STRAPDOWN NAVIGATION COMPUTER STUDIES

Final Report

Contract No. NAS 9-9705

MARCH 1970

Prepared for

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
MANNED SPACECRAFT CENTER
HOUSTON, TEXAS

SPERRY RAND RESEARCH CENTER
SUDBURY, MASSACHUSETTS 01778
NOTICE

THIS DOCUMENT HAS BEEN REPRODUCED FROM THE BEST COPY FURNISHED US BY THE SPONSORING AGENCY. ALTHOUGH IT IS RECOGNIZED THAT CERTAIN PORTIONS ARE ILLEGIBLE, IT IS BEING RELEASED IN THE INTEREST OF MAKING AVAILABLE AS MUCH INFORMATION AS POSSIBLE.
STRAPDOWN NAVIGATION COMPUTER STUDIES

Final Report
Contract No. NAS 9-9705

MARCH 1970

Prepared for
National Aeronautics and Space Administration
Manned Spacecraft Center
Houston, Texas
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>CONFIGURATION AND COMPONENT TECHNOLOGY FOR A STRAPDOWN COMPUTER</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>CONVERSION METHODS FOR TRANSFORMATION FROM DODECAHEDRON TO TRIAD AXES</td>
<td>11</td>
</tr>
<tr>
<td>3.1</td>
<td>Direct Realization by Use of Pseudo-Inverse and Status Matrices</td>
<td>11</td>
</tr>
<tr>
<td>3.1.1</td>
<td>Theoretical Analysis of Approach</td>
<td>11</td>
</tr>
<tr>
<td>3.1.2</td>
<td>Implementation of Pseudo-Inverse and Status Matrices Method Using LSI Packaged Basic Elements</td>
<td>13</td>
</tr>
<tr>
<td>3.2</td>
<td>Conversion from Dodecahedron to Triad by Sensor Correction</td>
<td>18</td>
</tr>
<tr>
<td>3.2.1</td>
<td>Description</td>
<td>18</td>
</tr>
<tr>
<td>3.2.2</td>
<td>Parity Integration</td>
<td>25</td>
</tr>
<tr>
<td>3.2.3</td>
<td>Constant Rate Multiplier</td>
<td>29</td>
</tr>
<tr>
<td>4</td>
<td>INCREMENTAL COMPUTER CONFIGURATION FOR A REDUNDANT CMG CONTROL SYSTEM</td>
<td>35</td>
</tr>
<tr>
<td>4.1</td>
<td>Introduction</td>
<td>35</td>
</tr>
<tr>
<td>4.2</td>
<td>General Description of the CMG Control System</td>
<td>35</td>
</tr>
<tr>
<td>4.2.1</td>
<td>The CMG Configuration</td>
<td>35</td>
</tr>
<tr>
<td>4.2.2</td>
<td>CMG Steering Laws</td>
<td>37</td>
</tr>
<tr>
<td>4.2.3</td>
<td>The Rate Gyro Configuration</td>
<td>40</td>
</tr>
<tr>
<td>4.2.4</td>
<td>Control Computer Description</td>
<td>45</td>
</tr>
<tr>
<td>4.2.5</td>
<td>Fail-Operational Computation</td>
<td>49</td>
</tr>
<tr>
<td>4.3</td>
<td>Steering Law Computations</td>
<td>51</td>
</tr>
<tr>
<td>4.3.1</td>
<td>Equations and Computer Structure</td>
<td>51</td>
</tr>
<tr>
<td>4.3.2</td>
<td>Incremental Computation</td>
<td>58</td>
</tr>
<tr>
<td>4.3.3</td>
<td>The $\Delta A$ Computation</td>
<td>59</td>
</tr>
<tr>
<td>4.3.4</td>
<td>The $\Delta B$ Computation</td>
<td>66</td>
</tr>
<tr>
<td>4.3.5</td>
<td>The $\Delta C$ Computation</td>
<td>70</td>
</tr>
<tr>
<td>4.3.6</td>
<td>The $\Delta d_0$ Computation</td>
<td>72</td>
</tr>
<tr>
<td>4.3.7</td>
<td>The $\Delta D$ Computation</td>
<td>72</td>
</tr>
<tr>
<td>4.3.8</td>
<td>The $\Delta u$ Computation</td>
<td>76</td>
</tr>
<tr>
<td>4.3.9</td>
<td>The $\Delta \omega_0$ Computation</td>
<td>78</td>
</tr>
<tr>
<td>4.4</td>
<td>Other Computations</td>
<td>81</td>
</tr>
<tr>
<td>4.4.1</td>
<td>Trial Conversion</td>
<td>81</td>
</tr>
<tr>
<td>4.4.2</td>
<td>Failure Monitoring and Mode Control</td>
<td>85</td>
</tr>
<tr>
<td>4.5</td>
<td>Conclusions and Recommendations</td>
<td>86</td>
</tr>
<tr>
<td>5</td>
<td>SIMU DIRECTION COSINE SIMULATION PROGRAM FOR CHECKOUT RUNS</td>
<td>88</td>
</tr>
<tr>
<td>5.1</td>
<td>Description of Method and Results</td>
<td>88</td>
</tr>
<tr>
<td>Chapter</td>
<td>Page</td>
<td></td>
</tr>
<tr>
<td>---------------------------------</td>
<td>------</td>
<td></td>
</tr>
<tr>
<td>5.1.1 Types of Runs</td>
<td>88</td>
<td></td>
</tr>
<tr>
<td>5.1.2 Results</td>
<td>88</td>
<td></td>
</tr>
<tr>
<td>5.2 Program Description</td>
<td>90</td>
<td></td>
</tr>
</tbody>
</table>

APPENDIX A - Smooth Pulse Sequences
## LIST OF ILLUSTRATIONS

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-1</td>
<td>Block diagram of SIMU computer.</td>
<td>3</td>
</tr>
<tr>
<td>2-2</td>
<td>Resolver direction cosine unit.</td>
<td>5</td>
</tr>
<tr>
<td>2-3</td>
<td>Sequencing of all-resolver cosine computer.</td>
<td>5</td>
</tr>
<tr>
<td>2-4</td>
<td>TMR resolver.</td>
<td>10</td>
</tr>
<tr>
<td>3-1</td>
<td>Incremental input encoding for the six-gyro system</td>
<td>14</td>
</tr>
<tr>
<td>3-2</td>
<td>Block diagram of six-gyro system using pseudo-inverse and status matrices method.</td>
<td>19</td>
</tr>
<tr>
<td>3-3</td>
<td>Block diagram of sensor correction method.</td>
<td>19</td>
</tr>
<tr>
<td>3-4</td>
<td>Dodecahedron gyro correction resolver.</td>
<td>22</td>
</tr>
<tr>
<td>3-5</td>
<td>Triad axis generator.</td>
<td>24</td>
</tr>
<tr>
<td>3-6</td>
<td>DDA solution.</td>
<td>26</td>
</tr>
<tr>
<td>3-7</td>
<td>Parity integrator.</td>
<td>27</td>
</tr>
<tr>
<td>3-8</td>
<td>Pulse rate multiplier.</td>
<td>30</td>
</tr>
<tr>
<td>3-9</td>
<td>Continued-fraction pulse rate multiplier.</td>
<td>33</td>
</tr>
<tr>
<td>4-1</td>
<td>Model of the Sperry 6-GAMS configuration.</td>
<td>36</td>
</tr>
<tr>
<td>4-2</td>
<td>CMG configuration reference system</td>
<td>36</td>
</tr>
<tr>
<td>4-3</td>
<td>Dodecahedron axis system.</td>
<td>41</td>
</tr>
<tr>
<td>4-4</td>
<td>Three-variate control law.</td>
<td>47</td>
</tr>
<tr>
<td>4-5</td>
<td>Six-variate control law.</td>
<td>47</td>
</tr>
<tr>
<td>4-6</td>
<td>CMG control system block diagram.</td>
<td>50</td>
</tr>
<tr>
<td>4-7</td>
<td>Method of failure detection.</td>
<td>50</td>
</tr>
<tr>
<td>4-8</td>
<td>Organization of the pseudo-inverse steering law computer.</td>
<td>53</td>
</tr>
<tr>
<td>4-9</td>
<td>Basic incremental computer elements.</td>
<td>60</td>
</tr>
<tr>
<td>4-10</td>
<td>Incremental multiplier.</td>
<td>60</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>----------------------------</td>
<td>------</td>
</tr>
<tr>
<td>4-11</td>
<td>ΔA computer.</td>
<td>61</td>
</tr>
<tr>
<td>4-12</td>
<td>ΔB computer.</td>
<td>67</td>
</tr>
<tr>
<td>4-13</td>
<td>ΔC computer.</td>
<td>71</td>
</tr>
<tr>
<td>4-14</td>
<td>ød₀ computer.</td>
<td>73</td>
</tr>
<tr>
<td>4-15</td>
<td>ΔD computer.</td>
<td>74</td>
</tr>
<tr>
<td>4-16</td>
<td>Δu computer.</td>
<td>77</td>
</tr>
<tr>
<td>4-17</td>
<td>Δξₖ computer.</td>
<td>79</td>
</tr>
<tr>
<td>5-1</td>
<td>Plot of error in repeatability.</td>
<td>89</td>
</tr>
<tr>
<td>5-2</td>
<td>Flow charts of simulation program.</td>
<td>91</td>
</tr>
<tr>
<td>5-3</td>
<td>Card input format.</td>
<td>96</td>
</tr>
<tr>
<td>Table</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>-------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>2.1</td>
<td>Direction Cosine Orthogonality Conditions</td>
<td>6</td>
</tr>
<tr>
<td>2.2</td>
<td>Direction Cosine Covering by Orthogonality Equations</td>
<td>7</td>
</tr>
<tr>
<td>3.1</td>
<td>Twenty Five Required Data-Word Matrix</td>
<td>12</td>
</tr>
<tr>
<td>3.2</td>
<td>Status Matrices</td>
<td>15</td>
</tr>
<tr>
<td>3.3</td>
<td>Failure Detection Parity Equations</td>
<td>20</td>
</tr>
<tr>
<td>3.4</td>
<td>Table for Substitution in General 1-Gyro Restoring Equation</td>
<td>21</td>
</tr>
<tr>
<td>3.5</td>
<td>Table of Binary Constants for Dodecahedron Gyro Correction</td>
<td>23</td>
</tr>
<tr>
<td>3.6</td>
<td>$E_j$ Subset for $\lambda_i$</td>
<td>28</td>
</tr>
<tr>
<td>3.7</td>
<td>Comparison of Resolution of Resolver and Continued Fraction Rate Multipliers.</td>
<td>34</td>
</tr>
<tr>
<td>5.1</td>
<td>SIMU Simulation Data</td>
<td>90</td>
</tr>
<tr>
<td>5.2</td>
<td>Results of SIMU Simulation Program for Checkout Runs</td>
<td>97</td>
</tr>
<tr>
<td>5.3</td>
<td>Oscillatory Runs - Ten Cycles 0 to 1024 Pulses and Return to 0 Again</td>
<td>115</td>
</tr>
<tr>
<td>5.4</td>
<td>Program Listing</td>
<td>122</td>
</tr>
</tbody>
</table>
CHAPTER I
INTRODUCTION

The text which follows constitutes the final report on work performed at the Sperry Rand Research Center (SRRC) under contract NAS 9-9705 with the NASA Manned Spacecraft Center, Houston, Texas. The objectives of this program were twofold. First, it was to consider optimum organization and component technology for a strapdown transformation computer based upon the assumption of gyroscope and accelerometer inputs denoting components of angular and velocity change measured about an orthogonal axis triad. Second, it was to treat optimum methods of implementing a reliable attitude reference or control system when the gyroscope and accelerometer inputs were redundant and configured about the six normals to faces of a dodecahedron.

The work which is reported here divides into four major activities, each corresponding to a chapter. These will be described briefly. The first part of the activity, described in Chapter 2, is a study of means whereby a strapdown computer such as the one constructed for laboratory use under a previous contract may best be organized and implemented in reliable, flight-ready form. The second activity, covered in Chapter 3, is an investigation of optimum means for performing attitude reference computations when redundant dodecahedron sensor arrays are utilized. The third part of the study, described in Chapter 4, is a study of an attitude control system employing redundant control moment gyros as actuators. This part of the work was performed at the Sperry Flight Systems Division, under subcontract to SRRC. The final part of the study, Chapter 5, contains the results of further computer simulations of the truncation-error-free direction-cosine computation used in SIMU, methods and results are described.
CHAPTER 2
CONFIGURATION AND COMPONENT TECHNOLOGY FOR A STRAPDOWN COMPUTER

In a previous contract a strapdown coordinate-conversion computer was designed and constructed which used a truncation-error-free algorithm to compute direction cosines relating a rotating-axis system to one which was inertially stabilized. The organization of this computer was based upon a building block concept in which most blocks were digital accumulators with overflow detection and ternary input gating. In the present study, where the goal is to determine optimum system organization and component technology for such a computer, several structures, including the accumulator block, were studied.

Before describing the system, the underlying reasons for the particular choice of implementation will be given. First of all, since discussion centers about a spaceborne computer, weight, volume and power must be minimized and reliability must be maximized. From the point of view of weight, volume and power, it is clear that minimalization is provided by incorporating as high a level of circuit integration as is feasible within the constraints of present day technology. Studies have shown that maximizing the level of integration also provides the greatest reliability, since circuit bonds and off-chip interconnections, which are the primary sources of failure, are minimized. At the present time, reasonable device yields are obtained for integrated circuit chips with dimensions of 120 mils on each side. Interline spacings of 0.1 mil are readily obtainable. For metal-oxide-silicon (MOS) circuitry, active device areas are approximately 1.0 mil square. Allowing for metallization patterns and bonding pads, this means that an active device count of 2000 to 5000 is reasonable for MOS integrated circuits. Four-phase or ratioless MOS circuitry allows transfer rates from 2 to 5 MHz to be realized, and provides the highest circuit density presently available. For this reason, the preferred mechanization of the strapdown computer would use four-phase MOS LSI.

As in most applications, cost is also of importance. Development and manufacturing costs for the strapdown computer can be reduced by limiting the number of different chip types required. For this reason and the reasons above, the approach which was followed in formulating the systems structures was one in which maximum use was made of the fewest possible different circuit types, while maintaining the highest level of integration consistent with present technology and while trying to minimize the number of interconnections between chips.

The block diagram assumed for the triad system is shown in Fig. 2-1. It is essentially that of the SIMU computer, except that sections intended primarily for laboratory use have been eliminated. A set of three signals from orthogonal triad gyroscope sensors provides the input to the direction cosine computation. These signals are assumed to be corrected for drift scale factor errors by equipment preceding the navigation computer. The gyro inputs are added in the buffer to triad rotational rate commands. These commands replace the earth-rate correction system in the SIMU computer and are assumed to be generated by the main guidance and navigation computer. They enter the strapdown computer referred to stable coordinates and are transformed to rotational coordinates by the "commanded rate transformation" unit. Triad strapdown accelerometer signals are transformed to stable coordinates by the "AV
Corrected gyro inputs

Gyro buffer

Direction cosine computation

Commanded rate transformation

Commanded rate

Direction cosines out

Stabilized ΔV out

Accelero-meter inputs

ΔV transformation

FIG. 2-1 Block diagram of SIMU computer.
transformation" unit, as in SIMU. System outputs are the stabilized AV's and the nine direc-
tion cosines.

Besides the configuration used in the SIMU computer, a new form has been developed for
system implementation. This will be described first, and then methods for increasing system
reliability will be discussed for both cases.

The new method of organization is very similar in form to SIMU except that 16-bit digital
resolvers are used for the direction cosine computation instead of the accumulators. A dia-
agram of one of the three identical direction cosine units is shown in Fig. 2-2. Each direc-
tion cosine resides in a triple of resolvers, the Y resolver storing the upper 16 bits, the
R resolver storing the middle 16 bits, and the E resolver storing the lower 16 bits.

The updating algorithm used is the original truncation-error-free method involving three
passes. The operations for a $\theta_x$ pulse are specified by the matrix equation

$$
\begin{align*}
W_x &= \begin{bmatrix}
1 & 0 & 0 & 1 & 0 & 0 \\
0 & 1-h^2 & 0 & 0 & 1 & 0 \h \\
0 & 0 & 1-h^2 & 0 & -h & 1 \\
\end{bmatrix}
\end{align*}
$$

(2.1)

Similar operations hold for $\theta_y$ and $\theta_z$.

Each of the resolvers is equipped with its own serial adder operating in a closed loop
with the storage register. Thus, simultaneous accumulation by all resolvers is allowable.

By judicious sequencing of operations, a three-to-one speedup in processing rate is obtained
over the originally proposed SIMU computer. This sequencing is best clarified by means of the
schedule shown in Fig. 2-3. Each term in the schedule signifies the updating of the contents
stored in the specified resolver by adding to it the contents of another resolver, as deter-
mined by the updating matrices. The superscripts indicate whether it is the x-, y-, or z-matrix
equation, and the priming of the superscripts specifies the right, middle, or left matrix,
respectively. Thus, $E_{1y}^2$ indicates the updating of resolver $E_1$ for the second pass of $\theta_y$
updating. This updating involves subtracting the contents of resolver $R_3$ from the contents
of resolver $E_1$ and storing the difference in $E_1$. It is clear that the updating for $x$, $y$
and $z$ pulses of any rank of resolvers (say, the E resolvers) requires nine word-times.
Since overlapping of cycles is allowable, a new set of gyro pulses may be entered every nine
word-times. If the clock used is the same as in the SIMU computer, a processing rate speedup
of two-to-one over SIMU is obtained. The speedup over the originally proposed system is three-
to-one. Besides the speedup in processing rate (or, conversely, the ability to use a lower
clock rate), another advantage of this form of implementation is that it provides greater uni-
formity of components throughout the system, since the resolvers used in the cosine computa-
tion are identical to those used to provide the coordinate transformations elsewhere in the
system. An appropriate circuit embodiment would be the realization of one or more resolvers
as a single integrated circuit. The approximate device count for a simplex realization of
such a circuit is 350 transistors, so that such an embodiment is quite feasible using MOS/LSI
technology.
FIG. 2-2 Resolver direction cosine unit.

FIG. 2-3 Sequencing of all-resolver cosine computer.
Several techniques have been investigated for increasing reliability over the "simplex" SIMU computer. These will be described, although no revolutionary developments have emerged. The first technique makes use of the orthogonality properties of the direction cosine matrix for self checking. If $C$ is the cosine matrix and $C^T$ is its transpose, then orthogonality implies that

$$C \cdot C^T = C^T \cdot C = I$$

(2.2)

where $I$ is the identity matrix. Carrying out the term-by-term multiplications, the twelve distinct equations in Table 2.1 result.

**TABLE 2.1**

<table>
<thead>
<tr>
<th>Eq. No.</th>
<th>Equation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$C_{11}^2 + C_{12}^2 + C_{13}^2 - 1 = 0$</td>
</tr>
<tr>
<td>2</td>
<td>$C_{11}C_{21} + C_{12}C_{22} + C_{13}C_{23} = 0$</td>
</tr>
<tr>
<td>3</td>
<td>$C_{11}C_{31} + C_{12}C_{32} + C_{13}C_{33} = 0$</td>
</tr>
<tr>
<td>4</td>
<td>$C_{21}^2 + C_{22}^2 + C_{23}^2 - 1 = 0$</td>
</tr>
<tr>
<td>5</td>
<td>$C_{21}C_{31} + C_{22}C_{32} + C_{23}C_{33} = 0$</td>
</tr>
<tr>
<td>6</td>
<td>$C_{31}^2 + C_{32}^2 + C_{33}^2 - 1 = 0$</td>
</tr>
<tr>
<td>7</td>
<td>$C_{11} + C_{21}^2 + C_{31}^2 - 1 = 0$</td>
</tr>
<tr>
<td>8</td>
<td>$C_{11}C_{12} + C_{21}C_{22} + C_{31}C_{32} = 0$</td>
</tr>
<tr>
<td>9</td>
<td>$C_{11}C_{13} + C_{21}C_{23} + C_{31}C_{33} = 0$</td>
</tr>
<tr>
<td>10</td>
<td>$C_{12}^2 + C_{22}^2 + C_{32}^2 - 1 = 0$</td>
</tr>
<tr>
<td>11</td>
<td>$C_{12}C_{13} + C_{22}C_{23} + C_{32}C_{33} = 0$</td>
</tr>
<tr>
<td>12</td>
<td>$C_{13}^2 + C_{23}^2 + C_{33}^2 - 1 = 0$</td>
</tr>
</tbody>
</table>

The failure of some subset of the equations in Table 2.1 to be satisfied may be used as a form of parity check to determine faulty computation of the direction cosines. For instance, an error in $C_{11}$ alone is indicated by lack of satisfaction of Eqs. (2.1) and (2.7). Similar criteria may be derived from the covering relationships expressed by Table 2.2. At "X" at
the intersection of a particular column and row indicates that the cosine corresponding to that row is covered by the equation corresponding to the column.

TABLE 2.2
Direction Cosine Covering by Orthogonality
Equations (see Fig. 2-1)

<table>
<thead>
<tr>
<th>Cosine</th>
<th>Equation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 2 3 4 5 6 7 8 9 10 11 12</td>
</tr>
<tr>
<td>$C_{11}$</td>
<td>X X X X X X</td>
</tr>
<tr>
<td>$C_{12}$</td>
<td>X X X X X X</td>
</tr>
<tr>
<td>$C_{13}$</td>
<td>X X X X X X</td>
</tr>
<tr>
<td>$C_{21}$</td>
<td>X X X X X X</td>
</tr>
<tr>
<td>$C_{22}$</td>
<td>X X X X X X</td>
</tr>
<tr>
<td>$C_{23}$</td>
<td>X X X X X X</td>
</tr>
<tr>
<td>$C_{31}$</td>
<td>X X X X X X</td>
</tr>
<tr>
<td>$C_{32}$</td>
<td>X X X X X X</td>
</tr>
<tr>
<td>$C_{33}$</td>
<td>X X X X X X</td>
</tr>
</tbody>
</table>

If a single bit error is introduced into one of the direction cosines, say $C_{11}$, the error will propagate to cosines $C_{12}$ and $C_{13}$ as a function of angular input increments received. No error, however, will propagate to the other cosines, since their computation "loops" do not intersect those of $C_{11}$. The dependence of $C_{12}$, for instance, upon $C_{11}$ may be expressed by the partial differential equation pair below:

\[
\frac{\partial C_{11}}{\partial z} = -C_{12} \tag{2.3}
\]

\[
\frac{\partial C_{12}}{\partial z} = C_{11} \tag{2.4}
\]

Consider the inversion of a single bit in the representation of $C_{11}$ and its effect upon $C_{12}$. The bit inversion will be equivalent to adding some quantity $S$ to $C_{11}$. Let the error in $C_{11}$ be $E_1$ and the error in $C_{12}$ be $E_2$. Then

\[
\frac{\partial C_{11}}{\partial z} + \frac{\partial E_1}{\partial z} = -C_{12} - E_2 \tag{2.5}
\]
\[
\frac{\partial C_{12}}{\partial \theta} + \frac{\partial E_2}{\partial \theta} = C_{11} + E_1 \tag{2.6}
\]

with initial values

\[E_1 = S\]
\[E_2 = 0\]

Subtracting left and right equivalents from this set, the resulting error equations are

\[\frac{\partial E_1}{\partial \theta} = -E_2 \tag{2.7}\]
\[\frac{\partial E_2}{\partial \theta} = E_1 \tag{2.8}\]

with the obvious solution

\[E_1 = S \cos \theta\] \tag{2.9}
\[E_2 = S \sin \theta\] \tag{2.10}

If the checking equations of Table 2.1 are solved at fixed intervals and involve the upper \( n \) bits of the cosines, then the maximum period between solutions such that a single bit error in one of the direction cosines can be detected before it affects the other cosines in its loop is a function of \( n \) and the maximum angular rate which the system can accommodate. If the worst case bit inversion (i.e., \( S=1 \)) is assumed, the maximum period between solutions is set by the inequality below, where \( \omega_{\text{max}} \) is the maximum angular velocity and \( T_{\text{max}} \) is the maximum period

\[T_{\text{max}} \leq \frac{1}{\omega_{\text{max}}} \sin^{-1} 2^{-n}. \tag{2.11}\]

After detecting the cosine which is in error, the problem of what to do about it still remains. If the angles were unchanging, it would be a simple matter to recompute the correct values of the cosines using the relationships of Table 2.1. This, however, is not generally the case. One possibility is to retain in storage at some point the last complete set of correct cosines computed before the error was detected. These values can be reinserted and the computations restarted. Although this method introduces an overall error if vehicle attitude has changed during the time since restart values were observed, the amount of error can be limited if the checking equations are run frequently enough. Another method which leads to no
error accumulation is to stop the direction cosine computations as soon as an error is detected and store the input angle increments in a buffer, maintaining the proper sequence among the three axes. The equations in Table 2.1 then can be used to restore the proper value to the faulty cosine, after which the stored input increments are utilized to update the cosine matrix to its proper value. This method will succeed only if the allowable updating rate of the cosine matrix is greater than the maximum allowable pulse input rate from the gyros.

In the event that the error is caused by permanent failure of one of the components within the direction cosine computer, more than one-time correction will be required. A possible solution to this situation is obtained by noting that the nine cosines are computed as three independent triads. If a spare triad computer is included and is equipped with the proper connections to allow its output to be switched to replace any of the three primary computer outputs, then upon detection of a failure in one of the three primary units the restart operation can involve the spare unit, after which it is switched into the system in place of the faulty element.

In the methods described above that make use of the orthogonality properties of the direction cosine matrix for self correction, the nature of the computations and their rate is such that corrections can be handled by the general purpose guidance computer associated with the system. It is possible, and also more straightforward, to include logical redundance within the direction cosine computer itself rather than using external means. In both the SIMU and all-resolver systems, numerous computational feedback loops are used. If it is desired that the addition of redundance will effect correction of transient errors as well as permanent component failures, it is advisable to add the redundance in such a way that error correction occurs within all feedback loops. One scheme for providing this sort of correction is shown in Fig. 2-4, which depicts a triply modular redundant resolver. Voting circuits are introduced in such a way that they intersect the tightest feedback paths and still provide correct outputs from all three of the resolver segments for the maximum number of component failures within the segment.

It is difficult to supply all-inclusive arguments for the level at which redundance should be introduced in the strapdown computer, but since the resolver is the element which seems best suited for use as the basic building block, larger or smaller elements not providing the same lead-count and circuit-type minimality, and since the packaging of three interconnected resolvers with voting elements on a single substrate is presently within the state of the art, this configuration is recommended as the form to be used for implementation of the strapdown computer.

REFERENCES

FIG. 2-4 TMR resolver.
CHAPTER 3
CONVERSION METHODS FOR TRANSFORMATION FROM DODECAHEDRON-TO-TRIAD AXES

3.1 DIRECT REALIZATION BY USE OF PSEUDO-INVERSE AND STATUS MATRICES

3.1.1 Theoretical Analysis of Approach

In this part of the report we describe a method of transforming the six-vector

\[
\mathbf{m} = \begin{pmatrix}
m_a \\
m_b \\
m_c \\
m_d \\
m_e \\
m_f
\end{pmatrix}
\]  

of the dodecahedron reference system into the three-vector

\[
\mathbf{b} = \begin{pmatrix}
b_x \\
b_y \\
b_z
\end{pmatrix}
\]  

of the orthogonal triad reference system.

The geometric relationship between the two systems is given by

\[
\mathbf{m} = \mathbf{Hb}
\]  

where

\[
\mathbf{H} = \begin{pmatrix}
S & 0 & C \\
-S & 0 & C \\
C & S & 0 \\
C & -S & 0 \\
0 & C & S \\
0 & C & -S
\end{pmatrix}
\]  

\[
S = \sin \alpha = \frac{\sqrt{5 - \sqrt{5}}}{10}
\]  

\[
C = \cos \alpha = \frac{\sqrt{5 + \sqrt{5}}}{10}
\]
Since every $3 \times 3$ submatrix of $H$ is nonsingular, any three out of the six equations of (3.3) can be used to determine the $b$ components in terms of the corresponding three components of $m$. The nature of the problem, however, is such that the instruments providing the components of $m$ are subject to errors and, as was pointed out by Gilmore, the solution of Eq. (3.3) for $b$ which will best fit the geometric configuration of the well functioning instruments is given by

$$ b = (H^T \lambda H)^{-1} H^T \lambda m $$

(3.7)

where $\lambda = \text{diag}(\lambda_a, \lambda_b, \lambda_c, \lambda_d, \lambda_e, \lambda_f)$ is the "status" matrix, with $\lambda_i = 1$ or 0 according to whether instrument $i$, $i=a,b,c,d,e,f$, is error free or not.

Of course, for this solution to be meaningful, at least three out of the six instruments has to be error free. There are 42 different combinations with at least three of the $\lambda_i$ being equal to 1. After having evaluated explicitly the matrix

$$ H^I = (H^T \lambda H)^{-1} H^T \lambda $$

(3.8)

for all the proper 42 combinations of the $\lambda_i$, it becomes evident that the total number of different nonzero entries of $H^I$ for the various combinations is only 25. In terms of $S$ and $C$, these 25 entries are as shown in Table 3.1.

**TABLE 3.1**

Twenty Five Required Data-Word Matrix

<table>
<thead>
<tr>
<th>$N_1$</th>
<th>$N_{11}$</th>
<th>$N_{21}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$1/10(2S + C)$</td>
<td>$1/4(S + 4C)$</td>
<td>$2S + C$</td>
</tr>
<tr>
<td>$N_2$</td>
<td>$N_{12}$</td>
<td>$N_{22}$</td>
</tr>
<tr>
<td>$1/10(7S + C)$</td>
<td>$1/4(3S + C)$</td>
<td>$3S + C$</td>
</tr>
<tr>
<td>$N_3$</td>
<td>$N_{13}$</td>
<td>$N_{23}$</td>
</tr>
<tr>
<td>$1/5(S + 3C)$</td>
<td>$3/4 S$</td>
<td>$3S + 3C$</td>
</tr>
<tr>
<td>$N_4$</td>
<td>$N_{14}$</td>
<td>$N_{24}$</td>
</tr>
<tr>
<td>$1/10(-S + 7C)$</td>
<td>$3/4 C$</td>
<td>$S - C$</td>
</tr>
<tr>
<td>$N_5$</td>
<td>$N_{15}$</td>
<td>$N_{25}$</td>
</tr>
<tr>
<td>$1/10(-S + 2C)$</td>
<td>$1/4(-S + 3C)$</td>
<td>$-S + 3C$</td>
</tr>
<tr>
<td>$N_6$</td>
<td>$N_{16}$</td>
<td>$N_{26}$</td>
</tr>
<tr>
<td>$1/5(3S - C)$</td>
<td>$1/4(2S - C)$</td>
<td>$3S - C$</td>
</tr>
<tr>
<td>$N_7$</td>
<td>$N_{17}$</td>
<td>$N_{27}$</td>
</tr>
<tr>
<td>$1/10(2S - C)$</td>
<td>$1/4(S + 2C)$</td>
<td>$S + 2C$</td>
</tr>
<tr>
<td>$N_8$</td>
<td>$N_{18}$</td>
<td>$N_{28}$</td>
</tr>
<tr>
<td>$1/2 S$</td>
<td>$1/2(-S + 2C)$</td>
<td>$-S + 2C$</td>
</tr>
<tr>
<td>$N_9$</td>
<td>$N_{19}$</td>
<td>$N_{29}$</td>
</tr>
<tr>
<td>$1/2 C$</td>
<td>$1/2(2S + C)$</td>
<td>$2S + C$</td>
</tr>
<tr>
<td>$N_{10}$</td>
<td>$N_{20}$</td>
<td></td>
</tr>
<tr>
<td>$1/4(4S - C)$</td>
<td>$-S + 2C$</td>
<td></td>
</tr>
</tbody>
</table>
The 42 explicit evaluations of $H^T$ are given in Table 3.2 in terms of the indices of the above listed $N_i$'s. For instance, the $H^T$ matrix for the combination $\lambda_a=\lambda_b=\lambda_e=1$ and $\lambda_c=\lambda_d=\lambda_f=0$, which is given by

$$H^T = \begin{bmatrix} N_{19} & -N_{19} & 0 & 0 & 0 & 0 \\ -N_{22} & -N_{22} & 0 & 0 & N_{20} & 0 \\ N_{18} & N_{18} & 0 & 0 & 0 & 0 \end{bmatrix}.$$  

In Table 3.2 $0$ stands for the number zero and the numbers shown represent the $N_i$ terms by listing only the subscript number ($\pm i$), e.g., $N_{12}=12$. For the expression in Eq. (3.9), this yields

$$\lambda_a \lambda_b \lambda_e = 1$$  

$$= \begin{bmatrix} 19 & -19 & 0 & 0 & 0 & 0 \\ -22 & -22 & 0 & 0 & 20 & 0 \\ 18 & 18 & 0 & 0 & 0 & 0 \end{bmatrix}.$$  

3.1.2 Implementation of Pseudo-Inverse and Status Matrices Method Using LSI Packaged Basic Elements

The implementation of the transformation described in 3.1.1 essentially involves the use of a read-only memory and an arithmetic unit to provide the appropriate overflows to the 3-axis system's input buffer logic.

The following discussion will treat each section of the system in enough detail to make clear the functional interaction of the sections in achieving the required final results. There are four sections of the system: they are (1) the six input gyro-sync logic, (2) the read-only memory, (3) the arithmetic unit, and (4) the direction cosine computer's (SIMU) input buffer logic.

The gyro-sync logic performs two operations. First, the six ($\pm x$, $\pm y$, $\pm z$) asynchronous gyro inputs are synchronized to computer word time. Second, these six signals are encoded so as to be sent out as three octal variables (see Fig. 3-1).

The read-only memory's address selection is a function of two sets of input variables. These are encoded gyro outputs described above and the set of variables ($\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5$ and $\lambda_6$) that indicate which gyros are operating satisfactorily at any given time. These nine selection lines are used to select one of 252 memory locations. In each of these memory locations is an address of a second read-only memory. The second read-only memory contains 25 sixteen-bit words.

To understand the need for this memory configuration refer to Table 3.2, which shows the manner in which the basic constants shown in Table 3.1 are used. The entries in Table 3.1 are a summary of the various products of a constant times $\frac{1}{2} \sin \alpha$ and $\frac{1}{2} \cos \alpha$ as discussed in Sec. 3.1.1.
FIG. 3-1 Incremental input encoding for the six-gyro system.
### TABLE 3.2
Status Matrices

<table>
<thead>
<tr>
<th>( \lambda_a, \lambda_b, \lambda_c = 1 )</th>
<th>( \lambda_a, \lambda_c, \lambda_f = 1 )</th>
</tr>
</thead>
</table>
| \[
\begin{bmatrix}
19 & -19 & 0 & 0 & 0 & 0 \\
23 & 23 & 21 & 0 & 0 & 0 \\
18 & 18 & 0 & 0 & 0 & 0
\end{bmatrix}
\] | \[
\begin{bmatrix}
21 & 0 & 0 & 0 & -23 & 23 \\
0 & 0 & 0 & 0 & 18 & 18 \\
0 & 0 & 0 & 0 & 19 & -19
\end{bmatrix}
\] |

<table>
<thead>
<tr>
<th>( \lambda_a, \lambda_b, \lambda_d = 1 )</th>
<th>( \lambda_a, \lambda_c, \lambda_d = 1 )</th>
</tr>
</thead>
</table>
| \[
\begin{bmatrix}
19 & -19 & 0 & 0 & 0 & 0 \\
23 & -23 & 0 & -21 & 0 & 0 \\
18 & 18 & 0 & 0 & 0 & 0
\end{bmatrix}
\] | \[
\begin{bmatrix}
0 & 0 & 18 & 18 & 0 & 0 \\
0 & 0 & 19 & -19 & 0 & 0 \\
0 & 20 & 22 & 22 & 0 & 0
\end{bmatrix}
\] |

<table>
<thead>
<tr>
<th>( \lambda_a, \lambda_b, \lambda_e = 1 )</th>
<th>( \lambda_b, \lambda_c, \lambda_e = 1 )</th>
</tr>
</thead>
</table>
| \[
\begin{bmatrix}
19 & -19 & 0 & 0 & 0 & 0 \\
-22 & -22 & 0 & 0 & 20 & 0 \\
18 & 18 & 0 & 0 & 0 & 0
\end{bmatrix}
\] | \[
\begin{bmatrix}
0 & 18 & 23 & 0 & -19 & 0 \\
0 & -19 & -18 & 0 & 23 & 0 \\
0 & 23 & 19 & 0 & -18 & 0
\end{bmatrix}
\] |

<table>
<thead>
<tr>
<th>( \lambda_a, \lambda_c, \lambda_f = 1 )</th>
<th>( \lambda_b, \lambda_c, \lambda_f = 1 )</th>
</tr>
</thead>
</table>
| \[
\begin{bmatrix}
0 & 0 & 18 & 18 & 0 & 0 \\
n & 0 & 19 & -19 & 0 & 0 \\
20 & 0 & -22 & -22 & 0 & 0
\end{bmatrix}
\] | \[
\begin{bmatrix}
0 & 18 & 22 & 0 & 0 & 19 \\
0 & 19 & 18 & 0 & 0 & 22 \\
0 & 19 & 18 & 0 & 0 & 22
\end{bmatrix}
\] |

<table>
<thead>
<tr>
<th>( \lambda_a, \lambda_c, \lambda_d = 1 )</th>
<th>( \lambda_b, \lambda_d, \lambda_e = 1 )</th>
</tr>
</thead>
</table>
| \[
\begin{bmatrix}
22 & 0 & 19 & 0 & -18 & 0 \\
-18 & 0 & 22 & 0 & 19 & 0 \\
19 & 0 & -18 & 0 & 22 & 0
\end{bmatrix}
\] | \[
\begin{bmatrix}
0 & 18 & 0 & 23 & 0 & 19 \\
0 & 19 & 0 & 18 & 0 & 23 \\
0 & 23 & 0 & 19 & 0 & 18
\end{bmatrix}
\] |

<table>
<thead>
<tr>
<th>( \lambda_a, \lambda_c, \lambda_f = 1 )</th>
<th>( \lambda_b, \lambda_c, \lambda_e = 1 )</th>
</tr>
</thead>
</table>
| \[
\begin{bmatrix}
-18 & 0 & 23 & 0 & 0 & -19 \\
19 & 0 & -18 & 0 & 23 & 0 \\
23 & 0 & -19 & 0 & 0 & 18
\end{bmatrix}
\] | \[
\begin{bmatrix}
0 & -21 & 0 & 0 & 23 & -23 \\
0 & 0 & 0 & 0 & 18 & 18 \\
0 & 0 & 0 & 0 & 19 & -19
\end{bmatrix}
\] |

<table>
<thead>
<tr>
<th>( \lambda_a, \lambda_d, \lambda_e = 1 )</th>
<th>( \lambda_c, \lambda_d, \lambda_e = 1 )</th>
</tr>
</thead>
</table>
| \[
\begin{bmatrix}
-18 & 0 & 0 & 23 & 19 & 0 \\
-19 & 0 & 0 & 23 & 19 & 0 \\
23 & 0 & 0 & -19 & -18 & 0
\end{bmatrix}
\] | \[
\begin{bmatrix}
0 & 0 & 18 & 18 & 0 & 0 \\
0 & 0 & 19 & -19 & 0 & 0 \\
0 & 0 & -23 & 23 & 21 & 0
\end{bmatrix}
\] |

<table>
<thead>
<tr>
<th>( \lambda_a, \lambda_d, \lambda_f = 1 )</th>
<th>( \lambda_c, \lambda_d, \lambda_f = 1 )</th>
</tr>
</thead>
</table>
| \[
\begin{bmatrix}
22 & 0 & 0 & 19 & 0 & 18 \\
18 & 0 & 0 & -22 & 0 & 19 \\
-19 & 0 & 0 & -10 & 0 & -22
\end{bmatrix}
\] | \[
\begin{bmatrix}
0 & 0 & 18 & 18 & 0 & 0 \\
0 & 0 & 19 & -19 & 0 & 0 \\
0 & 0 & 23 & -23 & 0 & -21
\end{bmatrix}
\] |
<table>
<thead>
<tr>
<th>Table 3.2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Status Matrices (cont.)</strong></td>
</tr>
</tbody>
</table>

\[
\begin{pmatrix}
  0 & 0 & 20 & 0 & -22 & -22 \\
  0 & 0 & 0 & 0 & 18 & 18 \\
  0 & 0 & 0 & 0 & 19 & -19
\end{pmatrix}
\begin{pmatrix}
  13 & -12 & 17 & 0 & 0 & -25 \\
  16 & 17 & 10 & 0 & 0 & 11 \\
  15 & 14 & -24 & 0 & 0 & -26
\end{pmatrix}
\]

\[
\begin{pmatrix}
  0 & 0 & 20 & 22 & 22 \\
  0 & 0 & 0 & 0 & 18 & 18 \\
  0 & 0 & 0 & 0 & 19 & -19
\end{pmatrix}
\begin{pmatrix}
  12 & -13 & 0 & 17 & 0 & 25 \\
  17 & 16 & 0 & -10 & 0 & 11 \\
  14 & 15 & 0 & 24 & 0 & -16
\end{pmatrix}
\]

\[
\begin{pmatrix}
  8 & -8 & 9 & 0 & 0 & 0 \\
  0 & 0 & 19 & -19 & 0 & 0 \\
  18 & 18 & 0 & 0 & 0 & 0
\end{pmatrix}
\begin{pmatrix}
  16 & 0 & 15 & 14 & 0 & -24 \\
  25 & 0 & 13 & -12 & 0 & 17 \\
  11 & 0 & -16 & -17 & 0 & -10
\end{pmatrix}
\]

\[
\begin{pmatrix}
  19 & -19 & 0 & 0 & 0 & 0 \\
  0 & 0 & 0 & 0 & 18 & 18 \\
  9 & 9 & 0 & 0 & 8 & -8
\end{pmatrix}
\begin{pmatrix}
  0 & -16 & 14 & 15 & 0 & 24 \\
  0 & 25 & 12 & -13 & 0 & 17 \\
  0 & 11 & 17 & 16 & 0 & -10
\end{pmatrix}
\]

\[
\begin{pmatrix}
  0 & 0 & 18 & 18 & 0 & 0 \\
  0 & 0 & 8 & -8 & 9 & 9 \\
  0 & 0 & 0 & 0 & 19 & -19
\end{pmatrix}
\begin{pmatrix}
  10 & 0 & 11 & 0 & -17 & -16 \\
  24 & 0 & 16 & 0 & 14 & 15 \\
  17 & 0 & -25 & 0 & 12 & -13
\end{pmatrix}
\]

\[
\begin{pmatrix}
  12 & -13 & 17 & 0 & -25 & 0 \\
  -17 & -16 & 10 & 0 & 11 & 0 \\
  14 & 15 & 24 & 0 & 16 & 0
\end{pmatrix}
\begin{pmatrix}
  0 & -10 & 11 & 0 & -16 & -17 \\
  0 & -24 & 16 & 0 & 15 & 14 \\
  0 & 17 & 25 & 0 & 13 & -12
\end{pmatrix}
\]

\[
\begin{pmatrix}
  13 & -12 & 0 & 17 & 25 & 0 \\
  -16 & -17 & 0 & -10 & 11 & 0 \\
  15 & 14 & 0 & -24 & 16 & 0
\end{pmatrix}
\begin{pmatrix}
  19 & 0 & 0 & 11 & 16 & 17 \\
  -24 & 0 & 0 & -16 & 15 & 14 \\
  17 & 0 & 0 & -25 & 13 & -12
\end{pmatrix}
\]

\[
\begin{pmatrix}
  16 & 0 & 14 & 15 & 24 & 0 \\
  -25 & 0 & 12 & -13 & 17 & 0 \\
  11 & 0 & -17 & -16 & 10 & 0
\end{pmatrix}
\begin{pmatrix}
  0 & -10 & 0 & 11 & 17 & 16 \\
  0 & 24 & 0 & -16 & 14 & 15 \\
  0 & 17 & 0 & 25 & 12 & -13
\end{pmatrix}
\]

\[
\begin{pmatrix}
  0 & -16 & 15 & 14 & -24 & 0 \\
  0 & -25 & 13 & -12 & 17 & 0 \\
  0 & 11 & 16 & 17 & 10 & 0
\end{pmatrix}
\begin{pmatrix}
  8 & -8 & 9 & 9 & 0 & 0 \\
  -1 & -1 & 2 & -2 & 3 & 0 \\
  4 & 4 & 5 & -5 & 6 & 0
\end{pmatrix}
\]
TABLE 3.2
Status Matrices (cont.)

\[ \lambda_e = 0 \begin{bmatrix} 8 & -8 & 9 & 9 & 0 & 0 \\ 1 & 1 & 2 & -2 & 0 & 3 \\ 4 & 4 & -5 & 5 & 0 & -6 \end{bmatrix} \]

\[ \lambda_d = 0 \begin{bmatrix} 2 & -2 & 3 & 0 & -7 & -7 \\ 5 & -5 & 6 & 0 & 4 & 4 \\ 9 & 9 & 0 & 0 & 8 & -8 \end{bmatrix} \]

\[ \lambda_c = 0 \begin{bmatrix} 2 & -2 & 0 & 3 & 7 & 7 \\ -5 & 5 & 0 & -5 & 4 & 4 \\ 9 & 9 & 0 & 0 & 8 & -8 \end{bmatrix} \]

\[ \lambda_a = 0 \begin{bmatrix} 6 & 0 & 4 & 4 & 5 & -5 \\ 0 & 0 & 8 & -8 & 9 & 9 \\ 0 & 3 & 1 & 1 & 2 & -2 \end{bmatrix} \]

\[ \lambda_b = 0 \begin{bmatrix} 2 & -2 & 3 & 0 & -7 & -7 \\ 5 & -5 & 6 & 0 & 4 & 4 \\ 9 & 9 & 0 & 0 & 8 & -8 \end{bmatrix} \]

\[ \lambda_a, \lambda_b, \lambda_c, \lambda_d, \lambda_e = 1 \begin{bmatrix} 8 & -8 & 9 & 9 & 0 & 0 \\ 0 & 0 & 8 & -8 & 9 & 9 \\ 0 & 3 & 1 & 1 & 2 & -2 \end{bmatrix} \]

In operation, the first read-only memory determines which matrix in Table 3.2 will be required to process the sensors which are operable (\( \lambda_i = 1 \)). The second memory selects the words specified by this matrix. These words are stored as in Table 3.1. They are processed with the appropriate timing to allow each of the triad axes to be updated for any gyro outputs that have occurred. These words are used in 42 different sequences in performing the transformations in the arithmetic unit. There are three sets of memories as described above for processing from six to three axes, one for each of the triad axes.

The arithmetic unit consists of an adder and five data registers used in parallel processing to transform to \( \Delta \theta_x \), \( \Delta \theta_y \) and \( \Delta \theta_z \) (see Fig. 3-2). The sequence of events is as follows. Starting the updating to obtain a \( \Delta \theta_x \) input to the computer, the first specified word from the second read-only memory (ROM) (of the set of six words specified by the first ROM for the x-axis update) is loaded into the constant buffer. From the buffer the scaled increment is added to the previous remainder of \( \Delta \theta_x \) and the result is held in the general \( \Delta \theta \) buffer register. Once this is done the selection of the next constant for the x-axis update from the ROM can begin. The \( \Delta \theta_x \) register is loaded again to prepare for the next component of the \( \Delta \theta_x \) update. After the six cycles are completed the remainder will be in the \( \Delta \theta_x \) remainder register and an output will be sent to the gyro-buffer logic if the entire update produced either a positive or negative overflow.

The gyro-buffer will recognize this overflow as the signal to produce a positive or negative output pulse that will update the appropriate direction cosines. This sequence will be followed by the second set of ROM's and its arithmetic register to provide a \( \Delta \theta_y \) input to the direction cosine matrix, if the inputs are of sufficient magnitude to produce an overflow. The same sequence will take place for the final set of ROM's to produce a \( \Delta \theta_z \) input pulse when required.
The above constitutes a complete update cycle and takes place within the system update
time of 57.6 μs. The computer described above can function continuously at any input rate
desired that does not exceed this pulse repetition rate.

3.2 CONVERSION FROM DODECAHEDRON TO TRIAD BY SENSOR CORRECTION

3.2.1 Description

In this section a method for converting sensor signals from redundant dodecahedron to
equivalent triad axes is described in which signals missing because of sensor failure are
restored by synthesis from operational sensor outputs. The resulting six inertial signals
are then used as the input to a fixed transformation whose output is the desired equivalent
triad set. A block diagram of the system is shown in Fig. 3-3. It consists of a sensor
failure detection unit, a correction unit, and a dodecahedron-to-triad conversion unit. The
detection unit is the parity equation integrator, which will be described in Sec. 3.2.2. The
detection unit produces as its output a six-element binary vector, \( \lambda \), each of whose elements,
when equal to unity, affirms the operation of one of the sensors. The \( \lambda \) vector and the sen­sor outputs serve as inputs to the sensor correction unit. If a particular sensor is opera­tive, the output of the correction unit corresponding to that sensor is the sensor signal
itself. If a sensor has failed, then the correction unit output corresponding to that sensor
is an estimate of what the sensor signal should be, derived from the operative sensor signals
It is clear that at least three sensors must be operative in order that such an estimate can
be made.

The six sensor signals (synthetic or actual) from the correction unit are used as inputs
to the dodecahedron-to-triad conversion unit. Since six sensor signals are always assumed to
be present at the input of this unit, even when as many as three sensors have actually failed,
a simple, fixed conversion algorithm may be used. The techniques employed and the system
itself will now be described in greater detail.

The method used for synthesizing failed-sensor signals is based upon solution of one of
the parity equations for the missing signal in terms of the three valid signals. The parity
equations, as defined by Gilmore, \(^1\) are reproduced here as Table 3.3. As an example, if sensor
A has failed but sensors B, C, and D are still operating, then Eq. (1) in Table 3.3 may
be used for estimating \( A \), viz,

\[
A = B + \gamma (C + D)
\]  

(3.11)

where

\[
\gamma = \frac{\sin(\alpha)}{\cos(\alpha)} = \frac{\sqrt{5} - 1}{2} = 0.618033
\]  

(3.12)

\( A \) is the estimate of the output that would be obtained from sensor \( A \) if it were operating
and \( \alpha \) is the angle between two intersecting normals to adjacent faces of a dodecahedron. A
general equation may be written for sensor correction:
FIG. 3-2 Block diagram of six-gyro system using pseudo-inverse and status matrices method.

FIG. 3-3 Block diagram of sensor correction method.
\[ \hat{I} = \lambda_1 1 + \lambda_2 1 \left( \lambda_2 \lambda_3 \lambda_4 \lambda_5 e_c + \lambda_2 \lambda_3 \lambda_4 \lambda_5 E_a \right) + \lambda_2 \lambda_3 \lambda_4 \lambda_5 E_B + \lambda_2 \lambda_3 \lambda_4 \lambda_5 E_C + \lambda_2 \lambda_3 \lambda_4 \lambda_5 E_D + \lambda_2 \lambda_3 \lambda_4 \lambda_5 E_E + \lambda_2 \lambda_3 \lambda_4 \lambda_5 E_F + \lambda_2 \lambda_3 \lambda_4 \lambda_5 E_G + \lambda_2 \lambda_3 \lambda_4 \lambda_5 E_H \]  

(3.13)

\[ c = \cos(\alpha) = \left( \frac{\sqrt{5} + 5}{10} \right)^{1/2} \approx 0.605065 \]

\[ s = \sin(\alpha) = \left( \frac{5 - \sqrt{5}}{10} \right)^{1/2} \approx 0.32574 \]

**TABLE 3.3**

Failure Detection Parity Equations

<table>
<thead>
<tr>
<th>No.</th>
<th>Instrument</th>
<th>Equation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ABCD</td>
<td>((A - B)c - (C + D)s = 0)</td>
</tr>
<tr>
<td>2</td>
<td>ABCE</td>
<td>((B + C)c - (A + E)s = 0)</td>
</tr>
<tr>
<td>3</td>
<td>ABCF</td>
<td>((C - A)c + (B - F)s = 0)</td>
</tr>
<tr>
<td>4</td>
<td>ABDE</td>
<td>((D - A)c + (B + E)s = 0)</td>
</tr>
<tr>
<td>5</td>
<td>ABDF</td>
<td>((B - D)c - (A - F)s = 0)</td>
</tr>
<tr>
<td>6</td>
<td>ABCE</td>
<td>((E - F)c - (A + B)s = 0)</td>
</tr>
<tr>
<td>7</td>
<td>ACDE</td>
<td>((D + E)c - (A - C)s = 0)</td>
</tr>
<tr>
<td>8</td>
<td>ACDF</td>
<td>((E - C)c - (A + D)s = 0)</td>
</tr>
<tr>
<td>9</td>
<td>ACEF</td>
<td>((A + F)c - (C + E)s = 0)</td>
</tr>
<tr>
<td>10</td>
<td>ADEF</td>
<td>((E - A)c + (D - F)s = 0)</td>
</tr>
<tr>
<td>11</td>
<td>BCDE</td>
<td>((E - C)c + (D - B)s = 0)</td>
</tr>
<tr>
<td>12</td>
<td>BCFD</td>
<td>((F + D)c + (B - C)s = 0)</td>
</tr>
<tr>
<td>13</td>
<td>BCEF</td>
<td>((B - E)c + (C + F)s = 0)</td>
</tr>
<tr>
<td>14</td>
<td>BDEF</td>
<td>((B + F)c + (D - E)s = 0)</td>
</tr>
<tr>
<td>15</td>
<td>CDEF</td>
<td>((C - D)c - (E + F)s = 0)</td>
</tr>
</tbody>
</table>

To interpret Eq. (3.13) let \( \hat{I} \) represent the output of sensor 1 and let \( \hat{I} \) represent the estimate of this quantity. The various \( \lambda \)'s are the operational indicators of their respective sensors or, when barred, the complements of these quantities. Finally, the quantities \( E_A, E_B, \) etc., represent the parity equations to be used in computing \( \hat{I} \). Thus, Eq. (3.13) serves to select which (if any) equation will be used to synthesize the sensor output, based upon the operational status of all sensors in the system.
Equation (3.13) can be made specific to any particular sensor by means of Table 3.4, which indicates the substitution of subscripts for correction of a given sensor. The synthesis of a failed-sensor output from one of the parity equations involves multiplying certain of the operational outputs by either unity, \( \gamma \) or \( \gamma^{-1} = 1 - \gamma \) and then summing these quantities. The selection of quantities to be summed to produce the desired synthetic sensor output is governed by the \( \lambda \) vector. A resolver implementation of sensor output synthesis is shown in Fig 3-4 for the general case. Six such resolvers are required, one for each dodecahedron axis. The resolver signals \( S_i \), either multiplied by \( \gamma \) or by unity, are gated into a resolver by sequencing signals \( T_{ij} \) and control signals \( C_{ij} \). The overflow output of this resolver is the dodecahedron sensor signal, either synthetic or real, to be used in the dodecahedron-to-triad conversion. The control signals \( C_{ij} \) are derived from Eq. (3.13) as functions of the \( \lambda \) vector and are given in general form by Table 3.5. Table 3.5 is made specific to a particular sensor by means of Table 3.4. Implementation of the control functions is by means of a read-only memory whose inputs are the five \( \lambda_i \) and whose outputs are the \( C_{ij} \). This memory has a capacity of 16 words, each 15 bits long. Six identical memories are used, one for each of the dodecahedron sensors. Each memory is made specific to an axis by the combination and permutation of \( \lambda_i \) used as its input. These are specified by Table 3.4.

**TABLE 3.4**

Table for Substitution in General 1-Gyro Restoring Equation

<table>
<thead>
<tr>
<th>Sensor</th>
<th>( \lambda_1 )</th>
<th>( \lambda_2 )</th>
<th>( \lambda_3 )</th>
<th>( \lambda_4 )</th>
<th>( \lambda_5 )</th>
<th>( \lambda_6 )</th>
<th>( E_A )</th>
<th>( E_B )</th>
<th>( E_C )</th>
<th>( E_D )</th>
<th>( E_E )</th>
<th>( E_F )</th>
<th>( E_G )</th>
<th>( E_H )</th>
<th>( E_I )</th>
<th>( E_J )</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>( \lambda_A )</td>
<td>( \lambda_B )</td>
<td>( \lambda_C )</td>
<td>( \lambda_D )</td>
<td>( \lambda_E )</td>
<td>( \lambda_F )</td>
<td>( E_1 )</td>
<td>( E_2 )</td>
<td>( E_3 )</td>
<td>( E_4 )</td>
<td>( E_5 )</td>
<td>( E_6 )</td>
<td>( E_7 )</td>
<td>( E_8 )</td>
<td>( E_9 )</td>
<td>( E_{10} )</td>
</tr>
<tr>
<td>B</td>
<td>( \lambda_B )</td>
<td>( \lambda_A )</td>
<td>( \lambda_C )</td>
<td>( \lambda_D )</td>
<td>( \lambda_F )</td>
<td>( \lambda_E )</td>
<td>( E_1 )</td>
<td>( E_2 )</td>
<td>( E_3 )</td>
<td>( E_4 )</td>
<td>( E_5 )</td>
<td>( E_6 )</td>
<td>( E_{12} )</td>
<td>( E_{13} )</td>
<td>( E_{14} )</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>( \lambda_C )</td>
<td>( \lambda_D )</td>
<td>( \lambda_E )</td>
<td>( \lambda_F )</td>
<td>( \lambda_A )</td>
<td>( \lambda_B )</td>
<td>( E_{15} )</td>
<td>( E_7 )</td>
<td>( E_{11} )</td>
<td>( E_{12} )</td>
<td>( E_1 )</td>
<td>( E_9 )</td>
<td>( E_{13} )</td>
<td>( E_{12} )</td>
<td>( E_{11} )</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>( \lambda_D )</td>
<td>( \lambda_C )</td>
<td>( \lambda_E )</td>
<td>( \lambda_F )</td>
<td>( \lambda_B )</td>
<td>( \lambda_A )</td>
<td>( E_{15} )</td>
<td>( E_{11} )</td>
<td>( E_7 )</td>
<td>( E_{12} )</td>
<td>( E_8 )</td>
<td>( E_1 )</td>
<td>( E_{14} )</td>
<td>( E_{10} )</td>
<td>( E_4 )</td>
<td>( E_5 )</td>
</tr>
<tr>
<td>E</td>
<td>( \lambda_E )</td>
<td>( \lambda_F )</td>
<td>( \lambda_A )</td>
<td>( \lambda_B )</td>
<td>( \lambda_C )</td>
<td>( \lambda_D )</td>
<td>( E_6 )</td>
<td>( E_9 )</td>
<td>( E_{10} )</td>
<td>( E_{13} )</td>
<td>( E_{14} )</td>
<td>( E_{15} )</td>
<td>( E_2 )</td>
<td>( E_4 )</td>
<td>( E_7 )</td>
<td>( E_{11} )</td>
</tr>
<tr>
<td>F</td>
<td>( \lambda_F )</td>
<td>( \lambda_E )</td>
<td>( \lambda_A )</td>
<td>( \lambda_B )</td>
<td>( \lambda_D )</td>
<td>( \lambda_C )</td>
<td>( E_6 )</td>
<td>( E_{10} )</td>
<td>( E_9 )</td>
<td>( E_{14} )</td>
<td>( E_{13} )</td>
<td>( E_{15} )</td>
<td>( E_3 )</td>
<td>( E_8 )</td>
<td>( E_{12} )</td>
<td></td>
</tr>
</tbody>
</table>
FIG. 3-4 Dodecahedron gyro correction resolver (6 required).
### TABLE 3.5
Table of Binary Constants for Dodecahedron Gyro Correction (see Fig. 3-4)

<table>
<thead>
<tr>
<th>Eq.</th>
<th>( \lambda_2 \lambda_3 \lambda_4 \lambda_5 \lambda_6 )</th>
<th>( c_{21} c_{22} c_{31} c_{32} c_{41} c_{42} c_{51} c_{52} c_{61} c_{62} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>11100 1000 1 0 1 0 0 0 0 0 0 0 0</td>
<td>1 0 0 0 0 1 0 1 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>B</td>
<td>11010 100 1 0 1 0 0 0 0 1 0 0 0 0 0</td>
<td>1 0 1 0 0 0 0 0 1 0 0 0 0 0</td>
</tr>
<tr>
<td>C</td>
<td>11001 110 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1</td>
<td>0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1</td>
</tr>
<tr>
<td>D</td>
<td>10110 10 0 0 1 0 0 0 0 1 0 0 0 0</td>
<td>0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1</td>
</tr>
<tr>
<td>E</td>
<td>10101 1 0 1 0 0 0 1 0 1 0 0 0 1 0 0 0 0 0 0 0 1</td>
<td>1 0 1 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 1</td>
</tr>
<tr>
<td>F</td>
<td>10011 0 1 0 0 0 0 0 0 0 1 0 1 0 1 0 1 1 1</td>
<td>0 1 0 0 0 0 0 0 1 0 1 0 1 1 1</td>
</tr>
<tr>
<td>G</td>
<td>01110 1 0 0 0 1 0 0 0 1 0 1 1 0 1 0 0 0 0 0 0 0 1</td>
<td>0 0 0 0 0 1 0 0 0 1 0 1 1 0 1 0 0 0 0 0 0 1</td>
</tr>
<tr>
<td>H</td>
<td>01101 0 1 0 0 1 0 1 0 0 1 0 0 0 0 0 0 0 1 1 0 1 1 1 1</td>
<td>0 1 0 1 0 1 0 0 0 1 0 1 0 0 0 0 0 0 1 1 0 1 1 1</td>
</tr>
<tr>
<td>I</td>
<td>01011 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 1 0 0 0 0 1</td>
<td>0 1 0 1 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 1 1 0 1 1 1</td>
</tr>
<tr>
<td>J</td>
<td>00111 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 1</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 1</td>
</tr>
</tbody>
</table>

When all six dodecahedron sensor signals are assumed present, the dodecahedron-to-triad conversion takes the form

\[
\begin{bmatrix}
S_X \\
S_Y \\
S_Z
\end{bmatrix} = \frac{1}{2} \begin{bmatrix}
\sin \alpha & -\sin \alpha & \cos \alpha & \cos \alpha & 0 & 0 \\
0 & 0 & \sin \alpha & -\sin \alpha & \cos \alpha & \cos \alpha \\
\cos \alpha & \cos \alpha & 0 & 0 & \sin \alpha & -\sin \alpha
\end{bmatrix} \cdot
\begin{bmatrix}
S_A \\
S_B \\
S_C \\
S_D \\
S_E \\
S_F
\end{bmatrix}
\]

The implementation of the conversion is by means of three resolver circuits, each identical to the one shown in Fig. 3-5. Whole-number representations of the constant half-sines and half-cosines are gated into the resolver by synthetic sensor outputs \( \hat{S}_1 \), in accordance with Eq. (3.14) and by sequencing signals \( T_1 \). The overflow output of each resolver is the respective triad signal.
FIG. 3-5 Triad axis generator (3 required).
3.2.2 Parity Integration

To implement the parity equation determination of gyro failure, a weighted integration technique will be adopted. The reasoning behind the use of integration is that quantization effects in both the gyros and the computer must be separated from significant gyro deviations. Weighting is included to mask out long-term drift due to roundoff and other effects within the parity computation itself. Consistent with other sections of the computer, an operational digital procedure will be used in the parity section of the system.

The form of the parity conditions used closely resembles that shown by Gilmore. That is, for the first parity equation

$$Q_1 = (S_A - S_B)\cos \alpha - (S_C + S_D)\sin \alpha \quad (3.15)$$

where $S_A$ is the output of gyro $A$ and is a measure of $\omega_A$. For perfect gyros, $Q_1 = 0$. Thus $Q_1$ represents an angular-rate error. Since pulsed gyros are assumed, where each output pulse represents an increment of angle, the first parity equation may be written in angle form as

$$dX_1 = Q_1 dt = (d\theta_A - d\theta_B)\cos \alpha - (d\theta_C + d\theta_D)\sin \alpha \quad (3.16)$$

and similarly for the other equation. Temporal weighting and integration are introduced by defining the quantity $P_1$, where

$$P_1 = \int_0^t e^{-K(t-\tau)} dX_1 \quad (3.17)$$

or

$$P_1 = \int_0^t e^{-K(t-\tau)} \left(\frac{dX_1(\tau)}{d\tau}\right) d\tau \quad (3.18)$$

If $dX_1/dt = 0$ for $t < 0$ then $P_1$ is seen to be the convolution of $dX_1/dt$ with $e^{-Kt}$. The differential form of the above integral equation may readily be shown to be

$$\frac{dP_1}{dt} = -KP_1 + \frac{dX_1}{dt} \quad (3.19)$$

or

$$dP_1 = -KP_1 dt + dX_1 \quad (3.20)$$
The final form implies a simple DDA solution, as shown in Fig. 3-6.

![DDA Solution Diagram]

The implementation of this configuration using resolvers is shown in Fig. 3-7. A complete solution to allow correction of two failures and detection of three simultaneously requires 15 units of the type in Fig. 3-7, although the elements so designated may be shared among the 15 systems. It is also possible to mechanize a single-failure correction scheme using just six of the units in Fig. 3-7 in which the $dX_j$ inputs are switched among the parity networks. The gyro failures (up to two) are indicated by the binary quantities $\lambda_i$, $i=A,B,C,D,E,F$. The value $\lambda_i=1$ corresponds to the condition that gyro $i$ is operative. The $\lambda_i$ are binary functions of the parity threshold functions $E_J$. Specifically,

$$\bar{\lambda}_i = \prod_{i \in \text{subset}} (E_j)$$  \hspace{1cm} (3.21)

The $E_j$ subset for each $\lambda_i$ is indicated by Table 3.6. The $E_j$ numbering corresponds to Table 2 in Gilmore. Detection of three or more failures is given by

$$\bar{\lambda}_T = \prod_{j=1}^{15} E_j$$  \hspace{1cm} (3.22)

The condition $\lambda_T=0(\bar{\lambda}_T=1)$ indicates that at least three gyros have failed, but the failures cannot be isolated. This is equivalent to complete system failure.

The rationale behind the use of the weighted integral form of parity equation follows that discussed by Keene.\(^2\) The prevalent type of sensor failure is performance degradation, evidenced by either scale factor change or by excessive bias. Let $g$ be the sensor output. Then

$$g = (a + \epsilon)w + b$$  \hspace{1cm} (3.23)

where $a$ is the scale factor, $\epsilon$ is the scale factor error, $b$ is the bias, and $w$ is the measured quantity. Since integrating sensors are normally used, the output of interest is

$$\int gdt = (a + \epsilon) \int wdt + bt$$  \hspace{1cm} (3.24)

Then for constant $w=w_0$. 

- 26 -
FIG. 3-7 Parity integrator.
TABLE 3.6

$E_j$ Subset for $\lambda_i$

<table>
<thead>
<tr>
<th>$\lambda_A$</th>
<th>$\lambda_B$</th>
<th>$\lambda_C$</th>
<th>$\lambda_D$</th>
<th>$\lambda_E$</th>
<th>$\lambda_F$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$E_1$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>$E_2$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>$E_3$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>$E_4$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>$E_5$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>$E_6$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>$E_7$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>$E_8$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>$E_9$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>$E_{10}$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>$E_{11}$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>$E_{12}$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$E_{13}$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$E_{14}$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$E_{15}$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$$\delta = \int gd dt = a_n t + (e^{a_0} + b) t$$  \hspace{1cm} (3.25)

Implies a ramp error behavior. Since $X_1$ is a linear combination of the $\delta$'s, it too exhibits ramp behavior under these conditions. Let $X_1 = \delta_0 t$. Then Eq. (3.19) becomes

$$\frac{dP}{dt} = -KP_1 + Q_0$$  \hspace{1cm} (3.26)

or

$$P_1 = \frac{Q_0}{K} (1 - e^{-Kt})$$  \hspace{1cm} (3.27)

If $M$ is defined as the maximum allowable value of $P_1$, then threshold is exceeded for

$$P_1 > M$$
or

\[
\frac{Q_0}{K} \left(1 - e^{-Kt}\right) > N.
\]

(3.28)

Solving for the time \( t_M \) at which the inequality is satisfied,

\[
t_M > \frac{1}{K} \ln\left(\frac{Q_0}{Q_0 - KM}\right).
\]

(3.29)

Thus, \( KM \) is established as the minimum error rate for parity failure, and the time until such failure is detected is adjusted by \( K \) and decreases with increasing \( Q_0 \).

3.2.3 Constant Rate Multipliers

Although it has been suggested that standard resolvers be used to effect the multiplication of a sensor or other pulse-rate signal by a constant quantity, another interesting approach has been developed. The basis for this approach is given in Appendix A (Smooth Sequences) and stems from the realization that the pulse rate resulting from the multiplication of the source rate by a fixed quantity (less than unity) should preserve as well as possible the relative pulse density of the source. Although the use of resolver-type rate multipliers does in fact preserve relative densities optimally, several techniques have been developed which perform the same function with appreciable circuit savings. One of these well suited for applications in the dodecahedron sensor transformation computer will be described. Consider Fig. 3-8, which shows an interconnection of pulse counters and inhibit elements. The upper counter produces one output pulse for every \( a_1 \) input pulses, and so forth. Each time a pulse counter produces an output pulse it inhibits one input pulse to the counter immediately above it. Analysis of this circuit shows that the pulse output frequency is equal to the pulse output frequency multiplied by \( P/Q \), where \( P \) and \( Q \) are relatively prime integers and

\[
\frac{P}{Q} = \frac{1}{a_1} + \frac{1}{a_2 + \frac{1}{a_3 + \frac{1}{a_4 + \ldots}}}.
\]

(3.30)

Equivalently, \( P \) output pulses are produced for every \( Q \) input pulses. The \( P \) output pulses so produced are synchronous with input pulses, but are spaced as uniformly over the input pulses as is possible.

The representation of \( P/Q \) in Eq. (3.30) is as a simple continued fraction. The properties of such fractions are well known. One property of particular interest here is the
FIG. 3-8 Pulse rate multiplier.
representation of irrational numbers by their approximation in continued fraction form. Theory shows that all algebraic irrational numbers may be represented by an infinite continued fraction whose partial quotients \( a_i \) in Eq. (3.30) are bounded. The theory shows further that the approximation of any rational or irrational number by a truncated simple continued fraction is optimal in the sense that the absolute error incurred is smallest over all rational approximations with denominator no larger than that of the continued-fraction-derived approximation.

A case of special interest in the dodecahedron-to-triad conversion system is the multiplication of sensor output pulse rates by the quantities

\[
\gamma = \frac{\sin \alpha}{\cos \alpha} = \frac{\sqrt{5} - 1}{2}
\]

and

\[
\gamma^{-1} = \frac{\cos \alpha}{\sin \alpha} = \frac{\sqrt{5} + 1}{2} = 1 + \gamma
\]

where \( 2\alpha \) is the angle measured between any two dodecahedron normals. The expansion of \( \gamma \) as an infinite simple continued fraction is given in Eq. (3.33)

\[
\gamma = \frac{1}{1 + \frac{1}{1 + \frac{1}{1 + \frac{1}{1 + \ddots}}}}
\]

This leads to a very simple pulse-rate multiplier of the type shown in Fig. 3-8, since all the \( a_i \) are unity and are therefore just direct connections. The resolution of this multiplier is determined by the number of stages used. For a resolver-type multiplier the resolution is doubled with each stage that is added to the storage register, since this has the effect of adding one more bit to the number added. It is interesting to compare the resolution per stage of the resolver to that of the continued fraction multiplier. Clearly, the resolution of an \( n \)-bit resolver-type multiplier is \( 1/2^n \) and is independent of the quantity by which the rate is multiplied. The approximate resolution of the continued fraction multiplier can be calculated using known properties of continued fraction expansions. The \( n \)th convergent of a continued fraction is defined as the rational fraction obtained by truncating the continued fraction after the \( n \)th partial quotient. If \( C_n \) is the \( n \)th convergent of some continued fraction, then

\[
C_n = \frac{P_n}{Q_n}
\]

and

\[
\ldots 31 \ldots
\]
The odd-order convergents form a monotonic sequence that converges to \( Y \) from above and the even-order convergents form a monotonic sequence that converges to \( Y \) from below. The absolute error \( E_n = |C_n - Y| \) is therefore bounded by the difference between the \( n \)th and \( n-1 \)st convergents. Letting this difference be \( D_n \),

\[
D_n = |C_n - C_{n-1}| = \frac{1}{Q_n Q_{n-1}} \tag{3.36}
\]

and

\[
D_{n-1} = |C_{n-1} - C_{n-2}| = \frac{1}{Q_{n-1} Q_{n-2}} \tag{3.37}
\]

but

\[
lim_{n \to \infty} Q_{n-1} = \gamma Q_n \tag{3.37}
\]

Therefore, in the limit,

\[
D_n \approx \gamma^2 D_{n-1} \tag{3.38}
\]

Thus, each additional partial quotient included in the continued fraction approximation to \( Y \) increases the resolution by a factor of approximately \( \gamma^2 = (3 + \sqrt{5})/2 = 2.618 \), and as \( n \) increases the resolution approaches \( 1/\gamma^{-2n} = 1/(1+\gamma)^{2n} \). The resolution of resolver-type and continued-fraction rate multipliers is compared in Table 3.7. In the strapdown system a resolution equivalent to a word length of 16 bits is generally required. It may be seen from Table 3.7 that this resolution is obtained using only twelve stages of the continued fraction multiplier. Another advantage of the continued fraction multiplier is that it is essentially a parallel system with no carry propagation delays. Therefore, slower electronic components may be used to attain the same speeds as the equivalent resolver.

A working model of a 12-stage continued fraction multiplier was constructed. The circuit diagram of this unit appears as Fig. 3-9.
All flip-flops TI 7474
All NAND gates TI 7400
All NOR gates TI 7402
All inverters TI 7404

FIG. 3-9 Continued-fraction pulse rate multiplier.
<table>
<thead>
<tr>
<th>n</th>
<th>Resolution</th>
<th>Cont. Fraction Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>15</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>40</td>
</tr>
<tr>
<td>5</td>
<td>32</td>
<td>104</td>
</tr>
<tr>
<td>6</td>
<td>64</td>
<td>273</td>
</tr>
<tr>
<td>7</td>
<td>128</td>
<td>714</td>
</tr>
<tr>
<td>8</td>
<td>256</td>
<td>1870</td>
</tr>
<tr>
<td>9</td>
<td>512</td>
<td>4895</td>
</tr>
<tr>
<td>10</td>
<td>1024</td>
<td>12916</td>
</tr>
<tr>
<td>11</td>
<td>2048</td>
<td>33552</td>
</tr>
<tr>
<td>12</td>
<td>4096</td>
<td>87841</td>
</tr>
<tr>
<td>13</td>
<td>8192</td>
<td>229979</td>
</tr>
<tr>
<td>14</td>
<td>16384</td>
<td>602070</td>
</tr>
<tr>
<td>15</td>
<td>32768</td>
<td>1576239</td>
</tr>
<tr>
<td>16</td>
<td>65536</td>
<td>4126648</td>
</tr>
<tr>
<td>17</td>
<td>131072</td>
<td>10803704</td>
</tr>
</tbody>
</table>

REFERENCES


4.1 INTRODUCTION

The objective of this study is to present an incremental computer structure for a space vehicle attitude control system that uses a set of six skewed, single-gimbal, control-moment gyros (CMG's) as actuators and a set of six rate gyros in a dodecahedron configuration as rate sensors. The computer is to employ the redundancy in both the sensors and actuators to provide fail-operational performance. The CMG configuration of six skewed, single-gimbal gyros (exemplified by the Sperry 6-GAMS configuration) can be sized to provide three-axis control with failures in as many as three of its gyros, and the dodecahedron-configured rate gyro package can provide vehicle rate information from any three of its six rate gyros.

4.2 GENERAL DESCRIPTION OF THE CMG CONTROL SYSTEM

Since the computations to be performed by the CMG control computer are directly related to the specific configuration and steering law selected for the CMG's and also to the rate sensor configuration, these items are described briefly in paragraphs 4.2.1, 4.2.2, and 4.2.3, respectively. Paragraph 4.2.4 identifies the functions to be performed by the computer, including a discussion on three-variable versus six-variable input data to the steering law computer. A brief discussion on computer failure detection is presented in paragraph 4.2.5.

4.2.1 The CMG Configuration

Previous studies conducted jointly by Sperry and Lockheed for the Air Force\(^1\) have identified a family of skewed, single-gimbal CMG configurations, denoted as the GAMS family of CMG configurations, to be the most efficient in terms of weight, power, and volume in providing redundant three-axis attitude control. Specifically, the 6-GAMS configuration, illustrated in Figure 4.2-1, is particularly suitable where control is to be maintained with any two of the six CMG's not operating. Although the control computer in this study is directed principally toward this configuration, the results are expected to be adaptable to other configurations.

The black arrows in the CMG configuration model shown in Figure 4-1 depict the angular momentum vectors \( \mathbf{h}_i \) of the six gyros, all of approximately equal magnitude. These vectors are shown in their respective reference directions in the figure. The notation used in describing the configuration angles is shown in Figure 4-2. All gimbal axes are tilted at an angle \( \beta \) from the \( x \)-axis, and the projections of the gimbal axes on the \( yz \)-plane are directed at angles \( \gamma = \{0, 60, 120, 180, 240, 300\} \) degrees from the \( y \)-axis. The gimbal angles denoted by \( \alpha = \{\alpha_1, \ldots, \alpha_6\} \) are measured from their respective reference directions in the counterclockwise direction about the gimbal axes as shown by
FIG. 4-1 Model of the Sperry 6-GAMS configuration.

FIG. 4-2 CMG configuration reference system.
the angular scales of each gyro in Figure 4-1. For \( \alpha = \{ 0, ..., 0 \} \), the net angular momentum vector, \( \mathbf{H} \), of the CMG configuration is zero, assuming that all the individual momenta, \( \{ h_i \} \), are of equal magnitude. In general, the projections of \( \mathbf{H} \) onto the vehicle principal axes are given by equations (4.1), where \( S(\cdot) \) denotes \( \sin(\cdot) \) and \( C(\cdot) \) denotes \( \cos(\cdot) \).

\[
\begin{align*}
\mathbf{H}_x &= \sum_{i=1}^{6} h_i S_{\beta_i} S_{\gamma_i} \\
\mathbf{H}_y &= \sum_{i=1}^{6} h_i \left( S_{\gamma_i} C_{\alpha_i} + C_{\beta_i} C_{\gamma_i} S_{\alpha_i} \right) \\
\mathbf{H}_z &= \sum_{i=1}^{6} h_i \left( C_{\gamma_i} C_{\alpha_i} - C_{\beta_i} S_{\gamma_i} S_{\alpha_i} \right)
\end{align*}
\]

Equations (4.1) do not include the angular momentum due to the rotations of the gimbals and rotors about the gimbal axes which, for the purpose of steering law computations, may be assumed to be negligibly small in comparison with the angular momentum due to the rotation of the rotors about their spin axes.

CMG systems provide attitude control by appropriate transfer of angular momentum between the CMG's and the vehicle. The mathematical law by which desired vehicle angular accelerations (or torques) are translated to gimbal rates is referred to as a steering law.

4.2.2 CMG Steering Laws

The equation of rigid-body angular motion for the vehicle (not including the CMG rotors and gimbals) is given by

\[
\mathbf{T}_V = \mathbf{I}_V \dot{\omega} + \Omega \times \mathbf{I}_V \omega
\]

where

\[
\begin{align*}
\mathbf{T}_V &= \text{3 x 1 matrix of projections (onto the vehicle principal axes) of the net torque vector } \mathbf{T}_V \text{ acting on the vehicle} \\
\mathbf{I}_V &= \text{diagonal matrix of vehicle inertias about the principal axes} \\
\omega &= \text{3 x 1 matrix of projections of the vehicle rate vector } \bar{\omega} \\
\dot{\omega} &= \text{3 x 1 matrix of projections of the vehicle acceleration } \bar{\dot{\omega}}
\end{align*}
\]
Similarly, the equation of motion for each CMG rotor can be expressed in terms of angular momentum as

\[
\mathbf{T}_{R_i} = \mathbf{H}_i + \Omega \mathbf{H}_i
\]

where \( \mathbf{T}_{R_i} \) and \( \mathbf{H}_i \) are the \( 3 \times 1 \) matrices of projections of the torque and angular momentum vectors, respectively, for CMG number \( i \). The net torque, \( T \), produced by all six CMG's on the vehicle, neglecting gimbal inertias, is then given by

\[
T = -\sum_{i=1}^{6} \mathbf{T}_{R_i} = -\mathbf{H} - \Omega \mathbf{H}
\]

where

\[
\mathbf{H} = \sum_{i=1}^{6} \mathbf{H}_i
\]

By the law of conservation of angular momentum,

\[
I_v \dot{\omega} + \mathbf{H} = \mathbf{H}_n
\]

where

\[
\mathbf{H}_n = I_v \omega_0 + \int_{0}^{t} T_d \, dt
\]

\( \mathbf{H}_n \) = net angular momentum of the vehicle plus the CMG's

\( \omega_0 = \omega \) at time \( t = 0 \)

\( T_d \) = torque acting on the vehicle not caused by the CMG,

\( \text{i.e., } T_v = T + T_d \).

The CMG system is normally initialized so that \( \mathbf{H} \) is close to zero when \( \omega = 0 \); i.e., \( \omega_0 \) is normally close to zero.

From the previous equations, the vehicle angular acceleration is given by

\[
\dot{\omega} = -I_v^{-1} \left[ \dot{\mathbf{H}} + \Omega (\mathbf{H}_n) + T_d \right]
\]

\[
\approx -I_v^{-1} \mathbf{H} \cdot 
\]

\[ -38 - \]
This approximation becomes an equality if \( \omega_c = 0 \) and \( T_d = 0 \). The vehicle acceleration, \( \omega \), can thus be controlled by controlling \( H \). If \( H \) can be controlled so that

\[
\dot{H} = -I_v \dot{\omega}_c
\]  

(4.10)

where \( \omega_c \) is a commanded \( \omega \), the torque produced on the vehicle by the CMG's is given by

\[
T = I_v \dot{\omega}_c + \Omega I_v (\omega - \omega_n)
\]  

(4.11)

where \( \omega_n = I_v^{-1} \hat{\omega}_n \), the vehicle rate not caused by the CMG system. For \( \omega_n \approx 0 \), the CMG system thus inherently compensates for the centrifugal couple, \( \Omega I_v \omega \), produced when \( \omega \) has components in two or more of the principal axes for vehicles with unequal inertias.

By taking the time-derivatives of the components of \( H \), given in equations (4.1) for the 6-GAMS configuration, \( H \) can be expressed by

\[
\dot{H} = A \dot{\alpha}
\]  

(4.12)

where \( A \) is a \( 3 \times 6 \) matrix of elements, \( a_{ij} \), given by

\[
a_{1j} = h_j S_\beta C_{\alpha_l}
\]

\[
a_{2j} = -h_j (S_\gamma_j S_\alpha_j - C_\beta C_{\gamma_l} C_{\alpha_l})
\]  

(4.13)

\[
a_{3j} = -h_j (C_\gamma_j S_\alpha_j + C_\beta S_\gamma_j C_{\alpha_l})
\]

for the stated configuration. Thus, the matrix \( A \) is a function of the gimbal angles \( \alpha = \{\alpha_1, \ldots, \alpha_6\} \).

\( \dot{H} \) can therefore be controlled by controlling the gimbal rates, \( \dot{\alpha} \). To control \( \dot{H} \) according to equation (4.10), a solution of equation (4.12) is required. Since \( A \) is singular, any solution that exists is not unique. The specific solution that is adopted for a CMG system, even if it is only an approximate solution, is commonly referred to as the CMG steering law.

The selection of a steering law significantly affects the sizing (magnitude of \( h_4 \)) required for the CMG's in order for the system to be able to meet a specified angular momentum envelope requirement. The most prominent steering law is based on the pseudo-inverse \(^{2,3}\) of \( A \) and is referred to as the
pseudo-inverse steering law. The significant feature of this inverse, denoted by $A^\dagger$, is that it provides the unique minimum norm solution if a solution exists. In this case, $A^\dagger$ can be computed by

$$A^\dagger = A^T (AA^T)^{-1}.$$  \hfill (4.14)

In minimizing the norm of the gimbal rates, this steering law tends to emphasize the gimbal rates of the CMG's that most efficiently provide momentum transfer in the commanded direction, thereby tending to maximize the net angular momentum envelope obtainable for the CMG configuration. Although other steering laws have been investigated (such as the superposition of solutions for three CMG's, and the transpose steering law which provides a very approximate solution), they have been found to be either less efficient with no simplification in the computations, or very inefficient with considerable computational simplification. It may be possible to improve on the pseudo-inverse steering law to increase the efficiency of angular momentum utilization even further, but a study to identify such a law is beyond the scope of this project. Without further justification, the pseudo-inverse steering law is selected for the CMG control computer described in this report.

The steering law computations thus provide gimbal rate commands, $\dot{\alpha}_g$, to the individual gimbal control systems in response to vehicle angular acceleration commands, $\dot{\omega}_v$, in accordance with the following equation:

$$\dot{\alpha}_g = A^\dagger \dot{\omega}_v = -A^\dagger I \dot{\omega}_v.$$  \hfill (4.15)

4.2.3 The Rate Gyro Configuration

The vehicle rate information required for attitude control system stability and/or vehicle rate control is to be obtained from a set of six rate gyros configured to sense vehicle rates along the six dodecahedron axes shown in Figure 4-3. These axes correspond to the normals of the faces of a regular dodecahedron and have the unique property that the acute angles between any two axes are equal, given by $2\beta \approx 63.4$ degrees. This configuration is described by Gilmore. The significant feature of this sensor configuration is that the vehicle rates can be obtained from any three of the six dodecahedron rates, denoted by $r = \{r_1, \ldots, r_6\}$.
FIG. 4-3 Dodecahedron axis system.
The vehicle rates, $\omega$, can be projected onto the dodecahedron axes to obtain the dodecahedron rates $\mathbf{r}$ by

$$ \mathbf{r} = \mathbf{E} \omega \tag{4.16} $$

where

$$ \mathbf{E} = \begin{bmatrix} s & o & c \\ -s & 0 & c \\ c & s & o \\ c & -s & 0 \\ 0 & c & s \\ 0 & o & -s \end{bmatrix} \tag{4.17} $$

and

$$ s = \sin \delta = \left( \frac{5 - \sqrt{5}}{10} \right)^{1/2} \approx 0.526 \tag{4.18} $$

$$ c = \cos \delta = \left( \frac{5 + \sqrt{5}}{10} \right)^{1/2} \approx 0.850 \tag{4.19} $$

If all six rate gyros are operating properly, the signals they generate, denoted by $\mathbf{r}_s$, are approximately equal to $\mathbf{r}$. If a failure is detected in rate gyro number $i$, however, this gyro is disengaged; i.e., $\mathbf{r}_s = 0$. Let

$$ \mathbf{E} = \text{diag} \{ \lambda_1, \ldots, \lambda_6 \} \tag{4.19} $$

where $\lambda_i = 1$ if rate gyro number $i$ is engaged, and $\lambda_i = 0$ if it is disengaged.

Then

$$ \mathbf{r}_s = \lambda \mathbf{r} = \mathbf{E} \omega \tag{4.20} $$

where $\mathbf{E} = \lambda \mathbf{E}$. To determine $\omega$ from $\mathbf{r}_s$ requires the solution of equation (4.20).

Although $\mathbf{E}$ is a constant matrix for every state of $\lambda$, there are 12 states of $\lambda$ for which at least three of its elements are 1's; and there are 20 states for which exactly three of its elements are 1's. Therefore, at least 20 different solutions are required in order to find $\omega$ under all failure conditions (up to three failures).
Since any three rate gyros contain sufficient information to find \( \omega \), only the 20 solutions are required to meet all failure conditions. By using all the rate gyros available, however, the effects of inaccuracies in the signals can be minimized. The pseudo-inverse solution given by equations (4.21) and (4.22) provides the solution for \( \omega \) that minimizes the norm of \( \lambda r - r_s \) (the "least-squares-fit" solution)

\[
\omega_s = E^\dagger r_s
\]  

(4.21)

where

\[
E^\dagger = \left( E^T E \right)^{-1} E^T
\]

\[
= \left( E^T \lambda E \right)^{-1} \left( E^T \lambda \right)
\]

(4.22)

and where \( \omega_s \) is the set of sensed and transformed signals for \( \omega \). This solution is identical to a three-signals-only solution when all but the three respective elements of \( \lambda \) are set to zero.

The detection and isolation of failures in the rate gyro package is also a major task for the CMG control computer in order to generate \( \lambda \). Catastrophic failures can usually be detected by various types of monitors, but errors in the signals are not as easy to detect. Gilmore describes a set of parity equations that isolate two gyro failures and detect a third failure. These equations are presented in Table 4.1 under the notational convention adopted for this report.
TABLE 4.1

Parity Equations for Rate Gyro Failure Detection and Isolation

<table>
<thead>
<tr>
<th>Equation No.</th>
<th>Equation</th>
<th>Signals Compared</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>((r_1 - r_2) c - (r_3 + r_4) s = 0)</td>
<td>1234</td>
</tr>
<tr>
<td>2</td>
<td>((r_2 + r_3) c - (r_1 + r_5) s = 0)</td>
<td>1235</td>
</tr>
<tr>
<td>3</td>
<td>((r_3 - r_1) c + (r_2 - r_6) s = 0)</td>
<td>1236</td>
</tr>
<tr>
<td>4</td>
<td>((r_4 - r_1) c + (r_2 - r_5) s = 0)</td>
<td>1245</td>
</tr>
<tr>
<td>5</td>
<td>((r_2 + r_4) c - (r_1 - r_6) s = 0)</td>
<td>1246</td>
</tr>
<tr>
<td>6</td>
<td>((r_5 - r_6) c - (r_1 + r_2) s = 0)</td>
<td>1256</td>
</tr>
<tr>
<td>7</td>
<td>((r_6 + r_7) c - (r_1 - r_3) s = 0)</td>
<td>1345</td>
</tr>
<tr>
<td>8</td>
<td>((r_6 - r_3) c + (r_1 + r_4) s = 0)</td>
<td>1346</td>
</tr>
<tr>
<td>9</td>
<td>((r_1 + r_6) c - (r_3 + r_5) s = 0)</td>
<td>1356</td>
</tr>
<tr>
<td>10</td>
<td>((r_5 - r_1) c + (r_4 - r_6) s = 0)</td>
<td>1456</td>
</tr>
<tr>
<td>11</td>
<td>((r_5 - r_3) c + (r_4 - r_2) s = 0)</td>
<td>2345</td>
</tr>
<tr>
<td>12</td>
<td>((r_6 + r_4) c + (r_2 - r_3) s = 0)</td>
<td>2346</td>
</tr>
<tr>
<td>13</td>
<td>((r_2 - r_5) c + (r_3 + r_6) s = 0)</td>
<td>2356</td>
</tr>
<tr>
<td>14</td>
<td>((r_2 + r_6) c + (r_4 - r_5) s = 0)</td>
<td>2456</td>
</tr>
<tr>
<td>15</td>
<td>((r_4 - r_3) c + (r_5 - r_6) s = 0)</td>
<td>3456</td>
</tr>
</tbody>
</table>
4.2.4 Control Computer Description

The principal function of the CMG control computer is to generate gimbal rate command signals to the six CMG's of the configuration described in paragraph 4.2.1, such that the vehicle responds in accordance with input command signals for vehicle rate and acceleration, using the rate signals provided by the six dodecahedron configured rate gyro's described in paragraph 4.2.3. An objective of the computer structure is to take advantage of the redundancy in both the CMG and rate gyro configurations to provide fail-operational performance, including failures in the computer.

The control law for the vehicle rate control system, illustrated in Figure 4-4, can be written in the form

\[ \dot{\omega}_c = \dot{\omega}_{cc} + G_3 (\omega_c - \omega_s) \]  

where

\[ \dot{\omega}_{cc} = \text{vehicle acceleration commanded directly as an input to the rate control system} \]

\[ \omega_c - \omega_s = \text{vehicle rate error} \]

\[ G_3 = 3 \times 3 \text{ diagonal matrix of compensation functions} \]

The input signals \( \dot{\omega}_{cc} \) and \( \omega_s \) are provided either by a control law for the vehicle attitude control system or by manual commands. The vehicle rate signal \( \omega_s \) is obtained by converting the six dodecahedron axes rate signals \( \tau_s \) to vehicle axes signals.

Since both the sensor signals \( \tau_s \) and the gimbal rate command signals \( \dot{\omega}_c \) are 6-tuples, the question has been raised as to whether there are any advantages in performing the computations in six variables without first reducing the rate signals to the three-variable format. It has been suggested that by performing computations of data in a redundant format, perhaps some reliability advantage can be obtained with incremental computers, for which simultaneous computations are performed by separate components. The thought is that if one or more of such components should fail without affecting the remainder of the computations, the redundancy of the data and the computations would permit fail-operational performance. Investigations into this approach indicate that fail-operational performance is obtainable for certain component failures, but the complexity of such a system is considerably greater than for a system with redundancy at the overall computer level. In addition, there are many types of computer failures that do not permit fail-operational performance. Two complete
Computers with failure monitoring or three such computers with majority voting provide greater reliability and appear to be less complex for reasons discussed subsequently.

The vehicle accelerations in the dodecahedron axes for which rate gyro signals are available, $\dot{\alpha}$, can be related to gimbal rates by combining equations (4.9), (4.12) and (4.20) to obtain

$$\dot{\alpha} = -Ra$$  \hspace{1cm} (4.24)

where

$$R = EI^{-1}_{\alpha}$$  \hspace{1cm} (4.25)

and where the disturbance terms of equation (4.9) are neglected for simplicity. The steering law in this case is required to solve equation (4.24) for $\alpha_c$ in terms of vehicle acceleration command signals $\dot{r}_c$ in the six-variate format. This command signal is given by

$$\dot{r}_c = R\dot{r}_c$$  \hspace{1cm} (4.26)

and for the vehicle rate control law of equation (4.22)

$$\dot{r}_c = E\dot{r}_c + G^3(\omega - \omega_s)$$  \hspace{1cm} (4.27)

By requiring that the three diagonal elements of $G^3$ be identical, thereby requiring that the control system characteristic be identical in the three vehicle axes, the product $E G^3$ can be written as $G^6 E$, where $G^6$ is a diagonal 6 x 6 matrix having diagonal elements identical to $G^3$. Equation (4.27) can then be written as

$$\dot{r}_c = E\dot{r}_c + G^6 (\dot{E}_{\omega} - \dot{r}_s)$$

$$= \dot{r}_c + G^6 (r_c - r_s)$$  \hspace{1cm} (4.28)

Figure 4-5 shows the structure for this control law.

The steering law in this case is required to invert the 6 x 6 matrix $R$ and the pseudo-inverse is selected for the reasons given in paragraph 4.2.2. The simplest method for computing the pseudo-inverse of a 6 x 6 matrix of rank 3 is to first factor it into the product of a 6 x 3 matrix $M$ times a 3 x 6 matrix $N$, which is always possible. The pseudo-inverse $S$ of $R = MN$ can then be computed by

$$S = R^\dagger = N^T (NN^T)^{-1} (M^TM)^{-1} M^T$$  \hspace{1cm} (4.29)
**FIG. 4-4** Three-variate control law.

**FIG. 4-5** Six-variate control law.
Some authors define the pseudo-inverse by equation (4.29). Therefore
\[ S = N^\dagger M^\dagger \] (4.30)

S can be factored as follows to obtain
\[ S = \left( I_v - A \right)^\dagger E^\dagger = A^\dagger I_v E^\dagger \] (4.31)

Although there are infinitely many ways of factoring R to obtain the unique \( S = R^\dagger \), equation (4.30) indicates that the pseudo-inverse of this 6 x 6 matrix is equivalent to a transformation of the six-variate acceleration commands \( \ddot{r}_c \) to some three-axis format, followed by a transformation to the six-variable gimbal rate commands \( \dot{a}_c \). The question is whether implementation of the 6 x 6 matrix S can provide any advantages in terms of simplicity or reliability over cascading the 6 x 3 and 3 x 6 matrix factors of S.

Direct computation of the elements of S can provide limited fail-operational performance if such failures can be detected and isolated. By expanding the steering law computations
\[ \ddot{a}_c = -S \ddot{r}_c \] (4.32)
to obtain
\[ \ddot{a}_{c1} = -S_{11} \ddot{r}_{c1} - S_{12} \ddot{r}_{c2} - \cdots - S_{16} \ddot{r}_{c6} \]
\[ \ddot{a}_{c2} = -S_{21} \ddot{r}_{c1} - S_{22} \ddot{r}_{c2} - \cdots - S_{26} \ddot{r}_{c6} \] (4.33)
\[ \ddots \]
\[ \ddot{a}_{c6} = -S_{61} \ddot{r}_{c1} - S_{62} \ddot{r}_{c2} - \cdots - S_{66} \ddot{r}_{c6} \]
it can be observed that the effects of a failure in the computation of the element, \( S_{ij} \), can be nullified by letting \( \lambda_j = 0 \), thus causing \( \ddot{r}_{c1} \) to be zero. The steering law is a function of \( \lambda_i \) and is therefore automatically modified to compensate for this step. Since the elements of column j in S are nullified by this procedure, isolated computation of the elements of this column is required. However, all 18 elements of \( A^\dagger \) must be computed for each of the six columns of S, necessitating six isolated computations of these elements, each element being a relatively complex function of the six gimbal angles. An increase in these computations, which constitute the bulk of the steering law computations, by a
factor of six represents a very significant increase in the complexity of the entire computer. Yet it provides fail-operational performance for only a limited set of failure types, namely those failures which can be isolated to the computation of the elements of a column. Further use of the rate gyro corresponding to the failed column is unfortunately also nullified by this procedure, thereby reducing the redundancy of the rate gyro package.

To perform steering law computations on the six-variable signals also requires conversion of the three-variate input command signals, $\omega_{cc}$ and $\omega_c$, to $\omega_{cc}$ and $\omega_c$. In addition, six rate-loop compensation filters are required instead of the three required with the three-variable system.

Based on these observations, it appears that parallel redundancy of the entire computer is preferred both from standpoints of complexity and reliability. The remainder of this report, therefore, considers only the three-variate structure for the control law.

Figure 4-6 shows the basic structure for the CMG control system in terms of its subcomputers, the CMG configuration, the vehicle, and the rate gyro package. Redundancy in the computation is not shown; this aspect is discussed in the following section.

4.2.5 Fail-Operational Computation

Fail-operational performance for a computation function can be obtained with two or more complete channels of computation plus monitors that are able to detect and identify a channel failure. In some cases, the only reliable technique for identifying a failure is to "vote" between at least three channels for a single failure, at least four channels for two failures, and so on. If it is possible to reliably detect a failure in a single channel by a simple method, one less channel of computation is required in comparison with the voting method. It is therefore worthwhile to investigate techniques for failure monitoring.

The steering law computation and the dodecahedron inversion both consist of the inversion of matrices that are very much simpler to compute in the forward direction than to invert. By performing the forward transformation $M$ on the output of an inversion transformation $M^{-1}$ and comparing the resulting output with the original input, an indication of a failure in either of these transformations is obtained. This principle is illustrated in Figure 4-7.

The scope of this study does not permit comparisons of various methods for detecting and isolating failures, and the approach suggested here requires further study.
FIG. 4-6 CMG control system block diagram.

FIG. 4-7 Method of failure detection.
4.3 STEERING LAW COMPUTATION

Since the steering law computation is by far the most complex function to be performed by the CMG control computer, most of the effort under this program has been devoted to this function. Paragraph 4.3.1 presents the equations to be computed by the steering law computer, paragraph 4.3.2 describes the principles of incremental computation, and the remaining paragraphs describe how the equations can be implemented with incremental computation.

4.3.1 Equations and Computer Structure

The steering law computer is required to generate gimbal rate command signals $\dot{\omega}_c$ to the individual gimbal control systems of the CMG's in response to vehicle angular acceleration signals $\ddot{\omega}_v$ in accordance with equations given in paragraph 4.2.2 and repeated below for convenience

$$
\dot{\omega}_c = -A^T I_v \dot{\omega}_v
$$

$$
\dot{\omega}_c = \begin{bmatrix}
\dot{\omega}_{c1} \\
\dot{\omega}_{c2} \\
\dot{\omega}_{c3} \\
\dot{\omega}_{c4} \\
\dot{\omega}_{c5} \\
\dot{\omega}_{c6}
\end{bmatrix}
\dot{\omega}_v = \begin{bmatrix}
\dot{\omega}_{cx} \\
\dot{\omega}_{cy} \\
\dot{\omega}_{cz}
\end{bmatrix}
$$

$I_v$ is the inertia of the vehicle about its principal axes, and $[A^T = A^T (AA^T)^{-1}]$ is the pseudo-inverse of $A$.
To simplify notations in the subsequent development, the following definitions are made.

\[ B = AA^T \]  
\[ C = \text{adj} B \]  
\[ d_0 = \text{det} B \]  
\[ D = A^T J \]  
\[ u = \left( \frac{1}{d_0} \right) \left( I \omega \right) \]  

Then,

\[ A^T = D \left( \frac{1}{d_0} \right) \]  

and

\[ \delta_c = -Du \]  

By leaving \( A^T \) in the factored form, as in equation (4.39) only three divisions by the scalar \( d_0 \) are required compared to 18 such divisions if the elements of \( A^T \) were to be computed. Since \( B \) is only a 3x3 matrix, the adjoint method for inverting \( B \) is used. The steering law computer can now be structured of sub-computers related to the above defined variables so that they can be discussed separately. Figure 4-8 illustrates this structure of the steering law computer.

The \( A \) computer computes the elements of \( A \) given by

\[ a_{1j} = h_j S \beta \alpha_j \]  
\[ a_{2j} = h_j \left( S \gamma_j S \alpha_j - C \beta \gamma_j \alpha_j \right) \]  
\[ a_{3j} = -h_j \left( C \gamma_j S \alpha_j + C \beta \gamma_j \alpha_j \right) \]  

- 52 -
FIG. 4-8 Organization of the pseudo-inverse steering law computer.
Element \( a_{ij} \) may be factored into

\[
a_{ij} = \rho_{ij} \sigma_j
\]

(4.42)

where

\[
\rho_{ij} = (m_{ij} S_{\alpha_j} + n_{ij} C_{\alpha_j})
\]

(4.43)

and where \( \sigma_j \) is the speed of rotor \( j \) in revolutions per second. The constants \( m_{ij} \) and \( n_{ij} \) are given by

\[
m_{ij} = 0 \quad n_{ij} = 2n I_R S_j
\]

\[
m_{2j} = 2n I_R S_j \quad n_{2j} = -2n I_R C_j
\]

(4.44)

\[
m_{3j} = -2n I R C_j \quad n_{3j} = -2n I_R S_j
\]

where \( I_R \) is the inertia of the CMG rotor about its spin axis. For the 6-GAMS configuration described in paragraph 4.2.1, the \( \gamma \) angles are 0, 60, 120, 180, 240, and 300 degrees so \{S_{\gamma_j}\} and \{C_{\gamma_j}\} are specified, but the gimbal axis tilt angle \( \beta \) depends on the requirements for the net angular momentum envelope.

Thus, the \( A \) computer for each of the 18 elements \( \{a_{ij}\} \) performs the scaling and addition of the input signals, \( S_{\alpha_j} \) and \( C_{\alpha_j} \), as indicated by equation (4.3-10) and multiplies the result by the input signal for rotor speed, \( \sigma_j \).

To help visualize the complexity associated with each of the subcomputers of the steering law computer, the related matrix expressions are expanded in the following discussion. Since \( B = AA^T \) is a symmetric 3x3 matrix, only six elements such as those making up the upper triangle of \( B \) are required. These elements are

\[
b_{11} = a_{11}^2 + a_{12}^2 + a_{13}^2 + a_{14}^2 + a_{15}^2 + a_{16}^2
\]

\[
b_{12} = a_{11} a_{21} + a_{12} a_{22} + a_{13} a_{23} + a_{14} a_{24} + a_{15} a_{25} + a_{16} a_{26}
\]

\[
b_{13} = a_{11} a_{31} + a_{12} a_{32} + a_{13} a_{33} + a_{14} a_{34} + a_{15} a_{35} + a_{16} a_{36}
\]

\[
b_{22} = a_{21}^2 + a_{22}^2 + a_{23}^2 + a_{24}^2 + a_{25}^2 + a_{26}^2
\]

\[
b_{23} = a_{21} a_{31} + a_{22} a_{32} + a_{23} a_{33} + a_{24} a_{34} + a_{25} a_{35} + a_{26} a_{36}
\]

\[
b_{33} = a_{31}^2 + a_{32}^2 + a_{33}^2 + a_{34}^2 + a_{35}^2 + a_{36}^2
\]

(4.45)
The C computer evaluates the upper triangle elements of \( C = \text{adj} B \) since this matrix is also symmetric:

\[
\begin{align*}
    c_{11} &= b_{22}b_{33} - b_{23}^2 \\
    c_{12} &= b_{13}b_{23} - b_{12}b_{33} \\
    c_{13} &= b_{12}b_{23} - b_{13}b_{22} \\
    c_{22} &= b_{11}b_{33} - b_{13}^2 \\
    c_{23} &= b_{12}b_{13} - b_{11}b_{23} \\
    c_{33} &= b_{11}b_{22} - b_{12}^2.
\end{align*}
\] (4.46)

Elements of \( B \) and \( C \) are used to compute \( d_0 \):

\[
d_0 = b_{11}c_{11} + b_{12}c_{12} + b_{13}c_{13} \] (4.47)

The 18 elements of \( D = A^Tc \) are computed as follows:

\[
\begin{align*}
    d_{11} &= a_{11}c_{11} + a_{21}c_{12} + a_{31}c_{13} \\
    d_{12} &= a_{11}c_{12} + a_{21}c_{22} + a_{31}c_{23} \\
    d_{13} &= a_{11}c_{13} + a_{21}c_{23} + a_{31}c_{33} \\
    d_{21} &= a_{12}c_{11} + a_{22}c_{12} + a_{32}c_{13} \\
    d_{22} &= a_{12}c_{12} + a_{22}c_{22} + a_{32}c_{23} \\
    d_{23} &= a_{12}c_{13} + a_{22}c_{23} + a_{32}c_{33} \\
    d_{31} &= a_{13}c_{11} + a_{23}c_{12} + a_{33}c_{13} \\
    d_{32} &= a_{13}c_{12} + a_{23}c_{22} + a_{33}c_{23} \\
    d_{33} &= a_{13}c_{13} + a_{23}c_{23} + a_{33}c_{33} \\
    d_{41} &= a_{14}c_{11} + a_{24}c_{12} + a_{34}c_{13} \\
    d_{42} &= a_{14}c_{12} + a_{24}c_{22} + a_{34}c_{23} \\
    d_{43} &= a_{14}c_{13} + a_{24}c_{23} + a_{34}c_{33}
\end{align*}
\] (4.48)
The only divisions required in the steering law computer are in the u computer. The elements of this $3 \times 1$ matrix are

$$
u_1 = \frac{\dot{I}_x \dot{\omega}_x}{\dot{d}_0}
$$

$$
u_2 = \frac{\dot{I}_x \dot{\omega}_y}{\dot{d}_0}
$$

$$
u_3 = \frac{\dot{I}_x \dot{\omega}_z}{\dot{d}_0}
$$

The $\dot{d}_0$ computer then provides the outputs of the steering law computer by performing the following computations:

$$
\dot{d}_{c1} = d_{11} u_1 + d_{12} u_2 + d_{13} u_3
$$

$$
\dot{d}_{c2} = d_{21} u_1 + d_{22} u_2 + d_{23} u_3
$$

$$
\dot{d}_{c3} = d_{31} u_1 + d_{32} u_2 + d_{33} u_3
$$

$$
\dot{d}_{c4} = d_{41} u_1 + d_{42} u_2 + d_{43} u_3
$$

$$
\dot{d}_{c5} = d_{51} u_1 + d_{52} u_2 + d_{53} u_3
$$

$$
\dot{d}_{c6} = d_{61} u_1 + d_{62} u_2 + d_{63} u_3
$$

(4.50)
A summary of the types and numbers of mathematical operations required for the subcomputations of the steering law computer is presented in Table 4.2.

**TABLE 4.2**  
Mathematic Operations for the Steering Law Computer

<table>
<thead>
<tr>
<th>Subcomputer</th>
<th>Additions</th>
<th>Multiplications</th>
<th>Divisions</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>18</td>
<td>18</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>30</td>
<td>36</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>6</td>
<td>12</td>
<td>0</td>
</tr>
<tr>
<td>d_o</td>
<td>2</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>D</td>
<td>36</td>
<td>54</td>
<td>0</td>
</tr>
<tr>
<td>u</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>d_c</td>
<td>12</td>
<td>18</td>
<td>0</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>104</strong></td>
<td><strong>141</strong></td>
<td><strong>3</strong></td>
</tr>
</tbody>
</table>
### 3.2 Incremental Computation

An incremental computer is a special purpose digital computer that has a number of features which make it attractive for solving equations of the type required for the pseudo-inverse steering law. In contrast with the general purpose type digital computer, which performs entire computations during each computation cycle, an incremental computer updates previous computations. Such computations are generally much simpler and are performed simultaneously on the numerous variables of the problem. The time required to complete each computation cycle is therefore much less than for a general purpose computer, permitting higher sampling rates.

In an incremental computer, the variables of the computations are stored in binary registers, referred to as Y registers. The content (numerical value) stored in a Y register is increased by one when so commanded by a pulse, \( \Delta Y \), representing an incremental increase in the variable \( Y \). The \( \Delta Y \) pulse can also be negative, in which case the Y register is decreased by one. In addition, the content of a Y register is added to the content of an R register when so commanded by a positive \( \Delta X \) pulse, or it is subtracted from the R register for a negative \( \Delta X \) pulse.

Figure 4.3-2 shows the schematic symbols adopted for the incremental computer elements. Since the R register has the same capacity as the Y register, repeated \( \Delta X \) pulses of the same sign will cause the R register to overflow. Each time the R register overflows in the positive direction, a positive \( \Delta Z \) pulse is generated; if the overflow is in the negative direction, a negative \( \Delta Z \) pulse is generated. This \( \Delta Z \) pulse can then serve as a \( \Delta X \) pulse or a \( \Delta Y \) pulse for other Y and R registers. Let

\[
Z(n) = \sum_{i=1}^{n} \Delta Z(i) \tag{4.51}
\]

where \( \Delta Z(i) \) is the \( \Delta Z \) pulse produced by the \( i \)'th \( \Delta X \) pulse, \( \Delta X(i) \) \[ \Delta Z(i) \) may be 0, +1, or -1]. Let \( Y(i) \) and \( R(i) \) be the contents of the Y and R registers, respectively, at the occurrence of \( \Delta X(i) \), and let \( c \) be the capacity of the registers. Then

\[
cZ(n) + R(n) = R(1) + \sum_{i=1}^{n} Y(i) \Delta X(i) \tag{4.52}
\]

Therefore, for \( |Y(1)| > 1 \),

\[
Z(n) \approx \frac{1}{c} \sum_{i=1}^{n} Y(i) \Delta X(i) \tag{4.53}
\]
For the case where \(Y(i)\) is a function of \(X(i)\),

\[
Z(n) = \frac{1}{c} \int_0^X(n) Y[X(i)] \, dX(i). \tag{4.54}
\]

This computation structure may thus be used to integrate functions of independent variables, and combinations of such elements may be used to generate functions that are solutions of differential equations. Computers structured by interconnecting integrators of this type are commonly referred to as DDA's (digital differential analyzers).

The computations for the steering law require only additions, multiplications, and divisions. The sum of several variables can be obtained by feeding their respective \(\Delta\) pulses to a common \(Y\) register and properly controlling their pulse times so that they do not occur simultaneously. Such timing control can be implemented by various methods, and will be discussed later.

Multiplication is readily obtained by noting that

\[
d(Y_1Y_2) = Y_1dY_2 + Y_2dY_1 \tag{4.55}
\]

Therefore, let \(\Delta X_1 = \Delta Y_2\) and \(\Delta X_2 = \Delta Y_1\), and let the two \(Y\) registers add into a common \(R\) register as shown in Figure 4-10. Then

\[
\Delta Z = \frac{1}{c} \left( Y_1 \Delta Y_2 + Y_2 \Delta Y_1 \right) \approx \frac{1}{c} \Delta(Y_1Y_2). \tag{4.56}
\]

The accuracy of this computation improves with the size \(c\) of the registers. It can also be improved by proper sequencing of the operations when \(\Delta Y_1\) and \(\Delta Y_2\) occur simultaneously. Investigations of such methods are beyond the scope of this study.

### 4.3.3 \(\Delta A\) Computation

The \(\Delta A\) computer generates increments \(\{\Delta a_{ij}\}\) to increase or decrease the values in the set of 18 registers that store the elements of the \(A\) matrix, given by equations (4.42), (4.43) and (4.44). Column \(j\) of \(A\) corresponds to CMG number \(j\). The incremental computer structure for the three elements of this column is shown in Figure 4-11.

The \(Y\) register corresponding to some variable, \(y\), is identified by this variable, and it stores the numerical value \(K_y\) where \(K_y\) is a scale factor. The \(R\) register that generates \(\Delta y\) is identified by \(\nabla y\). Constant numbers are represented by the oval-shaped areas in Figure 4-11. For example, when a positive pulse for \(\Delta a_{ij}\) occurs, the \(Y\) register containing \(K_{yj}\) is increased by the constant number \(K_{\Delta a_{ij}}\).
FIG. 4-9 Basic incremental computer elements.

FIG. 4-10 Incremental multiplier.
FIG. 4-11 ΔA computer.
The rotor speed $\sigma_j$ (revolutions per second) can be held constant by the spin motor electronics to some degree of accuracy, but high accuracy may be difficult to achieve. If the rotor speed is sensed and included in the steering law computations, rotor speed control is not required at all, permitting simple spin motor electronics. The rotor speed is therefore included in the steering law computations.

The rotor speed is computed by counting revolutions $\theta_j$ over a period of time $T_\sigma$. A pulse $\Delta \theta_j$ produced for each rotor revolution increases the counter by one. A pulse $\Delta T_\sigma$ is produced each $T_\sigma$ seconds by some clock. $T_\sigma$ must be quite long, such as several seconds, to permit sufficient resolution in $\sigma_j$. When the $\Delta T_\sigma$ pulse occurs, the $\theta_j$ register is first set to $K_{\theta_j} \theta_j$, where $\theta_j$ is the present number in the counter and where $K_{\theta_j} = K_{\sigma_j} / T_\sigma$ (chosen to be a power of 2 for simplicity). The counter is then reset to zero. Next, $K_{\theta_j} \theta_j$ and $\theta_j$ are compared by adding the first to, and subtracting the second from, the $\theta_j$ register. If this difference is positive, a positive $\Delta \sigma_j$ pulse is generated and causes the $\sigma_j$ register to increase by $K_{\sigma_j}$, or vice versa for a negative pulse.

To obtain maximum accuracy in incremental computations, the $Y$ registers should be scaled so low that they are as full as possible without overflowing. Therefore, $K_{\sigma_j}$ is selected so that $K_{\sigma_j}$ times maximum $\sigma_j$ is close to but does not exceed the register capacity $C$. $T_\sigma$ (or $K_{\theta_j} = K_{\sigma_j} / T_\sigma$) should be selected to minimize the error in $\sigma_j$. If $T_\sigma$ is very large, the maximum error in $\sigma_j$ due to resolution, given by $\sigma_j / \theta_j = 1 / T_\sigma$, is small. However, the maximum error due to a change in $\sigma_j$ during the $T_\sigma$ interval, $\sigma_m T_\sigma$ (where $\sigma_m = \max \sigma$, assumed constant in this interval), is large. The $T_\sigma$ for which these errors are equal is given by $T_\sigma = 1 / \sqrt{\sigma_m}$.

Incremental signals for $\sin \sigma_j$ and $\cos \sigma_j$ can be obtained by numerous methods, but at the present state of the art a set of six resolvers with a single time-shared A/D converter for all six gimbal angles appears to be the best candidate. Selection of a preferred method is not performed under this study. The scale factor for the registers in the A/D converter that store $S_{\sigma_j}$ and $C_{\sigma_j}$ is $K_{sc}$; i.e., a $\Delta S_{\sigma_j}$ pulse corresponds to a change of $1/K_{sc}$ in $S_{\sigma_j}$, etc.
When a positive $\Delta a_{ij}^{+}$ pulse or $\Delta c_{ij}^{+}$ pulse occurs, the $\tilde{a}_{ij}$ registers are increased by the respective constants, $m_{ij}$ and $n_{ij}$. If this produces a positive overflow of a $\rho_{ij}$ register, the resulting $\Delta \rho_{ij}^{+}$ pulse causes $\rho_{ij}$ to increase by one, and $\sigma_{j}$ to be added to $\tilde{a}_{ij}$. Overflows in the $\tilde{a}_{ij}$ registers then produce the output pulses of the AA computer, which are held in flip-flops until called for in the next computation cycle. Additions to the $\tilde{a}_{ij}$ registers are also produced by a $\Delta \sigma_{j}^{+}$ pulse.

Although there are many ways to perform the functional operations symbolized in Figure 4-11, considerable savings in hardware may be realized by serializing many of the variables on a single, long, circulating register. Table 4.3 describes one method for sequencing the additions into the $\tilde{a}_{ij}$ registers. All 18 variables $\{\rho_{ij}\}$ are serialized on a single, long register in the sequence shown in the table, and the size variables $\{\sigma_{j}\}$ are serialized on a single register such that exactly three cycles of the $\sigma$ register coincide with one cycle of the $\rho$ register, and such that the least significant bits of $\sigma_{1}$, $\sigma_{11}$, and $\tilde{a}_{ij}$ coincide, etc, as shown. The 18 $\tilde{a}$ registers are the same length as each word of $\rho$ and $\sigma$. During word interval 1, the $\rho$ register adds into the $\tilde{a}_{11}$ register if $\Delta \rho_{1} = 1$ (or subtracts if $\Delta \rho_{1} = -1$), the $\sigma$ register adds into $\tilde{a}_{16}$ if $\Delta \sigma_{16} = 1$ (or subtracts if $\Delta \sigma_{16} = -1$), etc, as shown in the table. It is also possible to serialize the $a_{ij}$ variable with increased logic complexity and longer cycle time.
TABLE 4.3
A Computation Sequence

<table>
<thead>
<tr>
<th>Interval</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\rho_1$</td>
<td>$\rho_{11}$</td>
<td>$\rho_{21}$</td>
<td>$\rho_{31}$</td>
<td>$\rho_{12}$</td>
<td>$\rho_{22}$</td>
<td>$\rho_{32}$</td>
<td>$\rho_{13}$</td>
<td>$\rho_{23}$</td>
<td>$\rho_{33}$</td>
<td>$\rho_{14}$</td>
<td>$\rho_{24}$</td>
<td>$\rho_{34}$</td>
<td>$\rho_{15}$</td>
<td>$\rho_{25}$</td>
<td>$\rho_{35}$</td>
<td>$\rho_{16}$</td>
<td>$\rho_{26}$</td>
<td>$\rho_{36}$</td>
</tr>
</tbody>
</table>

| $\sigma_1$ | $\sigma_{11}$ | $\sigma_{21}$ | $\sigma_{31}$ | $\sigma_{12}$ | $\sigma_{22}$ | $\sigma_{32}$ | $\sigma_{13}$ | $\sigma_{23}$ | $\sigma_{33}$ | $\sigma_{14}$ | $\sigma_{24}$ | $\sigma_{34}$ | $\sigma_{15}$ | $\sigma_{25}$ | $\sigma_{35}$ | $\sigma_{16}$ | $\sigma_{26}$ | $\sigma_{36}$ |

...
The $\Delta a_{ij}$ pulses accumulate in the $a_{ij}$ register (which is part of the AB computer). The number stored in this register is given by [per equation (4.56)]

$$K_{a_{ij}} = \frac{1}{c} \left( K_p \rho_{ij} \right) \left( K_0 \sigma_{ij} \right).$$  \hspace{1cm} (4.57)

Therefore, since $a_{ij} = \rho_{ij} \sigma_{ij}$ [equation (4.42)], the scale factors are required to satisfy

$$K_{a_{ij}} = \frac{K_p K_0}{c}$$  \hspace{1cm} (4.58)

In addition, the $Y$ registers cannot be allowed to overflow. Therefore,

$$K_{\rho_{ij}} \max |\rho_{ij}| \leq c \hspace{1cm} (4.59)$$

$$K_{\sigma_{ij}} \max |\sigma_{ij}| \leq c \hspace{1cm} (4.60)$$

$$K_{a_{ij}} \max |a_{ij}| \leq c \hspace{1cm} (4.61)$$

If more than one $\Delta a_{ij}$ pulse is permitted to occur for each computation cycle, the complexity of the $\Delta a_{ij}$ adder must be increased significantly. To avoid this situation, let

$$K_{\rho_{ij}} \max |\rho_{ij}| + K_{\sigma_{ij}} \max |\sigma_{ij}| \leq c \hspace{1cm} (4.62)$$

Maximum accuracy is obtained with the scale factors as large as possible, subject to the above constraints.

Let the constants $m_{ij}$ and $n_{ij}$ have the same scale factor, $K_{m_{ij}}$. Then

$$K_{\rho_{ij}} \rho_{ij} = \frac{1}{c} K_{m_{ij}} K_{sc} \left( m_{ij} s_{ij} + n_{ij} c_{ij} \right).$$  \hspace{1cm} (4.63)

Since $\rho_{ij} = m_{ij} s_{ij} + n_{ij} c_{ij}$ [equation (4.43)], the scale factors must satisfy

$$K_{\rho_{ij}} = \frac{K_{m_{ij}} K_{sc}}{c}.$$

Also, to prevent more than one $\Delta \rho_{ij}$ pulse per computation cycle,

$$K_{m_{ij}} \left( |m_{ij}| + |n_{ij}| \right) \leq c.$$  \hspace{1cm} (4.65)
4.3.4 \( \Delta B \) Computation

The \( \Delta B \) computer calculates the increments for the six upper triangle elements of the \( B \) matrix given in equations (4.45). Figure 4-12 shows the structure of this computer, which has 18 \( Y \) registers to store the \( A \) matrix elements and six \( R \) registers to generate the \( \Delta b_{ij} \) pulses.

Each of the three diagonal elements of \( B \) is the sum of the squares of the six elements in a row of \( A \). By letting \( \Delta X = \Delta Y \) (see Figure 4-9), we obtain

\[
\Delta Z = \frac{1}{c} Y^2 Y
\]

\[
\approx \frac{1}{2c} \Delta (Y^2).
\]

Therefore, by adding \( K_{a_{ij}} a_{ij} \) to \( K_{b_{ii}} b_{ii} \) when \( \Delta a_{ij} = +1 \) for each \( j \), the content of the \( Y \) register that accumulates the \( \Delta b_{ii} \) pulses will be

\[
K_{b_{ii}} b_{ii} \approx \frac{1}{2c} \sum_{j=1}^{6} (K_{a_{ij}} a_{ij})^2.
\]

In order for

\[
b_{ii} = \sum_{j=1}^{6} a_{ij}^2
\]

the scale factors for the diagonal elements of \( B \) must satisfy

\[
K_{b_{ii}} = \frac{K_{a_{ii}}^2}{2c}
\]

where \( K_{a_{i}} = K_{a_{ij}} \) for all \( j \).

For the off-diagonal elements, multiplication is performed as described previously. In this case,

\[
K_{b_{ij}} b_{ij} = \frac{1}{c} \sum_{k=1}^{6} K_{a_{ik}} a_{ik} K_{a_{jk}} a_{jk}.
\]

Therefore,

\[
K_{b_{ij}} = \frac{K_{a_{i}} K_{a_{j}}}{c}, \quad i \neq j.
\]
FIG. 4-12 \( \Delta B \) computer.
The AC computer, described in paragraph 3.5, requires that all elements of B have the same scale factor. Therefore, let $K_{bij} = K_b$ for all $i$ and $j$. The result is that $K_{bij}$ must equal $K_b/2$, which may be accomplished either by adding twice to the $b_{ii}$ registers for each $A_{aij}$ or by delaying the additions by one bit, thus doubling the number added to the register.

In order to ensure that, at most, one $A_{bij}$ pulse is produced during each computation cycle, the following constraint is placed on $K_a$:

$$K_a \sum_{k=1}^{6} \max |a_{ik}| + \max |a_{jk}| \leq c \text{ for all } i, j.$$  \hspace{1cm} (4.72)

This constraint limits $K_a$ to a lower value than the constraint given by equation (4.71). Since $K_{aij} = K_a$ for all $i$ and $j$ and since it is reasonable to let $K_{oij} = K_o$ for all $j$ (all wheels have the same speed), equation (4.50) indicates that $K_{oij} = K_o$ for all $i$ and $j$. Equation (4.53) is therefore replaced with

$$K_a = \frac{K_o K_p}{c}. \hspace{1cm} (4.73)$$

Table 4.4 shows a computation sequence for the additions to the $\tilde{b}_{ij}$ registers where the variables $a_{ij}$ are stored serially on a single circulating register in the order shown. Each $b_{ij}$ register recycles in the period of a one-word interval of the A register. During interval 1, twice the number in the A register during that interval is added to the $\tilde{b}_{11}$ register if $A_{a11} = 1$. The A register also adds to the $\tilde{b}_{12}$ register if $A_{a21} = 1$ during this interval, and to the $\tilde{b}_{13}$ register if $A_{a31} = 1$. All the computations of the AC computer are completed in one cycle of the A register.
TABLE 4.4

B Computation Sequence

<table>
<thead>
<tr>
<th>Interval</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>a_11</td>
<td>a_21</td>
<td>a_31</td>
<td>a_12</td>
<td>a_22</td>
<td>a_32</td>
<td>a_13</td>
<td>a_23</td>
<td>a_33</td>
<td>a_14</td>
<td>a_24</td>
<td>a_34</td>
<td>a_15</td>
<td>a_25</td>
<td>a_35</td>
<td>a_16</td>
<td>a_26</td>
<td>a_36</td>
</tr>
<tr>
<td>b_11</td>
<td>2A a_11</td>
<td></td>
<td></td>
<td>2A^₂ a_12</td>
<td></td>
<td></td>
<td>2A^₂ a_13</td>
<td></td>
<td>2A^₂ a_14</td>
<td></td>
<td>2A a_15</td>
<td></td>
<td>2A a_16</td>
<td></td>
<td>2A a_17</td>
<td></td>
<td>2A a_18</td>
<td></td>
</tr>
<tr>
<td>b_12</td>
<td>A a_21</td>
<td>A a_11</td>
<td>A a_22</td>
<td>A a_12</td>
<td>A a_23</td>
<td>A a_13</td>
<td>A a_24</td>
<td>A a_14</td>
<td>A a_25</td>
<td>A a_15</td>
<td>A a_26</td>
<td>A a_16</td>
<td>A a_27</td>
<td>A a_17</td>
<td>A a_28</td>
<td>A a_18</td>
<td>A a_29</td>
<td>A a_19</td>
</tr>
<tr>
<td>b_13</td>
<td>A a_31</td>
<td>A a_22</td>
<td>A a_32</td>
<td>A a_23</td>
<td>A a_33</td>
<td>A a_24</td>
<td>A a_34</td>
<td>A a_25</td>
<td>A a_35</td>
<td>A a_26</td>
<td>A a_36</td>
<td>A a_27</td>
<td>A a_37</td>
<td>A a_28</td>
<td>A a_38</td>
<td>A a_29</td>
<td>A a_39</td>
<td>A a_30</td>
</tr>
<tr>
<td>b_22</td>
<td>2A a_21</td>
<td>A a_31</td>
<td>A a_32</td>
<td>A a_33</td>
<td>A a_43</td>
<td>A a_42</td>
<td>A a_34</td>
<td>A a_43</td>
<td>A a_44</td>
<td>A a_45</td>
<td>A a_46</td>
<td>A a_47</td>
<td>A a_48</td>
<td>A a_49</td>
<td>A a_50</td>
<td>A a_51</td>
<td>A a_52</td>
<td>A a_53</td>
</tr>
<tr>
<td>b_23</td>
<td>A a_22</td>
<td>A a_33</td>
<td>A a_42</td>
<td>A a_34</td>
<td>A a_43</td>
<td>A a_44</td>
<td>A a_45</td>
<td>A a_46</td>
<td>A a_47</td>
<td>A a_48</td>
<td>A a_49</td>
<td>A a_50</td>
<td>A a_51</td>
<td>A a_52</td>
<td>A a_53</td>
<td>A a_54</td>
<td>A a_55</td>
<td>A a_56</td>
</tr>
<tr>
<td>b_33</td>
<td>2A a_31</td>
<td>2A a_32</td>
<td>2A a_33</td>
<td>2A a_34</td>
<td>2A a_35</td>
<td>2A a_36</td>
<td>2A a_37</td>
<td>2A a_38</td>
<td>2A a_39</td>
<td>2A a_40</td>
<td>2A a_41</td>
<td>2A a_42</td>
<td>2A a_43</td>
<td>2A a_44</td>
<td>2A a_45</td>
<td>2A a_46</td>
<td>2A a_47</td>
<td>2A a_48</td>
</tr>
</tbody>
</table>

Note: The table represents a sequence of computations in interval x for a specific scenario, with entries indicating the specific operations or states at each step.
4.3.5 AC Computation

The AC computer, illustrated in Figure 4.13, computes increments for the adjoint of the B matrix given by equations (4.46). As in the AB computer, when squaring computation is performed, the addition to an R register is doubled. This can be accomplished by delaying the number to be added by one bit time. The doubled addition is signified in the figure by a "2" placed next to the AX arrowhead which indicates addition of Y to R. As for the previous computers, the scale factors must satisfy

\[ K_c = \frac{K_b^2}{c} \]  

(4.74)

By requiring that

\[ K_b \cdot \sum |b_{ij}| \leq c \]  

(4.75)

where the summation is taken over the four terms \( b_{ij} \) of each of equations (4.46), only one Ac pulse will be produced during each computation cycle.

Table 4.5 presents a computation sequence for the additions to the \( \~c \) registers for this computation, where the elements of B are stored serially on a single circulating register in the sequence shown.

<table>
<thead>
<tr>
<th>Interval:</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>B:</td>
<td>( b_{11} )</td>
<td>( b_{12} )</td>
<td>( b_{13} )</td>
<td>( b_{22} )</td>
<td>( b_{23} )</td>
<td>( b_{33} )</td>
</tr>
<tr>
<td>( ~c_{11} )</td>
<td>( -B\Delta b_{33} )</td>
<td>( B\Delta b_{23} )</td>
<td>( -B\Delta b_{22} )</td>
<td>( B\Delta b_{13} )</td>
<td>( -2B\Delta b_{12} )</td>
<td>( B\Delta b_{11} )</td>
</tr>
<tr>
<td>( ~c_{12} )</td>
<td>( B\Delta b_{33} )</td>
<td>( B\Delta b_{23} )</td>
<td>( -B\Delta b_{22} )</td>
<td>( -B\Delta b_{13} )</td>
<td>( B\Delta b_{12} )</td>
<td>( B\Delta b_{11} )</td>
</tr>
<tr>
<td>( ~c_{13} )</td>
<td>( -B\Delta b_{23} )</td>
<td>( B\Delta b_{13} )</td>
<td>( B\Delta b_{12} )</td>
<td>( -B\Delta b_{11} )</td>
<td>( 2B\Delta b_{12} )</td>
<td>( -B\Delta b_{11} )</td>
</tr>
<tr>
<td>( ~c_{22} )</td>
<td>( B\Delta b_{33} )</td>
<td>( B\Delta b_{13} )</td>
<td>( -B\Delta b_{12} )</td>
<td>( B\Delta b_{11} )</td>
<td>( -2B\Delta b_{12} )</td>
<td>( B\Delta b_{11} )</td>
</tr>
<tr>
<td>( ~c_{23} )</td>
<td>( -B\Delta b_{23} )</td>
<td>( B\Delta b_{13} )</td>
<td>( B\Delta b_{12} )</td>
<td>( -B\Delta b_{11} )</td>
<td>( B\Delta b_{11} )</td>
<td>( 2B\Delta b_{12} )</td>
</tr>
<tr>
<td>( ~c_{33} )</td>
<td>( B\Delta b_{33} )</td>
<td>( B\Delta b_{13} )</td>
<td>( -B\Delta b_{12} )</td>
<td>( B\Delta b_{11} )</td>
<td>( B\Delta b_{11} )</td>
<td>( -2B\Delta b_{12} )</td>
</tr>
<tr>
<td>C:</td>
<td>( c_{22} )</td>
<td>( c_{23} )</td>
<td>( c_{33} )</td>
<td>( c_{11} )</td>
<td>( c_{12} )</td>
<td>( c_{13} )</td>
</tr>
<tr>
<td>( ~d_{0} )</td>
<td>( B\Delta c_{11} )</td>
<td>( B\Delta c_{12} )</td>
<td>( B\Delta c_{13} )</td>
<td>( C\Delta b_{11} )</td>
<td>( C\Delta b_{12} )</td>
<td>( C\Delta b_{13} )</td>
</tr>
</tbody>
</table>
FIG. 4-13 ΔC computer.

- 71 -
4.3.6 $\Delta d_0$ Computation

Computation of the increments for $d_0$, the determinant at the B matrix [equation (4.47)] is performed by adding elements of B and C to a single R register, as shown in Figure 4-14. The sequence of these additions is shown in Table 4.5, where the elements of C are stored on a single register in the sequence shown. The scale factor for $d_0$ is given by

$$K_{d_0} = \frac{K_B K_C}{c}$$

(4.76)

and by constraining $K_B$ and $K_C$ so that

$$K_B K_C \sum_{j=1}^{3} \max |b_{1j}| + \max |c_{1j}| \leq c$$

(4.77)

with only one $\Delta d_0$ pulse produced during each computation cycle.

4.3.7 $\Delta d$ Computation

Figure 4-15 illustrates computation of the increments for the 18 D elements given by equations (4.48). Similarly to the previous computers, the scale factors for the elements of D must satisfy

$$K_{d_{1j}} = \frac{K_{d}}{c} = K_d$$

(4.78)

Therefore, they must all be equal. By requiring that

$$K_B K_C = \sum_{k=1}^{3} \max |a_{k1}| + \max |c_k| \leq c \text{ for all } i, j$$

(4.79)

only one $\Delta d$ pulse is produced during each computation cycle.

Table 4.6 presents a computation sequence for additions to the $\Delta d$ registers where the elements of A and C are stored serially on respective single registers in the relative sequence shown.
FIG. 4-14 $\Delta d_0$ computer.
FIG. 4-15 $\Delta D$ computer.
TABLE 4.6

\~D Computation Sequence

| Intervals | 1   | 2   | 3   | 4   | 5   | 6   | 7   | 8   | 9   | 10  | 11  | 12  | 13  | 14  | 15  | 16  | 17  | 18  |
|-----------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| d_1       | a_11| a_21| a_31| a_12| a_22| a_32| a_13| a_23| a_33| a_14| a_24| a_34| a_15| a_25| a_35| a_16| a_26| a_36|
| d_2       | a_41| a_51| a_61| a_42| a_52| a_62| a_43| a_53| a_63| a_44| a_54| a_64| a_45| a_55| a_65| a_46| a_56| a_66|
| d_3       | a_71| a_81| a_91| a_72| a_82| a_92| a_73| a_83| a_93| a_74| a_84| a_94| a_75| a_85| a_95| a_76| a_86| a_96|
| d_4       | a_101|a_111|a_121|a_102|a_112|a_122|a_103|a_113|a_123|a_104|a_114|a_124|a_105|a_115|a_125|a_106|a_116|a_126|
| d_5       | a_131|a_141|a_151|a_132|a_142|a_152|a_133|a_143|a_153|a_134|a_144|a_154|a_135|a_145|a_155|a_136|a_146|a_156|
| d_6       | a_171|a_181|a_191|a_172|a_182|a_192|a_173|a_183|a_193|a_174|a_184|a_194|a_175|a_185|a_195|a_176|a_186|a_196|
| d_7       | a_201|a_211|a_221|a_202|a_212|a_222|a_203|a_213|a_223|a_204|a_214|a_224|a_205|a_215|a_225|a_206|a_216|a_226|
| d_8       | a_231|a_241|a_251|a_232|a_242|a_252|a_233|a_243|a_253|a_234|a_244|a_254|a_235|a_245|a_255|a_236|a_246|a_256|
| d_9       | a_261|a_271|a_281|a_262|a_272|a_282|a_263|a_273|a_283|a_264|a_274|a_284|a_265|a_275|a_285|a_266|a_276|a_286|
| d_10      | a_291|a_301|a_311|a_292|a_302|a_312|a_293|a_303|a_313|a_294|a_304|a_314|a_295|a_305|a_315|a_296|a_306|a_316|
| d_11      | a_321|a_331|a_341|a_322|a_332|a_342|a_323|a_333|a_343|a_324|a_334|a_344|a_325|a_335|a_345|a_326|a_336|a_346|
| d_12      | a_351|a_361|a_371|a_352|a_362|a_372|a_353|a_363|a_373|a_354|a_364|a_374|a_355|a_365|a_375|a_356|a_366|a_376|
| d_13      | a_381|a_391|a_401|a_382|a_392|a_402|a_383|a_393|a_403|a_384|a_394|a_404|a_385|a_395|a_405|a_386|a_396|a_406|
| d_14      | a_411|a_421|a_431|a_412|a_422|a_432|a_413|a_423|a_433|a_414|a_424|a_434|a_415|a_425|a_435|a_416|a_426|a_436|
| d_15      | a_441|a_451|a_461|a_442|a_452|a_462|a_443|a_453|a_463|a_444|a_454|a_464|a_445|a_455|a_465|a_446|a_456|a_466|
| d_16      | a_471|a_481|a_491|a_472|a_482|a_492|a_473|a_483|a_493|a_474|a_484|a_494|a_475|a_485|a_495|a_476|a_486|a_496|
| d_17      | a_501|a_511|a_521|a_502|a_512|a_522|a_503|a_513|a_523|a_504|a_514|a_524|a_505|a_515|a_525|a_506|a_516|a_526|
| d_18      | a_531|a_541|a_551|a_532|a_542|a_552|a_533|a_543|a_553|a_534|a_544|a_554|a_535|a_545|a_555|a_536|a_546|a_556|
4.3.8 Au Computation

The Au computer performs the divisions of equations (4.49) by multiplications in the feedback paths, as illustrated in Figure 4-16.

When the number stored in the \( \tilde{e}_1 \) register is positive, the element identified by the letter S in the figure (and inappropriately but conventionally termed a "servo"), produces positive Au pulses at the computation cycle rate. When \( \tilde{e}_1 \) is negative, the servo produces negative Au pulses. No pulses are produced when \( \tilde{e}_1 \) is zero.

When a positive \( \Delta \omega_{cx} \) pulse occurs, the number \( K_{I_x} I_x \) which is scaled to be nearly as large as \( c \), is added to the \( \tilde{e}_1 \) register. Similarly, a positive \( \Delta d_o \) pulse causes \( K_u u_1 \) to be added to \( \tilde{e}_1 \). The resulting \( \Delta u_1 \) pulses produced cause \( K_d d_o \) to be repeatedly subtracted from \( \tilde{e}_1 \) until \( \tilde{e}_1 \) is zero. (The register may not go to zero exactly, in which case the servo will generate alternate positive and negative pulses.)

The change in \( \tilde{e}_1 \) due to \( \Delta \omega_{cx} \) pulses or \( \Delta d_o \) pulses can be expressed as

\[
\Delta \tilde{e}_1 = K_{I_x} I_x \Delta \omega_{cx} - K_u u_1 \Delta d_o - K_d d_o \Delta u_1
\]

(4.80)

\[
\Delta \tilde{e}_1 = \Delta \left[ \left( K_{I_x} I_x \right) \left( K_{\omega_{cx}} \right) \right] - \Delta \left[ \left( K_u u_1 \right) \left( K_{d_o} d_o \right) \right] .
\]

(4.81)

Since \( \tilde{e}_1 = 0 \), and with proper initialization of the \( d_o \) and \( u_1 \) registers,

\[
u_1 = \frac{K_{I_x} K_{\omega_{cx}}}{K_u K_{d_o}} \frac{I_x \omega_{cx}}{d_o} .
\]

(4.82)

To satisfy equations (4.49), the scale factors must therefore satisfy

\[
\frac{K_{I_x} K_{\omega}}{K_u K_{d_o}} = 1 .
\]

(4.83)

Similar requirements apply to the y and z axes.

To ensure that the \( e_1 \) register does not overflow, the scale factors should also satisfy

\[
K_{I_x} I_x + K_u \max |u_1| + K_{d_o} \max |d_o| \leq c .
\]

(4.84)
FIG. 4-16 $\Delta u$ computer.
Table 4.7 presents a computation sequence for \( \tilde{\xi}_1, \tilde{\xi}_2, \) and \( \tilde{\xi}_3 \), where \( u_1, u_2, \) and \( u_3 \) are stored serially on a single register as shown, and where \( d_0 \) is stored on a single register that recycles at the rate of each element of \( u \). The computation is completed in two cycles of the \( u \) register.

**TABLE 4.7**

<table>
<thead>
<tr>
<th>Interval:</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>( u_1 )</td>
<td>( u )</td>
<td>( u_2 )</td>
<td>( u_3 )</td>
<td>( u )</td>
<td>( u_2 )</td>
<td>( u_3 )</td>
</tr>
<tr>
<td>( d_0 )</td>
<td>( d_0 )</td>
<td>( d_0 )</td>
<td>( d_0 )</td>
<td>( d_0 )</td>
<td>( d_0 )</td>
<td>( d_0 )</td>
</tr>
<tr>
<td>( \tilde{\xi}_1 )</td>
<td>( u \Delta d_0 )</td>
<td>( u \Delta d_0 )</td>
<td>( d_0 \Delta u_1 )</td>
<td>( d_0 \Delta u_2 )</td>
<td>( d_0 \Delta u_3 )</td>
<td></td>
</tr>
<tr>
<td>( \tilde{\xi}_2 )</td>
<td>( u \Delta d_0 )</td>
<td>( u \Delta d_0 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \tilde{\xi}_3 )</td>
<td>( u \Delta d_0 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**4.3.9 \( \Delta a_c \) Computation**

The \( \Delta a_c \) computer, illustrated in Figure 4.17, computes increments for the desired outputs of the steering law computer [equations (4.50)]. As in the previous cases, the scale factor for the registers that accumulate the \( \Delta a_c \) pulses (not shown) must satisfy

\[
K_a = \frac{K_\xi K_t}{c} \quad (4.85)
\]

By requiring that

\[
K_a \frac{3}{2} \max_j \left| d_{1j} \right| + \max_j \left| u_j \right| \leq c \quad \text{for each } j
\]

(4.86)

only one \( \Delta a_c \) register overflow can occur during each computation cycle.

Table 4.8 shows a computation sequence for the \( \Delta a_c \) registers, where the elements of \( D \) and \( u \) are each stored serially on single long registers in the sequence shown.

Since \( \Delta a_c \) is most likely desired in analog form for the CMG gimbal control system, the \( \Delta a_c \) pulses may be fed into counters that are parts of a D/A conversion system.
FIG. 4-17 $\Delta \alpha_c$ computer.
| Interval: | 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9  | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| $D_1$    | $d_{11}$ | $d_{12}$ | $d_{13}$ | $d_{21}$ | $d_{22}$ | $d_{23}$ | $d_{31}$ | $d_{32}$ | $d_{33}$ | $d_{41}$ | $d_{42}$ | $d_{43}$ | $d_{51}$ | $d_{52}$ | $d_{53}$ | $d_{61}$ | $d_{62}$ | $d_{63}$ |
| $u_1$    | $u_1$ | $u_2$ | $u_3$ | $u_1$ | $u_2$ | $u_3$ | $u_1$ | $u_2$ | $u_3$ | $u_1$ | $u_2$ | $u_3$ | $u_1$ | $u_2$ | $u_3$ |
| $\tilde{a}_{d1}$ | $\tilde{a}_{d1}$ | $\tilde{a}_{d2}$ | $\tilde{a}_{d3}$ | $\tilde{a}_{d4}$ | $\tilde{a}_{d5}$ | $\tilde{a}_{d6}$ | $\tilde{a}_{d7}$ | $\tilde{a}_{d8}$ | $\tilde{a}_{d9}$ | $\tilde{a}_{d10}$ | $\tilde{a}_{d11}$ | $\tilde{a}_{d12}$ | $\tilde{a}_{d13}$ | $\tilde{a}_{d14}$ | $\tilde{a}_{d15}$ | $\tilde{a}_{d16}$ | $\tilde{a}_{d17}$ | $\tilde{a}_{d18}$ |
4.4 OTHER COMPUTATIONS

4.4.1 Triad Conversion

The triad converter changes the six rate signals \( r_s = \{ r_{s1}, \ldots, r_{s6} \} \) from the dodecahedron configured rate gyros, described in paragraph 4.2.3, to vehicle rates \( \omega_s = \{ \omega_{sx}, \omega_{sy}, \omega_{sz} \} \). The equations to be computed are given by equations (4.21) and (4.22) and are repeated here for convenience:

\[
\omega_s = E^T r_s
\]

where

\[
E^T = (E^T E)^{-1} E^T
\]

\[
E = \lambda \bar{E}
\]

\[
\bar{E} = \begin{bmatrix}
    s & 0 & c \\
   -s & 0 & c \\
     s & c & 0 \\
     s & -s & 0 \\
     0 & c & s \\
     0 & c & -s \\
\end{bmatrix}
\]

\[
\lambda = \text{diag} \{ \lambda_1, \ldots, \lambda_6 \}
\]

\[
s = \sin \theta \approx 0.526
\]

\[
c = \cos \theta \approx 0.850
\]

If rate gyro number 1 is used, \( \lambda_1 = 1 \); if it is switched off because it has failed, \( \lambda_1 = 0 \). Let

\[
F = E^T \bar{E}
\]

\[
P = \text{adj} F
\]

\[
Q = PE^T
\]

\[
q_0 = \det F
\]

Then

\[
\omega_s = \frac{1}{q_0} Q r_s
\]
Since $E$ changes with $\lambda$, the $Q$ matrix can be recomputed by performing these computations each time $\lambda$ changes. This method requires minimum memory and is probably the preferred method with a general purpose computer. The elements of $Q$ can also be expressed in terms of the elements of $\lambda$. This method is preferred for incremental and analog computation, and is described as follows.

Let

$$
\mu_1 = 2s^4c^\lambda \\
\mu_2 = 2s^5 + \mu_1 \\
\mu_3 = \mu_1 + \mu_2 \\
\mu_4 = \mu_2 + \mu_3
$$

Then the 18 elements of $Q$ are given by the following expressions:

$$
q_{11} = \mu_2 + \lambda_{123} + \mu_{12} + \mu_{125} \\
\mu_2 + \lambda_{123} + \mu_{12} + \mu_{125} \\
\mu_2 + \lambda_{123} + \mu_{12} + \mu_{125} \\
\mu_2 + \lambda_{123} + \mu_{12} + \mu_{125}
$$

$$
q_{12} = \mu_2 + \lambda_{123} + \mu_{12} + \mu_{125} \\
\mu_2 + \lambda_{123} + \mu_{12} + \mu_{125} \\
\mu_2 + \lambda_{123} + \mu_{12} + \mu_{125} \\
\mu_2 + \lambda_{123} + \mu_{12} + \mu_{125}
$$

$$
q_{13} = \mu_2 + \lambda_{123} + \mu_{12} + \mu_{125} \\
\mu_2 + \lambda_{123} + \mu_{12} + \mu_{125} \\
\mu_2 + \lambda_{123} + \mu_{12} + \mu_{125} \\
\mu_2 + \lambda_{123} + \mu_{12} + \mu_{125}
$$

$$
q_{14} = \mu_2 + \lambda_{123} + \mu_{12} + \mu_{125} \\
\mu_2 + \lambda_{123} + \mu_{12} + \mu_{125} \\
\mu_2 + \lambda_{123} + \mu_{12} + \mu_{125} \\
\mu_2 + \lambda_{123} + \mu_{12} + \mu_{125}
$$

$$
\lambda_{14k} = \lambda_{14} + \mu_{12} + \mu_{125} \\
\lambda_{14k} = \lambda_{14} + \mu_{12} + \mu_{125} \\
\lambda_{14k} = \lambda_{14} + \mu_{12} + \mu_{125} \\
\lambda_{14k} = \lambda_{14} + \mu_{12} + \mu_{125}
$$

- 82 -
\[ q_{15} = -\mu_3 \lambda_{135} + \mu_2 \lambda_{145} - \mu_3 \lambda_{156} \]
\[ -\mu_2 \lambda_{235} + \mu_3 \lambda_{245} + \mu_3 \lambda_{256} \]
\[ -\mu_2 \lambda_{356} + \mu_2 \lambda_{456} \]
\[ q_{16} = -\mu_2 \lambda_{136} + \mu_3 \lambda_{146} + \mu_3 \lambda_{156} \]
\[ -\mu_3 \lambda_{236} + \mu_2 \lambda_{246} - \mu_3 \lambda_{256} \]
\[ -\mu_2 \lambda_{356} + \mu_2 \lambda_{456} \]
\[ q_{21} = -\mu_3 \lambda_{123} + \mu_3 \lambda_{124} - \mu_2 \lambda_{125} \]
\[ +\mu_2 \lambda_{126} - \mu_3 \lambda_{135} + \mu_2 \lambda_{136} \]
\[ -\mu_2 \lambda_{145} + \mu_3 \lambda_{146} \]
\[ q_{22} = \mu_3 \lambda_{123} + \mu_3 \lambda_{124} - \mu_2 \lambda_{125} \]
\[ +\mu_2 \lambda_{126} - \mu_2 \lambda_{235} + \mu_3 \lambda_{236} \]
\[ -\mu_3 \lambda_{245} + \mu_2 \lambda_{246} \]
\[ q_{23} = 2\mu_2 \lambda_{123} + \mu_4 \lambda_{134} + \mu_2 \lambda_{135} \]
\[ -\mu_1 \lambda_{136} + \mu_4 \lambda_{234} - \mu_1 \lambda_{235} \]
\[ +\mu_2 \lambda_{236} + \mu_2 \lambda_{345} + \mu_2 \lambda_{346} \]
\[ q_{24} = -2\mu_2 \lambda_{124} - \mu_4 \lambda_{134} - \mu_1 \lambda_{145} \]
\[ -\mu_2 \lambda_{146} - \mu_4 \lambda_{234} - \mu_2 \lambda_{245} \]
\[ +\mu_1 \lambda_{246} - \mu_2 \lambda_{345} - \mu_2 \lambda_{346} \]
\[ q_{25} = 2\mu_3 \lambda_{125} + \mu_4 \lambda_{135} + \mu_3 \lambda_{145} \]
\[ +\mu_1 \lambda_{156} + \mu_3 \lambda_{235} + \mu_4 \lambda_{245} \]
\[ +\mu_1 \lambda_{256} + \mu_3 \lambda_{356} + \mu_3 \lambda_{456} \]

(4.97)
\( q_{26} = 2\mu_3 \lambda_{126} + \mu_3 \lambda_{136} + \mu_4 \lambda_{146} \\
+ \mu_3 \lambda_{156} + \mu_1 \lambda_{236} + \mu_3 \lambda_{246} \\
+ \mu_1 \lambda_{256} + \mu_3 \lambda_{356} + \mu_3 \lambda_{456} \\
q_{31} = \mu_1 \lambda_{123} + \mu_1 \lambda_{124} + \mu_3 \lambda_{135} \\
+ \mu_3 \lambda_{126} + 2\mu_3 \lambda_{134} + \mu_1 \lambda_{135} \\
- \mu_3 \lambda_{136} + \mu_3 \lambda_{145} + \mu_4 \lambda_{146} \\
q_{32} = \mu_1 \lambda_{123} + \mu_1 \lambda_{124} + \mu_3 \lambda_{125} \\
+ \mu_3 \lambda_{126} + 2\mu_3 \lambda_{234} + \mu_3 \lambda_{235} \\
+ \mu_1 \lambda_{236} + \mu_1 \lambda_{245} + \mu_3 \lambda_{246} \\
q_{33} = -\mu_2 \lambda_{134} - \mu_3 \lambda_{135} - \mu_2 \lambda_{136} \\
+ \mu_2 \lambda_{234} + \mu_2 \lambda_{235} + \mu_3 \lambda_{236} \\
- \mu_3 \lambda_{345} + \mu_3 \lambda_{346} \\
q_{34} = -\mu_2 \lambda_{134} - \mu_2 \lambda_{145} - \mu_3 \lambda_{146} \\
+ \mu_2 \lambda_{234} + \mu_3 \lambda_{245} + \mu_2 \lambda_{246} \\
+ \mu_3 \lambda_{345} - \mu_3 \lambda_{346} \\
q_{35} = \mu_2 \lambda_{135} - \mu_1 \lambda_{145} + \mu_2 \lambda_{156} \\
- \mu_1 \lambda_{235} + \mu_2 \lambda_{245} + \mu_2 \lambda_{256} \\
+2\mu_2 \lambda_{345} + \mu_4 \lambda_{356} + \mu_4 \lambda_{456} \\
q_{36} = \mu_1 \lambda_{136} - \mu_2 \lambda_{146} - \mu_2 \lambda_{156} \\
- \mu_2 \lambda_{236} + \mu_1 \lambda_{246} - \mu_2 \lambda_{256} \\
-2\mu_2 \lambda_{346} - \mu_4 \lambda_{356} - \mu_4 \lambda_{456} \)
To express $q_o$ in a similar manner, let

$$\mu_5 = 4s^2 c^2 = (s^3 - s^3)^3$$

$$\mu_6 = 4s^2 c^4 = (s^3 + s^3)^2$$  \hspace{1cm} (4.99)

Then

$$q_o = \mu_5^{123} + \mu_5^{124} + \mu_6^{125}$$

$$+ \mu_6^{136} + \mu_6^{145} + \mu_6^{135}$$

$$+ \mu_5^{136} + \mu_5^{145} + \mu_6^{146}$$

$$+ \mu_5^{156} + \mu_6^{234} + \mu_5^{135}$$

$$+ \mu_6^{236} + \mu_6^{245} + \mu_5^{246}$$

$$+ \mu_5^{256} + \mu_5^{345} + \mu_6^{346}$$

$$+ \mu_6^{356} + \mu_6^{456}$$ \hspace{1cm} (4.100)

All of the preceding elements are sums of the constants $\{\mu_1, \ldots, \mu_6\}$ weighted by $\lambda$'s that are either one or zero. A possible method for computing $\omega_s$ [equation (4.91)] by the incremental method described in paragraph 4.3 is to initially store the 16 $q$'s and the $q_o$ that correspond to no failures in the $Y$ registers; then when a failure occurs, change the $q$'s in accordance with the preceding equations. Only the six constants $\{\mu_1, \ldots, \mu_6\}$ need to be stored to accomplish the required changes in the $q$'s and $q_o$. The incremental computation for $\Delta r_s$ will then be similar to the $\omega u$ and $\Delta ac$ computations described in paragraph 4.3.

### 4.4.2 Control Compensation, Failure Monitoring, and Mode Control

The three compensators required for the vehicle rate loop (see Figure 4-6) can readily be accomplished by incremental computation. The application of this computation technique to flight controls, for example, has been studied by several flight control system manufacturers, including Sperry. Due to the limited scope of this study, this section of the CMG control computer will not be investigated.
The failure monitoring and mode control function of the CMG control computer (Figure 4-6) generates the λ's for the triad conversion; the rotor speeds Ω for the steering law computer; and supplies the display signals required to monitor the failure status of the CMG's and sensors. The major computations required for this function are the parity check equations for the sensors listed in Table 4.1. A computer structure for the computations required by the failure monitoring and mode control function is not presented in this study due to its limited scope.

4.5 CONCLUSIONS AND RECOMMENDATIONS FOR FURTHER STUDY

The steering law computations are by far the most complex computations required by a CMG control computer, and they outweigh the remaining computing functions in selecting the type of computer to employ for controlling CMG's. Since these computations are time-variant and nonlinear, digital computation is required. A general purpose computer can certainly perform these computations but it is considerably more limited in speed than an incremental computer. The time required to do the steering law computations on a UNIVAC® 1109 computer (designed for airborne use) is estimated to be 6.37 milliseconds. If we roughly estimate that the time required to perform the total CMG control computer computations is 10 milliseconds, then a sampling rate of 100 samples per second saturates the computer. For some of the anticipated requirements for future CMG control systems, sampling rates of this order are required. It can therefore be concluded that the use of a central, general purpose computer, which is also required to perform other than CMG control functions, is not feasible for the type of CMG system described in this report.

An incremental computer can be made much faster than a general purpose computer, but its speed depends on the level of serialization of its computations. A bit-time of 0.5 microsecond is easily accomplished, and if a 300-bit register is used to serially store a set of 18 Y variables (such as the 18 a's or d's), a computation cycle takes 0.15 millisecond. Of course, there are also lags in incremental computers that must be considered when comparing them with the general purpose type, but it is clear that incremental computers are much faster. Also, since computations are performed in parallel, adding functions to an incremental computer does not require additional computation time.

In terms of complexity, it is not obvious at this stage which type of computer is the simplest. Further study would be required to determine parts counts. They may be quite competitive in this aspect.
Since incremental CMG control computers have some definite advantages over general purpose computers for this application, it is recommended that a further study be initiated to design a prototype incremental computer for a specific CMG control system requirement, and to compare it with a general purpose type, programmed for the same function, in terms of complexity (parts count), accuracy, reliability, weight, and power consumption.

REFERENCES


5 Ibid.

6 Deutsch, p 83.

7 Ibid, pp 83 - 84.
5.1 DESCRIPTION OF METHOD AND RESULTS

The purpose of the simulation of SIMU operation was to determine the error accumulation for large angle changes in position and also the worst-case mode of error accumulation during extended runs.

5.1.1 Types of Runs

Three types of runs were made and are described below.

(1) Runs simulating a motion to a given angle and then back to the zero position. The angles used range from less than $10^\circ$ to $180^\circ$.

(2) Oscillatory runs taken near the zero reference position where the curve of data from the test above indicates that the worst error accumulation occurs.

(3) Three-axis rotations where the inputs are: (a) up to one radian for $x$, followed by one radian for $y$ and then one radian for $z$, followed in reverse order for the return to zero; (b) single pulse increments in $x$, $y$ and $z$ followed successively until each axis has had the pulse required to read up to one radian rotation (32,768), then the reverse order of the same increments of $x$, $y$ and $z$ until zero is reached.

5.1.2 Test Results

The results of these tests show the following:

Figure 5-1 shows that error accumulations in direction cosine values after rotation to a particular angle (of $180^\circ$ or less) and then returning to zero degrees were limited to a maximum value of approximately one part in 37. Refer also to Table 5.1.

The error moves up steeply until approximately 2000 unidirectional input pulses have been accumulated. This is approximately a range of one bit of error for every two pulses fed the system by the time the rotation back to zero is completed. (See steep initial slope on Fig 5-1.)

The three axis runs were made for continuous inputs of a fixed number of pulses for $x$, $y$ and then $z$ sequentially. They were also made for runs where the input pulses were increments of $\Delta x$, $\Delta y$ and $\Delta z$ repeated in that order until the required number of sets of pulses were processed. Thereafter the reverse sequences of pulses were given to return the system to the zero reference. Table 5.2 lists the above data plus that for each set of two axis updates for 8, 128 and 32,768 sets of input pulses, the value 22,768 approximating one radian of rotation. The results show the order of magnitude of error for the number of pulses specified to be the same as for single axis operation. However, as is expected, they are not identical values.
Plot of error in repeatability vs magnitude of excursion

Incremental inputs for positive rotation

*E is error in direction cosine reading (in octal \times 2^{-48}) incurred in rotation by indicated number of pulses and then same number of negative pulses in order to return to zero.

FIG. 5-1 Plot of error in repeatability.
The oscillatory runs consisted of ten cycles from 0 to 1024 pulses and back to determine the effect of extended testing on the maximum error rate that the system initially produced. (See Fig. 5-1.) The result was that the error increased proportionally for as many input pulses as were applied. See Table 5.3

These tests were run to provide a reference to determine the correct operation of the SIMU.

**TABLE 5.1**

SIMU Simulation Data

**Single Axis Values After n(Δθ) Pulses**

<table>
<thead>
<tr>
<th>n</th>
<th>C11</th>
<th>C13</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,024</td>
<td>037770</td>
<td>000052</td>
</tr>
<tr>
<td>2,048</td>
<td>037740</td>
<td>001252</td>
</tr>
<tr>
<td>4,096</td>
<td>037600</td>
<td>025245</td>
</tr>
<tr>
<td>8,192</td>
<td>037002</td>
<td>124477</td>
</tr>
<tr>
<td>32,768</td>
<td>021224</td>
<td>050037</td>
</tr>
<tr>
<td>106,496</td>
<td>140140</td>
<td>026754</td>
</tr>
</tbody>
</table>

**Single Axis Return to Zero Values After n(Δθ) Pulses and n(-Δθ)**

<table>
<thead>
<tr>
<th>n</th>
<th>C11</th>
<th>C13</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,024</td>
<td>040000</td>
<td>000000</td>
</tr>
<tr>
<td>2,048</td>
<td>040000</td>
<td>000000</td>
</tr>
<tr>
<td>4,096</td>
<td>040000</td>
<td>000000</td>
</tr>
<tr>
<td>8,192</td>
<td>040000</td>
<td>000000</td>
</tr>
<tr>
<td>12,288</td>
<td>040000</td>
<td>000000</td>
</tr>
<tr>
<td>16,384</td>
<td>040000</td>
<td>000000</td>
</tr>
<tr>
<td>32,768</td>
<td>040000</td>
<td>000000</td>
</tr>
<tr>
<td>65,536</td>
<td>040000</td>
<td>000000</td>
</tr>
<tr>
<td>98,304</td>
<td>040000</td>
<td>000000</td>
</tr>
<tr>
<td>106,496</td>
<td>040000</td>
<td>000000</td>
</tr>
</tbody>
</table>

5.2 PROGRAM DESCRIPTION

The program, written for the UNIVAC 1108 in Fortran IV, computes new direction cosines from old direction cosines using the truncation-error-free algorithm. See the flow chart (Fig. 5-2) to determine how this simulation was programmed.

The nature of the iterative process is controlled by punched card inputs in the format described below. Provision was made for controlling iteration by selecting angular increments in each of the three axial senses, x̂, ẑ and ẑ. Let the three selection variables be DX, DY, DZ. DX ≠ 0 means skip an angular increment in this sense. DX ≠ 0 means perform the appropriate iteration for an angular increment in this sense (this angular increment assumed fixed). Refer to Fig. 5-3.
Input and Control Initialization

FIG. 5-2 Flow charts of simulation program.
Choice of Angular Increment

FIG. 5-2 Flow charts of simulation program (cont.).
ITER is the subroutine that actually performs the computation on the pair of direction cosines that are the arguments in the call statement. ITER itself calls another routine SCALE which scales the 3 integer variables of each $C_{ij}$ to lie within 0 and $2^{16} - 1$.

The variable $Z$ is set to +1 for positive increments and -1 for negative increments.

Printing is done after each call to iter depending on the control variable PR.

Computation for New Direction Cosines

FIG. 5-2 Flow charts of simulation program (cont.).

- 93 -
arguments: \((S, C)\)

*S* represents the three integer variables holding the three 16-bit parts of the 1st direction cosine, and \(C\) is used to represent the 2nd. \(JFS, JEC, \text{etc.}\), represent corresponding high order bits for each part.

Computation is achieved by adding or subtracting the parts of \(S, C, S\times2^{-32}, C\times2^{-32}, \text{etc.}\), that are of corresponding order. The high order bits are used to perform rounding and "shifting". See program for details.

\(S', C', S'', C''\) represent intermediary results which are in the same format as \(S\) and \(C\).

Actual Computation of New Direction Cosines

FIG. 5-2 Flow charts of simulation program (cont.)

- 94 -
The direction cosine \( S \) is represented by 3 integer variables, \( S_{\text{high}} \), \( S_{\text{med}} \), \( S_{\text{low}} \).

Scaling of 16-Bit Parts of Direction Cosine

*FIG. 5-2* Flow charts of simulation program (cont.).
Two further variables, LIM1 and LIM2, control the length of the iterative procedure. Two specific settings of LIM1 and LIM2 are of interest:

1. LIM1=1, LIM2=K means iterate K times in the first selected axial sense, then K times in the second, etc.

2. LIM1=K, LIM2=1 means perform K times the sequence (one iteration in the first selected sense, followed by one iteration in the second selected sense) etc.

The program performs the "mirror-image" of the iteration sequence in order to return to the starting point.

One further variable, PR, is provided to control print frequency. If PR=K, then each Nth iteration in any selected axial sense forces output of all the direction cosines.

Input variables C11, C12, C13 are provided to enable the user to specify different initial direction cosines. C11, C12, C13 are always entered and printed out as 6 digit octal numbers.

The program performs only integer arithmetic, using one Fortran integer variable to hold the three 16-bit parts of each 48-bit $c_{ij}$, as well as intermediary results (see Table 5.4).

Card Column

<table>
<thead>
<tr>
<th>Card</th>
<th>Column</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>DX</td>
</tr>
<tr>
<td>110</td>
<td>110</td>
</tr>
</tbody>
</table>

110 = integer, field width of 10

Card 2

<table>
<thead>
<tr>
<th>Card</th>
<th>Column</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>PR</td>
</tr>
<tr>
<td>110</td>
<td>110</td>
</tr>
</tbody>
</table>

Card 3-5

<table>
<thead>
<tr>
<th>Card</th>
<th>Column</th>
</tr>
</thead>
<tbody>
<tr>
<td>3-5</td>
<td>$c_{11}$</td>
</tr>
<tr>
<td></td>
<td>$\theta F$</td>
</tr>
</tbody>
</table>

$\theta F$ = octal integer, field width of F

Card 6

<table>
<thead>
<tr>
<th>Card</th>
<th>Column</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>DX'</td>
</tr>
<tr>
<td>110</td>
<td>110</td>
</tr>
</tbody>
</table>

At the end of a sequence specified by one set of data, the program reads a new DX', DY', DZ'. If DX' = -1 the program stops, otherwise, computation resumes with a new set of data which is assumed to follow card 6, etc.

FIG. 5-3 Card input format.
Single axis - 0 to 8 pulses and return to 0 again.

<table>
<thead>
<tr>
<th>0</th>
<th>24</th>
<th>28</th>
<th>32</th>
<th>40</th>
<th>48</th>
<th>56</th>
<th>64</th>
<th>72</th>
<th>80</th>
<th>88</th>
<th>96</th>
<th>104</th>
<th>112</th>
<th>120</th>
</tr>
</thead>
<tbody>
<tr>
<td>= 0</td>
<td>= 0</td>
<td>= 0</td>
<td>= 0</td>
<td>= 0</td>
<td>= 0</td>
<td>= 0</td>
<td>= 0</td>
<td>= 0</td>
<td>= 0</td>
<td>= 0</td>
<td>= 0</td>
<td>= 0</td>
<td>= 0</td>
<td>= 0</td>
</tr>
<tr>
<td>= 0</td>
<td>= 0</td>
<td>= 352</td>
<td>= 26760</td>
<td>= 12580</td>
<td>= 16622</td>
<td>= 55504</td>
<td>= 175600</td>
<td>= 17734</td>
<td>= 171600</td>
<td>= 172740</td>
<td>= 174740</td>
<td>= 174000</td>
<td>= 174740</td>
<td>= 176340</td>
</tr>
<tr>
<td>= 0</td>
<td>= 0</td>
<td>= 37520</td>
<td>= 377777</td>
<td>= 377777</td>
<td>= 377777</td>
<td>= 377777</td>
<td>= 377777</td>
<td>= 377777</td>
<td>= 377777</td>
<td>= 377777</td>
<td>= 377777</td>
<td>= 377777</td>
<td>= 377777</td>
<td>= 377777</td>
</tr>
<tr>
<td>= 0</td>
<td>= 0</td>
<td>= 0</td>
<td>= 0</td>
<td>= 0</td>
<td>= 0</td>
<td>= 0</td>
<td>= 0</td>
<td>= 0</td>
<td>= 0</td>
<td>= 0</td>
<td>= 0</td>
<td>= 0</td>
<td>= 0</td>
<td>= 0</td>
</tr>
</tbody>
</table>

Single axis - 0 to 128 pulses and return to 0 again.

**TABLE 5.2**

Results of SIMU Simulation Program for Checkout Runs

<table>
<thead>
<tr>
<th>0</th>
<th>24</th>
<th>28</th>
<th>32</th>
<th>40</th>
<th>48</th>
<th>56</th>
<th>64</th>
<th>72</th>
<th>80</th>
<th>88</th>
<th>96</th>
<th>104</th>
<th>112</th>
<th>120</th>
</tr>
</thead>
<tbody>
<tr>
<td>= 0</td>
<td>= 0</td>
<td>= 0</td>
<td>= 0</td>
<td>= 0</td>
<td>= 0</td>
<td>= 0</td>
<td>= 0</td>
<td>= 0</td>
<td>= 0</td>
<td>= 0</td>
<td>= 0</td>
<td>= 0</td>
<td>= 0</td>
<td>= 0</td>
</tr>
</tbody>
</table>
Single axis - 0 to 32,768 pulses and return to 0 again.

Two axis x,y sequentially - 0 to 8 pulses and return to 0 again.

<table>
<thead>
<tr>
<th>1</th>
<th>1</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

TABLE 5.2 (cont.)
<table>
<thead>
<tr>
<th>C</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-7</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>C4000</td>
<td>00000</td>
</tr>
<tr>
<td>3</td>
<td>C4000</td>
<td>00000</td>
</tr>
<tr>
<td>4</td>
<td>C4000</td>
<td>00000</td>
</tr>
<tr>
<td>5</td>
<td>C4000</td>
<td>00000</td>
</tr>
<tr>
<td>6</td>
<td>C4000</td>
<td>00000</td>
</tr>
<tr>
<td>7</td>
<td>C4000</td>
<td>00000</td>
</tr>
</tbody>
</table>

Two axis y,z sequentially - 0 to 6 pulses and return to 0 again.

<table>
<thead>
<tr>
<th>C</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-7</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>C4000</td>
<td>00000</td>
</tr>
<tr>
<td>3</td>
<td>C4000</td>
<td>00000</td>
</tr>
<tr>
<td>4</td>
<td>C4000</td>
<td>00000</td>
</tr>
<tr>
<td>5</td>
<td>C4000</td>
<td>00000</td>
</tr>
<tr>
<td>6</td>
<td>C4000</td>
<td>00000</td>
</tr>
<tr>
<td>7</td>
<td>C4000</td>
<td>00000</td>
</tr>
</tbody>
</table>

Two axis x,z sequentially - 0 to 6 pulses and return to 0 again.

TABLE 5.2 (cont.)
Three axis x,y,z sequentially - 0 to 8 pulses and return to 0 again.

Two axis Δx,Δy incrementally - 0 to 8 pulses and return to 0 again.

<table>
<thead>
<tr>
<th>TABLE 5.2 (cont.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>- 100 -</td>
</tr>
</tbody>
</table>
Two axis $\Delta y, \Delta z$ incrementally - 0 to 8 pulses and return to 0 again.

<table>
<thead>
<tr>
<th>$\Delta y$</th>
<th>$\Delta z$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Two axis $\Delta x, \Delta z$ incrementally - 0 to 8 pulses and return to 0 again.

**TABLE 5.2** (cont.)
Three axis $\Delta x, \Delta y, \Delta z$ incrementally - 0 to 8 pulses and return to 0 again.

<table>
<thead>
<tr>
<th>$\Delta x$</th>
<th>$\Delta y$</th>
<th>$\Delta z$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>10</td>
<td>11</td>
<td>12</td>
</tr>
<tr>
<td>13</td>
<td>14</td>
<td>15</td>
</tr>
<tr>
<td>16</td>
<td>17</td>
<td>18</td>
</tr>
<tr>
<td>19</td>
<td>20</td>
<td>21</td>
</tr>
<tr>
<td>22</td>
<td>23</td>
<td>24</td>
</tr>
<tr>
<td>25</td>
<td>26</td>
<td>27</td>
</tr>
<tr>
<td>28</td>
<td>29</td>
<td>30</td>
</tr>
<tr>
<td>31</td>
<td>32</td>
<td>33</td>
</tr>
<tr>
<td>34</td>
<td>35</td>
<td>36</td>
</tr>
<tr>
<td>37</td>
<td>38</td>
<td>39</td>
</tr>
<tr>
<td>40</td>
<td>41</td>
<td>42</td>
</tr>
<tr>
<td>43</td>
<td>44</td>
<td>45</td>
</tr>
<tr>
<td>46</td>
<td>47</td>
<td>48</td>
</tr>
<tr>
<td>49</td>
<td>50</td>
<td>51</td>
</tr>
<tr>
<td>52</td>
<td>53</td>
<td>54</td>
</tr>
<tr>
<td>55</td>
<td>56</td>
<td>57</td>
</tr>
<tr>
<td>58</td>
<td>59</td>
<td>60</td>
</tr>
<tr>
<td>61</td>
<td>62</td>
<td>63</td>
</tr>
<tr>
<td>64</td>
<td>65</td>
<td>66</td>
</tr>
<tr>
<td>67</td>
<td>68</td>
<td>69</td>
</tr>
<tr>
<td>70</td>
<td>71</td>
<td>72</td>
</tr>
<tr>
<td>73</td>
<td>74</td>
<td>75</td>
</tr>
<tr>
<td>76</td>
<td>77</td>
<td>78</td>
</tr>
<tr>
<td>79</td>
<td>80</td>
<td>81</td>
</tr>
<tr>
<td>82</td>
<td>83</td>
<td>84</td>
</tr>
<tr>
<td>85</td>
<td>86</td>
<td>87</td>
</tr>
<tr>
<td>88</td>
<td>89</td>
<td>90</td>
</tr>
<tr>
<td>91</td>
<td>92</td>
<td>93</td>
</tr>
<tr>
<td>94</td>
<td>95</td>
<td>96</td>
</tr>
<tr>
<td>97</td>
<td>98</td>
<td>99</td>
</tr>
<tr>
<td>100</td>
<td>101</td>
<td>102</td>
</tr>
<tr>
<td>103</td>
<td>104</td>
<td>105</td>
</tr>
<tr>
<td>106</td>
<td>107</td>
<td>108</td>
</tr>
<tr>
<td>109</td>
<td>110</td>
<td>111</td>
</tr>
<tr>
<td>112</td>
<td>113</td>
<td>114</td>
</tr>
<tr>
<td>115</td>
<td>116</td>
<td>117</td>
</tr>
<tr>
<td>118</td>
<td>119</td>
<td>120</td>
</tr>
<tr>
<td>121</td>
<td>122</td>
<td>123</td>
</tr>
<tr>
<td>124</td>
<td>125</td>
<td>126</td>
</tr>
<tr>
<td>127</td>
<td>128</td>
<td>129</td>
</tr>
</tbody>
</table>

Two axis, $x, y$ sequentially - 0 to 128 pulses and return to 0 again.

TABLE 5.2 (cont.)

- 102 -
Two axis, \( y, z \) sequentially - 0 to 128 pulses and return to 0 again.

<table>
<thead>
<tr>
<th>( x )</th>
<th>( y )</th>
<th>( z )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>128</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>00000</td>
<td>00000</td>
</tr>
<tr>
<td>16</td>
<td>00000</td>
<td>00000</td>
</tr>
<tr>
<td>32</td>
<td>00000</td>
<td>00000</td>
</tr>
<tr>
<td>48</td>
<td>00000</td>
<td>00000</td>
</tr>
<tr>
<td>64</td>
<td>00000</td>
<td>00000</td>
</tr>
<tr>
<td>80</td>
<td>00000</td>
<td>00000</td>
</tr>
<tr>
<td>96</td>
<td>00000</td>
<td>00000</td>
</tr>
<tr>
<td>112</td>
<td>00000</td>
<td>00000</td>
</tr>
<tr>
<td>128</td>
<td>00000</td>
<td>00000</td>
</tr>
<tr>
<td>16</td>
<td>037777</td>
<td>17760</td>
</tr>
<tr>
<td>32</td>
<td>037777</td>
<td>17760</td>
</tr>
<tr>
<td>48</td>
<td>037777</td>
<td>17760</td>
</tr>
<tr>
<td>64</td>
<td>037777</td>
<td>17760</td>
</tr>
<tr>
<td>80</td>
<td>037777</td>
<td>17760</td>
</tr>
<tr>
<td>96</td>
<td>037777</td>
<td>17760</td>
</tr>
<tr>
<td>112</td>
<td>037777</td>
<td>17760</td>
</tr>
<tr>
<td>128</td>
<td>037777</td>
<td>17760</td>
</tr>
</tbody>
</table>

Two axis, \( x, z \) sequentially - 0 to 128 pulses and return to 0 again.

**TABLE 5.2 (cont.)**
Three axis, x,y,z sequentially - 0 to 128 pulses and return to 0 again.

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>z</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Two axis Δx,Δy incrementally - 0 to 128 pulses and return to 0 again.

TABLE 5.2 (cont.)
Two axis $\omega y, \omega z$ incrementally - 0 to 128 pulses and return to 0 again.

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|}
\hline
L & 1 & 1 & 1 \\
\hline
1 & 1 & 128 & \\
\hline
-12 & \ldots & \ldots & \ldots \\
\hline
-12 & 177417 & 177422 & 177772 \\
\hline
1 & 177422 & 177772 & 177777 \\
\hline
32 & \ldots & \ldots & \ldots \\
\hline
12 & \ldots & \ldots & \ldots \\
\hline
4 & 177777 & 177777 & 177777 \\
\hline
4 & \ldots & \ldots & \ldots \\
\hline
112 & \ldots & \ldots & \ldots \\
\hline
112 & \ldots & \ldots & \ldots \\
\hline
120 & \ldots & \ldots & \ldots \\
\hline
125 & \ldots & \ldots & \ldots \\
\hline
125 & \ldots & \ldots & \ldots \\
\hline
Two axis $\Delta x, \Delta z$ incrementally - 0 to 128 pulses and return to 0 again.
\end{table}

TABLE 5.2 (cont.)

- 105 -
Two axis sequentially - 0 to 32,768 pulses and return to 0 again.

<table>
<thead>
<tr>
<th>Table 5.2 (cont.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Two axis sequentially - 0 to 32,768 pulses and return to 0 again.</td>
</tr>
</tbody>
</table>
TABLE 5.2 (cont.)

<table>
<thead>
<tr>
<th>C</th>
<th>1</th>
<th>T276.c</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>4.936</td>
<td>5.775</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>6.000</td>
<td>6.000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>12.356</td>
<td>12.356</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>15.331</td>
<td>15.331</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>24.576</td>
<td>24.576</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>24.576</td>
<td>24.576</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>32.768</td>
<td>32.768</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>32.768</td>
<td>32.768</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>64.000</td>
<td>64.000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>64.000</td>
<td>64.000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>128.000</td>
<td>128.000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>128.000</td>
<td>128.000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>256.000</td>
<td>256.000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>256.000</td>
<td>256.000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>512.000</td>
<td>512.000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>512.000</td>
<td>512.000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>1.000</td>
<td>1.000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>1.000</td>
<td>1.000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>2.000</td>
<td>2.000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>2.000</td>
<td>2.000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>4.000</td>
<td>4.000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>4.000</td>
<td>4.000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>8.000</td>
<td>8.000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>8.000</td>
<td>8.000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>16.000</td>
<td>16.000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>16.000</td>
<td>16.000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>32.000</td>
<td>32.000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>32.000</td>
<td>32.000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>64.000</td>
<td>64.000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>64.000</td>
<td>64.000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>128.000</td>
<td>128.000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>128.000</td>
<td>128.000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>256.000</td>
<td>256.000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>256.000</td>
<td>256.000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
</tbody>
</table>

Two axis, \( y,z \) sequentially - 0 to 32,768 pulses and return to 0 again.

<table>
<thead>
<tr>
<th>C</th>
<th>1</th>
<th>T276.c</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>4.936</td>
<td>5.775</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>6.000</td>
<td>6.000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>12.356</td>
<td>12.356</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>15.331</td>
<td>15.331</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>24.576</td>
<td>24.576</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>24.576</td>
<td>24.576</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>32.768</td>
<td>32.768</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>32.768</td>
<td>32.768</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>64.000</td>
<td>64.000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>64.000</td>
<td>64.000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>128.000</td>
<td>128.000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>128.000</td>
<td>128.000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>256.000</td>
<td>256.000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>256.000</td>
<td>256.000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>512.000</td>
<td>512.000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>512.000</td>
<td>512.000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>1.000</td>
<td>1.000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>1.000</td>
<td>1.000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>2.000</td>
<td>2.000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>2.000</td>
<td>2.000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>4.000</td>
<td>4.000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>4.000</td>
<td>4.000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>8.000</td>
<td>8.000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>8.000</td>
<td>8.000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>16.000</td>
<td>16.000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>16.000</td>
<td>16.000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>32.000</td>
<td>32.000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>32.000</td>
<td>32.000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>64.000</td>
<td>64.000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>64.000</td>
<td>64.000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>128.000</td>
<td>128.000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>128.000</td>
<td>128.000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>256.000</td>
<td>256.000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>256.000</td>
<td>256.000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
</tbody>
</table>

Two axis \( x,z \) sequentially - 0 to 32,768 pulses and return to 0 again.

- TABLE 5.2 (cont.)

- 107 -
Three axis x, y, z sequentially - 0 to 32,768 pulses and return to 0 again.

Two axis \( \Delta x, \Delta y \) incrementally - 0 to 32,768 pulses and return to 0 again.

<p>| TABLE 5.2 (cont.) | - 109 - |</p>
<table>
<thead>
<tr>
<th>( \Delta y )</th>
<th>( \Delta z )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>256</td>
</tr>
<tr>
<td>2</td>
<td>512</td>
</tr>
<tr>
<td>4</td>
<td>1024</td>
</tr>
<tr>
<td>8</td>
<td>2048</td>
</tr>
<tr>
<td>16</td>
<td>4096</td>
</tr>
<tr>
<td>32</td>
<td>8192</td>
</tr>
<tr>
<td>64</td>
<td>16384</td>
</tr>
<tr>
<td>128</td>
<td>32768</td>
</tr>
</tbody>
</table>

Two axis \( \Delta y, \Delta z \) incrementally - 0 to 32,768 pulses and return to 0 again.

<table>
<thead>
<tr>
<th>( \Delta x )</th>
<th>( \Delta z )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>256</td>
</tr>
<tr>
<td>2</td>
<td>512</td>
</tr>
<tr>
<td>4</td>
<td>1024</td>
</tr>
<tr>
<td>8</td>
<td>2048</td>
</tr>
<tr>
<td>16</td>
<td>4096</td>
</tr>
<tr>
<td>32</td>
<td>8192</td>
</tr>
<tr>
<td>64</td>
<td>16384</td>
</tr>
<tr>
<td>128</td>
<td>32768</td>
</tr>
</tbody>
</table>

Two axis \( \Delta x, \Delta z \) incrementally - 0 to 32,768 pulses and return to 0 again.  

**TABLE 5.2 (cont.)**

- 109 -
Three axis $\Delta x, \Delta y, \Delta z$ incrementally - 0 to 32,768 pulses and return to 0 again.

TABLE 5.2 (cont.)
<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>037777</td>
<td>177777</td>
</tr>
<tr>
<td>1</td>
<td>037777</td>
<td>177777</td>
</tr>
<tr>
<td>2</td>
<td>037777</td>
<td>177777</td>
</tr>
<tr>
<td>3</td>
<td>037777</td>
<td>177777</td>
</tr>
<tr>
<td>4</td>
<td>037777</td>
<td>177777</td>
</tr>
<tr>
<td>5</td>
<td>037777</td>
<td>177777</td>
</tr>
<tr>
<td>6</td>
<td>037777</td>
<td>177777</td>
</tr>
<tr>
<td>7</td>
<td>037777</td>
<td>177777</td>
</tr>
<tr>
<td>8</td>
<td>037777</td>
<td>177777</td>
</tr>
<tr>
<td>9</td>
<td>037777</td>
<td>177777</td>
</tr>
</tbody>
</table>

Three axis $\Delta y, \Delta z, \Delta x$ incrementally - 0 to 8 pulses and return to 0 again.

**TABLE 5.2** (cont.)
<table>
<thead>
<tr>
<th>1</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$0.0000$</td>
<td>$0.0000$</td>
</tr>
<tr>
<td>1</td>
<td>$0.0000$</td>
<td>$0.0000$</td>
</tr>
<tr>
<td>2</td>
<td>$0.0000$</td>
<td>$0.0000$</td>
</tr>
<tr>
<td>3</td>
<td>$0.0000$</td>
<td>$0.0000$</td>
</tr>
<tr>
<td>4</td>
<td>$0.0000$</td>
<td>$0.0000$</td>
</tr>
<tr>
<td>5</td>
<td>$0.0000$</td>
<td>$0.0000$</td>
</tr>
</tbody>
</table>

Three axis $\Delta z, \Delta x, \Delta y$ incrementally - 0 to 6 pulses and return to 0 again.

**TABLE 5.2 (cont.)**

- 112 -
<table>
<thead>
<tr>
<th>( i )</th>
<th>( j )</th>
<th>( k )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>11</td>
</tr>
</tbody>
</table>

Three axes \( \Delta y, \Delta z, \Delta x \) incrementally - 0 to 32,768 pulses and return to 0 again.

**Table 5.2 (cont.)**

- 113 -
Three axis $\Delta z, \Delta x, \Delta y$ incrementally - 0 to 32,768 pulses and return to 0 again.

**TABLE 5.2 (cont.)**

<table>
<thead>
<tr>
<th>$\Delta z$</th>
<th>$\Delta x$</th>
<th>$\Delta y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
</tbody>
</table>

- 114 -
Single Axis - Runs 1-5

**TABLE 5.3**

Oscillatory Runs-Ten Cycles 0 to 1024

Pulses and Return to 0 Again

<table>
<thead>
<tr>
<th>C</th>
<th>0</th>
<th>1</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>512</td>
<td>19374</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>0</td>
<td>040300</td>
<td>000000</td>
<td>040300</td>
</tr>
<tr>
<td>1024</td>
<td>037770</td>
<td>000000</td>
<td>000000</td>
</tr>
<tr>
<td>-512</td>
<td>037770</td>
<td>000000</td>
<td>000000</td>
</tr>
<tr>
<td>-1024</td>
<td>040300</td>
<td>000000</td>
<td>000000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>C</th>
<th>0</th>
<th>1</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>512</td>
<td>377770</td>
<td>000000</td>
<td>037770</td>
</tr>
<tr>
<td>1024</td>
<td>037770</td>
<td>000000</td>
<td>000000</td>
</tr>
<tr>
<td>-512</td>
<td>037770</td>
<td>000000</td>
<td>000000</td>
</tr>
<tr>
<td>-1024</td>
<td>040300</td>
<td>000000</td>
<td>000000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>C</th>
<th>0</th>
<th>1</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>512</td>
<td>377770</td>
<td>000000</td>
<td>037770</td>
</tr>
<tr>
<td>1024</td>
<td>037770</td>
<td>000000</td>
<td>000000</td>
</tr>
<tr>
<td>-512</td>
<td>037770</td>
<td>000000</td>
<td>000000</td>
</tr>
<tr>
<td>-1024</td>
<td>040300</td>
<td>000000</td>
<td>000000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>C</th>
<th>0</th>
<th>1</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>512</td>
<td>377770</td>
<td>000000</td>
<td>037770</td>
</tr>
<tr>
<td>1024</td>
<td>037770</td>
<td>000000</td>
<td>000000</td>
</tr>
<tr>
<td>-512</td>
<td>037770</td>
<td>000000</td>
<td>000000</td>
</tr>
<tr>
<td>-1024</td>
<td>040300</td>
<td>000000</td>
<td>000000</td>
</tr>
</tbody>
</table>

- 115 -
### TABLE 5.3 (cont.)

#### Single Axis - Runs 6-10
Two Axis - Runs 1-5

TABLE 5.3 (cont.)  

- 117 -
<table>
<thead>
<tr>
<th>Run</th>
<th>040000</th>
<th>050000</th>
<th>060000</th>
<th>070000</th>
<th>080000</th>
<th>090000</th>
<th>000000</th>
<th>010000</th>
<th>020000</th>
<th>030000</th>
<th>040000</th>
<th>050000</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>040000</td>
<td>050000</td>
<td>060000</td>
<td>070000</td>
<td>080000</td>
<td>090000</td>
<td>000000</td>
<td>010000</td>
<td>020000</td>
<td>030000</td>
<td>040000</td>
<td>050000</td>
</tr>
<tr>
<td>1</td>
<td>040000</td>
<td>050000</td>
<td>060000</td>
<td>070000</td>
<td>080000</td>
<td>090000</td>
<td>000000</td>
<td>010000</td>
<td>020000</td>
<td>030000</td>
<td>040000</td>
<td>050000</td>
</tr>
<tr>
<td>2</td>
<td>040000</td>
<td>050000</td>
<td>060000</td>
<td>070000</td>
<td>080000</td>
<td>090000</td>
<td>000000</td>
<td>010000</td>
<td>020000</td>
<td>030000</td>
<td>040000</td>
<td>050000</td>
</tr>
<tr>
<td>3</td>
<td>040000</td>
<td>050000</td>
<td>060000</td>
<td>070000</td>
<td>080000</td>
<td>090000</td>
<td>000000</td>
<td>010000</td>
<td>020000</td>
<td>030000</td>
<td>040000</td>
<td>050000</td>
</tr>
<tr>
<td>4</td>
<td>040000</td>
<td>050000</td>
<td>060000</td>
<td>070000</td>
<td>080000</td>
<td>090000</td>
<td>000000</td>
<td>010000</td>
<td>020000</td>
<td>030000</td>
<td>040000</td>
<td>050000</td>
</tr>
<tr>
<td>5</td>
<td>040000</td>
<td>050000</td>
<td>060000</td>
<td>070000</td>
<td>080000</td>
<td>090000</td>
<td>000000</td>
<td>010000</td>
<td>020000</td>
<td>030000</td>
<td>040000</td>
<td>050000</td>
</tr>
<tr>
<td>6</td>
<td>040000</td>
<td>050000</td>
<td>060000</td>
<td>070000</td>
<td>080000</td>
<td>090000</td>
<td>000000</td>
<td>010000</td>
<td>020000</td>
<td>030000</td>
<td>040000</td>
<td>050000</td>
</tr>
<tr>
<td>7</td>
<td>040000</td>
<td>050000</td>
<td>060000</td>
<td>070000</td>
<td>080000</td>
<td>090000</td>
<td>000000</td>
<td>010000</td>
<td>020000</td>
<td>030000</td>
<td>040000</td>
<td>050000</td>
</tr>
<tr>
<td>8</td>
<td>040000</td>
<td>050000</td>
<td>060000</td>
<td>070000</td>
<td>080000</td>
<td>090000</td>
<td>000000</td>
<td>010000</td>
<td>020000</td>
<td>030000</td>
<td>040000</td>
<td>050000</td>
</tr>
<tr>
<td>9</td>
<td>040000</td>
<td>050000</td>
<td>060000</td>
<td>070000</td>
<td>080000</td>
<td>090000</td>
<td>000000</td>
<td>010000</td>
<td>020000</td>
<td>030000</td>
<td>040000</td>
<td>050000</td>
</tr>
</tbody>
</table>

Two Axis - Runs 6-10

TABLE 5.3 (cont.)

- 118 -
<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r_1$</td>
<td>$r_2$</td>
<td>$r_3$</td>
</tr>
<tr>
<td>1024</td>
<td>$C_{1024}$</td>
<td>$C_{1024}$</td>
</tr>
<tr>
<td>64</td>
<td>$C_{64}$</td>
<td>$C_{64}$</td>
</tr>
<tr>
<td>32</td>
<td>$C_{32}$</td>
<td>$C_{32}$</td>
</tr>
<tr>
<td>16</td>
<td>$C_{16}$</td>
<td>$C_{16}$</td>
</tr>
<tr>
<td>8</td>
<td>$C_{8}$</td>
<td>$C_{8}$</td>
</tr>
<tr>
<td>4</td>
<td>$C_{4}$</td>
<td>$C_{4}$</td>
</tr>
<tr>
<td>2</td>
<td>$C_{2}$</td>
<td>$C_{2}$</td>
</tr>
</tbody>
</table>

**TABLE 5.3 (cont.)**

Three Axis - Runs 1-4

- 119 -
<table>
<thead>
<tr>
<th>Runs</th>
<th>5</th>
<th>Runs</th>
<th>6</th>
<th>Runs</th>
<th>7</th>
<th>Runs</th>
<th>8</th>
<th>Runs</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>0</td>
<td>C</td>
<td>0</td>
<td>C</td>
<td>0</td>
<td>C</td>
<td>0</td>
<td>C</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Three Axis - Runs 5-8
<p>| | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
</tr>
<tr>
<td>1024</td>
<td>1024</td>
<td>1024</td>
<td>1024</td>
<td>1024</td>
<td>1024</td>
<td>1024</td>
<td>1024</td>
<td>1024</td>
<td>1024</td>
<td>1024</td>
</tr>
<tr>
<td>-1024</td>
<td>-1024</td>
<td>-1024</td>
<td>-1024</td>
<td>-1024</td>
<td>-1024</td>
<td>-1024</td>
<td>-1024</td>
<td>-1024</td>
<td>-1024</td>
<td>-1024</td>
</tr>
</tbody>
</table>

<p>| | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
</tr>
<tr>
<td>1024</td>
<td>1024</td>
<td>1024</td>
<td>1024</td>
<td>1024</td>
<td>1024</td>
<td>1024</td>
<td>1024</td>
<td>1024</td>
<td>1024</td>
<td>1024</td>
</tr>
<tr>
<td>-1024</td>
<td>-1024</td>
<td>-1024</td>
<td>-1024</td>
<td>-1024</td>
<td>-1024</td>
<td>-1024</td>
<td>-1024</td>
<td>-1024</td>
<td>-1024</td>
<td>-1024</td>
</tr>
</tbody>
</table>

Three Axis - Runs 9 and 10

TABLE 5.3 (cont.)
Main Program

| TABLE 5.4 |
| Program Listing |
Main Program (cont.)

TABLE 5.4 (cont.)
Main Program (cont.)

TABLE 5.4 (cont.)
Subroutine ITER (performs one iteration on two direction cosines)

TABLE 5.4 (cont.)
Subroutine ITER (cont.)

TABLE 5.4 (cont.)
Subroutine SCALE

(scales one direction cosine by operating on each of the three integer parts)

TABLE 5.4 (cont.)
Subroutine SCALE (cont.)

TABLE 5.4 (cont.)
APPENDIX A

SMOOTH PULSE SEQUENCES
(SRBC-RR-68-48)
SMOOTH PULSE SEQUENCES

A. J. Lincoln, M. Cohn and S. Even
Sperry Rand Research Center, Sudbury, Massachusetts

INTRODUCTION

The necessity sometimes arises in digital systems to generate external pulse rates whose frequency is related to the internal clock frequency by some proper rational fraction while still maintaining synchronism with the internal clock source. In many such cases, it is a requirement that pulses of the generated frequency be as uniformly spaced as possible within the constraint of synchronism. We define such pulse sequences as smooth pulse sequences in contrast with uniform sequences in which each pulse is separated from its predecessor by a fixed interval. The smooth sequences possess a number of interesting properties, both in their structure and in the ways they can be generated. Some of these properties will be treated in the following discussion.

Although it has not, to the authors' knowledge, been previously identified as such, a method has been in use for some time for generating smooth sequences. This is the constant multiplier used in digital differential analyzers (DDA's). Another technique often used for forming synchronous frequency multiples uses the "rate multiplier," and involves detecting the one-to-zero transitions of the stages of a binary counter chain. The rate multiplier technique, although capable of forming relatively smooth sequences, does not form a true smooth sequence, as can be easily demonstrated.
Other methods for generating smooth sequences can be described. Our interest here, however, is to provide a formal foundation for the subject and to prove several important properties of smooth pulse sequences.

PROPERTIES OF SMOOTH SEQUENCES

Choosing an arbitrary reference point in time, and assuming, for convenience, a unit-period clock, we define two types of pulse sequences: Uniform Sequences and Smooth Sequences. Each can be described by a doubly-infinite sequence of real numbers, the \(i^{th}\) number giving the arrival time, as counted by the clock, of the \(i^{th}\) pulse. Since our reference-time zero and the position of the index zero can be chosen arbitrarily and independently, we wish to consider equivalent those sequences which differ only by a shift in time or by reindexing. In other words, only the "pattern" of a sequence is important. Accordingly, we first make the following definition:

**Definition 1:**

Two arrival-time sequences \(\{a_i\}\) and \(\{b_i\}\) are equivalent if for some integers \(t\) and \(\tau\),

\[
a_i = b_{i+\tau} + t, \quad -\infty \leq i \leq \infty.
\]

A uniform pulse sequence is an ideally smooth sequence with rate \(p/q\) times that of the clock. It consists of \(p\) uniformly spaced pulses during every \(q\) clock periods.

**Definition 2:**

A Uniform Pulse Sequence of rate \(p/q\) has pulses at times \(\{u_i\}\), where \(u_i = \frac{i}{p}\), \(-\infty \leq i \leq \infty\).
Unless \( q/p \) is an integer, the pulses of the uniform sequence do not all coincide in time with clock pulses, so the problem of synchronous pulse rate generation cannot generally be solved by a uniform sequence. The concept, though, is crucial in defining and characterizing smooth sequences, which, heuristically, are the best synchronous approximation to uniform sequences.

**Definition 3:**

A smooth pulse sequence of rate \( p/q \) has pulses only at integral clock times \( \{s_i\}, -\infty \leq i \leq \infty \); moreover, these pulses can be put into one-to-one correspondence with the pulses of the uniform sequence in such a way that

\[
\max(s_i - u_i) - \min(s_i - u_i) < 1.
\]

To construct a concrete example of a smooth pulse sequence for the fraction \( 5/8 \), we place pulses at those clock times which coincide with or immediately follow pulses of the uniform sequence.

**Example 1**

<table>
<thead>
<tr>
<th>Time</th>
<th>-2</th>
<th>-1</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clock</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Smooth</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Uniform</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[0/5 \quad 8/5 \quad 0/5 \quad 8/5 \quad 8/5 \quad 8/5 \quad 8/5\]

Here the arrival times of the smooth sequence are \( \ldots, -1, 0, 2, 4, 5, 7, 8, \ldots \). If the pulse pattern between, say, time 0 through time 7 is...
repeated indefinitely, an infinite, smooth sequence of rate $\frac{5}{8}$ results.

Since this sequence was constructed to be periodic, it suffices to observe the smoothness of a single period. Under the natural correspondence (shown by arrows), the deviations between the smooth and the uniform arrival times range between 0 and $\frac{4}{5}$, satisfying Definition 3.

The last condition in Definition 3 bounds the range of time deviations between pulses of a smooth sequence and their correspondents in a uniform sequence. We next observe that the bound can be sharpened and that within the equivalence relation the deviations themselves can be bounded.

Since the $s_i$ are integers, and the $u_i$ are integral multiples of $q/p$, the differences $(s_i - u_i)$ between corresponding pulse times must be integral multiples of $1/p$, hence

Lemma 1: A smooth sequence satisfies the inequality

$$\max_{i} (s_i - u_i) - \min_{i} (s_i - u_i) \leq \frac{p-1}{p} .$$

Now suppose that $i_0$ is an index where the difference $(s_i - u_i)$ achieves its minimum; that is,

$$\min_{i} (s_i - u_i) = (s_{i_0} - u_{i_0}) .$$

By an integral shift in time and a reindexing, we form the new sequence $\{s'_i\}$, where

$$s'_i = s_i + i_0 - s_{i_0} .$$

To be rigorous here we should note that Lemma 1 and its preceding discussion imply that the differences $(s_i - u_i)$ take on at most $p$ distinct values, and that these values are bounded below by $s_j - u_j - (p-1)/p$, where $j$ is an index chosen arbitrarily. Therefore, the sequence has a greatest lower bound, which is $s_{i_0} - u_{i_0}$ for some $i_0$.
The minimum deviation of \( \{s'_i\} \) from the uniform sequence is given by

\[
\begin{align*}
\min_i (s'_i - u_i) &= \min_i (s_{i+i_0} - s_{i_0} - u_i) \\
&= \min_i (s_{i+i_0} - u_{i+i_0} - s_{i_0} + u_{i_0}) \\
&= \min_i (s_{i+i_0} - u_{i+i_0}) - (s_{i_0} - u_{i_0}).
\end{align*}
\]

But by definition of \( i_0 \) this quantity vanishes, so that all deviations of \( \{s'_i\} \) from the uniform must be nonnegative. Invoking Lemma 1, we can now state:

**Lemma 2:** Within equivalence, every smooth sequence satisfies

\[
0 \leq (s'_i - u_i) \leq \frac{\beta-1}{p}, \quad -\infty \leq i \leq \infty.
\]

The sequence in Example 1 was designed to satisfy this bound without an equivalence transformation.

We are now able to prove the fundamental theorem of smooth sequences.

**Theorem 1:** A smooth sequence of rate \( p/q \) is unique to within equivalence.

**Proof:** Given any two smooth sequences of rate \( p/q \), consider their equivalent forms complying with Lemma 2. The \( i^{th} \) pulses of both smooth sequences must coincide with clock pulses, and must coincide with, or lag by less than a clock period, the same correspondent, \( u_{i_1} \), in the uniform sequence of rate \( p/q \). Therefore, the \( i^{th} \) pulses in the two smooth sequences must coincide with the same clock pulse, hence with each other. QED

This is an extremely strong result, following, as it does, only from the definitions of uniform and smooth sequences and the notion of equivalence.
So far we have used arrival times to describe pulse sequences. It is convenient now to introduce a dual description, the cumulative count. If $A(t)$ is the number of pulses in the sequence $\{a_i\}$ which have arrived by time $t$, we define the "count sequence" $\{\alpha(t)\}$, where

$$\alpha(t) = A(t) - A(a_0).$$

Thus $\alpha(t)$ is an integer-valued function of time, satisfying

**Lemma 3:** If $\{a_i\}$ and $\{\alpha(j)\}$ are, respectively, the arrival time sequence and the count sequence for a synchronous pulse sequence,

$$\alpha(j) = 1 \text{ for } a_i \leq j < a_{i+1},$$

$$a_\alpha(j) \leq j < a_{\gamma(j)+1}.$$

As done in Lemma 3, we will always use the lowercase Greek equivalent of the arrival-time character for the corresponding count sequence. Also, by analogy with the notation $\{u_i\}$ for the arrival times of pulses in the uniform sequence, we make the following definition.

**Definition 4.** A uniform sequence of rate $p/q$ has a (rational) count sequence $\{u(j)\}$, where

$$u(j) = j \frac{p}{q}, \quad -\infty \leq j \leq \infty.$$

We are now ready to prove the important relationship between the arrival time sequence and the count sequence.
Theorem 2 (Duality Theorem):

If \( \{a_i\} \) and \( \{\alpha(j)\} \) describe a synchronous pulse sequence of rate \( \frac{p}{q} \),

\[
\max\left(\frac{a_i - u_i}{q} \right) - \min\left(\frac{a_i - u_i}{q} \right) \leq \frac{p-1}{p}
\]

if and only if

\[
\max\left(\frac{u(j) - \alpha(j)}{q} \right) - \min\left(\frac{u(j) - \alpha(j)}{q} \right) \leq \frac{q-1}{q}
\]

Proof: To prove the "if" part, observe that for any \( a_i \),

\[
\frac{q}{p} \min\left(\frac{u(j) - \alpha(j)}{q} \right) = \frac{q}{p} \min\left(\frac{p}{q} j - \alpha(j) \right) \leq \frac{q}{p} \left( \frac{p}{q} a_i - \alpha(a_i) \right) = \left( a_i - u_i \right)
\]

Also, since

\[
\frac{q}{p} \max\left(\frac{u(j) - \alpha(j)}{q} \right) \geq \frac{q}{p} \left( \frac{p}{q} (a_i - 1) - \alpha(a_i - 1) \right) = \left( a_i - 1 - \frac{q}{p} (i-1) \right)
\]

\[
\left( 1 - \frac{q}{p} \right) + \frac{q}{p} \max\left(\frac{u(j) - \alpha(j)}{q} \right) \geq \left( a_i - u_i \right)
\]

From these two results,

\[
\left( 1 - \frac{q}{p} \right) + \frac{q}{p} \max\left(\frac{u(j) - \alpha(j)}{q} \right) \geq \max\left(\frac{a_i - u_i}{q} \right)
\]

\[
\frac{q}{p} \min\left(\frac{u(j) - \alpha(j)}{q} \right) \leq \min\left(\frac{a_i - u_i}{q} \right)
\]

By subtraction and hypothesis,

\[
\left( 1 - \frac{q}{p} \right) + \frac{q}{p} \left( \frac{q-1}{q} \right) = \frac{p-1}{p} \geq \max\left(\frac{a_i - u_i}{q} \right) - \min\left(\frac{a_i - u_i}{q} \right)
\]

\[\blacksquare\]
To prove the "only if" part, observe that for any clock time \( j \), there is a pair of successive arrival times such that \( a_1 \leq j \leq a_{i+1}-1 \), so that \( \sigma(j) = i \). Therefore,

\[
\frac{p}{q} a_i - i \leq \frac{p}{q} j - \alpha(j) \leq \frac{p}{q} (a_{i+1}-1) - 1 ,
\]

\[
\frac{p}{q} (a_i - u_i) \leq u(j) - \alpha(j) \leq \frac{p}{q} (a_{i+1} - u_{i+1}) - \frac{p}{q} + 1 .
\]

But then

\[
\left(1 - \frac{p}{q}\right) + \frac{p}{q} \max(a_{i+1} - u_{i+1}) \geq \max(u(j) - \alpha(j)) ,
\]

\[
\frac{p}{q} \min(a_i - u_i) \leq \min(u(j) - \alpha(j)) ;
\]

again by subtraction and hypothesis,

\[
\left(1 - \frac{p}{q}\right) + \frac{p}{q} \left(\frac{p-1}{p}\right) = \frac{q-1}{q} \geq \max(u(j) - \alpha(j)) - \min(u(j) - \alpha(j)) .
\]

QED

An obvious consequence of Theorem 2 and Definition 3 is

**Corollary 2.1:** A synchronous pulse sequence described by the count sequence \( \{a(j)\} \) is smooth if and only if

\[
\max(u(j) - \alpha(j)) - \min(u(j) - \alpha(j)) \leq \frac{q-1}{q} .
\]

In the discussion which follows, several of the results have analogous statements and analogous proofs in terms of arrival times as well as count sequences. In such cases, both statements, but only one proof, will be given.
**Lemma 3:** If, for all $i$, $a_i+c=a_i+d$, then for any integer $k$, $a_{i+k}c=a_i+kd$.

**Lemma 3':** If, for all $j$, $\alpha(j+d)=\alpha(j)+c$, then for any integer $k$, $\alpha(j+kd)=\alpha(j)+kc$.

**Proofs:** Both proofs are simple inductions on $k$.

**Lemma 4:** If, for all $i$, $a_i+c_1=a_i+d_1$ and $a_i+c_2=a_i+d_2$, then $c_1/c_2=d_1/d_2$.

**Lemma 4':** If, for all $j$, $\alpha(j+d_1)=\alpha(j)+c_1$ and $\alpha(j+d_2)=\alpha(j)+c_2$, then $c_1/c_2=d_1/d_2$.

**Proofs:** Consider $\alpha(j+d_1d_2)$; applying Lemma 3' twice,

$$\alpha(j)+q_1c_2=\alpha(j)+d_2-c_1$$

for all $j$, so that

$$c_1/c_2=d_1/d_2$$.

**Lemma 5:** The arrival time sequence $\{s_i\}$ for a smooth sequence of rate $\frac{p}{q}=\frac{rp}{rq}$, $p$ and $q$ relatively prime, satisfies $s_{i+p}=s_i+q$ for all $i$.

**Lemma 5':** The count sequence $\{\sigma(j)\}$ for a smooth sequence of rate $\frac{p'}{q'}=\frac{rp}{rq}$, $p$ and $q$ relatively prime, satisfies $\sigma(j+q)=\sigma(j)+p$ for all $j$. 

$\mathcal{F}^9$
Proofs:  \[ \sigma(j+q) - \sigma(j) = \sigma(j+q) - \sigma(j+q-1) - \sigma(j+q-1) + \sigma(j+q-2) \]
\[ = \{\sigma(j+q) - \sigma(j+q-1) - \sigma(j) + \sigma(j+q-2)\} + p. \]

The difference on the left-hand side of this equation is clearly an integer, while the bracketed quantity on the right is less than unity (by Cor. 2.1). Therefore, the bracketed quantity must be zero, proving the lemma.

We now define two new sequences, the first differences of the arrival times and the count sequence. The first, called the "gap" sequence, consists of the clock-time intervals between pulses. The second, called the "binary" sequence, is unity at those clock times when pulses are present, and is zero at all others. Thus, it depicts the pattern of the pulse sequence.

Definition 6: The gap sequence is given by
\[ g_i = a_i - a_{i-1}, \quad -\infty \leq i \leq \infty. \]

Definition 7: The binary sequence is given by
\[ b(j) = \sigma(j) - \sigma(j-1), \quad -\infty \leq j \leq \infty. \]

Theorem 3: A smooth pulse sequence is periodic. If its rate is \[ \frac{p}{q} = \frac{p}{r/q}, \] \( p \) and \( q \) relatively prime, then the period is \( q \) clock times during which \( p \) pulses appear.

Proof: \[ b(j+q) - b(j) = \sigma(j+q) - \sigma(j+q-1) - \sigma(j+q-1) + \sigma(j+q-2) \]
\[ = \sigma(j+q) - \sigma(j+q-1) - \sigma(j+q-1) + \sigma(j+q-2) \]
\[ = p - p = 0. \]

Lemma 5' makes explicit the number of pulses in each period, and Lemma 4' proves that if there were a shorter period than \( q \), then \( p \) and \( q \) could not be relatively prime.

QED
These last results simplify the investigation of smooth sequences in two ways. First, only rates in reduced form, that is, with \( p \) and \( q \) relatively prime, need be considered. Second, a single period suffices to describe any smooth sequence. This permits the use of the following convenient notations for a smooth sequence of rate \( p/q \):

(i) Arrival times: \((s_0, s_1, \ldots, s_{p-1})\);

(ii) Gaps
\[
\left\{ g_0, g_1, \ldots, g_{p-1}, \right. \quad \text{where } g_i = s_i - s_{i-1} \pmod{p}, \\
\left. \quad \text{and } \sum_{i=0}^{p-1} g_i = q \quad \text{indices modulo } p. \right\}
\]

(iii) Counts \((\sigma(0), \sigma(1), \ldots, \sigma(q-1))\);

(iv) Binary \((\beta(0), \beta(1), \ldots, \beta(q-1))\),
where \( \beta(j) = \sigma(j) - \sigma(j-1) \pmod{q} \)
from \( j=0 \) to \( j=q-1 \).

Finally, we consider the difference-sequences between smooth and uniform sequences.

Lemma 6. The sequence \((s_i - u_i)\) has period \( q \).

Lemma 6'. The sequence \((u(j) - \sigma(j))\) has period \( q \).

Proofs: Since \( u_i = \frac{q}{p} i, u_{i+p} = u_i + q \) for all \( i \). By Lemma 5, \( \{s_i\} \) satisfies the same recursion, so the difference \((s_i - u_i)\) has a period of \( q \) clock times (or \( p \) pulses). But since \( p \) and \( q \) are relatively prime, \( u_i \) and \((s_i - u_i)\) are integral only when \( i \) is a multiple of \( p \). Therefore \((s_i - u_i)\) has no period less than \( q \).
With this knowledge and the observation that all \((s_i - u_i)\) are proper fractions with denominator \(p\), and all \((u(j) - \sigma(j))\) are proper fractions with denominator \(q\), we can complete the refinement, begun in Lemmas 1 and 2, of the characterization of smooth sequences.

**Theorem 4:** A smooth sequence of rate \(\frac{p}{q}\) satisfies
\[
\max_{i} (s_i - u_i) - \min_{i} (s_i - u_i) = \frac{p-1}{p}.
\]
\[
\max_{j} (u(j) - \sigma(j)) - \min_{j} (u(j) - \sigma(j)) = \frac{q-1}{q}.
\]

**Proof:** Since the difference \((s_i - u_i)\) periodically assumes \(p\) different values, all of which are proper fractions with denominator \(p\),
\[
\max_{i} (s_i - u_i) - \min_{i} (s_i - u_i) \geq \frac{p-1}{p}.
\]

This inequality combined with Definition 3 proves the theorem for the arrival times. A similar argument can be made for the count sequence.

**QED**

**Summary**

Starting from a definition of smooth sequences as synchronous pulse trains with bounded range of deviation from an ideal uniform rate, we have seen that the pattern of such a sequence is unique and periodic, and that the deviations from the ideal are quantized and less than unity. Dual descriptions have been given in terms of arrival times and cumulative counts, as well as in terms of their first differences. The deviations of smooth sequences from their uniform counterparts have been characterized.

In closing we note that from the uniqueness of these sequences, which holds even under our rather broad definition of smoothness, it follows that smooth sequences minimize all reasonable measures of deviation from uniform spacing.