{j=1}^{M} \tilde{a}_K(\epsilon_{j_c} + \eta_j, t_1),
\]

(4.29)

where both unions belong to \( J_{n^2} \).

The result of Proposition 2.13M may be analogously obtained employing (4.23) and would be written as

\[
\bigcup_{j=1}^{M} \bar{a}_K(\epsilon_{j_c} + \eta_j, t_1) \supseteq \bigcup_{j=1}^{M} \tilde{a}_K(\epsilon_{j_c} + \eta_j, t_1)
\]

converging to

\[
\bigcup_{j=1}^{M} \bar{a}(\epsilon_{j_c} + \eta_j, t_1) \supseteq \bigcup_{j=1}^{M} \tilde{a}(\epsilon_{j_c} + \eta_j, t_1).
\]

(4.30)

Turning now to the subproblem solution method, each numerical subproblem bounding interval arithmetic solution (denoted \( \tilde{x}_j(t_1) \)) is computed as a bounding interval arithmetic matrix product (see Equation 4.10 for the analogy) and

\[
\varepsilon \equiv 2 \cdot \left( \frac{16^{-p}}{1 - 16^{-p}} \right) .
\]

(4.28)
provides the set inclusion relationship

\[
\{[1-\varepsilon, 1+\varepsilon] \cdot \mathfrak{a}(\theta_j + \eta_j, t_1)\} x(0) \supset \tilde{x}_j(t_1)
\]

\[
= \hat{\mathfrak{a}}_K(\theta_j + \eta_j, t_1)x(0) \supset \mathfrak{a}(\theta_j + \eta_j, t_1)x(0)
\]

\[
\supset \mathfrak{a}(\theta_j + \eta_j, t_1)x(0) \supset \bigcup_{\eta \in \eta_j} \bigcup_{x_0 \in x(0)} \{\mathfrak{a}(\theta_j + \eta, t_1)x_0\},
\]

\[(4.31)\]

where \(x_0\) is meant to indicate any degenerate interval (real) vector initial condition. Although (4.26) provides an analytic error bound for the elements of \(\hat{\mathfrak{a}}_K\) with respect to \(\mathfrak{a}\), a similar type of expression is not possible for \(\tilde{x}_j(t_1)\) by virtue of the complexity of even the exact interval arithmetic (2.4) involved in (4.31).

Finally, denoting the algorithm interval solution for (4.9) as \(\tilde{x}(t_1)\), obtain the relationship

\[
\bigcup_{j=1}^{M} \{[1-\varepsilon, 1+\varepsilon] \cdot \mathfrak{a}(\theta_j + \eta_j, t_1)\} x(0) \supset \tilde{x}(t_1) \equiv \bigcup_{j=1}^{M} \tilde{x}_j(t_1)
\]

\[
\supset \bigcup_{j=1}^{M} \mathfrak{a}(\theta_j + \eta_j, t_1)x(0) \supset \bigcup_{j=1}^{M} \overline{\mathfrak{a}}(\theta_j + \eta_j, t_1)x(0)
\]

\[
\supset \bigcup_{j=1}^{M} \bigcup_{\eta \in \eta_j} \bigcup_{x_0 \in x(0)} \{\overline{\mathfrak{a}}(\theta_j + \eta, t_1)x_0\},
\]

\[(4.32)\]
where again an analytic error bounding expression is not possible for $\tilde{x}(t_1)$.

While Theorem 2.14M is directly applicable to $\{\tilde{a}_K^+\}$, in the case of $\{\tilde{a}_K\}$ a similar type of result may be expected by analogy with the arguments employed in extending Propositions 2.12M and 2.13M to the case of $\{\tilde{a}_K\}$. As a consequence of these exact interval arithmetic results, the philosophy of the algorithm is to increase the number of subdivisions in the perturbation parameter interval with the expectation of obtaining less conservative results for the overall solution technique.

Having completed discussion of the assumptions necessary to connect the analytic theory developed in Chapter II with the bounding interval arithmetic computation of the subproblem pseudo-fundamental interval matrices and the general solution technique philosophy, turn now to the implementation of the algorithm, referring to Figure B-1 (the overall algorithm operation flowchart) and the documented algorithm and bounding interval arithmetic function listings in the Appendix.

In order to provide the greatest user versatility, the algorithm is designed so that the operator writes the controlling Fortran main program incorporating subordinate
features which he desires\(^1\) and including a specified sequence of subroutine calls. An example and explanation of a user written main program is included in the algorithm listing. The algorithm subroutines and COMPLEX bounding interval arithmetic functions may then be link-edited in their object forms, with the Fortran subroutines having been previously compiled with the large storage requirement optimizing \(H\) compiler and the assembler functions having been previously optimally assembled. The required EISPAC and SIMPLOT subroutines are link-edited from their SYSLIB object modules.

Referring to Figure B-1, the first required subroutine call (A) is to subroutine NORM. The "interval" matrix \(A\) (see the discussion following Equation 4.8) is input and the matrices \(G_1\) and \(G_2\) of (4.8a) are computed in the bounding interval arithmetic. The other necessary algorithm parameters also input at this time are the matrix dimension, the two bounding interval arithmetic COMPLEX function operating parameters, the selection of the Householder matrix norm mode and the number of subdivisions of the perturbation parameter interval. By the repeated sequence

\(^1\)For example, the user may incorporate multiple problem run looping and punched interval solution output data as well as other features by appropriate functioning of the user written main program.
of three \textsuperscript{1} subroutine calls \((A_{K_{1}})\) to the EISPAC subroutines (47), the \(R*8\) computed subproblem Householder matrix norms, positive eigenvectors and positive eigenvalues (see the discussions accompanying Equations 2.53, 2.55 and 2.87) are returned and stored in \(R*4\) arrays for subsequent use by the algorithm. (For built in output, error diagnostics and fault traps, see the documented subroutine listing.)

The next required subroutine call (B) is to subroutine GCOMP where the balance of the entire sequence of "G" interval matrices \(G_3,...,G_{230}\) is computed by (2.68) and stored for subsequent use. This represents the required matrices necessary for computing the subproblem pseudo-fundamental matrices \(\hat{A}_{K}\) of (2.90) with a maximum value of \(K=20\).

Then subroutine PRCOMP is called (C) where the entire set of the interval binomial coefficients (see Equations 2.74 and 2.78b) is precomputed by the method of "completing" Pascal's triangle and stored. The algorithm integer \(P\) is input, the test factor \(16^{-P}\) of (4.25) is computed and the centered form interval representations for the subintervals (4.12) are computed by (4.14).

\textsuperscript{1}The specified sequence of three particular EISPAC subroutine calls results in rapid computational operation of the eigenvalue-eigenvector package as contrasted with the considerably slower single general purpose "driver" operating call to EISPAC (47).
The next required subroutine call (D) is to subroutine DRIVER which controls the algorithm solution technique. Figure B-2 of the Appendix is a flowchart of the essential operating characteristics of this subroutine. The subroutine opening sequence inputs the necessary algorithm parameters including the fixed algorithm solution time increment, the number of steps in the time increment for which the solution is desired and the problem interval vector initial condition (4.9). Additionally, the "rolling" subproblem solution technique control parameter is initialized (IROLL=1) and the first solution storage location of each subproblem (4.13) is initialized to the input problem interval vector initial condition (4.9), completing the subroutine opening sequence.

The subroutine operation then enters the outside recursive solution A-loop, computing the solution time parameter, $t_1$. On this first iteration, subroutine PHCOMP is called (call $D_1$ in Figure B-1) which computes and returns (assume successfully and therefore IROLL is not changed) the subproblem pseudo-fundamental matrices $\hat{a}_k$ of (4.26). The "pre-roll" subproblem solution storage is then in effect and the subroutine enters the inside recursive subproblem B-loop where each of the subproblem vector interval solutions $\tilde{x}_j(t_1)$ of (4.31) is computed and stored in the second subproblem storage location. Finally, the algorithm
vector interval solution $\bar{x}(t^*_1)$ of (4.32) is computed and the end of the A-loop is encountered, indexing the loop counter.

The "rolling" subproblem solution storage operation of the algorithm is activated in one of two ways.

First, since considerable core would be required to store all of the subproblem solutions (100K bytes for the algorithm limits), the algorithm is designed to provide local temporary storage for nine subproblem solutions (the initial condition and eight solutions, requiring only 9K bytes of storage for the algorithm limits). Assume that PHCOMP has successfully computed the subproblem pseudo-fundamental matrices for $t^*_i$, $i=1,\ldots,8$ with the most recent value calculated for time $t^*_8$ currently in storage. Then subroutine DRIVER has sequentially computed the subproblem solutions for these times and stored them respectively in locations 2,\ldots,9. The initial conditions are stored in location 1. The ninth A-loop recursion calls subroutine PHCOMP ($D_9$) where this counted ninth entry triggers a bypass of all computations, retaining the subproblem pseudo-fundamental matrices computed for time $t^*_8$, sets $\text{IROLL}=2$ and returns to subroutine DRIVER. The return initializes the "rolling" subproblem solution storage modular arithmetic parameter $\text{ILoop}$ with the consequence that the inside B-loop computes the subproblem solutions for time $t^*_9$ and
stores the results in storage location 1, using the sub-
problem solutions for time $t_1$ stored in location 2 as
initial conditions and the $t_8$ time subproblem pseudo-funda-
mental matrices. The algorithm vector interval solution
$\tilde{x}(t_9)$ is computed as before and the end of the A-loop is
encountered, indexing the loop counter. The balance of
the operation of the A-loop uses the "rolling" subproblem
"initial condition" and solution storage location scheme
and the fixed subproblem pseudo-fundamental matrices com-
puted for time $t_8$, with the additional benefit of greatly
increasing the algorithm computational speed at which the
subproblem solutions are obtained, since subroutine PHCOMP
is not called again.

The second way in which the "rolling" subproblem solu-
tion storage mechanism is activated is a result of a "fail-
ure" of subroutine PHCOMP to compute the pseudo-fundamental
matrices according to the user controlled error criteria
which is input on the opening sequence of subroutine PHCOMP.
The "rolling" storage mechanism operation in the second case
is analogous to the first, with one exception. If the $L$th
call ($D_L$, $1 < L \leq 8$) to subroutine PHCOMP results in a
"failure" which returns IROLL=2, only the first $L$ subproblem
storage locations are available to the "rolling" scheme.
The subproblem pseudo-fundamental matrices computed for time
$t_{L-1}$ are then used for the balance of the operation of the
A-loop.

The output listing format for the algorithm interval solutions and an explanation of the timed CPU clock "early termination" mechanism will be found in the documented listing.

Figure B-3 of the Appendix is a flowchart of the salient operating characteristics of subroutine PHCOMP.

The first time the subroutine is called \( (D_1) \), an opening sequence of operations is initiated in which several user selected algorithm parameters are input. First, the starting value \( (K\text{K}) \) for the number of terms in the series to be initially attempted in the computation of the sub-problem pseudo-fundamental matrix elements is input. Then the limit numbers \( \text{NKNT}1 \) and \( \text{NKNT}2 \) (see the subsequent discussion of Equation 4.25) are read for the numbers of single- and double-fault errors which are acceptable before a "failure" is declared for any one set of subproblem pseudo-fundamental matrices (that is, for any one subroutine entry). The limit number for the number of individual subroutine entry "failures" \( (\text{NFAIL}) \) accepted before a PHCOMP failure is declared, is then input. Following this, a limit number \( (\text{NRUNUP}) \) is read which is the number of "run-ups" allowed in the automatic increase in the number of terms used in the truncated pseudo-fundamental matrix series in attempts to satisfy \( (4.25) \) before an "acceptance" of that
matrix element is forced. The \( R^8 \) factorials necessary in the computation of \( \sum u_j \) of (4.25) are computed and the starting value array for the number of terms to be initially attempted in the computation of each of the first (and subsequent, if successful) subproblem \((I,J)\)th pseudo-fundamental matrix elements for this first subroutine entry is initialized \((\text{KKPAST}(I,J)=\text{KK})\). The "failure" counter \((\text{IFAIL})\) is set to zero, completing the "first time called" opening sequence.

The subroutine computational operation begins with initialization of the degenerate interval time variable to the time \( t_i \) of the calling subroutine \( \text{DRIVER} \) and the zero initializing of the single- and double-fault error counters \((\text{KNT1} \text{ and } \text{KNT2})\) for this subroutine entry. The external \( A \)-loop (the recursive \((I,J)\)-loop) starts and then the internal \( B \)-loop (the recursive subproblem loop for fixed \( I,J \)) begins with the initializing of the "runup" counter \((\text{IRUNUP}=0, \text{for each value of the internal } B \text{-loop subproblem index})\). Next, for the first \( B \)-loop subproblem of each \( A \)-loop \((I,J)\), the variable \( \text{KLOOP} \) for the number of terms to be initially used in attempting to compute this pseudo-fundamental matrix element for this subroutine entry is initialized \((\text{KLOOP}=\text{KKPAST}(I,J))\). Entry point \( C \) is encountered next, which is an internal \( B \)-loop entry point used in the event that \( \text{KLOOP} \) is automatically increased ("runup")
because of a single- or double-fault. Then, for the fixed A-loop index, the nested interval matrix elements $B(1), \ldots, B(K\text{LOOP}+1)$ of (2.75a and b) are computed. These values depend only on the interval time variable, the A-loop value of $(I,J)$ and $K\text{LOOP}$. Unless $K\text{LOOP}$ has been automatically changed and a backloop entry point $C$ has been initiated, these values are not recomputed for the next B-loop index value. This is the reason for bypass BP2 on the flowchart.

Next, the value of $\tilde{\mathbf{u}}$ (see Equations 2.88 and 4.25) is computed in $R^*8$ precision and truncated to $R^*4$ precision, using the current value of $K\text{LOOP}$, the B-loop index "assigned" $R^*8$ value of the Householder matrix norm (previously computed in $R^*8$ and stored in $R^*4$ by subroutine NORM) and the $R^*8$ factorials (computed in the opening sequence of this subroutine). Following this the nested interval matrix element values $E(1), \ldots, E(K\text{LOOP}+1)$ of (2.78a and b) are computed. These values are a function of the current values of $B(1), \ldots, B(K\text{LOOP}+1)$ and the subroutine PRCOMP computed and stored B-loop index value of the centered form interval midpoint (4.14) and the interval binomial coefficients. Then the nested interval matrix element value of $\tilde{\mathbf{\alpha}}_{K\text{LOOP}}$ (see Equations 2.80 and 4.25) is computed. This value is a function of $E(1), \ldots, E(K\text{LOOP}+1)$ and the subroutine PRCOMP computed and stored B-loop index
value of the centered form "zero-symmetric" interval (4.14).

Before discussing the operation of the error criterion testing and augmenting section of the subroutine operation, the single- and double-fault error descriptions and special circumstances will be discussed in terms of (4.25).

If the test of relationship (4.25) fails but passes with "max" replacing "min" on the right-hand side, a single-fault exists. If (4.25) fails and (4.25) with "max" replacing "min" also fails, a double-fault exists. Certain exceptions to the above cases are provided in the subroutine. If the interval result to be tested is a degenerate interval, testing is not performed and the result is augmented (see Equation 4.26) with the degenerate zero interval replacing (4.27). If the left (right) interval result endpoint is zero and (4.25) passes with "max" replacing "min", no fault is declared and the result is augmented with a zero left (right) endpoint replacement in (4.27). If the left (right) interval result endpoint is zero and (4.25) fails with "max" replacing "min", a double-fault is declared. In this latter case, if KLOOP=20 or if IRUNUP > NRUNUP, the double-fault counter KNT2 is indexed and the result is augmented with a zero left (right) endpoint replacement in (4.27) and "temporarily" accepted (stored).

Excluding the special cases which have already been explained, if a fault does not exist the augmenting
indicated in (4.26) is accomplished with (4.27) and this (I,J)th element for the B-loop index subproblem pseudo-fundamental matrix is "temporarily" stored. If a single- or double-fault exists, several actions may be taken. First, if KLOOP = 20 (computational storage limit built into the algorithm, see subroutine GCCMP), the single- or double-fault counter (KNT1 or KNT2, which accumulate only for each subroutine entry) is indexed and the result is augmented and "temporarily" stored. Second, if KLOOP < 20 and the number of times KLOOP has been previously increased by one (IRUNUP) for this B-loop index value is not greater than NRUNUP, KLOOP is increased by one and a "backloop" is performed returning to entry point C in an attempt to clear the fault. Or, in the second case if IRUNUP > NRUNUP, the single- or double-fault counter is indexed and the result is augmented and "temporarily" stored. Once the augmented result is obtained, the end of the B-loop is encountered, indexing that subproblem loop counter and the operation is repeated until all of the subproblem pseudo-fundamental matrix elements for this fixed (I,J) value of the A-loop are computed. It should be noted that the value of KLOOP is monotonically nondecreasing with the subproblem B-loop indexing (recall that a new KLOOP = KKAST(I,J) B-loop starting value is assigned for each new (I,J) value of the A-loop). Upon completion of the inner B-loop, the end of the
outside A-loop is encountered, the last value of KLOOP used in the completed B-loop is stored (KKPAST(I,J)=KLOOP) for starting value B-loop use by the subroutine on the next subroutine entry and the A-loop counter is indexed.

Once computation of all of the "temporary" subproblem pseudo-fundamental interval matrix elements is completed, the single- and double-fault error counters (KNT1 and KNT2) are checked against the input algorithm limit numbers (NKNT1 and NKNT2). If there is not an excess in either count, the "temporary" (local) subroutine results are stored in a common array and control is returned to subroutine DRIVER. If there is an excess in either count a failure is declared, the failure counter (IFAIL) is indexed\(^1\) and checked against the input algorithm limit number (NFAIL).

Prior to returning control to subroutine DRIVER, several actions are possible. If there is not an excess, the "temporary" subroutine results are stored in the common array. If additionally there is equality, a trap is set so that the next subroutine entry will simply be a "pass-through" without

---

\(^1\)An exception is built into the algorithm so that a subroutine failure on the first call results in an automatic program termination. The philosophy in this case is that, since the subroutine failed on its first call, rather than allow the CPU execution time (and cost) to build, the user should inspect the results and reexamine his input error criteria and problem parameter.
computation, but triggering the "roll" operation of subroutine DRIVER based on PHCOMP values returned in the present entry. If there is an excess, the "temporary" results are not stored and the "roll" operation of subroutine DRIVER is triggered so that it uses the PHCOMP values returned on the previous call to this subroutine.

This completes the description of the operation of subroutine PHCOMP as it is called by subroutine DRIVER. Various built-in diagnostic outputs, error warnings and messages may be determined from the subroutine listing. Upon completion of each subroutine call, the error counts and KKPAST (I,J) array are listed.

Finally subroutine CURVES is called (E) which, directed by the input data stream there, produces the graphical output of the algorithm interval solutions on the CalComp drum type incremental plotter (48). Since an annotated listing of this subroutine (which includes a complete operating description) is found in the Appendix, its functional characteristics will not be discussed here and may be obtained by referring to the listing. (It will additionally be necessary to refer to the SIMPLOT operating manual (49).)

---

1This case can only arise if the user selected PHCOMP error parameter NFAIL is zero.
In order to assist the user in operating problem analysis, output "dumps" are individually controlled by the user from the main program for the listing of supplementary computational results in each subroutine, with the exception of subroutine CURVES. The information provided by these "dumps" may be determined by examining the individual subroutine listings in the Appendix.

In effect then, the algorithm yields discrete-time interval vector solution bounds or envelopes to the set of perturbation parameter subproblems

\[ \dot{x} = (G1 + \theta \cdot G2)x \quad , \quad (4.33) \]

where

(i) for \( 0 \leq t \leq t_i, \ i=1, \ldots, L, \)

the real scalar \( \theta \) and the real initial vector

\[ x_0 \] may take on any fixed values

\[ \theta \in \Theta_j \quad (4.12) \]

\[ x_0 \in x(0) \quad \begin{cases} j=1, \ldots, M \end{cases} \quad (4.33a) \]

and

(ii) for \( t_i \leq t \leq t_{i+L}, \ i+L=L+1, \ldots, \text{NSTEPS}, \)

the real scalar \( \theta \) and the real "initial vector"

(at time \( t_i \)) \( x(t_i) \) may take on any fixed values

\[ \theta \in \Theta_j \quad (4.12) \]

\[ x(t_i) \in x_j(t_i) \quad \begin{cases} j=1, \ldots, M \end{cases} \quad , \quad (4.33b) \]
where

\[ x_j(t_i) \equiv \bigcup_{\theta \in \Theta_j} \bigcup_{x(0) \in x_0} \{ g(\theta, t_i, x_0) \} , i = 1, \ldots, L \]

\[ x_j(t_i) \equiv \bigcup_{\theta \in \Theta_j} \bigcup_{x(t_{i-L}) \in x(t_{i-L})} \{ g(\theta, t_{i-L}, x(t_{i-L})) \} , \]

\[ i = L + 1, \ldots, \text{NSTEPS} - L . \]

The algorithm operation before "roll" is identified with (4.33a) while the "roll" operation agrees with (4.33b).

This essentially completes the discussion regarding the algorithm and the connection to the analytic theory. However, several concluding general remarks are in order.

Analytically, if the initial condition \( x(0) \) is a real vector and \( G_1 \) and \( G_2 \) are real matrices, as \( M \) is increased the set of subproblems (4.33, 4.33a and b) tends toward the original problem (4.9).

Algorithmically, if the initial condition \( x(0) \) is an interval vector with interval components that are relatively narrow and if the elements of the interval matrices \( G_1 \) and \( G_2 \) are near degenerate as a result of the bounding interval arithmetic calculations (4.8a), hopefully as \( M \) increases the algorithm solution (to the set of subproblems 4.33, 4.33a and b) tends toward the bounds or envelopes to the set of solutions to the original problem (4.9). Additionally, it is obvious that the longer the algorithm can
successfully operate in the "pre-roll" condition (4.33a),
the empirically less conservative the overall results of
the algorithm will be for a given fixed number of re-
quired solution points. (The fact that the original non-
linear algorithm implementation did not yield to the use
of this linearity concept was one of the contributing
factors in its instability. See for example, Equation 4.5
and Figure 4.1.)

A few remarks are in order regarding the core require-
ments to operate the algorithm. With fixed storage limits
for a maximum of 20 terms in (2.67),\(^1\) five dependent (state)
variables in the ordinary differential equation system, 100
algorithm interval vector solution points, 25 subproblems
and an array for nine "rolling" subproblem storage loca-
tions, exclusive of the compiled user written main program
but including the necessary link-edited EISPAC, SIMPLOT and
IBCOM# module subroutines, approximately 129K bytes of core
are required to load the algorithm. While it is possible
to employ overlay techniques to somewhat reduce this storage
requirement, the 46K byte "G" matrix array requirement
(which necessarily would be located in the root segment)
hampers efforts toward this end allowing a final overlayed

\(^1\) This corresponds to \((20)(20+3)/2=230\) interval "G"
matrices in (2.69a) which requires 46K bytes of storage.
core requirement of approximately 106K bytes. Such a relatively small core reduction would be cost-wise obscured by the additional expense of the overlay operations.

Finally, as an indication of the computational execution time required in operating the algorithm, for a five state, 25 subproblem, 36 point solution for one of the examples ("roll" occurred at L=9), 236 seconds of CPU time were required.
CHAPTER V. NUMERICAL INTERVAL RESULTS

Linear Interval Integration Algorithm Examples

The determination of changes in system performance due to changes in parameters is of great importance in engineering analysis and design. Such perturbation parameters in automatic control systems may model the effect of uncertainty in manufacturing tolerances (necessarily nonzero), "aging" of components and environmental causes, to name a few.

At the present time there is a wide interest in the problem of sensitivity related to the construction of modern control systems (50, 51). One common approach to analyzing the sensitivity of the solutions of nonlinear systems of ordinary differential equations with a perturbation parameter is to assume "sufficiently small" variations of the parameter and linearize about the nominal solution (52). If the nonlinear differential equation is written as

\[ \dot{x}(t) = f(x(t), \theta), \quad t \in [0, t_1] \quad \text{and} \]
\[ x(0) = \text{initial condition, where} \]
\[ \theta = \theta_0 + \delta \theta = \text{scalar perturbation parameter with} \]
\[ \text{nominal value } \theta_0 \text{ and perturbation} \]
\[ \delta \theta = \theta - \theta_0 \quad , \]

the commonly used sensitivity analysis approach is to
approximate the solution \( x(t) \) by the "first approximation"

\[
x(t) \approx x_0(t) + \delta \theta \cdot \delta x(t),
\]

where

\[
x_0(t) = \text{nominal parameter value solution to (5.1)} \quad \text{and}
\]

\[
\delta x(t) = \text{the "first variation"}. \quad (5.2)
\]

The "first variation" is the solution to the differential equation

\[
\delta x(t) = f_x(x_0(t), \theta_0) \delta x(t) + f_\theta(x_0(t), \theta_0) \delta \theta \quad \text{and}
\]

\[
\delta x(0) = 0 , \quad (5.3)
\]

where \( f_x \) and \( f_\theta \) are matrices of partial derivatives. In the analogous linear problem

\[
\dot{x}(t) = (A + \Theta B)x(t) \quad (5.4)
\]

the "first approximation" solution corresponding to (5.2) becomes

\[
x(t) \approx x_0(t) + \delta \Theta \int_0^t \bar{\alpha}(t-\tau)B\bar{\alpha}(\tau) \dot{x}(0) , \quad \text{where}
\]

\[
\bar{\alpha}_0(t) = e^{(A + \Theta_0 B)t} . \quad (5.5)
\]

The parenthesized term on the right hand side of (5.5) will be recognized as the first-order terms in the series expansion given by Bellman (12, p. 175),
Thus, even in the linear case, if the variation is not "sufficiently small" the error in the "first approximation" sensitivity method may necessarily dictate a "higher-order" analysis, which in itself suffers from increased complexity.

Toward the goal of determining quantitative bounds for "large" perturbation parameter sensitivity effects on the nominal solution to (5.4), it must be remarked that it is defective rational to expect that all of the "perturbed" solutions will be bounded between the two solutions associated with the parameter values at the endpoints of the perturbation interval.¹

The interval algorithm developed in Chapter IV then may be considered as a numerical technique which computes bounds (albeit, perhaps conservative) for the "large" perturbation parameter sensitivity effects on the nominal solution to (5.4), including the effects of algorithmic numerical truncation errors.

¹This will be demonstrated in the discussion of the parameterized RLC circuit example which follows.
The parameterized RLC circuit—a second-order problem

The differential equation of this example depicts the transient that arises upon closing a series circuit which consists of a resistor, inductor, capacitor and battery. Using electrical charge, \( q(t) \), as the dependent variable, the differential equation is

\[
L \ddot{q} + R \dot{q} + \frac{1}{C} q = E \ u(t) , \quad \text{where}
\]

\[
u(t) =
\begin{cases}
1 , & t \geq 0 \\
0 , & t < 0
\end{cases}
\]  

and the initial conditions \( q(0) \) and \( \dot{q}(0) \) are zero. The normalized\(^1\) circuit values are \( R = 0.4 \ (\pm 0.2) \) ohms, \( L = 1.0 \) henries, \( C = 1.0 \) farads and \( E = 1.0 \) volts.

Equation 5.7 corresponds to the frequently analyzed classical unit step driven constant coefficient linear second-order differential equation (53, pp. 148-152)

\[
\ddot{x} + 2 \delta \omega_n \dot{x} + \omega_n^2 x = u(t) ,
\]

with the undamped natural frequency \( \omega_n = 1.0 \) and where the system damping ratio is tolerated, \( \delta = 0.2 \ (\pm 0.1) \). The equation can be written in a driven "companion system" form

\(^1\)These normalized values correspond to the scaling of a "practical" circuit where the real time variable \( t' \) is scaled by the frequency normalizing constant (54, pp. 6-8) \( f = 1000 \) according to \( t' = t/f \) and the circuit values are \( R = 0.4 \ (\pm 0.2) \) ohms, \( L = 1.0 \) millihenries, \( C = 1000.0 \) microfarads and \( E = 1000.0 \) volts.
and the driving function can be augmented into a larger homogeneous system by defining the variables $x_1 = q$, $\dot{x}_1 = x_2$ and $\dot{x}_3 = 0$ ($x_3 = \text{Eu}(t)$ and let $\theta$ denote the value of $R$ in the "toleranced" interval $[0.2, 0.6]$), deriving the linear three state system

$$\dot{x} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & -\theta & 1 \\ 0 & 0 & 0 \end{bmatrix} x, \quad x(0) = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}. \quad (5.9)$$

In order to demonstrate that all of the solutions of (5.7) will not necessarily be bounded between the two solutions associated with the parameter values at the endpoints of the perturbation interval, employing the known analytic solution for $q(t)$ (53, p. 150)

$$q(t) = x_1(t) = 1 - e^{-\delta t} \cos(\sqrt{1-\delta^2} t - \theta)$$

where

$$\theta = \tan^{-1} \frac{\delta}{\sqrt{1-\delta^2}}. \quad (5.10)$$

double-precision solution values for the nominal and endpoint values of the damping ratio parameter were calculated in time incremental steps of 0.02 for the small time interval $[4.5, 4.7]$ and the enlarged SIMPLOT results are shown in Figure 5.1.

The problem was input to the interval algorithm utilizing the following operating control parameters discussed
Figure 5.1. Solution response for the second-order example (calculated in double-precision) for three damping ratio values
in Chapters III and IV:

1. $\text{NSUB}=25$, the number of parameter interval subdivisions (4.14);

2. $\text{MODE}=2$, the transpose Householder matrix norm (2.55);

3. $\text{P}=4$, the subproblem pseudo-fundamental matrix element "pass-fail-augment" error test inequality parameter (4.25);

4. $\text{AUGMNT}=0000000000000000$, the bounding interval arithmetic COMPLEX function parameter;

5. $\text{NSTEPS}=36$, the number of steps in the time increment for which the solution is desired;

6. $\text{TDELTA}=0.5$, the value in seconds of the solution time increment;

7. $\text{KK}=12$, the starting algorithm value for the number of terms to be initially attempted (that is, for $t = 1+\text{DELTA}$) in the computation of each of the first subproblem (I,J)th pseudo-fundamental matrix elements;

8. $\text{NKNT1}=3$ and $\text{NKNT2}=1$, the respective single- and double-fault error count limits acceptable in the computation of each set of subproblem pseudo-fundamental matrix elements; and

9. $\text{NFAIL}=1$, the number of times the subproblem pseudo-fundamental matrices may be computed with error counts exceeding those given in (8) above and
still be accepted before the "rolling storage" mechanism is placed in effect.

The algorithm computational execution time was 82.8 seconds and the "rolling storage" solution mechanism began operation using the 7*TDELTA computed subproblem pseudo-fundamental matrices which contained no single-fault and 14 double-fault elements in the total of 225 elements. The final KKPAST array for 7*TDELTA was

\[
\text{KKPAST} = \begin{bmatrix}
20 & 20 & 19 \\
20 & 20 & 20 \\
12 & 12 & 12
\end{bmatrix}
\]

The resulting algorithm plots are shown in Figure 5.2 where the (a) figure gives the interval bound solution for \( x_1(t) = q(t) \) with \( x_3(t) = u(t) \) superimposed and the (b) figure gives the interval bound solution for \( x_2(t) = \dot{q}(t) \). Comparison of the interval bound solutions of Figure 5.2a with the transient response curves for the unit step input second-order system with damping ratio values on the interval \([0.1, 0.3]\) (26, p. 88) effectively demonstrates that the algorithm has successfully produced response envelopes.

\[1\] These values indicate the number of terms which the algorithm used in the series computation (\( K \) in Equation 2.67) for the 25th subproblem pseudo-fundamental matrix \((I, J)\) elements for \( t = 7 \times T \Delta T \). The algorithm storage limit is 20 and the array indicates that for the \((1, 3)\) element, \( k = 19 \) was the final value used.
Figure 5.2. Interval bound solutions for the second-order example ($\delta = [0.1, 0.3]$)

a. top: $x_1(t) = q(t)$

b. bottom: $x_2(t) = \dot{q}(t)$
The linearized instrument servomechanism with load inertia parameter—a fourth-order problem

The example in this section is encountered in a practical text on instrument servomechanism design by Chubb (55). Figure 5.3 presents the original nonlinear system block diagram for the positional motor-generator instrument servomechanism. Since the reference begins analysis with the linearization of the system, the three nonlinear blocks have been drawn omitting the nonlinear "corner" designations. The following list of parameters will assist the perusal of the block diagram, where it will be noted that the motor has been referred through the gear train to the load side:

\[ K_f = \text{followup gain effective at the amplifier (v/rad.)}; \]
\[ K_g = \text{generator gain effective at the amplifier (v/1000 rpm)}; \]
\[ K_a = \text{amplifier gain (v/v)}; \]
\[ K_m = \text{motor torque gain (oz-in/v)}; \]
\[ \phi_f = \text{followup carrier phase shift effective at the amplifier (deg.)}; \]
\[ \phi_g = \text{generator carrier phase shift effective at the amplifier (deg.)}; \]
\[ \phi_a = \text{amplifier carrier phase shift (deg.)}; \]
\[ N = \text{gear ratio (N rev. motor = 1 rev. load)}; \]
Figure 5.3. Nonlinear positional motor-generator instrument servomechanism
\( B_m \) = motor damping (oz-in/rad/sec.);

\( J_L \) = motor inertia (gm-cm\(^2\));

\( K_s \) = gear train stiffness (oz-in/rad.);

\( \Theta_L \) = load displacement (rad.);

\( \Theta_m \) = motor displacement (rad.);

\( \Theta_i \) = positional displacement input (rad.);

\( T_L \) = load torque (oz-in);

\( T_f \) = load friction torque (oz-in);

\( T_s \) = stall torque (oz-in);

\( T_d \) = motor damping torque (oz-in);

\( T_g \) = gear train torque (oz-in);

\( E_f \) = followup voltage (v); and

\( E_g \) = generator damping voltage (v).

The linearized system is obtained by (1) replacing the coulomb friction by viscous damping \( f_c \) (oz-in/rad/sec.), (2) setting the motor starting voltage equal to zero, (3) assuming that the amplifier saturation limits are sufficiently large so that they may be neglected and (4) removing the backlash deadzone.

Defining the variables \( x_1 = \dot{\Theta}_m, x_2 = \Theta_m, x_3 = \dot{\Theta}_L, x_4 = \Theta_L, x_5 = \Theta_i \), assuming that the followup, generator and amplifier carrier phase shifts are each zero and assuming a unit
step positional displacement input, the homogeneous first-order linear differential equation system may be written by inspection of Figure 5.3 as

\[
\begin{bmatrix}
-\frac{B_m + K_T a_m}{J_m} & -\frac{K_s}{N^2 J_m} & 0 & \frac{K_s - K_T a N K_m}{N^2 J_m} & \frac{K_T K a_m}{N J_m} \\
1 & 0 & 0 & 0 & 0 \\
0 & \frac{K_s}{J_L} & -\frac{f_L}{J_L} & -\frac{K_s}{J_L} & 0 \\
0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 \\
\end{bmatrix}
\]

\[
X = \begin{bmatrix}
X(0) = (0 0 0 0 1)^T
\end{bmatrix}
\]

The unit system for the calculations will be the oz-in-sec system and the conversion factor for inertia is \(1.42 \times 10^{-5}\) oz-in-sec\(^2\)/gm-cm\(^2\). The following coefficient values are given in the reference (55, p. 103) with the exception of the tolerance on the load inertia:

\[
K_T = 18.0 \text{ v/rad.} \quad K_m = 0.0026 \text{ oz-in/v}
\]

\[
K_a = 240.0 \text{ v/v} \quad J_L = 3000.0 (\pm 300.0) \text{ gm-cm}^2
\]
\( K_g = 0.14 \text{ v/1000 rpm} \quad K_s = 1500.0 \text{ oz-in/rad} \)

\( B_m = 0.0002 \text{ oz-in/rad/sec} \quad N = 100.0 \)

\( J_m = 1.0 \text{ gm-cm}^2 \quad f_L = 0.0 \text{ oz-in/rad/sec} \)

The 10\% tolerated load inertia arises in the denominator of the expressions for two of the elements in the coefficient matrix of (5.11). In order to insure the "signed" load inertia parameter dependence in the interval algorithm, the two "interval" coefficients are computed as

\[
\begin{bmatrix}
-a_{34}^L & a_{34}^R \\
-a_{32}^L & -a_{32}^R
\end{bmatrix} = \begin{bmatrix}
\frac{K_s}{J_L} & -\frac{K_s}{J_L} \\
\frac{K_s}{J_L} & -\frac{K_s}{J_L}
\end{bmatrix},
\]

where

\[
J_L \equiv [J_L^L, J_L^R] = [2700.0, 3300.0].
\]

The problem was input to the interval algorithm with the following operational control parameters (see the discussion of the preceding example):

1. NSUB=25
2. MODE=2
3. P=3
4. TDELT=0.0035
5. TDELTA=0.0035
6. MODE=2
7. KK=12
8. NKNT1=5

\(^1\text{This value is incorrectly given in the reference (55, p. 103) as 6000.0. For the nominal value of } J_L = 3000.0 \text{ and only for the value } K_s = 1500.0 \text{ is it possible to reproduce the } \Theta_L \text{ response curve of the reference, using a standard single-precision integration package.}\)
The algorithm computational execution time was 170.3 seconds and the "rolling storage" solution mechanism began operation using the 7*TDELTA computed subproblem pseudo-fundamental matrices which contained 6 single-fault and 18 double-fault elements in the total of 625 elements. The final KKPAST array for 7*TDELTA was

\[
\begin{bmatrix}
20 & 20 & 20 & 20 & 19 \\
19 & 20 & 20 & 20 & 18 \\
20 & 19 & 20 & 19 & 19 \\
19 & 20 & 19 & 20 & 18 \\
12 & 12 & 12 & 12 & 12
\end{bmatrix}
\]

The interval algorithm results are shown in Figure 5.4 where the discrete time interval solution bound endpoints are plotted. Superimposed on each plot are continuous solution curves for the nominal and endpoint values of the load inertia parameter which were obtained using the single-precision SYSLIB numerical integration package NODE\(^2\).  

\(^1\)The first example was input to an earlier version of the interval algorithm which did not contain the NRUNUP limit feature described in Chapter IV.  

\(^2\)NODE uses a 4th-order predictor-corrector technique which initially computes the necessary backpoints by the Runge-Kutta-Gill single-step method (40, pp. 122, 194).
Figure 5.4. Interval bound solution endpoints for the fourth-order example ($J_L = [2700.0, 3300.0]$)

a. top: $x_1(t) = \dot{\theta}_m(t)$

b. bottom: $x_2(t) = \theta_m(t)$
Figure 5.4 (Continued)
c. top: $x_3(t) = \dot{\theta}_L(t)$
d. bottom: $x_4(t) = \theta_L(t)$
Examination of Figures 5.4a through d comparing the interval bound solution points with the continuous curves produced by the numerical integration package NODE for the three values of the load parameter successfully demonstrates the ability of the interval algorithm to produce "tracking" solution envelopes for the load perturbation parameter system. It will be observed that for increasing time the trend of the solution envelopes is to become more conservative, reflecting the effects of the bounding interval arithmetic and the "rolling solution" mechanism of the algorithm.

The minimum plant sensitivity optimal linear regulator design—a fifth-order problem

The example presented in this section is one which Ćirić' (36) uses to demonstrate the results of his research (see Chapter II), the minimizing of plant perturbation effects on the cost functional for an optimally compensated nominal plant design by selecting the arbitrary parameter values in the optimal compensator which satisfy a first-variational analysis.

The completely controllable and completely observable linear state variable system to be compensated is given by (2.104), where
A_{11} = -2.0 and \( a^{22} = -1.0 \). The negative feedback compensated system of the research optimally minimizes the performance index (2.105) for the nominal plant, where in this example the minimum value of \( p=1 \) satisfies the design criteria

\[
\text{rank}[C^T, A_0^T C^T, ..., (A_0^T)^p C^T] = n
\]

and the performance index parameter values are selected as \( Q = I, \gamma_0 = 1 \) and \( \gamma_1 = 1 \). The reformulated system (2.104a) with

\[
z = \begin{bmatrix} x \\ u \end{bmatrix}, \quad \hat{A}_0 = \begin{bmatrix} A_0 & b \\ 0^T & 0 \end{bmatrix} \quad \text{and} \quad \hat{b} = (0 \ 0 \ 0 \ 0 \ 1)^T
\]

and where the "control" \( U \) is to optimally minimize the reformulated performance index (2.105a) with

\[
\hat{Q} = \begin{bmatrix} Q & 0 \\ 0^T & \Gamma \end{bmatrix}, \quad \Gamma = \gamma_0 \quad \text{and} \quad \gamma = \gamma_1
\]
is the well known linear state-regulator optimization problem.

For the resulting nominal plant optimal compensator design in this example, examination of (2.113) indicates that \( m(p+1) - n \equiv k = 2 \) of the compensator \( \beta_i \) coefficients are arbitrary. The arbitrary coefficients are selected with \( \hat{\beta} = (\beta_1^1, \beta_1^2)^T \) and (2.113) is reformulated as (2.114), where the vector \( \mathbf{k} \) is the result of the optimal linear state-regulator solution.

The perturbed linear system with the nominal plant optimal compensator will be analyzed in the state variable form

\[
\dot{\mathbf{x}} = \hat{\mathbf{A}} \mathbf{x} + \hat{\mathbf{b}} \mathbf{u}, \quad \text{where} \quad \hat{\mathbf{A}} = \begin{bmatrix} \mathbf{A} & \mathbf{b} \\ \cdots & \cdots \\ \mathbf{0}^T & 0 \end{bmatrix}
\]  

(5.12)

and \( \mathbf{u} \) is given by (2.115). This reflects the fixed nominal plant optimal compensator design with the arbitrary design parameter values unspecified and with the perturbed plant. (The actual nominal plant optimal compensator design will not be considered here and would be obtained from Equations 2.107, 2.109, 2.110, 2.111, 2.112, 2.113 and 2.114.)

The formulation for \( \mathbf{u} \) given in (2.115) may be realized by "substituting" the optimal nominal plant design result (2.114) and the perturbed plant value \( \mathbf{A} \) in (2.110), (2.111) and (2.112). The matrices in (2.115) are then found to be
\[
\gamma^T = (0 \ 0 \ k_4 \ 0 \ 0) \quad \text{and} \quad \kappa^T = \begin{bmatrix}
1 & 0 & -1 & 0 & 0 \\
0 & 1 & 0 & 0 & 0
\end{bmatrix}.
\]

Substituting these into (2.115) and the resulting expression for \( U \) into (5.12) produces the homogeneous state variable system (which models the perturbed plant, the fixed nominal plant optimal compensator design and the effects of the arbitrary \( \hat{p} \))

\[
\dot{\mathbf{z}} = \begin{bmatrix}
a_{11} & 1 & 0 & 0 & 0 \\
0 & a_{22} & 1 & 0 & 0 \\
0 & 0 & -1.5 & 1 & 0 \\
0 & 0 & 0 & -0.5 & 1 \\
a_{51} & a_{52} & a_{53} & a_{54} & a_{55}
\end{bmatrix} \mathbf{z},
\]

(5.13)

where

\[
a_{51} = -\beta_1^1 (a_{11} + 2) - k_1,
\]

\[
a_{52} = -\beta_2^2 (a_{22} + 1) - k_2,
\]

\[
a_{53} = -k_3,
\]

\[
a_{54} = -k_4 \quad \text{and}
\]

\[
a_{55} = -k_5.
\]

In this example, the initial condition is selected as \( \mathbf{z}(0) \equiv \mathbf{z}_0 = (1 \ 1 \ 1 \ 1 \ 1)^T \).
The value of the arbitrary coefficients vector \( \hat{\beta} = (\beta_1^1, \beta_2^1)^T \) which yields the minimum first-order sensitivity of the cost functional (2.105) with respect to the \( \ell = 2 \) vector \( \text{a} = (\text{a}_1^1, \text{a}_2^1)^T \) for the optimally compensated nominal plant system is found to be \( \hat{\beta}^* = (0.417, 1.588)^T \). Although the computation of \( \hat{\beta}^* \) is largely algebraic, a procedural listing is given here to provide an understanding of the computations which were required:

1. (2.108) - find \( P_\infty \) using the eigenvector matrix inversion technique indicated
2. (2.107) - find \( \hat{k} \)
3. (2.116) -
   a. find the \( (n+p) \times (n+p) \) matrix \( \hat{P} \)
   b. find \( \hat{P}^{-1} \) and the \( (n+p) \)-vector \( \text{v} = \hat{P}^{-1} \text{z}_0 \)
   c. find the \( (n+p) \times (n+p) \) matrix \( \hat{K} \)
   d. find the \( (n+p) \times (n+p) \) matrix \( \text{N} \)
   e. find the \( (n+p) \times (n+p) \) matrix \( \text{E} \)
   f. find the \( (n+p) \)-vector \( \text{w} \)
   g. find the distinct \( n+p \) eigenvalues of \( \Lambda \)
   h. find the diagonal \( (n+p) \times (n+p) \) matrix \( \tilde{M} \)
   i. find the \( (n+p) \times (n+p) \times \ell \) tensor \( \left( \frac{\partial \hat{A}_0}{\partial \text{a}} \right)_{klj} \)

\(^1\text{Note, therefore, that the minimizing } \hat{\beta}^* \text{ will be a function of the initial condition.}\)
j. find the \( k \times l \) matrix \( D^T \) from the indicated matrix tensor product

k. find the \( (n+p) \times (n+p) \times l \) tensor \( (R)_{ijk} \) from the indicated matrix tensor product

l. find the \( (n+p) \times (n+p) \times l \) tensor \( (T)_{ijk} \)

m. find the transpose \( l \)-vector \( \bar{d}^T \) from the indicated vector tensor product

n. since \( D^T \) is not singular, find the minimum sensitivity "arbitrary" coefficient vector value \( \hat{\xi}^* = -D^{-1}\bar{d} = (+0.417,+1.588)^T \)

Assume that the two plant parameter variations for this example are linearly dependent on one scalar parameter. In this sense then, the example values (36) for the algorithm input plant perturbation parameter "intervals" should be \( a_{11} = [-2.0,-1.8] \) and \( a_{22} = [-1.0,-1.5] \) and the corresponding effects of the perturbation parameter on the homogeneous state variable model (5.13) should be input as the "intervals"

\[
\begin{align*}
a_{51} &= [-k_1,-\beta_1^1(-1.8+2.0)-k_1] \quad \text{and} \\
a_{52} &= [-k_2,-\beta_2^2(-1.5+1.0)-k_2] \quad \text{.} 
\end{align*}
\]

(5.14)

It should be remarked that this forces a "signed" parameter dependence of the algorithm with the effect that for the nominal value of \( a_{11} \) and \( a_{22} \), the model will exactly represent the optimally compensated unperturbed nominal system.
In order to appraise the effects on the perturbed system performance of selecting the arbitrary parameters in the nominal plant optimal compensator which minimizes the first-variation of the cost functional, the problem (5.13) was input to the interval algorithm with the "interval" values $a_{51}$ and $a_{52}$ in (5.14) computed for $\hat{\beta} = \hat{\beta}^*$ and $\hat{\beta} = \hat{\beta}^+ = (10.0, -10.0)^T$. In each case the algorithm operational control parameters were as follows:

(1) $\text{NSUB}=25$  
(2) $\text{MODE}=1$  
(3) $\text{P}=3$  
(4) $\text{AUGMNT}=00000000F0000000$  
(5) $\text{NSTEPS}=36$  
(6) $\text{TDELTA}=0.5$  
(7) $\text{KK}=12$  
(8) $\text{NKNT1}=5$  
(9) $\text{NFAIL}=1$  
(10) $\text{NRUNUP}=1$

The algorithm computational execution times were 236.1 seconds for the $\hat{\beta}^*$ problem and 198.5 seconds for the $\hat{\beta}^+$ problem. In the $\hat{\beta}^*$ case, the "rolling solution" storage mechanism began operation using the $8\ast\text{TDELTA}$ computed subproblem pseudo-fundamental matrices which contained 17 single-fault and 57 double-fault elements in the total of 625 elements (the corresponding $7\ast\text{TDELTA}$ computations contained 4 single- and 0 double-fault elements). The elements of the KKpast array for $8\ast\text{TDELTA}$ were each 20 with the exception of the $(i,j)$, $i=2,3,4$; $j=4,5$ elements which were
each 19. In the $\hat{\beta}^+$ case, the "rolling solution" storage mechanism began operation using the $7^*TDELTA$ computed subproblem pseudo-fundamental matrices which contained 9 single-fault and 82 double-fault elements in the total of 625 elements (the corresponding $6^*TDELTA$ computations contained 5 single- and 0 double-fault elements). The elements of the KKPAST array for $7^*TDELTA$ were each 20 with the exception of the (3,5) element which was 19.

The resulting algorithm plots of the interval bound solutions for each of the state variables are shown in Figure 5.5a through j, where for each state the $\hat{\beta}^*$ and $\hat{\beta}^+$ results are placed on the same page for comparison. Examination of the $\hat{\beta}^*$ and $\hat{\beta}^+$ interval bound solutions for each of the state variables vividly defends the feasibility of selecting the minimum sensitivity $\hat{\beta}^*$ for the nominal plant optimal compensator design. Additionally, as a result of the "signed" algorithm dependence on a single perturbation parameter (see 2.115 and 5.14), both the $\hat{\beta}^*$ and $\hat{\beta}^+$ interval bound solutions for the homogeneous state variable system models (5.13) must contain the unperturbed nominal plant optimally compensated solution trajectory. Utilizing this property, it is possible to investigate the "intersection" of the two interval bound solutions by overlaying the state plots and interestingly observe that, for the following states and time intervals the unperturbed optimal trajectory is
Figure 5.5. Interval bound solutions for the fifth-order example ($a_{11} = [-2.0,-1.8]$, $a_{22} = [-1.0,-1.5]$)

a. top: $x_1(t)$ for $\hat{\beta}^*$
b. bottom: $x_1(t)$ for $\hat{\beta}^+$
Figure 5.5 (Continued)
c. top: $x_2(t)$ for $\hat{\beta}^*$
d. bottom: $x_2(t)$ for $\hat{\beta}^+$
Figure 5.5 (Continued)
e. top: $x_3(t)$ for $\hat{\beta}^*$
f. bottom: $x_3(t)$ for $\hat{\beta}^+$
Figure 5.5 (Continued)
g. top: $x_4(t)$ for $\hat{\beta}^*$
h. bottom: $x_4(t)$ for $\hat{\beta}^+$
Figure 5.5 (Continued)
i. top: $u(t)$ for $\hat{\beta}^*$
j. bottom: $u(t)$ for $\hat{\beta}^+$
precisely indicated since the only intersection is the lower bound for \( \hat{E}^* \) and the upper bound for \( \hat{E}^+ \):

\[
\begin{align*}
x_1(t), & \quad t = [4.0, 5.25] \\
x_2(t), & \quad t = [4.0, 4.75] \\
x_3(t), & \quad t = [0.0, 4.00] \\
x_4(t), & \quad t = [0.0, 3.25] \\
u(t), & \quad t = [0.0, 2.00]
\end{align*}
\]

Bounding Interval Arithmetic—Subdistributivity

Considerations

Since the machine implementations of the bounding interval arithmetic considered in Chapter III produce numerical bounds for the single-precision floating-point arithmetic, it appeared feasible to use the concept as an empirical instrument to investigate and compare the results when applied to contrasting algorithmic techniques encountered in certain problems in automatic control theory, perhaps pointing toward a premise of "good" computer programming technique of adhering to the concept of "subdistributivity" (2.5) for even the "real" machine arithmetic.

Since the Kalman estimation algorithms precisely belong to this class of problems, the following two estimation problems were investigated in this respect and the results are
presented here. (While the chronology of the investigation of the two examples was the reverse of the order of discussion here, they are nonetheless so ordered in keeping with the natural development of the mathematical models presented in Chapter II.)

Computation order dependence in an undriven observable random walk process covariance equation—an observation

The estimation algorithm considered in this section is the Kalman covariance equation (see Chapter II) investigated by Duven (30, pp. 114-122) for a random process which has a 2-dimensional random walk mode that is undriven and observable. The process dynamics, measurement connection, and process disturbance and measurement error covariances are time invariant and are given by\(^1\)

\[
\Phi = \begin{bmatrix} -1 & 4/3 \\ -3 & 3 \end{bmatrix} = \begin{bmatrix} 2 & -1 \\ 3 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & -1 \\ 3 & 0 \end{bmatrix}^{-1}, \quad H = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}
\]

\[
M = (-1 \quad 1) \quad \text{and} \quad V = \begin{bmatrix} 1 \end{bmatrix}
\]  

The only equilibrium solution to the covariance equation in

---

\(^1\)Because of the time invariance associated with this model, the time dependency subscript \(k\) may be omitted from the given matrices. The transition matrix is also written in its Jordan decomposition \(T \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} T^{-1}\) displaying the random walk modes and since the decoupled system measurement connection matrix is \(MT = (1 \ 1)\), both modes are observable (23, p. 432).
this problem is the null matrix (30, p. 115).

Substituting the alternate Sorenson form (2.102) of the a posteriori covariance matrix into (2.98), the covariance equation may be rewritten as

\[ \mathbf{P}_{k+1}^* = (\mathbf{a} - \mathbf{aK}_k \mathbf{M}) \mathbf{P}_k^* (\mathbf{a} - \mathbf{aK}_k \mathbf{M})^T + (\mathbf{aK}_k) \mathbf{V} (\mathbf{aK}_k)^T + \mathbf{H} \quad (5.16) \]

Defining the matrices

\[ \mathbf{F}_k \equiv \mathbf{a} - \mathbf{aK}_k \mathbf{M} \quad , \]

\[ \hat{\mathbf{K}}_k \equiv \mathbf{aK}_k \quad \text{and} \]

\[ \mathbf{W}_k \equiv \mathbf{M}\mathbf{P}_k^* \mathbf{M}^T + \mathbf{V} \quad \text{(see Equation 2.94)} \quad , \quad (5.17) \]

then

\[ \hat{\mathbf{K}}_k = \mathbf{aP}_k^* \mathbf{M}^T \mathbf{W}_k^{-1} \quad , \]

\[ \mathbf{F}_k = \mathbf{a} - \hat{\mathbf{K}}_k \mathbf{M} \quad \text{and} \]

\[ \mathbf{P}_{k+1}^* = \mathbf{F}_k \mathbf{P}_k^* \mathbf{F}_k^T + \hat{\mathbf{K}}_k \mathbf{W}_k \hat{\mathbf{K}}_k^T + \mathbf{H} \quad . \quad (5.18) \]

The matrices \( \mathbf{W}_k \), \( \hat{\mathbf{K}}_k \) and \( \mathbf{F}_k \) must be computed (dropping the \( k \) subscripts for convenience) by

\[ \mathbf{W} = \sum_{i=1}^{2} \sum_{j=1}^{2} (m_i \mathbf{p}_{ij}^*) \mathbf{m}_i + \mathbf{V} \quad , \]

\[ \text{These computation orders must be followed in order to duplicate the results of Duven.} \]
\[ \hat{k}_i = \left( \sum_{p=1}^{2} \sum_{q=1}^{2} \phi_{ip} p_{pq} m_q \right) / w \] and

\[ f_{ij} = \phi_{ij} - \hat{k}_i m_j, \quad (5.19) \]

where the computation order is precisely as indicated. The updated \((t_{k+1})\) covariance equation result must be computed in "expanded" (5.20) and "nested" (5.21) forms (observing exact computation order) by

\[ p_{ij}^*(t_{k+1}) = \left( \sum_{p=1}^{2} \sum_{q=1}^{2} (f_{ip} p_{pq}^* f_{jq}) + (\hat{k}_i V) k_j \right) + h_{ij} \quad (5.20) \]

and

\[ p_{ij}^*(t_{k+1}) = \left( \sum_{p=1}^{2} \sum_{q=1}^{2} (f_{ip} \sum_{q=1}^{2} p_{pq}^* f_{jq}) + (\hat{k}_i V) k_j \right) + h_{ij} \quad (5.21) \]

Using the single-precision arithmetic with an initial value of

\[ p_0^* = \begin{bmatrix} 0.1 & 0 \\ 0 & 0 \end{bmatrix}, \]

Duven plotted the traces of the covariance equation results for (5.20) and (5.21) versus \(k\), illustrating the vivid

\[ ^1 \text{These computation orders must be followed in order to duplicate the results of Duven.} \]
truncation arithmetic effects on the solution behavior. The single-precision expanded form (5.20) trace closely approximated its stable double-precision computed counterpart which monotonically first increased to a peak positive value of approximately 4.8 at \( k = 4 \) and then decayed toward zero in an exponential appearing fashion (with approximate values of 1.0 at \( k = 37 \) and 0.1 at \( k = 396 \)). The single-precision nested form (5.21) trace began to diverge in the negative direction from the (5.20) values at \( k = 147 \), became negative for the first time at \( k = 238 \) and subsequently executed several cotangent appearing transitions from large negative values to large positive values.

From an interval arithmetic point of view, computations for the nested form (5.21) should be contained in those for the expanded form (5.20), which is contrary to the single-precision arithmetic results described above. Because of this "contradiction" in the expected behavior from an interval arithmetic outlook, the two methods were programmed.

---

Duven's original program was written in PL-1. In attempting to first obtain exactly his R*4 results using the Fortran G compiler, it became necessary (because of the extreme numerical sensitiveness to computational order) to obtain an assembly listing of several of his PL-1 equations to determine the exact order in which that compiler performed the computations. Parenthetically, it was also learned that the WATFIV and G compiler computations differed and that the WATFIV initialization of a R*8 variable with a R*4 value (arithmetic assignment) did not clear the low-order 8 hexadecimal digits as the G compiler did. Only then
using the pseudo-bounding interval arithmetic implementations of Chapter III. Initially it was decided to separately compute the interval results from (5.20) and (5.21) using the bounding interval arithmetic parameter hexadecimal value \( \text{AUGMNT}=00000000\text{F0000000} \), which would intrinsically produce bounds for the truncated single-precision "guard digit" arithmetic of the IBM 360/65 machine (see Chapter III). In addition, the nested form of the standard covariance equation (2.101),

\[
P_{k+1}^* = \delta \{ P_k^* [I - M_k^T (M_k P_k^T + V)^{-1} M_k^T] \} \delta^T + H \quad , \quad (5.22)
\]

was also programmed in the interval arithmetic and the results of the three methods are given in Table 5.1. Conjointly, the hexadecimal R*4 results for the three methods are also given for the iterations. (The parenthesized number is the last one or two hexadecimal digit base 16 difference with respect to the double-precision rounded result.)

While for each iteration the results demonstrated that even using the guard digit bounding interval arithmetic the

\[1\]

was it possible to program the methods in the interval arithmetic. The PL-1 duplicated computation orders are exactly those given in Equations 5.19, 5.20 and 5.21.

\[1\] Referring to the discussions of Chapter III, it is easy to see that the above operation of the functions MUL and DIV produce exact bounds for the R*4 "guard digit" arithmetic. However, in the case of ADD and SUB, the results go somewhat beyond exact bounds for the guard digit arithmetic producing bounds as a result of fraction alignments beyond
nested interval results were contained in the expanded interval results, it was disappointing to learn that the results became so conservative so rapidly that the sixth iteration in each technique produced "singular" values of $W_k$, terminating each algorithm. It may also be noted that for the first two iterations the standard form produced the least conservative interval results, while for the next three iterations the nested form produced the best results. Examining the $R^4$ hexadecimal results of the three methods and the parenthesized hexadecimal difference with respect to the single-precision rounded $R^8$ hexadecimal result, it may be noteworthy that the expanded results were always less than the double-precision results and for the peak value at $k=4$, the other methods produced values which were both larger than the double-precision result. Since $k=4$ represents the "spike" position of the trace, perhaps this negative error may have been an "indicator" to the future behavior of the nested and standard form $R^4$ "cotangent

the guard digit (into the 8th through 14th fraction digits and the 15th fraction double-precision guard digit) and the effect of the double-precision arithmetic operation on the $R^4$ guard digit.

"Singular" in this context indicates that the interval $W_k$ contains zero and therefore $W_k^{-1}$ is not defined. For this reason an interval algorithm trap stopped the computations.
Table 5.1. Interval and $R^*4$ (hexadecimal) results for the trace of the covariance equation, $\text{Tr}(P^*_K)$, computed three ways

<table>
<thead>
<tr>
<th>Equation 5.16</th>
<th>Equation 5.22</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;expanded&quot; form (5.20)</td>
<td>&quot;nested&quot; form (5.21)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$\text{Tr}(P^*_1)$</th>
<th>[0.9090899E00, 0.9090911E00]</th>
<th>[0.9090900E00, 0.9090906E00]</th>
<th>0.9090906D00</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>40E8BA1F(-B)</td>
<td>40E8BA1F(-B)</td>
<td>40E8BA20(-A)</td>
</tr>
<tr>
<td>$\text{Tr}(P^*_2)$</td>
<td>[0.2999954E01, 0.3000042E01]</td>
<td>[0.2999963E01, 0.3000024E01]</td>
<td>0.2999999D01</td>
</tr>
<tr>
<td></td>
<td>412FFFB(-4)</td>
<td>412FFFB8(-7)</td>
<td>412FFFF9(-6)</td>
</tr>
<tr>
<td>$\text{Tr}(P^*_3)$</td>
<td>[0.4411403E01, 0.4421932E01]</td>
<td>[0.4413381E01, 0.4419961E01]</td>
<td>0.4416667D01</td>
</tr>
<tr>
<td></td>
<td>414D3332(-4)</td>
<td>414D333D(+7)</td>
<td>414D3338(+2)</td>
</tr>
<tr>
<td>$\text{Tr}(P^*_4)$</td>
<td>[0.3983109E01, 0.5669246E01]</td>
<td>[0.4450019E01, 0.5201854E01]</td>
<td>0.4825003D01</td>
</tr>
<tr>
<td></td>
<td>414D3332(-4)</td>
<td>414D333D(+7)</td>
<td>414D3338(+2)</td>
</tr>
<tr>
<td>$\text{Tr}(P^*_5)$</td>
<td>[-0.6233402E04, 0.7189414E04]</td>
<td>[-0.6391095E02, 0.9069911E02]</td>
<td>-0.1532338E04,</td>
</tr>
<tr>
<td></td>
<td>414B52A4(-17)</td>
<td>414B52D2(+17)</td>
<td>414B52AF(-C)</td>
</tr>
<tr>
<td>$W_6^a$</td>
<td>[-0.1193050E05, 0.1220159E05]</td>
<td>[-0.1304617E03, 0.1359403E03]</td>
<td>-0.2544175E04,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.2536532E04</td>
</tr>
</tbody>
</table>

$^a$In each computation technique, $W_K$ was computed by Equation 5.19.
appearing" instabilities which occurred much later. Im-
mediately following the "spike" (at k=5), the expanded form
had the largest negative error. (The R*4 standard method
trace initially became negative at k=219 as opposed to
k=238 for the nested method.) Along these lines, the in-
terval results may be indicative of "computational mode"
switching points for the algorithm.

Since the initial attempts to investigate the accumu-
lated effects of the "guard digit" truncation by using the
interval arithmetic for each computational form of the co-
variance equation terminated so early in the iterations,
the three interval techniques were incorporated into a
single program where each interval solution step for each of
the three methods was initialized with the degenerate inter-
val matrix generated from the truncated previous step R*8
covariance solution\(^1\) and the single-step interval covariance
matrix trace endpoint errors were computed relative to the
updated R*8 covariance matrix trace. The results of the
program are plotted in Figure 5.6 from which several ob-
servations may be made.

\(^1\) Necessary precautions were taken to insure that the
interval and R*8 covariance equation techniques each used
identical R*4 equation constants by reading degenerate in-
terval values in the first case and by using R*4 to R*8
arithmetic assignments in the second case. The interval matrix was input in hexadecimal form.
In every case the nested form produced less conservative results than the expanded form. (Although the entire plot has not been reproduced here, this was true through values of k=500.) For values of k up to 20, the nested form tended to be superior to the standard form. In the range of k=20 to k=60, the tendency was for the standard form to be less conservative while for larger values of k the nested form again produced superior results. From Duven's continuous plot of the $R^s_8$ covariance equation trace versus k (30, p. 121), for values of k larger than 4 the slope of the curve was negative ($d\text{Tr}(P_k^s)/dk < 0$). It was noticeably in the same relatively narrow range of k=20 to k=60 (where the standard form tended to yield superior interval results) that the slope of the curve made a decided change ($d^2\text{Tr}(P_k^s)/dk^2 > 0$) toward considerably less negative values. In this respect then, perhaps the interval results may indicate a range of values of k for which the standard computational method should be employed.

Two methods for the delayed-state inertial navigation problem solution—a comparison

The estimation problem of this section arises in a combined inertial/Doppler-satellite navigation system (32). A low altitude (600 mile) polar orbit satellite transmits a continuous radio frequency signal which contains precise orbital and time reference information and provides the
Figure 5.6. Single-step covariance trace guard digit interval arithmetic endpoint error results for three computational techniques initialized with R*8 covariance solution. Error = (R*4 guard digit endpoint result - R*8 solution result)/(R*8 solution result)
prime near-earth vehicle (in this example, stationary, 45°N, 0°W) inertial system with Doppler counts\(^1\) and the Kalman filter\(^2\) can be used to reset the inertial system with each satellite pass.

In his thesis, Stuva (34) considered the computational aspects of the implementation of the Kalman filter for this integrated system, comparing the results of the recursive Equations 2.98, 2.103a, 2.94a and 2.96a of Brown and Hartman (33) and a new computational scheme consisting of the recursive Equations 2.98, 2.103b, 2.94b and 2.96a.\(^3\) Since the purpose here was to employ the bounding interval arithmetic and compare the truncation arithmetic effects on the two computational schemes, the development and description of the physical model will not be given, except to indicate that the error dynamics (2.93) consists of a 15 state system.

---

\(^1\)The Doppler counts, which provide a measurement of the change in relative vehicle-satellite positions over the count time intervals, are assumed available in this example for a 12 minute pass over the navigation vehicle, with the satellite and vehicle located at the same latitude at the center of the pass.

\(^2\)The algorithm estimates the inertial system error forcing functions as well as the position errors.

\(^3\)In this sense then, the recursive equations do not involve the processing of any actual observed data. The first set of equations will be called the Hartman scheme while the second set will be referred to as the Stuva method.
with seven states arising from the "$\psi$-equations" and the "Schuler dynamics", three each from the body mounted gyro and accelerometer biases and one each from the altitude and Doppler count bias errors (34, p. 29).

The two algorithms were programmed in the Washington State bounding interval arithmetic and subsequently\(^1\) in the pseudo-bounding interval arithmetic (with AUGMNT = 0000000000000000000). Stuva had previously computed the time invariant one-second transition matrix process disturbance covariance matrix and initial value a posteriori error covariance matrix in double-precision and the resulting values were truncated to single-precision and input to the interval algorithm in the A4 format. The transition matrix was computed using the truncated matrix exponential series method, with the exception of the slow series converging elements $\phi_{ij}$, $i,j=8,14$ which were recomputed from their analytic solutions (the states $x_8$ and $x_{14}$ were coupled but uncoupled from the rest of the system). The initial value of the a posteriori error covariance matrix, $P_0$, was computed by using the variances typical of a one-knot

\(^1\) As previously remarked, this problem was investigated prior to the one of the previous section. Because of the excessive execution time necessary to operate the Washington State routines on this problem, the pseudo-bounding interval arithmetic COMPLEX functions were devised and the problem was reprogrammed using the new implementation as well.
inertial system and projecting ahead (without updating) for one hour using a 20 second transition matrix in the equation

\[ p_{i+1} = \alpha P_i \alpha^T + H . \]

Constants necessary to compute the time varying dynamics involved in the measurement connection matrices \( M_K \) and \( N_K \) were input to the interval algorithm as double-precision values in the AS format. A time invariant scalar measurement error covariance value of \( V = 25.0 \) was used.

The operation of the interval algorithm then was to initialize as degenerate interval matrices from the "R*4" Stuva computed input data, \( P_0, \alpha, H \) and \( V \). For each step of the two interval computation techniques, a subroutine was called to compute the R*8 time varying values of \( M_K \) and \( N_K \) and the truncated R*4 results were then placed in degenerate interval form. The interval algorithm was devised to trap a \( QK \) failure (interval containing zero) for either computation method and in that event to "shut off" that method. Because of the excessive computation times involved when using the Washington State bounding interval arithmetic, the algorithm was reprogrammed once taking advantage of the sparseness of the \( \alpha \) matrix (eliminating 40% of the interval arithmetic subroutine calls) and minimizing the computations involved in symmetrizing the \( P^*_K \) and \( P_K \).
matrices.

As a point of miscellaneous comparisons of the two computation methods and the two bounding interval arithmetic implementations, Table 5.2 contains data relative to the "identical" interval algorithm operations. Referring to this data, it should be remarked that the pseudo-bounding interval arithmetic algorithm computed a larger number of iterations than the Washington State bounding interval arithmetic algorithm and that the use of the pseudo-bounding interval arithmetic demonstrated a computational speed improvement of better than 17:1.

The graphical results of Figures 5.7, 5.8 and 5.9 were obtained by computing the respective differences between the Stuva and Hartman interval algorithm endpoint results ("sup" and "inf") and the R*8 computed Stuva results. The same type of error results are also shown for the R*4 Stuva and Hartman algorithms. The (a) figures represent the Washington State bounding interval arithmetic results and the (b) figures represent the pseudo-bounding interval arithmetic results. Liberty has been taken in placing the positive and negative error logarithmic ordinates "back-to-back" and

\[\text{It should be remarked that the SIMPLOT routines do not have the capability of producing "back-to-back" logarithmic plots. Additionally, the nonstandard axes, center line, axes tics and tic labels all had to be concurrently input to the SIMPLOT routines and plotted as separate "curves".}\]
in crossing this discontinuity with the straight line connections for the R*4 error data. Figures 5.7 and 5.8 respectively represent the 4th and 6th gain vector component results (2.95a and 2.95b) while Figure 5.9 represents the a posteriori error covariance matrix update term (2.103a and 2.103b).

The plots of the interval arithmetic results appear to give additional credence to the desirability of using the Stuva computational form of the Kalman filter; however, a cursory examination of the two computational techniques does not indicate if in fact "subdistributivity" considerations are involved. An overlaid comparison of the Washington State and pseudo-bounding interval arithmetic logarithmic plots for the Stuva computational scheme in the case of $Q_k$ and the 4th component of $K_k$ indicate that the additional conservativeness of the Washington State arithmetic arises within the first several iterations and then remains relatively "constant".\(^1\) In the case of the error plots for the 6th component of $K_k$, the differences appear to be growing linearly\(^2\) with the number of iterations. While it has not

\(^1\)Since these are logarithmic plots, the ratio of the errors for the two arithmetics remain constant.

\(^2\)In this case, the ratio of the errors for the two arithmetics is growing exponentially with the number of iterations.
been considered here, since the 4th and 6th components of the gain vector $K_x$ (2.95b) directly effect the estimates of the position states (34, p. 44), the marked difference in the behavior of the additional conservativeness error, although certainly numerical in nature, is suspected of resulting from the physical configuration of the problem and perhaps in this respect is is noteworthy.
Table 5.2. Miscellaneous comparison data for the two bounding interval arithmetic implementations of the two computational methods

<table>
<thead>
<tr>
<th>Bounding interval arithmetic implementation</th>
<th>Washington State</th>
<th>Pseudo</th>
</tr>
</thead>
<tbody>
<tr>
<td>Combined CPU time, a sec.</td>
<td>527.35</td>
<td>30.94</td>
</tr>
</tbody>
</table>

Shut off $Q_k$ "singular" values

<table>
<thead>
<tr>
<th></th>
<th>$Q_3$</th>
<th>$Q_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hartman</td>
<td>$[-0.4094000E04, 0.3795000E04]$</td>
<td>$[-0.4780000E03, 0.1629000E04]$</td>
</tr>
<tr>
<td></td>
<td>$Q_{29}$</td>
<td>$Q_{30}$</td>
</tr>
<tr>
<td>Stuva</td>
<td>$[-0.8245059E04, 0.9241965E04]$</td>
<td>$[0.4485630E03, 0.5676233E03]$</td>
</tr>
</tbody>
</table>

First $P_k$ diagonal element to become negative

<table>
<thead>
<tr>
<th></th>
<th>$P_2$</th>
<th>$P_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hartman</td>
<td>$[-0.1969140E04, 0.6357954E03]$</td>
<td>$[-0.3653840E03, 0.5107842E03]$</td>
</tr>
<tr>
<td></td>
<td>$P_{16}$</td>
<td>$P_{21}$</td>
</tr>
<tr>
<td>Stuva</td>
<td>$[-0.5465378E01, 0.6513354E02]$</td>
<td>$[-0.1639203E01, 0.5012177E02]$</td>
</tr>
</tbody>
</table>

aBased on the indicated algorithm shut offs.

bNot singular, but the last iteration result computed using the pseudo-bounding interval arithmetic.

cThe $P_{15.15}$ matrix element was always the first diagonal element of $P_k$ to become negative.
Figure 5.7. Errors for the bounding INTRVAL arithmetic and R*4 results: 4th component of $K_k$ (error = R*4 result - R*8 result)
a. Washington State bounding interval arithmetic

b. Pseudo-bounding interval arithmetic

Figure 5.8. Errors for the bounding INTRVAL arithmetic and R*4 results: 6th component of K_k (error = R*4 result - R*8 result)
a. Washington State bounding interval arithmetic

b. Pseudo-bounding interval arithmetic

Figure 5.9. Errors for the bounding INTERVAL arithmetic and R*4 results: scalar $Q_k$ (error = R*4 result - R*8 result)
CHAPTER VI. CONCLUSIONS

The initial objective of this investigation was to extend Moore's Kth-order interval integration method for initial-value problems of nonlinear systems of first-order differential equations with rational right-hand sides (2, pp. 100-118) to those that involved a perturbation parameter in an interval which was not necessarily small. The nonlinear interval integration algorithm which was developed to compute these interval bounding solutions or envelopes implemented the Kth-order method employing techniques for introducing arbitrary rational right-hand sides and for the machine generation of the required Taylor coefficients. Although the interval arithmetic employed at that time was a direct single-precision implementation of the analytic definitions without recourse to machine bounding of computational errors, the results were frustrating. In addition to the disappointment in the conservativeness of the results which were discussed in Chapter IV, the prognosis of formidable storage requirements and execution times further dampened enthusiasm in this direction.

Consequently, as is the case in many early investigations, the dissertation objectives were modified to consider the more tractable but nonetheless significant subclass of problems, initial-values problems of linear first-order
systems of autonomous homogeneous differential equations which involve a perturbation parameter.

Since the scheme of probable interval solution techniques in the linear systems pointed toward the computation of interval fundamental matrices, it became necessary to consider the infinite series representation of the interval matrix exponential functions which would not qualify as Moore's rational interval matrix functions that contain their corresponding united extensions (2, p. 18). Toward this end, the research presented in Chapter II has been successful in analytically constructing a complete metric space of the continuous matrix functions of an interval variable in which are embedded not only the interval matrix exponential functions which contain their corresponding united extensions but also the continuous matrix functions of a real variable. Similarly included in this space are all of the corresponding united extensions of the interval matrix functions of the space and the united extensions take on significance in a considerably less abstract topological structure than in Moore (2, p. 18). The research has additionally been successful in proving an analytic convergence in union of the perturbation parameter partition subinterval matrix exponential functions to the corresponding united extension of the matrix exponential function of the non-partitioned perturbation parameter interval, providing a
philosophy for construction of the linear interval integration algorithm.

Attention was then successfully directed toward devising an approximation method to obtain an interval matrix result which would contain the infinite series interval fundamental matrix for the perturbation parameter partition subintervals and which would elementwise satisfy relative interval endpoint error bounds. The approximation technique employed the optimal Householder matrix norm for Frobenius irreducible matrices, to obtain sharper bounds on the metric series representing a measure of the error created by truncating the infinite series interval representation and to provide a means for the additive interval augmenting of each element in the truncated series thereby satisfying the required containment property and the error criterion. The nested and centered form interval evaluation practices were also incorporated in the approximation technique to reduce the conservativeness of the results.

Since the correct numerical implementation of interval arithmetic demands the bounding of all possible truncation errors produced in the machine computations for the interval result endpoints, a bounding interval arithmetic package was obtained from Washington State University. Once the package was compiled and tested on the computer it became obvious that its use would result in expenditures beyond a
reasonable computer budget, necessitating construction of an alternative faster bounding interval arithmetic implementation. The operation of the Washington State University package and the development of the fast interval arithmetic routines were discussed in Chapter III where the use of the fast routines successfully indicated a comparative increase in execution speed of better than 17:1.

The linear interval integration algorithm was described in Chapter IV where the adoption of the analytic convergence philosophy for the interval fundamental matrix (with respect to partitioning of the perturbation parameter interval) defined a set of subproblems to the original differential equation. The algorithm employed the approximation technique for computing subproblem pseudo-fundamental interval matrices, obtained the subproblem interval bounding solutions by post-multiplication with the interval vector initial condition and obtained the problem interval bounding solution as the union over the subproblem interval solutions. The subproblem pseudo-fundamental interval matrices were computed for increasing values of time until it was no longer possible to satisfy the error criterion or until the subproblem solution storage limits were reached. Then the algorithm switched subproblem solution computation modes employing the final subproblem pseudo-fundamental interval matrices and a rolling subproblem solution storage technique.
While the convergence result of Chapter II provided a computational philosophy for the interval integration scheme in the case of the exact interval arithmetic, the implementation of machine bounding for the arithmetic, finite partitioning of the perturbation parameter interval, the approximation technique for the subproblem pseudo-fundamental interval matrices and the repeated use of the fundamental matrix multiplicative property rendered explicit error analysis for the algorithm results an impractical task. Nevertheless, the algorithm results presented in Chapter V for the three examples appeared reasonable. An unavoidable effect of the bounding interval arithmetic and the other algorithmic techniques mentioned above was the observed "opening" of the interval bounding solutions for increasing time. Chapter V also included the results of employing the bounding interval arithmetic strictly as an empirical instrument for observing comparative truncation effects for various computational schemes in two Kalman estimation problems. Several preliminary speculations were made by observing these results.

In conjunction with the theory and the numerical results presented here, there are many questions of interest which would appear fertile for future investigation.

Perhaps preliminarily, the necessary assembly language steps should be incorporated in the fast bounding interval
function ADD(SUB) to complete a design which duplicates the Washington State bounding interval arithmetic results. Then, once the IBM System/370 Model 158 computer presently being installed is fully operational, it would seem appropriate to use the extended-precision capability and devise fast double-precision bounding interval arithmetic COMPLEX*16 functions. From experience here it would appear that as an empirical investigating tool, the bounding, rounding and variable bounding or rounding capability of the double-precision functions would be useful, perhaps even in the finite word length effects in digital filtering methods.

With respect to the analytic work accomplished here, the proof that the sequence of rational nested centered form interval matrix functions (arising from the partial sums associated with the infinite series representation for the interval matrix exponential) is Cauchy should be completed. It would also appear advisable to investigate the possibility that the united extensions of every function in the metric function space exhibit the inclusion property associated with the rational functions (Proposition 2.9M).

In regard to the linear interval integration algorithm, further attempts should be made to investigate a measure of the conservativeness of the interval bounding solutions in the case of the exact interval arithmetic. It would also seem feasible to investigate the possibility of employing
large scale system decomposition techniques with regard to computation of the subproblem pseudo-fundamental matrices and in this sense possibly obtain a similar convergence result using the vector dimensioned norm concepts of (56, 57). Investigation of sparse matrix methods and the case of "stiff" linear differential equations (58, 59, 60) would also appear to be of value. In view of the wide interest in linear system sensitivity (50, 51), this would also appear to be an area for further study.

While the initial experience with the nonlinear differential equations was disappointing in the earlier framework, the experience gained in the linear systems would seem to renew an interest in investigating the possibility of employing a modified "rolling" subproblem solution storage approach to this problem. Since there appears to be a certain similarity of concepts, the "generalized differential equations" (61, 62) must at least be noted in passing.
LITERATURE CITED


49. Scranton, D. G. and Manchester, E. G. The Use of SIMPLOTTER, a High Level Plotting System, Document No. 4, Revision No. 1. (Computation Center, Iowa State University), 1972.


ACKNOWLEDGMENTS

The author wishes to express his appreciation to Dr. A. N. Michel for his constant enthusiastic encouragement and direction concerning the research reported in this dissertation.

Additionally, gratitude is extended to Dr. J. M. Yohe, Assistant Director of the Mathematics Research Center at the University of Wisconsin and Mrs. Kathleen Edwards, Computing Center Librarian at the University of Washington for cooperation in providing their interval arithmetic package. Special thanks are offered to Dr. R. G. Brown of this department for his support and interest in the utilization of the package and to Dr. J. A. Hoekstra of the University Computing Center for his considerable guidance in realizing the operation of the routines.

The author would be remiss if he did not express his appreciations to Dr. S. A. Nelson of the Mathematics Department for his noteworthy assistance and to Mr. J. R. Pavlat of this department for the valued convenience of his computer expertise. Special thanks are extended to remaining members of his committee, Dr. C. J. Triska and Dr. H. W. Hale of this department and Dr. R. H. Homer of the Mathematics Department, and to Mrs. Marlys Phipps for her patience and endurance in the typing of this manuscript.

Foremost, to his wife Ellen and his son Eric, whose
sacrifices have been understandably profound, the author expresses his heartfelt thanks.

The author also wishes to express his appreciation to the Engineering Research Institute at Iowa State University for the financial support provided through Project Themis under the Office of Naval Research Contract No. N00014-68-A-0162.
APPENDIX A. BOUNDING INTERVAL ARITHMETIC
FLOWCHARTS AND DOCUMENTATION
By G and H compiler convention, R15 contains the entry point address.

Note 1. General registers are denoted R0,...,R15 (32 bits each).
Note 2. Floating-point registers are denoted 0, 2, 4 and 6 (64 bits each).
Note 3. By G and H compiler linkage convention, COMPLEX function return values are placed in floating-point registers 0 and 2.
Note 4. Function arguments are A = [A(1),A(2)] and B = [B(1),B(2)].

Figure A-1. Abbreviated assembler flowchart for COMPLEX function ADD(A,B) with multiple entry point COMPLEX function SUB(A,B)
Figure A-1 (Continued)
branch if result \( \leq \) zero

Do not augment right endpoint since truncation does this if \( 2 \leq \text{zero} \).

Augment (bound) right endpoint, since \( 2 > \text{zero} \).

\begin{enumerate}
\item store short \( 2 \rightarrow \text{TEMP} \\
\item move sign/exponent byte, TEMP\rightarrow\text{AUGMNT} \\
\item add long, \\
\qquad 2 = 2 + \text{AUGMNT}
\end{enumerate}

Figure A-1 (Continued)
By G and H compiler convention, R15 contains the entry point address.

Note 1. General registers are denoted R0,...,R15 (32 bits each).
Note 2. Floating-point registers are denoted 0, 2, 4 and 6 (64 bits each).
Note 3. By G and H compiler linkage convention, COMPLEX function return values are placed in floating-point registers 0 and 2.
Note 4. Function arguments are $A = [A(1),A(2)]$ and $B = [B(1),B(2)]$.

Figure A-2. Abbreviated assembler flowchart for COMPLEX function $MUL(A,B)$ with multiple entry point
COMPLEX function $DIV(A,B)$
Branch if MUL entry, passing DIVFLT test with respect to EPSDVR.

DIVFLT test \([-\text{EPSDVR},+\text{EPSDVR}]nB\). If void, bypass diagnostics and termination section.

DIVFLT diagnostics and termination section.

MUL/DIV A-setup.

Branch if not MUL entry.

MUL A*B
0 = A(1)*B(1)
2 = A(1)*B(2)
4 = A(2)*B(1)
6 = A(2)*B(2)

Figure A-2 (Continued)
Figure A-2 (Continued)
Figure A-3. Abbreviated MINMAX assembler flowchart insert for COMPLEX function MUL(DIV)
Entry floating-point register contents:

\[
\begin{align*}
\text{MUL} & : & 0 &= A(1)B(1) & 0 &= A(1)/B(2) \\
& : & 2 &= A(1)B(2) & 2 &= A(1)/B(1) \\
& : & 4 &= A(2)B(2) & 4 &= A(2)/B(2) \\
& : & 6 &= A(2)B(2) & 6 &= A(2)/B(1)
\end{align*}
\]

Figure A-3 (Continued)
PSEUDO INTERVAL *COMPLEX* FUNCTION ADD(A,B), WITH
*COMPLEX* FUNCTION SUB(A,B) ENTRY

ADD
CSECT
STM 14,12,12(13)  *SAVE REGISTERS, ADD ENTRY
LR 12,15  *LOAD 15, ADDR OF ADD, INTO 12
USING ADD,12  *DECLARE IMPLIED BASE REGISTER
B BEGIN  *BRANCH AROUND SUB ENTRY SETUP
DROP 12  *DROP BASE REGISTER 12

SUB
STM 14,12,12(13)  *SAVE REGISTERS, SUB ENTRY
USING SUB,15  *DECLARE TEMPORARY BASE REGISTER
L 12,XADD  *FIND AND LOAD XADD WRT BASE REG 15
DROP 15  *DROP BASE REGISTER 15
USING ADD,12  *DECLARE IMPLIED BASE REGISTER

BEGIN
L 5,XCOM  *LOAD EXTNL ADDR OF NAMED-COMMON
LM 2,3,0(1)  *LOAD ADDR'S COMPLX A,B IN 2,3 RESP
LD 0,ZIP  *LOAD LONG, FLTG PT ZERO, ZIP
LDR 2,0  *
LDR 4,0  *
LDR 6,0  *
LE 0,0(2)  *LOAD SHORT, FLTG PT, A(1) IN 0
LE 2,4(2)  *A(2) IN 2
C 15,XADD  *COMPARE FOR ADD OR SUB ENTRY
BNE SUBTR  *IF NE, BR TO SUBTR SET UP
LE 4,0(3)  *LOAD SHORT, FLTG PT, B(1) IN 4
LE 6,4(3)  *B(2) IN 6
B READY  *ADD SETUP COMPLETE, BR TO READY

SUBTR
LE 4,4(3)  *LOAD SHORT, FLTG PT, B(2) IN 4
LCER 4,4  *-B(2) IN 4
LE 6,0(3)  *B(1) IN 6
LCER 6,6  *-B(1) IN 6

READY
ADR 0,4  *LONG ADDN, 0 + 4 IN 0, LWR BND
BNM FLAG1  *BRANCH IF LWR BND NOT MINUS
STE 0,TEMP  *STORE SHORT, NEG LWR BND RESULT
MVC 0(1,5),TEMP  *MOVE EXP BYTE OF RESULT INTO AUGMNT
AD 0,0(5)  *AUGMENT NEGATIVELY, NEG LWR BND
<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>FLAG1</strong></td>
<td><strong>ADR</strong> 2,6</td>
</tr>
<tr>
<td><strong>BNP</strong></td>
<td><strong>FLAG2</strong></td>
</tr>
<tr>
<td><strong>STE</strong></td>
<td><strong>TEMP</strong></td>
</tr>
<tr>
<td><strong>MVC</strong></td>
<td><strong>0(1,5),TEMP</strong></td>
</tr>
<tr>
<td><strong>AD</strong></td>
<td><strong>2,0(5)</strong></td>
</tr>
<tr>
<td><strong>FLAG2</strong></td>
<td><strong>LM</strong> 14,12,12(13)</td>
</tr>
<tr>
<td><strong>MVI</strong></td>
<td><strong>12(13),X'FF'</strong></td>
</tr>
<tr>
<td><strong>SR</strong></td>
<td><strong>15,15</strong></td>
</tr>
<tr>
<td><strong>BR</strong></td>
<td><strong>14</strong></td>
</tr>
<tr>
<td>ENTRY</td>
<td><strong>SUB</strong></td>
</tr>
<tr>
<td><strong>XADD</strong></td>
<td><strong>DC A(ADD)</strong></td>
</tr>
<tr>
<td><strong>XCOM</strong></td>
<td><strong>DC V(COMARG)</strong></td>
</tr>
<tr>
<td><strong>ZIP</strong></td>
<td><strong>DC D'O'</strong></td>
</tr>
<tr>
<td><strong>TEMP</strong></td>
<td><strong>DS E</strong></td>
</tr>
<tr>
<td><strong>END</strong></td>
<td></td>
</tr>
</tbody>
</table>

246
PSEUDO INTERVAL 'COMPLEX' FUNCTION MUL(A,B), WITH
'COMPLEX' FUNCTION DIV(A,B) ENTRY

MUL CSECT
STM 14,12,12(13) *SAVE REGISTERS, MUL ENTRY
LR 12,15 *LOAD 15, ADDR OF MUL, INTO 12
USING MUL,12 *DECLARE IMPLIED BASE REGISTER
B BEGIN *BRANCH AROUND DIV ENTRY SETUP
DROP 12 *DROP BASE REGISTER 12

DIV CSECT
STM 14,12,12(13) *SAVE REGISTERS, DIV ENTRY
USING DIV,15 *DECLARE TEMPORARY BASE REGISTER
L 12,XMUL *FIND AND LOAD XMUL WRT BASE REG 15
DRCP 15 *DROP BASE REGISTER 15
USING MUL,12 *DECLARE IMPLIED BASE REGISTER
BEGIN L 5,XCOM *LOAD EXTNL ADDR OF NAMED-COMMON
LM 2,3,(0) *LOAD ADDR'S COMPLX A,B IN 2,3 RESP
LD 0,ZIP *LOAD LONG, FLTG PT ZERO, ZIP
LDR 2,0 *
LDR 4,0 *
LDR 6,0 *
LE 0,0(3) *LOAD B(1) SHORT
STD 0,TEMP1 *STORE LONG B(1) IN TEMP1
LE 0,4(3) *LOAD B(2) SHORT
STD 0,TEMP2 *STORE LONG B(2) IN TEMP2
C 15,XMUL *COMPARE FOR MUL OR DIV ENTRY
BE PASS *IF EQ, BR TO MUL/DIV SETUP, PASS
LE 0,8(5) *LOAD SHORT, EPSDVR IN FLTG PT 0
CE 0,TEMP1 *EPSDVR LT B(1)?
BL PASS *IF YES, BR TO MUL/DIV SETUP, PASS
LNER 0,0 *LOAD MINUS EPSDVR
CE 0,TEMP2 *EPSDVR GT B(2)?
BH PASS *IF YES, BR TO MUL/DIV SETUP, PASS

DIVISOR EPSDVR FAULT, WRITE ERROR MSG
AND TERMINATE PGM EXEC VIA IBCOM#
LR 6,13  *LOAD MAIN PGM SAVE AREA ADDR IN 6
LA 13,SAVEDIV  *LOAD THIS PGM SAVEDIV ADDR IN 13
ST 13,8(0,6)  *STORE THIS PGM SAVEDIV ADDR IN MAIN SAVE
ST 6,4(0,13)  *STORE MAIN SAVE ADDR IN SAVEDIV
CNOP 0,4  *FORMATTED WRITE STATEMENT
L 15,=V(IBC0M#)  *SECOND LIST ITEM, FORMATTED
BAL 14,4(15)
DC XL4'00000006'
DC XL1'01',AL3(FMT)
L 15,=V(IBC0M#)*END I/O LIST
BAL 14,8(15)
DC XL4'08903000'
L 15,=V(IBC0M#)*STOP EXECUTION
BAL 14,16(15)
L 15,=V(IBC0M#)*END OF DIVISOR EPSDVR FAULT SECTION
BAL 14,52(15)
DC X'0540404040F0'

* END OF DIVISOR EPSDVR FAULT SECTION
*  
PASS
LE 0,0(2)  *LOAD SHORT, A(1) IN 0
LER 2,0  *  A(1) IN 2
LE 4,4(2)  *  A(2) IN 4
LER 6,4  *  A(2) IN 6
C 15,XMUL  *COMPARE FOR MUL OR DIV ENTRY
BNE DIVOP  *BR TO DIVISION OPERATIONS, DIVOP
MD 0,TEMP1  *LONG MULN, A(1)*B(1)
MD 2,TEMP2  *  A(1)*B(2)
MD 4,TEMP1  *  A(2)*B(1)
MD 6,TEMP2  *  A(2)*B(2)
B MINMAX  *BR TO MINMAX ROUTINE
DIVOP DD 0,TEMP2  *LONG DIVN, A(1)/B(2)
DD 2,TEMP1  *  A(1)/B(1)
DD 4,TEMP2  *  A(2)/B(2)
DD 6,TEMP1  *  A(2)/B(1)

* MINMAX SECTION, RESULT: MIN(0), MAX(2)
<table>
<thead>
<tr>
<th>ADDR</th>
<th>CDR</th>
<th></th>
<th>B</th>
<th>ADDR</th>
</tr>
</thead>
<tbody>
<tr>
<td>CDR</td>
<td>0.2</td>
<td></td>
<td>B</td>
<td>ADDR3</td>
</tr>
<tr>
<td>BH</td>
<td>ADDR3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CDR</td>
<td>0.4</td>
<td></td>
<td>B</td>
<td>ADDR1</td>
</tr>
<tr>
<td>BNL</td>
<td>ADDR1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADDR1</td>
<td>CDR</td>
<td>2.6</td>
<td>BH</td>
<td>ADDR9</td>
</tr>
<tr>
<td>CDR</td>
<td>2.6</td>
<td></td>
<td>B</td>
<td>ADDR9</td>
</tr>
<tr>
<td>ADDR2</td>
<td>LDR</td>
<td>2.6</td>
<td>B</td>
<td>ADDR9</td>
</tr>
<tr>
<td>LDR</td>
<td>2.6</td>
<td></td>
<td>B</td>
<td>ADDR9</td>
</tr>
<tr>
<td>ADDR3</td>
<td>CDR</td>
<td>4.6</td>
<td>BH</td>
<td>ADDR10</td>
</tr>
<tr>
<td>CDR</td>
<td>2.4</td>
<td></td>
<td>B</td>
<td>ADDR7</td>
</tr>
<tr>
<td>ADDR3</td>
<td>CDR</td>
<td>0.6</td>
<td>BH</td>
<td>ADDR4</td>
</tr>
<tr>
<td>CDR</td>
<td>0.6</td>
<td></td>
<td>B</td>
<td>ADDR5</td>
</tr>
<tr>
<td>ADDR4</td>
<td>LDR</td>
<td>6.0</td>
<td>B</td>
<td>ADDR12</td>
</tr>
<tr>
<td>LDR</td>
<td>6.0</td>
<td></td>
<td>B</td>
<td>ADDR12</td>
</tr>
<tr>
<td>ADDR5</td>
<td>LDR</td>
<td>0.2</td>
<td>B</td>
<td>ADDR12</td>
</tr>
<tr>
<td>LDR</td>
<td>0.2</td>
<td></td>
<td>B</td>
<td>ADDR12</td>
</tr>
<tr>
<td>ADDR6</td>
<td>LDR</td>
<td>2.6</td>
<td>B</td>
<td>ADDR12</td>
</tr>
<tr>
<td>ADDR7</td>
<td>CDR</td>
<td>0.6</td>
<td>B</td>
<td>ADDR8</td>
</tr>
<tr>
<td>CDR</td>
<td>0.6</td>
<td></td>
<td>B</td>
<td>ADDR2</td>
</tr>
<tr>
<td>ADDR8</td>
<td>LDR</td>
<td>2.0</td>
<td>B</td>
<td>ADDR12</td>
</tr>
<tr>
<td>LDR</td>
<td>2.0</td>
<td></td>
<td>B</td>
<td>ADDR12</td>
</tr>
<tr>
<td>ADDR9</td>
<td>LDR</td>
<td>0.4</td>
<td>B</td>
<td>ADDR12</td>
</tr>
<tr>
<td>LDR</td>
<td>0.4</td>
<td></td>
<td>B</td>
<td>ADDR12</td>
</tr>
<tr>
<td>ADDR10</td>
<td>CDR</td>
<td>2.6</td>
<td>B</td>
<td>ADDR12</td>
</tr>
<tr>
<td>CDR</td>
<td>2.6</td>
<td></td>
<td>B</td>
<td>ADDR12</td>
</tr>
<tr>
<td>ADDR11</td>
<td>LDR</td>
<td>2.0</td>
<td>B</td>
<td>ADDR12</td>
</tr>
<tr>
<td>LDR</td>
<td>2.0</td>
<td></td>
<td>B</td>
<td>ADDR12</td>
</tr>
<tr>
<td>*</td>
<td>LDR</td>
<td>0.6</td>
<td>B</td>
<td>ADDR12</td>
</tr>
<tr>
<td>LDR</td>
<td>0.6</td>
<td></td>
<td>B</td>
<td>ADDR12</td>
</tr>
</tbody>
</table>
* END OF MINMAX SECTION *

ADDR12 CD 0, ZIP *TEST LWR BND
BNL FLAG1 *BRANCH IF LWR BND NOT NEG
STE 0, TEMP *STORE SHORT, NEG LWR BND RESULT
MVC 0(1,5), TEMP *MOVE EXP BYTE OF RESULT INTO AUGMNT
AD 0,0(5) *AUGMENT NEGATIVELY, NEG LWR BND

FLAG1 CD 2, ZIP *TEST UPPER BND
BNH FLAG2 *BRANCH IF UPPER BND NOT POS
STE 2, TEMP *STORE SHORT, POS UPPER BND RESULT
MVC 0(1,5), TEMP *MOVE EXP BYTE OF RESULT INTO AUGMNT
AD 2,0(5) *AUGMENT POSITIVELY, POS UPPER BND

FLAG2 LM 14, 12, 12(13) *RESTORE REGISTERS
MVI 12(13), X'FF' *INDICATE RETURN
SR 15, 15 *CLEAR REGISTER 15 BEFORE RETURN
BR 14 *RETURN

ENTRY DIV *DECLARE ENTRY PT DIV EXTNL
XMUL DC A(MUL) *DETERMINE ADDR OF MUL
XCOM DC V(COMARG) *DETERMINE EXTNL NAMED-COMMON ADDR
ZIP DC D'0' *DEFINE LONG FLTG PT VIRTUAL ZERO
TEMP DS E *DEFINE WORK SPACE FOR EXP BYTE ACCESS
TEMP1 DS D *DEFINE WORK SPACE FOR 2ND ARG
TEMP2 DS D *DEFINE WORK SPACE FOR 2ND ARG
SAVEDIV DS 18F *SAVE AREA THIS SUBPROGRAM FUNCTION
FMT DC C'*** DIVFLT * PGM TERMINATED ***', //, T21, Z8, TX
     41, Z8, //
END =V(IBC0M#)
Figure B-1. Abbreviated flowchart for the algorithm operation indicating the required user subroutine calls A-E
Subroutine DRIVER
entry

"Opening" Sequence:
algorithm inputs:
NSTEPS
x(I,1), I=1,...,NN
TDELTA
initialize:
XSUB(I,NSUBI,1)=X(I,1),
I=1,...,NN,
NSUBI=1,...,NSUB.
(vector notation,
x(\cdot,1) and
XSUB(\cdot,NSUBI,1).
IROLL=1

NSTEPI=1,...,NSTEPS loop

compute:
\[ t_{\text{NSTEPI}} = \text{NSTEPI} \times \text{TDELTA} \]

Figure B-2. Abbreviated flowchart for subroutine DRIVER operation
$k_2 = 1, \ldots, M_2$

$M_2 \leq 9^a$

*Case 1:* The first time PHCOMP failure counter IFAIL=NFAIL, the $\hat{a}_k$'s computed that time are stored, control is set for a next PHCOMP entry bypass, but still IROLL=1. On the next PHCOMP entry, bypass is complete and IROLL=2 is initialized for "roll" startup here.

*Case 2:* If PHCOMP failure count IFAIL<NFAIL but this is the 9th entry to PHCOMP, bypass is complete and IROLL=2 is initialized for "roll" startup here (9 subproblem "rolling solution" storage limitation trap).

*Case 3:* If PHCOMP failure count IFAIL>NFAIL, the $\hat{a}_k$'s computed that time are not stored, the last PHCOMP entry values are used and IROLL=2 is initialized for "roll" startup here.

Figure B-2 (Continued)
"Pre-roll" subproblem solution storage indexing:
IX = NSTEPI + 1
IX0 = 1

"Rolling" subproblem solution storage indexing:
IX = MOD(NSTEPI, ILOOP) + 1
IX0 = MOD(NSTEPI + 1, ILOOP) + 1

NSUBI = 1, ..., NSTEPS loop

compute subproblem solutions:
XSUB(•, NSUBI, IX) = \sum_{\tau} (e_{NSUBI}^{\tau} x_{NSUBI})_{NSUBI, IX0} \cdot XSUB(•, NSUBI, IX0),
\tau = \begin{cases} t_{NSTEPI} & \text{if } IROLL = 1 \\ t_{ILOOP-1} & \text{if } IROLL = 2 \end{cases}

end NSUBI loop
compute algorithm solutions:
X(•, NSTEPI + 1) = \bigcup_{NSUBI} XSUB(•, NSUBI, IX)

end NSTEPI loop
RETURN

Figure B-2 (Continued)
Bypasses \texttt{BP1}(*):

1. If 9th subroutine entry, bypass entire subroutine and set for "roll" in DRIVER.
2. If previous entry resulted in IFAIL=NFAIL, bypass entire subroutine and set for "roll" in DRIVER.
3. Otherwise, if not first entry, bypass "opening" sequence here.

First subroutine entry "opening" sequence:

- **Algorithm inputs:**
  - KK,
  - NKNT1
  - NKNT2
  - NFAIL
  - NRUNUP

- **Compute:** \( R^8 \) factorials for \( J_{ij} \) computations.

- **Initialize:**
  - \( KK_{PAST}(I,J)=KK, I,J=1,\ldots,NN \)
  - IFAIL=0

- **Initialize:**
  - \( T=[t_{1,1}^{\text{NSTEP1}},t_{1,2}^{\text{NSTEP1}}] \)
  - KNT1=0
  - KNT2=0

Note: The "augmenting" process for \( \hat{K} \) in (2.90) and (4.26) is not shown (see the subroutine listing).

Figure B-3. Abbreviated flowchart for subroutine PHCOMP operation
FOR (I,J)-th element of NSUBI-th subproblem, compute nested interval values of $B(KLOOP+1),...,B(1)$ (2.75a and b); functions only of $T$, $KLOOP$ and problem fixed (subroutine GCCMP) $G_1,...,G_{230}$ (2.68), not functions of NSUBI.

Figure B-3 (Continued)
compute $R^8$, store $R^4$
for NSUBI-th subproblem

\[ (2.88) \]

function of KLOOP and
NSUBI-th (subroutine
NORM) subproblem results.

For $(I,J)$-th element of
NSUBI-th subproblem,
compute nested interval
values of

\[ E(KLOOP+1), \ldots, E(I) \]

(2.78a and b)
functions of $\alpha_{NSUBI_C}$,
$\beta(KLOOP+1), \ldots, \beta(I)$ and
fixed (subroutine PRCOMP)
interval binomial coefficients.

For $(I,J)$-th element of NSUBI-th
subproblem, compute nested
interval value of

\[ \xi_{KLOOP}(\alpha_{NSUBI_C} + \eta_{NSUBI_T}) \]

(2.80)
function of $E(KLOOP+1), \ldots, E(I)$
and $\eta_{NSUBI}$. 

Figure B-3 (Continued)
Test error criterion (2.89 and 4.25) function of NSUBI-th subproblem results from subroutine NORM, and above value of $\hat{a}_{KLOOP}$.

Fail/Pass mechanism (abbreviated—see subroutine listing for details).

Fail

Fail

Single-fault

Pass

double-fault

No

Yes

$KLOOP<20$

$KLOOP<20$

Yes

$KLOOP=KLOOP+1$

$KLOOP=KLOOP+1$

$KNT1=KNT1+1$

Yes

No

"Augment" computation (see subroutine listing for details)

(I,J)-th element of NSUBI-th subproblem to temporary storage $\hat{a}_{KLOOP}=\hat{a}_{KLOOP} + Z_{KLOOP}$ (2.90, 4.26 and 4.27).

"runup backloop"

$\hat{a}_{KLOOP} = KLOOP+1$ is called a "runup" and is counted by IRUNUP for each value of I,J and NSUBI. Once IRUNUP>NRUNUP (algorithm input), $KLOOP$ is not "runup" and the fault is accepted with appropriate fault counting in counters KNT1 or KNT2.

Figure B-3 (Continued)
end NSUBI loop

end I,J loop

KKPAST(I,J)=KLOOP
(i.e., store final value
KLOOP for starting value
next entry this subroutine,
for same I,J value).

Test total error counts
for subroutine failure
of KLOOP's:
KNT1>NKNT1 or
KNT2>NKNT2

Yes (PHCOMP fail)

No (PHCOMP pass)

Bypasses BP3(*)
(1) If first entry subroutine,
"BOMB" program.
(2) Otherwise, count
IFAIL=IFAIL+1 and if
IFAIL<NFAIL, accept
faults storing all new
KLOOP elements.
(3) If IFAIL>NFAIL, keep
previous subroutine entry
KLOOP values in storage
and set for "roll" opera­
tion of subroutine
DRIVER.

Store all temporary
I,J,NSUBI interval
matrix elements KLOOP.

EXIT

RETURN

Figure B-3 (Continued)
MAIN PROGRAM

NUMERICAL INTERVAL ANALYSIS PROGRAM TO SOLVE THE INITIAL-VALUE
PROBLEM FOR A LINEAR CONSTANT COEFFICIENT HOMOGENEOUS FIRST-ORDER
VECTOR ORDINARY DIFFERENTIAL EQUATION SYSTEM IN WHICH THE
COEFFICIENT MATRIX CONTAINS ELEMENTS THAT LINEARLY DEPEND ON
A PARAMETER (IE, THE COEFFICIENT MATRIX MAY BE WRITTEN AS THE
MATRIX SUM

A + THETA * B

WHERE A AND B ARE CONSTANT REAL MATRICES AND THETA IS A REAL
PARAMETER WHICH MAY ASSUME A FIXED VALUE ON THE CLOSED INTERVAL
<-1,+1>.

THE INTERVAL SOLUTION TECHNIQUE IS TO SUBDIVIDE THE <-1,+1>
PARAMETER INTERVAL INTO NSUB EQUAL WIDTH SUBINTERVALS AND FOR
SOLUTION TIME(2)=1*THETA, TO COMPUTE FOR EACH SUBINTERVAL A
NESTED CENTERED FORM INTERVAL MATRIX RESULT IN THE BOUNDING
INTERVAL ARITHMETIC WHICH SET-THEORETICALLY INCLUDES THE SET
OF FUNDAMENTAL MATRICES FOR EACH VALUE OF THE PARAMETER THETA
IN THAT SUBINTERVAL. SUBPROBLEM INTERVAL VECTOR SOLUTION
VALUES FOR TIME(I) ARE THEN COMPUTED AS THE INTERVAL PRODUCT
OF THESE PSEUDO FUNDAMENTAL MATRICES AND THE INITIAL-VALUE
VECTORS. THE UNION OVER THESE SUBPROBLEM SOLUTIONS YIELDS
THE ALGORITHM SOLUTION VECTOR. FOR EACH INCREMENTED SOLUTION
TIME(I)=(I-1)*THETA, NEW SUBPROBLEM PSEUDO FUNDAMENTAL
MATRICES ARE COMPUTED UNTIL THEY FAIL AN ERROR CRITERION.
ONCE THIS OCCURS (EG, AT TIME(L+1)), THE SUBPROBLEM SOLUTIONS
AT TIME(L+J), J=1,2,..., ARE COMPUTED BY EMPLOYING THE PSEUDO
FUNDAMENTAL MATRICES COMPUTED AT TIME(L) AND RESPECTIVELY
SETTING THE SUBPROBLEM INITIAL-VALUES TO THEIR INTERVAL
SOLUTION VALUES AT TIME(J+1).

IN ORDER TO PROVIDE THE GREATEST USER VERSATILITY, THE INTERVAL
ANALYSIS PROGRAM HAS BEEN DESIGNED SO THAT THE OPERATOR WRITES
THE CONTROLLING FORTRAN MAIN PROGRAM INCORPORATING THE SUBORDINATE
FEATURES WHICH HE DESIRES (SUCH AS MULTIPLE PROBLEM RUN LOOPING,
PUNCHED OUTPUT, ETC.) AND A SPECIFIED SEQUENCE OF SUBROUTINE
CALLS.

THE PROGRAM SUBROUTINES AND COMPLEX BOUNDING INTERVAL ARITHMETIC
FUNCTIONS ARE IN OBJECT FORM (THE SUBROUTINES HAVING BEEN COMPILED
PREVIOUSLY WITH THE LARGE STORAGE REQUIREMENT OPTIMIZING H
COMPILER AND THE FUNCTIONS HAVING BEEN PREVIOUSLY ASSEMBLED) AND
THE LINK-EDIT JOB CONTROL STEP PROVIDES THE NECESSARY CONNECTION.
(SYSLIB LINK-EDIT JOB CONTROL IS ALSO NECESSARY TO CONNECT UP TO
THE EISPACK ROUTINES - SEE SUBROUTINE NORM; SIMILARLY, SIMPLOTTER
JOB CONTROL IS ADDITIONALLY REQUIRED - SEE SUBROUTINE CURVES.)

IN ORDER TO FACILITATE WRITING THE MAIN PROGRAM, THE FOLLOWING
EXAMPLE AND SUBSEQUENT EXPLANATIONS ILLUSTRATE THE PROGRAM
WRITTEN TO PRODUCE THE MINIMUM PLANT SENSITIVITY OPTIMAL REGULATOR
DESIGN PROBLEM (OPTIMAL BETA) RESULTS OF CHAPTER FIVE.
**EXAMPLE OF MAIN PROGRAM**

```fortran
* COMPLEX X(5,100)
* REAL TIME(100)
* COMM/CMN/PARAM3/ INORM,IGCMP,IPRCMP,IDRVR,IPHCM
* COMM/CMN/SOLN/ TIME,X,NPTS,ISTOP
* ISTOP=1500
* INCRM=1
* IGCMP=1
* IPRCMP=1
* IDRVR=1
* IPHCM=1
* CALL NORM
* CALL GCCMP
* CALL PRCMP
* CALL CRIVER
* DO 100 I=1,NPTS
* 100 WRITE(7,11) TIME(I), (X(J,1), J=1,5)
* 11 FORMAT(20A4)
* CALL CURVES
* STOP
* END
```

**EXPLANATIONS OF MAIN PROGRAM EXAMPLE**

NPTS = THE NUMBER OF SOLUTION POINTS ACTUALLY OBTAINED (INCLUDING THE INITIAL-VALUES - SEE SUBROUTINE DRIVER)

X = THE COMPLEX ARRAY CONTAINING THE INTERVAL SOLUTION VALUES

TIME = THE REAL ARRAY CONTAINING THE CORRESPONDING SOLUTION TIMES

THE PARAMETERS INORM,IGCMP,IPRCMP,IDRVR,IPHCM, ARE BUILT IN SUBROUTINE 'ADDITIONAL OUTPUT' CONTROL PARAMETERS. SETTING A PARAMETER TO 1 SHUTS OFF ADDITIONAL OUTPUT, WHILE A 2 PROVIDES THE EXTRA OUTPUT (SEE CORRESPONDING SUBROUTINES).
ISTOP IS AN INTEGER WHICH IS USED BY SUBROUTINE DRIVER TO COMPARE
WITH THE RESULTS OF ITS CALL CLOCK(IBAL) AT THE END OF EACH
SOLUTION ITERATION. IBAL INDICATES THE BALANCE OF THE CPU TIME
FOR THE JOBSTEP IN 100THS OF SECONDS AND IF (ISTOP .GE. IBAL) IS TRUE
SUBROUTINE DRIVER TRIGGERS AN EARLY RETURN TO THE MAIN PROGRAM
WITHOUT COMPLETING NSTEPS OF SOLUTION INTERATIONS. THUS FOR THE
EXAMPLE HERE WITH ISTOP=1500 (15 SECONDS), IT IS POSSIBLE TO
INSURE THAT THE SOLUTION DATA IS AT LEAST PUNCHED AND THE
GRAPHICAL OUTPUT IS ATTEMPTED. THE DO 100 LOOP PUNCHES THE
SOLUTION RESULTS IN A-FORMAT (MOST COMPACT PUNCHED OUTPUT - ONE
CARD COLUMN PER BYTE) PRIOR TO THE GRAPHICAL OUTPUT ATTEMPT.

THE SEQUENCE OF SUBROUTINE CALLS NECESSARY FOR THE PROBLEM IS CALL
NORM, GCOMP, PRCOMP, DRIVER AND CURVES. PHCOMP IS MULTIPLY
CALLED FROM DRIVER.

INPUT DEFINITIONS:

NN = DIMENSION OF THE COEFFICIENT MATRIX (MAX=5)
AUGMNT = R*8 HEX-INITIALIZED PARAMETER FOR THE BOUNDING
INTERVAL ARITHMETIC COMPLEX FUNCTIONS ADD, SUB,
MUL AND DIV
EPSDVR = R*4 HEX-INITIALIZED PARAMETER FOR THE BOUNDING
INTERVAL ARITHMETIC COMPLEX FUNCTION DIV
HSNGL(2,5,5,2) = R*4 ARRAY FOR INPUTTING THE PROBLEM INTERVAL
COEFFICIENT MATRIX
MODE = INPUT DECISION FOR WHICH HOUSEHOLDER MATRIX
NORM IS TO BE USED (1=NORMAL; 2=TRANSPOSE)
NSUB = NUMBER OF <-1,+1> PARAMETER INTERVAL SUBDIVISIONS
TO BE USED (MAX=25)
P = SUBROUTINE PHCOMP ERROR CRITERIA INTEGER FOR
COMPUTATION OF EPSMUL=1.0/16.0**P
NSTEPS = NUMBER OF STEPS IN TDELTA FOR WHICH SOLUTION
IS TO BE RUN (MAX=99)
$XSNGL(2,5) = R*4$ ARRAY FOR INPUTTING THE PROBLEM INTERVAL

INITIAL-VALUES

$TDELTA = $ ALGORITHM TIME INCREMENT

$KK = $ INITIAL ALGORITHM SETTING FOR NUMBER OF MATRIX TERMS TO BE USED IN COMPUTING THE SUBPROBLEM PSEUDO FUNDAMENTAL MATRICES (CALLED PHI’S; MAX=20)

$NKNT1 =$ MAX TOTAL NUMBER OF SINGLE ENDPOINT FAILURES ACCEPTABLE IN SUBPROBLEM PHI’S FOR EACH SOLUTION TIME

$NKNT2 =$ MAX TOTAL NUMBER OF DOUBLE ENDPOINT FAILURES ACCEPTABLE IN SUBPROBLEM PHI’S FOR EACH SOLUTION TIME

$NFAIL =$ MAX TOTAL NUMBER OF SOLUTION TIME PHI FAILURES ACCEPTED BEFORE LAST SUBPROBLEM PHI’S ONLY WILL BE USED FOR BALANCE OF ALGORITHM

$NRUNUP =$ MAX TOTAL NUMBER OF RUNUPS IN KK ACCEPTABLE BEFORE AN ENDPOINT FAILURE WILL BE ALLOWED FOR A SUBPROBLEM PHI ELEMENT

$NPRI =$ NUMBER OF PRIMARY CALLS TO GRAPH

(A) $XLAB = $ HORIZONTAL AXIS LABEL

(A) $YLAB = $ VERTICAL AXIS LABEL

(A) $GLAB = $ GRAPH LABEL

(A) $XSIZE = (SEE SIMPLOT MANUAL, P17)$

(A) $YSIZE = ($ DITTO , P17)

(A) $XSF =$ ( DITTO , P20)

(A) $XMIN =$ ( DITTO , P20)

(A) $NSEC =$ NUMBER OF SECONDARY CALLS TO GRAPHS

(A) $NLTRS =$ NUMBER OF CALLS TO LETTRS

(A) $YSF = (SEE SIMPLOT MANUAL, P20)$

(A) $YMIN =$ ( DITTO , P20)

(B) $ISYM =$ ( DITTO , P51)

(B) $MODE =$ ( DITTO , P28 + P16)

(B) $IBND =$ DATA SET FOR PLOTTING (1=LEFT ENDPOINT; 2=RIGHT ENDPOINT)

(B) $ISTATE =$ DATA SET INTERVAL STATE VARIABLE FOR PLOTTING
C (B)   DATLAB = (SEE SIMPLOT MANUAL, P27)
C (C)   XO = ( DITTO , P42)
C (C)   YO = ( DITTO , P42)
C (C)   HEIGHT = ( DITTO , P42)
C (C)   THETA = ( DITTO , P43)
C (C)   NCHAR = ( DITTO , P43)
C (C)   STRING = ( DITTO , P42)

NOTES:
C (A) ONCE EACH PRIMARY CALL
C (B) ONCE FOR THE PRIMARY CALL AND ONCE FOR EACH SECONDARY CALL
C (C) ONCE EACH CALL TO LETTRS

ORDER OF DATA INPUT CARDS:

* * DATA * *

(SUBROUTINE NORM)
C 1 NN
C 2 AUGMNT, EPSDVR
C 3 (MC) (((HSNGL(I,J,K),K=1,2),J=1,NN),I=1,NN) (8F10.6)
C 4 MODE
C 5 NSUB

(SUBROUTINE PRCOMP)
C 6 P

(SUBROUTINE DRIVER)
C 7 NSTEPS
C 8 (MC) (((XSNGL(I,J),I=1,2),J=1,NN) (8F10.6)
C 9 TDELTA

(SUBROUTINE PHCOMP)
C 10 KK
C 11 NKNT1,NKNT2,NFAIL,NRUNUP

* * FORMAT * *
<table>
<thead>
<tr>
<th>(SUBROUTINE CURVES)</th>
</tr>
</thead>
<tbody>
<tr>
<td>12 NPRI ( (4I10,5A4) )</td>
</tr>
<tr>
<td>13(A) XLAB, YLAB, GLAB ( (20A4) )</td>
</tr>
<tr>
<td>14(A) XSIZE, YSIZE, XSF, XMIN, NSEC, NLTRS, YSF, YMIN ( (4F10.6, 2I10, 2F10.6) )</td>
</tr>
<tr>
<td>15(B) ISYM, MODE, IBND, ISTATE, DATLAB ( (4I10, 5A4) )</td>
</tr>
<tr>
<td>16(C) X0, Y0, HEIGHT, THETA, NCHAR ( (4F10.6, 2I10, 2F10.6) )</td>
</tr>
<tr>
<td>17(C) STRING ( (20A4) )</td>
</tr>
</tbody>
</table>

**NOTES:**
- (MC) MULTIPLE CARD SEQUENCE
- (A) ONCE EACH PRIMARY CALL
- (B) ONCE FOR THE PRIMARY CALL AND ONCE FOR EACH SECONDARY CALL
- (C) ONCE EACH CALL TO LETTRS

**REMARKS:**
- IF NPRI = 2, 1ST NSEC = 3, 1ST NLTRS = 4, 2ND NSEC = 2, 2ND NLTRS = 3,
- THE SEQUENCE OF INPUT CARDS TO BE READ AFTER 12 WOULD BE
- (1ST PRI) 13-14/15-15-15-15/16-17-16-17-16-17-16-17, THEN
SUBROUTINE NORM

THE PURPOSE OF THE SUBROUTINE IS TO:

1. INPUT THE INTERVAL COEFFICIENT MATRIX
   (TO ACCOMMODATE A SIGNED DEPENDENCE ON THE PARAMETER THETA,
   THE INTERVAL COEFFICIENT MATRIX ELEMENT ENDPOINTS MAY ASSUME
   AN INVERTED ORDER)

2. COMPUTE THE MATRICES A AND B IN THE PARAMETERIZED
   REPRESENTATION OF THE INTERVAL COEFFICIENT MATRIX,
   \( A + \theta \cdot B \),
   WHERE DENOTE
   \( G_1(I, J, 1) = G_1 = \text{A} \),
   \( G_1(I, J, 2) = G_1 = \text{B} \),
   AND
   \( \theta = < -1, +1 > \).

3. SUBDIVIDE THE \( < -1, +1 > \) INTERVAL INTO NSUB EQUAL SUBINTERVALS,
   \( \theta(I), I = 1, \ldots, \text{NSUB} \).

4. FOR EACH \( I = 1, \ldots, \text{NSUB} \), AND FOR THE INPUT VALUE OF MODE:
   (A) COMPUTE R*8 DUPLICATE SUBPROBLEM MATRICES A AND AA
       CONSISTING OF THE ELEMENTS OF
       \( |G_1 + \theta \cdot G_2| \),
       WHERE \( | | \) INDICATES THE INTERVAL MAGNITUDE OPERATION
   (B) DETERMINE THE R*8 EIGENVALUES \( \{W_R(L) + j \cdot W_I(L), L = 1, \ldots, \text{NN} \} \),
       AND EIGENVECTORS (IN MATRIX ARRAY FORM
       \( Z_P(M, L), M = 1, \ldots, \text{NN}; L = 1, \ldots, \text{NN} \) OF MATRIX AA, FROM THE
       EISPAC ROUTINES (ON SYSLIB)
   (C) STORE THE MAXIMAL EIGENVALUE IN R*4 ARRAY ELEMENT
       \( EIVAL(I) \) (WILL BE REAL AND POSITIVE)
   (D) STORE THE CORRESPONDING EIGENVECTOR (WILL BE POSITIVE
       AND REAL) IN R*4 ARRAY ELEMENTS \( EIVEC(I, J), J = 1, \ldots, \text{NN} \).
   (E) COMPUTE THE R*8 HOUSEHOLDER MATRIX NORM FOR A AND
       STORE IN THE R*4 ARRAY ELEMENT \( ANORM(I) \)
SUBROUTINE NORM READ SEQUENCE
INPUT INTERVAL COEFFICIENT MATRIX IS STORED IN DUMMY COMPLEX ARRAY
H (BY EQUIVALENCING TO HSNGL ARRAY) WITH 'LEFT' ENDPOINTS IN
H(I,J,1) AND 'RIGHT' ENDPOINTS IN H(I,J,2)
READ(5,1) NN
1 FORMAT(I10)
READ(5,2) AUGMNT,EPSDVR
2 FORMAT(Z16,Z8)
READ(5,3) ((HSNGL(1,I,J,K),K=1,2),J=1,NN),I=1,NN
3 FORMAT(8F10.6)
READ(5,4) MODE
4 FORMAT(I10)
READ(5,4) NSUB
WRITE SUBROUTINE NORM INPUT SEQUENCE VALUES FOR USER VERIFICATION
WRITE(6,13) NN,NSUB,MODE,AUGMNT,EPSDVR
13 FORMAT('PROBLEM INPUT PARAMETERS READ BY SUBROUTINE NORM:',' /ONN=',4X,I5,' NSUB=',2X,I5,' MODE=',2X,I5, '/ INTERVAL FUNCTION HEXADECIMAL PARAMETERS --',' / AUGMNT=',2X,Z16,' EPSDVR=',10X,Z8, '/ DIFFERENTIAL EQUATION INTERVAL PLANT MATRIX --)
    DO 100 I=1,NN
100 WRITE(6,14) ((HSNGL(1,I,J,K),K=1,2),J=1,NN)
C
C COMPLETE DEGENERATE INTERVAL INITIALIZATION OF DUMMY ARRAY H
C AND COMPUTE INTERVAL MATRICES G1 AND G2
C
DO 110 I=1,NN
DO 110 J=1,NN
    HSNGL(2,I,J,1)=HSNGL(1,I,J,1)
    HSNGL(2,I,J,2)=HSNGL(1,I,J,2)
    G(I,J,1)=DIV(ADD(H(I,J,2),H(I,J,1)),TWO)
110 G(I,J,2)=DIV(SUB(H(I,J,2),H(I,J,1)),TWO)
C
C USER MAIN PROGRAM CONTROLLED BYPASS AND ADDITIONAL OUTPUT
C
GO TO (118,112),INORM
112 WRITE(6,990)
990 FORMAT('BOUNDING ARITHMETIC COMPUTED MATRICES A + THETA * B A
*RE:')
    DO 991 K=1,2
    WRITE(6,992)
    DO 991 I=1,NN
991 WRITE(6,993) (G(I,J,K),J=1,NN)
C
C COMPUTE PARAMETER INTERVAL <-1,+1> SUBDIVISION INTO EQUAL WIDTH
C SUBINTERVALS THETA(I), I=1,...,NSUB
C
118 THETA(1)=-1.0
NSUB1=NSUB-1
NSUB2 = 2 * NSUB
THETA(NSUB2) = +1.0
ADVR = NSUB
DO 120 NSUBI = 1, NSUB1
    ID = 2 * NSUBI
    ANUM = -NSUB + ID
    THETA(ID) = ANUM / ADVR
120    THETA(ID + 1) = THETA(ID)

BEGINNING OF DO LOOP FOR STEPS 4-A, B, C, D AND E
NSUBI = 1, ..., NSUB
DO 290 NSUBI = 1, NSUB

COMPUTE TEMPORARY R*8 DUPLICATE SUBPROBLEM MATRICES A AND AA

= \left| G1 + \text{THETA} \cdot G2 \right|, \text{ according to mode}

DO 150 I = 1, NN
    DO 150 J = 1, NN
        DUM = ADD(G(I, J, 1), MUL(THETA(NSUBI), G(I, J, 2)))
        TEMP1 = AMAX1(ABS(ADUM1), ABS(ADUM2))
        GO TO (130, 140), MODE
130    A(I, J) = TEMP1
    AA(I, J) = TEMP1
    GO TO 150
140    A(J, I) = TEMP1
    AA(J, I) = TEMP1
150    CONTINUE

COMPUTE EIGENVALUES AND EIGENVECTORS OF AA (DESTROYED IN PROCESS)
WITH EISPAC ROUTINES, USING SPECIFIC (FAST OPERATION) CALL LIST

CALL ORTHES(5, NN, 1, NN, AA, WI)
CALL ORTRAN(5, NN, 1, NN, AA, WI, ZP)
CALL HQR2(5, NN, 1, NN, AA, WR, WI, ZP, IERR)

USER MAIN PROGRAM CONTROLLED BYPASS AND ADDITIONAL OUTPUT
GO TO (158,152),INORM
152 WRITE(6,994) NSUBI
994 FORMAT('OFOR THETA SUBDIVISION NSUBI=',2X,I5,', ABMATRIX, EIGE
*NVALUES AND EIGENVECTORS ARE:')
DO 995 I=1,NN
995 WRITE(6,996) (A(I,J),J=1,NN)
996 FORMAT(' ',5(D14.7,5X))
DO 997 I=1,NN
997 WRITE(6,996) WR(I),WI(I)
998 WRITE(6,996) (ZP(I,J),J=1,NN)
C
C DETERMINE MAXIMAL R*8 EIGENVALUE AND STORE IN R*4 ARRAY
C ELEMENT EIVAL(NSUBI)
C
158 INDEX=1
DO 160 I=2,NN
IF(WR(INDEX).GE.WR(I)) GO TO 160
INDEX=I
160 CONTINUE
EIVAL(NSUBI)=WR(INDEX)
C
C USER MAIN PROGRAM CONTROLLED BYPASS AND ADDITIONAL OUTPUT
C
GO TO (164,162),INORM
162 WRITE(6,999) INDEX,WR(INDEX),WI(INDEX)
999 FORMAT(' HSE-SELECTED INDEX=',2X,I5,', HSE-EIVAL=',/ ',2(D14.
*7,5X))
C
C DETERMINE R*8 EIGENVECTOR CORRESPONDING TO MAXIMAL EIGENVALUE,
C INSPECT FOR ALL POSITIVE COMPONENTS (EISPAC MAY RETURN ALL
C NEGATIVE ELEMENTS) AND BRANCH TO ERROR DIAGNOSTICS AND PROGRAM
C TERMINATION SECTION IF MIXED SIGN OCCURS
C
164 IFLT=1
165 CONTINUE
DO 190 I=1,NN
   IF(ZP(I,INDEX) .GE.0.0D+00) GO TO 190
   GO TO (170,200), IFLT
170 IFLT=2
   DO 180 J=1,NN
180 ZP(J,INDEX)=-ZP(J,INDEX)
   GO TO 165
190 CONTINUE
   GO TO 210

C MIXED SIGN EIGENVECTOR ERROR MSG AND PROGRAM TERMINATION

200 WRITE(6,11) NSUBI,(ZP(I,INDEX),I=1,NN)
   11 FORMAT('NSUBI=',2X,15,2X,'HSE-EIVEC MIXED SIGN - STOP RUN',
      *(/' ',D14.7))
   CALL EXIT

C INSPECT MAXIMAL EIGENVECTOR TO INSURE MIN COMPONENT NOT LESS THAN
C MAX COMPONENT * 1/16**5. BRANCH TO ERROR DIAGNOSTICS AND PROGRAM
C TERMINATION SECTION IF INSPECTION FAILS

210 TEMPI=0.0D+00
   DO 220 I=1,NN
220 TEMPI=DMAX1(TEMPI,ZP(I,INDEX))
   TEMPI=TEMPI/(16.0D+00)**5
   DO 230 I=1,NN
      IF(ZP(I,INDEX) .LT. TEMPI) GO TO 240
230 CONTINUE
   GO TO 250

C EIGENVECTOR RELATIVE COMPONENT SIZE FAILURE ERROR MSG AND
C PROGRAM TERMINATION

240 WRITE(6,12) NSUBI,(ZP(I,INDEX),I=1,NN)
   12 FORMAT('NSUBI=',2X,15,2X,'HSE-EIVEC TOO SMALL - STOP RUN',
      *(/' ',D14.7))
   CALL EXIT
STCRE R*8 MAXIMAL EIGENVECTOR COMPONENTS IN R*4 ARRAY
EIVEC(NSUBI), I=1,...,NN.

DO 260 I=1,NN
260 EIVEC(NSUBI)=ZP(I,INDEX)

USER MAIN PROGRAM CONTROLLED BYPASS AND ADDITIONAL OUTPUT
GO TO (268,262),INORM
262 WRITE(6,1000) INDEX,(ZP(I,INDEX),I=1,NN)
1000 FORMAT(' HSE-SELECTED INDEX=',2X,15,'HSE-EIVEC=',/(' ','D14.7*)

COMPUTE R*8 HOUSEHOLDER MATRIX NORM FOR A AND MODE AND STORE
RESULT IN R*4 ARRAY ANORM(NSUBI)

TEMP1=0.00+00
DO 280 I=1,NN
TEMP2=0.00+00
DO 270 J=1,NN
270 TEMP2=TEMP2+ZP(J,INDEX)*A(I,J)
TEMP2=TEMP2/ZP(I,INDEX)
280 TEMP1=DMAX1(TEMP1,TEMP2)
ANORM(NSUBI)=TEMP1

USER MAIN PROGRAM CONTROLLED BYPASS AND ADDITIONAL OUTPUT
GO TO (290,282),INORM
282 WRITE(6,1001) TEMP1
1001 FORMAT(' HSE-COMPUTED ANORM=',/(' ','D14.7)
290 CONTINUE

END OF DO LOOP FOR STEP 4
RETURN
END
SUBROUTINE GCOMP

THE PURPOSE OF THE SUBROUTINE IS TO COMPUTE THE ENTIRE PRELIMINARY SEQUENCE OF INTERVAL MATRICES G(1),...,G(230), WHERE FOR CONVENIENCE IN NOTATION THE MATRICES HAVE BEEN DENOTED BY G(#) = G(L,M, #) AND (L,M) ARE THE USUAL MATRIX ELEMENT SUBSCRIPTS (NOTE THAT G(1) AND G(2) HAVE BEEN PREVIOUSLY COMPUTED BY SUBROUTINE NORM).

THE SEQUENCE OF MATRICES WILL THEREFORE BE COMPUTED FOR I=2,...,20 AND J=0,...,I BY THE DEFINITION

\[ G(I*(I+1)/2+J) = (1/I) * \sum_{K} G((I-I) * I/2 + J - (K-1)) * G(K) \]

WHERE THE K-SUMMATION INDEX K RUNS FROM K=(I IF J=0,...,I-1; 2 IF J=I) TO K=(1 IF J=0; 2 IF J=1,...,I)

THIS SEQUENCE OF MATRICES WILL BE REQUIRED FOR THE CENTERED FORM COMPUTATIONS OF SUBROUTINE PHCOMP.

COMPLEX ADD, SUB, MUL, DIV, G(5, 5, 230), GG(5, 5, 2), FACTL
REAL*8 AUGMNT, GDUM(5750), GGDUM(50)
REAL FACT(2)
EQUIVALENCE (FACTL, FACT(1)), (GDUM(1), G(1, 1, 1)), *(GGDUM(1), GG(1, 1, 1))
COMMON/COMARG/, AUGMNT, EPSDVR
COMMON/PARM1/KK, NN, MODE, NSUB
COMMON/PARM3/ INORM, IGCMP, IPRCMP, IDVR, IPHCMP
STORE INTERVAL MATRICES G(I), I=1,2 FROM SUBROUTINE NORM STORAGE

DO 100 I=1,50
   100 GDUM(I)=GDUM(I)

ZERO INTERVAL MATRICES G(I), I=3,...,230

DO 110 I=51,5750
   110 GDUM(I)=0.0D+00

COMPUTE INTERVAL MATRICES G(I), I=3,...,230

L3=1
L4=2
DO 130 K=2,20
   AK=K
   FACT(1)=AK
   FACT(2)=AK
   L1=L3
   L2=L4
   L3=L4+1
   L4=L3+K
   K1=K-1
   DO 130 I=1,NN
   DO 130 J=1,NN
   DO 120 L=L1,L2
   DO 120 M=1,2
      KLM=K1+L+M
      120 G(I,J,KLM)=ADD(G(I,J,KLM),MUL(G(I,N,L),G(N,J,M)))
   DO 130 L=L3,L4
130  \text{G(I,J,L)} = \text{DIV(G(I,J,L), FACTL)}

\text{C USER MAIN PROGRAM CONTROLLED BYPASS AND ADDITIONAL OUTPUT C}

\text{GO TO (138,132), IGCMP}

132  \text{WRITE(6,990)}

990  \text{FORMAT('INTERVAL COEFFICIENT MATRICES COMPUTED BY SUBROUTINE GCOM MP: ')}

\text{DO 991 K=1,230}

\text{WRITE(6,992) K}

\text{DO 991 I=1,NN}

991  \text{WRITE(6,993) (G(I,J,K),J=1,NN)}

992  \text{FORMAT('OK=',2X,15)}


138  \text{RETURN}

\text{END}
SUBROUTINE PRCOMP

THE PURPOSE OF THE SUBROUTINE IS TO:

1. COMPUTE THE ENTIRE PRELIMINARY SEQUENCE OF INTERVAL BINOMIAL COEFFICIENTS C(1), ..., C(210) NECESSARY FOR THE CENTERED FORM COMPUTATIONS OF SUBROUTINE PHCOMP. THE SINGLE SUBSCRIPT HAS BEEN DEVISED FROM THE PARTIAL PASCAL'S TRIANGLE

\[
\begin{array}{ccccccc}
1 & & & & & & \\
& 1 & & & & & \\
& & 2 & 2 & & & \\
& & 1 & 2 & 3 & & \\
& & 1 & 2 & 3 & 3 & \\
& & 1 & 2 & 3 & 3 & 3 \\
& & & & & & \\
& & & & & & \\
& & & & & & \\
1 & 2 & 3 & 3 & 3 & 3 & 3 \\
\end{array}
\]
THE COEFFICIENTS ARE COMPUTED BY 'COMPLETING' PASCAL'S TRIANGLE

2. READ P, COMPUTE EPSMUL=1.0/16.0**P AND WRITE P AND EPSMUL
   FOR USER VERIFICATION (EPSMUL IS USED IN SUBROUTINE PHCOMP)

3. FOR EACH SUBINTERVAL OF <-(1,1), THETA(NSUBI), NSUBI=1,...,
   SUB, COMPUTE THE CENTERED FORM SUBINTERVAL REPRESENTATION
   (RE-USED THE VARIABLE NAME THETA)
   THETA(NSUBI) + ETA(NSUBI)

WHERE THE NEW THETA IS THE COMPUTED VALUE FOR THE CENTER OF
THE SUBINTERVAL AND ETA IS THE 'RADIUS INTERVAL' FOR THE
SUBINTERVAL WHICH IS SYMMETRIC ABOUT ZERO

COMPLEX ADD, SUB, MUL, DIV, C(210), THETA(25), ETA(25), CMPLX, ONE
REAL*8 AUGMNT
INTEGER P
COMMON/COMARG/ AUGMNT, EPSDVR
CGMCGN/PARM1/ KK, NN, MODE, NSUB
CGMCMN/PARMS/ TDELT, TSPAN, EPSMUL, IROLL, IENT
COMMON/PARM3/ INORM, IGCMP, IPRCMP, IDRVR, IPHCMP
COMM-CN/ARRAY1/ THETA, ETA
CMCMCN/ARRAY3/ C

COMPUTE THE BINOMIAL COEFFICIENTS C(1),...,C(210)

ONE=(1.0,1.0)
C(1)=ONE
KR=1
DO 100 I=2,20
KL=KR+1
KR=KL+I-1
DUM=I
C(KL)=CMPLX(DUM,DUM)
160 C(KR)=ONE
KL=2
DO 110 I=3,20
I1=I-1
I2=I-2
KLP=KL
KL=KL+I1
DO 110 J=1,I2
110 C(KL+J)=ADD(C(KLP+J-1),C(KLP+J))
C READ P, COMPUTE EPSMUL AND WRITE BOTH
C
WRITE(6,11)
11 FORMAT('OPROBLEM INPUT PARAMETERS READ AND COMPUTED BY SUBROUTINE
*PRCOMP:*)
REAC(5,1) P
1 FORMAT(I10)
EPSMUL=((1.0D+00)/((16.0D+00)**P))
WRITE(6,12) P, EPSMUL
12 FORMAT('OP=',5X,I5,', EPSMUL=1.0/16.0**P=',2X,E14.7)
C USER MAIN PROGRAM CONTROLLED BYPASS AND ADDITIONAL OUTPUT
C
GO TO (118,112),IPRCMP
112 WRITE(6,990)
990 FORMAT('OBOUNDING ARITHMETIC COMPUTED PASCAL'S TRIANGLE BINOMIAL
*COEFFICIENTS ARE:*)
WRITE(6,991) C
COMPUTE SUBINTERVAL CENTERED FORM REPRESENTATIONS

118 ADVR=NSUB
   DO 120 NSUBI=1,NSUB
   NDUM=-NSUB+2*NSUBI-1
   ANUML=NDUM-1
   ANUMC=NDUM
   ANUMR=NDUM+1
   TLT=ANUML/ADVR
   TCTR=ANUMC/ADVR
   TRT=ANUMR/ADVR
   THETA(NSUBI)=CMPLX(TCTR,TCTR)
   T1=TCTR-TLT
   T2=TRT-TCTR
   ETAMAX=AMAX1(T1,T2)
   120 ETA(NSUBI)=CMPLX(-ETAMAX,+ETAMAX)

USER MAIN PROGRAM CONTROLLED BYPASS AND ADDITIONAL OUTPUT

GO TO (128,122), IPRCMP

122 WRITE(6,992) NSUB
992 FORMAT('BOUNDBOUNDING ARITHMETIC COMPUTED SUBDIVISION OF THE (-1.0, +1.0) INTERVAL - CENTERED FORM - THETA + ETA',/
    ' NSUB = ', 2X, I5, ':
    WRITE(6,991) (THETA(NSUBI),NSUBI=1,NSUB)
    WRITE(6,991) (ETA(NSUBI),NSUBI=1,NSUB)
128 RETURN

END
SUBROUTINE DRIVER

THE PURPOSE OF THE SUBROUTINE IS TO:

1. INPUT THE VECTOR DIFFERENTIAL EQUATION INTERVAL INITIAL-VALUES (X(I,1), I=1,...,NN) THROUGH EQUIVALENCING, ALGORITHM TIME-INCREMENT (TDELTA) AND NUMBER OF TIME-INCREMENT STEPS (NSTEPS) REQUIRED FOR THE SOLUTION.

2. INITIALIZE EACH SUBPROBLEM VECTOR DIFFERENTIAL EQUATION TO THE INPUT INITIAL-CONDITIONS (XSUB(I,NSUBI,1), I=1,...,NN).

3. CONTROL (DRIVE) THE ALGORITHM ITERATIVE SOLUTION TECHNIQUE FOR EACH VALUE OF NSTEPI=1,...,NSTEPS, INITIALIZING IROLL=IENT=1:
   (A) COMPUTE PRESENT SOLUTION TIME(NSTEPI+1)=NSTEPI*TDELTA=TSPAN, PASSING TSPAN IN COMMON TO SUBROUTINE PHCOMP
   (B) CALL PHCOMP, RETURNING THE SUBPROBLEM PSEUDO FUNDAMENTAL MATRICES PHI(I,J,NSUBI) IN COMMON FOR THE ABOVE VALUE OF TSPAN (IF IROLL HAS BEEN PREVIOUSLY SET EQUAL TO 2, PHCOMP IS NOT CALLED HERE AND STEP (C) BELOW IS PERFORMED NEXT)
   (1) IF PHCOMP ENCOUNTERS A FAILURE TO PASS ITS PSEUDO FUNDAMENTAL MATRIX COMPUTATION ERROR CRITERIA, IT SETS IROLL=2 AND RETURNS WITHOUT CHANGING THE PREVIOUSLY COMPUTED PHI'S FOR (NSTEPI-1) AT TIME(NSTEPI)=(NSTEPI-1)*TDELTA
   (2) IF THE CALL TO PHCOMP IS THE 9TH SUCH CALL, PHCOMP AUTOMATICALLY SETS IROLL=2 AND RETURNS WITHOUT ALTERING THE PREVIOUSLY COMPUTED PHI'S (SINCE THE SUBPROBLEM SOLUTION STORAGE IS LIMITED TO THE 8 MOST RECENT VALUES AND THE PRESENT VALUE).
(C) Compute the subproblem solutions XSUB (depending on the value of IROLL) by the matrix interval calculations

\[ XSUB(i,NSUB,i) = PHI(i,J,NSUB) \times XSUB(j,NSUB,i) \]

Where

1. If IROLL=1, IX=NSTEPI+1 and IX0=1
2. If IROLL=2 (the first time IROLL=2 is encountered, ILOOP is given the constant value of NSTEPI at that time), IX=MOD(NSTEPI, ILOOP)+1, IX0=MOD(NSTEPI+1, ILOOP)+1

As an example of (B) and (C), suppose NSTEPI=5 and PHCOMP fails returning IROLL=2. The following sequence of calculations occurs (omitting the subproblem, matrix and vector subscripts):

<table>
<thead>
<tr>
<th>NSTEPI</th>
<th>IX</th>
<th>IX0</th>
<th>TSPAN = TIME(NSTEPI+1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>XSUB(2) = PHI \times XSUB(1); 1*TDELTA = TIME(2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>XSUB(3) = PHI \times XSUB(1); 2*TDELTA = TIME(3)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>XSUB(4) = PHI \times XSUB(1); 3*TDELTA = TIME(4)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>XSUB(5) = PHI \times XSUB(1); 4*TDELTA = TIME(5)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>PHCOMP FAILURE</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(PHI at 4*TDELTA=TIME(5) retained, PHCOMP will not be called again)

ROLL=2
ILOOP=NSTEPI=5
IX =MOD(5,5)+1=1
IX0=MOD(6,5)+1=2
XSUB(1) = PHI \times XSUB(2); 5*TDELTA = TIME(6)
XSUB(2) = PHI \times XSUB(3); 6*TDELTA = TIME(7)
XSUB(3) = PHI \times XSUB(4); 7*TDELTA = TIME(8)
XSUB(4) = PHI \times XSUB(5); 8*TDELTA = TIME(9)
XSUB(5) = PHI \times XSUB(1); 9*TDELTA = TIME(10)
XSUB(1) = PHI \times XSUB(2); 10*TDELTA = TIME(11)

ETC.

Thus the effect is to 'roll' the limited storage available.
FGR THE SUBPROBLEM SOLUTION VALUES XSUB

(C) COMPUTE THE VECTOR DIFFERENTIAL EQUATION INTERVAL SOLUTION BY

\[
X(I, NSTEP + 1) = \text{UNION OF } XSUB(I, NSUB, IX) \text{ OVER } NSUB = 1, \ldots, NSUB
\]

(E) UPDATE NPTS, THE NUMBER OF SOLUTION POINTS AVAILABLE FOR GRAPHICAL REPRESENTATION

NPTS = NSTEP + 1

(IE, INCLUDES THE INITIAL-VALUE AT TIME(1) = 0.0)

(F) TEST CPU CLOCK (SEE EXPLANATION GIVEN WITH THE SAMPLE MAIN PROGRAM)

COMPLEX ADD, SUB, MUL, DIV, XSUB(5, 25, 9), X(5, 100),
*PHI(5, 5, 25), DUM, DUMSB
REAL*8 AUGMNT, DUM, DUMSB, DXSUB(5, 25, 9), DX(5, 100)
REAL XSNGL(2, 5), TIME(100), SDUM(2), SDUMSB(2)
.EQUIVALENCE (X(1, 1), XSNGL(1, 1), DX(1, 1)), (DUM, DDUM, SDUM(1)),
*(DUMSB, DDUMSB, SDUMSB(1)), (XSUB(1, 1, 1), DXSUB(1, 1, 1))
COMMON/COMARG/ AUGMNT, EPSDVR
COMMON/PARM1/ KK, NN, MODE, NSUB
COMMON/PARM2/ TDELTA, TSPAN, EPSMUL, IROLL, IENT
COMMON/PARM3/ INORM, IGCMP, IPRCM, IDVR, IPHCMP
COMMON/ARRAY4/ PHI
COMMON/SOLN/ TIME, X, NPTS, ISTOP

SUBROUTINE DRIVER READ SEQUENCE

READ(5, 1) NSTEPS
1 FORMAT(I10)
READ(5, 2) ((XSNGL(I, J), I = 1, 2), J = 1, NN)
2 FORMAT(8F10.6)
READ(5,2) TDELTA
TFINAL=NSTEPS*TDELTA

WRITE SUBROUTINE DRIVER INPUT SEQUENCE VALUES FOR VERIFICATION
BY USER

WRITE(6,11) NSTEPS,TDELTA,TFINAL
11 FORMAT('PROBLEM INPUT PARAMETERS READ AND COMPUTED BY SUBROUTINE
*DRIVER: ',NSTEPS,'I5,/' SOLUTION TIME INCREMENT TDELTA=',
'E14.7,/' FINAL SOLUTION TIME = NSTEPS * TDELTA =',/
'E14.7,' DIFFERENTIAL EQUATION INTERVAL INITIAL CONDITIONS -'
WRITE(6,12) (X(I,1),I=1,NN)

INITIALIZE SUBPROBLEMS

DO 100 I=1,NN
   DO 100 NSUBI=1,NSUB
   100 DXSUB(I,NSUBI,1)=DX(I,1)
      TIME(1)=0.0

BEGINNING OF DO LOOP FOR STEPS 3-A,B,C,D,E AND F,
NSTEP1=1,...,NSTEPS

IROLL=1
IENT=1
DO 200 NSTEP1=1,NSTEPS

COMPUTE TSPAN=NSTEP1*TCELTA=TIME(NSTEP1+1)

NSTEP1=NSTEP1+1
TSPAN=NSTEP1*TDELTA
TIME(NSTEP1)=TSPAN

CALL PHCOMP AND COMPUTE IX AND IXO, ACCORDING TO IROLL
(SEE STEPS (B) AND (C) ABOVE)
GO TO (110, 135), IROLL
110 CALL PHCOMP
   GO TO (120, 130), IROLL
120 IX=NSTEP1
   I XO=1
   GO TO 140
130 I LOOP=NSTEP1
135 IX=MOD(NSTEP1, I LOOP)+1
   IXO=M CD(NSTEP1, I LOOP)+1

C COMPUTE SUBPROBLEM SOLUTIONS XSUB AND VECTOR DIFFERENTIAL EQUATION INTERVAL SOLUTION, ACCORDING TO IROLL (SEE STEPS (C) AND (D) ABOVE)

140 DO 190 I=1, NN
   IPASS=1
   DO 180 NSUBI=1, NSUB
   DDUMSB=0.0D+00
   DO 150 J=1, NN
   150 DDUMSB=DDUMSB+MUL(P HI(I, J, NSUBI), XSUB(J, NSUBI, IXO))
   GO TO (160, 170), IPASS
160 IPASS=2
   DDUM=DDUMSB
   GO TO 180
170 DDUM(1)=AMIN1(DDUM(1), DDUMSB(1))
   DDUM(2)=AMAX1(DDUM(2), DDUMSB(2))
180 D XSUB(I, NSUBI, IX)=DDUMSB
190 D X(I, NSTEP1)=DDUM

C OUTPUT ALGORITHM INTERVAL SOLUTIONS

WRITE(6, 13) N STEP1, TSPAN
13 FORMAT('ONSTEP1=', I5, 'TTIME=', E14.7, 'INTERVAL SOLUTION X(I),
*I = 1, . . . , NN -'
) WRITE(6, 12) (X(I, NSTEP1), I=1, NN)

C
C USER MAIN PROGRAM CONTROLLED BYPASS AND ADDITIONAL OUTPUT

GO TO (198,192), IDRVR
192 WRITE(6,990) NSTEPI, TSPAN
990 FORMAT('ONSTEP = ',I5, ' TIME = ',E14.7, ' INTERVAL SOLUTION XSUB(I
* , NSUBI), NSUBI = 1, . . . , NSUB; I= 1, . . . , NN - 1)
DO 993 I=1, NN
993 WRITE(6,991) (XSUB(I,NSUBI,IX),NSUBI=1,NSUB)

C UPDATE NPTS
C
198 NPTS=NSTP1
C
TEST CPU CLOCK AND BRANCH TO EARLY TERMINATION OF SOLUTION IF
INSUFFICIENT CPU STEP-TIME REMAINING (SEE EXPLANATION GIVEN
WITH SAMPLE MAIN PROGRAM)
C
CALL CLOCK(IBAL)
IF(ISTOP.GE.IBAL) GO TO 202
200 CONTINUE
C
END DC LOOP FOR STEP 3
C
GO TO 210
C
EARLY TERMINATION DIAGNOSTICS BRANCHPOINT
C
202 TBAL=IBAL*1.0E-02
TSTOP=ISTOP*1.0E-02
WRITE(6,14) TBAL, TSTOP, NPTS
14 FORMAT(' *  EARLY TERMINATION OF SOLUTION * * * * * ',/
** ' ,10X, ' BALANCE JOBSTEP CPU TIME = ',F10.3, /
** ' ,10X, ' INPUT CPU CUTOFF TIME = ',F10.3, /
** ' ,10X, ' ACTUAL NO. SOLN PTS RUN = ',I10)
210 RETURN
END
SUBROUTINE PHCOMP

THE PURPOSE OF THE SUBROUTINE IS TO:

1. FOR THE FIRST ENTRY INTO PHCOMP (SEE DRIVER INITIALIZATION OF IENT=1)
   (A) INPUT KK, NKNT1, NKNT2, NFAIL, NRUNUP
   (B) COMPUTE R*8 FACTORIALS NECESSARY IN THE COMPUTATION OF THE ZETA'S
   (C) INITIALIZE THE 'HISTORY' INTEGER ARRAY KKPast(I,J) TO THE PROBLEM STARTING VALUE NUMBER OF TERMS TO BE USED IN COMPUTING ALL SUBPROBLEM PSEUDO FUNDAMENTAL MATRICES FOR THE (I,J)TH ELEMENTS, KK
   (D) INITIALIZE PHCOMP FAILURE AND PHCOMP ENTRY COUNTERS, SETTING THE FIRST PROBLEM ENTRY SHUTOFF PARAMETER IENT=2
   (E) WRITE THE INPUT PARAMETERS FOR USER VERIFICATION

2. COMPUTE THE SUBPROBLEM PSEUDO FUNDAMENTAL INTERVAL MATRICES PHI(I,J,NSUBI) FOR USE IN SUBROUTINE DRIVER, AS DESCRIBED BELOW

COMPLEX ADD, SUB, MUL, DIV, C(210), THETA(25), ETA(25), ZAUG,
* G(5, 5, 230), DUM, D, CMPLX, B(21), ONE, D(210), E(21), PMID, PINT,
* PHI(5, 5, 25), LPHI(5, 5, 25)
REAL*8 AUGMNT, Z1, Z2, Z3, DPHI(625), DLPHI(625), FACTL(21), DB(21),
* DD(210), DE(21), DPMID, DPINT, DZAUG, DTHETA(25), DETA(25), DDUM
REAL EIVEC(25, 5), EVAL(25), ANORM(25), DUMSGL(2), LDUM
INTEGER KKPAST(5,5), KKDUM(25)
EQUIVALENCE (KKPAST(1,1), KKDUM(1)), (DUMSGL(1), DUM, DDUM),
*(PHI(1,1,1), DPHI(1)), (LPHI(1,1,1), DLPHI(1)), (B(1), DB(1)),
*(D(1), DD(1)), (E(1), DE(1)), (DZAUG, ZAUG), (DPMID, PMID),
*(PINT, DPINT), (THETA(1), DTHETA(1)), (ETA(1), DETA(1))
COMMON/COMARG/ AUGMN, EPSDV,
COMMON/PARM1/ KK, NN, MODE, NSUB,
COMMON/PARM2/ TDELTA, TSPAN, EPSMUL, IROLL, IENT,
COMMON/PARM3/ INORM, IGCMP, IPRCMP, IDRVR, IPHCMP,
COMMON/NORM1/ EIVEC, EVAL, ANORM,
COMMON/ARRAY1/ THETA, ETA,
COMMON/ARRAY2/ G,
COMMON/ARRAY3/ C,
COMMON/ARRAY4/ PHI

C
C
C FIRST PROBLEM PHCOMP ENTRY SEQUENCE AND CONTROL. IENT=1 BY DRIVER
C RESULTS IN FIRST ENTRY SEQUENCE BELOW WHERE IENT=2 SHUTS OFF
C SEQUENCE FOR SUBSEQUENT ENTRIES. UNACCEPTABLE PHCOMP RESULTS
C (IFAIL=NFAIL, SEE BELOW) SETS IENT=3 SO SUBSEQUENT PHCOMP ENTRY
C BYPASSES ALL COMPUTATIONS HERE.
C
GO TO (100, 120, 455), IENT
100 READ(5,1) KK
   1 FORMAT(8I10)
   READ(5,1) NKNT1, NKNT2, NFAIL, NRUNUP
   FACTL(1) = 1.0D+00
   DO 105 I = 2, 21
   105 FACTL(I) = FACTL(I-1) * I
   DO 110 I = 1, 25
   110 KKDUM(I) = KK
ONE = (1.0, 1.0)
IENT = 2
IFAIL = 0
IPHENT = 0
WRITE(6,11) KK,NKNT1,NKNT2,NFAIL,NRUNUP
11 FORMAT('OPROBLEM INPUT PARAMETERS READ AND COMPUTED BY SUBROUTINE
*PHCOMP:',/' KK=',4X,I5,' NKNT1=',1X,I5,' NKNT2=',1X,15,' NFAIL=',1X,I5,' NRUNUP=',5X,I5)
KOUNTS=NN*NN*NSUB

C PHCOMP ENTRY COUNTER. IF 9TH ENTRY, BYPASS ALL COMPUTATIONS HERE
C SETTING IROLL=2 AND RETURNING TO DRIVER (FOR REASONS, SEE DRIVER)
C
120 IPFENT=IPHENT+1
IF(IPFENT.EQ.9) GO TO 455

C INITIALIZE PHCCMP INTERVAL (COMPLEX) TIME VARIABLE T AS THE
C DEGENERATE INTERVAL <TSPAN,TSPAN>
C
T=CMPLX(TSPAN,TSPAN)

C FOR EACH PHCOMP ENTRY, ZERO THE SINGLE AND DOUBLE (FULL) FAULT
C ERROR COUNTERS
C
KNT1=0
KNT2=0

C (I,J)-LOOP:
C FOR EACH (I,J), COMPUTE THE SUBPROBLEM PSEUDO FUNDAMENTAL
C MATRIX ELEMENTS
C
DO 420 I=1,NN
DO 420 J=1,NN

C INITIALIZE IJENT=1, BYPASS PARAMETER WITHIN NSUBI-LOOP
C
IJENT=1

C NSUBI-LOOP:
C FOR EACH NSUBI=1,...,NSUB, COMPUTE THE PSEUDO FUNDAMENTAL MATRIX
C (I,J)TH ELEMENT
CO 410 NSUBI=1,NSUB
INITIALIZE IRUNUP AND THE NSUBI-TH CENTERED FORM REPRESENTATION
NON-SUBSCRIPTED PARAMETERS (ALGORITHM EFFICIENCY MEASURE)
IRUNUP=0
DPMID=DTHETA(NSUBI)
DPINT=DETA(NSUBI)
IJENT BYPASS CONTROL FOR NSUBI=2,...,NSUB
GO TO (125,180),IJENT
INITIALIZE KLOOP=KKPAST(I,J), THE MONOTONICALLY INCREASABLE
NSUBI-LOOP PARAMETER DETERMINING THE NUMBER OF TERMS TO BE USED
IN COMPUTING THE NSUBI-TH AND SUBSEQUENT, (I,J)TH ELEMENT OF THE
PSEUDO FUNDAMENTAL MATRIX
125 KLOOP=KKPAST(I,J)
COMPUTE THE NESTED FORM INTERVAL MATRIX ELEMENT (LETTING L=KLOOP
AND I = THE IDENTITY MATRIX)
B(L+1)=\ldots(K(L*(L+1)/2)*T+G((L-1)*L/2))T+\ldots+G(1))T+I, AND
B(L+1-M)=\ldots(K(L*(L+1)/2+M)*T+G((L-1)*L/2+M))T+
\ldots+G(M*(M+1)/2+M))T**M, M=1,...,L
REMARK - NOTE THAT THE B MATRIX ELEMENT DOES NOT DEPEND ON THE
SUBPROBLEM PARAMETER SUBINTERVAL INDEX NSUBI BUT IT
DOES DEPEND ON THE PARTICULAR VALUE OF KLOOP SET FOR
THE NSUBI,NSUBI+1,...,NSUB (I,J)TH ELEMENTS (A RUNUP
MAY CHANGE THE VALUE OF KLOOP AS NSUBI INCREASES)
130 KL1=KLOOP+1
KL2=KL1+1
KL3=KL0P-1
KL4=(KLOOP*KL1)/2
DUM=MUL(G(I,J,KL4),T)
DO 140 L=1,KL3
KR=((KLOOP-L)*(KL1-L))/2
140 DUM=MUL(ADD(DUM,G(I,J,KR)),T)
IF(I.EQ.J) DUM=ADD(DUM,ONE)
CB(KL1)=DDUM
DO 160 L=1,KLOOP
KI=KL4+L
LR=KL1-L
DUM=MUL(G(I,J,KI),T)
IF(LR.EQ.1) GO TO 160
LR1=LR-1
DO 150 M=1,LR1
LM=((KLOOP-M)*(KL1-M))/2+L
150 DUM=MUL(ADD(DUM,G(I,J,LM)),T)
160 CB(LR)=DDUM
DO 170 L=1,KL3
DO 170 M=1,L
170 B(M)=MUL(B(M),T)
C USER MAIN PROGRAM CONTROLLED BYPASS AND ADDITIONAL OUTPUT
C GO TO (180,172),IPHCMP
172 WRITE(6,990) I,J,NSUBI,KLOOP
990 FORMAT('I,J,NSUBI,KLOOP=',29X,I5,2X,I5,2X,I5,2X,I5,2X,I5,2X,
*INTERVAL COEFFICIENTS B(1), ..., B(KLOOP+1) ARE:')
WRITE(6,991) (B(L),L=1,KL1)
C SET PSEUDO FUNDAMENTAL MATRIX ELEMENT 'ACCEPTED' SINGLE OR
C DOUBLE ENDPOINT ERROR TRIGGER TO PASS OUTPUT DIAGNOSTIC
C MESSAGE, ITRIG=1. IF A FAULT OCCURS WHICH IS ACCEPTED, AN
C ITRIG=2 STATEMENT IS ACTUATED AND THE DIAGNOSTIC MESSAGE
C WILL BE PRODUCED
C 180 ITRIG=1
C COMPUTE (IN R*8) THE R*4 HOUSEHOLDER MATRIX NORM BOUND ZETA ON
THE REMAINDER OF THE MATRIX EXPONENTIAL FOR THIS (I,J)TH AND
NSUBI-TH SUBPROBLEM ELEMENT

\[
Z1 = \text{ANORM}(\text{NSUBI})
\]

\[
Z1 = Z1 \times TSPAN
\]

190 \[
Z2 = \text{FACTL}(KL1)
\]

\[
\zeta = \left((Z1^{*KL1}/Z2) \times (1.0D+00/(1.0D+00-Z1/KL2))\right)
\]

RECURSIVELY COMPUTE INTERVAL MATRIX ELEMENTS D(1),...,D(K*(K+1)/2)
INVOLVED IN THE NESTED COMPUTATIONS (FOR CONVENIENCE HERE,
P=PMID=\text{THETA}(\text{NSUBI}) AND K=KLOOP)

\[
E(K-(L-1)) = \begin{array}{cccccc}
\ldots & (B(1)C \times P+B(2)C \times P+ & \\
 & \ldots+B(K-L)C \times P+B(K-(L-1))C \times P+ & \\
L+1 & L & L & L & L & L
\end{array}, \quad L=1,\ldots,K
\]

\[
K \quad K-1 \quad L+1 \quad L \quad L \quad L \quad L
\]

\[
= \begin{array}{cccccc}
\ldots & (D \times P+D \times P+ \ldots+D \times P+D) \times P+ & \\
L & L & L & L & L & L
\end{array}, \quad L=1,\ldots,K
\]

\[
K-(M-1) \quad K-(M-1)
\]

WHERE D \[
= B(M)C, \quad M=1,\ldots,K-(L-1) \quad \text{AND} \quad L=1,\ldots,K
\]

THE SINGLE SUBSCRIPT FOR C = C(#) AND D = D(#) IS DEFINED AS

\[
#=I*(I-1)/2+J
\]

DO 200 L=1,KLOOP

L1=KL1-L
LM=((KL1-L)*(KL2-L))/2+1
DO 200 M=1,LR1
LMR=LM-M
200 D(LMR)=MUL(C(LMR),B(L))
C
C COMPUTE THE MATRIX ELEMENT DEFINED BY THE NESTED COMPUTATION
C (K AND P AS ABOVE)
C E(K+1)=(...(B(1)*P+B(2))*P+...+B(K))*P+B(K+1)
C
DDUM=CB(1)
DO 210 L=2,KL1
210 DUM=ADD(MUL(DUM,PMID),B(L))
DE(KL1)=DDUM
C
C USING THE D ELEMENTS, COMPUTE E(K-(L-1)), L=1,...,K, IN
C NESTED FORM AS DEFINED ABOVE
C
KC=(KLOOP*KL3)/2
DO 230 L=1,KLOOP
LR=KL1-L
KI=KC+L
DDUM=DD(KI)
IF(LR.EQ.1) GO TO 230
LR1=LR-1
DO 220 M=1,LR1
LM=(({(KLOOP-M)*KL3-M)/2)+L)
220 DUM=ACC(PUL(DUM,PMID),C(LM))
230 DE(LR)=DCUM
C
C COMPUTE THE NESTED INTERVAL RESULT (FOR CONVENIENCE HERE,
C Q=PINT=ETA(NSUBI) AND K=KLOOP)
C DUM=(...(E(1)*Q+E(2))*Q+...+E(K))*Q+E(K+1),
C WHICH IS THE NESTED CENTERED FORM INTERVAL RESULT FOR THE K-TH
C PARTIAL SUM (I,J)TH ELEMENT OF THE NSUBI-TH SUBPROBLEM
C FUNDAMENTAL MATRIX
C
DDUM=DE(1)
DO 240 L=2,KL1


THE FOLLOWING IS A BRIEF EXAMPLE OF TESTING AND AUGMENTING OF THE (KLOOP)TH PARTIAL SUM DUM (IE, THE (I,J)TH ELEMENT FOR SUBPROBLEM NSUBI)

LET MODE=1 (IE, NORMAL HOUSEHOLDER MATRIX NORM, NOT TRANSPOSE)
LET DUM=<LDUM,RDUM> AND ASSUME 0.0 < LDUM < RDUM.
LET THE I AND J-TH ELEMENTS OF THE HOUSEHOLDER EIGENVECTOR BE U(I) AND U(J) (IE, THE REAL POSITIVE EIGENVECTOR CORRESPONDING TO THE REAL POSITIVE MAXIMAL EIGENVALUE)
RECALL EPSNUL=1/16**P (SEE SUBROUTINE PRCOMP)
THE ERROR CRITERIA TEST IN THE EXAMPLE IS PASSED IF CKZETA=(U(I)/U(J))*ZETA IS LESS THAN OR EQUAL TO LDUM*EPSMUL.
DUM IS THEN AUGMENTED TO FORM THE PSEUDO ELEMENT
DUM=DUM+<-CKZETA,+CKZETA>.

THIS PSEUDO SUBPROBLEM FUNDAMENTAL INTERVAL MATRIX ELEMENT SET-THEORETICALLY INCLUDES THE ACTUAL FUNDAMENTAL INTERVAL MATRIX ELEMENT AND THE CORRESPONDING INTERVAL ENDPOINT RELATIVE ERRORS ARE LESS THAN 2*(16**(-P)/(1-16**(-P))).

(A) IF TESTING FAILS THE ERROR CRITERIA, WHERE

A SINGLE FAULT (KNT1) = MIN ABSOLUTE VALUE ENDPOINT FAILURE, AND
A DOUBLE FAULT (KNT2) = MAX ABSOLUTE VALUE ENDPOINT FAILURE (THIS INCLUDES THE MIN ABSOLUTE VALUE ENDPOINT FAILURE),
THE OPERATION IS TO FIRST RUNUP (INCREASE) KLOOP BY 1
(ASSUMING IT WAS LESS THAN 20) AND TRY AGAIN (WHICH WILL MEAN
RECOMPUTING DUM BEGINNING WITH THE B ELEMENTS). IF THE TEST
STILL FAILS AND KLOOP=20 OR THE NUMBER OF RUNUPS ATTEMPTED
(IRUNUP) IS GREATER THAN NRUNUP, INDEX THE 'ACCEPTED' ELEMENT
ERROR COUNTER KNT1 OR KNT2 (NOTE THAT THESE COUNTERS
ACCUMULATE FOR ALL I, J AND ALL NSUBI IN THIS PHCOMP ENTRY)
AND AUGMENT DUM APPROPRIATELY STORING THE RESULT IN THE
TEMPORARY (LOCAL) PSEUDO SUBPROBLEM FUNDAMENTAL INTERVAL
MATRIX ARRAY ELEMENT LPHI(I,J,NSUBI).

(B) IF TESTING PASSES THE ERROR CRITERIA, DUM IS AUGMENTED
APPROPRIATELY AND THE RESULT STORED IN LPHI(I,J,NSUBI),
WITHOUT 'ACCEPTED' ERROR COUNTING.

NOTE: THE LAST STATEMENT OF THE NSUBI-LOOP IS '410 IJENT=2', WITH
THE RESULT THAT COMPUTATIONS FOR THE NEXT VALUE OF NSUBI
(I,J REMAIN THE SAME) WILL USE THE SAME VALUE OF KLOOP
AND THEREFORE THE SAME B'S, BYPASSING COMPUTATION OF THE
B ELEMENTS WHICH DO NOT DEPEND ON NSUBI.

LDUM=DUMSGL(1)
RDUM=DUMSGL(2)
GO TO (250,260),MODE
250 Z3=EIVEC(NSUBI,I)
Z3=Z3/EIVEC(NSUBI,J)
GO TO 270
260 Z3=EIVEC(NSUBI,J)
Z3=Z3/EIVEC(NSUBI,I)
270 CKZETA=Z3*ZETA
ISTAT=1
IF(LDUM.EQ.RDUM) GO TO 390
IF(LDUM.GT.0.0) GO TO 300
IF(RDUM.LT.0.0) GO TO 280
IF(LDUM.EQ.0.0) GO TO 290
IF(RDUM.EQ.0.0) GO TO 310
RLMIN=EPSMUL*AMIN1(-LDUM,RDUM)
IF(CKZETA.LE.RLMIN) GO TO 370
ISTAT=2
RLMAX=EPSMUL*AMAX1(-LDUM,RDUH)
IF(CKZETA.LE.RLMAX) GO TO 320
GO TO 340
280 RDUM=EPSMUL*RDUM
IF(RDUM.LE.-CKZETA) GO TO 370
ISTAT=2
290 LDUM=EPSMUL*LDUM
IF(LDUM.LE.-CKZETA) GO TO 330
GO TO 340
300 LDUM=EPSMUL*LDUM
IF(CKZETA.LE.LDUM) GO TO 370
ISTAT=2
310 RDUM=EPSMUL*RDUM
IF(CKZETA.LE.RDUM) GO TO 320
IF(ISTAT.EQ.1) ISTAT=3
GO TO 340
320 IF(ISTAT.EQ.1) ISTAT=3
330 GO TO (350,331,3601,ISTAT
331 IRUNUP=IRUNUP+1
IF(IRUNUP.GT.NRUNUP) GO TO 332
IF(KLCP.LT.20) GO TO 380
332 KNT1=KNT1+1
GO TO 342
340 IF(KLCP.LT.20) GO TO 380
KNT2=KNT2+1
342 ITRIG=2
GO TO (350,370,360),ISTAT
350 ZAUG=CMPLX(-CKZETA,0.0)
GO TO 400
360 ZAUG=CMPLX(0.0,CKZETA)
GO TO 400
370 ZAUG=CMPLX(-CKZETA,CKZETA)
GO TO 400
C
C KLCOP RUNUP WARNING DIAGNOSTIC MESSAGE
380 WRITE(6,12) I,J,NSUBI,KLOOP,Z1,Z2,ZETA
KLOOP=KLOOP+1
GO TO 130
390 DZAUG=0.0D+00
400 LPHI(I,J,NSUBI)=ADD(DUM,ZAUG)
GO TO (404,402),ITRIG
C
C 'ACCEPTED' SINGLE OR DOUBLE FAULT ERROR DIAGNOSTIC MESSAGE
C
402 WRITE(6,15) I,J,NSUBI,KLOOP,Z1,Z2,ZETA
15 FORMAT('** HALF FAULT WITH KLOOP.LE.20 OR FULL FAULT WITH **KLOOP.EQ.20',/ ** I, J, NSUBI, KLOOP=','29X,I5,2X,I5,2X,I5,2X,I5,2X,1X,'; Z1, Z2, ZETA, ZAUG, DUM, PHI(I,J,NSUBI)=','/ ** D14.7,2X,** ** DUM=E14.7) **
GO TO 408
C
C USER MAIN PROGRAM CONTROLLED BYPASS AND ADDITIONAL OUTPUT
C
404 GO TO (410,406),IPHCMP
406 WRITE(6,992) I,J,NSUBI,KLOOP,Z1,Z2,ZETA
GO TO 408
408 WRITE(6,16) ZAUG,DUM,LPHI(I,J,NSUBI)
410 IJENT=2
C
C NSUBI-LOOP COMPLETE. KKPAST(I,J) IS SET EQUAL TO THE VALUE OF **
C KLOOP SO THAT THE NEXT PHCOMP ENTRY WITH THESE VALUES OF I,J AND **
C WITH A LARGER VALUE OF TSPAN BEGINS WITH KLOOP=KKPAST(I,J). **
C
420 KKPAST(I,J)=KLOOP
C PSEUDO SUBPROBLEM FUNDAMENTAL INTERVAL MATRIX COMPUTATIONS
C COMPLETE. OUTPUT ERROR COUNTS AND ARRAY KKPAST(I,J).
C
WRITE(6,17) KNT1,KNT2,KOUNTS
17 FORMAT('O- - - - PHI(I,J,NSUBI) COMPLETE. ERROR COUNT:*,7X,
*I5,2X,'SNGL FAULTS',*,2X,I5,2X,'DBLE FAULTS',*,2X,I5,2X,'TOTAL PHI E
ELEMENTS. *')
WRITE(6,18)
18 FORMAT('O- - - - FINAL KKPAST(I,J) HISTORY, I.E., SETUP FOR NEXT
* ENTRY INTO SUBROUTINE PHCOMP:')
DO 425 I=1,NN
425 WRITE(6,19) (KKPAST(I,J),J=1,NN)
19 FORMAT(' ',I5,2X,I5,2X,I5,2X,I5)

C CHECK TO DETERMINE IF 'ACCEPTED' ERROR COUNTS KNT1 OR KNT2 HAVE
C EXCEEDED USER INPUT VALUES NKNT1 AND NKNT2 FOR THIS PHCOMP ENTRY.
C IF NOT, STORE RESULTS LPHI IN ARRAY PHI WHICH IS COMMON TO DRIVER
C AND RETURN. IF THERE IS AN EXCESS, THE PHCOMP FAILURE COUNTER
C IFAIL IS INCREASED BY 1. IF IFAIL IS GREATER THAN OR EQUAL TO
C THE USER INPUT NFAIL, THE SUBROUTINE SETS IENT=3 (CAUSING A
C COMPLETE NEXT PHCOMP ENTRY BYPASS) AND IROLL=2 (CAUSING DRIVER
C TO REVERT TO THE PREVIOUS PHI'S) AND RETURNS. IF IFAIL IS LESS
C THAN NFAIL, THE LPHI RESULTS ARE STORED IN PHI AND A NORMAL RETURN
C TO DRIVER OCCURS.
C
C CNE OTHER PHCOMP FAILURE MECHANISM IS POSSIBLE. IF A PHCOMP
C FAILURE OCCURS ON THE FIRST ENTRY, A FATAL ERROR MESSAGE IS
C PRODUCED AND THE PROGRAM IS TERMINATED.
C
IF(KNT2.GT.NKNT2) GO TO 450
IF(KNT1.GT.NKNT1) GO TO 450
430 DO 440 I=1,625
440 DPHI(I)=DLPHI(I)
RETURN
450 IF(IP=ENT.EQ.1) GO TO 460
IFAIL=IFAIL+1
IF(IFAIL.GE.NFAIL) IENT=3
```
WRITE(6,13) IPHENT, IFAIL, KNT1, KNT2
13 FORMAT(*0* * * * * FAIL TO SATISFY PHI ERROR CRITERIA FOR TSPAN='*,I5,'*TDELTA. IFAIL='*,I5,'*, KNT1='*,I5,'*, KNT2='*,I5)'
    IF(IFAIL.LE.NFAIL) GO TO 430
455 IROLL=2
RETURN
460 WRITE(6,14) IPHENT
14 FORMAT(*0* * * * * FAIL TO SATISFY PHI ERROR CRITERIA FOR TSPAN='*,I5,'*TDELTA * * * STOP RUN SINCE FIRST ENTRY IN SUBROUTINE PHCOMP **)
    CALL EXIT
END
```
SUBROUTINE CURVES

THE PURPOSE OF THE SUBROUTINE IS TO PRODUCE MULTIPLE SOLUTION
GRAPHS ON THE CALCOMP DRUM-TYPE INCREMENTAL PLOTTER. THE
SUBROUTINE USES THE FORTRAN SIMPLOTTER ROUTINES AND THE USER
DETERMINES THE SEQUENCE OF GRAPHS TO BE PRODUCED BY THE INPUT
SEQUENCE OF DATA CARDS.

THE NUMBER OF SEPARATE GRAPHS TO BE PRODUCED (IE, PRIMARY CALLS,
CALL GRAPH(LONG ARGUMENT LIST)) IS DETERMINED BY THE INPUT INTEGER
NPRI (MIN OF 1).

THE NUMBER OF SECONDARY CURVES TO BE PRODUCED ON A PARTICULAR
GRAPH (IE, SUPERPOSITION CALLS, CALL GRAPH(ABBREVIATED ARGUMENT
LIST)) IS DETERMINED BY THE INPUT INTEGER NSEC (MAY BE 0).

THE NUMBER OF SUPPLEMENTARY LETTER ANNOTATIONS TO BE PRODUCED
ON A PARTICULAR GRAPH (IE, SUPERPOSITION CALLS, CALL LETTRS(
ABBREVIATED ARGUMENT LIST)) IS DETERMINED BY THE INPUT INTEGER
NLTRS (MIN OF 1).

USER INPUT DATA CONTROL OF THE SUBROUTINE IS READILY APPARENT
FROM THE SUBSEQUENT LISTING AND THE VARIABLE DEFINITIONS
AND SUBROUTINE INPUT SEQUENCE ANNOTATIONS ACCOMPANYING THE MAIN
PROGRAM.

COMPLEX X(5,100)
REAL TIME(100),XPLT(100),XSNGL(2, 5, 100),XLAB(5),YLAB(5),
*GLAB(5),DATLAB(5),STRING(20)
EQUIVALENCE (X(1, 1),XSNGL(1, 1, 1))
COMMCM/ SOLN/ TIME, X, NPTS, ISTOP

READ(5, 1) NPRI
1 FORMAT(4I10, 5A4)
  DO 150 NPLT = 1, NPRI
  READ(5, 2) XLAB, YLAB, GLAB
2 FORMAT(20A4)
  READ(5, 3) XSIZE, YSIZE, XSF, XMIN, NSEC, NLTRS, YSF, YMIN
3 FORMAT(4F10.6, 2I10, 2F10.6)
  NSEC = NSEC + 1
  ITRIG = 1
  DO 130 NSUP = 1, NSEC
  READ(5, 1) ISYM, MODE, IBND, ISTATE, DATLAB
  DO 100 J = 1, NPTS
100 XPLT(J) = XSNGL(IBND, ISTATE, J)
  GO TO (110, 120), ITRIG
110 CALL GRAPH(NPTS, TIME, XPLT, ISYM, MODE, XSIZE, YSIZE, XSF, XMIN, YSF,
               *YSF, XMIN, YSF, YMIN, XLAB, YLAB, GLAB, DATLAB)
  ITRIG = 2
  GO TO 130
120 CALL GRAPHSCNPTS(NPTS, TIME, XPLT, ISYM, MODE, DATLAB)
130 CONTINUE
  DO 140 NWRIT = 1, NLTRS
  READ(5, 3) XO, YO, HEIGHT, THETA, NCHAR
  READ(5, 2) STRING
140 CALL LETTRS(XO, YO, HEIGHT, STRING, THETA, NCHAR)
150 CONTINUE
RETURN
END
APPENDIX C. SELECTED RESULTS FROM FUNCTIONAL ANALYSIS

The analytic results developed in Chapter II utilize several well known metric space results from functional analysis. In an attempt to make this dissertation reasonably self-contained, a limited collection of definitions and theorems for this area will be presented. By necessity the treatment will be brief and incomplete. It will be assumed that the reader is familiar with standard mathematical notation and set theory.

**Definition C-1:** A **metric space** is a pair of objects, a set \( X \) and a real valued distance function \( d \), called the metric, defined for every \( x, y, z \in X \) and satisfying the following properties:

(i) \( d(x,y) = d(y,x) \);

(ii) \( d(x,y) \geq 0 \), \( d(x,y) = 0 \) if and only if \( x = y \);

and

(iii) \( d(x,y) \leq d(x,z) + d(y,z) \).

The properties are respectively called **symmetry**, **positive** and **strictly positive** and the **triangle inequality**.

**Definition C-2:** If \( \{X,d\} \) is a metric space and \( Y \subseteq X \), \( \{Y,d\} \) is called a **metric subspace** of \( \{X,d\} \).
Definition C-3: A sequence \( \{x_n\} \) in a metric space \( \{X,d\} \) is said to be a Cauchy sequence if for every \( \varepsilon > 0 \) there exists a positive integer \( N(\varepsilon) \) such that \( d(x_n, x_m) < \varepsilon \) for every \( n, m \geq N(\varepsilon) \).

Definition C-4: A mapping \( f \) of a metric space \( \{X,d_1\} \) onto a metric space \( \{Y,d_2\} \) is said to be an isometry if for every \( x, y \in \{X,d_1\}, d_1(x,y) = d_2(f(x),f(y)) \). In this case \( \{X,d_1\} \) and \( \{Y,d_2\} \) are said to be isometric.

Definition C-5: A metric space \( \{X,d\} \) is said to be complete if every Cauchy sequence in \( \{X,d\} \) is a convergent sequence in \( \{X,d\} \).

Theorem C-6: If \( \{X,d\} \) is a complete metric space and \( \{Y,d\} \) is a metric subspace of \( \{X,d\} \), \( \{Y,d\} \) is complete if and only if \( Y \) is a closed set in \( \{X,d\} \).

Theorem C-7: If the metric spaces \( \{X,d_1\} \) and \( \{Y,d_2\} \) are isometric, then \( \{X,d_1\} \) is complete if and only if \( \{Y,d_2\} \) is complete.

Definition C-8: A metric space \( \{X,d\} \) is said to be compact (Heine-Borel compact) if every open covering of \( \{X,d\} \) contains a finite open subcovering. A set \( A \) in a metric space \( \{X,d\} \) is said to be a compact set if the metric subspace \( \{A,d\} \) is compact.
The topology for a metric space consists of the family of all open sets induced by the metric. An open covering for a set in a metric space is a collection of open sets selected from the topology such that the set is contained in the union of the collection. Thus compactness is a topological property.

**Theorem C-9:** If $\mathcal{S}$ is an open covering of the set $S$ in $\{\mathbb{R}^k, D_2\}$, where for $x, y \in \mathbb{R}^k$,

$$D_2(x, y) \equiv \left\{ \frac{1}{k} \sum_{i=1}^{k} (x_i - y_i)^2 \right\}^{1/2},$$

and if $S$ is both closed and bounded, then a finite number of the sets in $\mathcal{S}$ will suffice to cover $S$ (Heine-Borel theorem).

The metrics

$$D_1(x, y) \equiv \sum_{i=1}^{k} |x_i - y_i|, \quad D_\infty(x, y) \equiv \max_{i} |x_i - y_i|$$

and $D_2$ yield the same topology for $\mathbb{R}^k$ and the metrics are equivalent. Thus any closed and bounded set in $\{\mathbb{R}^k, D_i\}$, where $i$ denotes 1, 2 or $\infty$, is compact.

**Definition C-10:** If $\{X_1, d_1\}, \ldots, \{X_n, d_n\}$ are metric spaces, the metric space $\{X, d\}$ with $X \equiv X_1 \circledast X_2 \circledast \ldots \circledast X_n$, where $x \equiv (x_1, \ldots, x_n) \in X$ means $x_1 \in X_1, \ldots, x_n \in X_n$, and with $d(x, y) \equiv \max_{i} d_i(x_i, y_i)$, is called a metric product space.
The set of all ordered pairs \((x_1, \ldots, x_n), x_1 \in X_1, \ldots, x_n \in X_n\) is called the **Cartesian product** \(X_1 \times X_2 \times \cdots \times X_n\).

**Theorem C-11:** If the metric spaces \(\{X_1, d_1\}, \ldots, \{X_n, d_n\}\) are each compact, the metric product space of Definition C-10 is compact.

**Theorem C-12:** The metric product space of Definition C-10 is complete if and only if the metric spaces \(\{X_1, d_1\}, \ldots, \{X_n, d_n\}\) are each complete.

**Definition C-13:** A function \(f\) mapping the metric space \(\{X, d_1\}\) into the metric space \(\{Y, d_2\}\) is said to be **continuous** at the point \(x_0 \in \{X, d_1\}\) if for every \(\epsilon > 0\) there exists a \(\delta(\epsilon, x_0) > 0\) such that \(d_2(f(x_0), f(x)) < \epsilon\) whenever \(d_1(x, x_0) < \delta(\epsilon, x_0)\). An alternative criterion: \(f\) is continuous at \(x\) if and only if \(\{x_n\} \to x\) always implies \(\{f(x_n)\} \to f(x)\).

**Theorem C-14:** If \(f\) and \(g\) are respectively continuous functions mapping the metric spaces \(\{X_1, d_1\}\) into \(\{X_2, d_2\}\) and \(\{X_2, d_2\}\) into \(\{X_3, d_3\}\), then the composition \(h = gf\) is a continuous function mapping \(\{X_1, d_1\}\) into \(\{X_3, d_3\}\).

**Definition C-15:** A function \(f\) mapping the metric space \(\{X, d_1\}\) into the metric space \(\{Y, d_2\}\) is said to be **uniformly continuous** on \(\{X, d_1\}\) if for each \(\epsilon > 0\) there exists a
\( \delta(\varepsilon) > 0 \) such that for any \( x_0 \in X, d_2(f(x_0),f(x)) < \varepsilon \) whenever \( d_1(x,x_0) < \delta(\varepsilon) \).

**Definition C-16:** Let the function \( f \) and the sequence of functions \( \{f_n\} \) map the metric space \( (X,d_1) \) into the metric space \( (Y,d_2) \). The sequence is said to converge to \( f \) uniformly on \( X \) if for every \( \varepsilon > 0 \) there exists a positive integer \( N(\varepsilon) \) such that \( d_2(f_n(x),f(x)) < \varepsilon \) for all \( n \geq N(\varepsilon) \), for each \( x \in X \).

**Theorem C-17:** If \( f \) is a continuous function mapping the metric space \( (X,d_1) \) into the metric space \( (Y,d_2) \) and if \( \{X,d_1\} \) is compact, then the range of \( f \) is a compact set in \( \{Y,d_2\} \).

**Theorem C-18:** If \( f \) is a continuous function mapping the metric space \( (X,d_1) \) into the metric space \( (Y,d_2) \) and if \( \{X,d_1\} \) is compact, then \( f \) is uniformly continuous on \( \{X,d_1\} \).