APPLICATION OF MODEL FOLLOWING CONTROL AND ESTIMATION TECHNIQUES TO ATTITUDE CONTROL OF MANOEUVRING SPACECRAFT

PART II: SIMULATION SOFTWARE DESCRIPTION

BY

A.P. TERPSTRA, R.F. VAN DEN DAM, P.Th. L.M. VAN WOERKOM AND T. ZWARTBOL
Reprints from this publication may be made on condition that full credit be given to the Nationaal Lucht- en Ruimtevaartlaboratorium (National Aerospace Laboratory NLR) and the Author(s)
This document describes two computer programs developed during a study of the application of modern control techniques in the area of spacecraft attitude control. Theory and simulation results are described in part I of this report. In part II a general overview of the tasks of the simulation program is presented, and program design aspects are discussed. Each module is then described separately and flow charts are given. Finally, implementation aspects specific to the computer used, as well as some execution characteristics are briefly discussed.
APPLICATION OF MODEL FOLLOWING CONTROL AND ESTIMATION TECHNIQUES TO ATTITUDE CONTROL OF MANOEUVRING SPACECRAFT
PART II: SIMULATION SOFTWARE DESCRIPTION
by
A.P. Terpstra, R.F. van den Dam, P.Th.L.M. van Woerkom
and T. Zwartbol

SUMMARY

This document describes two computer programs developed during a study of the application of modern control techniques in the area of spacecraft attitude control. Theory and simulation results are described in part I of this report. In part II a general overview of the tasks of the simulation program is presented, and program design aspects are discussed. Each module is then described separately and flow charts are given. Finally, implementation aspects specific to the computer used, as well as some execution characteristics are briefly discussed.

This investigation has been prepared under contract with the Netherlands Agency for Aerospace Programs (NIVR) Contract Nos. 1771 and 1858
This document presents an approach to sampled data, on-board estimation and control of the attitude motion of manoeuvring spacecraft. Algorithms which are developed include:

- a- model following control of attitude manoeuvres
- b- estimation of spacecraft state (attitude, angular velocity, disturbance torque)
- c- optical-inertial attitude determination, with estimation of gyro parameters (drift rate bias, scale factor error).

The algorithms were validated in single-axis software simulations of an attitude control system of the type as used in the Infra Red Astronomical Satellite (IRAS). The considered control system comprises a strapdown rate-integrating gyro and a slit-type star sensor for optical-inertial attitude sensing, a reaction wheel actuator, and a digital on-board computer. The truth models for simulation of the hardware components are discussed. The estimation and control algorithms are described, design trade-offs are discussed, and simulation results are presented.
### CONTENTS (continued)

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>PLOTTING PROGRAM DESCRIPTION</td>
<td></td>
</tr>
<tr>
<td>3.1</td>
<td>General overview</td>
<td>18</td>
</tr>
<tr>
<td>3.2</td>
<td>Module description</td>
<td></td>
</tr>
<tr>
<td>3.2.1</td>
<td>Program PLOT</td>
<td>19</td>
</tr>
<tr>
<td>3.2.2</td>
<td>Subroutine PLINIT</td>
<td>20</td>
</tr>
<tr>
<td>3.2.3</td>
<td>Subroutine FILL</td>
<td>21</td>
</tr>
<tr>
<td>3.3</td>
<td>Supporting software</td>
<td>22</td>
</tr>
<tr>
<td>4</td>
<td>IMPLEMENTATION CONSIDERATIONS</td>
<td></td>
</tr>
<tr>
<td>4.1</td>
<td>Program code</td>
<td>23</td>
</tr>
<tr>
<td>4.2</td>
<td>Execution statistics</td>
<td>24</td>
</tr>
<tr>
<td>4.3</td>
<td>Software maintenance</td>
<td>25</td>
</tr>
<tr>
<td>5</td>
<td>REFERENCES</td>
<td></td>
</tr>
</tbody>
</table>

- 2 tables
- 15 figures

(41 pages in total)
LIST OF SYMBOLS

A  mean

$B^2$  variance

N  number of stars in reference table

n  number of samples

RE  gyro resolution element

RUNTIM  total simulation run time

$S_n$  sum of n samples

t  actual simulation time

$\Delta T$  sampling period

$\Delta \omega_{gc}(k)$  corrected gyro output at time instant k

$\hat{\epsilon}_1$  estimated scale factor error at time instant 1

$\mu$  mean

$\sigma^2$  variance

$\hat{\omega}_{b1}$  estimated drift rate bias at time instant 1

The notation $N(\mu, \sigma^2)$ indicates a normal (i.e. Gaussian) stochastic distribution with mean $\mu$ and variance $\sigma^2$.

ACRONYMS

ANSI  American National Standards Institute

CPU  Central Processing Unit

IRAS  Infra Red Astronomical Satellite

NIVR  Netherlands Agency for Aerospace Programs

NLR  National Aerospace Laboratory

OBC  On Board Computer

SI  Système Internationale

SOP  Spacecraft Operations Plan
INTRODUCTION

The software described in this document has been developed by the National Aerospace Laboratory NLR in order to assist in various phases of a research program sponsored by the Netherlands Agency for Aerospace Programs NIVR. This program was initiated to investigate the application of modern estimation and control techniques in the area of spacecraft attitude control. For an overview of the program, see Part I of this report (Ref. 1, chapter 1).

The following software, developed during this research project, will be described in the sequel:
- a batch simulation program;
- an interactive plotting program.

1.1 General overview

After an initial theoretical exploration phase, a generic satellite of the IRAS type was chosen for further development of the algorithms. The relevant configuration is depicted in figure 1.1.

Only single-axis rotational motion is considered. For attitude control the spacecraft, consisting of a rigid body, is equipped with:
- a reaction wheel as torque actuator;
- an optical/inertial measurement system consisting of a slit-type star sensor and a rate integrating gyroscope;
- a digital onboard computer for autonomous operation.

Both measurement processing and actuator commanding are performed at a fixed, OBC determined, frequency. Details of the different hardware components are given in reference 1, chapter 5.

1.2 Software set-up

As observed in the introduction, the software considered consists of two programs: a batch simulation program and an interactive plotting program. Apart from simulation of the spacecraft attitude hardware named in section 1.1, the simulation program has a number of other tasks, which will be described in section 2.1. The plotting program can be used after a simulation run has been completed. It permits the user to graphically display selected data from a simulation run.
In this context it seems appropriate to describe the various stages in a typical simulation (see Fig. 1.2). As the simulation program is batch oriented, each specific simulation should be carefully prepared. Prior to execution of the program all input data must have been collected; together with the appropriate operating system commands it must then be submitted as a complete job deck (steps I and II in Fig. 1.2).

Step III in figure 1.2 comprises the actual simulation, executed in batch mode. Output is generated by the program according to a set of processing options, which are part of the input data. The simulation results are obtained in two ways:

- line printer output: consists of tabular, paginated dumps of simulation quantities, which have been functionally grouped together (step IV.A in Fig. 1.2);
- graphical output: on request the simulation program creates a binary disk file, containing the values of various basic simulation quantities at each sampling instant.

The user can subsequently execute the plotting program on an interactive graphical display terminal to generate plots (step IV.B in Fig. 1.2). Suitable graphs can be reproduced using a hardcopy unit.

1.3 Organization of the report

The remainder of the report is concerned with the following items:

- description of the simulation program (chapter 2);
- description of the plotting program (chapter 3);
- some remarks on the implementation of both programs at the NLR computer and terminal network (chapter 4).

The specific modules of each program are globally described in the relevant chapters; full details are contained in the program listings.

2 SIMULATION PROGRAM DESCRIPTION

This chapter is concerned with the batch simulation program. It contains the following items:

- tasks of the simulation program (section 2.1);
- a description of the basic simulation modules (section 2.2);
- a description of supporting software (section 2.3).
2.1 Program tasks

The simulation program has been designed to serve the following purposes (compare Fig. 1.1):

- simulation of spacecraft attitude hardware (block "SPACECRAFT");
- implementation of digital estimation and control algorithms (block "OBC"); together with the previous block we thus obtain a closed control loop;
- external interfaces, i.e. communication with the outside world:
  i. input of various simulation parameters and initial values;
  ii. simulation of input and processing of telemetry SOP data for commanding the spacecraft rotation around a single axis from a ground station (block "SOP");
  iii. output of simulation parameters and initial values;
  iv. monitoring the simulation by dumping various simulation quantities at specified time intervals.

These tasks will be described in more detail in sections 2.1.1 - 2.1.3.

2.1.1 Spacecraft hardware truth modelling

An important task of the simulation program is the digital simulation of the spacecraft attitude hardware. To this end, a truth model of each component has been developed (Ref. 1). Each truth model has been implemented as a separate module.

A related task is the simulation of the spacecraft environment. Here too, suitable models have been developed and implemented.

All software models mentioned will be discussed in section 2.2.3 (subroutine SATACS).

2.1.2 Digital estimation and control algorithms

The second major task of the simulation program is the execution of the onboard computer programs relevant for attitude control of the spacecraft. Specifically, the following modules are concerned:

- GYRCOR (section 2.2.4) - contains the gyro output correction;
- GYREST (section 2.2.5) - contains the gyro parameter estimator;
- STATE (section 2.2.6) - contains the spacecraft state estimator;
- CONTRL (section 2.2.9) - contains the control law.

The algorithms used in these modules are described in reference 1.
It must be noted that the actual OBC characteristics have not been simulated. As a result, the algorithms are executed in the arithmetic precision of the simulation computer. This inevitably leads to a "slightly different" performance than would have been obtained with the actual OBC.

Further discussion of the estimation and control modules can be found in the relevant module descriptions in section 2.2.

2.1.3 External interfaces

The last major task of the simulation program consists of the communication of the running program with the outside world. This is motivated by the following considerations:

- this kind of simulation program needs a facility to perform simulations with different parameters and initial values, which are supplied by the user;
- a simulation program does not provide much information if it lacks a facility for monitoring various quantities of interest;
- in this specific case, the spacecraft motion is commanded externally; it would not make sense to store these commands as fixed quantities in the program.

The first two arguments can be considered to apply to more general simulation software, whereas the last one is more or less specific to the present simulation program.

The various input/output tasks, described in section 2.1, are implemented in several program modules. The relevant modules, which will be discussed in the sequel, are:

- input of simulation parameters and initial values: INPUT (section 2.3.3.1);
- input and processing of telemetry data: SOP (section 2.2.7), COMMAND (section 2.2.8);
- output of simulation parameters and initial values: INPOUT (section 2.3.3.2);
- output of telemetry data and simulation variables: SOP (section 2.2.7), OUTPUT (section 2.3.3.3).
2.2 Module descriptions

Sections 2.2.1 - 2.2.9 contain descriptions of those modules which specifically concern the simulation of the spacecraft attitude control system. All other modules, including those performing the various input/output tasks of the program, are described in section 2.3.

2.2.1 Program MAIN

This module is the driver of the simulation program and reflects its primary organization. It is based on the spacecraft configuration scheme given in figure 1.1.

Since the simulation is digital, the loop in figure 1.1 (which is in fact a continuous-time or analog loop) has to be cut somewhere. It was decided to do this immediately before a commanded reaction wheel torque is processed by the spacecraft hardware truth models, i.e. after block "OBC"; a dummy initial torque command then has to be specified.

The program is completed with an initialization routine (INIT) and a variable monitoring routine (OUTPUT). The flowchart of the resulting program can be found in figure 2.1.

2.2.2 Subroutine INIT

This module initializes all variables and parameters, needed by the simulation program, which are not initialized internally in the program. The majority of these are read from the user-specified input file; only those which are functions of other parameters and initial values are computed in this module. Some consequences of this approach are:

- the user needs only specify the independent initial values and parameters, which decreases the amount of input data;
- on the other hand, since there are no default values for whatever independent initial value or parameter, all of these must explicitly be assigned, thereby increasing the amount of input data for a given series of simulations. In practice, however, this will be a minor drawback since it is highly probable that the user will create a default job deck (see section 1.2) for each new series of simulations and apply selected modifications to it.

An advantage is that with each simulation a user will know the values of all independent initial values and parameters, so that he
(presumably) has a reasonable idea of the results to be expected. A flowchart of this module is given in figure 2.2.

2.2.3 Subroutine SATACS

This module contains all truth models of the spacecraft attitude hardware and its environment. As noted in section 2.1.1, the following hardware is modelled:

- the rigid spacecraft body;
- a reaction wheel;
- a gyroscope;
- a star sensor.

The spacecraft environment modelling comprises:

- external disturbance torques;
- a star position reference table.

External disturbance torques have primarily been used in test and optimization phases, using a very simple truth model.

All other truth models are described in reference 1. The specific models can be located in the following modules:

- rigid spacecraft body and external disturbance torques in subroutine SATACS (this section);
- reaction wheel in subroutine RWL (section 2.2.3.1);
- gyroscope in subroutine GYRO (section 2.2.3.4);
- star sensor and star position reference table in the subroutines SECTOR and SSE (sections 2.2.3.2 and 2.2.3.3).

A flowchart of module SATACS is given in figure 2.3.

2.2.3.1 Subroutine RWL

In this module the reaction wheel truth model described in reference 1, section 5.4, is implemented. Only some additional information will be given here.

Firstly, the nonlinearity coefficient is modelled as a function which is equal to zero when present and previous torque setpoint have the same sign, and is equal to a constant when these are of opposite sign.

Secondly, the reaction wheel angular velocity is also computed in this module; it is obtained by simple integration of the actual reaction torque.
Thirdly, the stiction model used is in fact somewhat more complicated; with small torques and wheel velocities near zero some special effects occur, which have also been modelled.

A flowchart of this module is given in figure 2.4.

2.2.3.2 Subroutine SECTOR

This module is used by the main module SATACS (section 2.2.3) to determine whether a reference star from the position table has been in the line of sight of the star sensor during the previous sampling interval (Ref. 1, section 5.6). The method used will be described below.

Assuming that the number of stars in the table is N, a full 2 rotation of the spacecraft is divided into N "sectors" as shown in figure 2.5. A star measurement is now recognized a posteriori by establishing a difference in sector number before and after the current sampling interval. In this case subroutine SSE (section 2.2.3.3) is called to generate a measurement time and an absolute attitude measurement. This method fails only in a few pathological situations.

2.2.3.3 Subroutine SSE

This module implements the star measurement model described in reference 1, section 5.6. It is assumed that the star measurement time, calculated in this module, is exactly known in the gyro parameter estimator module GYREST (section 2.2.5).

2.2.3.4 Subroutine GYRO

This module implements the gyroscope truth model described in reference 1. The flowchart of this module can be found in figure 2.6.

2.2.4 Subroutine GYRCOR

This module computes the "corrected gyro output" (see Ref. 1, section 4.2) from the number of pulses counted by the gyro electronics. The algorithm used is quite simple; it is complementary to the truth model and is described in figure 2.7.
2.2.5 Subroutine GYREST

This module contains the gyro parameter estimator, which uses the absolute attitude measurement provided by the star sensor upon detection of a reference star crossing. It recursively computes minimum variance estimates of the following quantities:

1. accumulated gyro reference attitude error;
2. gyro drift rate bias;
3. gyro scale factor error.

During scanning motions of the spacecraft the scale factor error and drift rate bias are not observable separately; as described in reference 2 the effect of the scale factor error is then estimated in combination with that of the drift rate bias.

The rather complex computations involved in the specific implementation of this time-varying Kalman filter have been fully described in reference 2. The global organization of the module is given in figure 2.8.

2.2.6 Subroutine STATE

This module contains the spacecraft state filter, which uses the integrated "corrected gyro output" (subroutine GYRCOR, section 2.2.4) to recursively compute minimum variance estimates of the spacecraft attitude, velocity and disturbance torque. The algorithms used have been described in reference 1; the global organization of this module is given in figure 2.9.

2.2.7 Subroutine SOP

This module contains the telemetry SOP data interface. It is called whenever the end of a SOP block is reached. It reads four values from the user-specified input file:

1. a flag indicating whether the scale factor error is to be estimated separately in module GYREST (see section 2.2.5) during this SOP-block;
2. the initial commanded spacecraft attitude;
3. the commanded spacecraft velocity, constant over the entire time interval of the SOP-block;
4. the end time of the SOP-block.

After reading these values and performing some simple checks, such as making sure that the initial commanded attitude lies in the range
[0,2\pi), information about this SOP-block is listed in the line printer output stream.

2.2.8 Subroutine CMAND

This module implements the "Setpoint Generation" program described in reference 1, section 3.2.

2.2.9 Subroutine CONTRL

This module implements the digital bi-modal model-following control law described in reference 1, chapter 3. It uses the available estimates of spacecraft attitude, velocity and disturbance torque (provided by the spacecraft state estimator in module STATE, section 2.2.6) to compute a reaction wheel setpoint (= torque level) for the next sampling interval. The global organization of this module is given in figure 2.10.

2.3. Supporting software

Sections 2.3.1 - 2.3.3 describe the supporting software that forms part of the program; the modules have been divided into two categories:

- input/output modules (section 2.3.1);
- "computational" modules (section 2.3.2), which perform some simple calculations needed at various places in the program.

The relevant module descriptions are contained in section 2.3.3.

2.3.1 Input/output modules

In section 2.1.3 the need for input/output capabilities in the program was indicated. For the sake of completeness the types of input/output to be performed are repeated here:

- telemetry SOP data input/output:
  already discussed in section 2.2.7 (subroutine SOP);
- simulation parameters and initial values input/output:
  to be described in sections 2.3.3.1 and 2.3.3.2 (subroutines INPUT and INPOUT);
- monitoring of simulation variables: to be described in section 2.3.3.3 (subroutine OUTPUT).
For the calling sequence of these modules the reader is referred to figures 2.1 and 2.2.

Finally, there are a few auxiliary modules, which are used only to support program output. These are described in sections 2.3.3.4 and 2.3.3.5 (subroutines DATIME and HEADER).

2.3.2 Computational modules

The following computational modules are used by the simulation program:

- a routine to provide samples of discrete-time Gaussian white noise (function NORMAL, section 2.3.3.6);
- two routines to scale computed attitudes and attitude differences into fixed ranges throughout the program (functions SCLATT and SCLDIF, section 2.3.3.7).

2.3.3 Module descriptions

This section contains descriptions of all simulation program modules not described in section 2.2, except for the specific line printer output modules FILOUT, CROUT, STROUT, RWLOUT and GYROUT (see section 2.3.3.3).

2.3.3.1 Subroutine INPUT

This module reads all independent initial values and parameters from the user-specified input file (see also subroutine INIT, section 2.2.2). In addition to these numerical values the following data is read in this module:

- a run identification character string, which is used in all program output;
- the values of the processing options, controlling the execution of the simulation program.

These last items are read in a fixed format. All other (numerical) values are read in free format; since this input file constitutes the primary means of communication between the human operator and the program it is essential to use an input format which is easy to use. Therefore the use of fixed formats for input of numerical data has been avoided.
2.3.3.2 Subroutine INPOUT

Output of all simulation initial values and parameters is performed on request in this module. The values specified in the input file, as well as those derived from them in the module INIT (section 2.2.2), are reproduced. In contrast with the data input described in the preceding section, output of numerical values can be performed quite satisfactorily in fixed formats.

If the user has requested the creation of a binary plot data file (see section 1.2) simulation time interval and sampling period, along with run identification, date and time, are written to this file for use by the plotting program.

2.3.3.3 Subroutine OUTPUT

All results of the simulation are output on request in this module. As explained in section 1.2, two types of output can be generated:

- line printer output;
- a binary plot data disk file.

The latter is generated using FORTRAN 77 unformatted write statements. To reduce disk storage requirements it was decided to write only the minimum number of variables to the binary data file; this means that in order to generate all quantities of interest the plotting program must also perform some computations.

For instance, both actual and commanded spacecraft attitude are written to the plot data file; it is then no longer necessary to also dump the attitude error (= actual attitude - commanded attitude). Since the number of quantities that can be plotted (45) considerably exceeds the number of variables dumped (30) a large saving of disk space is achieved, at the expense of a slight increase in plotting program size and execution time.

With respect to the output reproduced on a line printer, the quantities of interest have been divided into a few functionally connected groups. These groups are:

- filter results (subroutine FILOUT):
  both the spacecraft state and the gyro parameter estimation results;
- control results (subroutine CTROUT):
  the results of the different parts of the digital control law;
- star measurement results (subroutine STROUT):
  these include measurement time and error, but also the gains and error
  standard deviations computed by the gyro parameter estimator;
- reaction wheel results (subroutine RWLOUT):
  requested torque and actual reaction wheel data;
- gyro results (subroutine GYROUT):
  actual attitude increment and gyro measurement data.

  All of these output groups can be selected or deselected separately
  with the user-specified processing options.

  The units used for displaying values, both here and in the plotting
  program, were selected to convey as much information as possible to the
  user; it was therefore decided to use degrees for attitude, arcmin/s
  for velocity, arcsec for attitude errors and so on. Consistent units were
  used throughout the programs.

2.3.3.4 Subroutine DATIME

  This auxiliary module saves date and time at the beginning of the simula-
  tion run for output purposes. They are used both in the line printer out-
  put page headers and in the graphs produced by the plotting program. In
  this way, corresponding outputs can be easily matched.

2.3.3.5 Subroutine HEADER

  This auxiliary module is used to print a page header on each page of
  line printer output produced by the simulation program. The actual
  paginating is to be performed by the calling module.

2.3.3.6 Function NORMAL

  To simulate a stochastic environment random number generators are
  needed. In this simulation program discrete-time Gaussian white noise is
  used; since the available FORTRAN library only provides random numbers,
  uniformly distributed between 0 and 1, a special routine has been written
  to generate a normal distribution.

  The algorithm used is derived through the following stochastic
  argument. Let n samples be taken from a certain stochastic distribution
  with mean A and variance B^2. The central limit theorem now states that
  the sum S_n of these n samples, for n sufficiently large, approximates a
sample from a normal distribution with mean $nA$ and variance $nB^2$.

Using the computational rules for expectations and variances, and the fact that a linear function of a normally distributed stochastic variable is also normally distributed, we can easily produce a sample from an approximately normal distribution with specified mean $\mu$ and variance $\sigma^2$ by the following manipulations:

$$S_n \sim N(nA, nB^2) \text{ (approximately)}$$

$$(B\sqrt{n})^{-1}(S_n - nA) \sim N(0, 1)$$

$$\mu + \sigma(B\sqrt{n})^{-1}(S_n - nA) \sim N(\mu, \sigma^2).$$

For a uniform distribution between 0 and 1 we have

$$A = 1/2;$$

$$B^2 = 1/12.$$

By substituting these values we finally arrive at the expression used ($n = 6$ was selected):

$$\mu + \sigma\sqrt{2}(S - 3).$$

where $S$ is the sum of 6 successive samples taken from the internal uniform distribution between 0 and 1.

2.3.3.7 Functions SCLATT and SCLDIF

In order to provide a consistent range for all computed attitudes in the program, it was decided to systematically scale them into the range $[0, 2\pi)$. Function SCLATT provides this capability.

Analogously, function SCLDIF scales all computed attitude differences into the range $(-\pi, \pi]$. 

3 PLOTTING PROGRAM DESCRIPTION

The plotting program acts as a postprocessor to the simulation program: it can only be used after the simulation program has produced a complete binary plot data file. The plotting program, which is completely separated from the simulation program, must be run at an interactive graphical display terminal. The user specifies a selection criterion,
according to which the plot arrays are to be filled from the binary data file; then he can produce plots of any quantity which is predefined in the plotting program.

This chapter contains the following items:
- a general overview of the program set-up (section 3.1);
- a description of the basic plot modules (section 3.2);
- a description of supporting software (section 3.3).

3.1 General overview

The state of the art in plotting software is that nearly every terminal with graphical display capabilities needs its own plotting software. Although some effort already has been spent to remove this limitation through development of plotting programs which produce a terminal-independent plot file and only need a specific postprocessor for each individual graphical display terminal (Refs. 3, 4), these have not yet reached the power of some terminal-dependent packages.

For this reason the plotting software used in the program is the terminal-dependent PLOT-IO software (Ref. 5), developed specifically for terminals of the Tektronix 401X series. For a description of the PLOT-IO modules used the reader is referred to reference 5.

Selection of specific plotting software bounds the set of features that can be included in the application program to be developed. Since, for the application at hand, the capabilities of the PLOT-10 software are more than sufficient, this presents no difficulties. Rather, it becomes necessary to select a small subset of possibilities to keep the program simple, yet flexible.

Each feature implemented in the plotting program belongs to one of the following classes:
1) data selection;
2) data display.

Data selection criteria are fixed throughout an execution of the plotting program; they are described in section 3.2.2 (subroutine PLINIT).

Data display options, however, can be specified for each new plot to be generated; these are discussed in section 3.2.1 (program PLOT).
Module descriptions

Sections 3.2.1 - 3.2.3 contain descriptions of the main program and those modules which are concerned with the program organization. All other modules, except for the PLOT-10 modules, are briefly described in section 3.3.

3.2.1 Program PLOT

In this module the desired graphs are displayed on the graphical terminal according to the specified selection criteria and display options. As already observed in section 3.1, the selection criteria are established only once per execution of the plotting program; this is handled by subroutine PLINIT (section 3.2.2). The display options implemented at present are:

- the user can specify his own maximum and minimum values for the x- and y-coordinates; otherwise, automatic scaling is used, ensuring that every (x,y)-pair has coordinates that lie within the screen limits;
- the user can specify that, after a curve has been plotted (with user-defined or automatic scaling), a second curve must be plotted on the same graph using the same x-values and the same y-axis scaling. This second curve is then plotted as a dashed line while the first is a solid line.

Although the majority of plots that will be generated with the program will show a certain quantity as a function of time, occasionally one needs to plot quantities against each other. For instance, a phase plane plot of the commanded motion following errors can be generated by plotting the velocity error versus the attitude error. The plotting program provides the possibility to select any two quantities as x- and y-values, and any third one to plot on the same graph.

A flowchart of program PLOT is given in figure 3.1. The various descriptive text blocks should be clear. One specific action, however, must be described in more detail: the filling of the plots arrays (i.e. providing the (x,y)-coordinate pairs to appear on the screen) from the binary data file. This is handled with the aid of subroutine FILL (section 3.2.3).
3.2.2 Subroutine PLINIT

This module performs two major tasks:
1) establishment of the data selection criterion;
2) initialization of the PLOT-10 plotting software.

The latter simply consists of a call to the PLOT-10 initialization routine INITT with specification of the transmission rate of characters from the host computer to the graphical display terminal.

In order to define the type of data selection criterion used it is necessary to briefly discuss the structure of the plot data file and the actual plotting process. The structure of the binary data file produced by the simulation program is depicted in figure 3.2. As can be seen from the figure, the file consists of two initial parameter records, followed by a run dependent number of data records, one for each sampling instant.

The plotting process consists of the following four steps:
1) reading the binary data file;
2) processing these temporary values to obtain the desired quantities;
3) storing this information in the plot arrays;
4) generating a graph from these arrays.

The last step is entirely performed by the PLOT-10 plotting software; the other three steps are the responsibility of the application program.

There are two reasons for limiting the size of the plot arrays:
- the arrays reside in the computer's central memory; the larger they are, the larger the load on the computer system, possibly leading to an increase in turnaround time;
- if the curve to be plotted is the graph of a function (i.e. to every x-value there corresponds only one y-value) it is useless to use more points then the actual resolution of the terminal screen, since an individual screen point is the smallest unit of information.

Therefore the size of the plot arrays is derived from the screen resolution (4096 x 4096 for a Tektronix 4014 terminal with Enhanced Graphics Module). It is clear that for data files having more then 4096 data records a selection must be made; this is achieved as follows.

First the run identification information, number of data records on the plot file and simulation time interval will be displayed on the screen. The user is then prompted for three parameters:
- the end points of the time interval he wishes to consider (two values);
- the number of points to skip after each data point (one value).
The latter parameter is useful for plotting very large time intervals or obtaining a quick look, by using for example only one in every 20 datapoints.

These three parameters constitute the data selection criterion, which will be used throughout the same execution of the plotting program. It must be noted, however, that for time plots the time interval of the plot can be further reduced (but not expanded) by specifying appropriate x-coordinate scaling (see section 3.2.1).

Finally, these values are checked with the available numerical run information to ensure that the user-defined time interval is within the available simulation interval, and that the plot arrays are filled properly. If a user-specified selection criterion leads to an incorrect number of datapoints, he is reprompted for the correct values.

The module ends with displaying a table of quantities available for plotting and a call to the PLOT-10 initialization routine mentioned before. A flowchart of the module is given in figure 3.3.

3.2.3. Subroutine FILL

The function of this module is to compute desired quantities from the raw data read from the binary plot file. All numerical values on the plot data file which represent physical quantities are expressed in SI-units. However, as has been argued in section 2.3.3.3 (subroutine OUTPUT), sometimes other units are better suited to interpretation.

For some quantities, like actual spacecraft attitude, the only processing done is conversion to other units, while others are computed from two or more raw data values. For a list of quantities that can be produced in this module the reader is referred to table 1.

3.3 Supporting software

Two auxiliary modules exist in the plotting program. The first is a simple module which uses some of the basic PLOT-10 character output routines to provide the plot with the units used for x- and y-values (subroutine PLUNIT); it is called from program PLOT and will not be described further. The second is the function SCLDIFF described previously in section 2.3.3.7; it is used in subroutine PUT.
IMPLEMENTATION CONSIDERATIONS

This chapter contains some additional information about the version of the program in use at NLR. It may be of use to anyone using the software at NLR or transporting it to another computer system.

The chapter is organized as follows. Section 4.1 presents some information about the programs' source code. Section 4.2 describes the memory requirements for execution of either program and how these can be reduced using program segmentation; also it contains information about execution times. Finally, section 4.3 contains some observations about maintenance of the software.

4.1 Program code

The two programs discussed in the previous chapters are written in FORTRAN 77; they comply with the ANSI standard (Ref. 6) with the following exceptions:

- for noise simulation in function NORMAL (section 2.3.3.6) the intrinsic function RANF is used. This non-ANSI function returns upon calling a real number from a uniform distribution between 0 and 1. Since the routine is coded in assembly language, it cannot be transported to other operating systems (except of course the related CDC operating systems).

- one of the PLOT-10 routines used (CHECK) has been renamed locally at NLR to AGCHECK, which is a 7-character symbol and therefore non-ANSI. This is not a problem in itself, since a different implementation of the program will either have to use other plotting software, as PLOT-10 may not be available, or the local names for any of the PLOT-10 routines will conform to the standards of the particular FORTRAN 77 compiler.

The simulation program consists of a main program and 26 subprograms. The number of FORTRAN source lines is about 2400, of which approximately 50% consists of comment. Depending on the various compiler options selected it takes the compiler between 1 and 4 CPU seconds and between 24000 and 27000 60-bit central memory words to translate the source program into relocatable binary object code. Of course, these data are specific to the NLR computer network, in which a Control Data CYBER 170-855 serves as the main computer.
The plotting program consists of a main program and 5 subprograms. This number does not include the PLOT-10 modules, which are loaded at execution time from a system library. The number of FORTRAN source lines here is about 700, of which 55% is comment. Compilation requirements vary between 24000 and 25000 central memory words and between 0.2 and 1 CPU seconds.

4.2 Execution statistics

The amount of central memory required for execution of the two programs depends on the optimization level selected at compilation time, as well as on the amount of output requested from the simulation program. Using high optimization (OPT = 2, see Ref. 7) the maximum execution field length for the simulation program, with all output options selected, is 21000 words, and that for the plotting program is 34000 words.

However, due to the logical structure of the simulation program, a field length reduction can be obtained by segmentation of the program, instead of using a normal load sequence (Ref. 8). Although in the present case the reduction is moderate (about 1500 central memory words) it may be worthwhile for long simulation runs. Moreover, the method used applies to many similar programs; in some cases even drastic improvements are possible.

Consider the flowchart of program SIMUL (Fig. 2.1). For each simulation run the primary initialization routine INIT is called only one, at the beginning; it is never used again during the entire rest of the run. Normally, i.e. with a basic load sequence, this module would occupy a certain fixed part of computer memory all the time. With the segment loader, however, it is possible to re-use that piece of storage for other modules, after the routine has come to completion. Of course, not only the body of INIT is affected, but also any module used only within INIT, such as the modules DATIME, INPUT and INPOUT (compare Fig. 2.2).

In principle, the same process could be applied to the interactive plotting program; however, it is considered to be less useful there. Besides, it would be quite difficult to subdivide the plotting routines due to their complex interactions.

Finally, some figures about typical simulation execution times are provided. Since disk input/output is an important factor contributing to both memory requirements and execution time, this will strongly vary with the processing options selected.
To give some indication of the processing time needed for a particular simulation, data are collected in table 2 for a number of simulations with all output options selected (i.e. both complete line printer output and a binary plot data disk file). The term "simulation output frequency" refers to the number of times the line printer output routines are called per second.

4.3 Software maintenance

The source text of the two programs is maintained on the Control Data computer system at NLR in the form of an UPDATE old program library (Ref. 9). UPDATE is a Control Data program very useful for maintenance and manipulation of software in source code form. Important qualities of UPDATE are the possibility of simultaneous maintenance of several versions of a program and the relatively simple testing of software modifications.

A very useful property is the possibility of identifying a piece of code occurring at more than one place in a program as a so-called "common deck". For instance, all common block descriptions in the two programs are separate "common decks". The reader is referred to reference 9 for a description of these and other characteristics of UPDATE.

REFERENCES


   PASPLOT - a Pascal plotting package.
Pascal software tools write-up,
University Computer Centre, University of Minnesota, 1979.

5. PLOT-10: Terminal Control System 3.0
   user's manual/PLOT-10: Advanced Graphing
   II user's manual.

6. American National Standard Programming
   Language FORTRAN.
   ANSI X 3.9-1978, American National
   Standards Institute, 1978.

7. FORTRAN Version 5 Reference Manual,
   Revision E.
   Publication no. 60481300,
   Control Data Corporation, 1981.

8. CYBER Loader Version 1 Reference Manual,
   Revision E.
   Publication no. 60449900,
   Control Data Corporation, 1982.

9. UPDATE Version 1 Reference Manual,
   Revision E.
   Publication no. 60449900, Control
   Data Corporation, 1982.
<table>
<thead>
<tr>
<th>No.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Time (s)</td>
</tr>
<tr>
<td>2</td>
<td>Commanded attitude (degrees)</td>
</tr>
<tr>
<td>3</td>
<td>Commanded velocity (arcmin/s)</td>
</tr>
<tr>
<td>4</td>
<td>Actual spacecraft attitude (degrees)</td>
</tr>
<tr>
<td>5</td>
<td>Actual spacecraft velocity (arcmin/s)</td>
</tr>
<tr>
<td>6</td>
<td>Reaction wheel torque (Nm)</td>
</tr>
<tr>
<td>7</td>
<td>External disturbance torque (Nm)</td>
</tr>
<tr>
<td>8</td>
<td>Actual gyro drift rate bias (arcsec/s)</td>
</tr>
<tr>
<td>9</td>
<td>Actual gyro scale factor error (o/oo)</td>
</tr>
<tr>
<td>10</td>
<td>Estimated spacecraft attitude (degrees)</td>
</tr>
<tr>
<td>11</td>
<td>Estimated spacecraft velocity (arcmin/s)</td>
</tr>
<tr>
<td>12</td>
<td>Estimated total disturbance torque (Nm)</td>
</tr>
<tr>
<td>13</td>
<td>Estimated gyro drift rate bias (arcsec/s)</td>
</tr>
<tr>
<td>14</td>
<td>Estimated gyro scale factor error (o/oo)</td>
</tr>
<tr>
<td>15</td>
<td>Spacecraft state estimator measurement residual (arcsec)</td>
</tr>
<tr>
<td>16</td>
<td>Spacecraft state estimator gain switch</td>
</tr>
<tr>
<td>17</td>
<td>Computed spacecraft target attitude (degrees)</td>
</tr>
<tr>
<td>18</td>
<td>Computed spacecraft target velocity (arcmin/s)</td>
</tr>
<tr>
<td>19</td>
<td>Computed feedforward target control torque (Nm)</td>
</tr>
<tr>
<td>20</td>
<td>Computed feedback regulator control torque (Nm)</td>
</tr>
<tr>
<td>21</td>
<td>Computed total control torque (Nm)</td>
</tr>
<tr>
<td>22</td>
<td>Computed reaction wheel setpoint</td>
</tr>
<tr>
<td>23</td>
<td>Reaction wheel torque error (Nm)</td>
</tr>
<tr>
<td>24</td>
<td>Reaction wheel angular velocity (rad/s)</td>
</tr>
<tr>
<td>25</td>
<td>Actual spacecraft attitude increment (arcsec)</td>
</tr>
<tr>
<td>26</td>
<td>Number of counted gyro pulses</td>
</tr>
<tr>
<td>27</td>
<td>Computed gyro reference attitude (degrees)</td>
</tr>
<tr>
<td>28</td>
<td>Computed corrected gyro output (arcsec)</td>
</tr>
<tr>
<td>29</td>
<td>Actual total gyro drift rate (arcsec/s)</td>
</tr>
<tr>
<td>30</td>
<td>Star sensor attitude measurement error (arcsec)</td>
</tr>
<tr>
<td>31</td>
<td>Actual total disturbance torque (Nm)</td>
</tr>
<tr>
<td>32</td>
<td>Actual total spacecraft attitude error (arcsec)</td>
</tr>
<tr>
<td>33</td>
<td>Actual total spacecraft velocity error (arcsec/s)</td>
</tr>
<tr>
<td>34</td>
<td>Spacecraft attitude estimation error (arcsec)</td>
</tr>
<tr>
<td>35</td>
<td>Spacecraft velocity estimation error (arcsec/s)</td>
</tr>
<tr>
<td>36</td>
<td>Spacecraft disturbance torque estimation error (Nm)</td>
</tr>
<tr>
<td>37</td>
<td>Gyro drift rate bias estimation error (arcsec/s)</td>
</tr>
<tr>
<td>38</td>
<td>Gyro scale factor error estimation error (o/oo)</td>
</tr>
<tr>
<td>39</td>
<td>Estimated spacecraft attitude - gyro reference attitude (arcsec)</td>
</tr>
<tr>
<td>40</td>
<td>Estimated total spacecraft attitude error (arcsec)</td>
</tr>
<tr>
<td>41</td>
<td>Estimated total spacecraft velocity error (arcsec/s)</td>
</tr>
<tr>
<td>42</td>
<td>Computed spacecraft target attitude error (arcsec)</td>
</tr>
<tr>
<td>43</td>
<td>Computed spacecraft target velocity error (arcsec/s)</td>
</tr>
<tr>
<td>44</td>
<td>Estimated model following attitude error (arcsec)</td>
</tr>
<tr>
<td>45</td>
<td>Estimated model following velocity error (arcsec/s)</td>
</tr>
</tbody>
</table>

Note: The first 30 quantities are taken directly from the binary plot data file, after conversion to the desired units; the remaining 15 quantities are computed from two or more of the first 30.
TABLE 2
Simulation program execution time for various runs
(see chapter 4 for additional information)

<table>
<thead>
<tr>
<th>User-specified run time (s)</th>
<th>Simulation output frequency (s⁻¹)</th>
<th>Execution time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>90</td>
<td>2</td>
<td>5.9</td>
</tr>
<tr>
<td>1000</td>
<td>2</td>
<td>50.5</td>
</tr>
<tr>
<td>1600</td>
<td>2</td>
<td>92.8</td>
</tr>
<tr>
<td>6000</td>
<td>0.1</td>
<td>83.6*</td>
</tr>
<tr>
<td>9000</td>
<td>0.1</td>
<td>129.5*</td>
</tr>
<tr>
<td>20000</td>
<td>0.01</td>
<td>205.5</td>
</tr>
</tbody>
</table>

*) obtained as the mean of two identical runs.

Note: These figures apply to the CYBER 73-28 computer, which was recently replaced by a CYBER 170-855 system. Experience with both systems indicate that the execution times for the simulation program have decreased by a factor 4 to 5.
Fig. 1.1 Spacecraft configuration
Fig. 1.2 Simulation overview
Fig. 2.1 Program SIMUL flow chart
PERFORM VARIOUS INITIALIZATIONS AND NORMALIZATIONS

SQUARE STANDARD DEVIATIONS TO OBTAIN VARIANCES

RETURN

Fig. 2.2 Module INIT flow chart
SAVE OLD ATTITUDE, VELOCITY AND SECTOR NUMBER

RWL

DISTURBANCE TORQUE MODEL

COMPUTE NEW ATTITUDE, VELOCITY AND SECTOR NUMBER

SECTOR NUMBER CHANGED?

Y

SSE

N

GYRO

t = t + ΔT

RETURN

Fig. 2.3 Module SATACS flow chart
REQUESTED TORQUE

SETPOINT SIGN CHANGE?

Y

NOI-LINEARITY

N

WHEEL SPEED ZERO?

Y

SATURATION MODEL

N

COULOMB-AND VISCOUS FRICTION

STICITION

TORQUE NOISE

ZERO WHEEL SPEED EFFECTS

SAVE OLD SETPOINT AND WHEEL SPEED

CALCULATE NEW WHEEL SPEED

RETURN

Fig. 2.4 Module RWL flow chart
Fig. 2.5 Star measurement truth modelling

```
  0  |  1  | k-1  | k  | k+1  | N  |
    |     |      |    |      |    |
<--- SECTOR 1 --- SECTOR k --- SECTOR 1 --->

2π
```

Fig. 2.6 Module GYRO flow chart

```
DRIFT MODEL

SCALE FACTOR ERROR MODEL

GYROSCOPE MODEL

RETURN
```

Fig. 2.7 Blockdiagram of the OBC gyro output correction

```
NGYR (k) * RE(1 + \hat{E}_k)

\hat{\Delta}_b_k

\Delta T

\hat{\Delta}_g_c (k)

\Delta \phi_{gc} (k)
```
TRANSFORM ABSOLUTE SSE ATTITUDE MEASUREMENT TO PREVIOUS SAMPLING TIME

ADJUST GYRO STATE VECTOR LENGTH TO ESTIMATION MODE

COMPUTE TIME-VARYING ESTIMATOR GAINS

COMPUTE NEW GYRO STATE VECTOR

UPDATE GYRO MEASURED AND ESTIMATED ATTITUDE; RESET FIRST ELEMENT OF GYRO STATE VECTOR

RESET INTEGRATION COUNTERS FOR RECURSIVE COVARIANCE COMPUTATION

RECURSIVELY COMPUTE STATE ERROR COVARIANCE

RETURN

Fig. 2.8 Module GYREST flow chart
Fig. 2.9 Module STATE flow chart
Fig. 2.10 Module CONTRL flow chart
PLINIT

LET USER SPECIFY QUANTITIES TO BE PLOTTED

FILL THE PLOT ARRAYS FROM THE BINARY DATA FILE

USER-SPECIFIED SCALING?

N

LET USER ENTER EXTREME VALUES FOR X AND Y

Y

PLOT FIRST CURVE

ADD TEXT TO THE GRAPH

2 CURVES REQUESTED?

N

PLOT SECOND CURVE ON SAME GRAPH

Y

CONTINUE?

Y

STOP

Fig. 3.1 Program PLOT flow chart
<table>
<thead>
<tr>
<th>RECORD 1</th>
<th>RUN IDENTIFICATION INFORMATION (TEXT)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RECORD 2</td>
<td>RUN CONSTANTS (NUMERICAL VALUES)</td>
</tr>
<tr>
<td>RECORD 3</td>
<td>VARIABLES AT SAMPLING INSTANT 1</td>
</tr>
<tr>
<td></td>
<td>(NUMERICAL VALUES)</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>RECORD N+2</td>
<td>VARIABLES AT SAMPLING INSTANT N</td>
</tr>
<tr>
<td></td>
<td>(NUMERICAL VALUES)</td>
</tr>
</tbody>
</table>

Fig. 3.2 Structure of the binary plot data file
READ INITIAL TWO PARAMETER RECORDS FROM DATA FILE

PROMPT USER FOR DATA SELECTION CRITERION

CHECK ADMISSIBILITY OF CRITERION

ILLEGAL?

Y

N

DISPLAY TABLE OF AVAILABLE QUANTITIES

INITT

RETURN

Fig. 3.3 Module PLINIT flow chart
NATIONAAL LUCHT- EN
RUIMTEVAARTLABORATORIUM

Anthony Fokkerweg 2, 1059 CM AMSTERDAM
Postbus 90502, 1006 BM AMSTERDAM
Tel. (020) 5113113 - telex 11118 (niraa nl)
Telegram-adres: Windtunnel Amsterdam