MCR-84-502
Contract No. NAS8-34679

Phase 3
Final
Report

January 1984

Development of an Autonomous Video Rendezvous and Docking System

(NASA-CR-170969) DEVELOPMENT OF AN AUTONOMOUS VIDEO RENDEZVOUS AND DOCKING SYSTEM, PHASE 3 Final Report (Martin Marietta Corp.) 105 p HC A06/MF A01 CSCL 22B G3/18

N84-17249

Unclas
18288

MARTIN MARIETTA

MCR-84-502 | Phase 3 Final Report

Contract No. NAS8-34679 | January 1984

DEVELOPMENT OF AN AUTONOMOUS VIDEO RENDEZVOUS AND DOCKING SYSTEM

John C. Tietz

MARTIN MARIETTA AEROSPACE
DENVER AEROSPACE
P.O. Box 179
Denver, Colorado 80201

FOREWORD

This report presents the results of a six-month study by Martin Marietta for the National Aeronautics and Space Administration's George C. Marshall Space flight Center. The study was the third phase of Contract NAS8-34679, Development of an Autonomous Video Rendezvous and Docking System. It resulted in improvements to the spacecraft video guidance system developed under previous phases of the contract.

CONTENTS

Figure

Table

Summary

I. SUMMARY

Improvements have been made to the video rendezvous and docking system developed under this contract. The changes allow the system to dock with targets tumbling twice as fast as the old system could accommodate. They also improve reliability at lower tumble rates. The improved performance results from:

1) Adding a second Kalman filter to improve estimates of target attitude and allow anticipation of target attitude changes;

2) Changing the guidance strategy to make use of the data from the Kalman filter.

Other minor changes were made to improve performance. Larger thrusters were used on the sides of the chase spacecraft, and a higher-resolution (broadcast quality) television camera replaced the original 128-line camera.

Improving performance further will probably require multiple docking aids or an auxiliary radio frequency (RF) system, because the system is now limited primarily by the docking aid rolling out of sight behind the target. Although the Kalman filter allows dead reckoning, the accuracy of its position estimates deteriorates with time, especially when the chase spacecraft attempts to maneuver around the target.

Application of artificial intelligence in the guidance system might minimize this problem, and precision accelerometers could slow the growth of estimation error. However, the problem will still be difficult to solve without some form of additional sensor data from the back side of the target spacecraft.

Introduction

II. INTRODUCTION

The study reported here was the third phase of a contract to investigate techniques that could be used in an autonomous video rendezvous and docking system for spacecraft.

Under the first phase of the contract, we identified several techniques that appeared suitable for such a system, defined the equations and algorithms these techniques would use, and evaluated video guidance control systems based on these techniques through computer simulation.

To ensure that practical problems were considered, the simulation modeled not only the sensor, but also methods for dealing with a number of practical problems, e.g., maintaining control when the target spacecraft leaves the field of view of the guidance sensor. The simulation also modeled the characteristics and limitations of practical spacecraft to reveal subtle incompatibilities that might otherwise go unnoticed. A mission model was defined to serve as a basis for the simulation.

In this model, the chase vehicle (Fig. II-1) is a general-purpose spacecraft for repair, refurbishment, and retrieval of other spacecraft. After it is deployed from the Space Shuttle, it must rendezvous and dock with the long-duration exposure facility (LDEF), which, it is assumed, has been modified for this operation and is in a circular orbit at an altitude of 300 km. We will refer to LDEF as the target spacecraft, because, although a specific mission model was used for the simulations, the intent was that the guidance method be usable on a variety of spacecraft.

In the second phase of the contract, we conducted a physical simulation of the best technique evaluated under the first phase. This technique used a docking aid comprising three flashing lights mounted on the target spacecraft (Fig. II-2). The appearance of this pattern of lights uniquely defines both the relative positions and the relative attitudes of the two spacecraft.

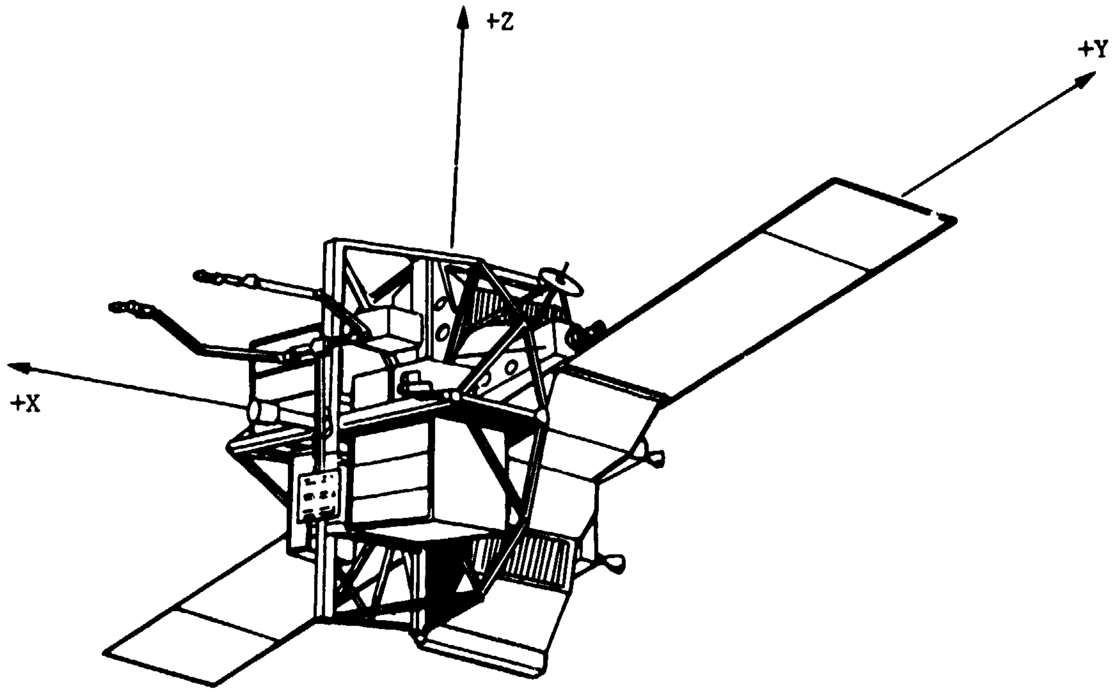


Figure II-1 Chase Vehicle Modeled in Simulation

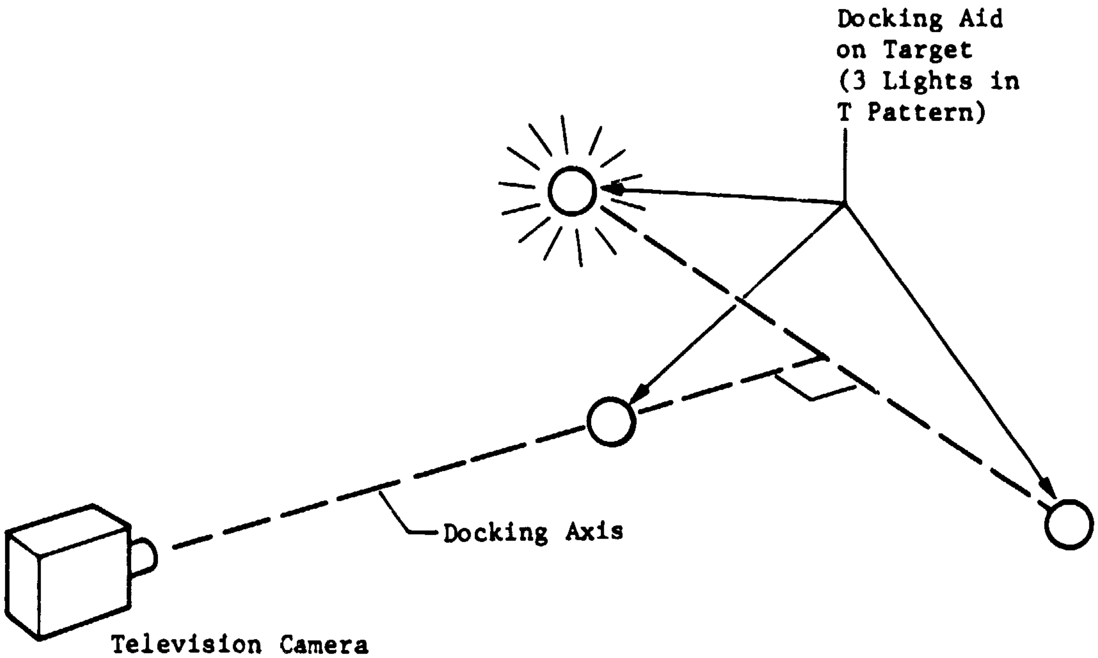


Figure II-2 Three Light Docking Aid

To simulate the entire operation from a range of 300 m to contact, three target-spacecraft models were required (Fig. II-3). Each model was built to a different scale and was used in a different part of the simulation. The smallest model was 1/100 scale and was used for ranges greater than approximately 30 m. A 1/10 scale model was used to simulate ranges between 3 and 30 m. For the final seconds of the docking operation, a full-scale model of a portion of one side of LDEF was used.

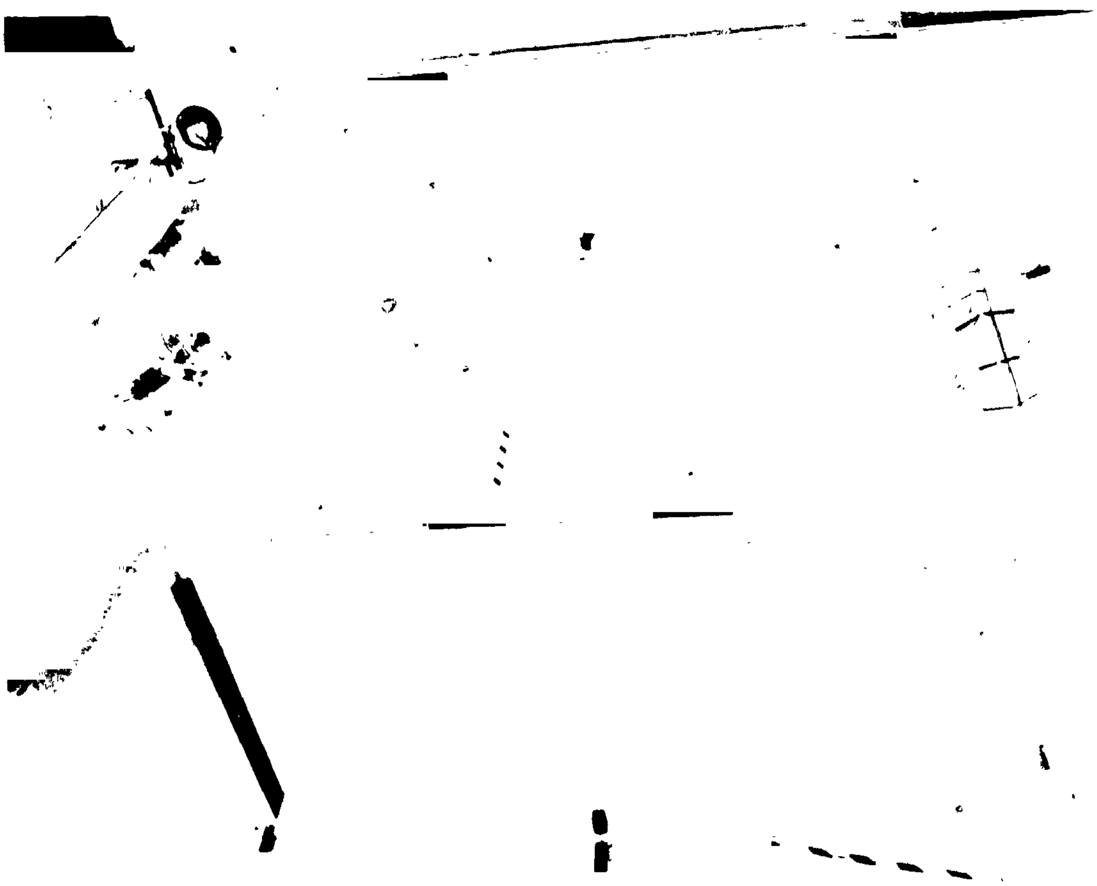


Figure II-3 Scale Models and Simulator Used for Physical Simulation

To simulate the servicer spacecraft (chase vehicle), we mounted a television camera on a six-degree-of-freedom simulator. The simulation computer sent servo commands to position the camera so that the television image would correspond to what a flight camera on a real chase vehicle would see. Video processing electronics (Fig. II-4) converted the imagery to a set of statistics that a computer can quickly analyze to determine the relative positions and attitudes of the two spacecraft. These statistics were transmitted to the simulation computer, which modeled the activity of the simulated flight computer and the dynamics of the two spacecraft.

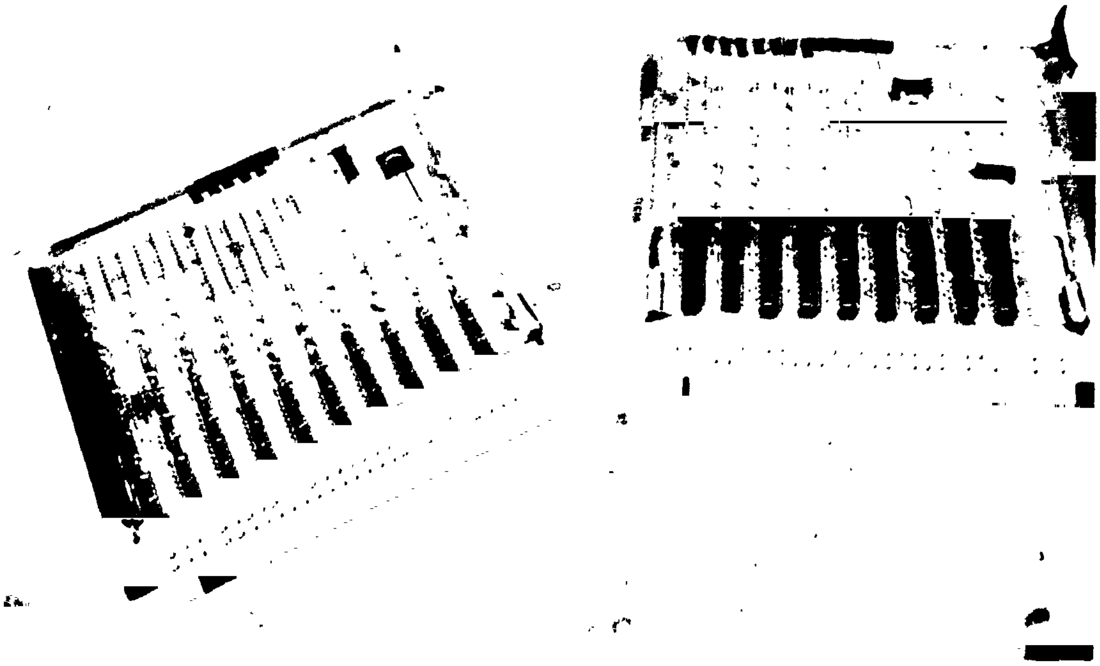


Figure II-4 Video Processing Electronics Used in Physical Simulation

Although the work under these two phases demonstrated the apparent practicality of a video guidance system, improvements were required for docking with tumbling spacecraft. The original system was unable to cope with target attitude rates materially over 1000 deg/h, and was unreliable at this rate.

Part of the problem was that the Kalman filter, which the guidance system used for dead reckoning, did not keep track of target attitude. This made it nearly impossible for the chase vehicle to recover gracefully when the docking aid on the target rotated out of view behind the target (Fig. II-5).

ORIGINAL PAGE IS
OF POOR QUALITY

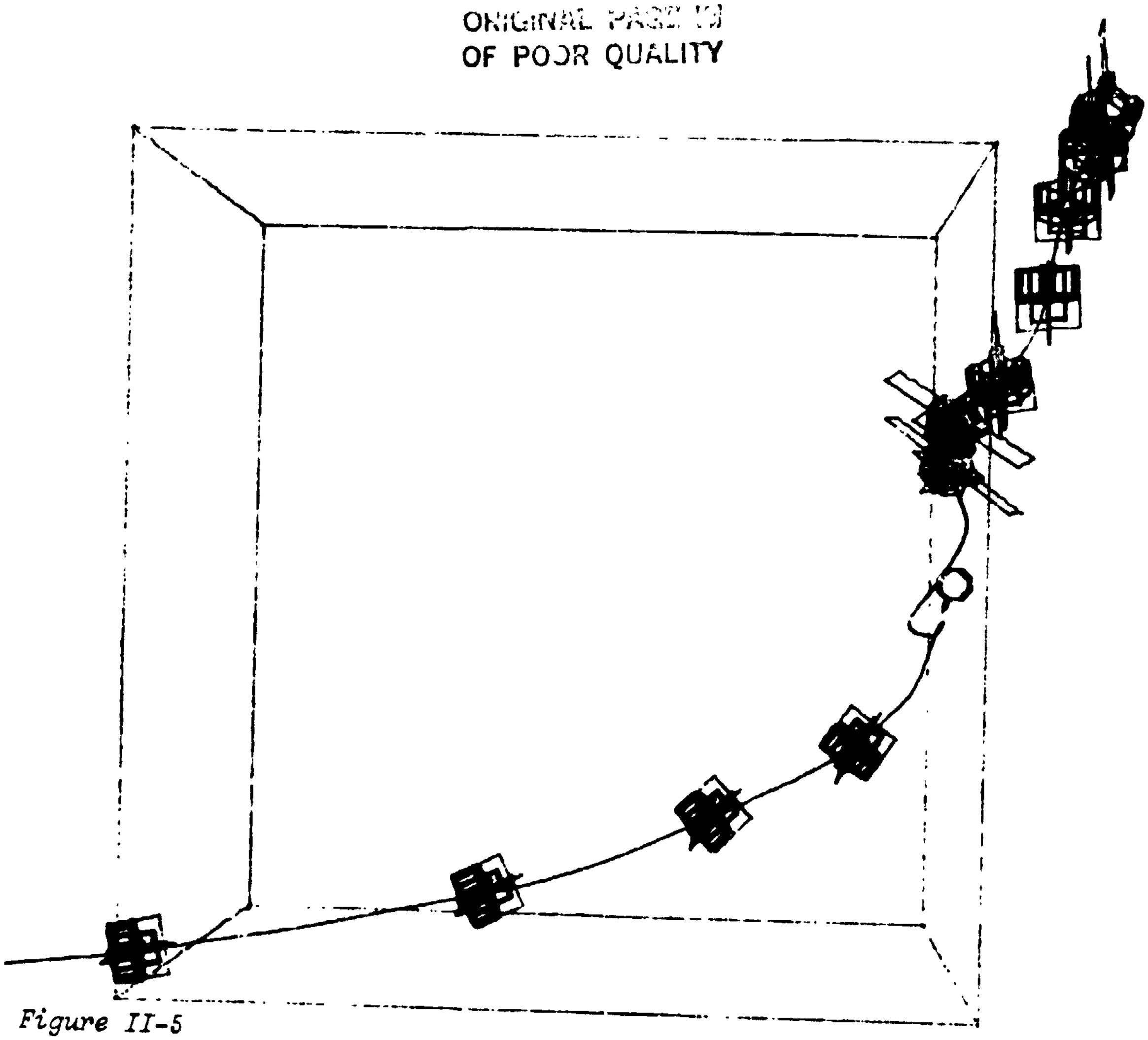


Figure II-5
Poor Recovery Characteristics of Old System When Docking Aid Could Not Been Seen

Another problem was strategy logic that did not plan ahead for a rotating target: the chase vehicle built up too much speed in approaching the target, using powerful thrusters at the rear of the spacecraft. When it arrived in the vicinity of the target, the target had rotated, and the chase vehicle had to fly sideways for the last few meters. It was then unable to stop quickly enough with the weaker side thrusters and overshot the target (Fig. II-6).

The activity under Phase 3 addressed these shortcomings by making improvements in the strategy logic and augmenting the Kalman filter to estimate target attitude and tumble rate.

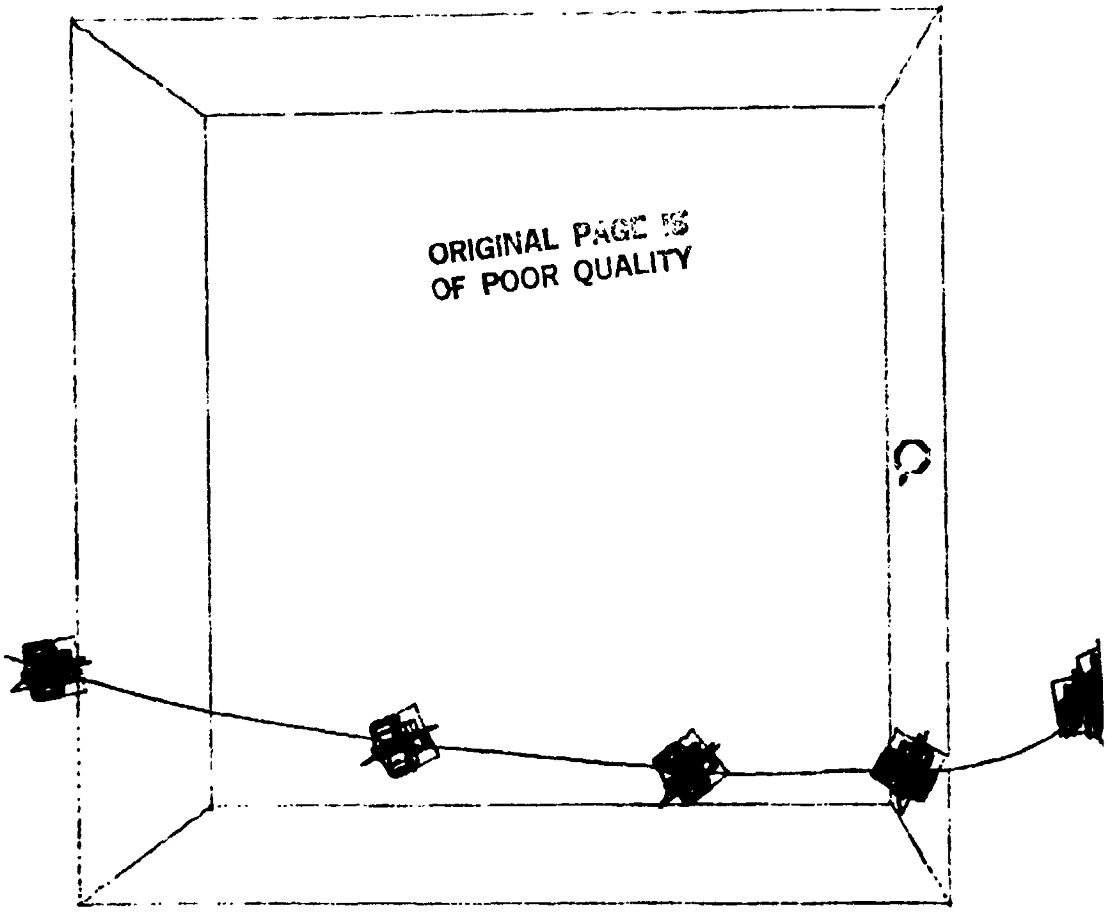


Figure II-6 Overshoot Problem with Old System

This report concentrates on the third phase of the contract and does not repeat very much of the information that was published in the final reports for Phases 1 and 2. The reader who has not read those reports will find it advantageous to read them before reading the more technical sections of this volume.

Conclusions and Recommendations

III. CONCLUSIONS AND RECOMMENDATIONS

A. NEW SYSTEM WORKS WITH HIGHER TARGET ATTITUDE RATES

The changes made under this contract phase have approximately doubled the tumble rates the chase vehicle can accommodate. This improvement was achieved primarily by:

1) Adding a second Kalman filter, which estimates target attitude and angular momentum;

2) Changing the goal-selection and attitude error computations.

The system now works reliably at rates up to 1000 deg/h and, depending on initial conditions, can cope with rates up to 4000 deg/h.

B. HIGHER RATES REQUIRE MORE LIGHTS OR AUXILIARY RF SYSTEM

The main factor that now limits rates the system can accommodate is the fact that the docking aid on the target rotates out of view before the chase vehicle can get close to the target. The result is that the chase vehicle must fly a significant distance on dead reckoning. It can do this for a short time, but maneuvering to the far side of the target requires considerable use of its thrusters.

Unfortunately, each time the thrusters are used, the system loses confidence in its position and velocity estimates. This is because the velocity change that results from thruster operation cannot be predicted or measured exactly. The thruster may produce slightly more or less thrust than was anticipated. To be safe, the guidance strategy must take this loss of confidence into account and back away from the target.

If the system cannot get another measurement for an extended period, it must retreat a considerable distance from the target. It is then in a poor position for a second approach when the docking aid is again visible.

If multiple docking aids were provided, the system could always get a position update and would not have to back away from the target. The result would be:

1) A significant savings in fuel consumption;

2) Shorter time of flight;

3) Greatly increased reliability;

4) Ability to accommodate significantly higher tumble rates.

An alternative method of avoiding the problem is an auxiliary RF guidance system that could provide at least range and direction to the target when the docking aid is out of view. Precision accelerometers in the existing system would also help by slowing the growth of estimation error.

C. FIELD-OF-VIEW LIMITATIONS PROVED TROUBLESOME

During the physical simulations under the second contract phase, we found that two different camera lens focal lengths were required. This requirement was confirmed under the current study. At great distances from the target, the system needs a lens with a long focal length to resolve details on the docking aid. At close range, however, such a lens becomes a problem; because of transient attitude excursions, portions of the docking aid frequently leave the camera's field of view. The system then cannot take new measurements and must back away from the target as its position estimation accuracy deteriorates.

We solved this problem by switching focal lengths at a range of 15 m. Alternately, the problem might be solved by using a second, smaller, docking aid, which would be activated after the chase vehicle approached within approximately 15 m. However, even a very small docking aid could leave the field of view of a long lens. Switching lenses appears to be the more practical solution.

D. HIGHER RESOLUTION WAS REQUIRED

We found it necessary to increase the camera resolution to approximately that of commercial broadcast cameras to cope with high tumble rates. The reason was that at rates over approximately 2000 deg/h, the docking aid often rotates out of sight behind the target before the chase vehicle gets close enough to get precision measurements with a lower resolution camera. If it is to go on dead reckoning for a significant distance, starting with a good initial state estimate is vital. The 128-line camera modeled in previous simulations did not provide a good enough estimate.

E. SIDE THRUSTERS WERE TOO WEAK

We found it necessary to increase the thrust authority of the thrusters on the top, bottom, and sides of the chase vehicle. There was a great difference in authority between these thrusters and those mounted on the front and back of the vehicle (an 8 to 1 ratio). The result was that the chase vehicle tended to greatly overshoot the target when it had to brake with the side-mounted thrusters. Part of this problem could be solved by changes to the control law, but these changes were not particularly effective.

At the same time, we increased the torque authority to cure problems with the docking aid leaving the field of view for extended periods during maneuvers.

F. POUNCE STRATEGY WOULD REQUIRE MULTIPLE DOCKING AIDS

During this study, astronauts were practicing with the Manned Maneuvering Unit simulator at Martin Marietta Denver Aerospace. They were training for a mission in which they are to dock with the Solar Maximum Spacecraft, which is, at the time of this writing, tumbling in orbit due to a malfunction. The similarity between their mission and the mission model for the video rendezvous system suggested that we should try to adopt techniques they found effective.

One of the things they learned was that it was most effective to stop at a convenient position close to the target spacecraft and wait for an opportune moment. They would then pounce on the target from close range, matching the tangential velocity component only during the last seconds of flight.

We incorporated this technique into the simulation program and ran a number of simulations. The results were disappointing.

The reason the technique failed became quite obvious: while the chase vehicle was waiting to pounce, the docking aid was on the opposite side of the target spacecraft. Because the system gets data only from observing the docking aid, it had to operate on dead reckoning for two minutes or more. As its confidence in its position deteriorated, it backed away from the target to prevent collision. In doing so, it used its thrusters, and the thrust uncertainty further reduced its confidence in its position estimate. As a result, it backed farther and farther from the target, so when the docking aid was again visible, the chase vehicle was as far from the target as when the simulation started. It went through cycles of approaching and retreating until it ran out of fuel.

The astronauts did not have this problem because they could obtain as much position data from the back side of the target as from the front.

If the chase vehicle could see several docking aids at various locations on the target, it too might be able to make effective use of the strategy. However, with a single docking aid, the most effective approach was to keep the docking aid in view as much as possible.

G. ARTIFICIAL INTELLIGENCE COULD HELP

One of the shortcomings of the guidance system is its inability to reason about the following:

1) Long-range goals - The guidance system treats each decision interval of approximately 1.2 s as a separate problem. It does not plan an optimal trajectory and stick to it; it does not think about the long-range consequences of its decisions. As a result, it often wastes time and fuel in undoing its previous actions.

2) Interaction of goals - The system knows that it must back away from the target for safety when it cannot see the docking aid. But in deciding to back away, it does not consider how much doing so will degrade its position estimates. By reasoning about this, it might decide to postpone the use of thrusters.

3) Alternate strategies - Although the algorithm used in the system does consider a variety of factors (safety, control loop bandwidth requirements, anticipated target motion) it is still a single strategy. The system does not predict the results of alternative strategies and select one. A system that considered alternative plans might perform better.

Although much of the reasoning process for an intelligent guidance system would require numerical computation, a large portion of the task involves symbol manipulation, tree-searching, backtracking and other operations that are difficult to perform in most computer languages. For example, a program to search a decision tree is easiest to write and understand if the computer language used allows recursive function calls, flexible data structures, and automatic garbage collection. FORTRAN is weak in all these operations, and although C and PL/I support some of them, these languages do not offer the flexibility of LISP and its derivatives in solving problems of this type.

A reasonable next step in developing a better guidance system would be to analyze the knowledge-base requirements of such a system and develop knowledge-representation schemes for automated reasoning about the factors discussed previously.

Simulations Results and Discussion

IV. SIMULATION RESULTS AND DISCUSSION

Although almost all the improvement in measurement accuracy in the new system (Table IV-1) can be attributed to the higher resolution television camera, the new system had much better control accuracy. The old system would often dock with more than 45-deg misalignment at target attitude rates as low as 500 deg/h, and at 1000 deg/h, it rarely docked with misalignment less than 15 deg. Furthermore, at rates of 1000 deg/h and above, it frequently crashed into the target or let the target get out of its field of view long enough that it was not able to recover.

Table IV-1 Measurement Errors

Range (m)	Position Errors (m)		Attitude Error (deg)
	Along Chase Vehicle x-Axis (1σ)	Along Chase Vehicle y- and z-Axes (1σ)	Pitch, Yaw, or Roll (1σ)
10	0.141	0.100	0.362
25	0.318	0.0964	0.628
50	1.76	0.303	0.941
100	9.80	0.970	1.85
286	117	6.32	9.33

The new system's performance at these rates is illustrated in the trajectory plots in Figures IV-1 through IV-5. In each of the simulations illustrated, the chase vehicle started from a randomly selected position approximately 300 m from the target. Because problems rarely developed until the range was reduced to 30 or 40 m, the figures show only the last 60 m of the flight. The boxes shown in the figures represent a 60-m cube. Its primary use was to enhance depth perception when stereo pairs of plots were viewed while we were running the experiments.

The primary reason for docking failures at the higher rates was the docking aid's rolling out of sight before the chase vehicle could get close enough to prevent it. This fact is illustrated dramatically in

ORIGINAL PAGE IS
OF POOR QUALITY

Figure IV-3: with a target tumble rate of 2000 deg/h, the lights often rotated out of sight while the chase vehicle was still some distance away. Because the tumble rate was low, the lights did not reappear for six minutes. By this time, the chase vehicle's state estimate had badly deteriorated. Although the chase vehicle was often able to recover from this by going around the target (Fig. IV-6a) or waiting for the docking aid to reappear (Fig. IV-6b), it generally used an excessive amount of fuel (Table IV-2). The success rate at 2000 deg/h was actually lower than at 3000 deg/h.

> Note:
> Neither system had trouble with roll about the docking axis. The study therefore concentrated on pitch and yaw axes.

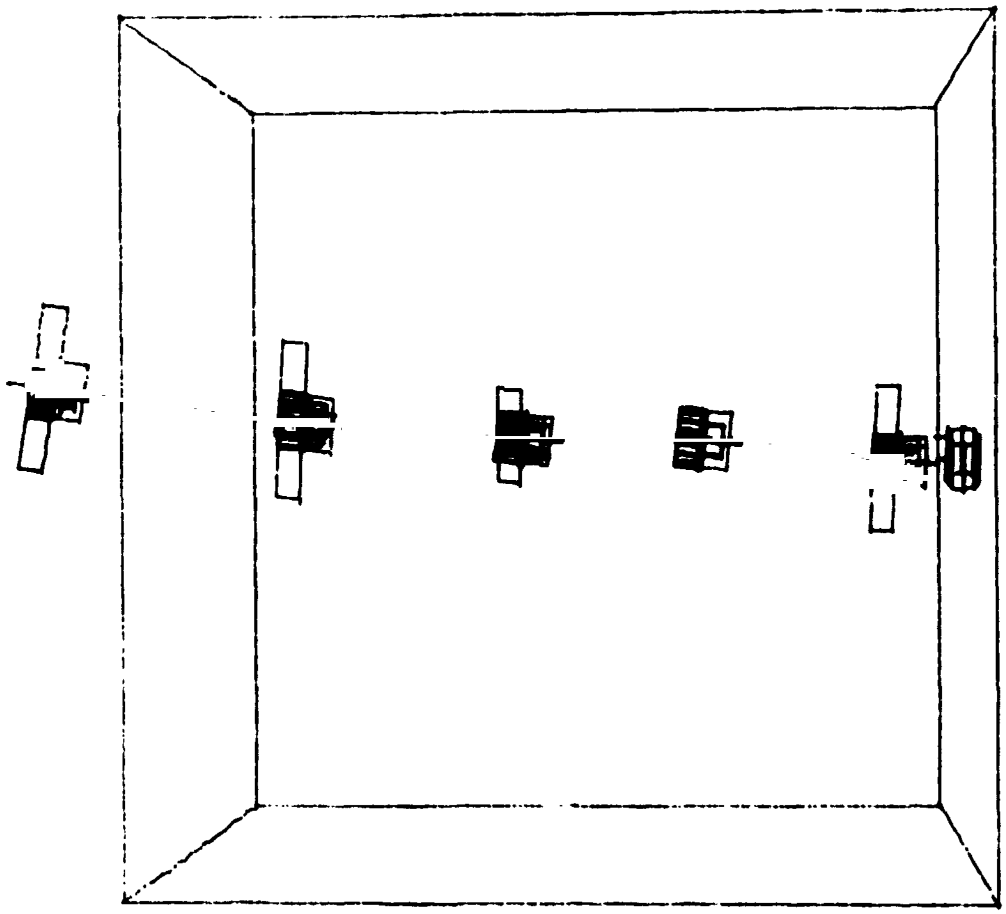


Figure IV-1 Typical Trajectory with Target Roll about Docking Axis

> Note:
>
> In (a) through (d), the pitch axis was the tumble axis. In (e) and (f), the target tumbled about its yaw axis.

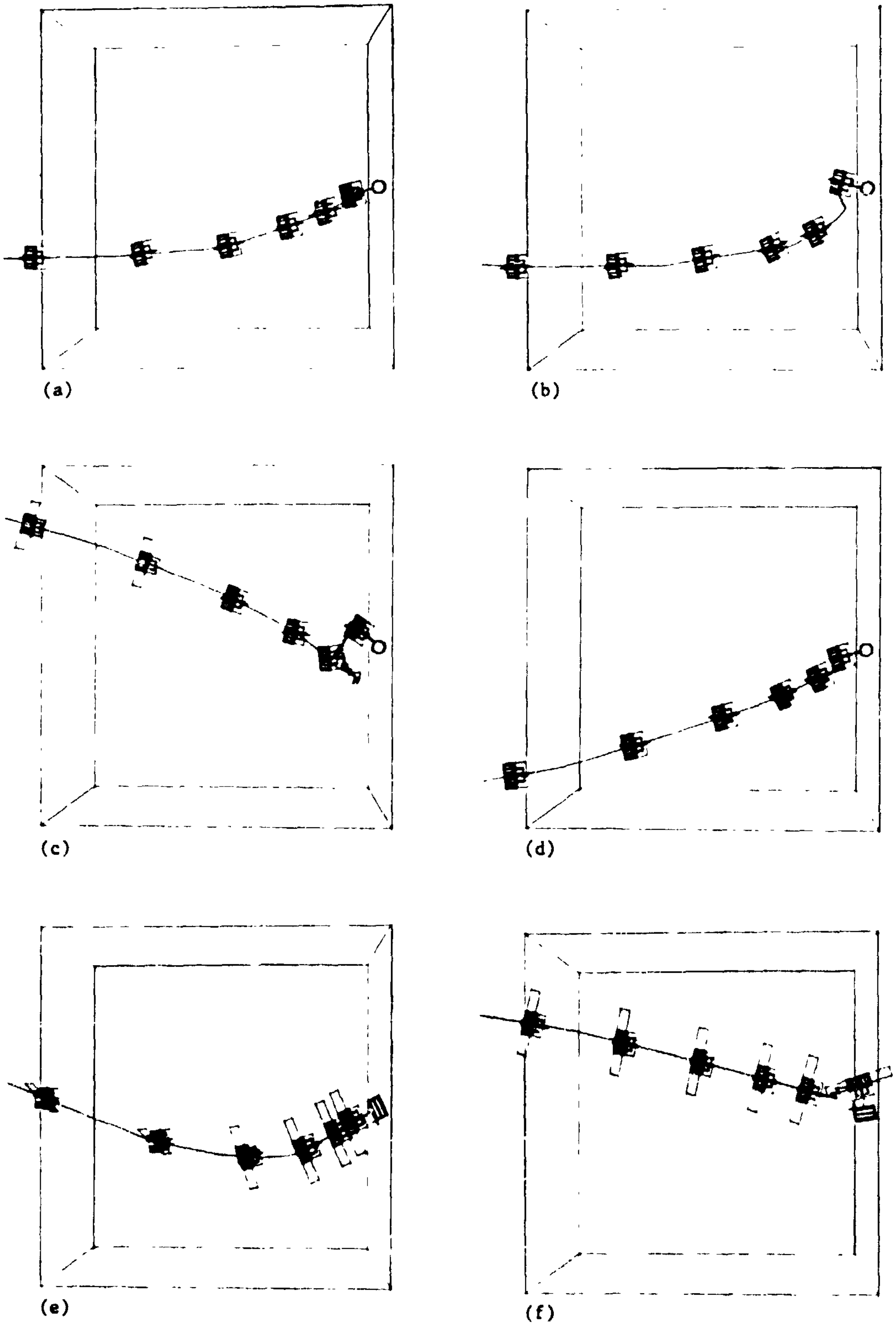


Figure IV-2 Performance with Pitch and Yaw Rates of 1000 deg/h

Note:

A rate of 2000 degrees per hour was the most troublesome, because the docking aid was out of sight for the longest time. In (a) and (b), the pitch axis was the tumble axis; in (c)-(f), the yaw axis was the tumble axis: Simulations (b), (e), and (f) were stopped when an arbitrary time limit was reached.

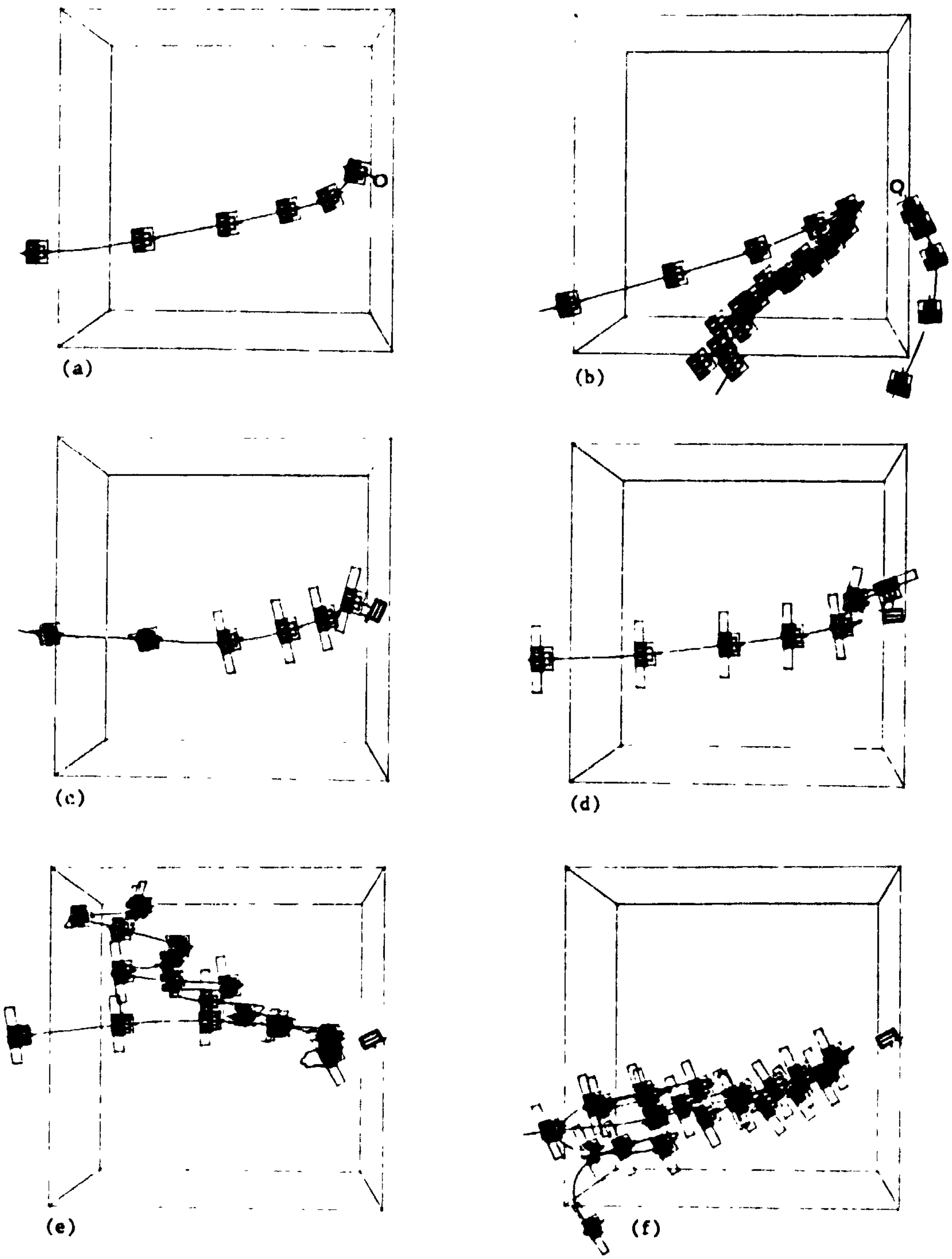
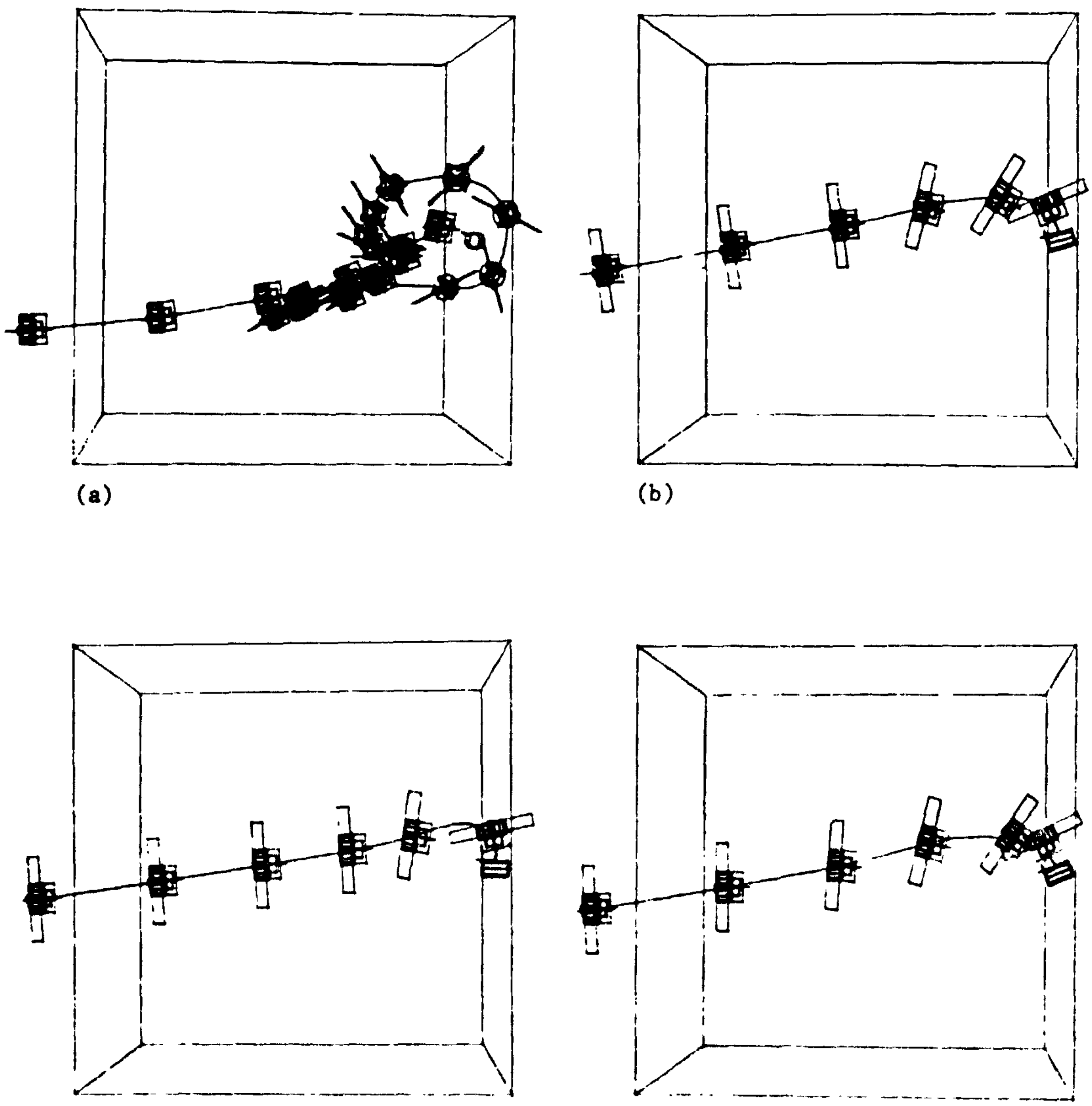


Figure IV-3 Performance with Pitch and Yaw Rates of 2000 deg/h

<u>Note</u>:

Tumble axis was pitch axis in (a), yaw in others.



(a) (b) (c) (d)

Figure IV-4 Performance with Pitch and Yaw Rates of 3000 deg/h

Note:

In (a) and (b), the tumble axis was the yaw axis; in (c)-(f) the tumble axis was the pitch axis. Simulation (e) was stopped when an arbitrary time limit was reached. In (f), the chase vehicle docked, but misalignment was extreme.

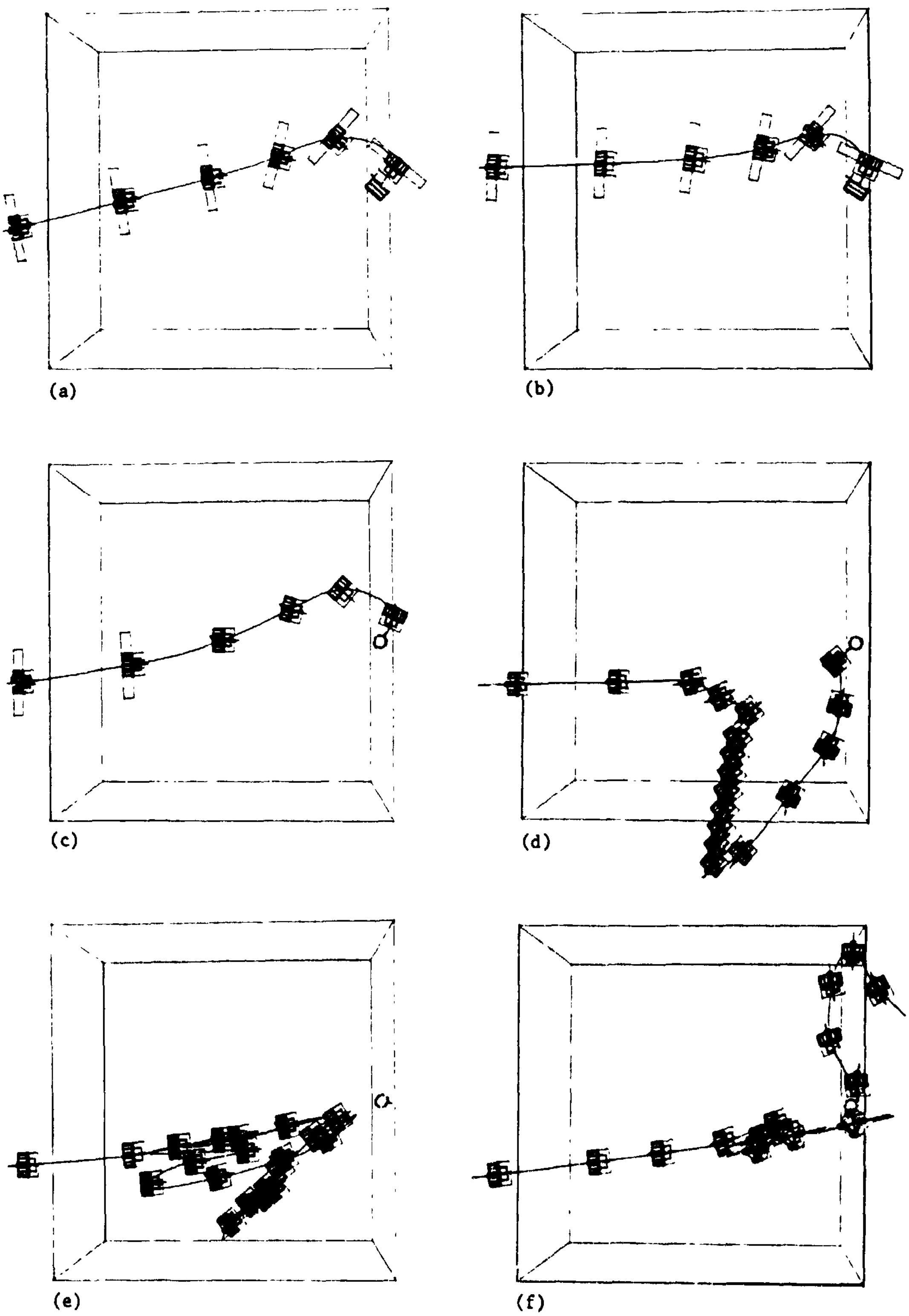
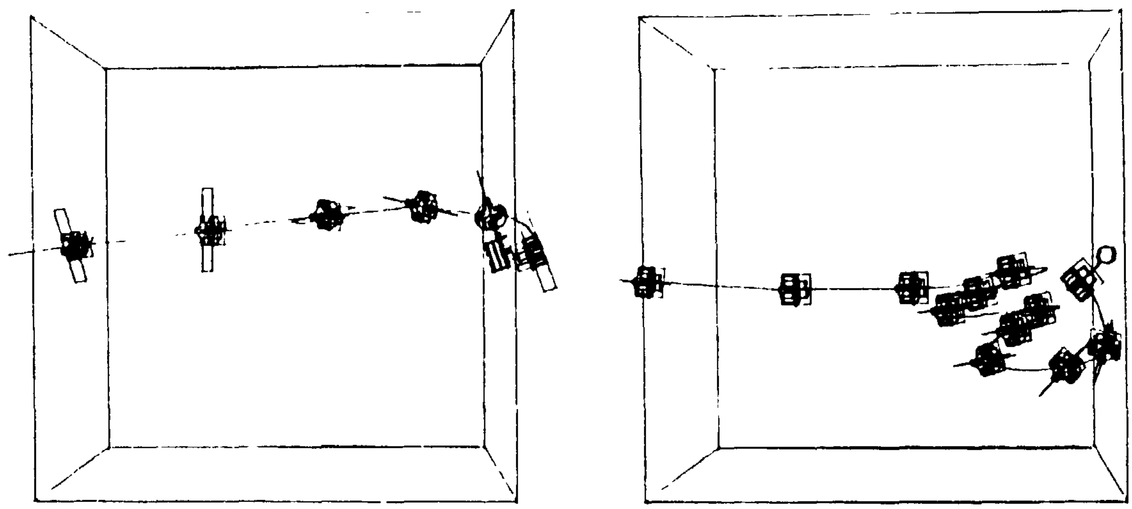


Figure IV-5 Performance with Pitch and Yaw Rates of 4000 deg/h

ORIGINAL PAGE IS
OF POOR QUALITY

Note:

In (a) the chase vehicle found a path around the target. More frequently it did not but waited for the docking aid to reappear (b).



(a) (b)

Figure IV-6 Recovery from Docking Aids Rolling Out of Sight

Table IV-2
Comparison of Fuel Use and Time of Flight for Old and New Systems

Pitch or Yaw Rate (deg/h)	Old System		New System	
	Median Time (s)	Median Fuel Consumption (Kg)	Median Time (s)	Median Fuel Consumption (Kg)
1000	192	28.4	227	53.9
2000	279	54.7	∞*	∞*
3000	201	27.2	∞*	∞*
4000	201	31.2	∞*	∞*

*Chase vehicle did not appear to have a chance of docking after 280 seconds of flight. Simulation terminated.

We partially succeeded in overcoming this problem by using a goal-modification strategy. The guidance logic analyzed the shortest path from the current position to the docked position. If it found that the path would pass too close to the target, it attempted to select an alternative near-term goal on the shortest circular path around the target at an acceptable radius. This approach was not as successful as we had hoped. It appears that in pursuing the new goal, it had to make large velocity adjustments, which resulted in increased uncertainty in its position knowledge. It then had to back away from the target for safety.

The largest contributor to fuel savings in docking with slowly tumbling targets was the widened deadband allowed in the new control system. The improvement was not large, but it was noticeable.

Kalman Filter Improvements

Kalman Filter Improvements

V. KALMAN FILTER IMPROVEMENTS

A. TWO SEPARATE FILTERS WERE USED

The original simulation program used a Kalman filter that estimated only position and velocity. The guidance algorithm was based on an assumption that the attitude measurements were accurate enough without filtering. The disadvantages of this scheme became apparent when significant target attitude rates were simulated: because the chase vehicle could not anticipate the motion of the docking fixture, it always headed toward the instantaneous docking-port position, not toward where the port would be at the time of arrival. When the attitude rate exceeded 1000 to 2000 deg/h about either the pitch or yaw axis, the guidance algorithm was not able to cope. The chase vehicle either circled the target indefinitely or crashed into the target. Furthermore, the chase vehicle's attitude control algorithm used the video imagery for guidance, attempting to keep the light pattern in the middle of the field of view. This strategy caused problems when the target was tumbling, because centering the lights in the field of view did not guarantee that the docking fixtures would be aligned.

The modified guidance system solves both of these problems, and the key to the solution was an expansion of the Kalman filter to include attitude and attitude rate in addition to translational quantities.

The first design decision was whether to add state elements to the existing Kalman filter or to split the problem into two independent subproblems, chase vehicle position, and target attitude. We considered the increase in burden on the flight computer, the likelihood of filter stability problems, the degree of coupling between position and attitude measurements, and the absence of coupling between position dynamics and attitude dynamics.

We concluded that if we could reduce the measurement coupling, two filters would give essentially the same accuracy as a single larger filter, but with less likelihood of instability problems and with a significant reduction in the burden on the flight computer.

Therefore, only slight modifications were made to the original position filter, and an independent target-attitude filter was added.

B. NEW FILTER ESTIMATES TARGET ATTITUDE AND ANGULAR MOMENTUM

For state variables, the added filter uses a quaternion and an angular momentum vector. Attitude is expressed with respect to the nonrotating primary reference frame used for chase vehicle guidance. The angular momentum vector is also expressed in this frame.

1. Selection of State Variables

We selected the quaternion parameterization of attitude to minimize the computational burden. The other parameterizations we considered were:

1) A set of three angles, e.g., yaw, pitch, and roll;

2) The Gibbs vector parameterization;

3) A direction cosine matrix;

4) Euler axis and angle;

5) The first three elements of a quaternion.

The three-angle parameterization was rejected because it requires complex formulas for use. For example, the simplest way to propagate the state estimate is to convert to one of the other parameterizations. Furthermore, the approach requires added logic to handle exceptions at singularities.

We rejected the Gibbs vector approach for the same reasons: the complexity of the formulas and the presence of a singularity that requires special handling.

Direction cosine matrices have three disadvantages. First, they have nine elements to compute, six of which are redundant. Second, the propagation formula requires approximately 30% more arithmetic than the formula for quaternions. Third, they require much more arithmetic than quaternions do for incorporating a new measurement.

The remaining two options were rejected because the simplest way to use them is by converting to quaternions for computations and then converting back to the original form. For example, the first three (or any three) elements of a quaternion can be used to express attitude in the theoretical minimum number of elements as long as the sign of the fourth element is known. This is true because the sum of the squares of the elements always equals 1.0. Because multiplying all four elements of a quaternion by -1.0 does not change the attitude expressed, it is always possible to manipulate the quaternion so that the last element is positive. When this is done, the last element is completely redundant and can be dropped. However, there is little to be gained by dropping an element and much to lose. The simplest way to use the three remaining elements is to recreate the fourth element. Furthermore, when this element is small, roundoff errors will prevent accurate reconstruction.

In summary, the quaternion representation appeared to be best for this application. Therefore, the first four state variables are the four elements of the quaternion that represents the target's attitude with respect to the primary reference frame.

The next three elements were to be some measure of attitude rate, which can also be parameterized in different ways. We considered:

1) The angular velocity vector in the current target reference frame;

2) The angular velocity vector in the primary frame;

3) The angular momentum vector in the current target frame;

4) The angular momentum vector in the primary frame.

We selected the angular momentum vector, expressed in the primary frame, because this vector does not change with time. This fact simplified state estimate propagation and anticipation of target attitude changes in the guidance strategy algorithm. In addition, this parameterization made it easier to analyze the filter's accuracy, stability, and rate of convergence while we were running simulations.

The remaining three elements of the state vector, then, are the x, y, and z components of the target's angular momentum vector, expressed in the primary reference frame. This makes a total of seven state variable elements. In the simulation program, they are the seven elements of the array ESTA.

2. State Estimate Propagation

Between observations, the filter propagates the state estimate covariance by linearizing about the current estimate. To do this, it computes a matrix of partial derivatives, F:

[1]

$$F = \begin{bmatrix} \frac{\partial \dot{x}_1}{\partial x_1} & \frac{\partial \dot{x}_1}{\partial x_2} & \frac{\partial \dot{x}_1}{\partial x_3} & \frac{\partial \dot{x}_1}{\partial x_4} & \frac{\partial \dot{x}_1}{\partial x_5} & \frac{\partial \dot{x}_1}{\partial x_6} & \frac{\partial \dot{x}_1}{\partial x_7} \\ \frac{\partial \dot{x}_2}{\partial x_1} & \frac{\partial \dot{x}_2}{\partial x_2} & \frac{\partial \dot{x}_2}{\partial x_3} & \frac{\partial \dot{x}_2}{\partial x_4} & \frac{\partial \dot{x}_2}{\partial x_5} & \frac{\partial \dot{x}_2}{\partial x_6} & \frac{\partial \dot{x}_2}{\partial x_7} \\ \frac{\partial \dot{x}_3}{\partial x_1} & \frac{\partial \dot{x}_3}{\partial x_2} & \frac{\partial \dot{x}_3}{\partial x_3} & \frac{\partial \dot{x}_3}{\partial x_4} & \frac{\partial \dot{x}_3}{\partial x_5} & \frac{\partial \dot{x}_3}{\partial x_6} & \frac{\partial \dot{x}_3}{\partial x_7} \\ & & & \vdots & & & \\ \frac{\partial \dot{x}_7}{\partial x_1} & \frac{\partial \dot{x}_7}{\partial x_2} & \frac{\partial \dot{x}_7}{\partial x_3} & \frac{\partial \dot{x}_7}{\partial x_4} & \frac{\partial \dot{x}_7}{\partial x_5} & \frac{\partial \dot{x}_7}{\partial x_6} & \frac{\partial \dot{x}_7}{\partial x_7} \end{bmatrix}$$

ORIGINAL PAGE IS
OF POOR QUALITY

where $\underline{x}$ and $\underline{\hat{x}}$ are the state vector and its rate of change. F is evaluated by assuming $\underline{x}$ equals $\underline{\hat{x}}$, the current state estimate.

From F it computes a state transition matrix, ignoring changes in F over the integration step:

[2] $$\Phi \approx 1 + \Delta t F + \tfrac{1}{2}(\Delta t)^2 F^2$$

where

Δt, represented in the simulation program as STEP, is the integration step;

Φ, represented in the simulation program as PHI, is the state transition matrix.

It can compute the state estimate at the end of the integration step by second-order Runge-Kutta integration:

[3] $$\underline{k}_1 = \left(\underline{\hat{q}} \odot \begin{bmatrix} \tfrac{1}{2} I^{-1} A_t(\underline{\hat{q}})\underline{\hat{L}} \\ \hline 0 \end{bmatrix}\right) \Delta t$$

[4] $$\underline{q}_t = (\underline{q} + \underline{k}_1)/(|\underline{q} + \underline{k}_1|)$$

[5] $$\underline{k}_2 = \underline{q}_t \odot \begin{bmatrix} \tfrac{1}{2} I^{-1} A_t(\underline{q}_t)\underline{\hat{L}}\Delta t \\ \hline 0 \end{bmatrix}$$

[6] $$\underline{\hat{x}} \leftarrow \begin{bmatrix} \underline{\hat{q}} + \tfrac{1}{2}(\underline{k}_1 + \underline{k}_2)/(|\underline{\hat{q}} + \tfrac{1}{2}(\underline{k}_1 + \underline{k}_2)|) \\ \hline \underline{\hat{L}} \end{bmatrix}$$

where

$\hat{\underline{q}}$, represented in the program as the first four elements of ESTA, is the quaternion portion of the state estimate vector;

$\hat{\underline{L}}$, represented in the program as the last three elements of ESTA, is the estimated target angular momentum;

$\hat{\underline{x}}$, represented in the program as ESTA, is the full state estimate;

I^{-1}, represented in the program as ININV, is the inverse of the target moment-of-inertia tensor, which is assumed known;

Δt is, again, the integration time step;

$\underline{k}_1$, $\underline{q}_t$, and $\underline{k}_2$, represented in the program as K1, QI, and K2, are quaternion-valued intermediate results;

$A_t(\underline{q})$ is the direction cosine matrix corresponding to quaternion $\underline{q}$ (for any $\underline{q}$);

The symbol $\odot$ denotes quaternion multiplication.

The estimate's covariance is computed from

[7] $$P \leftarrow \Phi P \Phi^T + Q$$

where

P, represented in the program as PA, is the state estimate's covariance;

Φ is the state transition matrix Equation [2];

Q, represented in the program as Q, is an empirical, constant, positive diagonal matrix that represents state noise, i.e., the uncertainty introduced by simplifying assumptions in the dynamics model, roundoff and other errors in numerical integration, and unmodeled torques.

ORIGINAL PAGE IS
OF POOR QUALITY

To compute F, the program first computes the intermediate results

[8] $J = I^{-1}A_t(\hat{q})$

where

I^{-1}, represented in the program as ININV, is the inverse of the target moment of inertia tensor;

$A_t(\hat{q})$, represented in the program as AT, is the direction cosine matrix that corresponds to the quaternion portion of the state estimate ESTA.

The program then computes

[9]
$$C = \begin{bmatrix} q_4 & -q_3 & q_2 \\ q_3 & q_4 & -q_1 \\ -q_2 & q_1 & q_4 \\ -q_1 & -q_2 & -q_3 \end{bmatrix}$$

[10]
$$D_1 = \begin{bmatrix} q_1 & q_2 & q_3 \\ q_2 & -q_1 & q_4 \\ q_3 & -q_4 & -q_1 \end{bmatrix}$$

[11]
$$D_2 = \begin{bmatrix} -q_2 & q_1 & -q_4 \\ q_1 & q_2 & q_3 \\ q_4 & q_3 & -q_2 \end{bmatrix}$$

[12]

$$D_3 = \begin{bmatrix} -q_3 & q_4 & q_1 \\ -q_4 & -q_3 & q_2 \\ q_1 & q_2 & q_3 \end{bmatrix}$$

ORIGINAL PAGE IS
OF POOR QUALITY

[13]

$$D_4 = \begin{bmatrix} q_4 & q_3 & -q_2 \\ -q_3 & q_4 & q_1 \\ q_2 & -q_1 & q_4 \end{bmatrix}$$

[14] $B = CI^{-1}$

[15] $\underline{\omega} = J\hat{\underline{L}}$

[16]

$$\Omega = \begin{bmatrix} 0 & \omega_3 & -\omega_2 & \omega_1 \\ -\omega_3 & 0 & \omega_1 & \omega_2 \\ \omega_2 & -\omega_1 & 0 & \omega_3 \\ -\omega_1 & -\omega_2 & -\omega_3 & 0 \end{bmatrix}$$

[17] $\underline{W}_i = BD_i\hat{\underline{L}}$ for $i = 1, 2, 3, 4$

where

$\underline{\omega}$, represented in the program as twice the variable HAV (to reduce computation), is the estimated angular velocity of the target;

$\hat{\underline{L}}$, represented in the program as the last three elements of ESTA, is the angular momentum portion of the state estimate ESTA;

$\underline{q}_i$ are elements of $\underline{q}$;

Other quantities are intermediate results or are as in previous equations.

The F matrix is assembled from these intermediate results:

[18]
$$F = \left[\begin{array}{c|c|c|c} [\underline{W}_1 \mid \underline{W}_2 \mid \underline{W}_3 \mid \underline{W}_4] + \tfrac{1}{2}\Omega & \tfrac{1}{2}\underline{q} \odot \begin{bmatrix} \underline{j}_1 \\ \hline 0 \end{bmatrix} & \tfrac{1}{2}\underline{q} \odot \begin{bmatrix} \underline{j}_2 \\ \hline 0 \end{bmatrix} & \tfrac{1}{2}\underline{q} \odot \begin{bmatrix} \underline{j}_3 \\ \hline 0 \end{bmatrix} \\ \hline 0_{3x4} & 0_{3x1} & 0_{3x1} & 0_{3x1} \end{array}\right]$$

where $\underline{j}_i$ represents column i of the matrix J, and the other symbols are as previously defined.

These calculations are done in subroutine PRPESA.

3. Filter Update

Each time the guidance system receives a new image interpretation, it updates the state estimate. The formulas used are based on the normal extended Kalman filter equations, except for one small change: normally linearization would be about the current estimated state. In this filter, however, a coordinate transformation is done first: the state estimate and covariance matrix are transformed into the (currently estimated) target body coordinate system. This approach was adopted in an attempt to minimize the effects of nonlinearities. After the state estimate and covariance matrix are updated, they are converted back to the primary coordinate system.

The formulas used are best presented procedurally:

First compute R, an empirical positive diagonal matrix that represents the measurement noise covariance. This calculation, done in subroutine ATMCOV, is based on a formula derived from fitting a curve to experimental data:

[19] $R = \text{diag}\ [v\ \ v\ \ v\ \ v/100]$

ORIGINAL PAGE IS
OF POOR QUALITY

where

[20] $v = 1.129 \times 10^{-9}|\underline{\ell}| + 0.0001$

Second, compute

[21]
$$P^* = \left[\begin{array}{c|c} T^T P_{11} T & T^T P_{12} \\ \hline (T^T P_{12})^T & P_{22} \end{array}\right]$$

where

T, represented in the program as T, is a transformation matrix formed from the elements of the quaternion portion of the state estimate ESTA:

[22]
$$T = \begin{bmatrix} q_4 & -q_3 & q_2 & q_1 \\ q_3 & q_4 & -q_1 & q_2 \\ -q_2 & q_1 & q_4 & q_3 \\ -q_1 & -q_2 & -q_3 & q_4 \end{bmatrix}$$

P*, represented in the program as PA, is the transformed covariance matrix;

P_{ij} are submatrices of P, which is partitioned between the fourth and fifth rows and between the fourth and fifth columns.

The program uses the array PA for both P and P* to save space, because P and P* are never needed simultaneously and because a portion of P does not change in the transformation to P*.

Third, calculate the Kalman gain matrix K, represented in the program as K:

[23] $$K = P^* G^T (R + G P^* G^T)^{-1}$$

In implementing this equation in the program, it was not necessary to explicitly multiply by the sensitivity matrix G, because it has the value

[24] $$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

and serves merely to select elements of P*.

The state estimate and covariance are now updated with the formulas:

[25] $$\underline{q}' = \begin{bmatrix} x_4 & x_3 & -x_2 & -x_1 \\ -x_3 & x_4 & x_1 & -x_2 \\ x_2 & -x_1 & x_4 & -x_3 \\ x_1 & x_2 & x_3 & x_4 \end{bmatrix} \underline{q}_{meas}$$

[26] $$\underline{x}^* \leftarrow \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} + K \, \mathrm{sign}(q_4') q'$$

ORIGINAL PAGE IS
OF POOR QUALITY

[27] $P^* \leftarrow (1 - KG)P^*$

where, again, multiplication by G is done implicitly, and $\underline{q}_{meas}$ is the measured quaternion representing target attitude.

Finally, the state estimate and covariance are transformed back to the primary coordinate system:

[28]

$$\hat{\underline{x}} \leftarrow \begin{bmatrix} T \begin{bmatrix} x_1^* \\ x_2^* \\ x_3^* \\ x_4^* \end{bmatrix} \\ \hline x_5^* \\ x_6^* \\ x_7^* \end{bmatrix}$$

[29]

$$P \leftarrow \left[\begin{array}{c|c} TP^*_{11}T^T & TP^*_{12} \\ \hline (TP^*_{12})^T & P^*_{22} \end{array}\right]$$

where

x_i and x^*_i are elements of $\hat{\underline{x}}$ and $\hat{\underline{x}}^*$;

P_{ij} and P^*_{ij} are submatrices of P and P*, which are partitioned as above.

These calculations are done in subroutine INCRPA.

C. POSITION FILTER CHANGED LITTLE

Two changes were made to the Kalman filter that estimates translational position and velocity:

1) The calculation for the measurement covariance matrix was revised to reflect the better measurements provided by a better camera and the improved image interpretation algorithm described in Section V. The new formulas also acknowledge that measurement errors in x, y, and z directions are correlated and unequal.

2) The covariance propagation formulas were modified to reflect less uncertainty in thruster forces. Like the measurement formulas, these formulas now acknowledge that uncertainties are not equal in each direction and that they are correlated.

1. Measurement Covariance

Subroutine ESTCOV estimates the measurement error covariance from two empirical equations, derived by fitting curves to experimental data:

[30] $$V_1 = 8 \times 10^{-7}(r - 5)^4 + 0.005$$

for errors along the chase vehicle x axis (camera boresight); and

[31] $$V_2 = 7.36 \times 10^{-7}(r^3) + 0.016$$

for errors along either the y or the z axis.

In these formulas r is the estimated range to the target.

The values computed from Equations [30] and [31] form the diagonal elements of the covariance matrix:

[32] $$R = \begin{bmatrix} V_1 & 0 & 0 \\ 0 & V_2 & 0 \\ 0 & 0 & V_2 \end{bmatrix}$$

which is expressed in vehicle coordinates. To be useful in the filter, the matrix must be converted to the primary reference frame:

[33] $$R \longleftarrow A_c^T R A_c$$

where A_c is the direction cosine matrix defining the chase vehicle's attitude with respect to the primary frame. A_c is supplied by the inertial measurement unit.

2. Covariance Propagation

Between measurements the state estimate's accuracy degrades, because:

1) Initial uncertainty in velocity leads to steadily increasing uncertainty in position;

2) If thrusters are used, the resulting acceleration cannot be known exactly; even accelerometer measurements will contain some error;

3) There will be differential gravitational accelerations between the chase vehicle and the target, due to gravity gradient, even when the thrusters are not used;

4) Oversimplifications in the dynamics model and numerical errors (roundoff, truncation, and approximations in formulas and values of variables) cause a steady growth in estimation errors.

Subroutine PROPES explicitly models thrust uncertainty and the effect of velo··y errors on future position errors. The remaining error sources are accounted for by adding a small, positive, constant, diagonal matrix to the covariance matrix during propagation.

D. NEWTON-RALPHSON ITERATION IMPROVES IMAGE INTERPRETATION

The original image-interpretation algorithm did not consider perspective effects, which become significant only at close range. During this contract phase, we added a subroutine (MPROVE) that accounts for perspective effects. This subroutine improves the interpretation accuracy and decreases correlation between errors in target attitude measurements and errors in chase vehicle position measurements.

The principle behind this routine is the Newton-Ralphson method for solving systems of nonlinear equations. This method starts with an initial guess, which is the interpretation provided by the original algorithm, and successively refines it.

The first three elements of the initial guess, $\underline{\hat{x}}$, are the x, y, and z components of the chase vehicle's position, expressed in the primary reference frame used for navigation. The remaining three elements are the first three elements of a quaternion that expresses the difference between the target spacecraft's attitude and some reference attitude. In the program, the reference attitude is taken to be the measured attitude, so this quaternion is the identity quaternion, and the first three elements are zero. The use of only three elements for the quaternions relies on the fact that a quaternion can be premultiplied by -1.0 if necessary to guarantee that its fourth element is positive. Therefore, it can be reconstructed from the other three elements with no ambiguity, because, by convention, all the quaternions have magnitudes of 1.0, and the sign of the missing element is now known.

The routine is given a measurement vector in which the first three elements are the horizontal image-plane coordinates of the three docking aid lamps. The remaining three vector elements are the vertical coordinates for these lamps.

If near-linear equations relate small changes in the viewing position and target attitude to changes in lamp image coordinates,

[34] $(\underline{v}_{meas} - \underline{v}_{pred}) = H(\underline{x} - \hat{\underline{x}})$

where

$\underline{v}_{meas}$ is the measurement vector described previously;

$\underline{x}$ is the true position/attitude vector (with the quaternion portion expressing the error in the quaternion portion of $\hat{\underline{x}}$);

$\underline{v}_{pred}$ is a measurement vector predicted from $\hat{\underline{x}}$;

$\hat{\underline{x}}$ is the initial guess at $\underline{x}$, derived from the original image-interpretation algorithm;

H is the matrix defining the near-linear relationship between changes in $\underline{x}$ and changes in $\underline{v}$.

Iterations based on this approximation converge to the true solution if the initial guess is close enough to the solution so that $(\underline{x} - \hat{\underline{x}})$ and $(\underline{v}_{meas} - \underline{v}_{pred})$ are small quantities. The matrix H is a matrix of partial derivatives:

[35]
$$H = \begin{bmatrix} \frac{\partial v_1}{\partial x_1} & \frac{\partial v_1}{\partial x_2} & \cdots & \frac{\partial v_1}{\partial x_6} \\ \frac{\partial v_2}{\partial x_1} & \frac{\partial v_2}{\partial x_2} & & \frac{\partial v_2}{\partial x_6} \\ \vdots & & \ddots & \vdots \\ \frac{\partial v_6}{\partial x_1} & \frac{\partial v_6}{\partial x_2} & \cdots & \frac{\partial v_6}{\partial x_6} \end{bmatrix}$$

The first three rows of H represent the sensitivity of the horizontal components of the three lamps' image-plane coordinates to changes in the position/attitude vector. Each of these rows can be computed from the same formula by changing the value of $\underline{h}_t$, the lamp's position in the target reference frame, to represent each lamp in turn. The expression for these rows is:

[36]
$$\begin{bmatrix} \frac{-r_2}{r_1} & 1 & 0 \end{bmatrix} A_c \left[\mathbf{1}_{3x3} \;\middle|\; 2A_t^T \begin{bmatrix} 0 & -h_{t_3} & h_{t_2} \\ h_{t_3} & 0 & -h_{t_1} \\ -h_{t_2} & h_{t_1} & 0 \end{bmatrix} \right] \frac{f}{r_1}$$

in which

$\underline{r}$ is the lamp's position in the camera's reference frame, computed from

[37]
$$\underline{r} = \left(A_c \left(A_t^T \underline{h}_t - \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \right) \right) - \underline{h}_c$$

$\underline{h}_t$ is the lamp's position in the target reference frame;

A_c and A_t are direction cosine matrices that specify chase-vehicle and target attitude with respect to the primary frame used for navigation;

$\underline{x}$ is the position/attitude vector described previously;

$\underline{h}_c$ is the camera's position in the chase-vehicle frame;

f is the lens focal length;

$\mathbf{1}_{3x3}$ is the 3x3 identity matrix.

For quantities that are unknown (A_t, $\underline{x}$), the measured value provided by the original image-interpretation algorithm is used. The state estimate from the Kalman filter could be used for this purpose, but use of the measurements allows the system to be self-starting.

Furthermore, it simplifies the repeated use of subroutine MPROVE for successively refining the interpretation. Each time the subroutine is reexecuted, it starts with the interpretation it produced the previous time. Because it reduces the interpretation error by approximately a factor of 10 each time it is executed, it can make the error insignificant in two executions. (The errors caused by camera noise cannot be removed by any interpretation scheme. They can only be averaged out by taking multiple measurements. This is the function of the Kalman filters, not of subroutine MPROVE.) The procedure calculates the last three rows of H from a formula almost identical to Equation [36]. These rows correspond to the sensitivity of the vertical components of the lamp-image coordinates to changes in $\underline{x}$. They are calculated from:

[38]

$$\begin{bmatrix} \frac{r_3}{r_1} & 0 & -1 \end{bmatrix} A_c \left[\mathbf{1}_{3x3} \;\middle|\; 2A_t^T \begin{bmatrix} 0 & -h_{t_3} & h_{t_2} \\ h_{t_3} & 0 & -h_{t_1} \\ -h_{t_2} & h_{t_1} & 0 \end{bmatrix} \right] \frac{f}{r_1}$$

Because a major portion of the calculation is the same for Equations [36] and [38], they are merged in the procedure to minimize the arithmetic. After calculating H, the procedure solves Equation [34] for ($\underline{x}$ - $\hat{\underline{x}}$), the difference between the refined estimate and the initial guess. This error is added to the initial guess. However, the quaternion portion of $\underline{x}$ expresses the error in the measured quaternion; it does not represent attitude with respect to the primary reference frame. To get the attitude with respect to this frame, the procedure multiplies the error quaternion by the measured quaternion.

Goal-Selection Strategy Changes

VI. GOAL-SELECTION STRATEGY CHANGES

A simple control system has no need for goal-selection logic. It simply tries to minimize the error between the commanded position and the current position. Its control law may include some form of anticipation of the future (e.g., a phase-lead network) or other compensation to stabilize the control loop or improve performance, but it does not have to think very much to perform satisfactorily.

Such a control system is not suitable for a video rendezvous guidance system that operates with tumbling target spacecraft. The guidance system must reduce the position error to zero, but it also must also do so without endangering itself or the target. This means that it must reason about avoiding a path that goes through the target. It must also avoid getting too close to the target when its knowledge of its position may be in error, and make allowance for the finite size of both the target and the chase vehicle so that the two spacecraft do not bump into each other at the side or rear end. Furthermore, it must try to keep the docking aid within the field of view of the television camera and, if possible, minimize fuel use.

The goal-selection logic implemented in the simulation program attempts to do all these things. To minimize the complexity of the task, we have divided the problem into two nearly independent subproblems, attitude and position control.

A. ATTITUDE GOAL SELECTION

Attitude control is by far the simpler of the two problem. The logic for attitude goal selection is in subroutine RPY, which replaces two subroutines (ESTRPY and RPY) of the original program. The original routines used the state estimate from the Kalman filter only when direct video imagery was unavailable. In contrast, the new routine always use the state estimate and never use the video image data directly. This is possible because the filter now provides target attitude information as well as position data.

The strategy of the subroutine is simple: it adjusts yaw and pitch angles to keep the chase vehicle's docking fixture pointed directly at the end of the target's docking fixture, and it adjusts the roll angle to align the camera with the docking aid lights.

It allows for target motion by predicting the target's attitude and the chase vehicle's position at then end of the sample interval, 1.2333 s into the future. This anticipation reduces errors by compensating for control system lag.

1. The Logic of RPY

Subroutine RPY first propagates the state estimates for position and target attitude 1.2333 s into the future so that all planning is based on where things will be at the end of the decision interval, not on where they are at the start of the interval. Because movement in this time interval will be small, the subroutine uses simple Euler numerical integration of the state estimates with the assumption that all thrusters are off.

It then computes the vector from the chase vehicle center of mass (at its assumed new position) to the tip of the target's docking fixture (at its assumed new position). This vector is expressed in the chase vehicle's coordinate system:

[39] $$\underline{p} = A_c(A_t^T \underline{h}_{dt} - \underline{\ell})$$

where

A_c and A_t are direction cosine matrices that describe the attitudes of the chase vehicle and the target;

$\underline{h}_{dt}$ is the position of the docking fixture tip in the target's body coordinate system;

$\underline{\hat{x}}$ is the estimated chase vehicle position.

The subroutine then estimates yaw and pitch errors from arc tangents of the ratios of the elements of $\underline{p}$:

[40] $$\text{(yaw error)} = \tan^{-1}\left(\frac{-p_2}{p_1}\right)$$

[41] $$\text{(pitch error)} = \tan^{-1}\left(\frac{p_3}{p_1}\right)$$

The roll error is found by calculating a unit vector $\underline{r}$ that is parallel to the target "-y" axis. This vector is expressed in the chase vehicle's body coordinate system. Then,

[42] $$\text{(roll error)} = \tan^{-1}\left(\frac{-r_3}{r_2}\right)$$

B. TRANSLATIONAL POSITION GOAL SELECTION

The logic of subroutine SETGOL selects a translational position goal. It starts by predicting the target's attitude, but it predicts farther into the future than the end of the decision interval, because the chase vehicle may take several minutes to reach the target. The number of seconds of anticipation is an empirically chosen function of range.

We found that bad performance resulted from too much anticipation: the chase vehicle did not match the docking fixture's tangential velocity component well. Too little anticipation was also bad: the chase vehicle lagged behind the docking fixture, resulting in poor alignment. The empirical formula that seemed to give best overall results allowed 0.2 s anticipation for each meter of range, with a maximum anticipation of 20 s.

Because nonlinearities could be significant in propagating attitude for 20 s, we used second-order Runge-Kutta numerical integration rather than the much simpler Euler integration. However, no perceptible performance improvement results from this, because significant errors occur only at a considerable distance from the target.

After predicting target attitude, the subroutine selects a goal on the chase vehicle's docking axis. The distance between the goal and the target is at least enough to accommodate the docking fixtures of the two spacecraft. Under certain circumstances, however, an additional safety margin is allowed.

First, at distances over 12 m from the target, the subroutine allows for a safety margin of twice the standard deviation associated with its state estimate, or approximately three times the probable error in its knowledge of its position. The accuracy information it needs to compute this margin is taken from the diagonal elements of the covariance matrix maintained by the Kalman filter.

Second, the subroutine allows additional margin for misalignment between the two spacecraft. For example, if the chase vehicle is in the right position but the wrong attitude, it might damage the target with its solar panels. The allowance for misalignment varies from zero to 19.5 m, depending on the amount of misalignment.

Third, subroutine MODGOL analyzes the path from the current position to the goal. If it finds that the goal is on the far side of the target and that the path passes too close to the target, it revises the goal. The new goal it selects is on the shortest circular path around the target at a safe distance.

Appendix

APPENDIX A--PROGRAM LISTING

The program listing in this appendix is provided to document the simulation methods used in analyzing the three-light video guidance system and running the simulation. It was written to run on a Prime 550 computer under the PRIMOS operating system, but it has few hardware-dependent subroutines. If it is to be run on another computer, the following information will prove useful.

Several library routines are used, and these are not shown in the listing. The routines include ASIN and ACOS, which compute the inverse trigonometric functions arc sine and arc cosine. The function RANFN is a random number generator that computes normally distributed random values with a specified mean and standard deviation. In addition, the matrix arithmetic routines MADD (addition), MSUB (subtraction), MMLT (multiplication), MINV (matrix inversion), MSCL (multiplication by a scalar), MIDN (setting an array equal to the identity matrix), and MTRN (forming the transpose of a matrix) are used from the Prime library MATHLB.

File handling may present conversion problems even if the program is to be run on another Prime 550 computer, because logical unit numbers, file names, and amount of disk storage vary from installation to installation. Standard Prime subroutines are used to open and close files. These subroutines (TSRC$$, EXST$A,CLOS$A, and DELE$A) are from the Prime library APPLIB.

Run time is approximately twice real time if the computer is dedicated to one user.

The perspective drawings shown in this report are not created directly by this program. They are drawn by a second program that uses the data file created by this program. This allows the creation of stereo plots and views from different perspectives.

Several WRITE statements in subroutine DOCK are rendered inactive by a character C in the first column of text. Removing this character will provide a printout at the operator's terminal for monitoring the progress of the simulation;

The first part of the listing is the text of a terminal session, which includes compilation, loading, and execution of the program.

```
ENTER PATHNAME FOR OUTPUT
DOKO
***FILE ALREADY EXISTS   OK TO OVERWRITE? YES

 ENTER THREE ROTATION ANGLES TO SPECIFY INITIAL TARGET
 ATTITUDE FOR A YAW-PITCH-ROLL SEQUENCE---ANGLES MUST BE
 IN DEGREES
0
0
0

 ENTER TUMBLE AXIS-- ENTER 1, 2, OR 3 FOR YAW, PITCH,
 OR ROLL, RESPECTIVELY
1

 ENTER TUMBLE RATE IN DEGREES PER HOUR
500
 INPUT 16-DIGIT RANDOM INTEGER
5296394709603709 54534
DO YOU WANT ANOTHER SIMULATION? NO
```

```
OK, CO C PHASE3A
OK, FTN MSFC PHASE3A -DCLVAR
0000 ERRORS [< MAIN  >FTN-REV17 4]
0000 ERRORS [<OPEN   >FTN-REV17 4]
0000 ERRORS [<INIPAR>FTN-REV17 4]
0000 ERRORS [<INISIM>FTN-REV17 4]
0000 ERRORS [<DOCK   >FTN-REV17 4]
0000 ERRORS [<FLASH  >FTN-REV17 4]
0000 ERRORS [<PROPTR>FTN-REV17 4]
0000 ERRORS [<COMPK1>FTN-REV17 4]
0000 ERRORS [<COMPK2>FTN-REV17 4]
0000 ERRORS [<COMPK3>FTN-REV17 4]
0000 ERRORS [<COMPK4>FTN-REV17 4]
0000 ERRORS [<POINT  >FTN-REV17 4]
0000 ERRORS [<POSIT  >FTN-REV17 4]
0000 ERRORS [<ATITUD>FTN-REV17 4]
0000 ERRORS [<FINDCV>FTN-REV17 4]
0000 ERRORS [<QUATRN>FTN-REV17 4]
0000 ERRORS [<INCORP>FTN-REV17 4]
0000 ERRORS [<COMPG  >FTN-REV17 4]
0000 ERRORS [<ESTCOV>FTN-REV17 4]
0000 ERRORS [<KALGAN>FTN-REV17 4]
0000 ERRORS [<UPDSTA>FTN-REV17 4]
0000 ERRORS [<UPDCOV>FTN-REV17 4]
0000 ERRORS [<PROPES>FTN-REV17 4]
0000 ERRORS [<SETGOL>FTN-REV17 4]
0000 ERRORS [<MODGOL>FTN-REV17 4]
0000 ERRORS [<THRUST>FTN-REV17 4]
0000 ERRORS [<CNTLAW>FTN-REV17 4]
0000 ERRORS [<ACCEL  >FTN-REV17 4]
0000 ERRORS [<SELECT>FTN-REV17 4]
0000 ERRORS [<TABLE1>FTN-REV17 4]
0000 ERRORS [<TABLE2>FTN-REV17 4]
0000 ERRORS [<TABLE3>FTN-REV17 4]
0000 ERRORS [<FIRTHR>FTN-REV17 4]
0000 ERRORS [<IMU    >FTN-REV17 4]
0000 ERRORS [<RPY    >FTN-REV17 4]
0000 ERRORS [<MPROVE>FTN-REV17 4]
0000 ERRORS [<MINV6  >FTN-REV17 4]
0000 ERRORS [<TOQUAT>FTN-REV17 4]
0000 ERRORS [<INCRPA>FTN-REV17 4]
0000 ERRORS [<XFORMP>FTN-REV17 4]
0000 ERRORS [<COMPT  >FTN-REV17 4]
0000 ERRORS [<ATMCOV>FTN-REV17 4]
0000 ERRORS [<PRPESA>FTN-REV17 4]
0000 ERRORS [<CMPCD  >FTN-REV17 4]
0000 ERRORS [<DOKCHK>FTN-REV17 4]
0000 ERRORS [<DIRMAT>FTN-REV17 4]
0000 ERRORS [<STPRIM>FTN-REV17 4]
0000 ERRORS [<FORCE  >FTN-REV17 4]
0000 ERRORS [<MPRIME>FTN-REV17 4]
0000 ERRORS [<LINACL>FTN-REV17 4]
0000 ERRORS [<TORQUE>FTN-REV17 4]
0000 ERRORS [<LPRIME>FTN-REV17 4]
0000 ERRORS [<MAKROT>FTN-REV17 4]
0000 ERRORS [<QPRIME>FTN-REV17 4]
0000 ERRORS [<SQR1   >FTN-REV17 4]
0000 ERRORS [<ANGVEC>FTN-REV17 4]
0000 ERRORS [<TRGATT>FTN-REV17 4]
0000 ERRORS [<QMLT   >FTN-REV17 4]
0000 ERRORS [<DPRD   >FTN-REV17 4]

OK, LOAD
$ LO B_MSFC PHASE3A
$ LO 'TIETZ>WORKING NERD>B_RANF
$ LO 'TIETZ>WORKING NERD>B_ASIN
$ LI APPLIB
$ LI MATHLB
$ LI
LOAD COMPLETE
$ SA *PHASE3A
$ QU

OK, CO -TTY
OK, R *PHASE3A
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
TRUE STATE

   STATE( 1)  X POSITION
   STATE( 2)  Y POSITION
   STATE( 3)  Z POSITION
   STATE( 4)  X VELOCITY
   STATE( 5)  Y VELOCITY
   STATE( 6)  Z VELOCITY
   STATE( 7)  ANGULAR MOMENTUM ABOUT X
   STATE( 8)  ANGULAR MOMENTUM ABOUT Y
   STATE( 9)  ANGULAR MOMENTUM ABOUT Z
   STATE(10) Q1 \
   STATE(11) Q2  \
   STATE(12) Q3   > ATTITUDE W R T  'TRUTH' AXES
   STATE(13) Q4  /
   STATE(14)  MASS, INCL FUEL

     (X,Y,Z) = 'TRUTH' AXES

 STATE ESTIMATE

   ESTATE(1)  X' POSITION
   ESTATE(2)  Y' POSITION
   ESTATE(3)  Z' POSITION
   ESTATE(4)  X' VELOCITY
   ESTATE(5)  Y' VELOCITY
   ESTATE(6)  Z' VELOCITY

     (X',Y',Z') = 'PRIMARY REF  FRAME' AXES

 ATTITUDE STATE ESTIMATE

   ESTA(1)-->ESTA(4) = TARGET ATTITUDE ESTIMATE (QUATERNION
   GIVING ATTITUDE WITH RESPECT TO 'PRIMARY' REFERENCE FRAME)
   ESTA(5)-->ESTA(7) = ESTIMATED TARGET ANGULAR MOMENTUM IN
   'PRIMARY' REFERENCE FRAME
```

ROUTINE	PAGE	ROUTINE-NBR	
<MAIN>	1	000	
ACCEL	42	481	1
ANGVEC	79	904	
ATITUD	25	440	
ATMCOV	63	4D3	
CMPCD	66	4E1	
CNTLAW	41	481	
COMPC	30	451	
COMPK1	18	421	
COMPK2	19	422	
COMPK3	20	423	
COMPK4	21	424	
COMPT	62	4E1	
DIRMAT	69	901	
DOCK	11	400	
DOKCHK	68	4F0	
DPRD	83	907	
ESTCOV	31	452	
FINDCV	26	441	
FIRTHR	51	483	
FLASH	15	410	
FORCE	71	902	1
IMU	52	490	
INCORP	29	450	
INIPAR	5	200	
INISIM	10	300	
KALGAN	32	453	
LINACL	73	902	3
LPRIME	75	902	5
MAKROT	76	902	6
MINV6	57	4C1	
MODGOL	39	471	
MPROVE	54	4C0	
MPRIME	72	902	2
OPEN	4	100	
POINT	22	425	
POSIT	32	430	
PROPES	35	460	
PROPTR	17	420	
PRPESA	64	4E0	
QMLT	82	906	
QPRIME	77	902	7
QUATRN	27	442	
RPY	53	4A0	
SELECT	44	482	
SETGOL	37	470	
SQRI	78	903	
STPRIM	70	902	
TABLE1	45	482	1
TABLE2	47	482	2
TABLE3	49	482	3
THRUST	40	480	
TOQUAT	58	4C2	
TORQUE	74	902	4
TRGATT	80	905	
UPDCOV	34	455	
UPDSTA	33	454	
XFORMP	61	4D1	

```
C
C      PROGRAM MSFC PHASE3 - DOCKING SIMULATION PROGRAM WITH ENHANCED KALMAN
C         FILTER AND IMPROVED GUIDANCE STRATEGY
C
C      CALLS
C         CLOS$A, DOCK, EXIT, INIPAR, INISIM, OPEN, YSNO$A
C
C
       LOGICAL CLOS$A
C         LIBRARY FUNCTION TO CLOSE FILES (FUNCTION VALUE INDICATES SUCCESS)
       REAL ESTATE(6), ESTA(7)
C         ESTIMATE OF STATE VECTOR (FOR TRANSLATION, TARGET ATTITUDE)
       REAL F(14)
C         FORCES FROM THRUSTERS AFTER SELECTION
       INTEGER OUT
C         FORTRAN UNIT NUMBER OF OUTPUT FILE
       REAL P(6,6), PA(7,7)
C         COVARIANCE OF STATE ESTIMATE CORRESPONDING TO ESTATE() & ESTA()
       REAL STATE(14)
C         CHASE VEHICLE STATE VECTOR
       REAL T
C         ELAPSED TIME
       INTEGER TERMNL
C         FORTRAN UNIT NUMBER FOR TERMINAL
       LOGICAL DOCKED
C         CODE SET TO ' TRUE ' IF THE CHASE VEHICLE IS WITHIN DOCKING RADIUS
       LOGICAL FILERR
C         ERROR CODE RETURNED FROM 'OPEN' SUBROUTINE
       LOGICAL YSNO$A
C         FUNCTION TO GET YES/NO ANSWER FROM USER AT TERMINAL
C
       COMMON/SIMUL/TERMNL, OUT
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
       OUT=6
       TERMNL=1
C      (* LOOP UNTIL USER WANTS NO MORE SIMULATIONS *)
   10  CONTINUE
C         (* 100 - OPEN DATA FILE *)
             CALL OPEN(FILERR, TERMNL, OUT)
C         (* 200 - INITIALIZE PROBLEM PARAMETERS *)
             CALL INIPAR(T, STATE, P, ESTATE, F, ESTA, PA)
C         (* 300 - INITIALIZE SIMULATOR *)
             CALL INISIM(STATE)
          DOCKED= FALSE
C         (* LOOP WHILE NOT DOCKED *)
   20     IF (DOCKED) GO TO 30
C            (* 400 - SIMULATE OBSERVATION & INTERVAL BETWEEN OBSERVATIONS *
                CALL DOCK(P, ESTATE, STATE, T, DOCKED, F, ESTA, PA)
             GO TO 20
   30     CONTINUE
C         (* CLOSE DATA FILE *)
             IF( NOT CLOS$A(OUT-4)) GO TO 40
   40     CONTINUE
C         (* ASK OPERATOR WHETHER HE WANTS ANOTHER SIMULATION *)
             IF(YSNO$A('DO YOU WANT ANOTHER SIMULATION',30,-1)) GO TO 10
       CALL EXIT
C
       END
```

A-5

```
C
       SUBROUTINE OPEN(FILERR, TERMNL, OUT)
C      (* 100 - OPEN FILES FOR OUTPUT  WARNING  HIGHLY INSTALLATION DEPENDENT *)
C
C      CALLS
C         DELE$A, EXST$A, TSRC$$, YSNO$A
C      CALLED BY
C         <MAIN>
C
C      INPUT
C         IN, OUT
C      OUTPUT
C         FILERR
C
       INTEGER CHRPOS(2)
C         CHARACTER POSITION/COUNT CODES FOR SYSTEM ROUTINE TSRC$$
       INTEGER CODE
C         ERROR CODE RETURNED BY TSRC$$ (=0 IF NO ERROR)
       INTEGER I
C         DO LOOP INDEX
       INTEGER OUT, TERMNL
C         FORTRAN LOGICAL UNIT NUMBER FOR OUTPUT
       INTEGER PATH(16)
C         ARRAY OF ASCII DEFINING PATH NAME
       INTEGER TERMNL
C         FORTRAN LOGICAL UNIT NUMBER OF USER TERMINAL
       INTEGER TYPE
C         TYPE CODE RETURNED BY TSRC$$ (UNUSED HERE)
       LOGICAL DELE$A
C         SYSTEM ROUTINE TO DELETE A FILE (RETURNS TRUE IF SUCCESSFUL)
       LOGICAL EXST$A
C         SYSTEM ROUTINE TO CHECK EXISTENCE OF FILE (TRUE IF IT EXISTS)
       LOGICAL FILERR
C         ERROR CODE RETURNED TO CALLING ROUTINE
       LOGICAL YSNO$A
C         FUNCTION TO GET YES/NO ANSWER FROM USER AT TERMINAL
C
$INSERT SYSCOM>KEYS F
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
       CHRPOS(2)=32
       TYPE=0
   10  CONTINUE
             WRITE(TERMNL, 902)
             READ(TERMNL, 901)(PATH(I), I=1,16)
          IF( NOT EXST$A(PATH, 32)) GO TO 20
             IF( NOT YSNO$A('***FILE ALREADY EXISTS   OK TO OVERWRITE',
     A          40,-1)) GO TO 10
       IF( NOT DELE$A(PATH, 32)) GO TO 20
   20  CONTINUE
       CHRPOS(1)=0
       CALL TSRC$$(K$WRIT+K$NSAM, PATH, OUT-4, CHRPOS, TYPE, CODE)
       IF(CODE NE 0) GO TO 30
       FILERR= FALSE
       RETURN
C
   30  FILERR= TRUE
       RETURN
C
  901  FORMAT(16A2)
  902  FORMAT('ENTER PATHNAME FOR OUTPUT     ')
C
       END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
      SUBROUTINE INIPAR(T, STATE, P, ESTATE, F, ESTA, PA)
C     (* 200 - INITIALIZE PROBLEM PARAMETERS *)
C
C     CALLS
C        SQRT, RANFN, COS, SIN, MIDN
C     CALLED BY
C        <MAIN>
C
C     INPUT
C        <NONE>
C     OUTPUT
C        AK1, AK2, AK3, DOKRAD, ESTATE, FOCLEN, FULDIS, F1, INERCV, MEMPTY, P, TPHIH,
C        TRNATO, STATE, TERMNL, T, F, HC, HT, A, B
C
      REAL A, B
C        LAMP SEPARATION ON DOCKING AID (A=BASE TO EITHER SIDE LAMP, B=BASE
C        TO LAMP ON DOCKING AXIS) - VALUES IN METERS
      REAL AK1(14), AK2(3,14), AK3(3,14)
C        MATRICES SPECIFYING CHASE VEHICLE
C           AK1 GIVES FUEL USE PER NEWTON-SECOND OF THRUST FOR EACH THRUSTER
C           AK2 GIVES FORCE DIRECTION FOR EACH THRUSTER
C           AK3 GIVES TORQUE/NEWTON & TORQUE DIRECTION FOR EACH THRUSTER
      REAL DOKRAD
C        RADIUS FROM TARGET SPACECRAFT WITHIN WHICH THE SPACECRAFT ARE
C        CONSIDERED DOCKED
      REAL ESTATE(6), ESTA(7)
C        ESTIMATE OF STATE VECTORS FOR CHASE VEHICLE TRANSLATION AND
C        TARGET ROTATION, RESPECTIVELY
      REAL FOCLEN
C        LENS FOCAL LENGTH IN METERS
      REAL FULDIS(3,3)
C        CHASE VEHICLE FUEL-DISTRIBUTION INERTIA TENSOR
      REAL F(14)
C        FORCES FROM THRUSTERS AFTER SELECTION
      REAL F1(14)
C        LIST OF MAXIMUM THRUST MAGNITUDES FROM EACH THRUSTER OF CHASE VEHICLE
      REAL HC(3), HT(3)
C        CAMERA POSITION IN CHASE VEHICLE BODY FRAME AND DOCKING AID POSITION
C        IN TARGET SPACECRAFT BODY FRAME, RESPECTIVELY
      REAL HDC(3), HDT(3)
C        DOCKING FIXTURE' POSITIONS IN THEIR RESPECTIVE SPACECRAFT COORDI-
C        NATE SYSTEMS
      INTEGER I, J
C        DO LOOP INDICES
      REAL INERCV(3,3)
C        INERTIA OF EMPTY CHASE VEHICLE
      REAL ININV(3,3)
C        INVERSE OF TARGET SPACECRAFT MOMENT OF INERTIA
      INTEGER ITUMBL
C        TUMBLE AXIS, VALUE OF 1, 2, OR 3 INDICATES YAW, PITCH, OR ROLL
C        TUMBLE, RESPECTIVELY
      REAL MEMPTY
C        MASS OF CHASE VEHICLE WITHOUT FUEL
      INTEGER OUT
C        FORTRAN UNIT NUMBER FOR OUTPUT FILE
      REAL P(6,6), PA(7,7)
C        COVARIANCE OF STATE ESTIMATES ESTATE() AND ESTA(), RESPECTIVELY
      REAL PHI, PSI, THETA
C        ANGLES SPECIFYING INITIAL TARGET ATTITUDE FOR A YAW-PITCH-ROLL
C        SEQUENCE
      REAL QO(4)
C        INITIAL ATTITUDE QUATERNION OF CHASE VEHICLE
      REAL TPHIH, TPHIV
C        TANGENTS OF HALF-FIELD-OF-VIEW ANGLES IN HORIZONTAL AND VERTICAL
C        DIRECTIONS FOR CAMERA SCANNING, RESPECTIVELY
      REAL STATE(14)
C        CHASE VEHICLE STATE VECTOR
      INTEGER TERMNL
C        FORTRAN LOGICAL UNIT NUMBER OF USER TERMINAL
      REAL T
C        ELAPSED TIME
      REAL TRNATO(3,3)
C        TRUE INITIAL TARGET DIRECTION COSINE MATRIX
      REAL TUMRAT
C        TARGET TUMBLE RATE
C
      COMMON/MASPRP/FULDIS, INERCV, MEMPTY
      COMMON/TGTI/ININV
      COMMON/OPTSYS/TPHIH, TPHIV, FOCLEN, A, B
      COMMON/VEHIC/AK1, AK2, AK3, DOKRAD, F1
      COMMON/SIMUL/TERMNL, OUT
      COMMON/HNGOFF/HC, HT, HDC, HDT
      COMMON/REF/QO
      COMMON/ATINFO/TRNATO, ITUMBL, TUMRAT
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
C        (* INPUT INITIAL ATTITUDE ANGLES *)
         WRITE(TERMNL,901)
         READ(TERMNL,*) PHI,PSI,THETA
C        (* CONVERT ANGLES TO RADIANS *)
            PHI=PHI* 01745329
            PSI=PSI* 01745329
            THETA=THETA* 01745329
C        (* COMPUTE TRANSPOSE OF TRUE INITIAL TARGET ATTITUDE *)
            TRNATO(1,1)=COS(PSI)*COS(PHI)
            TRNATO(2,1)=COS(PSI)*SIN(PHI)
            TRNATO(3,1)=-SIN(PSI)
            TRNATO(1,2)=SIN(THETA)*SIN(PSI)*COS(PHI)-COS(THETA)*SIN(PHI)
            TRNATO(2,2)=SIN(PHI)*SIN(THETA)*SIN(PSI)+COS(THETA)*COS(PHI)
            TRNATO(3,2)=SIN(THETA)*COS(PSI)
            TRNATO(1,3)=COS(THETA)*SIN(PSI)*COS(PHI)+SIN(THETA)*SIN(PHI)
            TRNATO(2,3)=SIN(PHI)*COS(THETA)*SIN(PSI)-SIN(THETA)*COS(PHI)
            TRNATO(3,3)=COS(THETA)*COS(PSI)
C        (* INPUT TUMBLE AXIS *)
            WRITE(TERMNL,902)
    3       READ(TERMNL,*) ITUMBL
               IF(ITUMBL EQ 1 OR ITUMBL EQ 2 OR ITUMBL EQ 3) GO TO 5
               WRITE(TERMNL,904)
               GO TO 3
    5    CONTINUE
C        (* INPUT TUMBLE RATE *)
            WRITE(TERMNL,903)
            READ(TERMNL,*) TUMRAT
C        (* CONVERT TUMBLE RATE FROM DEGREES/HOUR TO RADIANS/SECOND
            TUMRAT=TUMRAT*4 848137E-6
         A=1
         B=1
         DOKRAD= 4
         FOCLEN= 05
         F1(1)=360
         F1(2)=160
         F1(3)=160
         F1(4)=160
         F1(5)=160
         F1(6)=160
         F1(7)=160
         F1(8)=160
         F1(9)=160
         F1(10)=160
         F1(11)=160
         F1(12)=160
         F1(13)=160
         F1(14)=360
         DO 10 I=1,3
            DO 10 J=1,3
               INERCV(I,J)=0
   10    CONTINUE
         INERCV(1,1)=1055
         INERCV(2,2)=1905
         INERCV(3,3)=1905
         TPHIH= 5
         TPHIV= 5
         T=0 0
         DO 20 I=1,14
            F(I)=0
   20    CONTINUE
         DO 30 I=1,3
            DO 30 J=1,3
               FULDIS(I,J)=0
   30    CONTINUE
         FULDIS(1,1)= 781
         FULDIS(2,2)= 706
         FULDIS(3,3)= 706
         DO 40 I=1,14
            AK1(I)=4 636E-4
   40    CONTINUE
         DO 50 I=1,3
            DO 50 J=1,14
               AK2(I,J)=0
               AK3(I,J)=0
   50    CONTINUE
         AK2(1,1)=-1
         AK2(2,2)=1
         AK2(2,3)=1
         AK2(2,4)=-1
         AK2(2,5)=-1
         AK2(2,6)=-1
         AK2(2,7)=1
         AK2(2,8)=-1
         AK2(2,9)=1
         AK2(3,10)=-1
         AK2(3,11)=-1
         AK2(3,12)=1
         AK2(3,13)=1
         AK2(1,14)=1
         AK3(3,2)=- 75
         AK3(3,3)= 75
         AK3(3,4)=- 75
         AK3(3,5)= 75
         AK3(1,6)= 75
         AK3(1,7)=- 75
         AK3(1,8)=- 75
         AK3(1,9)= 75
         AK3(2,10)=- 75
         AK3(2,11)= 75
         AK3(2,12)= 75
         AK3(2,13)=- 75
         MEMPTY=1800
         DO 60 I=1,6
            DO 60 J=1,6
               P(I,J)=0
   60    CONTINUE
         CALL MIDN(ININV,3)
         CALL MIDN(PA,7)
         P(1,1)=400
         P(2,2)=563
         P(3,3)=563
         P(4,4)= 03
         P(5,5)= 03
         P(6,6)= 03
         ESTATE(1)=-300
         ESTATE(2)=0
         ESTATE(3)=0
         ESTATE(4)= 1
         ESTATE(5)= 1
         ESTATE(6)= 1
         ININV(1,1) = 4 E-5
         ININV(2,2) = 3 56E-5
         ININV(3,3) =3 53E-5
         ESTA(1)=0
         ESTA(2)=0
         ESTA(3)=0
         ESTA(4)=1
         ESTA(5)=0
         ESTA(6)=0
         ESTA(7)=0
         PA(1,1)=10
         PA(2,2)=10
         PA(3,3)=10
         PA(4,4)=10
         PA(5,5)=4 E6
         PA(6,6)=4 E6
         PA(7,7)=4 E6
         DO 70 I=1,6
            STATE(I)=RANFN(2,ESTATE(I),SQRT(P(I,I)))
   70    CONTINUE
         STATE(7)=0 0
         STATE(8)=0 0
         STATE(9)=0 0
         STATE(10)=0
         STATE(11)=0
         STATE(12)=0
         STATE(13)=1
         STATE(14)=3700
         QO(1)=STATE(10)
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
      QO(2)=STATE(11)
      QO(3)=STATE(12)
      QO(4)=STATE(13)
      HC(1)=2 0
      HC(2)=2 0
      HC(3)=0 0
      HT(1)=-1 25
      HT(2)=2 0
      HT(3)=0 0
      HDC(1)=3 49
      HDC(2)=0
      HDC(3)=0
      HDT(1)=-2
      HDT(2)=0
      HDT(3)=0
      RETURN
  901 FORMAT(/' ENTER THREE ROTATION ANGLES TO SPECIFY INITIAL TARGET'/
     A     ' ATTITUDE FOR A YAW-PITCH-ROLL SEQUENCE---ANGLES MUST BE'/
     B     ' IN DEGREES')
  902 FORMAT(/' ENTER TUMBLE AXIS-- ENTER 1, 2, OR 3 FOR YAW, PITCH, '/
     A     ' OR ROLL, RESPECTIVELY')
  903 FORMAT(/' ENTER TUMBLE RATE IN DEGREES PER HOUR ')
  904 FORMAT(/' A VALUE OF 1, 2, OR 3 MUST BE ENTERED FOR TUMBLE AXIS')
C
      END
```

```
C
      SUBROUTINE INISIM(STATE)
C     (* 300 - INITIALIZE SIMULATOR *)
C
C     CALLS
C         <NONE>
C     CALLED BY
C         <MAIN>
C
C     INPUT
C         <NONE>
C     OUTPUT
C         <NONE>
C
C     (* DUMMY ROUTINE - ACCOMODATES PHYSICAL SIMULATION *)
      RETURN
C
      END
```

```
C
      SUBROUTINE DOCK(P,ESTATE,STATE,T,DOCKED,F,ESTA,PA)
C     (* 400 - RUN SIMULATION FOR ONE MEASUREMENT INTERVAL *)
C
C     CALLS
C        SETGOL,INCORP,SQRT,FLASH,PROPTR,POSIT,ATITUD,PROPES,THRUST,
C        IMU,RPY,MMLT,MSUB,MADD,MPROVE,INCRPA,PRPESA,AMAX1,DOKCHK,DPRD
C     CALLED BY
C        <MAIN>
C
C     INPUT
C        P,ESTATE,DOKRAD,STATE,T,F,INERCV
C     OUTPUT
C        STATE,T,F,DOCKED,ESTATE,P
C
      REAL ACV(3,3),ACVT(3,3)
C        DIRECTION COSINE MATRIX DESCRIBING CURRENT CHASE VEHICLE ATTITUDE
C        (RELATIVE TO PRIMARY REFERENCE FRAME) AND TRANSPOSE OF ACV
      REAL AT(3,3)
C        MEASURED DIRECTION COSINE MATRIX OF TARGET SPACECRAFT
      REAL ATRATE(3)
C        MEASRED ATTITUDE RATES ABOUT CHASE VEHICLE AXES
      REAL CVPOS(3)
C        MEASURED CHASE VEHICLE POSITION IN PRIMARY REFERENCE FRAME
      REAL DEDLIN
C        TIME BY WHICH STRATEGY LOGIC WANTS CHASE VEHICLE IN GOAL 'BOX'
      REAL DOKRAD
C        RADIUS FROM TARGET AT WHICH CHASE VEHICLE IS CONSIDERED DOCKED
      REAL DU,DV
C        CHANGE IN CENTER-LAMP IMAGE COORDINATES BETWEEN FLASHES
      REAL ESTA(7)
C        STATE ESTIMATE FOR TARGET ATTITUDE
      REAL ESTATE(6)
C        ESTIMATE OF STATE VECTOR FOR CHASE VEHCLE TRANSLATION
      REAL F(14)
C        FORCES FROM CHASE VEHICLE THRUSTERS AFTER SELECTION
      REAL FOCLEN
C        LENS FOCAL LENGTH (MM)
      REAL GXL,GXH,GYL,GYH,GZL,GZH
C        X,Y,Z UPPER,LOWER LIMITS OF GOAL 'BOY'
      REAL HDC(3),HDT(3)
C        DOCKING FIXTURES' POSITIONS IN RESPECTIVE SPACECRAFT COORDINATE
C        SYSTEMS
      INTEGER OUT
C        FORTRAN UNIT NUMBER FOR OUTPUT FILE
      REAL P(6,6),PA(7,7)
C        COVARIANCE OF STATE ESTIMATES CORRESPONDING TO ESTATE() AND ESTA()
      REAL QMEAS(4)
C        MEASURED TARGET ATTITUDE QUATERNION (W R T  PRIMARY FRAME)
      REAL RELPOS(3)
C        RELATIVE CAMERA COORDINATES IN CURRENT DOCKING AID FRAME CENTERED
C        AT CENTER LIGHT
      REAL RHO
C        MEASURED RANGE, CAMERA TO TARGET DOCKING AID CENTER LIGHT
      REAL RPYERR(3)
C        MEASURED ROLL, PITCH, AND YAW ERRORS IN RADIANS
      REAL STATE(14)
C        CHASE VEHICLE STATE VECTOR
      REAL T
C        ELAPSED TIME
      REAL TACUT(3,3)
C        TRANSPOSE OF TARGET SPACECRAFT DIRECTION COSINE MATRIX
C        (WITH RESPECT TO TRUTH AXES) FOR OUTPUT ONLY
      REAL TPHIH,TPHIV
C        TANGENTS OF HALF-FIELD-OF-VIEW ANGLES FOR HORIZ,VERT
      REAL U(4),V(4)
C        COORDINATES OF LAMPS' IMAGES
      LOGICAL DOCKED
C        TRUE IF CHASE VEHICLE IS WITHIN DOCKING RADIUS
      LOGICAL VALID
C        TRUE IF MEASUREMENT VALID (LIGHT IN FIELD OF VIEW)
C
      COMMON/SIMUL/IDUMMY,OUT
      COMMON/HNGOFF/DUMMY3(6),HDC,HDT
      COMMON/VEHIC/DUMMY1(98),DOKRAD,DUMMY2(14)
      COMMON/OPTSYS/TPHIH,TPHIV,FOCLEN,DUMMY4(2)
```

```
C
C     (* TAKE MEASUREMENT *)
        VALID= TRUE
C         (* 410 - FLASH CENTER LIGHT *)
           CALL FLASH(U(4),V(4),2 T,STATE,VALID)
C         (* 420 - PROPAGATE TRUE STATE 1/30 SEC  *)
           CALL PROPTR( 03333333,F,STATE,T)
C         (* 410 - FLASH LEFT LIGHT *)
           CALL FLASH(U(1),V(1),1,T,STATE,VALID)
C         (* 420 - PROPAGATE TRUE STATE 1/30 SEC *)
           CALL PROPTR( 03333333,F,STATE,T)
C         (* 410 - FLASH RIGHT LIGHT *)
           CALL FLASH(U(3),V(3),3,T,STATE,VALID)
C         (* 420 - PROPAGATE TRUE STATE 1/30 SEC *)
           CALL PROPTR( 03333333,F,STATE,T)
C         (* 410 - FLASH CENTER LIGHT SECOND TIME *)
           CALL FLASH(U(2),V(2),2,T,STATE,VALID)
C         (* 490 - TA/E IMU MEASUREMENT *)
           CALL IMU(STATE,ACV,ACVT,ATRATE)
C         (* 420 - PROPAGATE TRUE STATE TO TIME OF THRUSTER COMMAND CHANGE *)
           CALL PROPTR( 53333333,F,STATE,T)
      IF( NOT VALID) GO TO 10
C          (* ADJUST CENTROIDS FOR VEHICLE MOTION *)
            DU=U(2)-U(4)
            DV=V(2)-V(4)
            U(1)=U(1)+ 6666667*DU
            V(1)=V(1)+ 6666667*DV
            U(3)=U(3)+ 3333333*DU
            V(3)=V(3)+ 3333333*DV
C          (* 430 - CALCULATE POSITONS AT TIME OF 4TH FLASH *)
            CALL POSIT(U,V,RELPOS,RHO)
C          (* 440 - CALC  TARGET ATTITUDE AND CHASE VEHICLE POSITION AT 4TH
C           FLASH *)
            CALL ATITUD(ACV,ACVT,RELPOS,RHO,U,V,GMEAS,CVPOS)
C          (* 4C0 - REFINE IMAGE INTERPRETATION TO ACCOUNT FOR PERSPECTIVE *)
            CALL MPROVE(U,V,ACV,GMEAS,CVPOS)
            CALL MPROVE(U,V,ACV,GMEAS,CVPOS)
C          (* 450 - USE MEASUREMENT TO UPDATE STATE ESTIMATE *)
            CALL INCORP(CVPOS,P,ESTATE,ACV,ACVT)
C          (* 4D0 - USE MEASUREMENT TO UPDATE ATTITUDE STATE ESTIMATE *)
            CALL INCRPA(PA,ESTA,ESTATE,GMEAS)
   10 CONTINUE
C      (* WRITE PARAMETERS TO DISK FILE *)
         CALL TRGATT(TAOUT,T)
         WRITE(DUT) T,STATE,TAOUT
C      (* 460 - PROPAGATE STATE ESTIMATE TO TIME OF THRUSTER COMMAND CHANGE *)
         CALL PROPES(P,ESTATE,F, 5333333,ACVT)
C      (* 4E0 - PROPAGATE ATTITUDE STATE ESTIMATE TO SAME TIME *)
         CALL PRPESA(PA,ESTA, 5333333)
C      (* 470 - SET GOAL FOR NEXT MEASUREMENT INTERVAL *)
         CALL SETGOL(ESTATE,P,T,ACV,ESTA,PA,GXL,GXH,GYL,GYH,GZL,GZH,
     A      DEDLIN)
C      (* 490 - GET LATEST IMU READING *)
         CALL IMU(STATE,ACV,ACVT,ATRATE)
C      (* 4A0 - CALCULATE ROLL, PITCH, YAW ERRORS *)
         CALL RPY(RPYERR,ACV,ESTATE,ESTA)
C      (* 4B0 - CONTROL THRUSTERS *)
         CALL THRUST(F,ESTATE,GXL,GXH,GYL,C:H,GZL,GZH,AMAXI(
     A      DEDLIN-T,0 ),ACV,ATRATE,RPYE:R)
C      (* 420 - PROPAGATE TRUE STATE TO TIME OF FIRST FLASH OF NEXT SEQUENCE *)
         CALL PROPTR(0 6,F,STATE,T)
C      (* 460 - PROP  STATE ESTIMATE TO TIME OF 4TH FLASH OF NEXT SEQUENCE *)
         CALL PROPES(P,ESTATE,F,0 7,ACVT)
C      (* 4E0 - PROPAGATE ATTITUDE STATE ESTIMATE TO SAME TIME *)
         CALL PRPESA(PA,ESTA,0 7)
C      (* 4F0 - SEE IF CHASE VEHICLE HAS DOCKED YET *)
         CALL DOKCHK(KDC,STATE,DOKRAD,HDT,DOCKED,T)
C      (* WRITE PARAMETERS TO DISK FILE *)
         CALL TRGATT(TAOUT,T)
         WRITE(DUT) T,STATE,TAOUT
C        WRITE(1,901) T,STATE(1),STATE(2),STATE(3)
C        WRITE(1,902) STATE(4),STATE(5),STATE(6)
C        WRITE(1,903) STATE(10),STATE(11),STATE(12),STATE(13)
C        WRITE(1,904) STATE(7),STATE(8),STATE(9)
C        WRITE(1,905) STATE(14)
C      (* SWAP LENSES IF NECESSARY *)
        IF(VALID AND DPRD(ESTATE,ESTATE,3) GT 225 ) GO TO 20
           TPHIH=1 0
           TPHIV=1 0
           FOCLEN= 025
           GO TO 30
   20   CONTINUE
           TPHIH=0 5
           TPHIV=0 5
           FOCLEN= 05
   30   CONTINUE
      RETURN
C
  901 FORMAT(/' -----TIME=',F10 2/' POSITION IS ',3G14 7)
  902 FORMAT(' VELOCITY IS ',3G14 7)
  903 FORMAT(' QUATERNION IS ',4G14 7)
  904 FORMAT(' ANGULAR MOMENTUM IS ',3G14 7)
  905 FORMAT(' MASS IS ',G14 7)
C
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
      SUBROUTINE FLASH(U,V,LAMP,T,STATE,VALID)
C     (* 410 - FLASH A LAMP *)
C
C     CALLS
C        TRGATT,MMLT,DIRMAT,MSUB,RANFN,MTRN,ABS,SQRT,DPRD
C     CALLED BY
C        DOCK
C
C     INPUT
C        FOCLEN,TPHIH,TPHIV,LAMP,STATE,A,B,VALID
C     OUTPUT
C        U,V,VALID
C
      REAL AC(3,3),TRNAC(3,3)
C        CHASE VEHICLE DIRECTION COSINE MATRIX AND ITS TRANSPOSE
      REAL A,B
C        LAMP SEPARATIONS (IN METERS)
      REAL AT(3,3),TRNAT(3,3)
C        TARGET DIRECTION COSINE MATRIX & ITS TRANSPOSE
      REAL HC(3),HT(3)
C        CAMERA POSITION IN CHASE VEHICLE BODY FRAME AND DOCKING AID POSITION
C        IN TARGET SPACECRAFT BODY FRAME, RESPECT.VELY
      REAL FOCLEN
C        LENS FOCAL LENGTH IN METERS
      INTEGER LAMP
C        LAMP NUMBER (1-3)   1=LEFT,2=CENTER,3=RIGHT
      REAL STATE(14)
C        TRUE CHASE VEHICLE STATE
      REAL T
C        ELAPSED TIME
      REAL TPHIH,TPHIV
C        TANGENTS OF HALF-FIELD-OF-VIEW ANGLES IN HORIZONTAL AND VERTICAL
C        DIRECTIONS FOR CAMERA SCANNING, RESPECTIVELY
      REAL U,V
C        COORDINATES OF CENTROID OF LAMP IMAGE IN METERS
      LOGICAL VALID
C        TRUE IF MEASUREMENT VALID (LIGHTS IN FIELD OF VIEW)
      REAL VLT(3)
C        COORDINATES OF LAMP WITH RESPECT TO CAMERA
      REAL VL(3),V1(3),V2(3),V3(3),V5(3),V6(3)
C        INTERMEDIATE RESULTS WITH NO PROBLEM-RELATED SIGNIFICANCE
C
      COMMON/OPTSYS/TPHIH,TPHIV,FOCLEN,A,B
      COMMON/HNGOFF/HC,HT,DUMMY(6)
C
C
C     (* FIND LAMP POSITION (=VLT) WITH RESPECT TO CAMERA *)
      (* 905 - COMPUTE TARGET ATTITUDE *)
         CALL TRGATT(TRNAT,T)
      VL(3)=HT(3)
      GO TO (10,20,30)  LAMP
   10    CONTINUE
            VL(1)=HT(1)
            VL(2)=HT(2)+A
            GO TO 40
   20    CONTINUE
            VL(1)=HT(1)-B
            VL(2)=HT(2)
            GO TO 40
   30    CONTINUE
            VL(1)=HT(1)
            VL(2)=HT(2)-A
   40 CONTINUE
      CALL MMLT(V1,TRNAT,VL,3,3,1)
      CALL MSUB(V2,V1,STATE,3,1)
      CALL DIRMAT(STATE(10),AC,TRNAC)
      CALL MMLT(V3,AC,V2,3,3,1)
      CALL MSUB(VLT,V3,HC,3,1)
C     (* BE SURE DOCKING AID IS IN FIELD OF VIEW OF CAMERA *
      VALID=VALID  AND  (VLT(1) GT 0 0)  AND  ABS(VLT(2)) LE TPHIH
     A   *VLT(1)  AND  ABS(VLT(3)) LE TPHIV*VLT(1)
      IF( NOT VALID) RETURN
C     (* FIND DEFLECTION *)
      U=-VLT(2)*FOCLEN/VLT(1)
      V=VLT(3)*FOCLEN/VLT(1)
C     (* BE SURE DOCKING AID IS ON VISIBLE SIDE OF TARGET SPACECRAFT *)
      CALL MMLT(V5,TRNAC,VLT,3,3,1)
      CALL MTRN(AT,TRNAT,3)
      CALL MMLT(V6,AT,V5 3,3,1)
      VALID=VALID AND V6(1) GT SQRT(DPRD(V6,V6,3))*0 16
C     (* CORRUPT CENTROID COORDINATES WITH NOISE *)
      U=RANFN(2,U,TPHIH*FOCLEN/600 )
      V=RANFN(2,V,TPHIV*FOCLEN/600 )
      RETURN
C
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
      SUBROUTINE PROPTR(DT,F,STATE,T)
C     (* 420 - PROPAGATE TRUE STATE BY 4TH-ORDER RUNGE-KUTTA INTEGRATION *)
C
C     CALLS
C        COMPK1,COMPK2,COMPK3,COMPK4,POINT,AMIN1,SQRT
C     CALLED BY
C        DOCK
C
C     INPUT
C        DT,F,STATE,T
C     OUTPUT
C        STATE,T
C
      REAL DT
C        INTEGRATION INTERVAL SIZE
      REAL F(14)
C        FORCES FROM THRUSTERS AFTER SELECTION
      INTEGER I
C        DO LOOP INDEX
      REAL K1(14),K2(14),K3(14),K4(14)
C        VECTORS K1 THRU K4 USED IN RUNGE-KUTTA INTEGRATION
      REAL OM
C        SQUARE ROOT OF SUM OF SQUARES OF QUATERNION ELEMENTS
      REAL STATE(14)
C        CHASE VEHICLE STATE VECTOR
      REAL STEP
C        STEP SIZE FOR INTEGRATION
      REAL T,TO
C        ELAPSED TIME AND TIME AT START OF INTEGRATION INTERVAL
      REAL TLEFT
C        TIME LEFT TO GO OUT OF DT
      PARAMETER STEPMX=0.2
C        MAXIMUM INTEGRATION STEP SIZE
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
      TLEFT=DT
      TO=T
   10 IF(TLEFT LE 0) GO TO 40
         STEP=AMIN1(STEPMX,TLEFT)
         TLEFT=TLEFT-STEP
C        (* 421 - COMPUTE K1 *)
            CALL COMPK1(F,K1,STATE,STEP,T)
C        (* 422 - COMPUTE K2 *)
            CALL COMPK2(F,K1,STATE,STEP,T,K2)
C        (* 423 - COMPUTE K3 *)
            CALL COMPK3(F,K2,STATE,STEP,T,K3)
C        (* 424 - COMPUTE K4 *)
            CALL COMPK4(F,K3,STATE,STEP,T,K4)
C        (* COMPUTE NEW STATE *)
            DO 20 I=1,14
               STATE(I)=STATE(I)+(K1(I)+2.*K2(I)+2.*K3(I)+K4(I))/6
   20       CONTINUE
C        (* NORMALIZE QUATERNION *)
            OM=SQRT(STATE(10)**2+STATE(11)**2+STATE(12)**2+
     A         STATE(13)**2)
            DO 30 I=10,13
               STATE(I)=STATE(I)/OM
   30       CONTINUE
C        (* 425 - POINT SIMULATION CAMERA *)
            CALL POINT(STATE)
         T=T+STEP
         GO TO 10
   40 CONTINUE
      T=TO+DT
      RETURN
C
      END
```

```
C
      SUBROUTINE COMPK1(F,K1,STATE,STEP,T)
C     (* 421 - COMPUTE VECTOR K1 FOR RUNGE-KUTTA INTEGRATION *)
C
C     CALLS
C        STPRIM,MSCL,MADD
C     CALLED BY
C        PROPTR
C
C     INPUT
C        F,STATE,STEP,T,FULDIS,INERCV,MEMPTY
C     OUTPUT
C        K1
C
      REAL DSTATE(14)
C        TIME DERIVATIVE OF STATE VECTOR
      REAL F(14)
C        FORCES FROM THRUSTERS AFTER SELECTION
      REAL FULDIS(3,3)
C        CHASE VEHICLE FUEL-DISTRIBUTION INERTIA TENSOR
      INTEGER I
C        DO LOOP INDEX
      REAL INERCV(3,3)
C        CHASE VEHICLE MOMENT OF INERTIA TENSOR
      REAL INERFL(3,3)
C        INERTIA OF FUEL
      REAL INERTA(3,3)
C        INERTIA OF CHASE VEHICLE (LOADED)
      REAL K1(14)
C        VECTOR K1 USED IN RUNGE-KUTTA INTEGRATION
      REAL MEMPTY
C        CHASE VEHICLE MASS WITHOUT FUEL
      REAL STATE(14)
C        CHASE VEHICLE STATE VECTOR
      REAL STEP
C        STEP SIZE FOR INTEGRATION
      REAL T
C        ELAPSED TIME
C
      COMMON/MASPRP/FULDIS,INERCV,MEMPTY
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* COMPUTE INERTIA *)
         CALL MSCL(INERFL,FULDIS,3,3,STATE(14)-MEMPTY)
         CALL MADD(INERTA,INERCV,INERFL,3,3)
C     (* 902 - COMPUTE STATE DERIVATIVE *)
         CALL STPRIM(STATE,F,INERTA,T,DSTATE)
C     (* COMPUTE K1=STEP*(STATE DERIVATIVE) *)
         DO 10 I=1,14
            K1(I)=STEP*DSTATE(I)
   10    CONTINUE
      RETURN
C
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
      SUBROUTINE COMPK2(F,K1,STATE,STEP,T,K2)
C     (* 422 - DETERMINE VECTOR K2, FOR RUNGE-KUTTA INTEGRATION *)
C
C     CALLS
C        MADD,MSCL,STPRIM
C     CALLED BY
C        PROPTR
C
C     INPUT
C        F,FULDIS,INERCV,K1,MEMPTY,STATE,STEP,T
C     OUTPUT
C        K2
C
      REAL DSTATE(14)
C        TIME DERIVATIVE OF STATE VECTOR
      REAL F(14)
C        FORCES FROM THRUSTERS AFTER SELECTION
      REAL FULDIS(3,3)
C        CHASE VEHICLE FUEL-DISTRIBUTION INERTIA TENSOR
      INTEGER I
C        DO LOOP INDEX
      REAL INERCV(3,3)
C        INERTIA OF EMPTY CHASE VEHICLE
      REAL INERFL(3,3)
C        INERTIA OF FUEL
      REAL INERTA(3,3)
C        INERTIA OF CHASE VEHICLE (LOADED)
      REAL K1(14),K2(14)
C        VECTORS K1 AND K2 USED IN RUNGE-KUTTA INTEGRATION
      REAL MEMPTY
C        MASS OF CHASE VEHICLE WITHOUT FUEL
      REAL STATE(14)
C        CHASE VEHICLE STATE VECTOR
      REAL STEP
C        STEP SIZE FOR INTEGRATION
      REAL T
C        ELAPSED TIME
      REAL TEMPST(14)
C        TEMPORARY STATE VECTOR
C
      COMMON/MASPRP/FULDIS,INERCV,MEMPTY
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* COMPUTE TEMPORARY STATE *)
         DO 10 I=1,14
            TEMPST(I)=STATE(I)+0 5*K1(I)
   10    CONTINUE
C     (* DETERMINE TEMPORARY INERTIA AS FUNCTION OF TEMPORARY
C     STATE MASS *)
         CALL MSCL(INERFL,FULDIS 3,3,TEMPST(14)-MEMPTY)
         CALL MADD(INERTA,INERCV,INERFL,3,3)
C     (* 902 - COMPUTE DERIVATIVE AT TEMPORARY STATE *)
         CALL STPRIM(TEMPST,F,INERTA,T+ 5*STEP,DSTATE)
C     (* COMPUTE K2=STEP*(DERIVATIVE AT TEMPORARY STATE *)
         DO 20 I=1,14
            K2(I)=STEP*DSTATE(I)
   20    CONTINUE
      RETURN
C
      END
```

```
C
      SUBROUTINE COMPK3(F,K2,STATE,STEP,T,K3)
C     (* 423 - DETERMINE VECTOR K3, FOR RUNGE-KUTTA INTEGRATION *)
C
C     CALLS
C        MADD,MSCL,STPRIM
C     CALLED BY
C        PROPTR
C
C     INPUT
C        F,FULDIS,INERCV,K2,MEMPTY,STATE,STEP,T
C     OUTPUT
C        K3
C
      REAL DSTATE(14)
C        TIME DERIVATIVE OF STATE VECTOR
      REAL F(14)
C        FORCES FROM THRUSTERS AFTER SELECTION
      REAL FULDIS(3,3)
C        CHASE VEHICLE FUEL-DISTRIBUTION INERTIA TENSOR
      INTEGER I
C        DO LOOP INDEX
      REAL INERCV(3,3)
C        INERTIA OF EMPTY CHASE VEHICLE
      REAL INERFL(3,3)
C        INERTIA OF FUEL
      REAL INERTA(3,3)
C        INERTIA OF CHASE VEHICLE (LOADED)
      REAL K2(14),K3(14)
C        VECTORS K2 AND K3 USED IN RUNGE-KUTTA INTEGRATION
      REAL MEMPTY
C        MASS OF CHASE VEHICLE WITHOUT FUEL
      REAL STATE(14)
C        CHASE VEHICLE STATE VECTOR
      REAL STEP
C        STEP SIZE FOR INTEGRATION
      REAL T
C        ELAPSED TIME
      REAL TEMPST(14)
C        TEMPORARY STATE VECTOR
C
      COMMON/MASPRP/FULDIS,INERCV,MEMPTY
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* COMPUTE TEMPORARY STATE *)
         DO 10 I=1,14
            TEMPST(I)=STATE(I)+0 5*K2(I)
   10    CONTINUE
C     (* DETERMINE TEMPORARY INERTIA AS A FUNCTION OF TEMPORARY
C              STATE MASS *)
         CALL MSCL(INERFL,FULDIS,3,3,TEMPST(14)-MEMPTY)
         CALL MADD(INERTA,INERCV,INERFL,3,3)
C     (* 902 - COMPUTE DERIVATIVE AT TEMPORARY STATE *)
         CALL STPRIM(TEMPST,F,INERTA,T+ 5*STEP,DSTATE)
C     (* COMPUTE K3=STEP*(DERIVATIVE AT TEMPORARY STATE *)
         DO 20 I=1,14
            K3(I)=STEP*DSTATE(I)
   20    CONTINUE
      RETURN
C
      END
```

```
C
      SUBROUTINE COMPK4(F,K3,STATE,STEP,T,K4)
C     (* 424 - DETERMINE VECTOR K4, FOR RUNGE-KUTTA INTEGRATION *)
C
C     CALLS
C        MADD,MSCL,STPRIM
C     CALLED BY
C        PROPTR
C
C     INPUT
C        F,FULDIS,INERCV,K3,MEMPTY,STATE,STEP,T
C     OUTPUT
C        K4
C
      REAL DSTATE(14)
C        TIME DERIVATIVE OF STATE VECTOR
      REAL F(14)
C        FORCES FROM THRUSTERS AFTER SELECTION
      REAL FULDIS(3,3)
C        CHASE VEHICLE FUEL-DISTRIBUTION INERTIA TENSOR
      INTEGER I
C        DO LOOP INDEX
      REAL INERCV(3,3)
C        INERTIA OF EMPTY CHASE VEHICLE
      REAL INERFL(3,3)
C        INERTIA OF FUEL
      REAL INERTA(3,3)
C        INERTIA OF CHASE VEHICLE (LOADED)
      REAL K3(14),K4(14)
C        VECTORS K3 AND K4 USED IN RUNGE-KUTTA INTEGRATION
      REAL MEMPTY
C        MASS OF CHASE VEHICLE WITHOUT FUEL
      REAL STATE(14)
C        CHASE VEHICLE STATE VECTOR
      REAL STEP
C        STEP SIZE FOR INTEGRATION
      REAL T
C        ELAPSED TIME
      REAL TEMPST(14)
C        TEMPORARY STATE VECTOR
C
      COMMON/MASPRP/FULDIS,INERCV,MEMPTY
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* COMPUTE TEMPORARY STATE *)
      DO 10 I=1,14
         TEMPST(I)=STATE(I)+K3(I)
   10 CONTINUE
C     (* DETERMINE TEMPORARY INERTIA AS A FUNCTION OF TEMPORARY
C              STATE MASS *)
      CALL MSCL(INERFL,FULDIS,3,3,TEMPST(14)-MEMPTY)
      CALL MADD(INERTA,INERCV,INERFL,3,3)
C     (* 902 - COMPUTE DERIVATIVE AT TEMPORARY STATE *)
      CALL STPRIM(TEMPST,F,INERTA,T+STEP,DSTATE)
C     (* COMPUTE K4=STEP*(DERIVATIVE AT TEMPORARY STATE *)
      DO 20 I=1,14
         K4(I)=STEP*DSTATE(I)
   20 CONTINUE
      RETURN
C
      END
```

```
C
      SUBROUTINE POINT(STATE)
C     (* 425 - POSITION SIMULATION CAMERA *)
C     (* THIS DUMMY SUBROUTINE IS FOR LATER EXPANSION TO PHYSICAL SIMULATION*)
C
C     CALLS
C        (NONE)
C     CALLED BY
C        PROPTR
C
      RETURN
C
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
      SUBROUTINE POSIT(U,V,RELPOS,RHO)
C     (* 430 - COMPUTE CAMERA POSITION (IN DOCKING-AID COORD  SYSTEM) FROM
C         LIGHT IMAGE CENTROIDS *)
C
C     CALLS
C         SQR1,ABS
C     CALLED BY
C         DOCK
C
C     INPUT
C         U,V
C     OUTPUT
C         RELPOS,RHO
C
      REAL A,B
C         LIGHT SPACING (FIXED FOR ANY ONE TARGET SPACECRAFT)
      REAL FOCLEN
C         LENS FOCAL LENGTH IN METERS
      REAL RELPOS(3)
C         RELATIVE CAMERA COORDINATES IN CURRENT TARGET DOCKING AID REFERENC
C         FRAME CENTERED AT CENTER LIGHT
      REAL RHO
C         DISTANCE BETWEEN LIGHTS AND CAMERA
      REAL U(4),V(4)
C         HORIZONTAL AND VERTICAL COMPONENTS FOR LEFT, 2ND CENTER, RIGHT,
C         AND FIRST CENTER CENTROID POSITIONS, RESPECTIVELY
      REAL AP,AK,BP,D,H,S,SP,UC,VC,XP,YP,ZP,VM,UM
C         INTERMEDIATE RESULTS, SEE ENGR NOTEBOOK #1035, P  44
C
      COMMON/OPTSYS/DUMMY(2),FOCLEN,A,B
C
      VM= 5*(V(1)+V(3))
      UM= 5*(U(1)+U(3))
      IF(V(1) NE V(3)) GO TO 10
          AP= 5*ABS(U(3)-U(1))
          UC=U(2)
          VC=V(1)
          H=ABS(V(1)-V(2))
          GO TO 30
   10 IF(U(1) NE U(3)) GO TO 20
          AP= 5*ABS(V(3)-V(1))
          UC=U(1)
          VC=V(1)
          H=ABS(U(1)-U(2))
          GO TO 30
   20 CONTINUE
          AP= 5*SQR1((U(3)-U(1))**2+(V(3)-V(1))**2)
          S=(V(1)-V(3))/(U(1)-U(3))
          SP=-1 /S
          UC=(V(2)-V(1)+S*U(1)-SP*U(2))/(S-SP)
          VC=SP*(UC-U(2))+V(2)
          H=SQR1((UC-U(2))**2+(VC-V(2))**2)
   30 CONTINUE
      BP=SQR1((VM-V(2))**2+(UM-U(2))**2)
      IF(AP NE 0  OR BP NE 0 ) GO TO 40
          RHO=800
          RELPOS(1)=-800
          RELPOS(2)=0
          RELPOS(3)=0
          RETURN
   40 IF(AP NE 0 ) GO TO 50
          RHO=B*FOCLEN/BP
          YP=RHO
          XP=0
          ZP=0
          GO TO 70
   50 IF(H NE 0 ) GO TO 60
          ZP=0
          D=BP*A/(B*AP)
          XP=-1 /SQR1(1 +D**2)
          YP=D*XP
          RHO=A*XP*FOCLEN/AP
          IF((U(1)-U(2))**2+(V(1)-V(2))**2 LT
     A         (U(3)-U(2))**2+(V(3)-V(2))**2) YP=-YP
          GO TO 70
   60 CONTINUE
          D=(AP*B/(BP*A))**2
          AK=BP*(1 +D)/(2 *H*SQR1(D))
          ZP=AK-SQR1(AK**2-1 )
          XP=-SQR1((D-ZP**2)/(1 +D))
          YP=SQR1((1 -D*ZP**2)/(1 +D))
          IF((U(3)-U(1))*(V(2)-VC)  GT
     A         (V(3)-V(1))*(U(2)-UC) )  ZP=-ZP
          IF((U(1)-U(2))**2+(V(1)-V(2))**2 LT
     A         (U(3)-U(2))**2+(V(3)-V(2))**2) YP=-YP
          RHO=A*SQR1(XP**2+ZP**2)*FOCLEN/AP
   70 CONTINUE
      RELPOS(1)=XP*RHO+B
      RELPOS(2)=YP*RHO
      RELPOS(3)=ZP*RHO
      RHO=SQR1(RELPOS(1)**2+RELPOS(2)**2+RELPOS(3)**2)
      RETURN
C
      END
```

```
C
      SUBROUTINE ATITUD(ACV,ACVT,RELPOS,RHO,U,V,QT,CVPOS)
C     (* 440 - DETERMINE TARGET ATTITUDE AND CHASE VEHICLE POSITION IN PRIMARY
C        REFERENCE FRAME *)
C
C     CALLS
C        FINDCV, DIRMAT, QUATRN, MADD, MMLT, MSUB
C     CALLED BY
C        DOCK
C
C     INPUT
C        RELPOS, RHO, U, V, ACV, ACVT, HC, HT, B
C     OUTPUT
C        AT, CVPOS
C
      REAL ACV(3,3),ACVT(3,3)
C        MEASURED DIRECTION COSINE MATRIX DESCRIBING CURRENT CHASE VEHICLE
C        ATTITUDE RELATIVE TO PRIMARY REFERENCE FRAME, AND ITS TRANSPOSE
      REAL AT(3,3),ATT(3,3)
C        DIRECTION COSINE MATRIX OF TARGET SPACECRAFT (IN PRIMARY REF  FRAME),
C        AND TRANSPOSE OF THIS MATRIX
      REAL B
C        DOCKING AID BASE-TO-CE .TER-LIGHT DISTANCE
      REAL CAMPOS(3)
C        MEASURED CAMERA POSITION IN COORD  SYS  CENTERED AT CENTER DOCKING
C        AID LIGHT AND PARALLEL TO PRIMARY REFERENCE FRAME
      REAL CVPOS(3)
C        MEASURED CHASE VEHICLE POSITION IN PRIMARY REFERENCE FRAME
      REAL HC(3),HT(3)
C        CAMERA POSITION IN CHASE VEHICLE BODY FRAME AND DOCKING AID POSITION
C        IN TARGET SPACECRAFT BODY FRAME, RESPECTIVELY
      REAL HTA(3)
C        HT PLUS LENGTH OF CENTER LIGHT SUPPORT ROD
      REAL QT(4)
C        QUATERNION CORRESPONDING TO AT
      REAL RELPOS(3)
C        RELATIVE CHASE VEHICLE COORDINATES IN CURRENT TARGET SPACECRAFT
C        REFERENCE FRAME
      REAL RHO
C        DISTANCE BETWEEN LIGHTS AND CAMERA
      REAL U(3),V(3)
C        HORIZONTAL AND VERTICAL COMPONENTS OF CENTER-LIGHT CENTROID
      REAL V1(3),V2(3),V3(3)
C        INTERMEDIATE RESULTS WITH NO PROBLEM-RELATED SIGNIFICANCE
C
      COMMON/HNGOFF/HC,HT,DUMMY(6)
      COMMON/OPTSYS/DUMMY1(4),B
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C
      (* 441 - COMPUTE CAMERA POSITION WITH RESPECT TO DOCKING AID *)
         CALL FINDCV(ACVT,RHO,U(2),V(2),CAMPOS)
C     (* 442 - COMPUTE TARGET ATTITUDE IN PRIMARY REFERENCE FRAME *)
         CALL QUATRN(CAMPOS,RELPOS,QT,ACV,U,V)
C     (* 901 - COMPUTE TARGET DIRECTION COSINE MATRIX FROM QUATERNION *)
         CALL DIRMAT(QT,AT,ATT)
C     (* COMPUTE POSITION OF CHASE VEHICLE CENTER OF GRAVITY IN PRIMARY REF
C        FRAME *)
         HTA(1)=HT(1)-B
         HTA(2)=HT(2)
         HTA(3)=HT(3)
         CALL MMLT(V2,ATT,HTA,3,3,1)
         CALL MADD(V1,CAMPOS,V2,3,1)
         CALL MMLT(V3,ACVT,HC,3,3,1)
         CALL MSUB(CVPOS,V1,V3,3,1)
      RETURN
C
      END
```

```
C
      SUBROUTINE FINDCV(ACVT,RHO,UC,VC,CAMPOS)
C     (* 441 - COMPUTE POSITION OF CAMERA FRAME PARALLEL TO PRIMARY REF  FRAME
C        BUT CENTERED AT CENTER TARGET LIGHT *)
C
C     CALLS
C        MMLT,SQRT
C     CALLED BY
C        ATITUD
C
C     INPUT
C        ACVT,RHO,UC,VC,FOCLEN
C     OUTPUT
C        CAMPOS
C
      REAL ACVT(3,3)
C        DIRECTION COSINE MARTIX DESCRIBING CURRENT CHASE VEHICLE ATTITUDE
C        RELATIVE TO PRIMARY REFERENCE FRAME
      REAL CAMPOS(3)
C        MEASURED CAMERA POSITION IN FRAME PARALLEL TO PRIMARY REF  FRAME
C        BUT CENTERED AT CENTER TARGET LIGHT
      REAL FOCLEN
C        CAMERA LENS FOCAL LENGTH, IN METERS
      REAL RATIO
C        INTERMEDIATE RESULTS, NO PROBLEM RELATED SIGNIFICANCE
      REAL RHO
C        MEASURED DISTANCE BETWEEN CENTER DOCKIN AID LIGHT AND CAMERA
      REAL UC,VC
C        HORIZONTAL AND VERTICAL COMPONENTS OF THE TARGET CENTER-LIGHT
C        IMAGE CENTROID
      REAL VEC1(3)
C        NEGATIVE OF TARGET POSITION IN CAMERA FRAME
C
      COMMON/OPTSYS/DUMMY1(2),FOCLEN,DUMMY2(2)
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* COMPUTE SCALAR MULTIPLIER *)
         RATIO=RHO/SQRT(FOCLEN**2+UC**2+VC**2)
C     (* FIND NEG  OF TARGET POSITION IN CAMERA FRAME *)
         VEC1(1)=-FOCLEN*RATIO
         VEC1(2)=UC*RATIO
         VEC1(3)=-VC*RATIO
C     (* CONVERT TO CAMERA POSITION IN TARGET CENTER-LIGHT FRAME *)
         CALL MMLT(CAMPOS,ACVT,VEC1,3,3,1)
      RETURN
C
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
      SUBROUTINE QUATRN(CAMPOS,RELPOS,QT,ACV,U,V)
C     (* 442 - COMPUTE TARGET ATTITUDE QUATERNION IN PRIMARY REFERENCE FRAME
C        FROM MEASUREMENT *)
C
C     CALLS
C        ATAN2,SIN,SQR1,COS,MSCL,DIRMAT,MMLT,SQRT
C     CALLED BY
C        ATITUD
C
C     INPUT
C        CAMPOS,RELPOS,ACV,U,V
C     OUTPUT
C        QT
C
      REAL ACV(3,3)
C        CHASE VEHICLE'S MEASURED DIRECTION COSINE MATRIX
      REAL AT(3,3),TRNAT(3,3)
C        DIRECTION COSINE MATRIX FOR TARGET (WITH RESPECT TO PRIMARY REF
C        FRAME, BEFORE CORRECTION) AND ITS TRANSPOSE
      REAL CAMPOS(3)
C        MEASURED CAMERA POSITION IN REFERENCE FRAME PARALLEL TO PRIMARY
C        REF  FRAME BUT CENTERED AT CENTER DOCKING AID LIGHT
      REAL DOTPRD
C        DOT PRODUCT OF CAMPOS, RELPOS
      REAL PHI
C        EULER ROTATION ANGLE
      REAL PRDMAG
C        MAGNITUDE OF CROSS PRODUCT
      REAL QC(4)
C        CORRECTION QUATERNION
      REAL QT1(4),QT(4)
C        TARGET ATTITUDE QUATERNION (MAYBE WITH ROTATION ABOUT LINE OF SIGHT)
C        AND CORRECTED VERSION OF SAME
      REAL RELPOS(3)
C        CAMERA COORDINATES IN CURRENT DOCKING AID REFERENCE FRAME
      REAL ROTLOS
C        COMPENSATING ROTATION ANGLE ABOUT LINE OF SIGHT
      REAL U(3),V(3)
C        LAMP IMAGE CENTROIDS
      REAL V1(3)
C        INTERMEDIATE RESULTS   CROSS PRODUCT COMPONENTS
      REAL V2(3),V3(3)
C        INTERMEDIATE RESULTS
C
C     (* COMPUTE COMPONENTS OF CROSS PRODUCT TO DEFINE ROTATION AXIS FROM
C        RELPOS TO CAMPOS *)
         V1(1)=-CAMPOS(2)*RELPOS(3)+CAMPOS(3)*RELPOS(2)
         V1(2)=-CAMPOS(3)*RELPOS(1)+CAMPOS(1)*RELPOS(3)
         V1(3)=-CAMPOS(1)*RELPOS(2)+CAMPOS(2)*RELPOS(1)
         PRDMAG=SQR1(V1(1)**2+V1(2)**2+V1(3)**2)
         DOTPRD=CAMPOS(1)*RELPOS(1)+CAMPOS(2)*RELPOS(2)+CAMPOS(3)*
     A      RELPOS(3)
         IF(PRDMAG GT 0 ) GO TO 20
C           (* VECTORS ARE PARALLEL OR ANTIPARALLEL  FIND OUT WHICH *)
               QT1(1)=0
               QT1(2)=0
               IF(DOTPRD GT 0 ) GO TO 10
                  QT1(3)=1 0
                  QT1(4)=0 0
                  GO TO 30
   10          CONTINUE
                  QT1(3)=0
                  QT1(4)=1 0
                  GO TO 30
   20    CONTINUE
C           (* COMPUTE EULER ROTATION ANGLE *)
               PHI=ATAN2(PRDMAG,DOTPRD)
C           (* COMPUTE QUATERNION COMPONENTS *)
               CALL MSCL(QT1,V1,3,1,SIN(PHI* 5)/PRDMAG)
               QT1(4)=COS(PHI* 5)
   30 CONTINUE
      IF(U(3) EQ U(1) AND V(3) EQ V(1)) RETURN
C     (* CORRECT FOR ROTATION ABOUT LINE OF SIGHT *)
C        (* 901 - CONVERT QT1 TO DIRECTION COSINE MATRIX *)
            CALL DIRMAT(QT1,AT,TRNAT)
C        (* FIND ROTATION OF LINE IN -Y DIRECTION ON TARGET, AS PREDICTED
C           BY 'AT', AS COMPARED TO LINE FROM LIGHT 1 TO LIGHT 3 *)
            V2(1)=-TRNAT(1,2)
            V2(2)=-TRNAT(2,2)
            V2(3)=-TRNAT(3,2)
            CALL MMLT(V3,ACV,V2,3,3,1)
            ROTLOS=ATAN2(V3(3),V3(2))-ATAN2(V(3)-V(1),U(1)-U(3))
C        (* FORM QUATERNION FOR CORRECTING ROTATION *)
            CALL MSCL(QC,RELPOS,3,1,SIN( 5*ROTLOS)/SQRT(RELPOS(1)**2+
     A         RELPOS(2)**2+RELPOS(3)**2))
            QC(4)=COS( 5*ROTLOS)
C        (* COMPUTE NET ROTATION OF QT1 FOLLOWED BY QC *)
            QT(1)= QC(4)*QT1(1)+QC(3)*QT1(2)-QC(2)*QT1(3)+QC(1)*QT1(4)
            QT(2)=-QC(3)*QT1(1)+QC(4)*QT1(2)+QC(1)*QT1(3)+QC(2)*QT1(4)
            QT(3)= QC(2)*QT1(1)-QC(1)*QT1(2)+QC(4)*QT1(3)+QC(3)*QT1(4)
            QT(4)=-QC(1)*QT1(1)-QC(2)*QT1(2)-QC(3)*QT1(3)+QC(4)*QT1(4)
      RETURN
C
      END
```

```
C
      SUBROUTINE INCORP(CVPOS P,ESTATE,ACV,ACVT)
C     (* 450 - INCORPORATE MEASUREMENT *)
C
C     CALLS
C        UPDCOV,UPDSTA,COMPG,ESTCOV,KALGAN,MINV,MSUB,MMLT
C     CALLED BY
C        DOCK
C
C     INPUT
C        ESTATE,P,R
C     OUTPUT
C        P,ESTATE
C
      REAL ACV(3,3),ACVT(3,3)
C        DIRECTION COSINE MATRIX MEASURED BY IMU, GIVING CHASE VEHICLE
C        ATTITUDE, AND THE TRANSPOSE OF THIS MATRIX
      REAL CVPOS(3)
C        MEASURED CHASE VEHICLE POSITON IN PRIMARY REFERENCE FRAME
      INTEGER ERR
C        ERROR CODE (=0 IF OKAY)
      REAL ESTATE(6)
C        ESTIMATE OF STATE VECTOR
      REAL G(3,6)
C        PARTIAL OF MEASUREMENT WITH RESPECT TO STATE
      REAL KGAIN(6,3)
C        KALMAN GAIN MATRIX
      INTEGER I,J
C        DO LOOP INDICES
      REAL P(6,6),PA(3,3),PAI(3,3)
C        COVARIANCE OF STATE ESTIMATE, ITS UPPER LEFT 3X3 SUBMATRIX, AND
C        THE INVERSE OF THAT SUBMATRIX
      REAL R(3,3)
C        MEASUREMENT COVARIANCE
      REAL T1(3),E(3),C,SCRACH(4,6)
C        TEMPORARY VARIABLES
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* CHECK FOR RIDICULOUS VALUE *)
      DO 10 I=1,3
         DO 10 J=1,3
            PA(I,J)=P(I,J)
   10 CONTINUE
      CALL MINV(PAI,PA,3,SCRACH,4,6,ERR)
      IF(ERR NE 0) GO TO 30
      CALL MSUB(E,CVPOS,ESTATE,3,1)
      CALL MMLT(T1,PAI,E,3,3,1)
      CALL MMLT(C,E,T1,1,3,1)
      IF(C LE 160 ) GO TO 20
         RETURN
   20 CONTINUE
C     (* 451 - COMPUTE JACOBIAN G *)
      CALL COMPG(ESTATE,G)
C     (* 452 - ESTIMATE MEASUREMENT COVARIANCE *)
      CALL ESTCOV(ESTATE,R,ACV,ACVT)
C     (* 453 - COMPUTE KALMAN GAIN MATRIX *)
      CALL KALGAN(R,P,G,KGAIN)
C     (* 454 - UPDATE STATE *)
      CALL UPDSTA(ESTATE,KGAIN,CVPOS)
C     (* 455 - UPDATE COVARIANCE MATRIX *)
      CALL UPDCOV(P,KGAIN,G)
      RETURN
C
   30 CONTINUE
C     WRITE(1,901)
      STOP
C
  901 FORMAT('P IS SINGULAR IN ''INCORP''')
C
      END
```

```
C
      SUBROUTINE COMPG(ESTATE,G)
C     (* 451 - COMPUTE PARTIAL OF MEASUREMENT WITH RESPECT TO STATE *)
C
C     CALLS
C        (NONE)
C     CALLED BY
C        INCORP
C
C     INPUT
C        ESTATE
C     OUTPUT
C        G
C
      REAL ESTATE(6)
C        STATE ESTIMATE
      REAL G(3,6)
C        JACOBIAN
      INTEGER I,J
C        DO LOOP INDICES
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* COMPUTE G *)
      DO 10 I=1,3
         DO 10 J=1,6
            G(I,J)=0
   10 CONTINUE
      DO 20 I=1,3
         G(I,I)=1
   20 CONTINUE
      RETURN
C
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
      SUBROUTINE ESTCOV(ESTATE,R,ACV,ACVT)
C     (* 452 - ESTIMATE MEASUREMENT COVARIANCE *)
C
C     CALLS
C        SQR1,DPRD,MMLT
C     CALLED BY
C        INCORP
C
C     INPUT
C        ESTATE
C     OUTPUT
C        R
C
      REAL ACV(3,3),ACVT(3,3)
C        DIRECTON COSINE MATRIX MEASURED BY IMU, GIVING CHASE VEHICLE
C        ATTITUDE, AND THE TRANSPOSE OF THIS MATRIX
      REAL ESTATE(6)
C        ESTIMATE OF STATE VECTOR
      INTEGER I,J
C        DO LOOP INDICES
      REAL R(3,3)
C        MEASUREMENT COVARIANCE
      REAL RANGE
C        ESTIMATED RANGE TO TARGET SPACECRAFT
      REAL R1(3,3)
C        INTERMEDIATE RESULTS
C- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
      DO 10 I=1,3
         DO 10 J=1,3
            R(J,I)=0 0
   10 CONTINUE
      RANGE=SQR1(DPRD(ESTATE,ESTATE,3))
      R(1,1)=8 E-7*(RANGE-5)**4+ 005
      R(2,2)=7 36E-7*RANGE**3+ 0016
      R(3,3)=R(2,2)
      CALL MMLT(R1,ACVT,R,3,3,3)
      CALL MMLT(R,R1,ACV,3,3,3)
      RETURN
C
      END
```

```
C
      SUBROUTINE KALGAN(R,P,G,KGAIN)
C     (* 453 - COMPUTE KALMAN GAIN MATRIX *)
C
C     CALLS
C        MADD,MINV,MMLT
C     CALLED BY
C        INCORP
C
C     INPUT
C        R,P,G
C     OUTPUT
C        KGAIN
C
      INTEGER ERR
C        ERROR CODE (=0 IF NO ERROR)
      REAL G(3,6)
C        PARTIAL OF MEASUREMENT WITH RESPECT TO STATE
      REAL GT(6,3)
C        TRANSPOSE OF G
      INTEGER I,J
C        DO LOOP INDICES
      REAL KGAIN(6,3)
C        KALMAN GAIN MATRIX
      REAL P(6,6)
C        STATE ESTIMATE COVARIANCE
      REAL R(3,3)
C        MEASUREMENT COVARIANCE
      REAL SCRACH(4,6)
C        SCRATCHPAD FOR ROUTINE MINV
      REAL TEMP1(6,3),TEMP2(3,3),TEMP3(3,3)
C        INTERMEDIATE RESULTS WITH NO PROBLEM-RELATED SIGNIFICANCE
      INTEGER TERMNL
C        FORTRAN UNIT NUMBER FOR TERMINAL
C
      COMMON/SIMUL/TERMNL,IDUMMY
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* COMPUTE KGAIN=P*TRN(G)*INV(R+G*P*TRN(G)) *)
         DO 10 I=1,3
            DO 10 J=1,6
               GT(J,I)=G(I,J)
   10    CONTINUE
         CALL MMLT(TEMP1,P,GT,6,6,3)
         CALL MMLT(TEMP2,G,TEMP1,3,6,3)
         CALL MADD(TEMP3,R,TEMP2,3,3)
         CALL MINV(TEMP2,TEMP3,3,SCRACH,4,6,ERR)
         IF(ERR NE 0) GO TO 20
         CALL MMLT(TEMP1,GT,TEMP2,6,3,3)
         CALL MMLT(KGAIN,P,TEMP1,6,6,3)
         RETURN
C
   20 CONTINUE
C     (* REPORT ERROR *)
         WRITE(TERMNL,901)
         STOP
C
  901 FORMAT('MATRIX INVERSION FAILURE IN SUBROUTINE KALGAN')
C
      END
```

```
C
      SUBROUTINE UPDSTA(ESTATE,KGAIN,CVPOS)
C     (* 454 - UPDATE STATE ESTIMATE *)
C
C     CALLS
C        MMLT
C     CALLED BY
C        INCORP
C
C     INPUT
C        RELPOS,AT,ESTATE
C     OUTPUT
C        ESTATE
C
      REAL CVPOS(3)
C        MEASURED CHASE VEHICLE POSITION IN PRIMARY REFERENCE FRAME
      REAL DELTA(6)
C        ADJUSTMENT TO ESTATE
      REAL ESTATE(6)
C        ESTIMATE OF STATE VECTOR
      INTEGER I
C        DO LOOP INDEX
      REAL KGAIN(6,3)
C        KALMAN GAIN MATRIX
      REAL POSERR(3)
C        MEASURED POSITION MINUS PREDICTED POSITION
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* COMPUTE ESTATE=ESTATE+KGAIN*(CVPOS-PREDICTED POSITION) *)
         DO 10 I=1,3
            POSERR(I)=CVPOS(I)-ESTATE(I)
   10    CONTINUE
         CALL MMLT(DELTA,KGAIN,POSERR,6,3,1)
         DO 20 I=1,6
            ESTATE(I)=ESTATE(I)+DELTA(I)
   20    CONTINUE
      RETURN
C
      END
```

```
C
      SUBROUTINE UPDCOV(P,KGAIN,C)
C     (* 455 - UPDATE COVARIANCE MATRIX FROM (P=(I-K*G)*P)  *)
C
C     CALLS
C        MMLT
C     CALLED BY
C        INCORP
C
C     INPUT
C        P,KGAIN,G
C     OUTPUT
C        P
C
      REAL G(3,6)
C        PARTIAL OF MEASUREMENT WITH RESPECT TO STATE
      INTEGER I,J
C        DO LOOP INDICES
      REAL KGAIN(6,3)
C        KALMAN GAIN MATRIX
      REAL P(6,6)
C        COVARIANCE OF STATE ESTIMATE
      REAL TEMP(6,6),P1(6,6)
C        INTERMEDIATE RESULTS WITH NO PROBLEM-RELATED SIGNIFICANCE
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* COMPUTE I-KGAIN*G *)
         CALL MMLT(TEMP,KGAIN,G,6,3,6)
         DO 10 I=1,6
            DO 10 J=1,6
               TEMP(I,J)=-TEMP(I,J)
   10    CONTINUE
         DO 20 I=1,6
            TEMP(I,I)=TEMP(I,I)+1.0
   20    CONTINUE
C     (* COMPUTE NEW P *)
         CALL MMLT(P1,TEMP,P,6,6,6)
         DO 30 I=1,6
            DO 30 J=1,6
               P(I,J)=P1(I,J)
   30    CONTINUE
      RETURN
C
      END
```

```
C
      SUBROUTINE ESTCOV(ESTATE,R,ACV,ACVT)
C     (* 452 - ESTIMATE MEASUREMENT COVARIANCE *)
C
C     CALLS
C        SQR1,DPRD,MMLT
C     CALLED BY
C        INCORP
C
C     INPUT
C        ESTATE
C     OUTPUT
C        R
C
      REAL ACV(3,3),ACVT(3,3)
C        DIRECTON COSINE MATRIX MEASURED BY IMU, GIVING CHASE VEHICLE
C        ATTITUDE, AND THE TRANSPOSE OF THIS MATRIX
      REAL ESTATE(6)
C        ESTIMATE OF STATE VECTOR
      INTEGER I,J
C        DO LOOP INDICES
      REAL R(3,3)
C        MEASUREMENT COVARIANCE
      REAL RANGE
C        ESTIMATED RANGE TO TARGET SPACECRAFT
      REAL R1(3,3)
C        INTERMEDIATE RESULTS
C- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
      DO 10 I=1,3
        DO 10 J=1,3
          R(J,I)=0 0
   10 CONTINUE
      RANGE=SQR1(DPRD(ESTATE,ESTATE,3))
      R(1,1)=8 E-7*(RANGE-5)**4+ 005
      R(2,2)=7 36E-7*RANGE**3+ 0016
      R(3,3)=R(2,2)
      CALL MMLT(R1,ACVT,R,3,3,3)
      CALL MMLT(R,R1,ACV,3,3,3)
      RETURN
C
      END
```

```
C
      SUBROUTINE KALGAN(R,P,G,KGAIN)
C     (* 453 - COMPUTE KALMAN GAIN MATRIX *)
C
C     CALLS
C        MADD,MINV,MMLT
C     CALLED BY
C        INCORP
C
C     INPUT
C        R,P,G
C     OUTPUT
C        KGAIN
C
      INTEGER ERR
C        ERROR CODE (=0 IF NO ERROR)
      REAL G(3,6)
C        PARTIAL OF MEASUREMENT WITH RESPECT TO STATE
      REAL GT(6,3)
C        TRANSPOSE OF G
      INTEGER I,J
C        DO LOOP INDICES
      REAL KGAIN(6,3)
C        KALMAN GAIN MATRIX
      REAL P(6,6)
C        STATE ESTIMATE COVARIANCE
      REAL R(3,3)
C        MEASUREMENT COVARIANCE
      REAL SCRACH(4,6)
C        SCRATCHPAD FOR ROUTINE MINV
      REAL TEMP1(6,3),TEMP2(3,3),TEMP3(3,3)
C        INTERMEDIATE RESULTS WITH NO PROBLEM-RELATED SIGNIFICANCE
      INTEGER TERMNL
C        FORTRAN UNIT NUMBER FOR TERMINAL
C
      COMMON/SIMUL/TERMNL,IDUMMY
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* COMPUTE KGAIN=P*TRN(G)*INV(R+G*P*TRN(G)) *)
         DO 10 I=1,3
           DO 10 J=1,6
             GT(J,I)=G(I,J)
   10    CONTINUE
         CALL MMLT(TEMP1,P,GT,6,6,3)
         CALL MMLT(TEMP2,G,TEMP1,3,6,3)
         CALL MADD(TEMP3,R,TEMP2,3,3)
         CALL MINV(TEMP2,TEMP3,3,SCRACH,4,6,ERR)
         IF(ERR NE 0) GO TO 20
         CALL MMLT(TEMP1,GT,TEMP2,6,3,3)
         CALL MMLT(KGAIN,P,TEMP1,6,6,3)
         RETURN
C
   20 CONTINUE
C     (* REPORT ERROR *)
         WRITE(TERMNL,901)
         STOP
C
  901 FORMAT('MATRIX INVERSION FAILURE IN SUBROUTINE KALGAN')
C
      END
```

```
C
      SUBROUTINE UPDSTA(ESTATE,KGAIN,CVPOS)
C     (* 454 - UPDATE STATE ESTIMATE *)
C
C     CALLS
C        MMLT
C     CALLED BY
C        INCORP
C
C     INPUT
C        RELPOS,AT,ESTATE
C     OUTPUT
C        ESTATE
C
      REAL CVPOS(3)
C        MEASURED CHASE VEHICLE POSITION IN PRIMARY REFERENCE FRAME
      REAL DELTA(6)
C        ADJUSTMENT TO ESTATE
      REAL ESTATE(6)
C        ESTIMATE OF STATE VECTOR
      INTEGER I
C        DO LOOP INDEX
      REAL KGAIN(6,3)
C        KALMAN GAIN MATRIX
      REAL POSERR(3)
C        MEASURED POSITION MINUS PREDICTED POSITION
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C
C     (* COMPUTE ESTATE=ESTATE+KGAIN*(CVPOS-PREDICTED POSITION) *)
         DO 10 I=1,3
            POSERR(I)=CVPOS(I)-ESTATE(I)
   10    CONTINUE
         CALL MMLT(DELTA,KGAIN,POSERR,6,3,1)
         DO 2C I=1,6
            ESTATE(I)=ESTATE(I)+DELTA(I)
   20    CONTINUE
      RETURN
C
      END
```

```
C
      SUBROUTINE UPDCOV(P,KGAIN,G)
C     (* 455 - UPDATE COVARIANCE MATRIX FROM (P=(I-K*G)*P)   *)
C
C     CALLS
C        MMLT
C     CALLED BY
C        INCORP
C
C     INPUT
C        P,KGAIN,G
C     OUTPUT
C        P
C
      REAL G(3,6)
C        PARTIAL OF MEASUREMENT WITH RESPECT TO STATE
      INTEGER I,J
C        DO LOOP INDICES
      REAL KGAIN(6,3)
C        KALMAN GAIN MATRIX
      REAL P(6,6)
C        COVARIANCE OF STATE ESTIMATE
      REAL TEMP(6,6),P1(6,6)
C        INTERMEDIATE RESULTS WITH NO PROBLEM-RELATED SIGNIFICANCE
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C
      (* COMPUTE I-KGAIN*G *)
         CALL MMLT(TEMP,KGAIN,G,6,3,6)
         DO 10 I=1,6
            DO 10 J=1,6
               TEMP(I,J)=-TEMP(I,J)
   10    CONTINUE
         DO 20 I=1,6
            TEMP(I,I)=TEMP(I,I)+1.0
   20    CONTINUE
C     (* COMPUTE NEW P *)
         CALL MMLT(P1,TEMP,P,6,6,6)
         DO 30 I=1,6
            DO 30 J=1,6
               P(I,J)=P1(I,J)
   30    CONTINUE
      RETURN
C
      END
```

```
C
      SUBROUTINE PROPES(P,ESTATE,F,STEP,ACVT)
C     (* 460 - PROPAGATE STATE ESTIMATE AND COVARIANCE *)
C
C     CALLS
C        MMLT,MSCL,MIDN,MTRN
C     CALLED BY
C        DOCK
C
C     INPUT
C        P,ESTATE,ACVT,F,STEP
C     OUTPUT
C        P,ESTATE
C
      REAL ACCEL(3)
C        ESTIMATED ACCELERATION
      REAL ACV(3,3),ACVT(3,3)
C        CHASE VEHICLE DIRECTION COSINE MATRIX AND ITS TRANSPOSE
      REAL AK2(3,14)
C        MATRIX RELATING FORCES FROM INDIVIDUAL THRUSTERS TO NET FORCE VECTOR
      REAL AVARX,AVARY,AVARZ
C        ESTIMATED ACCELERATION VARIANCE
      REAL DSTATE(6)
C        CHANGE IN STATE
      REAL DP(6,6)
C        CHANGE IN COVARIANCE
      REAL DPP(3,3),DPP1(3,3)
C        INTERMEDIATE RESULTS
      REAL ESTATE(6)
C        ESTIMATED STATE
      REAL F(14)
C        FORCES FROM THRUSTERS AFTER SELECTION
      REAL FBOD(3)
C        FORCES ALONG CHASE VEHICLE BODY AXES
      REAL FORCE(3)
C        FORCES IN PRIMARY REFERENCE FRAME
      INTEGER I,J
C        DO LOOP INDICES
      REAL P(6,6)
C        COVARIANCE OF STATE ESTIMATE
      REAL STEP
C        INTEGRATION TIME STEP
      PARAMETER AVGMAS=3650
C        ESTIMATED CHASE VEHICLE MASS
C
      COMMON/VEHIC/DUMMY1(14),AK2,DUMMY2(57)
C
C     (* COMPUTE FORCE=ACVT*AK2*F *)
      CALL MMLT(FBOD,AK2,F,3,14,1)
      CALL MMLT(FORCE,ACVT,FBOD,3,3,1)
C     (* ESTIMATE ACCELERATION *)
      CALL MSCL(ACCEL,FORCE,3,1,1./AVGMAS)
C     (* ESTIMATE ACCELERATION VARIANCE *)
      AVARX=(ACCEL(1)*.05)**2
      AVARY=(ACCEL(2)*.05)**2
      AVARZ=(ACCEL(3)*.05)**2
C     (* FIND POSITION CHANGE *)
      DO 10 I=1,3
         DSTATE(I)=(ESTATE(I+3)+.5*ACCEL(I)*STEP)*STEP
   10 CONTINUE
C     (* FIND VELOCITY CHANGE *)
      DO 20 I=4,6
         DSTATE(I)=STEP*ACCEL(I-3)
   20 CONTINUE
C     (* FIND COVARIANCE CHANGE *)
      DO 30 I=4,6
         DO 30 J=1,6
            DP(I,J)=0.0
   30 CONTINUE
      CALL MIDN(DPP,3)
      DPP(1,1)=AVARX
      DPP(2,2)=AVARY
      DPP(3,3)=AVARZ
      CALL MTRN(ACV,ACVT,3)
      CALL MMLT(DPP1,ACVT,DPP,3,3,3)
      CALL MMLT(DPP,DPP1,ACV,3,3,3)
      DO 40 I = 1,3
         DO 40 J = 1,3
            DP(I+3,J+3) = DPP(I,J)
            IF(I.EQ.J) DP(I+3,J+3)=DP(I+3,J+3)+.00002
   40 CONTINUE
      DO 50 I=1,3
         DO 50 J=1,6
            DP(I,J)=P(I+3,J)
   50 CONTINUE
      DO 60 I=1,6
         DO 60 J=1,3
            DP(I,J)=DP(I,J)+P(I,J+3)
   60 CONTINUE
C     (* COMPUTE NEW STATE AND COVARIANCE *)
      DO 70 I=1,6
         ESTATE(I)=ESTATE(I)+DSTATE(I)
         DO 70 J=1,6
            P(I,J)=P(I,J)+DP(I,J)*STEP
   70 CONTINUE
     RETURN
C
     END
```

```
C
      SUBROUTINE SETGOL(ESTATE,P,T,ACV,ESTA,PA,GXL,GXH,GYL,GYH,GZL,
     A   GZH,DEDLIN)
C     (* 470 - SET GOAL *)
C
C     CALLS
C        SQR1,AMAX1,MMLT,MSUB,DIRMAT,SQRT,DPRD,ABS,AMIN1,QMLT,MSCL,MADD
C     CALLED BY
C        DOCK
C
C     INPUT
C        ESTATE,P,DOKRAD,T,ACV,ESTA,PA
C     OUTPUT
C        GXL,GXH,GYL,GYH,GZL,GZH,DEDLIN
C
      REAL ACV(3,3),AT(3,3),TRNAT(3,3)
C        DIRECTION COSINE MATRICES FOR CHASE VEHICLE & TARGET SPACECRAFT
C        AND TRANSPOSE OF TARGET DIRECTION COSINE MATRIX
      REAL D,D1
C        ALLOWANCE FOR POSITION UNCERTAINTY AND MISALIGNMENT
      REAL DEDLIN
C        TIME BY WHICH STRATEGY LOGIC WANTS CHASE VEHICLE IN GOAL 'BOX'
      REAL DOKRAD
C        RADIUS FROM TARGET SPACECRAFT WITHIN WHICH THE SPACECRAFT ARE
C        CONSIDERED DOCKED
      REAL ESTATE(6),ESTA(7)
C        TRANSLATION AND TARGET ATTITUDE STATE ESTIMATE VECTORS
      REAL GXL,GXH,GYL,GYH,GZL,GZH
C        X,Y,Z LOWER AND UPPER LIMITS OF GOAL 'BOX'
      REAL HDC(3),HDT(3)
C        DOCKING FIXTURES' POSITIONS IN THEIR RESPECTIVE SPACECRAFT
C        COORDINATE SYSTEMS
      INTEGER I,J
C        DO LOOP INDICES
      REAL ININV(3,3)
C        INVERSE OF TARGET MOMENT OF INERTIA TENSOR
      REAL K1(4)
C        INTERMEDIATE RESULT
      REAL P(6,6),PA(7,7)
C        COVARIANCE OF STATE ESTIMATE FOR TRANSLATION AND TARGET ATTITUDE
      REAL QNEW(4)
C        PREDICTED ATTITUDE OF TARGET (QUATERNION)
      REAL RANGE
C        APPROXIMATE DOCKING FIXTURE SEPARATION
      REAL SLOP
C        ALLOWED ERROR IN POSITION (PER AXIS)
      REAL T
C        ELAPSED TIME
      REAL T1(3),T2(4),TP,R1,RM
C        INTERMEDIATE RESULTS
      REAL V1(3),V2(3),V3(3),R(3),V4(3)
C        INTERMEDIATE RESULTS
C
      COMMON/VEHIC/DUMMY1(98),DOKRAD,DUMMY2(14)
      COMMON/TGTI/ININV
      COMMON/HNGOFF/DUMMY3(6),HDC,HDT
C
C     (* PREDICT TARGET ATTITUDE *)
         TP=1.616667
         RANGE=SQRT(DPRD(ESTATE,ESTATE,3))-ABS(HDC(1))-ABS(HDC(1))
         IF(RANGE.GT.0.) TP=AMIN1(TP+.2*RANGE,20.)
         CALL DIRMAT(ESTA,AT,TRNAT)
         CALL MMLT(T1,AT,ESTA(5),3,3,1)
         CALL MMLT(T2,ININV,T1,3,3,1)
         T2(4)=0
         CALL QMLT(K1,ESTA,T2)
         CALL MSCL(K1,K1,4,1,TP/2.)
         CALL MADD(QNEW,ESTA,K1,4,1)
         CALL MSCL(QNEW,QNEW,4,1,1.0/SQRT(DPRD(QNEW,QNEW,4)))
         CALL DIRMAT(QNEW,AT,TRNAT)
      IF(SQR1(DPRD(ESTATE,ESTATE,3)).LT.12.) GO TO 10
         D=2.*AMIN1(SQR1(AMAX1(P(1,1),P(2,2),P(3,3))),100.)
         GO TO 20
   10 CONTINUE
         D=0
   20 CONTINUE
C     (* ADD EXTRA MARGIN FOR MISALIGNMENT *)
         D1=10.*(1.-AT(1,1)*ACV(1,1)-AT(1,2)*ACV(1,2)-AT(1,3)*ACV(1,3))
         IF(D1.GT.0.5) D=D+D1-.5
C     (* COMPUTE AIM POINT ON TARGET X AXIS *)
         R(1)=-D+HDT(1)
         R(2)=0.+HDT(2)
         R(3)=0.+HDT(3)
         CALL MSUB(V1,R,HDC,3,1)
         CALL MMLT(V2,TRNAT,V1,3,3,1)
         CALL MSUB(V3,V2,ESTATE,3,1)
         CALL MMLT(V4,ACV,V3,3,3,1)
C     (* MODIFY GOAL IF PATH GOES TOO CLOSE TO TARGET *)
         R1=SQR1(DPRD(ESTATE,ESTATE,3))
         RM=D+ABS(HDC(1))+ABS(HDT(1))
         IF(R1.GE.RM) GO TO 30
            CALL MSCL(V4,ESTATE,3,1,RM/R1)
            GO TO 40
   30    CONTINUE
            CALL MODGOL(ACV,ESTATE,V4,RM)
   40    CONTINUE
      SLOP=2
      IF(R1.LT.20.) SLOP=.1
      GXH=V4(1)+SLOP
      GYH=V4(2)+SLOP
      GZH=V4(3)+SLOP
      GXL=V4(1)-SLOP
      GYL=V4(2)-SLOP
      GZL=V4(3)-SLOP
      DEDLIN=T+.125*SQRT(V4(1)**2+V4(2)**2+V4(3)**2)
      RETURN
C
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
      SUBROUTINE MODGOL(ACV,ESTATE,V4,D)
C     (* 471 - ADJUST GOAL IF PATH TOO CLOSE TO TARGET SPACECRAFT *)
C
C     CALLS
C        MSCL,DPRD,SQR1,MSUB
C     CALLED BY
C        SETGOL
C
C     INPUT
C        V4,ESTATE,ACV
C     OUTPUT
C        V4
C
      REAL ACV(3,3)
C        DIRECTION COSINE MATRIX SPECIFYING CHASE VEHICLE ATTITUDE
      REAL D
C        SAFETY MARGIN THAT ACCOUNTS FOR UNCERTAINTY AND ATTITUDE MISALIGNMENT
      REAL ESTATE(3)
C        ESTIMATED CHASE VEHICLE POSITION
      REAL K
C        DISTANCE TO TARGET RELATIVE TO DISTANCE TO GOAL
      REAL V4(3)
C        GOAL POSITION
      REAL R(3),RMAG,VMAGSQ,V1(3),RNEW(3)
C        INTERMEDIATE RESULTS
C
      CALL MMLT(V1,ACV,ESTATE,3,3,1)
      CALL MADD(R,V1,V4,3,1)
      VMAGSQ=DPRD(V4,V4,3)
      IF(VMAGSQ EQ 0 0) GO TO 10
         K=-DPRD(V4,V1,3)/VMAGSQ
         IF(K GE 1 0 OR K LE 0 0) GO TO 10
            CALL MSCL(V4,V4,3,1,K)
            CALL MADD(RNEW,V4,V1,3,1)
            RMAG=SQR1(DPRD(RNEW,RNEW,3))
            CALL MSCL(RNEW,RNEW,3,1,D/RMAG)
            CALL MSUB(V4,RNEW,V1,3,1)
            CALL MSCL(V4,V4,3,1,20 /SQR1(DPRD(V4,V4,3)))
   10 CONTINUE
      RETURN
C
      END
```

```
C
      SUBROUTINE THRUST(F,ESTATE,GXL,GXH,GYL,GYH,GZL,GZH,TLIM,ACV
     A   ATRATE,RPYERR)
C     (* 480 -  SELECT THRUSTERS *)
C
C     CALLS
C        CNTLAW,FIRTHR,SELECT
C     CALLED BY
C        DOCK
C
C     INPUT
C        ESTATE,GXL,      GZH,TLIM,ESTATE,ACV,ATRATE,RPYERR
C     OUTPUT
C        F
C
      REAL ACV(3,3)
C        MEASURED CHASE VEHICLE ATTITUDE IN PRIMARY REF FRAME
      REAL ATRATE(3
C        MEASURED ATTITUDE RATES OF CHASE VEHICLE
      REAL E(14)
C        THRUSTER COMMAND ENTRIES FROM THRUSTER SELECT LOGIC
      REAL ESTATE(6)
C        STATE ESTIMATE
      REAL F(14)
C        FORCES FROM THRUSTERS AFTER SELECTION
      REAL F1(14)
C        LIST OF MAXIMUM THRUSTER FORCES
      REAL GXL,GXH,GYL,GYH,GZL,GZH
C        LOWER AND UPPER LIMITS OF GOAL 'BOX'
      INTEGER I
C        DO LOOP INDEX
      REAL ROTCMD(3),XLTCMD(3)
C        ROTATIONAL AND TRANSLATIONAL COMMANDS FROM CONTROL LAW
      REAL RPYERR(3)
C        MEASURED ATTITUDE ERROR IN ROLL, PITCH, AND YAW (RADIANS)
      REAL TLIM
C        TIME LIMIT
C
      COMMON/VEHIC/DUMMY(99),F1
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* 481 - USE CONTROL LAW TO DETERMINE NEEDED THRUST *)
         CALL CNTLAW(ROTCMD,XLTCMD,ESTATE,ACV,TLIM,GXL,GXH,GYL,GYH,GZL
     A      GZH,ATRATE,RPYERR)
C     (* 482 - SELECT THRUSTER SET TO GIVE NEEDED THRUST *)
         CALL SELECT(ROTCMD,XLTCMD,E)
         DO 10 I=1,14
            F(I)=E(I)*F1(I)
   10    CONTINUE
C     (* 483 - FIRE THRUSTERS *)
         CALL FIRTHR(E)
      RETURN
C
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
      SUBROUTINE CNTLAW(ROTCMD, XLTCMD, ESTATE, ACV, TLIM, GXL, GXH, GYL, GYH,
     A GZL, GZH, ATRATE, RPYERR)
C     (* 481  - CONTROL LAW *)
C
C     CALLS
C        MMLT, ACCEL, SQRT, AMIN1
C     CALLED BY
C        THRUST
C
C     INPUT
C        ESTATE, ACV, TLIM, GXL,    , GZH, ATRATE, RPYERR
C     OUTPUT
C        ROTCMD, XLTCMD
C
      REAL ACV(3,3)
C        MEASURED CHASE VEHICLE ATTITUDE IN PRIMARY REFER     AME
      REAL ATRATE(3)
C        MEASURED CHASE VEHICLE ATTITUDE RATES
      REAL ESTATE(6)
C        STATE ESTIMATE
      REAL GXL, GXH, GYL, GYH, GZL, GZH
C        LOWER AND UPPER LIMITS OF GOAL 'BOX'
      REAL ROTCMD(3), XLTCMD(3)
C        ROTATIONAL AND TRANSLATIONAL COMMANDS FROM CONTRO    AW
      REAL RANGE
C        ESTIMATED RANGE TO TARGET
      REAL RPYERR(3)
C        MEASURED ATTITUDE ERROR IN ROLL, PITCH, AND YAW
      REAL SLP
C        TOLERABLE ATTITUDE ERROR ABOUT ROLL AXIS
      REAL TLIM
C        TIME LIMIT
      REAL VBOD(3)
C        VELOCITY IN CHASE VEHICLE BODY COORDINATE SYSTEM
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* CONVERT VELOCITY TO BODY COORD  SYS  *)
         CALL MMLT(VBOD, ACV, ESTATE(4), 3, 3, 1)
C     (* 481 1 - FIND IDEAL ACCELERATIONS *)
         RANGE=SQRT(ESTATE(1)**2+ESTATE(2)**2+ESTATE(3)**2)
         SLP= 02*RANGE/15
         XLTCMD(1)=ACCEL(VBOD(1), GXL, GXH, TLIM, 1 233333, 027, 1)
         XLTCMD(2)=ACCEL(VBOD(2), GYL, GYH, TLIM, 1 233333, 027, 889)
         XLTCMD(3)=ACCEL(VBOD(3), GZL, GZH, TLIM, 1 233333, 027, 889)
         ROTCMD(1)=ACCEL(ATRATE(1), (-RPYERR(1)-SLP), (-RPYERR(1)+SLP),
     A      AMIN1(RANGE/6 ,20 ), 1 233333, 0473, 0975)
         ROTCMD(2)=ACCEL(ATRATE(2), -RPYERR(2)- 02, -RPYERR(2)+ 02,
     A      1 3, 1 233333, 0501, 103)
         ROTCMD(3)=ACCEL(ATRATE(3), -RPYERR(3)- 02, -RPYERR(3)+ 02,
     A      1 3, 1 233333, 0501, 103)
      RETURN
C
      END
```

```
C
      FUNCTION ACCEL(XDOT, XMIN, XMAX, TLIM, DT, AMIN, AMAX)
C     (* 481 1 - ESTIMATE ACCELERATION FOR ONE AXIS *)
C
C     CALLS
C        <NONE>
C     CALLED BY
C        CNTLAW
C
C     INPUT
C        X, XDOT, XMIN, XMAX, TLIM, DT
C     OUTPUT
C        <FUNCTION VALUE ONLY>
C
      REAL AMIN, AMAX
C        MIN AND MAX RELATIVE THRUST
      REAL DT
C        TIME STEP BETWEEN DECISIONS
      REAL K1, K2, K3, K4
C        CONSTANTS BASED ON MIN, MAX POSSIBLE ACCELERATION
      REAL TLIM
C        TIME LIMIT - SECONDS UNTIL DEADLINE
      REAL XDOT
C        VELOCITY IN THIS AXIS
      REAL XMIN, XMAX
C        MIN, MAX ACCEPTABLE FINAL X VALUE
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
      SUBROUTINE SELECT(ROTCMD,XLTCMD,E)
C     (* 482 - FROM CONTROL LAW OUTPUTS SELECT WHICH THRUSTERS TO FIRE *)
C
C     CALLS
C        TABLE1, TABLE2, TABLE3
C     CALLED BY
C        THRUST
C
C     INPUT
C        ROTCMD, XLTCMD
C     OUTPUT
C        E
C
      REAL E(14)
C        THRUSTER COMMAND ENTRIES FROM THRUSTER SELECT LOGIC
      REAL ROTCMD(3),XLTCMD(3)
C        ROTATIONAL AND TRANSLATIONAL COMMANDS FROM CONTROL LAW
      INTEGER I,J
C        DO LOOP INDICES
      INTEGER INDEX1,INDEX2,INDEX3
C        INDICES FOR COMMAND TABLES
      INTEGER ROTCOD(3),TRLCOD(3)
C        ROTATION AND TRANSLATION CODES USED TO COMPUTE INDICES
C        FOR TABLES
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* DETERMINE CODE FOR THRUSTER ACTIVATION *)
         DO 40 I=1,3
            IF(XLTCMD(I)) 10,20,30
   10       TRLCOD(I)=1
            GO TO 40
   20       TRLCOD(I)=0
            GO TO 40
   30       TRLCOD(I)=2
   40    CONTINUE
         DO 80 I=1,3
            IF(ROTCMD(I)) 50,60,70
   50       ROTCOD(I)=1
            GO TO 80
   60       ROTCOD(I)=0
            GO TO 80
   70       ROTCOD(I)=2
   80    CONTINUE
C     (* ZERO OUT THRUSTER COMMANDS *)
         DO 90 J=1,14
            E(J)=0.0
   90    CONTINUE
C     (* DETERMINE WHICH TABLE TO CONSULT FOR THRUSTER COMMANDS *)
         INDEX1=TRLCOD(1)+3*ROTCOD(2)+9*ROTCOD(3)
         INDEX2=TRLCOD(2)+3*ROTCOD(1)+9*ROTCOD(3)
         INDEX3=TRLCOD(3)+3*ROTCOD(1)+9*ROTCOD(2)
C     (* 482.1 THRU 482.3 - SELECT THRUSTER SET *)
         IF(INDEX1 EQ 0) GO TO 100
            CALL TABLE1(INDEX1,E)
  100    CONTINUE
         IF(INDEX2 EQ 0) GO TO 110
            CALL TABLE2(INDEX2,E)
  110    CONTINUE
         IF(INDEX3 EQ 0) GO TO 120
            CALL TABLE3(INDEX3,E)
  120    CONTINUE
      RETURN
C
      END

C
C
         K1= 5*AMAX*DT**2
         K2=AMAX*DT
         K3= 5/AMIN
         K4=AMIN* 5
         IF(XMAX GE 0 ) GO TO 50
            IF(XDOT LT 0 ) GO TO 10
               ACCEL=-1
               RETURN
   10       IF(XDOT*DT-K2*XDOT**2 GE XMIN) GO TO 20
               ACCEL=1
               RETURN
   20       IF(XDOT*DT-K1-K3*(K2-XDOT)**2 GE XMIN) GO TO 30
               ACCEL=0
               RETURN
   30       IF(XDOT*TLIM-K4*(TLIM-DT)**2 GT XMAX) GO TO 40
               ACCEL=0
               RETURN
   40       CONTINUE
               ACCEL=-1
               RETURN
   50    IF(XMIN LE 0 ) GO TO 100
            IF(XDOT GT 0 ) GO TO 60
               ACCEL=1
               RETURN
   60       IF(XDOT*DT+K3*XDOT**2 LE XMAX) GO TO 70
               ACCEL=-1
               RETURN
   70       IF(XDOT*DT+K1+K3*(K2+XDOT)**2 LE XMAX) GO TO 80
               ACCEL=0
               RETURN
   80       IF(XDOT*DT+K4*(TLIM-DT)**2 LT XMIN) GO TO 90
               ACCEL=0
               RETURN
   90       CONTINUE
               ACCEL=1
               RETURN
  100    IF(XDOT LE 0 ) GO TO 120
            IF(XDOT*DT+K3*XDOT**2 LE XMAX) GO TO 110
               ACCEL=-1
               RETURN
  110       CONTINUE
               ACCEL=0
               RETURN
  120    IF(XDOT*DT-K3*XDOT**2 GE XMIN) GO TO 130
            ACCEL=1
            RETURN
  130    CONTINUE
            ACCEL=0
            RETURN
C
         END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
      SUBROUTINE TABLE2(INDEX2,E)
C     (* 482 2 -  FIND COMINATION OF TABLE ENTRIES THAT SATISFY THRUSTER
C         SELECTION REQUIREMENTS *)
C
C     CALLS
C         <NONE>
C     CALLED BY
C         SELECT
C
C     INPUT
C         INDEX2
C     OUTPUT
C         E
C
      REAL E(14)
C         THRUSTER COMMAND ENTRIES FROM THRUSTER SELECT LOGIC
      INTEGER INDEX2
C         THRUSTER TABLE INDEX
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* DO SELECTION BASED ON INDEX *)
      GOTO(10,20,30,40,50,60,70,80,90,100,110,120,130,140,150,160,170,
     A    180,190,200,210,220,230,240,250,260), INDEX2
   10 E(4)=1 0
      E(5)=1 0
      GO TO 300
   20 E(2)=1 0
      E(3)=1 0
      GO TO 300
   30 E(7)=1 0
      E(8)=1 0
      GO TO 300
   40 E(4)=1 0
      E(5)=1 0
      E(7)=1 0
      E(8)=1 0
      GO TO 300
   50 E(2)=1 0
      E(3)=1 0
      E(7)=1 0
      E(8)=1 0
      GO TO 300
   60 E(6)=1 0
      E(9)=1 0
      GO TO 300
   70 E(4)=1 0
      E(5)=1 0
      E(6)=1 0
      E(9)=1 0
      GO TO 300
   80 E(2)=1 0
      E(3)=1 0
      E(6)=1 0
      E(9)=1 0
      GO TO 300
   90 E(2)=1 0
      E(4)=1 0
      GO TO 300
  100 E(2)=1 0
      E(4)=1 0
      E(6)=1 0
      E(8)=1 0
      GO TO 300
  110 E(2)=1 0
      E(4)=1 0
      E(7)=1 0
      E(9)=1 0
      GO TO 300
  120 E(2)=1 0
      E(4)=1 0
      E(7)=1 0
      E(8)=1 0
      GO TO 300
  130 E(4)=1 0
      E(8)=1 0
      GO TO 300
  140 E(2)=1 0
      E(7)=1 0
      GO TO 300
  150 E(2)=1 0
      E(4)=1 0
      E(6)=1 0
      E(9)=1 0
      GO TO 300
  160 E(4)=1 0
      E(6)=1 0
      GO TO 300
  170 E(2)=1 0
      E(9)=1 0
      GO TO 300
  180 E(3)=1 0
      E(5)=1 0
      GO TO 300
  190 E(3)=1 0
      E(5)=1 0
      E(6)=1 0
      E(8)=1 0
      GO TO 300
  200 E(3)=1 0
      E(5)=1 0
      E(7)=1 0
      E(9)=1 0
      GO TO 300
  210 E(3)=1 0
      E(5)=1 0
      E(7)=1 0
      E(8)=1 0
      GO TO 300
  220 E(5)=1 0
      E(8)=1 0
      GO TO 300
  230 E(3)=1 0
      E(7)=1 0
      GO TO 300
  240 E(3)=1 0
      E(5)=1 0
      E(6)=1 0
      E(9)=1 0
      GO TO 300
  250 E(5)=1 0
      E(6)=1 0
      GO TO 300
  260 E(3)=1 0
      E(9)=1 0
  300 RETURN
C
      END
```

```
C
      SUBROUTINE TABLE1(INDEX1,E)
C     (* 482.1 -  FIND COMINATION OF TABLE ENTRIES THAT SATISFY THRUSTER
C        SELECTION REQUIREMENTS *)
C
C     CALLS
C        <NONE>
C     CALLED BY
C        SELECT
C
C     INPUT
C        INDEX1
C     OUTPUT
C        E
C
      REAL E(14)
C        THRUSTER COMMAND ENTRIES FROM THRUSTER SELECT LOGIC
      INTEGER INDEX1
C        THRUSTER TABLE INDEX
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* DO SELECTION BASED ON INDEX *)
      GOTO(10,20,30,40,50,60,70,80,90,100,110,120,130,140,150,160,170,
     A   180,190,200,210,220,230,240,250,260),INDEX1
   10 E(1)=1.0
      GO TO 300
   20 E(14)=1.0
      GO TO 300
   30 E(10)=1.0
      E(13)=1.0
      GO TO 300
   40 E(1)=1.0
      E(10)=1.0
      E(13)=1.0
      GO TO 300
   50 E(10)=1.0
      E(13)=1.0
      E(14)=1.0
      GO TO 300
   60 E(11)=1.0
      E(12)=1.0
      GO TO 300
   70 E(1)=1.0
      E(11)=1.0
      E(12)=1.0
      GO TO 300
   80 E(11)=1.0
      E(12)=1.0
      E(14)=1.0
      GO TO 300
   90 E(2)=1.0
      E(4)=1.0
      GO TO 300
  100 E(1)=1.0
      E(2)=1.0
      E(4)=1.0
      GO TO 300
  110 E(2)=1.0
      E(4)=1.0
      E(14)=1.0
      GO TO 300
  120 E(2)=1.0
      E(4)=1.0
      E(10)=1.0
      E(13)=1.0
      GO TO 300
  130 E(1)=1.0
      E(2)=1.0
      E(4)=1.0
      GO TO 300
  140 E(2)=1.0
      E(4)=1.0
      E(10)=1.0
      E(13)=1.0
      E(14)=1.0
      GO TO 300
  150 E(2)=1.0
      E(4)=1.0
      E(11)=1.0
      E(12)=1.0
      GO TO 300
  160 E(1)=1.0
      E(2)=1.0
      E(4)=1.0
      E(11)=1.0
      E(12)=1.0
      GO TO 300
  170 E(2)=1.0
      E(4)=1.0
      E(11)=1.0
      E(12)=1.0
      E(14)=1.0
      GO TO 300
  180 E(3)=1.0
      E(5)=1.0
      GO TO 300
  190 E(1)=1.0
      E(3)=1.0
      E(5)=1.0
      GO TO 300
  200 E(3)=1.0
      E(5)=1.0
      E(14)=1.0
      GO TO 300
  210 E(3)=1.0
      E(5)=1.0
      E(10)=1.0
      E(13)=1.0
      GO TO 300
  220 E(1)=1.0
      E(3)=1.0
      E(5)=1.0
      E(10)=1.0
      E(13)=1.0
      GO TO 300
  230 E(3)=1.0
      E(5)=1.0
      E(10)=1.0
      E(13)=1.0
      E(14)=1.0
      GO TO 300
  240 E(3)=1.0
      E(5)=1.0
      E(11)=1.0
      E(12)=1.0
      GO TO 300
  250 E(1)=1.0
      E(3)=1.0
      E(5)=1.0
      E(11)=1.0
      E(12)=1.0
      GO TO 300
  260 E(3)=1.0
      E(5)=1.0
      E(11)=1.0
      E(12)=1.0
      E(14)=1.0
  300 RETURN
C
      END
```

```
C
      SUBROUTINE TABLE3(INDEX3,E)
C     (* 482 3 -  FIND COMINATION OF TABLE ENTRIES THAT SATISFY THRUSTER
C        SELECTION REQUIREMENTS *)
C
C     CALLS
C        <NONE>
C     CALLED BY
C        SELECT
C
C     INPUT
C        INDEX3
C     OUTPUT
C        E
C
      REAL E(14)
C        COMMAND ENTRIES FROM THRUSTER SELECT LOGIC
      INTEGER INDEX3
C        THRUSTER TABLE INDEX
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* DO SELECTION BASED ON INDEX *)
      GOTO(10,20,30,40,50,60,70,80,90,100,110,120,130,140,150,160,170,
     A   180,190,200,210,220,230,240,250,260),INDEX3
   10 E(10)=1.0
      E(11)=1.0
      GO TO 300
   20 E(12)=1.0
      E(13)=1.0
      GO TO 300
   30 E(7)=1.0
      E(8)=1.0
      GO TO 300
   40 E(7)=1.0
      E(8)=1.0
      E(10)=1.0
      E(11)=1.0
      GO TO 300
   50 E(7)=1.0
      E(8)=1.0
      GO TO 300
   60 E(6)=1.0
      E(9)=1.0
      GO TO 300
   70 E(6)=1.0
      E(9)=1.0
      E(10)=1.0
      E(11)=1.0
      GO TO 300
   80 E(6)=1.0
      E(9)=1.0
      E(12)=1.0
      E(13)=1.0
      GO TO 300
   90 E(10)=1.0
      E(13)=1.0
      GO TO 300
  100 E(10)=1.0
      GO TO 300
  110 E(13)=1.0
      GO TO 300
  120 E(7)=1.0
      E(8)=1.0
      E(10)=1.0
      E(13)=1.0
      GO TO 300
  130 E(7)=1.0
      E(8)=1.0
      E(10)=1.0
      GO TO 300
  140 E(7)=1.0
      E(8)=1.0
      E(13)=1.0
      GO TO 300
  150 E(6)=1.0
      E(9)=1.0
      E(10)=1.0
      E(13)=1.0
      GO TO 300
  160 E(6)=1.0
      E(9)=1.0
      E(10)=1.0
      GO TO 300
  170 E(6)=1.0
      E(9)=1.0
      E(13)=1.0
      GO TO 300
  180 E(11)=1.0
      E(12)=1.0
      GO TO 300
  190 E(11)=1.0
      GO TO 300
  200 E(12)=1.0
      GO TO 300
  210 E(7)=1.0
      E(8)=1.0
      E(11)=1.0
      E(12)=1.0
      GO TO 300
  220 E(7)=1.0
      E(8)=1.0
      E(11)=1.0
      GO TO 300
  230 E(7)=1.0
      E(8)=1.0
      E(12)=1.0
      GO TO 300
  240 E(6)=1.0
      E(9)=1.0
      E(11)=1.0
      E(12)=1.0
      GO TO 300
  250 E(6)=1.0
      E(9)=1.0
      E(12)=1.0
      GO TO 300
  260 E(6)=1.0
      E(9)=1.0
      E(12)=1.0
  300 RETURN
C
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
      SUBROUTINE FIRTHR(E)
C     (* 483 - PUT THRUSTER COMMANDS IN COMMAND PORTS *)
C
C     CALLS
C        <NONE>
C     CALLED BY
C        THRUST
C
C     INPUT
C        E
C     OUTPUT
C        <NONE>
C
      REAL E(14)
C        THRUSTER COMMAND ENTRIES FROM THRUSTER SELECT LOGIC
C
C     (* TEMPORARY DUMMY SUBROUTINE *)
      RETURN
C
      END
```

```
C
      SUBROUTINE IMU(STATE,ACV,ACVT,ATRATE)
C     (* 490 - SIMULATE IMU MEASUREMENT *)
C
C     CALLS
C        DIRMAT,MADD,MSCL,ANGVEC
C     CALLED BY
C        DOCK
C
C     INPUT
C        QO,STATE,FULDIS,INERCV,MEMPTY
C     OUTPUT
C        ACV,ACVT,ATRATE
C
      REAL ACV(3,3),ACVT(3,3)
C        DIRECTION COSINE MATRIX AND ITS TRANSPOSE FOR 'MEASURED' ATTITUDE
C        OF CHASE VEHICLE WITH RESPECT TO PRIMARY REFERENCE FRAME
      REAL ATRATE(3)
C        MEASURED ATTITUDE RATES ABOUT CHASE VEHICLE AXES
      REAL FULDIS(3,3)
C        TENSOR DESCRIBING FUEL DISTRIBUTION
      REAL INERCV(3,3)
C        TENSOR DESCRIBING INERTIA OF EMPTY CHASE VEHICLE
      REAL INERFL(3,3)
C        FUEL INERTIA TENSOR
      REAL INERTA(3,3)
C        INERTIA OF CHASE VEHICLE (LOADED)
      REAL MEMPTY
C        MASS OF EMPTY CHASE VEHICLE
      REAL QO(4),QP(4)
C        INITIAL CHASE VEHICLE ATTITUDE QUATERNION IN 'TRUTH' REFERENCE FRAME
C        AND CURRENT PRIMARY REFERENCE FRAME, RESPECTIVELY
      REAL STATE(14)
C        TRUE CHASE VEH  E STATE
      REAL TACV(3,3),TAC  (3,3)
C        CHASE VEHICLE DIRECTION COSINE MATRIX WITH RESPECT TO TRUTH
C        COORDINATE SYSTEM AND INVERSE OF THIS MATRIX
C
      COMMON/REF/QO
      COMMON/MASPRP/FULDIS,INERCV,MEMPTY
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* SUBTRACT INITIAL ATTITUDE FROM CURRENT ATTITUDE TO GET CURRENT
C        ATTITUDE IN PRIMARY REF  FRAME *)
         QP(1)=QO(4)*STATE(10)+QO(3)*STATE(11)-QO(2)*STATE(12)-
     A      QO(1)*STATE(13)
         QP(2)=-QO(3)*STATE(10)+QO(4)*STATE(11)+QO(1)*STATE(12)-
     A      QO(2)*STATE(13)
         QP(3)=QO(2)*STATE(10)-QO(1)*STATE(11)+QO(4)*STATE(12)-
     A      QO(3)*STATE(13)
         QP(4)=QO(1)*STATE(10)+QO(2)*STATE(11)+QO(3)*STATE(12)+
     A      QO(4)*STATE(13)
C     (* 901 - CONVERT QUATERNIONS TO DIRECTION COSINE MATRICES *)
         CALL DIRMAT(QP,ACV,ACVT)
         CALL DIRMAT(STATE(10),TACV,TACVI)
C     (* DETERMINE INERTIA *)
         CALL MSCL(INERFL,FULDIS,3,3,STATE(14)-MEMPTY)
         CALL MADD(INERTA,INERCV,INERFL,3,3)
C     (* 904 - FIND ANGULAR VELOCITY VECTOR *)
         CALL ANGVEC(INERTA,STATE(7),ATRATE,TACV
      RETURN
C
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
      SUBROUTINE RPY(RPYERR,ACV,ESTATE,ESTA)
C     (* 4A0 - CALCULATE ROLL, PITCH, AND YAW ERRORS *)
C
C     CALLS
C        ATAN2,MSUB,DPRD,SQRT,MADD,MSCL,QMLT,MMLT,DIRMAT
C     CALLED BY
C        DOCK
C
C     INPUT
C        ACV,ESTATE ESTA
C     OUTPUT
C        RPYERR
C
      REAL AT(3,3),TRNAT(3,3)
C        ESTIMATED TARGET ATTITUDE DIRECTION COSINE MATRIX
      REAL QNEW(4)
C        PROJECTED TARGET ATTITUDE QUATERNION
      REAL ESTA(7),ESTATE(6)
C        STATE ESTIMATES FOR ATTITUDE AND TRANSLATION, RESPECTIVELY
      REAL ININV(3,3)
C        INVERSE OF TARGET MOMENT OF INERTIA TENSOR
      REAL T1(3),T2(4),T3(3),K1(4),T4(3)
C        INTERMEDIATE RESULTS WITH NO SIMPLE PHYSICAL INTERPRETATION
      REAL HDT(3)
C        TARGET DOCKING FIXTURE LOCATION IN TARGET FRAME
      REAL ACV(3,3)
C        DIRECTION COSINE MATRIX FROM IMU, SPECIFYING CHASE VEHICLE ATTITUDE
      REAL P1(3),R(3)
C        POINTING VECTOR FROM CHASE VEHICLE CENTER OF MASS TO TARGET DOCKING
C        FIXTURE, AND UNIT VECTOR PARALLEL TO TARGET -Y AXIS    BOTH
C        IN CURRENT CHASE VEHICLE FRAME
      REAL RPYERR(3)
C        ROLL, PITCH AND YAW ERRORS (RADIANS)
C
      COMMON/TGTI/ININV
      COMMON/HNGOFF/DUMMY(9),HDT
C
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* CONVERT TARGET ATTITUDE QUATERNION TO DIRECTION COSINE MATRIX *)
         CALL DIRMAT(ESTA,AT,TRNAT)
C     (* PROPAGATE TARGET ATTITUDE 1 233333 SECONDS *)
         CALL MMLT(T1,AT,ESTA(5),3,3,1)
         CALL MMLT(T2,ININV,T1,3,3,1)
         T2(4)=0
         CALL QMLT(K1,ESTA,T2)
         CALL MSCL(K1,K1,4,1,0 6166667)
         CALL MADD(QNEW,ESTA,K1,4,1)
         CALL MSCL(QNEW,QNEW,4,1,1 0/SQRT(DPRD(QNEW,QNEW,4)))
         CALL DIRMAT(QNEW,AT,TRNAT)
C     (* COMPUTE P1 = VECTOR FROM CHASE VEHICLE CENTER OF MASS TO PRO-
C        JECTED TARGET DOCKING FIXTURE *)
         CALL MMLT(T1,TRNAT,HDT,3,3,1)
         CALL MSCL(T4,ESTATE(4),3,1,1 233333)
         CALL MADD(T4,ESTATE,T4,3,1)
         CALL MSUB(T3,T1,T4,3,1)
         CALL MMLT(P1,ACV,T3,3,3,1)
C     (* COMPUTE ATTITUDE CORRECTIONS TO ALIGN DOCKING FIXTURE ON CHASE
C        VEHICLE WITH P1 *)
         RPYERR(2) = ATAN2(P1(3),P1(1))
         RPYERR(3) = ATAN2(-P1(2),P1(1))
         CALL MMLT(R,ACV,TRNAT(1,2),3,3,1)
         IF(R(2) EQ 0 0  AND  R(3) EQ 0 0) GO TO 10
            RPYERR(1)=ATAN2(-R(3),R(2))
            GO TO 20
   10    CONTINUE
            RPYERR(1)=0 0
   20    CONTINUE
      RETURN
C
      END
```

```
C
      SUBROUTINE MPROVE(U,V,ACV,QT,CVPOS)
C     (* 4C0 - IMPROVE IMAGE INTERPRETATION *)
C
C     CALLS
C        MMLT,MSUB,MINV6,DIRMAT,SQRT,TOQUAT,MIDN,MADD,MSCL
C     CALLED BY
C        DOCK
C
C     INPUT
C        U,V,AT,CVPOS,A,B,FOCLEN,HT,HC,ACV
C     OUTPUT
C        AT,CVPOS
C
      REAL A B
C        LAMP SPACING ON DOCKING AID
      REAL AT(3,3), TRNAT(3,3), QT(4)
C        TARGET ATTITUDE DIRECTION COSINE MATRIX, ITS TRANSPOSE, AND THE
C        QUATERNION CORRESPONDING TO 'AT'
      REAL ACV(3,3)
C        CHASE VEHICLE ATTITUDE MEASURED BY IMU
      REAL CVPOS(3)
C        MEASURED POSITION OF CHASE VEHICLE IN PRIMARY REF  FRAME
      REAL DUV(6), D'(6)
C        CHANGES IN LAMP IMAGE COORDINATES (DUV) CAUSED BY CHANGES IN
C        POSITION (DX(1)-DX(3)) AND ATTITUDE 'DX(4)-DX'6) )
      INTEGER ERR
C        ERROR CODE RETURNED FROM MATRIX INVERSION ROUTINE (0 = 'OK')
      REAL FOCLEN
C        CAMERA LENS FOCAL LENGTH
      REAL H(6,6), HA(2,6)
C        SENSITIVITY OF LAMP IMAGE COORDINATES TO POSITION AND ATTITUDE (H)
C        AND INTERMEDIATE RESULTS USED IN COMPUTING TWO ROWS OF H
      REAL G(6,6)
C        INVERSE OF H()
      INTEGER J
C        DO-LOOP INDEX FOR SHORT LOOP THAT COPIES ARRAY
      INTEGER LAMP
C        LAMP NUMBER OF LAMP UNDER CONSIDERATION
      REAL SCRACH(7,12)
C        SCRATCHPAD USED IN ROUTINE MINV IN INVERTING MATRIX H
      REAL VL(3)
C        POSITION OF CURRENTLY CONSIDERED LAMP IN TARGET SPACECRAFT REF  FRAME
      REAL S1(2,3), S2(3,6), S4(2,3),  V1(3), V2(3), V3(3), HTS(3,3)
C        INTERMEDIATE RESULTS WITH NO SIMPLE PHYSICAL INTERPRETATION
      REAL VLT(3)
C        VECTOR IN CAMERA COORD SYS  POINTING FROM CAMERA TO CURRENTLY CON-
C        SIDERED LAMP
      REAL U(3),V(3),UP(3),VP(3)
C        HORIZONTAL (U) AND VERTICAL (V) IMAGE-PLANE COORDINATES OF
C        LAMP IMAGES FOR THE THREE LAMPS, AS MEASURED AND AS PREDICTED FROM
C        IMAGE INTERPRETATION INPUT TO THIS ROUTINE
      REAL HT(3),HC(3)
C        POSITIONS OF THE DOCKING AID AND CAMERA IN INDIVIDUAL SPACECRAFT
C        REFERENCE FRAMES
C
      COMMON/OPTSYS/DUMMY(2),FOCLEN,A,B
      COMMON/HNGOFF/HC,HT,DUMMY1(6)
```

```
C
C     (* COMPUTE DIRECTION COSINE MATRIX FOR TARGET ATTITUDE AND ITS
C        TRANSPOSE *)
         CALL DIRMAT(QT,AT,TRNAT)
      VL(3)=HT(3)
      HTS(1,1)=0
      HTS(2,2)=0
      HTS(3,3)=0
      DO 60 LAMP = 1,3
C        (* COMPUTE LAMP POSITIONS IN CAMERA REF FRAME (VLT()) *)
            GO TO (10,20,30),LAMP
   10          CONTINUE
                  VL(1)=HT(1)
                  VL(2)=HT(2)+A
                  GOTO 40
   20          CONTINUE
                  VL(1)=HT(1)-B
                  VL(2)=HT(2)
                  GOTO 40
   30          CONTINUE
                  VL(1)=HT(1)
                  VL(2)=HT(2)-A
   40       CONTINUE
            CALL MMLT(V1,TRNAT,VL,3,3,1)
            CALL MSUB(V2,V1,CVPOS,3,1)
            CALL MMLT(V3,ACV,V2,3,3,1)
            CALL MSUB(VLT(1),V3,HC,3,1)
C        (* COMPUTE PREDICTED LAMP-IMAGE COORDINATES UP(),VP() *)
            UP(LAMP)=-VLT(2)*FOCLEN/VLT(1)
            VP(LAMP)= VLT(3)*FOCLEN/VLT(1)
C        (* COMPUTE 2 ROWS OF SENSITIVITY MATRIX H *)
            HTS(2,1)=VL(3)*2
            HTS(1,2)=-HTS(2,1)
            HTS(1,3)=VL(2)*2
            HTS(3,1)=-HTS(1,3)
            HTS(3,2)=VL(1)*2
            HTS(2,3)=-HTS(3,2)
            CALL MMLT(S2(1,4),TRNAT,HTS,3,3,3)
            CALL MIDN(S2,3)
            S1(1,1)=-VLT(2)/VLT(1)
            S1(2,1)= VLT(3)/VLT(1)
            S1(1,2)=1
            S1(2,2)=0
            S1(1,3)=0
            S1(2,3)=-1
            CALL MMLT(S4,S1,ACV,2,3,3)
            CALL MMLT(HA,S4,S2,2,3,6)
            CALL MSCL(HA,HA,2,6, FOCLEN/VLT(1) )
            DO 50 J=1,6
               H( LAMP  J)    HA(1,J)
               H(LAMP+3,J) -  HA(2,J)
   50       CONTINUE
   60 CONTINUE
C     (* COMPUTE CORRECTION TO INITIAL MEASUREMENT *)
         CALL MINV6(G,H,6,SCRACH,7,12,ERR)
         IF(ERR NE 0) GOTO 70
         CALL MSUB( DUV   ,U,UP,3,1)
         CALL MSUB(DUV(4),V,VP,3,1)
         CALL MMLT(DX,G,DUV,6,6,1)
C     (* CALCULATE REFINED POSTION MEASUREMENT *)
         CALL MADD(CVPOS,CVPOS,DX,3,1)
C     (* CALCULATE REFINED ATTITUDE MEASUREMENT *)
         QT(1)=             QT(1) + DX(6)*QT(2) - DX(5)*QT(3) + DX(4)*QT(4)
         QT(2)= -DX(6)*QT(1) +            QT(2) + DX(4)*QT(3) + DX(5)*QT(4)
         QT(3)=  DX(5)*QT(1) - DX(4)*QT(2) +            QT(3) + DX(6)*QT(4)
         QT(4)= -DX(4)*QT(1) - DX(5)*QT(2) - DX(6)*QT(4) +            QT(4)
         CALL MSCL(QT,QT,4,1,1 /SQR1(QT(1)**2+QT(2)**2+QT(3)**2+
     A      QT(4)**2))
      RETURN
C
   70 CONTINUE
      WRITE(1,901)
         RETURN
C
  901 FORMAT(' *** MATRIX INVERSION FAILURE IN SUBROUTINE MPROVE ***')
      END
```

```
C
      SUBROUTINE TOQUAT(A,Q)
C     (* 4C2 - CONVERT DIRECTION COSINE MATRIX TO QUATERNION *)
C
C     CALLS
C        SQR1
C     CALLED BY
C        MPROVE
C
C     INPUT
C        A
C     OUTPUT
C        Q
C
      REAL A(3,3)
C        DIRECTION COSINE MATRIX
      REAL Q(4)
C        CORRESPONDING QUATERNION
      REAL FACTOR
C        INTERMEDIATE RESULTS
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
      Q(4)=0 5*SQR1(1 +A(1,1)+A(2,2)+A(3,3))
      IF(Q(4) LT 0 5) GO TO 10
         FACTOR=0 25/Q(4)
         Q(1)=FACTOR*(A(2,3)-A(3,2))
         Q(2)=FACTOR*(A(3,1)-A(1,3))
         Q(3)=FACTOR*(A(1,2)-A(2,1))
         GO TO 40
   10 Q(1)=0 5*SQR1(1 +A(1,1)-A(2,2)-A(3,3))
      IF(Q(1) LT 0 5) GO TO 20
         FACTOR=0 25/Q(1)
         Q(2)=FACTOR*(A(1,2)+A(2,1))
         Q(3)=FACTOR*(A(1,3)+A(3,1))
         Q(4)=FACTOR*(A(2,3)-A(3,2))
         GO TO 40
   20 Q(2)=0 5*SQR1(1 +A(2,2)-A(1,1)-A(3,3))
      IF(Q(2) LT 0 5) GO TO 30
         FACTOR=0 25/Q(2)
         Q(1)=FACTOR*(A(1,2)+A(2,1))
         Q(3)=FACTOR*(A(2,3)+A(3,2))
         Q(4)=FACTOR*(A(3,1)-A(1,3))
         GO TO 40
   30 CONTINUE
         Q(3)=0 5*SQR1(1 +A(3,3)-A(2,2)-A(1,1))
         FACTOR=0 25/Q(3)
         Q(1)=FACTOR*(A(1,3)+A(3,1))
         Q(2)=FACTOR*(A(2,3)+A(3,2))
         Q(4)=FACTOR*(A(1,2)-A(2,1))
   40 CONTINUE
      IF(Q(4) GE 0 ) GO TO 50
         Q(1)=-Q(1)
         Q(2)=-Q(2)
         Q(3)=-Q(3)
         Q(4)=-Q(4)
   50 CONTINUE
      RETURN
C
      END
```

```
C
      SUBROUTINE MINV6(G,H,IDUM1,DUMMY,IDUM2,IDUM3,ERR)
C     (* 4C1 - INVERT 6X6 ILL-CONDITIONED MATRIX *)
C
C     CALLS
C        DMINV
C     CALLED BY
C        MPROVE
C
C     INPUT
C        H
C     OUTPUT
C        G,ERR
C
      REAL G(6,6),H(6,6)
C        INVERSE AND ORIGINAL MATRIX
      DOUBLE PRECISION GD(6,6),HD(6,6),SCRACH(7,12)
C        COPIES OF G,H IN DOUBLE PRECISION AND SCRATCHPAD
      INTEGER ERR
C        ERROR CODE (=0 IF NO ERROR)
      INTEGER I,J
C        DO-LOOP INDICES
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
      DO 10 I = 1,6
         DO 10 J = 1,6
            HD(J,I) = H(J,I)
   10 CONTINUE
      CALL DMINV(GD,HD,6,SCRACH,7,12,ERR)
      DO 20 I = 1,6
         DO 20 J = 1,6
            G(J,I) = GD(J,I)
   20 CONTINUE
      RETURN
C
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
      SUBROUTINE INCRPA(PA,ESTA,ESTATE,QMEAS)
C     (* 4D0 - INCORPORATE ATTITUDE MEASUREMENT *)
C
C     CALLS
C        MMLT,MTRN,MINV,ATMCOV,COMPT,XFORMP,MSCL,MIDN,MSUB,SQRT
C     CALLED BY
C        DOCK
C
C     INPUT
C        PA,ESTA,ESTATE,QMEAS
C     OUTPUT
C        PA,ESTA
C
      REAL DX(7)
C        CHANGE IN ESTA()
      INTEGER ERR
C        ERROR FLAG FROM MATRIX INVERSION ROUTINE (0 = NO ERROR)
      REAL ESTA(7), ESTATE(6)
C        ESTIMATED STATES FOR ATTITUDE AND TRANSLATION
      INTEGER I,J
C        DO-LOOP INDICES
      REAL K(7,4)
C        KALMAN GAIN MATRIX
      REAL PA(7,7)
C        ATTITUDE COVARIANCE MATRIX
      REAL QMEAS(4)
C        MEASURED TARGET ATTITUDE QUATERNION
      REAL SCRACH(5,8)
C        SCRATCHPAD FOR MATRIX INVERSION ROUTINE
      REAL T(4,4), TT(4,4)
C        TRANSFORMATION MATRICES DERIVED FROM QUATERNION PORTION OF ESTA()
      REAL R(4,4)
C        MEASUREMENT COVARIANCE MATRIX CORRESPONDING TO FIRST 3 ELEMENTS
C        OF QMEAS()
      REAL R (4,4), RI(4,4), QMP(4), QNEW(4), COEFF(7,7), PNEW(7,7)
C        INTERMEDIATE RESULTS WITH NO SIMPLE PHYSICAL INTERPRETATION
C
C     (* COMPUTE TRANSFORMATION MATRIX T AND ITS INVERSE TT *)
      CALL COMPT(ESTA,T,TT)
C     (* 4D3 - COMPUTE ATTITUDE MEASUREMENT COVARIANCE *)
      CALL ATMCOV(ESTATE,R)
C     (* TRANSFORM STATE ESTIMATE COVARIANCE MATRIX PA *)
      CALL XFORMP(PA,T,TT)
C     (* COMPUTE KALMAN GAIN MATRIX K *)
      DO 10 I = 1,4
         DO 10 J = 1,4
            RI(I,J) = R(I,J) + PA(I,J)
   10 CONTINUE
      CALL MINV(RI,RI,4,SCRACH,5,8,ERR)
      IF(ERR.NE.0) GO TO 20
      CALL MMLT(K,PA,RI,7,4,4)
C     (* UPDATE STATE ESTIMATE *)
      QMP(1)= ESTA(4)*QMEAS(1)+ESTA(3)*QMEAS(2)-ESTA(2)*QMEAS(3)
     A       -ESTA(1)*QMEAS(4)
      QMP(2)=-ESTA(3)*QMEAS(1)+ESTA(4)*QMEAS(2)+ESTA(1)*QMEAS(3)
     A       -ESTA(2)*QMEAS(4)
      QMP(3)= ESTA(2)*QMEAS(1)-ESTA(1)*QMEAS(2)+ESTA(4)*QMEAS(3)
     A       -ESTA(3)*QMEAS(4)
      QMP(4)= ESTA(1)*QMEAS(1)+ESTA(2)*QMEAS(2)+ESTA(3)*QMEAS(3)
     A       +ESTA(4)*QMEAS(4)
      IF(QMP(4).LT.0.) CALL MSCL(QMP,QMP,4,1,-1.0)
      QMP(4)=QMP(4)-1.0
      CALL MMLT(DX,K,QMP,7,4,1)
      ESTA(1)=DX(1)
      ESTA(2)=DX(2)
      ESTA(3)=DX(3)
      ESTA(4)=DX(4)+1.0
      ESTA(5)=ESTA(5)+DX(5)
      ESTA(6)=ESTA(6)+DX(6)
      ESTA(7)=ESTA(7)+DX(7)
C     (* UPDATE COVARIANCE MATRIX *)
      CALL MIDN(COEFF,7)
      CALL MSUB(COEFF,COEFF,K,7,4)
      CALL MMLT(PNEW,COEFF,PA,7,7,7)
      CALL MSCL(PA,PNEW,7,7,1.0)
C     (* 4D1 - TRANSFORM PA BACK *)
      CALL XFORMP(PA,TT,T)
C     (* TRANSFORM STATE ESTIMATE & NORMALIZE QUATERNION *)
      CALL MMLT(QNEW,T,ESTA,4,4,1)
      CALL MSCL(ESTA,QNEW,4,1,1.0/SQRT(ESTA(1)**2+ESTA(2)**2+
     A     ESTA(3)**2+ESTA(4)**2))
      RETURN
C
   20 CONTINUE
      WRITE(1,901)
         STOP
C
  901 FORMAT('MATRIX INVERSION FAILURE IN SUBROUTINE INCRPA')
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
      SUBROUTINE XFORMP(PA,T,TT)
C      * 4D1 - TRANSFORM COVARIANCE MATRIX *)
C
C     CALLS
C        MMLT
C     CALLED BY
C        INCRPA
C
C     INPUT
C        PA,T,TT
C     OUTPUT
C        PA
C
      INTEGER I,J
C        DO-LOOP INDICES
      REAL PA(7,7)
C        COVARIANCE MATRIX
      REAL T(4,4), TT(4,4)
C        SKEW-SYMMETRIC MATRIX DERIVED FROM QUATERNION AND ITS INVERSE
      REAL P11B(4,4), P11A(4,4), P11(4,4),  P12A(4,3), P12(4,3)
C        INTERMEDIATE RESULTS WITH NO SIMPLE PHYSICAL INTERPRETATION
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* TRANSFORM UPPER LEFT 4X4 SUBMATRIX   P11=TT*UPPERLEFT(PA)*T *)
         DO 10 I = 1,4
            DO 10 J = 1,4
               P11B(J,I) = PA(J,I)
   10    CONTINUE
         CALL MMLT(P11A,P11B,T,4,4,4)
         CALL MMLT(P11,TT,P11A,4,4,4)
C     (* TRANSFORM UPPER RIGHT 4X3 SUBMATRIX & USE ITS TRANSPOSE FOR
C        TRANSFORMED LOWER LEFT 3X4 SUBMATRIX *)
         DO 20 I = 1,3
            DO 20 J = 1,4
               P12A(J,I) = PA(J,I+4)
   20    CONTINUE
         CALL MMLT(P12,TT,P12A,4,4,3)
C     (* COPY TRANSFORMED SUBMATRICES BACK INTO PA *)
         DO 30 I = 1,4
            DO 30 J = 1,4
               PA(J,I) = P11(J,I)
   30    CONTINUE
         DO 40 I = 1,4
            DO 40 J = 1,3
               PA(I,J+4) = P12(I,J)
               PA(J+4,I) = P12(I,J)
   40    CONTINUE
      RETURN
C
      END
```

```
C
      SUBROUTINE COMPT(Q,T,TT)
C     (* 4D2 - COMPUTE TRANSFORMATION MATRIX & ITS TRANSPOSE FROM
C        QUATERNION Q *)
C
C     CALLS
C        MTRN
C     CALLED BY
C        INCRPA
C
C     INPUT
C        Q
C     OUTPUT
C        T, TT
C
      REAL Q(4)
C        QUATERNION REPRESENTING ATTITUDE
      REAL T(4,4), TT(4,4)
C        TRANSFORMATION MATRIX CORRESPONDING TO Q(), AND ITS TRANSPOSE
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
      T(1,1) =   Q(4)
      T(2,1) =   Q(3)
      T(3,1) =  -Q(2)
      T(4,1) =  -Q(1)
      T(1,2) =  -Q(3)
      T(2,2) =   Q(4)
      T(3,2) =   Q(1)
      T(4,2) =  -Q(2)
      T(1,3) =   Q(2)
      T(2,3) =  -Q(1)
      T(3,3) =   Q(4)
      T(4,3) =  -Q(3)
      T(1,4) =   Q(1)
      T(2,4) =   Q(2)
      T(3,4) =   Q(3)
      T(4,4) =   Q(4)
      CALL MTRN(TT,T,4)
      RETURN
C
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
      SUBROUTINE ATMCOV(ESTATE,R)
C     (* 4D3 - ESTIMATE TARGET-ATTITUDE MEASUREMENT COVARIANCE MATRIX *)
C
C     CALLS
C        MIDN,MSCL,SQR1,DPRD
C     CALLED BY
C        INCRPA
C
C     INPUT
C        ESTATE
C     OUTPUT
C        R
C
      REAL ESTATE(3)
C        ESTIMATED TRANSLATIONAL POSITION OF CHASE VEHICLE IN PRIMARY FRAME
      INTEGER I
C        LOOP INDEX
      REAL R(4,4)
C        ESTIMATED COVARIANCE OF MEASUREMENT
      REAL RANGE
C        ESTIMATED DISTANCE TO TARGET
      REAL V
C        ESTIMATED VARIANCE PER ELEMENT OF MEASUREMENT
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
      RANGE=SQR1(DPRD(ESTATE,ESTATE,3))
      V=1 129E-9*RANGE**3+ 000:
      CAL       .R,4)
      CALL n.CL(R,R,4,4,V)
      R(4,4)=0 01*V
      RETURN
C
      END
```



```
C
      SUBROUTINE PRPESA(PA,ESTA,STEP)
C     (* 4EO - PROPAGATE ATTITUDE ESTIMATE AND COVARIANCE *)
C
C     CALLS
C        MMLT,MSCL,DIRMAT,CMPCD,MADD,MIDN,MTRN,QMLT,FLOAT,SQRT
C     CALLED BY
C        DOCK
C
C     INPUT
C        PA,ESTA,STEP
C     OUTPUT
C        PA,ESA
C
      REAL AT(3,3),TRNAT(3,3)
C        TARGET ATTITUDE DIRECTION COSINE MATRIX, DERIVED FROM ESTA
C        AND ITS TRANSPOSE
      REAL ESTA(7)
C        ATTITUDE STATE ESTIMATE   1ST 4 ELEMENTS = TARGET ATTITUDE QUATERNION,
C        NEXT 3 ELEMENTS = TARGET ANGULAR MOMENTUM IN PRIMARY FRAME
C        ESTNEW() IS UPDATED ESTA() BEFORE NORMALIZATION OF QUATERNION
      REAL F(7,7)
C        PARTIAL OF D(ESTA)/DT WITH RESPECT TO (ESTA)
      REAL HAV(4)
C        HALF TARGET ANGULAR VELOCITY IN CURRENT TARGET FRAME,AUGMENTED
C        WITH A ZERO AS FOURTH ELEMENT
      INTEGER I, II
C        DO-LOOP INDICES
      REAL ININV(3,3)
C        INVERSE OF TARGET MOMENT-OF-INERTIA MATRIX
      REAL JA(4)
C        AUGMENTED COLUMN FROM J()*0 5
      REAL PA(7,7)
C        ATTITUDE ESTIMATE COVARIANCE MATRIX CORRESPONDING TO ESTA()
      REAL PHI(7,7)
C        STATE TRANSITION MATRIX DERIVED FROM F()
      REAL Q(7,7)
C        EMPIRICAL STATE NOISE COVARIANCE MATRIX
      REAL STEP
C        TIME INTERVAL OVER WHICH STATE ESTIMATE & COVARIANCE ARE TO BE
C        PROPAGATED
      REAL J(3,3), B(4,3), C(4,3), D(3,3,4), V(3), PHI1(7,7), P1 7,7)
      REAL QI(4),AL(3),K1(4),K2(4),QTEMP(4)
C        INTERMEDIATE RESULTS WITH NO SIMPLE PHYSICAL INTERPRETATION
C
      COMMON/TGTI/ ININV
C
      DATA Q/5 E-5,7*0 ,5 E-5,7*0 ,5 E-5,7*0 ,5 E-5,7*
     *    4,7*0 , 4,7*0 , 4/
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
C      (* COMPUTE AT FROM ESTA *)
          CALL DIRMAT(ESTA,AT,TRNAT)
C      (* COMPUTE J *)
          CALL MMLT(J,ININV,AT,3,3,3)
C      (* COMPUTE B, C & D MATRICES *)
          CALL CMPCD(ESTA,C,D)
          CALL MMLT(B,C,ININV,4,3,3)
C      (* COMPUTE PARTIAL OF RATE OF CHANGE OF QUATERNION WITH RESPECT TO
C         THE QUATERNION *)
          DO 10 I=1,4
              CALL MMLT(V,D(1,1,I),ESTA(5),3,3,1)
              CALL MMLT(F(1,I),B,V,4,3,1)
   10     CONTINUE
C         (* COMPUTE AUGMENTED HALF ANGULAR VELOCITY *)
              CALL MMLT(HAV,J,ESTA(5),3,3,1)
              HAV(4)=0
              CALL MSCL(HAV,HAV,3,1,0 5)
          F(1,2) = F(1,2)+HAV(3)
          F(1,3) = F(1,3)-HAV(2)
          F(1,4) = F(1,4)+HAV(1)
          F(2,1) = F(2,1)-HAV(3)
          F(2,3) = F(2,3)+HAV(1)
          F(2,4) = F(2,4)+HAV(2)
          F(3,1) = F(3,1)+HAV(2)
          F(3,2) = F(3,2)-HAV(1)
          F(3,4) = F(3,4)+HAV(3)
          F(4,1) = F(4,1)-HAV(1)
          F(4,2) = F(4,2)-HAV(2)
          F(4,3) = F(4,3)-HAV(3)
C      (* COMPUTE PARTIAL OF RATE OF CHANGE OF QUATERNION WITH RESPECT TO
C         ANGULAR MOMENTUM *)
          DO 20 I = 1,3
              CALL MSCL(JA,J(1,I),3,1,0 5)
              JA(4)=0
              CALL QMLT(F(1,I+4),ESTA,JA)
   20     CONTINUE
C      (* FILL BOTTOM OF F WITH ZEROES *)
          DO 30 I = 1,7
              DO 30 I1 = 5,7
                  F(I1,I) = 0 0
   30     CONTINUE
C      (* COMPUTE STATE TRANSITION MATRIX *)
          CALL MIDN(PHI,7)
          DO 40 I = 1,2
              CALL MMLT(PHI1,F,PHI,7,7,7)
              CALL MSCL(PHI,PHI1,7,7,STEP/FL AT(3-I))
              DO 40 I1=1,7
                  PHI(I1,I1) = PHI(I1,I1)    1 0
   40     CONTINUE
C      (* COMPUTE PA = PHI * PA * TRN(PHI) + Q *)
          CALL MTRN(PHI1,PHI,7)
          CALL MMLT(P1,PHI,PA,7,7,7)
          CALL MMLT(PA,P1,PHI1,7,7,7)
          CALL MADD(PA,PA,Q,7,7)
C      (* PROPAGATE STATE BY 2ND ORDER RUNGE-KUTTA INTEGRATION *)
          CALL QMLT(K1,ESTA,HAV)
          CALL MSCL(K1,K1,4,1,STEP)
          CALL MADD(QTEMP,ESTA,K1,4,1)
          CALL MSCL(QTEMP,QTEMP,4,1,1 0/SQRT(' 'EMP(1)**2+QTEMP(2)**2+
     A        QTEMP(3)**2+QTEMP(4)**2))
          CALL DIRMAT(QTEMP,AT,TRNAT)
          CALL MMLT(AL,AT,ESTA(5),3,3,1)
          CALL MMLT(QI,ININV,AL,3,3,1)
          CALL MSCL(QI,QI,3,1,0 5*STEP)
          QI(4)=0
          CALL QMLT(K2,QTEMP,QI)
          CALL MADD(K2,K1,K2,4,1)
          CALL MSCL(K2,K2,4,1,0 5)
          CALL MADD(ESTA,ESTA,K2,4,1)
          CALL MSCL(ESTA,ESTA,4,1,1 0/SQRT(ESTA(1)**2+ESTA(2)**2+
     A        ESTA(3)**2+ESTA(4)**2))
       RETURN
C
       END
```

```
C
          SUBROUTINE CMPCD(Q,C,D)
C      /* 4E1 - COMPUTE 'C' AND 'D' MATRICES. INTERMEDIATE RESULTS USED IN
C           'PRPESA' *)
C
C         CALLS
C            <NONE>
C         CALLED BY
C            PRPESA
C
C         INPUT
C            Q
C         OUTPUT
C            C,D
C
          REAL Q(4)
C            QUATERNION PORTION OF ATTITUDE STATE ESTIMATE
          REAL C(12) , D(36)
C            INTERMEDIATE RESULTS DERIVED FROM Q. USED IN 'PRPESA'   NOTE
C            DEPENDS ON FACT THAT FORTRAN ARRAYS ARE STORED WITH FIRST SUBSCRIPT
C            'LEAST SIGNIFICANT', ALLOWING SINGLE SUBSCRIPT IN THIS ROUTINE TO
C            CORRESPOND TO MULTIPLE SUBSCRIPTS IN CALLING ROUTINE   THIS MAY CAUSE
C            CONVERSION PROBLEMS IF IT IS TRANSPORTED TO ANOTHER COMPUTER
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
          C(1)  = Q(4)
          C(2)  = Q(3)
          C(3)  =-Q(2)
          C(4)  =-Q(1)
          C(5)  =-Q(3)
          C(6)  = Q(4)
          C(7)  = Q(1)
          C(8)  =-Q(2)
          C(9)  = Q(2)
          C(10)=-Q(1)
          C(11)= Q(4)
          C(12)=-Q(3)
C
          D(1)  = Q(1)
          D(2)  = Q(2)
          D(3)  = Q(3)
          D(4)  = Q(2)
          D(5)  =-Q(1)
          D(6)  =-Q(4)
          D(7)  = Q(3)
          D(8)  = Q(4)
          D(9)  =-Q(1)
          D(10)=-Q(2)
          D(11)= Q(1)
          D(12)= Q(4)
          D(13)= Q(1)
          D(14)= Q(2)
          D(15)= Q(3)
          D(16)=-Q(4)
          D(17)= Q(3)
          D(18)=-Q(2)
          D(19)=-Q(3)
          D(20)=-Q(4)
          D(21)= Q(1)
          D(22)= Q(4)
          D(23)=-Q(3)
          D(24)= Q(2)
          D(25)= Q(1)
          D(26)= Q(2)
          D(27)= Q(3)
          D(28)= Q(4)
          D(29)=-Q(3)
          D(30)= Q(2)
          D(31)= Q(3)
          D(32)= Q(4)
          D(33)=-Q(1)
          D(34)=-Q(2)
          D(35)= Q(1)
          D(36)= Q(4)
C
          RETURN
          END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
      SUBROUTINE DOKCHK(HDC,STATE,DOKRAD,HDT,DOCKED,T)
C     (* 4FO - DETERMINE WHETHER CHASE VEHICLE HAS DOCKED YET *)
C
C     CALLS
C        MMLT,MADD,MSUB,SQRT,TRGATT,DIRMAT,AMAX1,ABS,SQR1,DPRD
C     CALLED BY
C        DOCK
C
C     INPUT
C        T,HDC,HDT,STATE,DOKRAD
C     OUTPUT
C        DOCKED
C
      REAL ACV(3,3),TRNAC(3,3)
C        DIRECTION COSINE MATRIX GIVING CHASE VEHICLE ATTITUDE IN TRUTH
C        FRAME, AND TRANSPOSE OF THIS MATRIX
      REAL AT(3,3),TRNAT(3,3)
C        DIRECTION COSINE MATRIX GIVING TARGET ATTITUDE IN TRUTH FRAME,
C        AND TRANSPOSE OF THIS MATRIX
      LOGICAL DOCKED
C        TRUE IF CHASE VEHICLE HAS DOCKED
      REAL DOKRAD
C        RADIUS AT WHICH CHASE VEHICLE IS CONSIDERED DOCKED
      REAL HDC(3),HDT(3)
C        HANGOFF OF DOCKING FIXTURES FROM CENTERS OF MASS FOR CHASE VEHICLE
C        AND TARGET, RESPECTIVELY
      REAL R(3)
C        SEPARATION OF DOCKING FIXTURES
      REAL R1(3),R2(3),R3(3)
C        INTERMEDIATE RESULTS IN COMPUTATION OF R()
      REAL STATE(3)
C        POSITION OF CHASE VEHICLE IN TRUTH FRAME
      REAL T
C        ELAPSED SECONDS SINCE START OF RENDEZVOUS OPERATION
C
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* 903 - DETERMINE TRUE TARGET ATTITUDE *)
         CALL TRGATT(TRNAT,T)
C     (* 901 - DETERMINE CHASE VEHICLE ATTITUDE *)
         CALL DIRMAT(STATE(10),ACV,TRNAC)
C     (* FIGURE SEPARATION OF DOCKING FIXTURES *)
         CALL MMLT(R1,TRNAT,HDT,3,3,1)
         CALL MMLT(R2,TRNAC,HDC,3,3,1)
         CALL MSUB(R3,R2,R1,3,1)
         CALL MADD(R,STATE,R3,3,1)
      DOCKED = (SQRT(R(1)**2+R(2)**2+R(3)**2) LE DOKRAD)  OR
     A   (T GT 550 )  OR   (STATE(1) LT -380 )  OR   (STATE(1) GT 300 )
     B   OR   (AMAX1(ABS(STATE(2)),ABS(STATE(3))) GT 300 )
     C   OR   (SQR1(DPRD(STATE,STATE,3)) LT 5.0)
      RETURN
C
      END
```

```
C
      SUBROUTINE DIRMAT(Q,A,TRNA)
C     (* 901 - COMPUTE A DIRECTION COSINE MATRIX FROM A QUATERNION *)
C
C     CALLS
C        MTRN
C     CALLED BY
C        FLASH,ATITUD,QUATRN,SETGOL,IMU,RPY,MPROVE,PRPESA,DOKCHK,STPRIM
C
C     INPUT
C        Q
C     OUTPUT
C        A,TRNA
C
      REAL A(3,3)
C        DIRECTION COSINE MATRIX FORMED FROM QUATERNION Q
      REAL TRNA(3,3)
C        TRANPOSE OF A
      REAL Q(4)
C        QUATERNION
C
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* COMPUTE ELEMENTS OF A *)
         A(1,1)=Q(1)**2-Q(2)**2-Q(3)**2+Q(4)**2
         A(1,2)=2 *(Q(1)*Q(2)+Q(3)*Q(4))
         A(1,3)=2 *(Q(1)*Q(3)-Q(2)*Q(4))
         A(2,1)=2 *(Q(1)*Q(2)-Q(3)*Q(4))
         A(2,2)=-Q(1)**2+Q(2)**2-Q(3)**2+Q(4)**2
         A(2,3)=2 *(Q(2)*Q(3)+Q(1)*Q(4))
         A(3,1)=2 *(Q(1)*Q(3)+Q(2)*Q(4))
         A(3,2)=2 *(Q(2)*Q(3)-Q(1)*Q(4))
         A(3,3)=-Q(1)**2-Q(2)**2+Q(3)**2+Q(4)**2
C     (* COMPUTE TRANSPOSE OF A *)
         CALL MTRN(TRNA,A,3)
      RETURN
C
      END
```

```
C
      SUBROUTINE STPRIM(STATE,F,INERTA,T,DSTATE)
C     (* 902 - DETERMINE TIME DERIVATIVE OF STATE VECTOR *)
C
C     CALLS
C        ANGVEC,FORCE,LINACL,LPRIME,MAKROT,MPRIME,QPRIME,TORQUE,DIRMAT
C     CALLED BY
C        COMPK1,COMPK2,COMPK3,COMPK4
C
C     INPUT
C        F,T,STATE,INERTA
C     OUTPUT
C        DSTATE
C
      REAL A(3,3)
C        'TRUE' DIRECTION COSINE MATRIX FOR CHASE VEHICLE
      REAL BODVEL(3)
C        ANGULAR VELOCITY ABOUT AXES OF CHASE VEHICLE
      REAL DSTATE(14)
C        TIME DERIVATIVE OF STATE VECTOR
      REAL F(14)
C        FORCES FROM CHASE VEHICLE THRUSTERS
      REAL INERTA(3,3)
C        INERTIA OF CASE VEHICLE (INCL  FUEL)
      REAL N(3)
C        TORQUE ON SPACECRAFT ABOUT ITS CENTER OF MASS
      REAL NTFORC(3)
C        NET FORCE FROM THRUSTER OPERATION
      REAL OMEGA(4,4)
C        ROTATION MATRIX OMEGA, TO BE APPLIED TO QUATERNION MATRIX 'Q'
C        TO FORM TIME DERIVATIVE OF Q (Q=ELEMENTS 10-13 OF STATE)
      REAL STATE(14)
C        CHASE VEHICLE STATE VECTOR
      REAL
C        ELAPSED TIME
      REAL TRNA(3,3)
C        TRANSPOSE OF A
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* 9.. - COMPUTE A,TRNA FROM QUATERNION  *)
      CALL DIRMAT(STATE(10),A,TRNA)
C     (* 902 1 - DETERMINE NET FORCE FROM THRUSTERS IN 'TRUTH' COORD SYS  *)
      CALL FORCE(F,NTFORC,TRNA)
C     (* 90  2 - COMPUTE TIME DERIVATIVE OF SPACECRAFT MASS *)
      CALL MPRIME(F,DSTATE(14))
C     (* 902 3 - COMPUTE LINEAR ACCELERATIONS *)
      CALL LINACL(T,STATE(14),NTFORC,STATE(1),DSTATE(4))
C     (* 904 - FIND ANGULAR VELOCITY VECTOR *)
      CALL ANGVEC(INERTA,STATE(7),BODVEL,A)
C     (* 902 4 - CALCULATE TORQUE ON SPACECRAFT FROM THRUSTER OPERATION *)
      CALL TORQUE(F,N,TRNA)
C     (* 902 5 - COMPUTE TIME DERIVATIVE   ANGULAR MOMENTUM VECTOR *)
      CALL LPRIME(N,DSTATE(7))
C     (* 902 6 - FORM ROTATION MATRIX OMEGA *)
      CALL MAKROT(BODVEL,OMEGA)
C     (* 902 7 - CALCULATE THE TIME DERIVATIVE OF QUATERNION *)
      CALL QPRIME(STATE(10),OMEGA,DSTATE(10))
C     (* DEFINE VELOCITY ELEMENTS OF DSTATE *)
      DSTATE(1)=STATE(4)
      DSTATE(2)=STATE(5)
      DSTATE(3)=STATE(6)
      RETURN
C
      END
```

```
C
      SUBROUTINE FORCE(F,NTFORC,TRNA)
C     (* 902 1 -  COMPUTE NET FORCE FROM THRUSTER OPERATION USING EQUATION
C            NET FORCE= A TRANSPOSE * AK2 * F *)
C
C     CALLS
C        MMLT
C     CALLED BY
C        STPRIM
C
C     INPUT
C        F,AK2,Q
C     OUTPUT
C        NTFORC
C
      REAL AK2(3,14)
C        CONSTANT MATRIX RELATING INDIVIDUAL THRUSTER OUTPUTS
C        TO NET THRUST IN SPACECRAFT REFERENCE FRAME
C        (THIS ACCOMODATES MISALIGNMENT OF THRUSTERS)
      REAL B(3)
C          INTERMEDIATE RESULTS
      REAL F(14)
C        FORCE VECTOR OF CHASE VEHICLE AFTER THRUSTER SELECTION
      REAL NTFORC(3)
C        NET FORCE FROM THRUSTER OPERATION IN TRUTH COORD  SYS
      REAL TRNA(3,3)
C        TRANSPOSE OF DIRECTION COSINE MATRIX A
C
      COMMON/VEHIC/DUMMY1(14),AK2,DUMMY2(57)
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* COMPUTE NET FORCE *)
      CALL MMLT(B,AK2,F,3,14,1)
      CALL MMLT(NTFORC,TRNA,B,3,3,1)
      RETURN
C
      END
```

```
C
      SUBROUTINE MPRIME(F,DMDT)
C     (* 902 2 -   COMPUTE TIME DERIVATIVE OF SPACECRAFT MASS *)
C
C     CALLS
C        MMLT
C     CALLED BY
C        STPRIM
C
C     INPUT
C        AK1,F
C     OUTPUT
C        DMDT
C
      REAL AK1(14)
C        CONSTANT VECTOR DERIVED FROM ENGINE SPECIFIC IMPULSE DATA
      REAL DMDT
C        TIME DERIVATIVE OF MASS
      REAL F(14)
C        FORCES FROM THRUSTERS AFTER SELECTION
C
      COMMON/VEHIC/AK1,DUMMY(99)
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* COMPUTE DMDT = AK1*F *)
         CALL MMLT(DMDT,AK1,F,1,14,1)
C     (* CHANGE SIGN   MASS LOSS IS NEGATIVE *)
         DMDT=-DMDT
      RETURN
C
      END
```

```
C
      SUBROUTINE LINACL(T,M,NTFORC,CURPOS,ACCEL)
C     (* 902 3 -   COMPUTE LINEAR ACCELERATIONS IN X, Y, AND Z DIRECTIONS *)
C
C     CALLS
C        SIN,COS,MSCL,FLOAT,MADD
C     CALLED BY
C        STPRIM
C
C     INPUT
C        T,M,NTFORC,CURPOS
C     OUTPUT
C        ACCEL
C
      REAL ACCEL(3)
C        SPACECRAFT ACCELERATIONS
      REAL AGG(3)
C        ACCELERATIONS DUE TO GRAVITY GRADIANT
      REAL CURPOS(3)
C        CURRENT CHASE VEHICLE POSITION VECTOR
      REAL M
C        MASS OF SPACECRAFT IS M
      REAL NTFORC(3)
C        NET FORCE FROM THRUSTER OPERATION
      REAL SWT,CWT,A,B
C        THESE ARE INTERMEDIATE RESULTS
      REAL T
C        ELAPSED TIME
      REAL WO
C        RELATIVE ANGULAR ACCELERATION IN RADS/SEC
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* PRECOMPUTE VALUES USED SEVERAL PLACES *)
         WO=1.1629E-3
         A=1.338355E-6
         B=WO*T
         SWT=SIN(B)
         CWT=COS(B)
C     (* COMPUTE GRAVITY GRADIENT ACCELERATION *)
         AGG(1)=A*(3.*(CURPOS(1)*CWT+CURPOS(3)*SWT)*CWT-CURPOS(1))
         AGG(2)=-A*CURPOS(2)
         AGG(3)=A*(3.*(CURPOS(1)*CWT+CURPOS(3)*SWT)*SWT-CURPOS(3))
C     (* COMPUTE NET ACCELERATION (GRAVITY PLUS THRUSTERS) *)
         CALL MSCL(ACCEL,NTFORC,3,1,1.0/FLOAT(M))
         CALL MADD(ACCEL,ACCEL,ACC,3,1)
      RETURN
C
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

C-2

```
C
      SUBROUTINE TORQUE(F,N,TRNA)
C     (* 902 4 -  CALCULATE TORQUE (N) *)
C
C     CALLS
C        MMLT
C     CALLED BY
C        STPRIM
C
C     INPUT
C        AK3,F,TRNA
C     OUTPUT
C        N
C
      REAL AK3(3,14)
C        MATRIX RELATING THRUSTER INPUTS TO TORQUES IN 'CV' REF  FRAME
      REAL F(14)
C        FORCES FROM THRUSTERS AFTER SELECTION
      REAL N(3)
C        TORQUE ON SPACECRAFT ABOUT ITS CENTER OF MASS IN TRUTH COORD  SYS
      REAL N1(3)
C        TORQUE IN C V  REFERENCE FRAME
      REAL TRNA(3,3)
C        TRANSPOSE OF DIRECTION COSINE MATRIX THAT GIVES C V  ATTITUDE
C
      COMMON/VEHIC/DUMMY1(56),AK3,DUMMY2(15)
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* CALCULATE N=TRNA*AK3*F *)
         CALL MMLT(N1,AK3,F,3,14,1)
         CALL MMLT(N,TRNA,N1,3,3,1)
      RETURN
C
      END
```

```
C
      SUBROUTINE LPRIME(N,DLDT)
C     (* 902 5 -  COMPUTE TIME DERIVATIVF OF ANGULAR MOMENTUM *)
C
C     CALLS
C        <NONE>
C     CALLED BY
C        STPRIM
C
C     INPUT
C        N
C     OUTPUT
C        DLDT
C
      REAL DLDT(3)
C        TIME DERIVATIVE OF ANGULAR MOMENTUM VECTOR
      INTEGER I
C        DO-LOOP INDEX
      REAL N(3)
C        TORQUE ON C V  ABOUT ITS CENTER OF MASS IN 'TRUTH' COORDINATE SYSTEM
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
      DO 10 I=1,3
         DLDT(I)=N(I)
   10 CONTINUE
      RETURN
C
      END
```

```
C
      SUBROUTINE MAKROT(BODVEL,OMEGA)
C     (* 902 6 -  FORM ROTATION MATRIX OMEGA FROM ANGULAR VELOCITY VECTOR *)
C
C     CALLS
C        <NONE>
C     CALLED BY
C        STPRIM
C
C     INPUT
C        ANGVEL
C     OUTPUT
C        OMEGA
C
      REAL BODVEL(3)
C        ANGULAR VELOCITY IN CV COORDINATE SYSTEM
      INTEGER I
C        DO LOOP INDEX
      REAL OMEGA(4,4)
C        ROTATION MATRIX OMEGA, TO BE APPLIED TO QUATERNION
C        TO FORM TIME DERIVATIVE OF QUATERNION
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
      DO 10 I=1,4
         OMEGA(I,I)=0 0
   10 CONTINUE
      OMEGA(1,2)=BODVEL(3)
      OMEGA(1,3)=-BODVEL(2)
      OMEGA(1,4)=BODVEL(1)
      OMEGA(2,1)=-BODVEL(3)
      OMEGA(2,3)=BODVEL(1)
      OMEGA(2,4)=BODVEL(2)
      OMEGA(3,1)=BODVEL(2)
      OMEGA(3,2)=-BODVEL(1)
      OMEGA(3,4)=BODVEL(3)
      OMEGA(4,1)=-BODVEL(1)
      OMEGA(4,2)=-BODVEL(2)
      OMEGA(4,3)=-BODVEL(3)
      RETURN
C
      END
```

```
C
      SUBROUTINE QPRIME(Q,OMEGA,QPRM)
C     (* 902 7 -  COMPUTE QPRM = TIME DERIVATIVE OF QUATERNION Q *)
C
C     CALLS
C        MMLT,MSCL
C     CALLED BY
C        STPRIM
C
C     INPUT
C        Q,OMEGA
C     OUTPUT
C        QPRM
C
      REAL OMEGA(4,4)
C        MATRIX FORMED FROM ANGULAR VELOCITY COMPONENTS
      REAL Q(4)
C        ATTITUDE QUATERNIION OF CHASE VEHICLE
      REAL QINT(4)
C        INTERMEDIATE VALUES, NO PROBLEM RELATED SIGNIFICANCE
      REAL QPRM(4)
C        TIME DERIVATIVE OF OF Q
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* COMPUTE TIME DERIVATIVE OF Q = 0 5*OMEGA*Q *)
         CALL MMLT(QINT,OMEGA,Q,4,4,1)
         CALL MSCL(QPRM,QINT,4,1,0 5)
      RETURN
C
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
      FUNCTION SQR1(X)
C     (* 903 - RETURNS SQUARE ROOT BUT ALLOWS SLIGHTLY NEGATIVE ARGUMENT *)
C
C     CALLS
C        SQRT,AMAX1
C     CALLED BY
C        FLASH,POSIT,QUATRN,ESTCOV,SETGOL,MPROVE,TOQUAT,ATMCOV,DOKCHK
C
C     INPUT
C        X
C     OUTPUT
C        SQR1 (FUNCTION VALUE ONLY)
C
      REAL X
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
      SQR1=SQRT(AMAX1(X,0.))
      RETURN
C
      END
```

```
C
      SUBROUTINE ANGVEC(INERTA,ANGMNT,BODVEL,A)
C     (* 904 -  COMPUTE TRUE ANGULAR VELOCITY VECTOR ANGVEL
C        IN CV BODY COORDINATE SYSTEM  *)
C
C     CALLS
C        MINV,MMLT
C     CALLED BY
C        STPRIM,IMU
C
C     INPUT
C        INERTA,ANGMNT,TERMNL,A
C     OUTPUT
C        BODVEL
C
      REAL A(3,3)
C        DIRECTION COSINE MATRIX (CHASE VEHICLE ATTITUDE WRT 'TRUTH' AXES)
      REAL ANGMNT(3)
C         ANGULAR MOMENTUM VECTOR
      REAL BODVEL(3)
C         CHASE VEHICLE ANGULAR VELOCITY IN CV COORDINATE SYSTEM
      REAL INERTA(3,3)
C        INERTIA OF CHASE VEHICLE (LOADED)
      REAL INVIN(3,3)
C         INVERSE OF MOMENT OF INERTIA TENSOR
      REAL TEMP1(3)
C        INTERMEDIATE RESULTS WITH NO PROBLEM-RELATED SIGNIFICANCE
      REAL WORK(4,6)
C        AN INTERMEDIATE WORKSPACE
      INTEGER IERR
C        ERROR FLAG (=0 IF NO ERROR)
      INTEGER TERMNL
C        FORTRAN LOGICAL UNIT NUMBER OF USER TERMINAL
C
      COMMON/SIMUL/TERMNL,IDUMMY
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* FORM INVERSE OF INERTIA TENSOR *)
         CALL MINV(INVIN,INERTA,3,WORK,4,6,IERR)
      IF(IERR NE 0) GO TO 10
C     (* CALCULATE BODVEL=INVIN*A*ANGMNT *)
         CALL MMLT(TEMP1,A,ANGMNT,3,3,1)
         CALL MMLT(BODVEL,INVIN,TEMP1,3,3,1)
      RETURN
C
   10 CONTINUE
C     (* REPORT ERROR CONDITION AND HALT *)
         WRITE(TERMNL,901)
         STOP
C
  901 FORMAT(' MATRIX INVERSION FAILURE IN SUBROUTINE ANGVEC')
C
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
      SUBROUTINE TRGATT(TRNAT, T)
C     (* 905 - COMPUTE TARGET ATTITUDE AS A FUNCTION OF TIME *)
C
C     CALLS
C        COS, SIN, MMLT
C     CALLED BY
C        FLASH, DOKCHK
C
C     INPUT
C        T
C     OUTPUT
C        TRNAT
C
      REAL CHGATT(3,3)
C        TRANSPOSE OF DIRECTION COSINE MATRIX REPRESENTING TRUE TARGET
C        ATTITUDE CHANGE
      INTEGER ITUMBL
C        TARGET TUMBLE AXIS--VALUE OF 1, 2, OR 3 REPRESENTS YAW,
C        PITCH, OR ROLL TUMBLE
      REAL PHI
C        ROTATION ANGLE
      REAL T
C        ELAPSED TIME
      REAL TRNAT(3,3)
C        TRANSPOSE OF TARGET'S TRUE-ATTITUDE DIRECTION COSINE MATRIX
      REAL TRNATO(3,3)
C        INITIAL VALUE OF TRNAT
      REAL TUMRAT
C        TUMBLE RATE (IN RAD/SEC)
C
      COMMON/ATINFO/TRNATO, ITUMBL, TUMRAT
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
      PHI=TUMRAT*T
C     (* SELECT APPROPRIATE SET OF FORMULAS FOR TARGET ATTITUDE *)
         GO TO (10,20,30), ITUMBL
   10    CONTINUE
C        (* COMPUTE ATTITUDE CHANGE RESULTING FROM YAW TUMBLE *)
            CHGATT(1,1)=COS(PHI)
            CHGATT(2,1)=SIN(PHI)
            CHGATT(3,1)=0.0
            CHGATT(1,2)=-SIN(PHI)
            CHGATT(2,2)=COS(PHI)
            CHGATT(3,2)=0.0
            CHGATT(1,3)=0.0
            CHGATT(2,3)=0.0
            CHGATT(3,3)=1.0
            GO TO 40
   20    CONTINUE
C        (* COMPUTE ATTITUDE CHANGE RESULTING FROM PITCH TUMBLE *)
            CHGATT(1,1)=COS(PHI)
            CHGATT(2,1)=0.0
            CHGATT(3,1)=-SIN(PHI)
            CHGATT(1,2)=0.0
            CHGATT(2,2)=1.0
            CHGATT(3,2)=0.0
            CHGATT(1,3)=SIN(PHI)
            CHGATT(2,3)=0.0
            CHGATT(3,3)=COS(PHI)
            GO TO 40
   30    CONTINUE
C        (* COMPUTE ATTITUDE CHANGE RESULTING FROM A ROLL TUMBLE *)
            CHGATT(1,1)=1.0
            CHGATT(2,1)=0.0
            CHGATT(3,1)=0.0
            CHGATT(1,2)=0.0
            CHGATT(2,2)=COS(PHI)
            CHGATT(3,2)=SIN(PHI)
            CHGATT(1,3)=0.0
            CHGATT(2,3)=-SIN(PHI)
            CHGATT(3,3)=COS(PHI)
   40 CONTINUE
C     (* COMPUTE TRUE TARGET ATTITUDE *)
         CALL MMLT(TRNAT, TRNATO, CHGATT, 3, 3, 3)
      RETURN
      END
```

```
C
      SUBROUTINE QMLT(QNET,Q1,Q2)
C     (* 906 - COMPUTE THE QUATERNION PRODUCT QNET = Q1 * Q2   *)
C
C     CALLS
C        <NONE>
C     CALLED BY
C        SETGOL,RPY,PRPESA
C
C     INPUT
C        Q1,Q2
C     OUTPUT
C        QNET
C
      REAL Q1(4),Q2(4),QNET(4)
C        QUATERNIONS TO BE MULTIPLIED AND RESULT QUATERNION
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
      QNET(1) = Q1(4)*Q2(1)-Q1(3)*Q2(2)+Q1(2)*Q2(3)+Q1(1)*Q2(4)
      QNET(2) = Q1(3)*Q2(1)+Q1(4)*Q2(2)-Q1(1)*Q2(3)+Q1(2)*Q2(4)
      QNET(3) =-Q1(2)*Q2(1)+Q1(1)*Q2(2)+Q1(4)*Q2(3)+Q1(3)*Q2(4)
      QNET(4) =-Q1(1)*Q2(1)-Q1(2)*Q2(2)-Q1(3)*Q2(3)+Q1(4)*Q2(4)
      RETURN
C
      END
```

```
C
      FUNCTION DPRD(A,B,LEN)
C     (* 907 - COMPUTE GENERALIZED DOT PRODUCT OF A,B *)
C
C     CALLS
C        <NONE>
C     CALLED BY
C        DOCK,FLASH,ESTCOV,SETGOL,RPY,ATMCOV,DOKCHK
C
C     INPUT
C        A,B,LEN
C     OUTPUT
C        <FUNCTION VALUE>
C
      INTEGER I
C        DO LOOP INDEX
      INTEGER LEN
C        DIMENSION OF A,B
      REAL A(LEN),B(LEN)
C        VECTORS TO BE MULTIPLIED
      REAL PROD
C        INTERMEDIATE RESULTS
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
      PROD=0.0
      DO 10 I = 1,LEN
         PROD=PROD+A(I)*B(I)
   10 CONTINUE
      DPRD=PROD
      RETURN
C
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY