A RAND NOTE

# Photogrammetric Algorithms and Software for Spacecraft Optical Imaging Systems

Tim R. Colvin

RAND

The research described in this report was sponsored by the Jet Propulsion Laboratory under Contract No. 958723.



Published 1992 by RAND
1700 Main Street, P.O. Box 2138, Santa Monica, CA 90407-2138

A RAND NOTE

N-3330-JPL

# Photogrammetric Algorithms and Software for Spacecraft Optical Imaging Systems

Tim R. Colvin

Prepared for the
Jet Propulsion Laboratory

RAND

## PREFACE

This Note describes photogrammetric algorithms and software that can be used with data obtained from spacecraft optical imaging systems. The algorithms and software solve for control point coordinates, camera orientation angles, the position of a target body's north pole, and the target body's rotation rate.

This work can be used to generate control point networks and ancillary data for target bodies, which can then be used to mosaic individual photographs into unified cartographic products. Hence, this work will be useful to many analysts in the planetary science community.

This Note is intended to be used as a brief reference, rather than an all-encompassing dissertation on the subject. The intended audience consists of planetary scientists familiar with the basic data and possessing an intimate knowledge of FORTRAN.

This work was done as part of the Galileo project for the Jet Propulsion Laboratory. It has been tested with early data from Galileo, viz., the encounter with the asteroid Gaspra, and with older Voyager data.

## SUMMARY

Photographic images taken by a spacecraft, and ancillary data for these images, are used to solve for control point variables (latitude, longitude, and radius), camera orientation angles (right ascension, declination, and twist), and target body pole variables (right ascension and declination of the north pole, and rotation rate).

This is done by first making pixel measurements on the images and transforming them to coordinates in the camera focal plane. Then, for the time at which an image was taken, spacecraft position, camera orientation angles, the position of the target body's north pole, the target body's rotation rate, control point position, and camera focal length are used to calculate corresponding coordinates in the camera focal plane. The observed and calculated coordinates are compared in a least-squares adjustment that solves for the above variables.

The mathematics of the photogrammetric least-squares procedure is described in detail. It is presented in such a manner that readers desiring to make modifications are accommodated.

The FORTRAN programs GALGAUSS and GALCGRAD are presented. They solve the normal equations resulting from the least-squares adjustment in different ways. One program employs Gaussian elimination, and the other uses the conjugate gradient method to cleverly solve large systems with minimal storage.

The appropriate input and output files for the two FORTRAN programs are discussed, and the programs themselves are extensively commented.

## CONTENTS

# 1. INTRODUCTION

Let's say I have in front of me a collection of images of some target body, e.g., the Moon, taken by a spacecraft with an optical imaging system. And let's say I want to deduce some things from these images. Specifically, I would like to know the latitude, longitude, and radius of certain features on the target body. These features are called control points. I would also like to know where the target body's north pole lies and how fast the body rotates. Finally, I want to know the precise direction the spacecraft's camera was pointing when it acquired an image.

These tasks are accomplished by first making pixel measurements on the images of the features in which I am interested. Because I know the focal length of the camera, I can transform these measurements to coordinates in the camera's focal plane. These are the observational data. I have obtained them using just the images and by knowing the focal length.

Remember our objectives from the first paragraph. I solve for these variables by comparing the observational data, in a least-squares fashion, with a set of calculated data. The calculated data are derived, in part, from approximations for the following variables: control point variables (latitude, longitude, and radius), camera orientation angles (right ascension, declination, and twist), and target body pole variables (right ascension and declination of the north pole, and rotation rate). Two other quantities are necessary to obtain our set of calculated data. They are the times at which the images were taken and the positions of the spacecraft at those times.

The least-squares fit of the calculated data to the observational data yields improvements to the approximated variables. I can continue to iterate in this manner until all variables converge. At this juncture I have a set of data that is self-consistent.

Section 2 presents the mathematics of the photogrammetric least-squares adjustment. I have attempted to be rigorous as well as general, so that a reader wishing to make modifications will have an easy time doing so.

Section 3 describes the programs GALGAUSS and GALCGRAD (program listings are in the appendices) and their inputs and outputs. Sample input and output files are given to demonstrate the formats of these files.

## 2. PHOTOGRAMMETRIC ALGORITHMS FOR SPACECRAFT OPTICAL IMAGING SYSTEMS

We begin with a collection of photographic images. Every image has associated with it a set of control points. Each control point on an image is coupled with a pair $(x_o, y_o)$, measured on the image and transformed to coordinates in the camera focal plane, and a latitude, longitude, and radius triplet $(\phi, \lambda, R)$.

The coordinates of a control point in a body-fixed frame can be given by

$$\mathbf{x} \equiv R\,(\cos\phi\cos\lambda,\ \cos\phi\sin\lambda,\ \sin\phi)^T \tag{1}$$

From (1),

$$\frac{\partial \mathbf{x}}{\partial \phi} = R(-\sin\phi\cos\lambda, -\sin\phi\sin\lambda, \cos\phi)^T$$

$$\frac{\partial \mathbf{x}}{\partial \lambda} = R(-\cos\phi\sin\lambda, \cos\phi\cos\lambda, 0)^T$$

$$\frac{\partial \mathbf{x}}{\partial R} = (\cos\phi\cos\lambda, \cos\phi\sin\lambda, \sin\phi)^T$$

Define the right ascension and declination of the target body's north pole by

$$\alpha \equiv \alpha_0 + \tilde{\alpha}(t) \tag{2}$$

$$\delta \equiv \delta_0 + \tilde{\delta}(t) \tag{3}$$

where t is the interval between the J2000 epoch and the time the image was taken, $\alpha_0$ and $\delta_0$ are constants, and $\tilde{\alpha}$ and $\tilde{\delta}$ are known functions of t.

Define the spin-axis-intertial to Earth-equatorial-inertial transformation matrix as

$$M \equiv \begin{bmatrix} -\sin\alpha & -\cos\alpha\sin\delta & \cos\alpha\cos\delta \\ \cos\alpha & -\sin\alpha\sin\delta & \sin\alpha\cos\delta \\ 0 & \cos\delta & \sin\delta \end{bmatrix} \tag{4}$$

From (2) and (4),

$$\frac{\partial M}{\partial \alpha_0} = \begin{bmatrix} -\cos\alpha & \sin\alpha\sin\delta & -\sin\alpha\cos\delta \\ -\sin\alpha & -\cos\alpha\sin\delta & \cos\alpha\cos\delta \\ 0 & 0 & 0 \end{bmatrix}$$

From (3) and (4),

$$\frac{\partial M}{\partial \delta_0} = \begin{bmatrix} 0 & -\cos\alpha\cos\delta & -\cos\alpha\sin\delta \\ 0 & -\sin\alpha\cos\delta & -\sin\alpha\sin\delta \\ 0 & -\sin\delta & \cos\delta \end{bmatrix}$$

Define the prime meridian of the target body by

$$w = w_0 + \dot{w}t + \tilde{w}(t) \tag{5}$$

where $w_0$ is a constant defining the zero meridian, $\dot{w}$ is the target body's rotation rate, and $\tilde{w}$ is a known function of t.

From (5),

$$\frac{\partial w}{\partial \dot{w}} = t$$

Define the body-fixed to spin-axis-inertial transformation matrix by

$$V \equiv \begin{bmatrix} \cos w & -\sin w & 0 \\ \sin w & \cos w & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{6}$$

From (6),

$$\frac{dV}{dw} = \begin{bmatrix} -\sin w & -\cos w & 0 \\ \cos w & -\sin w & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\frac{\partial V}{\partial \dot{w}} = \frac{dV}{dw} \bullet \frac{\partial w}{\partial \dot{w}}$$

The coordinates of a control point in the J2000 Earth equatorial inertial system are given by

$$\hat{\mathbf{s}} \equiv \mathrm{MV}\mathbf{x} \tag{7}$$

From (1)–(6),

$$\frac{\partial \hat{\mathbf{s}}}{\partial \mathrm{P}} = \mathrm{MV}\frac{\partial \mathbf{x}}{\partial \mathrm{P}}\ ,\ \mathrm{P} \in \{\phi,\ \lambda,\ \mathrm{R}\}$$

$$\frac{\partial \hat{\mathbf{s}}}{\partial \mathrm{P}} = \frac{\partial \mathrm{M}}{\partial \mathrm{P}}\mathrm{V}\mathbf{x}\ ,\ \mathrm{P} \in \{\alpha_0\ ,\ \delta_0\}$$

$$\frac{\partial \hat{\mathbf{s}}}{\partial \dot{\mathrm{w}}} = \mathrm{M}\frac{\partial \mathrm{V}}{\partial \dot{\mathrm{w}}}\mathbf{x}$$

Define $\mathbf{s}$ as the spacecraft position vector in J2000 coordinates. The range vector, i.e., the vector subtended between the spacecraft and the control point, is then given by

$$\mathbf{s}^{\mathrm{v}} \equiv \hat{\mathbf{s}} - \mathbf{s} \tag{8}$$

Then it is clear that

$$\frac{\partial \mathbf{s}^{\mathrm{v}}}{\partial \mathrm{P}} = \frac{\partial \hat{\mathbf{s}}}{\partial \mathrm{P}}\ ,\ \mathrm{P} \in \{\phi,\ \lambda,\ \mathrm{R},\ \alpha_0,\ \delta_0,\ \dot{\mathrm{w}}\}$$

Define the camera orientation matrix, which transforms Earth equatorial inertial coordinates to camera coordinates, by

$$\mathrm{C} \equiv \begin{bmatrix} -\sin\hat{\alpha}\cos\kappa - \cos\hat{\alpha}\sin\hat{\delta}\sin\kappa & \cos\hat{\alpha}\cos\kappa - \sin\hat{\alpha}\sin\hat{\delta}\sin\kappa & \cos\hat{\delta}\sin\kappa \\ \sin\hat{\alpha}\sin\kappa - \cos\hat{\alpha}\sin\hat{\delta}\cos\kappa & -\cos\hat{\alpha}\sin\kappa - \sin\hat{\alpha}\sin\hat{\delta}\cos\kappa & \cos\hat{\delta}\cos\kappa \\ \cos\hat{\alpha}\cos\hat{\delta} & \sin\hat{\alpha}\cos\hat{\delta} & \sin\hat{\delta} \end{bmatrix}\ , \tag{9}$$

where $\hat{\alpha}$ and $\hat{\delta}$ define the direction of the optical axis and $\kappa$ specifies the picture rotation angle in the camera focal plane system.

From (9),

$$\frac{\partial C}{\partial \hat{\alpha}} = \begin{bmatrix} -\cos\hat{\alpha}\cos\kappa + \sin\hat{\alpha}\sin\hat{\delta}\sin\kappa & -\sin\hat{\alpha}\cos\kappa - \cos\hat{\alpha}\sin\hat{\delta}\sin\kappa & 0 \\ \cos\hat{\alpha}\sin\kappa + \sin\hat{\alpha}\sin\hat{\delta}\cos\kappa & \sin\hat{\alpha}\sin\kappa - \cos\hat{\alpha}\sin\hat{\delta}\cos\kappa & 0 \\ -\sin\hat{\alpha}\cos\hat{\delta} & \cos\hat{\alpha}\cos\hat{\delta} & 0 \end{bmatrix}$$

$$\frac{\partial C}{\partial \hat{\delta}} = \begin{bmatrix} -\cos\hat{\alpha}\cos\hat{\delta}\sin\kappa & -\sin\hat{\alpha}\cos\hat{\delta}\sin\kappa & -\sin\hat{\delta}\sin\kappa \\ -\cos\hat{\alpha}\cos\hat{\delta}\cos\kappa & -\sin\hat{\alpha}\cos\hat{\delta}\cos\kappa & -\sin\hat{\delta}\cos\kappa \\ -\cos\hat{\alpha}\sin\hat{\delta} & -\sin\hat{\alpha}\sin\hat{\delta} & \cos\hat{\delta} \end{bmatrix}$$

$$\frac{\partial C}{\partial \kappa} = \begin{bmatrix} \sin\hat{\alpha}\sin\kappa - \cos\hat{\alpha}\sin\hat{\delta}\cos\kappa & -\cos\hat{\alpha}\sin\kappa - \sin\hat{\alpha}\sin\hat{\delta}\cos\kappa & \cos\hat{\delta}\cos\kappa \\ \sin\hat{\alpha}\cos\kappa + \cos\hat{\alpha}\sin\hat{\delta}\sin\kappa & -\cos\hat{\alpha}\cos\kappa + \sin\hat{\alpha}\sin\hat{\delta}\sin\kappa & -\cos\hat{\delta}\sin\kappa \\ 0 & 0 & 0 \end{bmatrix}$$

Define $\gamma \equiv (\xi, \eta, \zeta)^T$ as the range vector in camera coordinates,

$$\gamma \equiv C\, s^v \tag{10}$$

By (8) and (10),

$$\frac{\partial \gamma}{\partial P} = C\frac{\partial s^v}{\partial P}\,,\ P \in \{\phi, \lambda, R, \alpha_0, \delta_0, \dot{w}\}$$

$$\frac{\partial \gamma}{\partial P} = \frac{\partial C}{\partial P}\, s^v\,,\ P \in \{\hat{\alpha}, \hat{\delta}, \kappa\}$$

Define the coordinates of a control point in the camera focal plane by

$$x_c \equiv \frac{\xi f}{\zeta}\,,\ y_c \equiv \frac{\eta f}{\zeta}\quad, \tag{11}$$

where f is the focal length of the camera. From (11),

$$\left.\begin{aligned} \frac{\partial x_c}{\partial P} &= \frac{f}{\zeta^2}\left(\zeta\frac{\partial \xi}{\partial P} - \xi\frac{\partial \zeta}{\partial P}\right) \\ \frac{\partial y_c}{\partial P} &= \frac{f}{\zeta^2}\left(\zeta\frac{\partial \eta}{\partial P} - \eta\frac{\partial \zeta}{\partial P}\right) \end{aligned}\right\} P \in \{\phi, \lambda, R, \alpha_0, \delta_0, \dot{w}, \hat{\alpha}, \hat{\delta}, \kappa\}$$

Suppose there are a total of n control points, p pictures, m measurements and $k = 3n + 3p + 3$ unknowns.

Define by $\mathbf{e}_i$ the coordinate axis vectors in $\Re^k$, i.e.,

$$\mathbf{e}_1 = (1, 0, \cdots, 0)^T, \cdots, \mathbf{e}_k = (0, \cdots, 0, 1)^T$$

and define $\mathbf{u} \equiv \left(x_c^{(1)}, y_c^{(1)}, \cdots, x_c^{(m)}, y_c^{(m)}\right)^T$ .

Then the observation equations may be written in the form

$$\mathbf{Lz} = \mathbf{f} \tag{12}$$

where

$$\mathbf{Le}_1 = \frac{\partial \mathbf{u}}{\partial \phi_1}, \mathbf{Le}_2 = \frac{\partial \mathbf{u}}{\partial \lambda_1}, \mathbf{Le}_3 = \frac{\partial \mathbf{u}}{\partial R_1}$$

...

$$\mathbf{Le}_{3n-2} = \frac{\partial \mathbf{u}}{\partial \phi_n}, \mathbf{Le}_{3n-1} = \frac{\partial \mathbf{u}}{\partial \lambda_n}, \mathbf{Le}_{3n} = \frac{\partial \mathbf{u}}{\partial R_n}$$

$$\mathbf{Le}_{3n+1} = \frac{\partial \mathbf{u}}{\partial \hat{\alpha}_1}, \mathbf{Le}_{3n+2} = \frac{\partial \mathbf{u}}{\partial \hat{\delta}_1}, \mathbf{Le}_{3n+3} = \frac{\partial \mathbf{u}}{\partial \kappa_1}$$

...

$$\mathbf{Le}_{3n+3p-2} = \frac{\partial \mathbf{u}}{\partial \hat{\alpha}_p}, \mathbf{Le}_{3n+3p-1} = \frac{\partial \mathbf{u}}{\partial \hat{\delta}_p}, \mathbf{Le}_{3n+3p} = \frac{\partial \mathbf{u}}{\partial \kappa_p}$$

$$\mathbf{Le}_{3n+3p+1} = \frac{\partial \mathbf{u}}{\partial \alpha_0}, \mathbf{Le}_{3n+3p+2} = \frac{\partial \mathbf{u}}{\partial \delta_0}, \mathbf{Le}_{3n+3p+3} = \frac{\partial \mathbf{u}}{\partial \dot{w}}$$

$$\mathbf{z} = \left(\Delta\phi_1, \Delta\lambda_1, \Delta R_1, \cdots, \Delta\phi_n, \Delta\lambda_n, \Delta R_n, \Delta\hat{\alpha}_1, \Delta\hat{\delta}_1, \Delta\kappa_1, \cdots,\right.$$

$$\left.\Delta\hat{\alpha}_p, \Delta\hat{\delta}_p, \Delta\kappa_p, \Delta\alpha_0, \Delta\delta_0, \Delta\dot{w}\right)^T$$

If one does not wish to solve for some of the variables, then the observation equations degenerate in an obvious manner. Note also that a special case of the above equations allows for the solution of a single body-wide radius rather than a radius at each control point.

The normal equations are formed by multiplying both sides of (12) by $L^T$ to get

$$\left(L^T L\right) \mathbf{z} = L^T \mathbf{f} \tag{13}$$

Solving (13) gives the desired corrections, viz.,

$$\left.\begin{aligned} \phi_j^{(i+1)} &= \phi_j^{(i)} + \Delta\phi_j^{(i)} \\ \lambda_j^{(i+1)} &= \lambda_j^{(i)} + \Delta\lambda_j^{(i)} \\ R_j^{(i+1)} &= R_j^{(i)} + \Delta R_j^{(i)} \end{aligned}\right\} j = 1, 2, \cdots, n$$

$$\left.\begin{aligned} \hat{\alpha}_j^{(i+1)} &= \hat{\alpha}_j^{(i)} + \Delta\hat{\alpha}_j^{(i)} \\ \hat{\delta}_j^{(i+1)} &= \hat{\delta}_j^{(i)} + \Delta\hat{\delta}_j^{(i)} \\ \kappa_j^{(i+1)} &= \kappa_j^{(i)} + \Delta\kappa_j^{(i)} \end{aligned}\right\} j = 1, 2, \cdots, p$$

$$\begin{aligned} \alpha_0^{(i+1)} &= \alpha_0^{(i)} + \Delta\alpha_0^{(i)} \\ \delta_0^{(i+1)} &= \delta_0^{(i)} + \Delta\delta_0^{(i)} \\ \dot{w}^{(i+1)} &= \dot{w}^{(i)} + \Delta\dot{w}^{(i)} \end{aligned}$$

Iteration is continued until the variables no longer change.

## 3. SOFTWARE DESCRIPTION

This section outlines the three input files used by the programs GALGAUSS and GALCGRAD. Using the programs is easy if one has a good understanding of the input.

GALGAUSS uses LINPACK routines to solve the normal system by Gaussian elimination. One half of the Gaussian elimination procedure is an LU decomposition of the normal matrix, so it is simple to backsolve for standard errors of all variables in the normal system. These errors can be computed and printed by GALGAUSS.

Both programs print four measurement errors. The first and second are root-mean-square errors of each coordinate. The third is a root-mean-square error of both coordinates, and the fourth factors in the number of variables in the normal system.

GALCGRAD solves large normal systems with minimal storage. By using the conjugate gradient method and storing the non-zero upper-triangular elements of the normal matrix in packed form, GALCGRAD can solve larger problems than GALGAUSS. GALCGRAD prints the time when every one-hundredth iteration of the conjugate gradient method is performed so one can easily assess when the program will terminate.

Both programs pass arrays with variable dimensions. Upper bounds for these dimensions must be specified in the BLOCK DATA and PARAMETER statements. The dimensions required are maximum number of pictures, maximum number of points and maximum number of variables in the normal system. GALCGRAD requires that the maximum number of non-zero upper-triangular elements in the normal matrix also be specified.

Define:

$P \equiv$ number of points

$O \equiv$ number of pictures

$M \equiv$ number of measurements

$n_p \equiv$ number of types of point variables for which a solution is required

$n_o \equiv$ number of picture variables for which a solution is required

$n_r \equiv$ number of types of pole variables for which a solution is required

then the number of variables in the normal system is given by

$$P \bullet n_p + o \bullet n_o + n_r$$

and an upper bound on the number of non-zero upper-triangular elements in the normal matrix is

$$\frac{Pn_p(n_p+1)}{2}+\frac{On_o(n_o+1)}{2}+\frac{n_r(n_r+1)}{2}+Pn_pn_r+On_on_r+Mn_pn_o$$

There are three input files necessary to run the programs - a parameter file; a pole, points, and picture file; and a measurement file.

There are two possible output files, viz., an output pole, points and picture file containing improved values, and an output listing file.

## THE PARAMETER FILE

```
NPIC,NPOI,NMEA,NIT,IOUT,NTOT,NSW,ISOL,IEW,LIST          FORMAT(10I5) {GALGAUSS}
NPIC,NPOI,NMEA,NIT,IOUT,NTOT,NSW,ISOL,IEW,LIST,NFIRST  FORMAT(11I5) {GALCGRAD}

BODY                                                    FORMAT(A10)

FOR I=1 TO I=NTOT:
IDTOT(I),GWEIGHT(I)                                     FORMAT(2I5)

FOR I=1 TO I=NSW:
ISWT,SWEIGHT(ISWT)                                      FORMAT(2I5)
```

NPIC : number of pictures
NPOI : number of points
NMEA : number of measurements
NIT : number of times the normal system is updated and re-solved
IOUT : if IOUT=0 then an output pole, points and picture file is not produced
if IOUT=1 then an output pole, points and picture file is produced
NTOT : number of types of variables for which a solution is required
The variable types are:
1. latitudes of control points
2. longitudes of control points
3. radii of control points
4. right ascensions of optical axes of pictures
5. declinations of optical axes of pictures
6. twist angles of pictures
7. right ascension of target body north pole
8. declination of target body north pole
9. target body rotation rate
NSW : number of single variables to be weighted
ISOL : if ISOL=1 then individual radii are used at control points
if ISOL=2 then a single body-wide radius is used
IEW : if IEW=0 then east longitudes are used
if IEW=1 then west longitudes are used
LIST : if LIST=0 then output is directed to the terminal
if LIST=1 then output is directed to a user-specified file
NFIRST : number of conjugate-gradient iterations. This is usually set to twice the number of variables in the normal system.
BODY : name of the target body (all capitals and left justified)
IDTOT(I) : variable type for which a solution is required
GWEIGHT(I) : weight for all variables of the above type
ISWT : single variable for which a weight is required
SWEIGHT(ISWT) : weight to apply to the above variable

Both programs use a seemingly-odd system of weighting variables in the normal system. This is not the typical weighting of observations associated with least-squares formulations. It is merely a simple way to keep specified variables from changing too rapidly and causing the solution to diverge. When these weights are present, one is solving a perturbed normal system, so the programs must ultimately be run with no weights. This is done after the solution settles down. The weights used are exponents of the number ten. They are added to the appropriate diagonal element in the normal matrix. Weights are between -38 (no weight) and 15 (variable will not change).

## SAMPLE PARAMETER FILE

```
    7    6    22    3    1    4    0    1    0    1    28
CALLISTO
    1  -38
    2  -38
    7  -38
    8  -38
```

## THE POLE, POINTS AND PICTURE FILE

```
ALPHA0,DELTA0,WDOT                FORMAT(3D24.16)

FOR I=1 TO I=NPOI:
PHI(I),LAM(I),R(I),POINT(I)       FORMAT(3D24.15,2X,A5)

FOR I=1 TO I=NPIC:
JDATE(I),FDS(I)                   FORMAT(D24.16,2X,A10)
S(I,1),S(I,2),S(I,3)              FORMAT(3D24.16)
C(I,1),C(I,2),C(I,3)              FORMAT(3D24.16)
```

```
  ALPHA0 : right ascension of target body north pole in degrees
  DELTA0 : declination of target body north pole in degrees
    WDOT : target body rotation rate in degrees/day
  PHI(I) : latitude of control point in degrees
  LAM(I) : longitude of control point in degrees
    R(I) : radius of control point in kilometers
POINT(I) : control point identifier
JDATE(I) : Julian date when picture was taken
  FDS(I) : Flight Data Sequence number of picture
  S(I,1) : X component of spacecraft position vector in J2000 coordinates
           in meters
  S(I,2) : Y component of spacecraft position vector in J2000 coordinates
           in meters
  S(I,3) : Z component of spacecraft position vector in J2000 coordinates
           in meters
  C(I,1) : right ascension of optical axis of picture in degrees
  C(I,2) : declination of optical axis of picture in degrees
  C(I,3) : twist angle of picture in degrees
```

## SAMPLE POLE, POINTS AND PICTURE FILE

```
0.2687200000000000D+03  0.6483000000000000D+02  0.2157107150000000D+02
0.3952580000000000D+02  0.1797010000000000D+02  0.2408728000000000D+04     141
0.4371760000000000D+02  0.1728290000000000D+02  0.2408728000000000D+04     162
0.7851200000000000D+01  0.7274850000000000D+02  0.2408728000000000D+04     395
0.3178040000000000D+02  0.4174570000000000D+02  0.2408728000000000D+04     404
0.2635870000000000D+02  0.4631290000000000D+02  0.2408728000000000D+04     405
0.6522100000000000D+01  0.7153040000000000D+02  0.2408728000000000D+04     549
0.2443939083843310D+07      1642147                          JULIAN DATE&FDS
0.1787394299149900D+06 -0.8207424367927500D+05  0.8407637231401100D+05 SXSYSZ
0.1555201397182000D+03 -0.2334418175330000D+02  0.1339381124799000D+03 C1C2C3
0.2443939086065532D+07      1642151                          JULIAN DATE&FDS
0.1759448830885200D+06 -0.8140612164396100D+05  0.8465640197376300D+05 SXSYSZ
0.1555165780275000D+03 -0.2396345459940000D+02  0.1337459342270000D+03 C1C2C3
0.2443939088287766D+07      1642155                          JULIAN DATE&FDS
0.1731506273984900D+06 -0.8073803026414600D+05  0.8523639636148300D+05 SXSYSZ
0.1555340463569000D+03 -0.2456883634240000D+02  0.1335787870962000D+03 C1C2C3
0.2443939092730822D+07      1642203                          JULIAN DATE&FDS
0.1675643925122800D+06 -0.7940217869537900D+05  0.8639569138645000D+05 SXSYSZ
0.1547190766007000D+03 -0.2477769388190000D+02  0.1327370135619000D+03 C1C2C3
0.2443939094954444D+07      1642207                          JULIAN DATE&FDS
0.1647689395286700D+06 -0.7873359996571500D+05  0.8697575055915700D+05 SXSYSZ
0.1547572697554000D+03 -0.2556543508440000D+02  0.1325216205701000D+03 C1C2C3
0.2443939097176667D+07      1642211                          JULIAN DATE&FDS
0.1619754080405400D+06 -0.7806540811080500D+05  0.8755533234928400D+05 SXSYSZ
0.1547880101907000D+03 -0.2621317221640000D+02  0.1322491608156000D+03 C1C2C3
0.2443939099398889D+07      1642215                          JULIAN DATE&FDS
0.1591820604451300D+06 -0.7739719331583000D+05  0.8813481501770200D+05 SXSYSZ
0.1548473284289000D+03 -0.2696062950880000D+02  0.1322053340655000D+03 C1C2C3
```

## THE MEASUREMENTS FILE

```
FOR I=1 TO I=NMEA
FDSC,F,POINTC,XO,YO      FORMAT(A10,F15.5,2X,A5,2F15.5)
```

```
  FDSC : Flight Data Sequence number of picture on which measurement
         was made
     F : focal length of camera with which the picture was taken,
         in millimeters
POINTC : control point identifier
    XO : x camera focal plane coordinate of measurement
    YO : y camera focal plane coordinate of measurement
```

**SAMPLE MEASUREMENTS FILE**

| | | | | |
|---|---|---|---|---|
| 1642147 | 1500.19000 | 141 | 4.45230 | -4.25110 |
| 1642147 | 1500.19000 | 162 | 5.19160 | -5.26920 |
| 1642147 | 1500.19000 | 404 | -1.17360 | -5.27240 |
| 1642147 | 1500.19000 | 405 | -2.85280 | -4.66170 |
| 1642151 | 1500.19000 | 395 | -2.62500 | -5.03350 |
| 1642151 | 1500.19000 | 404 | 4.67700 | -5.56980 |
| 1642151 | 1500.19000 | 405 | 2.96740 | -4.98400 |
| 1642151 | 1500.19000 | 549 | -2.72170 | -4.49970 |
| 1642155 | 1500.19000 | 395 | 3.12350 | -4.51730 |
| 1642155 | 1500.19000 | 549 | 3.02780 | -4.01780 |
| 1642203 | 1500.19000 | 141 | -5.37900 | 1.39170 |
| 1642203 | 1500.19000 | 162 | -4.59440 | 0.32960 |
| 1642207 | 1500.19000 | 141 | 3.91480 | 0.02740 |
| 1642207 | 1500.19000 | 162 | 4.70210 | -1.04260 |
| 1642207 | 1500.19000 | 404 | -2.00330 | -1.22140 |
| 1642207 | 1500.19000 | 405 | -3.76210 | -0.65230 |
| 1642211 | 1500.19000 | 395 | -3.28250 | 0.11490 |
| 1642211 | 1500.19000 | 404 | 4.28370 | 0.02390 |
| 1642211 | 1500.19000 | 405 | 2.49650 | 0.60230 |
| 1642211 | 1500.19000 | 549 | -3.40530 | 0.67310 |
| 1642215 | 1500.19000 | 395 | 5.15220 | 0.33500 |
| 1642215 | 1500.19000 | 549 | 5.04100 | 0.89130 |

## SAMPLE OUTPUT LISTING FILE

```
 INITIAL CONFIGURATION FOR THESE DATA

  NPIC =      7    NPOI =      6    NMEA =     22    NUNK =     14

SOLVING FOR CORRECTIONS TO LATITUDES OF POINTS
SOLVING FOR CORRECTIONS TO LONGITUDES OF POINTS
SOLVING FOR CORRECTION TO RIGHT ASCENSION OF POLE
SOLVING FOR CORRECTION TO DECLINATION OF POLE

                                        WEIGHTS
-38 -38 -38 -38 -38 -38 -38 -38 -38 -38 -38 -38 -38 -38

                 PICTURE  POINT         XO-XC          YO-YC

                 1642147    141       -0.01629        0.02377
                 1642147    162       -0.00423        0.01637
                 1642147    404        0.00393       -0.00904
                 1642147    405        0.01153       -0.00075
                 1642151    395        0.00422        0.00514
                 1642151    404       -0.00023        0.01003
                 1642151    405       -0.00520       -0.00160
                 1642151    549        0.00251        0.01956
                 1642155    395        0.00364        0.01804
                 1642155    549        0.00570       -0.00818
                 1642203    141        0.00735        0.00613
                 1642203    162        0.01461       -0.00365
                 1642207    141       -0.00838        0.00609
                 1642207    162       -0.01171       -0.00114
                 1642207    404       -0.01173       -0.00263
                 1642207    405        0.00769       -0.00260
                 1642211    395        0.00646        0.00019
                 1642211    404       -0.00665       -0.01207
                 1642211    405        0.00350        0.00733
                 1642211    549       -0.00400        0.00707
                 1642215    395       -0.00165        0.00992
                 1642215    549        0.00154        0.00797

SIGL =      0.00774 SIGM =      0.01030 SIG1 =      0.00911 SIG2 =      0.01104

                 PICTURE  POINT         XO-XC          YO-YC

                 1642147    141        0.27893       -0.12516
                 1642147    162        0.30324       -0.09521
                 1642147    404        0.13666       -0.14857
                 1642147    405        0.10428       -0.13616
                 1642151    395       -0.02342       -0.01243
                 1642151    404        0.13496       -0.12891
                 1642151    405        0.08948       -0.13678
                 1642151    549       -0.01690        0.00684
                 1642155    395       -0.02352       -0.00069
                 1642155    549       -0.01328       -0.02179
                 1642203    141        0.32050       -0.13522
                 1642203    162        0.34052       -0.10596
                 1642207    141        0.30938       -0.13306
                 1642207    162        0.31895       -0.10082
                 1642207    404        0.13380       -0.13817
                 1642207    405        0.11058       -0.13607
                 1642211    395       -0.01851       -0.02353
                 1642211    404        0.14163       -0.14643
                 1642211    405        0.10860       -0.12545
                 1642211    549       -0.02106       -0.01027
                 1642215    395       -0.02610       -0.01500
```

| | | | |
|---|---|---|---|
| 1642215 | 549 | -0.01507 | -0.01026 |

SIGL = 0.17925 SIGM = 0.10324 SIG1 = 0.14627 SIG2 = 0.17714

| PICTURE | POINT | XO-XC | YO-YC |
|---|---|---|---|
| 1642147 | 141 | -0.02661 | -0.00045 |
| 1642147 | 162 | -0.02236 | -0.00071 |
| 1642147 | 404 | -0.00657 | -0.01125 |
| 1642147 | 405 | -0.00513 | -0.00429 |
| 1642151 | 395 | -0.00576 | 0.00388 |
| 1642151 | 404 | -0.01050 | 0.00934 |
| 1642151 | 405 | -0.02171 | -0.00392 |
| 1642151 | 549 | -0.00551 | 0.02010 |
| 1642155 | 395 | -0.00610 | 0.01658 |
| 1642155 | 549 | -0.00212 | -0.00777 |
| 1642203 | 141 | -0.00200 | -0.01004 |
| 1642203 | 162 | -0.00279 | -0.01239 |
| 1642207 | 141 | -0.01752 | -0.00792 |
| 1642207 | 162 | -0.02897 | -0.00764 |
| 1642207 | 404 | -0.02118 | 0.00329 |
| 1642207 | 405 | -0.00830 | 0.00033 |
| 1642211 | 395 | -0.00228 | -0.00211 |
| 1642211 | 404 | -0.01594 | -0.00438 |
| 1642211 | 405 | -0.01239 | 0.01166 |
| 1642211 | 549 | -0.01096 | 0.00701 |
| 1642215 | 395 | -0.01013 | 0.00747 |
| 1642215 | 549 | -0.00521 | 0.00784 |

SIGL = 0.01399 SIGM = 0.00883 SIG1 = 0.01169 SIG2 = 0.01416

| PICTURE | POINT | XO-XC | YO-YC |
|---|---|---|---|
| 1642147 | 141 | -0.01181 | 0.00540 |
| 1642147 | 162 | -0.00486 | 0.00582 |
| 1642147 | 404 | 0.00667 | -0.01092 |
| 1642147 | 405 | 0.00625 | -0.00568 |
| 1642151 | 395 | 0.00030 | -0.00257 |
| 1642151 | 404 | 0.00287 | 0.00977 |
| 1642151 | 405 | -0.01022 | -0.00524 |
| 1642151 | 549 | 0.00031 | 0.01324 |
| 1642155 | 395 | 0.00002 | 0.01016 |
| 1642155 | 549 | 0.00376 | -0.01462 |
| 1642203 | 141 | 0.01338 | -0.00360 |
| 1642203 | 162 | 0.01540 | -0.00519 |
| 1642207 | 141 | -0.00200 | -0.00132 |
| 1642207 | 162 | -0.01060 | -0.00026 |
| 1642207 | 404 | -0.00730 | 0.00418 |
| 1642207 | 405 | 0.00365 | -0.00063 |
| 1642211 | 395 | 0.00406 | -0.00837 |
| 1642211 | 404 | -0.00192 | -0.00335 |
| 1642211 | 405 | -0.00034 | 0.01081 |
| 1642211 | 549 | -0.00487 | 0.00028 |
| 1642215 | 395 | -0.00377 | 0.00123 |
| 1642215 | 549 | 0.00092 | 0.00113 |

SIGL = 0.00685 SIGM = 0.00706 SIG1 = 0.00696 SIG2 = 0.00842

## SAMPLE OUTPUT POLE, POINTS AND PICTURES FILE

```
0.2745093866906236D+03  0.5452716273030049D+02  0.2157107150000000D+02
0.4641662717768745D+02  0.1533399599978890D+02  0.2408728000000000D+04     141
0.5044724025220162D+02  0.1328023937070833D+02  0.2408728000000000D+04     162
0.1843312988171790D+02  0.7808993821840431D+02  0.2408728000000000D+04     395
0.4158124786101637D+02  0.4354783242631321D+02  0.2408728000000000D+04     404
0.3650993732124200D+02  0.4919535252423913D+02  0.2408728000000000D+04     405
0.1712931650029616D+02  0.7680313270624947D+02  0.2408728000000000D+04     549
0.2443939083843310D+07      1642147                          JULIAN_DATE&FDS
0.1787394299149900D+06 -0.8207424367927500D+05  0.8407637231401100D+05 SXSYSZ
0.1555201397182000D+03 -0.2334418175330000D+02  0.1339381124799000D+03 C1C2C3
0.2443939086065532D+07      1642151                          JULIAN_DATE&FDS
0.1759448830885200D+06 -0.8140612164396100D+05  0.8465640197376300D+05 SXSYSZ
0.1555165780275000D+03 -0.2396345459940000D+02  0.1337459342270000D+03 C1C2C3
0.2443939088287766D+07      1642155                          JULIAN_DATE&FDS
0.1731506273984900D+06 -0.8073803026414600D+05  0.8523639636148300D+05 SXSYSZ
0.1555340463569000D+03 -0.2456883634240000D+02  0.1335787870962000D+03 C1C2C3
0.2443939092730822D+07      1642203                          JULIAN_DATE&FDS
0.1675643925122800D+06 -0.7940217869537900D+05  0.8639569138645000D+05 SXSYSZ
0.1547190766007000D+03 -0.2477769388190000D+02  0.1327370135619000D+03 C1C2C3
0.2443939094954444D+07      1642207                          JULIAN_DATE&FDS
0.1647689395286700D+06 -0.7873359996571500D+05  0.8697575055915700D+05 SXSYSZ
0.1547572697554000D+03 -0.2556543508440000D+02  0.1325216205701000D+03 C1C2C3
0.2443939097176667D+07      1642211                          JULIAN_DATE&FDS
0.1619754080405400D+06 -0.7806540811080500D+05  0.8755533234928400D+05 SXSYSZ
0.1547880101907000D+03 -0.2621317221640000D+02  0.1322491608156000D+03 C1C2C3
0.2443939099398889D+07      1642215                          JULIAN_DATE&FDS
0.1591820604451300D+06 -0.7739719331583000D+05  0.8813481501770200D+05 SXSYSZ
0.1548473284289000D+03 -0.2696062950880000D+02  0.1322053340655000D+03 C1C2C3
```

## Appendix A

## FORTRAN PROGRAM GALGAUSS

```
c
c define maximum parameters for variable arrays
c
      block data
      common/dims/mpoi,mpic,munk
      data mpoi/300/,mpic/100/,munk/1203/
      end
c
c define variables
c
      parameter(mpoi=300,mpic=100,munk=1203)
      implicit double precision (a-h,o-z)
      character*5 point(mpoi),pointc
      character*10 fds(mpic),fdsc,body
      character*12 dum
      double precision phi(mpoi),lam(mpoi),r(mpoi),s(mpic,3),c(mpic,3),
     1jdate(mpic),ltl(munk,munk),del(munk),l(9,2),pxc(9),pyc(9),jdatec,
     2unit(munk,1),errs(munk),ptbd(3,1),em(3,3),vee(3,3),cee(3,3),
     3z1(3,1),z2(3,1),z3(3,3),sn(3,1),gamma(3,1)
      integer idtot(9),gweight(9),sweight(munk),fpointer,ppointer,
     &par(9),ipvt(munk),num1(2,3),num2(2,3),num3(2,3)
      common/outs/npoi,npic,nmea,nunk,il
      common/extras/ptbd,em,vee,rc,jdatec,calfc,salfc,cdelc,sdelc,cphic,
     1sphic,clamc,slamc,dx,dy,fpointer,ppointer,cee,calfhc,salfhc,
     2cdelhc,sdelhc,ckapc,skapc,sn,gamma,alpha0,delta0,wdot,body
      common/const/rad,f
  100 format(a12)
  110 format(a10)
  120 format(10i5)
  130 format(3d24.16,2x,a5)
  140 format(d24.16,2x,a10)
  150 format(3d24.16)
  160 format(a10,f15.5,2x,a5,2f15.5)
```

```
  170 format(' NO SUPPORT DATA FOR PICTURE ',a10)
  180 format(' NO SUPPORT DATA FOR POINT ',a5)
  190 format('   NPIC = ',i5,2x,' NPOI = ',i5,2x,' NMEA = ',i5,2x,
     &' NUNK = ',i5)
  200 format(' ',36x,'WEIGHTS')
  210 format(' ',16x,'PICTURE',2X,'POINT',9X,'XO-XC',10X,'YO-YC')
  220 format(20i4)
  230 format(d24.16,2x,a10,28X,'JULIAN_DATE&FDS')
  240 format(3d24.16,1x,'SXSYSZ')
  250 format(3d24.16,1x,'C1C2C3')
  260 format(' SIGL = ',f12.5,1x,'SIGM = ',f12.5,1x,'SIG1 = ',
     &f12.5,1x,'SIG2 = ',f12.5)
  270 format(' ',2x,'PARAMETER',5x,'ONE SIGMA ERROR')
  280 format(' SOLVING FOR CORRECTIONS TO LATITUDES OF POINTS')
  290 format(' SOLVING FOR CORRECTIONS TO LONGITUDES OF POINTS')
  300 format(' SOLVING FOR CORRECTIONS TO RADII OF POINTS')
  310 format(' SOLVING FOR CORRECTION TO RADIUS')
  320 format(' SOLVING FOR CORRECTIONS TO ALPHA C-MATRIX ANGLES')
  330 format(' SOLVING FOR CORRECTIONS TO DELTA C-MATRIX ANGLES')
  340 format(' SOLVING FOR CORRECTIONS TO KAPPA C-MATRIX ANGLES')
  350 format(' SOLVING FOR CORRECTION TO RIGHT ASCENSION OF POLE')
  360 format(' SOLVING FOR CORRECTION TO DECLINATION OF POLE')
  370 format(' SOLVING FOR CORRECTION TO ROTATION RATE OF POLE')
  380 format(' NUMBER OF POINTS MUST BE GREATER IN PROGRAM DECLARATION')
  390 format(' NUMBER OF PICTURES MUST BE GREATER IN PROGRAM DECLARATIO
     &N')
  400 format(' NUMBER OF UNKNOWNS MUST BE GREATER IN PROGRAM DECLARATION
     &')
c
c define radians constant
c
      rad=1.8d2/3.141592653589793d0
c
c read parameter file
c
      write(6,*)' parameter file ?'
      read(5,100)dum
      open(11,file=dum,status='old')
```

```
c
c read number of pictures,number of points,number of measurements,number of
c iterations,output printing parameter,number of kinds of variables to be
c solved for,number of single parameters to be weighted,solution type,
c longitude type and output listing parameter
c
      read(11,120)npic,npoi,nmea,nit,iout,ntot,nsw,isol,iew,list
      il=6
         if(list.eq.1)then
         il=16
         write(6,*)' output listing file?'
         read(5,100)dum
         open(16,file=dum,status='new')
         endif
         if(npic.gt.mpic)then
         write(il,390)
         stop
         elseif(npoi.gt.mpoi)then
         write(il,380)
         stop
         endif
      nmea2=2*nmea
c
c read celestial body
c
      read(11,110)body
c
c read variable types to be solved for & their group weights
c 1:latitude of point;2:longitude of point;3:radius of point;
c 4:alpha angle of C matrix;5:delta angle of C matrix;
c 6:kappa angle of C matrix;7:right ascension of pole;8:declination of pole;
c 9:rotation rate of pole
c
         do 410 i=1,ntot
  410    read(11,120)idtot(i),gweight(i)
      write(6,*)' input file?'
      read(5,100)dum
      open(12,file=dum,status='old')
```

```
c
c read alpha sub zero and delta sub zero in degrees
c and rotation rate in degrees/day
c
      read(12,150)alpha0,delta0,wdot
      alpha0=alpha0/rad
      delta0=delta0/rad
      wdot=wdot/rad
c
c read points (radius is in meters)
c iew=0 ==> east longitude   iew=1 ==> west longitude
c
         do 420 i=1,npoi
         read(12,130)phi(i),lam(i),r(i),point(i)
            if(iew.eq.0)lam(i)=3.6d2-lam(i)
            if(isol.eq.2)r(i)=r(1)
         phi(i)=phi(i)/rad
  420    lam(i)=lam(i)/rad
c
c for each picture read :
c julian date and FDS
c spacecraft position(in m) in J2000 coordinates
c c matrix angles (alpha hat,delta hat,kappa) in degrees
c
         do 430 i=1,npic
         read(12,140)jdate(i),fds(i)
         read(12,150)s(i,1),s(i,2),s(i,3)
         read(12,150)c(i,1),c(i,2),c(i,3)
         c(i,1)=c(i,1)/rad
         c(i,2)=c(i,2)/rad
  430    c(i,3)=c(i,3)/rad
      close(12)
c
c count number of unknowns, number of variables to be solved for in three
c categories (points,pictures,pole parameters) and keep track of which ones are
c in a particular category
c
      nunk=0
```

```
      n13=0
         do 450 i=1,3
            do 440 j=1,ntot
               if(i.eq.idtot(j).and.idtot(j).eq.3.and.isol.eq.2)then
               n13=n13+1
               num1(1,n13)=j
               num1(2,n13)=i
               nunk=nunk+1
               elseif(i.eq.idtot(j))then
               n13=n13+1
               num1(1,n13)=j
               num1(2,n13)=i
               nunk=nunk+npoi
               endif
  440       continue
  450    continue
      n46=0
         do 470 i=4,6
            do 460 j=1,ntot
               if(i.eq.idtot(j))then
               n46=n46+1
               num2(1,n46)=j
               num2(2,n46)=i
               nunk=nunk+npic
               endif
  460       continue
  470    continue
      n79=0
         do 490 i=7,9
            do 480 j=1,ntot
               if(i.eq.idtot(j))then
               n79=n79+1
               num3(1,n79)=j
               num3(2,n79)=i
               nunk=nunk+1
               endif
  480       continue
  490    continue
```

```
          if(nunk.gt.munk)then
          write(il,400)
          stop
          endif
c
c set weights
c
          if(isol.eq.1)then
             do 510 i=1,n13
             num=num1(1,i)
                do 500 j=i,n13*(npoi-1)+i,n13
  500           sweight(j)=gweight(num)
  510        continue
          imark=n13*npoi
          else
             if(n13.eq.1)then
             sweight(1)=gweight(1)
             imark=1
             elseif(n13.eq.2)then
                do 530 i=1,2
                num=num1(1,i)
                   if(num1(2,i).ne.3)then
                      do 520 j=1,npoi
  520                 sweight(j)=gweight(num)
                   else
                   sweight(npoi+1)=gweight(num)
                   endif
  530           continue
             imark=npoi+1
             elseif(n13.eq.3)then
                do 560 i=1,3
                num=num1(1,i)
                   if(num1(2,i).eq.1)then
                      do 540 j=1,2*npoi-1,2
  540                 sweight(j)=gweight(num)
                   elseif(num1(2,i).eq.2)then
                      do 550 j=2,2*npoi,2
  550                 sweight(j)=gweight(num)
```

```
                  else
                  sweight(2*npoi+1)=gweight(num)
                  endif
  560           continue
              imark=2*npoi+1
              endif
            endif
            do 580 i=1,n46
            num=num2(1,i)
               do 570 j=imark+i,imark+n46*(npic-1)+i,n46
  570          sweight(j)=gweight(num)
  580       continue
          imark=imark+n46*npic
            do 590 i=1,n79
  590       sweight(imark+i)=gweight(num3(1,i))
c
c read and set single parameter weights
c
            do 600 i=1,nsw
  600       read(11,120)iswt,sweight(iswt)
          close(11)
c
c for output open pole,points,julian date,fds,spacecraft position and
c C matrix angles output file
c
            if(iout.eq.1)then
            write(6,*)' output file?'
            read(5,100)dum
            open(15,file=dum,status='new')
            endif
c
c open measurements file
c
          write(6,*)' measurements file ?'
          read(5,100)dum
          open(14,file=dum,status='old')
c
c read measurements;time is in Julian days,focal length is in millimeters
```

```
c
         do 640 i=1,nmea
         read(14,160)fdsc,f,pointc,xo,yo
c
c match fds to fds tag
c
            do 610 j=1,npic
               if(fdsc.eq.fds(j))then
               fpointer=j
               goto 620
               endif
  610       continue
         write(il,170)fdsc
         stop
c
c match point to point tag
c
  620       do 630 j=1,npoi
               if(pointc.eq.point(j))then
               ppointer=j
               goto 640
               endif
  630       continue
         write(il,180)pointc
         stop
c
c write tag info,focal length and coordinates to default fortran file
c
  640    write(19)ppointer,fpointer,f,xo,yo
      close(14)
         do 870 i=1,nit
c
c if first iteration print number of points,pictures,measurements,unknowns,
c things to be solved for and weights
c
            if(i.eq.1)then
            write(il,*)
            write(il,*)' INITIAL CONFIGURATION FOR THESE DATA'
```

```
            write(il,*)
            write(il,190)npic,npoi,nmea,nunk
            write(il,*)
               do 650 j=1,ntot
                  if(idtot(j).eq.1)write(il,280)
                  if(idtot(j).eq.2)write(il,290)
                  if(idtot(j).eq.3.and.isol.eq.1)write(il,300)
                  if(idtot(j).eq.3.and.isol.eq.2)write(il,310)
                  if(idtot(j).eq.4)write(il,320)
                  if(idtot(j).eq.5)write(il,330)
                  if(idtot(j).eq.6)write(il,340)
                  if(idtot(j).eq.7)write(il,350)
                  if(idtot(j).eq.8)write(il,360)
                  if(idtot(j).eq.9)write(il,370)
  650          continue
            write(il,*)
            write(il,200)
            write(il,220)(sweight(iz),iz=1,nunk)
            write(il,*)
            endif
         rewind(19)
         sigmax=0.d0
         sigmay=0.d0
c
c initialize the normal matrix and right-hand-side vector to zero
c
            do 670 iz=1,nunk
            del(iz)=0.d0
               do 660 jz=1,nunk
  660          ltl(iz,jz)=0.d0
  670       continue
         write(il,210)
         write(il,*)
c
c for each measurement:
c
            do 780 iz=1,nmea
            call respar(phi,lam,r,jdate,fds,s,c,point)
```

```
c
c point variable partials
c
               do 690 j=1,n13
               num=num1(2,j)
c
c partials wrt latitude
c
                  if(num.eq.1)then
                  z1(1,1)=-rc*sphic*clamc
                  z1(2,1)=-rc*sphic*slamc
                  z1(3,1)= rc*cphic
                  call matmul(vee,3,3,z1,1,z2)
                  call matmul(em,3,3,z2,1,z1)
                  call matmul(cee,3,3,z1,1,z2)
                  call pxcyc(z2,gamma,pxc(1),pyc(1))
c
c partials wrt longitude
c
                  elseif(num.eq.2)then
                  z1(1,1)=-rc*cphic*slamc
                  z1(2,1)= rc*cphic*clamc
                  z1(3,1)= 0.
                  call matmul(vee,3,3,z1,1,z2)
                  call matmul(em,3,3,z2,1,z1)
                  call matmul(cee,3,3,z1,1,z2)
                  call pxcyc(z2,gamma,pxc(2),pyc(2))
c
c partials wrt radius
c
                  else
                     do 680 k=1,3
  680                z1(k,1)=ptbd(k,1)/rc
                  call matmul(vee,3,3,z1,1,z2)
                  call matmul(em,3,3,z2,1,z1)
                  call matmul(cee,3,3,z1,1,z2)
                  call pxcyc(z2,gamma,pxc(3),pyc(3))
                  endif
```

```
  690          continue
c
c picture variable partials
c
               do 700 j=1,n46
               num=num2(2,j)
c
c partials wrt C matrix angle alpha hat
c
                  if(num.eq.4)then
                  z3(1,1)=-calfhc*ckapc+salfhc*sdelhc*skapc
                  z3(1,2)=-salfhc*ckapc-calfhc*sdelhc*skapc
                  z3(1,3)= 0.d0
                  z3(2,1)= calfhc*skapc+salfhc*sdelhc*ckapc
                  z3(2,2)= salfhc*skapc-calfhc*sdelhc*ckapc
                  z3(2,3)= 0.d0
                  z3(3,1)=              -salfhc*cdelhc
                  z3(3,2)=               calfhc*cdelhc
                  z3(3,3)= 0.d0
                  call matmul(z3,3,3,sn,1,z1)
                  call pxcyc(z1,gamma,pxc(4),pyc(4))
c
c partials wrt C matrix angle delta hat
c
                  elseif(num.eq.5)then
                  z3(1,1)=-calfhc*cdelhc*skapc
                  z3(1,2)=-salfhc*cdelhc*skapc
                  z3(1,3)=       -sdelhc*skapc
                  z3(2,1)=-calfhc*cdelhc*ckapc
                  z3(2,2)=-salfhc*cdelhc*ckapc
                  z3(2,3)=       -sdelhc*ckapc
                  z3(3,1)=-calfhc*sdelhc
                  z3(3,2)=-salfhc*sdelhc
                  z3(3,3)=        cdelhc
                  call matmul(z3,3,3,sn,1,z1)
                  call pxcyc(z1,gamma,pxc(5),pyc(5))
c
c partials wrt C matrix angle kappa
```

```
c
                  else
                  z3(1,1)= salfhc*skapc-calfhc*sdelhc*ckapc
                  z3(1,2)=-calfhc*skapc-salfhc*sdelhc*ckapc
                  z3(1,3)=                      cdelhc*ckapc
                  z3(2,1)= salfhc*ckapc+calfhc*sdelhc*skapc
                  z3(2,2)=-calfhc*ckapc+salfhc*sdelhc*skapc
                  z3(2,3)=                     -cdelhc*skapc
                  z3(3,1)= 0.d0
                  z3(3,2)= 0.d0
                  z3(3,3)= 0.d0
                  call matmul(z3,3,3,sn,1,z1)
                  call pxcyc(z1,gamma,pxc(6),pyc(6))
                  endif
  700          continue
c
c pole variable partials
c
               do 710 j=1,n79
c
c partials wrt right ascension of pole
c
                  if(num3(2,j).eq.7)then
                  z3(1,1)=-calfc
                  z3(1,2)= salfc*sdelc
                  z3(1,3)=-salfc*cdelc
                  z3(2,1)=-salfc
                  z3(2,2)=-calfc*sdelc
                  z3(2,3)= calfc*cdelc
                  z3(3,1)= 0.d0
                  z3(3,2)= 0.d0
                  z3(3,3)= 0.d0
                  call matmul(vee,3,3,ptbd,1,z1)
                  call matmul(z3,3,3,z1,1,z2)
                  call matmul(cee,3,3,z2,1,z1)
                  call pxcyc(z1,gamma,pxc(7),pyc(7))
c
c partials wrt declination of pole
```

```
c
                  elseif(num3(2,j).eq.8)then
                  z3(1,1)= 0.d0
                  z3(1,2)=-calfc*cdelc
                  z3(1,3)=-calfc*sdelc
                  z3(2,1)= 0.d0
                  z3(2,2)=-salfc*cdelc
                  z3(2,3)=-salfc*sdelc
                  z3(3,1)= 0.d0
                  z3(3,2)=      -sdelc
                  z3(3,3)=       cdelc
                  call matmul(vee,3,3,ptbd,1,z1)
                  call matmul(z3,3,3,z1,1,z2)
                  call matmul(cee,3,3,z2,1,z1)
                  call pxcyc(z1,gamma,pxc(8),pyc(8))
c
c partials wrt rotation rate
c
                  else
                  pwpwdot=jdatec-2451545.d0
                  z3(1,1)=-sw*pwpwdot
                  z3(1,2)=-cw*pwpwdot
                  z3(1,3)= 0.d0
                  z3(2,1)= cw*pwpwdot
                  z3(2,2)=-sw*pwpwdot
                  z3(2,3)= 0.d0
                  z3(3,1)= 0.d0
                  z3(3,2)= 0.d0
                  z3(3,3)= 0.d0
                  call matmul(z3,3,3,ptbd,1,z1)
                  call matmul(em,3,3,z1,1,z2)
                  call matmul(cee,3,3,z2,1,z1)
                  call pxcyc(z1,gamma,pxc(9),pyc(9))
                  endif
  710          continue
c
c compute partials of coordinates
c
```

```
            index=0
               if(isol.eq.1)then
                  do 720 j=1,n13
                  index=index+1
                  par(index)=n13*(ppointer-1)+j
                  num=num1(2,j)
                  l(index,1)=pxc(num)
720               l(index,2)=pyc(num)
               jmark=n13*npoi
               else
                  do 730 j=1,n13
                  index=index+1
                  num=num1(2,j)
                     if(num.eq.3)then
                     par(index)=(n13-1)*npoi+1
                     else
                     par(index)=(n13-1)*(ppointer-1)+j
                     endif
                  l(index,1)=pxc(num)
730               l(index,2)=pyc(num)
               jmark=(n13-1)*npoi+1
               endif
               do 740 j=1,n46
               index=index+1
               par(index)=jmark+n46*(fpointer-1)+j
               num=num2(2,j)
               l(index,1)=pxc(num)
740            l(index,2)=pyc(num)
            jmark=jmark+n46*npic
               do 750 j=1,n79
               index=index+1
               par(index)=jmark+j
               num=num3(2,j)
               l(index,1)=pxc(num)
750            l(index,2)=pyc(num)
c
c load the normal matrix and the right-hand-side vector
c
```

```
                do 770 j=1,index
                jj=par(j)
                del(jj)=del(jj)+dx*l(j,1)+dy*l(j,2)
                   do 760 k=1,index
                   kk=par(k)
  760              ltl(jj,kk)=ltl(jj,kk)+l(j,1)*l(k,1)+l(j,2)*l(k,2)
  770           continue
             sigmax=sigmax+dx*dx
  780        sigmay=sigmay+dy*dy
          sigma1=dsqrt((sigmax+sigmay)/nmea2)
          sigma2=dsqrt((sigmax+sigmay)/(nmea2-nunk))
          sigmax=dsqrt(sigmax/nmea)
          sigmay=dsqrt(sigmay/nmea)
c
c report sigmas
c
          write(il,*)
          write(il,260)sigmax,sigmay,sigma1,sigma2
          write(il,*)
c
c weight the normal matrix
c
             do 790 j=1,nunk
  790        ltl(j,j)=ltl(j,j)+10.d0**sweight(j)
c
c solution of weighted normal system with IMSL subroutine
c
c compute upper triangular decomposition matrix
          call dgefa(ltl,ipvt,info)
             if(info.ne.0)then
             write(il,*)' SINGULAR JACOBIAN'
             stop
             endif
c back-solve to get normal equations solution
          call dgesl(ltl,ipvt,del)
c define a set of unit vectors and back-solve to get errors of parameters
c         write(il,270)
c         write(il,*)
```

```
c           do 810 j=1,nunk
c              do 800 k=1,nunk
c              unit(k,1)=0.d0
c                 if(j.eq.k)unit(k,1)=1.d0
c 800          continue
c           call dgesl(lt1,ipvt,unit)
c           errs(j)=sigma2*dsqrt(unit(j,1))
c 810       write(il,*)j,errs(j)
c        write(il,*)
c
c correct variables
c
            do 830 ii=1,n13
            num=num1(2,ii)
               do 820 j=1,npoi
                  if(isol.eq.1)then
                  k=n13*(j-1)+ii
                     if(num.eq.1)phi(j)=phi(j)+del(k)
                     if(num.eq.2)lam(j)=lam(j)+del(k)
                     if(num.eq.3)  r(j)=  r(j)+del(k)
                  else
                  k=(n13-1)*(j-1)+ii
                     if(num.eq.1)phi(j)=phi(j)+del(k)
                     if(num.eq.2)lam(j)=lam(j)+del(k)
                     if(num.eq.3)  r(j)=  r(j)+del((n13-1)*npoi+1)
                  endif
 820           continue
 830        continue
         imark=n13*npoi
            do 850 ii=1,n46
            num=num2(2,ii)
               do 840 j=1,npic
               k=imark+n46*(j-1)+ii
                  if(num.eq.4)c(j,1)=c(j,1)+del(k)
                  if(num.eq.5)c(j,2)=c(j,2)+del(k)
                  if(num.eq.6)c(j,3)=c(j,3)+del(k)
 840           continue
 850        continue
```

```
          imark=imark+n46*npic
             do 860 ii=1,n79
             num=num3(2,ii)
             k=imark+ii
                if(num3(2,ii).eq.7)alpha0=alpha0+del(k)
                if(num3(2,ii).eq.8)delta0=delta0+del(k)
                if(num3(2,ii).eq.9)wdot=wdot+del(k)
  860        continue
  870     continue
c
c recompute residuals
c
       rewind(19)
       sigmax=0.d0
       sigmay=0.d0
       write(il,210)
       write(il,*)
          do 880 i=1,nmea
          call respar(phi,lam,r,jdate,fds,s,c,point)
          sigmax=sigmax+dx*dx
  880     sigmay=sigmay+dy*dy
       write(il,*)
c
c report sigmas
c
       sigma1=dsqrt((sigmax+sigmay)/nmea2)
       sigma2=dsqrt((sigmax+sigmay)/(nmea2-nunk))
       sigmax=dsqrt(sigmax/nmea)
       sigmay=dsqrt(sigmay/nmea)
       write(il,260)sigmax,sigmay,sigma1,sigma2
       write(il,*)
          if(iout.eq.1)then
c
c write pole position and rotation rate;points;for each picture write Julian
c date,fds,spacecraft position and C matrix angles
c
          alpha0=alpha0*rad
          delta0=delta0*rad
```

```
          wdot=wdot*rad
          write(15,150)alpha0,delta0,wdot
             do 890 i=1,npoi
             dphi=phi(i)*rad
             dlam=lam(i)*rad
                if(dlam.gt.3.6d2)dlam=dmod(dlam,3.6d2)
                if(dlam.lt.0.d0)dlam=dlam+3.6d2
                if(iew.eq.0)dlam=3.6d2-dlam
  890        write(15,130)dphi,dlam,r(i),point(i)
             do 900 i=1,npic
             write(15,230)jdate(i),fds(i)
             write(15,240)s(i,1),s(i,2),s(i,3)
             dc1=c(i,1)*rad
             dc2=c(i,2)*rad
             dc3=c(i,3)*rad
  900        write(15,250)dc1,dc2,dc3
          close(15)
          endif
       end
       subroutine respar(phi,lam,r,jdate,fds,s,c,point)
       implicit double precision (a-h,o-z)
       character*5 point(mpoi),pointc
       character*10 fds(mpic),fdsc,body
       double precision phi(mpoi),lamc,lam(mpoi),jdate(mpic),r(mpoi),
      1s(mpic,3),c(mpic,3),z1(3,1),z2(3,1),jdatec,sc(3),cc(3),ptbd(3,1),
      2em(3,3),vee(3,3),cee(3,3),sn(3,1),gamma(3,1)
       integer fpointer,ppointer
       common/dims/mpoi,mpic,munk
       common/outs/npoi,npic,nmea,nunk,il
       common/extras/ptbd,em,vee,rc,jdatec,calfc,salfc,cdelc,sdelc,cphic,
      1sphic,clamc,slamc,dx,dy,fpointer,ppointer,cee,calfhc,salfhc,
      2cdelhc,sdelhc,ckapc,skapc,sn,gamma,alpha0,delta0,wdot,body
       common/const/rad,f
  100  format(' ',13x,a10,2x,a5,2f15.5)
c
c read measurement tag info
c
       read(19)ppointer,fpointer,f,xo,yo
```

```
c
c get latitude,longitude,radius and point number of point,
c julian date,fds,spacecraft position and c matrix angles
c
      phic=phi(ppointer)
      lamc=lam(ppointer)
      rc=r(ppointer)
      pointc=point(ppointer)
      jdatec=jdate(fpointer)
      fdsc=fds(fpointer)
         do 110 j=1,3
         sc(j)=s(fpointer,j)
  110    cc(j)=c(fpointer,j)
c
c compute position of point in body-fixed coordinates
c
      cphic=dcos(phic)
      sphic=dsin(phic)
      clamc=dcos(lamc)
      slamc=dsin(lamc)
      ptbd(1,1)=rc*cphic*clamc
      ptbd(2,1)=rc*cphic*slamc
      ptbd(3,1)=rc*sphic
c
c compute alpha, delta and W
c
      call iau(body,jdatec,delalf,deldel,w0,delw)
      alpha=alpha0+delalf
      delta=delta0+deldel
      w=w0+(jdatec-2451545.d0)*wdot+delw
c
c compute the M matrix
c
      calfc=dcos(alpha)
      salfc=dsin(alpha)
      cdelc=dcos(delta)
      sdelc=dsin(delta)
      em(1,1)=-salfc
```

```
      em(1,2)=-calfc*sdelc
      em(1,3)= calfc*cdelc
      em(2,1)= calfc
      em(2,2)=-salfc*sdelc
      em(2,3)= salfc*cdelc
      em(3,1)= 0.d0
      em(3,2)=       cdelc
      em(3,3)=       sdelc
c
c compute the V matrix
c
      cw=dcos(w)
      sw=dsin(w)
      vee(1,1)= cw
      vee(1,2)=-sw
      vee(1,3)= 0.d0
      vee(2,1)= sw
      vee(2,2)= cw
      vee(2,3)= 0.d0
      vee(3,1)= 0.d0
      vee(3,2)= 0.d0
      vee(3,3)= 1.d0
c
c compute position of surface point in J2000 coordinates
c
      call matmul(vee,3,3,ptbd,1,z1)
      call matmul(em,3,3,z1,1,z2)
c
c compute vector betweeen spacecraft and surface point in J2000
c
         do 120 j=1,3
  120    sn(j,1)=z2(j,1)-sc(j)
c
c compute the C matrix
c
      calfhc=dcos(cc(1))
      salfhc=dsin(cc(1))
      cdelhc=dcos(cc(2))
```

```
      sdelhc=dsin(cc(2))
      ckapc=dcos(cc(3))
      skapc=dsin(cc(3))
      cee(1,1)=-salfhc*ckapc-calfhc*sdelhc*skapc
      cee(1,2)= calfhc*ckapc-salfhc*sdelhc*skapc
      cee(1,3)=                      cdelhc*skapc
      cee(2,1)= salfhc*skapc-calfhc*sdelhc*ckapc
      cee(2,2)=-calfhc*skapc-salfhc*sdelhc*ckapc
      cee(2,3)=                      cdelhc*ckapc
      cee(3,1)= calfhc*cdelhc
      cee(3,2)= salfhc*cdelhc
      cee(3,3)=        sdelhc
c
c compute the range vector in camera coordinates
c
      call matmul(cee,3,3,sn,1,gamma)
c
c calculated coordinates
c
      xc=gamma(1,1)*f/gamma(3,1)
      yc=gamma(2,1)*f/gamma(3,1)
c
c find differences between observed & calculated coordinates
c
      dx=xo-xc
      dy=yo-yc
c
c report residuals
c
      write(il,100)fdsc,pointc,dx,dy
      end
c
c subroutine to multiply two matricies
c
      subroutine matmul(a,m,n,b,l,c)
      double precision sum,a(m,n),b(n,l),c(m,l)
         do 120 i=1,m
            do 110 k=1,l
```

```
            sum=0.d0
               do 100 j=1,n
  100          sum=sum+a(i,j)*b(j,k)
  110       c(i,k)=sum
  120    continue
      end
c
c subroutine to compute partials of xc and yc
c
      subroutine pxcyc(pgamma,gamma,pxc,pyc)
      double precision pgamma(3,1),gamma(3,1),f,pxc,pyc,rad
      common/const/rad,f
      pxc=f*(gamma(3,1)*pgamma(1,1)-gamma(1,1)*pgamma(3,1))/
     &     (gamma(3,1)*gamma(3,1))
      pyc=f*(gamma(3,1)*pgamma(2,1)-gamma(2,1)*pgamma(3,1))/
     &     (gamma(3,1)*gamma(3,1))
      end
c
c linear system solver routines
c
      subroutine dgefa(a,ipvt,info)
      double precision a(munk,munk),t
      integer ipvt(munk)
      common/dims/mpoi,mpic,munk
      common/outs/npoi,npic,nmea,nunk,il
      lda=nunk
      n=nunk
      info=0
      nm1=n-1
         if(nm1.lt.1) go to 160
         do 150 k=1,nm1
         kp1=k+1
         l=isamax(n-k+1,a(k,k))+k-1
         ipvt(k)=l
            if(a(l,k).eq.0.d0) go to 130
            if(l.eq.k) go to 100
         t=a(l,k)
         a(l,k)=a(k,k)
```

```
      a(k,k)=t
100   continue
      t=-1.d0/a(k,k)
      call dscal(n-k,t,a(k+1,k))
         do 120 j=kp1,n
         t=a(l,j)
            if(l.eq.k) go to 110
         a(l,j)=a(k,j)
         a(k,j)=t
110      continue
         call daxpy(n-k,t,a(k+1,k),a(k+1,j))
120      continue
      go to 140
130   continue
      info=k
140   continue
150   continue
160 continue
    ipvt(n)=n
       if(a(n,n).eq.0.d0) info=n
    end
    subroutine dgesl(a,ipvt,b)
    double precision a(munk,munk),b(munk),t
    integer ipvt(munk)
    common/dims/mpoi,mpic,munk
    common/outs/npoi,npic,nmea,nunk,il
    lda=nunk
    n=nunk
    nm1=n-1
       if(nm1.lt.1) go to 120
       do 110 k=1,nm1
       l=ipvt(k)
       t=b(l)
          if(l.eq.k) go to 100
       b(l)=b(k)
       b(k)=t
100    continue
       call daxpy(n-k,t,a(k+1,k),b(k+1))
```

```
110     continue
120 continue
        do 130 kb=1,n
        k=n+1-kb
        b(k)=b(k)/a(k,k)
        t=-b(k)
        call daxpy(k-1,t,a(1,k),b(1))
130     continue
    end
    subroutine dscal(n,sa,sx)
    double precision sa,sx(1)
       if(n.eq.0) return
    m=mod(n,5)
       if(m.eq.0) go to 110
       do 100 i=1,m
       sx(i)=sa*sx(i)
100    continue
       if(n.lt.5) return
110 mp1=m+1
       do 120 i=mp1,n,5
       sx(i)=sa*sx(i)
       sx(i+1)=sa*sx(i+1)
       sx(i+2)=sa*sx(i+2)
       sx(i+3)=sa*sx(i+3)
       sx(i+4)=sa*sx(i+4)
120    continue
    end
    subroutine daxpy(n,sa,sx,sy)
    double precision sx(1),sy(1),sa
       if(n.eq.0) return
       if(sa.eq.0.d0) return
    m=mod(n,4)
       if(m.eq.0) go to 110
       do 100 i=1,m
       sy(i)=sy(i)+sa*sx(i)
100    continue
       if(n.lt.4) return
110 mp1=m+1
```

```
         do 120 i=mp1,n,4
         sy(i)=sy(i)+sa*sx(i)
         sy(i+1)=sy(i+1)+sa*sx(i+1)
         sy(i+2)=sy(i+2)+sa*sx(i+2)
         sy(i+3)=sy(i+3)+sa*sx(i+3)
  120    continue
      end
      integer function isamax(n,sx)
      double precision sx(1),smax
      isamax=0
         if(n.lt.1) return
      isamax=1
         if(n.eq.1) return
      smax=abs(sx(1))
         do 100 i=2,n
            if(abs(sx(i)).le.smax) go to 100
         isamax=i
         smax=abs(sx(i))
  100    continue
      end
c
c subroutine to compute higher order IAU terms for alpha,delta and W
c
      subroutine iau(body,jdate,delalf,deldel,w0,delw)
      implicit double precision (a-h,o-z)
      character*10 body
      double precision jdate,j1,j2,j3,j4,j5,j6,j7,j8
      common/const/rad,f
      d=jdate-2451545.d0
      t=d/36525.d0
      e1=(125.045- 0.052992*d)/rad
      e2=(249.390- 0.105984*d)/rad
      e3=(196.694-13.012000*d)/rad
      e4=(176.630+13.340716*d)/rad
      e5=(358.219- 0.985600*d)/rad
      ce1=dcos(e1)
      se1=dsin(e1)
      ce2=dcos(e2)
```

```
se2=dsin(e2)
ce3=dcos(e3)
se3=dsin(e3)
ce4=dcos(e4)
se4=dsin(e4)
se5=dsin(e5)
j1=( 73.32+91472.9*t)/rad
j2=(198.54+44243.8*t)/rad
j3=(283.90+ 4850.7*t)/rad
j4=(355.80+ 1191.3*t)/rad
j5=(119.90+  262.1*t)/rad
j6=(229.80+   64.3*t)/rad
j7=(352.25+ 2382.6*t)/rad
j8=(113.35+ 6070.0*t)/rad
cj1=dcos(j1)
sj1=dsin(j1)
s2j1=dsin(2.d0*j1)
cj2=dcos(j2)
sj2=dsin(j2)
c2j2=dcos(2.d0*j2)
s2j2=dsin(2.d0*j2)
cj3=dcos(j3)
sj3=dsin(j3)
cj4=dcos(j4)
sj4=dsin(j4)
cj5=dcos(j5)
sj5=dsin(j5)
cj6=dcos(j6)
sj6=dsin(j6)
cj7=dcos(j7)
sj7=dsin(j7)
cj8=dcos(j8)
sj8=dsin(j8)
   if(body.eq.'MOON      ')then
   delalf=(-3.878*se1-0.120*se2+0.070*se3-0.017*se4)/rad
   deldel=( 1.543*ce1+0.024*ce2-0.028*ce3+0.007*ce4)/rad
   w0=38.314/rad
   delw=(3.558*se1+0.121*se2-0.064*se3+0.016*se4+0.025*se5)/rad
```

```
elseif(body.eq.'GASPRA     ')then
delalf=0.d0
deldel=0.d0
w0=0.d0
delw=0.d0
elseif(body.eq.'METIS      ')then
delalf=(-0.009*t)/rad
deldel=( 0.003*t)/rad
w0=302.24/rad
delw=0.d0
elseif(body.eq.'ADRASTEA   ')then
delalf=(-0.009*t)/rad
deldel=( 0.003*t)/rad
w0=5.75/rad
elseif(body.eq.'AMALTHEA   ')then
delalf=(-0.009*t-0.84*sj1+0.01*s2j1)/rad
deldel=( 0.003*t-0.36*cj1)/rad
w0=231.67/rad
delw=(0.76*sj1-0.01*s2j1)/rad
elseif(body.eq.'THEBE      ')then
delalf=(-0.009*t-2.12*sj2+0.04*s2j2)/rad
deldel=( 0.003*t-0.91*cj2+0.01*c2j2)/rad
w0=9.91/rad
delw=(1.91*sj2-0.04*s2j2)/rad
elseif(body.eq.'IO         ')then
delalf=(-0.009*t+0.094*sj3+0.024*sj4)/rad
deldel=( 0.003*t+0.040*cj3+0.011*cj4)/rad
w0=200.39/rad
delw=(-0.085*sj3-0.022*sj4)/rad
elseif(body.eq.'EUROPA     ')then
delalf=(-0.009*t+1.086*sj4+0.060*sj5+0.015*sj6+0.009*sj7)/rad
deldel=( 0.003*t+0.468*cj4+0.026*cj5+0.007*cj6+0.002*cj7)/rad
w0=35.72/rad
delw=(-0.980*sj4-0.054*sj5-0.014*sj6-0.008*sj7)/rad
elseif(body.eq.'GANYMEDE   ')then
delalf=(-0.009*t-0.037*sj4+0.431*sj5+0.091*sj6)/rad
deldel=( 0.003*t-0.016*cj4+0.186*cj5+0.039*cj6)/rad
w0=43.14/rad
```

```
      delw=(0.033*sj4-0.389*sj5-0.082*sj6)/rad
      elseif(body.eq.'CALLISTO  ')then
      delalf=(-0.009*t-0.068*sj5+0.590*sj6+0.010*sj8)/rad
      deldel=( 0.003*t-0.029*cj5+0.254*cj6-0.004*cj8)/rad
      w0=259.67/rad
      delw=(0.061*sj5-0.533*sj6-0.009*sj8)/rad
      endif
   end
```

## Appendix B

## FORTRAN PROGRAM GALCGRAD

```
c
c define maximum parameters for variable arrays
c
      block data
      common/dims/mpoi,mpic,munk,mzero
      data mpoi/300/,mpic/100/,munk/1203/,mzero/10000/
      end
c
c define variables
c
      parameter(mpoi=300,mpic=100,munk=1203,mzero=10000)
      implicit double precision (a-h,o-z)
      character*5 point(mpoi),pointc
      character*8 hour
      character*10 fds(mpic),fdsc,body
      character*12 dum
      double precision phi(mpoi),lam(mpoi),r(mpoi),s(mpic,3),c(mpic,3),
     1jdate(mpic),ltl(mzero),del(munk),l(9,2),pxc(9),pyc(9),jdatec,
     2ptbd(3,1),em(3,3),vee(3,3),cee(3,3),z1(3,1),z2(3,1),z3(3,3),
     3sn(3,1),gamma(3,1),rr(munk),pp(munk),xx(munk),uu(munk),
     4ltlxp(munk)
      real*16 gam,rho,eta
      integer idtot(9),gweight(9),sweight(munk),fpointer,ppointer,
     1par(9),ipvt(munk),num1(2,3),num2(2,3),num3(2,3),irow(mzero),
     2icol(mzero)
      common/outs/npoi,npic,nmea,nunk,nonzero,il
      common/extras/ptbd,em,vee,rc,jdatec,calfc,salfc,cdelc,sdelc,cphic,
     1sphic,clamc,slamc,dx,dy,fpointer,ppointer,cee,calfhc,salfhc,
     2cdelhc,sdelhc,ckapc,skapc,sn,gamma,alpha0,delta0,wdot,body
      common/const/rad,f
  100 format(a12)
  110 format(a10)
  120 format(11i5)
```

```
  130 format(3d24.16,2x,a5)
  140 format(d24.16,2x,a10)
  150 format(3d24.16)
  160 format(a10,f15.5,2x,a5,2f15.5)
  170 format(' NO SUPPORT DATA FOR PICTURE ',a10)
  180 format(' NO SUPPORT DATA FOR POINT ',a5)
  125 format(' ITERATION ',i4,' STARTED ON ',i2,'/',i2, ' AT ',a8)
  190 format('   NPIC = ',i5,2x,' NPOI = ',i5,2x,' NMEA = ',i5,2x,
     &' NUNK = ',i5)
  200 format(' ',36x,'WEIGHTS')
  210 format(' ',16x,'PICTURE',2X,'POINT',10X,'XO-XC',10X,'YO-YC')
  220 format(20i4)
  230 format(d24.16,2x,a10,28X,'JULIAN_DATE&FDS')
  240 format(3d24.16,1x,'SXSYSZ')
  250 format(3d24.16,1x,'C1C2C3')
  260 format(' SIGL = ',f12.5,1x,'SIGM = ',f12.5,1x,'SIG1 = ',
     &f12.5,1x,'SIG2 = ',f12.5)
  280 format(' SOLVING FOR CORRECTIONS TO LATITUDES OF POINTS')
  290 format(' SOLVING FOR CORRECTIONS TO LONGITUDES OF POINTS')
  300 format(' SOLVING FOR CORRECTIONS TO RADII OF POINTS')
  310 format(' SOLVING FOR CORRECTION TO RADIUS')
  320 format(' SOLVING FOR CORRECTIONS TO ALPHA C-MATRIX ANGLES')
  330 format(' SOLVING FOR CORRECTIONS TO DELTA C-MATRIX ANGLES')
  340 format(' SOLVING FOR CORRECTIONS TO KAPPA C-MATRIX ANGLES')
  350 format(' SOLVING FOR CORRECTION TO RIGHT ASCENSION OF POLE')
  360 format(' SOLVING FOR CORRECTION TO DECLINATION OF POLE')
  370 format(' SOLVING FOR CORRECTION TO ROTATION RATE OF POLE')
  380 format(' NUMBER OF POINTS MUST BE GREATER IN PROGRAM DECLARATION')
  390 format(' NUMBER OF PICTURES MUST BE GREATER IN PROGRAM DECLARATIO
     &N')
  400 format(' NUMBER OF UNKNOWNS MUST BE GREATER IN PROGRAM DECLARATION
     &')
  905 format(' NUMBER OF NON-ZERO ELEMENTS MUST BE GREATER IN PROGRAM DE
     &CLARATION')
  908 format( ' NONZERO = ',i10)
c
c define radians constant
c
```

```
      rad=1.8d2/3.1415926535897793d0
c
c read parameter file
c
      write(6,*)' parameter file ?'
      read(5,100)dum
      open(11,file=dum,status='old')
c
c read number of pictures,number of points,number of measurements,number of
c iterations,output printing parameter,number of kinds of variables to be
c solved for,number of single parameters to be weighted,solution type,longitude
c type,output listing parameter & number of conjugate gradient iterations
c
      read(11,120)npic,npoi,nmea,nit,iout,ntot,nsw,isol,iew,list,nfirst
      il=6
         if(list.eq.1)then
         il=16
         write(6,*)' output listing file?'
         read(5,100)dum
         open(16,file=dum,status='new')
         endif
         if(npic.gt.mpic)then
         write(il,390)
         stop
         elseif(npoi.gt.mpoi)then
         write(il,380)
         stop
         endif
      nmea2=2*nmea
c
c read celestial body
c
      read(11,110)body
c
c read variable types to be solved for & their group weights
c 1:latitude of point;2:longitude of point;3:radius of point;
c 4:alpha angle of C matrix;5:delta angle of C matrix;
c 6:kappa angle of C matrix;7:right ascension of pole;8:declination of pole;
```

```
c 9:rotation rate of pole
c
         do 410 i=1,ntot
  410    read(11,120)idtot(i),gweight(i)
       write(6,*)' input file?'
       read(5,100)dum
       open(12,file=dum,status='old')
c
c read alpha sub zero and delta sub zero in degrees
c and rotation rate in degrees/day
c
       read(12,150)alpha0,delta0,wdot
       alpha0=alpha0/rad
       delta0=delta0/rad
       wdot=wdot/rad
c
c read points (radius is in meters)
c iew=0 ==> east longitude   iew=1 ==> west longitude
c
         do 420 i=1,npoi
         read(12,130)phi(i),lam(i),r(i),point(i)
            if(iew.eq.0)lam(i)=3.6d2-lam(i)
            if(isol.eq.2)r(i)=r(1)
         phi(i)=phi(i)/rad
  420    lam(i)=lam(i)/rad
c
c for each picture read :
c julian date and FDS
c spacecraft position(in m) in J2000 coordinates
c c matrix angles (alpha hat,delta hat,kappa) in degrees
c
         do 430 i=1,npic
         read(12,140)jdate(i),fds(i)
         read(12,150)s(i,1),s(i,2),s(i,3)
         read(12,150)c(i,1),c(i,2),c(i,3)
         c(i,1)=c(i,1)/rad
         c(i,2)=c(i,2)/rad
  430    c(i,3)=c(i,3)/rad
```

```
      close(12)
c
c count number of unknowns, number of variables to be solved for in three
c categories (points,pictures,pole parameters) and keep track of which ones are
c in a particular category
c
      nunk=0
      n13=0
         do 450 i=1,3
            do 440 j=1,ntot
               if(i.eq.idtot(j).and.idtot(j).eq.3.and.isol.eq.2)then
               n13=n13+1
               num1(1,n13)=j
               num1(2,n13)=i
               nunk=nunk+1
               elseif(i.eq.idtot(j))then
               n13=n13+1
               num1(1,n13)=j
               num1(2,n13)=i
               nunk=nunk+npoi
               endif
  440       continue
  450    continue
      n46=0
         do 470 i=4,6
            do 460 j=1,ntot
               if(i.eq.idtot(j))then
               n46=n46+1
               num2(1,n46)=j
               num2(2,n46)=i
               nunk=nunk+npic
               endif
  460       continue
  470    continue
      n79=0
         do 490 i=7,9
            do 480 j=1,ntot
               if(i.eq.idtot(j))then
```

```
                n79=n79+1
                num3(1,n79)=j
                num3(2,n79)=i
                nunk=nunk+1
                endif
  480        continue
  490     continue
          if(nunk.gt.munk)then
          write(il,400)
          stop
          endif
c
c set weights
c
          if(isol.eq.1)then
             do 510 i=1,n13
             num=num1(1,i)
                do 500 j=i,n13*(npoi-1)+i,n13
  500           sweight(j)=gweight(num)
  510        continue
          imark=n13*npoi
          else
             if(n13.eq.1)then
             sweight(1)=gweight(1)
             imark=1
             elseif(n13.eq.2)then
                do 530 i=1,2
                num=num1(1,i)
                   if(num1(2,i).ne.3)then
                      do 520 j=1,npoi
  520                 sweight(j)=gweight(num)
                   else
                   sweight(npoi+1)=gweight(num)
                   endif
  530           continue
             imark=npoi+1
             elseif(n13.eq.3)then
                do 560 i=1,3
```

```
                num=num1(1,i)
                   if(num1(2,i).eq.1)then
                      do 540 j=1,2*npoi-1,2
  540                 sweight(j)=gweight(num)
                   elseif(num1(2,i).eq.2)then
                      do 550 j=2,2*npoi,2
  550                 sweight(j)=gweight(num)
                   else
                   sweight(2*npoi+1)=gweight(num)
                   endif
  560           continue
              imark=2*npoi+1
              endif
           endif
           do 580 i=1,n46
           num=num2(1,i)
              do 570 j=imark+i,imark+n46*(npic-1)+i,n46
  570         sweight(j)=gweight(num)
  580      continue
         imark=imark+n46*npic
           do 590 i=1,n79
  590      sweight(imark+i)=gweight(num3(1,i))
c
c read and set single parameter weights
c
           do 600 i=1,nsw
  600      read(11,120)iswt,sweight(iswt)
         close(11)
c
c for output open pole,points,julian date,fds,spacecraft position and
c C matrix angles output file
c
           if(iout.eq.1)then
           write(6,*)' output file?'
           read(5,100)dum
           open(15,file=dum,status='new')
           endif
c
```

```
c open measurements file
c
      write(6,*)' measurements file ?'
      read(5,100)dum
      open(14,file=dum,status='old')
c
c read measurements;time is in Julian days,focal length is in millimeters
c
         do 640 i=1,nmea
         read(14,160)fdsc,f,pointc,xo,yo
c
c match fds to fds tag
c
            do 610 j=1,npic
               if(fdsc.eq.fds(j))then
               fpointer=j
               goto 620
               endif
  610       continue
         write(il,170)fdsc
         stop
c
c match point to point tag
c
  620       do 630 j=1,npoi
               if(pointc.eq.point(j))then
               ppointer=j
               goto 640
               endif
  630       continue
         write(il,180)pointc
         stop
c
c write tag info,focal length and coordinates to default fortran file
c
  640    write(19)ppointer,fpointer,f,xo,yo
      close(14)
         do 870 i=1,nit
```

```
c
c if first iteration print number of points,pictures,measurements,unknowns,
c things to be solved for and weights
c
            if(i.eq.1)then
            write(il,*)
            write(il,*)' INITIAL CONFIGURATION FOR THESE DATA'
            write(il,*)
            write(il,190)npic,npoi,nmea,nunk
            write(il,*)
               do 650 j=1,ntot
                  if(idtot(j).eq.1)write(il,280)
                  if(idtot(j).eq.2)write(il,290)
                  if(idtot(j).eq.3.and.isol.eq.1)write(il,300)
                  if(idtot(j).eq.3.and.isol.eq.2)write(il,310)
                  if(idtot(j).eq.4)write(il,320)
                  if(idtot(j).eq.5)write(il,330)
                  if(idtot(j).eq.6)write(il,340)
                  if(idtot(j).eq.7)write(il,350)
                  if(idtot(j).eq.8)write(il,360)
                  if(idtot(j).eq.9)write(il,370)
  650          continue
            write(il,*)
            write(il,200)
            write(il,220)(sweight(iz),iz=1,nunk)
            write(il,*)
            endif
         rewind(19)
         sigmax=0.d0
         sigmay=0.d0
c
c initialize things for the normal system in packed form
c
            do 303 iz=1,mzero
            irow(iz)=0
  303       ltl(iz)=0.d0
            do 304 iz=1,nunk
            icol(iz)=iz
```

```
            ltl(iz)=10.d0**sweight(iz)
  304       del(iz)=0.d0
        nonzero=nunk
        write(il,210)
        write(il,*)
c
c for each measurement:
c
            do 780 iz=1,nmea
            call respar(phi,lam,r,jdate,fds,s,c,point)
c
c point variable partials
c
               do 690 j=1,n13
               num=num1(2,j)
c
c partials wrt latitude
c
                  if(num.eq.1)then
                  z1(1,1)=-rc*sphic*clamc
                  z1(2,1)=-rc*sphic*slamc
                  z1(3,1)= rc*cphic
                  call matmul(vee,3,3,z1,1,z2)
                  call matmul(em,3,3,z2,1,z1)
                  call matmul(cee,3,3,z1,1,z2)
                  call pxcyc(z2,gamma,pxc(1),pyc(1))
c
c partials wrt longitude
c
                  elseif(num.eq.2)then
                  z1(1,1)=-rc*cphic*slamc
                  z1(2,1)= rc*cphic*clamc
                  z1(3,1)= 0.
                  call matmul(vee,3,3,z1,1,z2)
                  call matmul(em,3,3,z2,1,z1)
                  call matmul(cee,3,3,z1,1,z2)
                  call pxcyc(z2,gamma,pxc(2),pyc(2))
c
```

```
c partials wrt radius
c
                  else
                     do 680 k=1,3
  680                z1(k,1)=ptbd(k,1)/rc
                  call matmul(vee,3,3,z1,1,z2)
                  call matmul(em,3,3,z2,1,z1)
                  call matmul(cee,3,3,z1,1,z2)
                  call pxcyc(z2,gamma,pxc(3),pyc(3))
                  endif
  690          continue
c
c picture variable partials
c
               do 700 j=1,n46
               num=num2(2,j)
c
c partials wrt C matrix angle alpha hat
c
                  if(num.eq.4)then
                  z3(1,1)=-calfhc*ckapc+salfhc*sdelhc*skapc
                  z3(1,2)=-salfhc*ckapc-calfhc*sdelhc*skapc
                  z3(1,3)= 0.d0
                  z3(2,1)= calfhc*skapc+salfhc*sdelhc*ckapc
                  z3(2,2)= salfhc*skapc-calfhc*sdelhc*ckapc
                  z3(2,3)= 0.d0
                  z3(3,1)=              -salfhc*cdelhc
                  z3(3,2)=               calfhc*cdelhc
                  z3(3,3)= 0.d0
                  call matmul(z3,3,3,sn,1,z1)
                  call pxcyc(z1,gamma,pxc(4),pyc(4))
c
c partials wrt C matrix angle delta hat
c
                  elseif(num.eq.5)then
                  z3(1,1)=-calfhc*cdelhc*skapc
                  z3(1,2)=-salfhc*cdelhc*skapc
                  z3(1,3)=       -sdelhc*skapc
```

```
                z3(2,1)=-calfhc*cdelhc*ckapc
                z3(2,2)=-salfhc*cdelhc*ckapc
                z3(2,3)=       -sdelhc*ckapc
                z3(3,1)=-calfhc*sdelhc
                z3(3,2)=-salfhc*sdelhc
                z3(3,3)=        cdelhc
                call matmul(z3,3,3,sn,1,z1)
                call pxcyc(z1,gamma,pxc(5),pyc(5))
c
c partials wrt C matrix angle kappa
c
                else
                z3(1,1)= salfhc*skapc-calfhc*sdelhc*ckapc
                z3(1,2)=-calfhc*skapc-salfhc*sdelhc*ckapc
                z3(1,3)=                      cdelhc*ckapc
                z3(2,1)= salfhc*ckapc+calfhc*sdelhc*skapc
                z3(2,2)=-calfhc*ckapc+salfhc*sdelhc*skapc
                z3(2,3)=                     -cdelhc*skapc
                z3(3,1)= 0.d0
                z3(3,2)= 0.d0
                z3(3,3)= 0.d0
                call matmul(z3,3,3,sn,1,z1)
                call pxcyc(z1,gamma,pxc(6),pyc(6))
                endif
  700         continue
c
c pole variable partials
c
              do 710 j=1,n79
c
c partials wrt right ascension of pole
c
                if(num3(2,j).eq.7)then
                z3(1,1)=-calfc
                z3(1,2)= salfc*sdelc
                z3(1,3)=-salfc*cdelc
                z3(2,1)=-salfc
                z3(2,2)=-calfc*sdelc
```

```
                    z3(2,3)= calfc*cdelc
                    z3(3,1)= 0.d0
                    z3(3,2)= 0.d0
                    z3(3,3)= 0.d0
                    call matmul(vee,3,3,ptbd,1,z1)
                    call matmul(z3,3,3,z1,1,z2)
                    call matmul(cee,3,3,z2,1,z1)
                    call pxcyc(z1,gamma,pxc(7),pyc(7))
c
c partials wrt declination of pole
c
                    elseif(num3(2,j).eq.8)then
                    z3(1,1)= 0.d0
                    z3(1,2)=-calfc*cdelc
                    z3(1,3)=-calfc*sdelc
                    z3(2,1)= 0.d0
                    z3(2,2)=-salfc*cdelc
                    z3(2,3)=-salfc*sdelc
                    z3(3,1)= 0.d0
                    z3(3,2)=      -sdelc
                    z3(3,3)=       cdelc
                    call matmul(vee,3,3,ptbd,1,z1)
                    call matmul(z3,3,3,z1,1,z2)
                    call matmul(cee,3,3,z2,1,z1)
                    call pxcyc(z1,gamma,pxc(8),pyc(8))
c
c partials wrt rotation rate
c
                    else
                    pwpwdot=jdatec-2451545.d0
                    z3(1,1)=-sw*pwpwdot
                    z3(1,2)=-cw*pwpwdot
                    z3(1,3)= 0.d0
                    z3(2,1)= cw*pwpwdot
                    z3(2,2)=-sw*pwpwdot
                    z3(2,3)= 0.d0
                    z3(3,1)= 0.d0
                    z3(3,2)= 0.d0
```

```
                z3(3,3)= 0.d0
                call matmul(z3,3,3,ptbd,1,z1)
                call matmul(em,3,3,z1,1,z2)
                call matmul(cee,3,3,z2,1,z1)
                call pxcyc(z1,gamma,pxc(9),pyc(9))
                endif
  710         continue
c
c compute partials of coordinates
c
            index=0
               if(isol.eq.1)then
                  do 720 j=1,n13
                  index=index+1
                  par(index)=n13*(ppointer-1)+j
                  num=num1(2,j)
                  l(index,1)=pxc(num)
  720             l(index,2)=pyc(num)
               jmark=n13*npoi
               else
                  do 730 j=1,n13
                  index=index+1
                  num=num1(2,j)
                     if(num.eq.3)then
                     par(index)=(n13-1)*npoi+1
                     else
                     par(index)=(n13-1)*(ppointer-1)+j
                     endif
                  l(index,1)=pxc(num)
  730             l(index,2)=pyc(num)
               jmark=(n13-1)*npoi+1
               endif
               do 740 j=1,n46
               index=index+1
               par(index)=jmark+n46*(fpointer-1)+j
               num=num2(2,j)
               l(index,1)=pxc(num)
  740          l(index,2)=pyc(num)
```

```
            jmark=jmark+n46*npic
               do 750 j=1,n79
               index=index+1
               par(index)=jmark+j
               num=num3(2,j)
               l(index,1)=pxc(num)
  750          l(index,2)=pyc(num)
c
c load the normal matrix and the right-hand-side vector
c
               do 338 j=1,index
               jj=par(j)
               temp1=l(j,1)
               temp2=l(j,2)
                  do 336 k=1,index
                  kk=par(k)
                     if(kk.lt.jj)goto 336
                  k1=jj
  337                if(kk.eq.icol(k1))goto 339
                  kold=k1
                  k1=irow(kold)
                     if(k1.ne.0)goto 337
                  nonzero=nonzero+1
                  k1=nonzero
                  irow(kold)=k1
                  icol(k1)=kk
  339             ltl(k1)=ltl(k1)+temp1*l(k,1)+temp2*l(k,2)
  336             continue
               del(jj)=del(jj)+temp1*dx+temp2*dy
  338          continue
            sigmax=sigmax+dx*dx
  780       sigmay=sigmay+dy*dy
            if(i.eq.1)then
            write(il,*)
            write(il,908)nonzero
            endif
            if(nonzero.gt.mzero)then
            write(il,905)
```

```
            stop
            endif
            do 341 j=1,nunk
            k=j
  343       kold=k
            k=irow(kold)
            irow(kold)=j
               if(k.ne.0)goto 343
  341       continue
         sigma1=dsqrt((sigmax+sigmay)/nmea2)
         sigma2=dsqrt((sigmax+sigmay)/(nmea2-nunk))
         sigmax=dsqrt(sigmax/nmea)
         sigmay=dsqrt(sigmay/nmea)
c
c report sigmas
c
         write(il,*)
         write(il,260)sigmax,sigmay,sigma1,sigma2
         write(il,*)
c
c scale normal matrix and right-hand side by beauty scale factor
c to keep subroutines from overflowing
c
         scale=1.d-3
            do 701 j=1,nonzero
  701       ltl(j)=scale*ltl(j)
            do 702 j=1,nunk
  702       del(j)=scale*del(j)
c
c solution of weighted normal system by conjugate gradients
c
            do 342 j=1,nunk
            rr(j)=del(j)
            pp(j)=rr(j)
  342       xx(j)=0.d0
            do 344 mfirst=1,nfirst
c spit out the time every one hundred iterations
            mmm=100*(mfirst/100)
```

```
            if(mfirst.eq.mmm.or.mfirst.eq.1)then
            call idate(imon,iday,iyr)
            call time(hour)
            write(6,125)mfirst,imon,iday,hour
            endif
          call dot2(gam,rr,rr)
          call mult2(uu,rr,ltl,irow,icol)
          call dot2(rho,rr,uu)
            if(mfirst.eq.1)goto 345
          beta=rho/rhoo
            do 346 j=1,nunk
  346       pp(j)=rr(j)+beta*pp(j)
  345     delta2=gam
          rhoo=rho
          call mult2(ltlxp,pp,ltl,irow,icol)
          call dot2(eta,ltlxp,ltlxp)
          alpha2=rho/eta
            do 348 j=1,nunk
            rr(j)=rr(j)-alpha2*ltlxp(j)
  348       xx(j)=xx(j)+alpha2*pp(j)
            if(delta2.lt.1.d-12)goto 349
  344     continue
c correct variables
c
  349     do 830 ii=1,n13
          num=num1(2,ii)
            do 820 j=1,npoi
              if(isol.eq.1)then
              k=n13*(j-1)+ii
                if(num.eq.1)phi(j)=phi(j)+xx(k)
                if(num.eq.2)lam(j)=lam(j)+xx(k)
                if(num.eq.3)  r(j)=  r(j)+xx(k)
              else
              k=(n13-1)*(j-1)+ii
                if(num.eq.1)phi(j)=phi(j)+xx(k)
                if(num.eq.2)lam(j)=lam(j)+xx(k)
                if(num.eq.3)  r(j)=  r(j)+xx((n13-1)*npoi+1)
              endif
```

```
  820          continue
  830       continue
         imark=n13*npoi
            do 850 ii=1,n46
            num=num2(2,ii)
               do 840 j=1,npic
               k=imark+n46*(j-1)+ii
                  if(num.eq.4)c(j,1)=c(j,1)+xx(k)
                  if(num.eq.5)c(j,2)=c(j,2)+xx(k)
                  if(num.eq.6)c(j,3)=c(j,3)+xx(k)
  840          continue
  850       continue
         imark=imark+n46*npic
            do 860 ii=1,n79
            num=num3(2,ii)
            k=imark+ii
               if(num3(2,ii).eq.7)alpha0=alpha0+xx(k)
               if(num3(2,ii).eq.8)delta0=delta0+xx(k)
               if(num3(2,ii).eq.9)wdot=wdot+xx(k)
  860       continue
  870    continue
c
c recompute residuals
c
      rewind(19)
      sigmax=0.d0
      sigmay=0.d0
      write(il,210)
      write(il,*)
         do 880 i=1,nmea
         call respar(phi,lam,r,jdate,fds,s,c,point)
         sigmax=sigmax+dx*dx
  880    sigmay=sigmay+dy*dy
      write(il,*)
c
c report sigmas
c
      sigma1=dsqrt((sigmax+sigmay)/nmea2)
```

```
      sigma2=dsqrt((sigmax+sigmay)/(nmea2-nunk))
      sigmax=dsqrt(sigmax/nmea)
      sigmay=dsqrt(sigmay/nmea)
      write(il,260)sigmax,sigmay,sigma1,sigma2
      write(il,*)
         if(iout.eq.1)then
c
c write pole position and rotation rate;points;for each picture write Julian
c date,fds,spacecraft position and C matrix angles
c
         alpha0=alpha0*rad
         delta0=delta0*rad
         wdot=wdot*rad
         write(15,150)alpha0,delta0,wdot
            do 890 i=1,npoi
            dphi=phi(i)*rad
            dlam=lam(i)*rad
               if(dlam.gt.3.6d2)dlam=dmod(dlam,3.6d2)
               if(dlam.lt.0.d0)dlam=dlam+3.6d2
               if(iew.eq.0)dlam=3.6d2-dlam
 890        write(15,130)dphi,dlam,r(i),point(i)
            do 900 i=1,npic
            write(15,230)jdate(i),fds(i)
            write(15,240)s(i,1),s(i,2),s(i,3)
            dc1=c(i,1)*rad
            dc2=c(i,2)*rad
            dc3=c(i,3)*rad
 900        write(15,250)dc1,dc2,dc3
         close(15)
         endif
      end
      subroutine respar(phi,lam,r,jdate,fds,s,c,point)
      implicit double precision (a-h,o-z)
      character*5 point(mpoi),pointc
      character*10 fds(mpic),fdsc,body
      double precision phi(mpoi),lamc,lam(mpoi),jdate(mpic),r(mpoi),
     1s(mpic,3),c(mpic,3),z1(3,1),z2(3,1),jdatec,sc(3),cc(3),ptbd(3,1),
     2em(3,3),vee(3,3),cee(3,3),sn(3,1),gamma(3,1)
```

```
      integer fpointer,ppointer
      common/dims/mpoi,mpic,munk,mzero
      common/outs/npoi,npic,nmea,nunk,nonzero,il
      common/extras/ptbd,em,vee,rc,jdatec,calfc,salfc,cdelc,sdelc,cphic,
     1sphic,clamc,slamc,dx,dy,fpointer,ppointer,cee,calfhc,salfhc,
     2cdelhc,sdelhc,ckapc,skapc,sn,gamma,alpha0,delta0,wdot,body
      common/const/rad,f
  100 format(' ',13x,a10,2x,a5,2f15.5)
c
c read measurement tag info
c
      read(19)ppointer,fpointer,f,xo,yo
c
c get latitude,longitude,radius and point number of point,
c julian date,fds,spacecraft position and c matrix angles
c
      phic=phi(ppointer)
      lamc=lam(ppointer)
      rc=r(ppointer)
      pointc=point(ppointer)
      jdatec=jdate(fpointer)
      fdsc=fds(fpointer)
         do 110 j=1,3
         sc(j)=s(fpointer,j)
  110    cc(j)=c(fpointer,j)
c
c compute position of point in body-fixed coordinates
c
      cphic=dcos(phic)
      sphic=dsin(phic)
      clamc=dcos(lamc)
      slamc=dsin(lamc)
      ptbd(1,1)=rc*cphic*clamc
      ptbd(2,1)=rc*cphic*slamc
      ptbd(3,1)=rc*sphic
c
c compute alpha, delta and W
c
```

```
      call iau(body,jdatec,delalf,deldel,w0,delw)
      alpha=alpha0+delalf
      delta=delta0+deldel
      w=w0+(jdatec-2451545.d0)*wdot+delw
c
c compute the M matrix
c
      calfc=dcos(alpha)
      salfc=dsin(alpha)
      cdelc=dcos(delta)
      sdelc=dsin(delta)
      em(1,1)=-salfc
      em(1,2)=-calfc*sdelc
      em(1,3)= calfc*cdelc
      em(2,1)= calfc
      em(2,2)=-salfc*sdelc
      em(2,3)= salfc*cdelc
      em(3,1)= 0.d0
      em(3,2)=       cdelc
      em(3,3)=       sdelc
c
c compute the V matrix
c
      cw=dcos(w)
      sw=dsin(w)
      vee(1,1)= cw
      vee(1,2)=-sw
      vee(1,3)= 0.d0
      vee(2,1)= sw
      vee(2,2)= cw
      vee(2,3)= 0.d0
      vee(3,1)= 0.d0
      vee(3,2)= 0.d0
      vee(3,3)= 1.d0
c
c compute position of surface point in J2000 coordinates
c
      call matmul(vee,3,3,ptbd,1,z1)
```

```
      call matmul(em,3,3,z1,1,z2)
c
c compute vector betweeen spacecraft and surface point in J2000
c
         do 120 j=1,3
  120    sn(j,1)=z2(j,1)-sc(j)
c
c compute the C matrix
c
      calfhc=dcos(cc(1))
      salfhc=dsin(cc(1))
      cdelhc=dcos(cc(2))
      sdelhc=dsin(cc(2))
      ckapc=dcos(cc(3))
      skapc=dsin(cc(3))
      cee(1,1)=-salfhc*ckapc-calfhc*sdelhc*skapc
      cee(1,2)= calfhc*ckapc-salfhc*sdelhc*skapc
      cee(1,3)=                     cdelhc*skapc
      cee(2,1)= salfhc*skapc-calfhc*sdelhc*ckapc
      cee(2,2)=-calfhc*skapc-salfhc*sdelhc*ckapc
      cee(2,3)=                     cdelhc*ckapc
      cee(3,1)= calfhc*cdelhc
      cee(3,2)= salfhc*cdelhc
      cee(3,3)=        sdelhc
c
c compute the range vector in camera coordinates
c
      call matmul(cee,3,3,sn,1,gamma)
c
c calculated coordinates
c
      xc=gamma(1,1)*f/gamma(3,1)
      yc=gamma(2,1)*f/gamma(3,1)
c
c find differences between observed & calculated coordinates
c
      dx=xo-xc
      dy=yo-yc
```

```
c
c report residuals
c
      write(il,100)fdsc,pointc,dx,dy
      end
c
c subroutine to multiply two matricies
c
      subroutine matmul(a,m,n,b,l,c)
      double precision sum,a(m,n),b(n,l),c(m,l)
         do 120 i=1,m
            do 110 k=1,l
            sum=0.d0
               do 100 j=1,n
  100          sum=sum+a(i,j)*b(j,k)
  110       c(i,k)=sum
  120    continue
      end
c
c subroutine to compute partials of xc and yc
c
      subroutine pxcyc(pgamma,gamma,pxc,pyc)
      double precision pgamma(3,1),gamma(3,1),f,pxc,pyc,rad
      common/const/rad,f
      pxc=f*(gamma(3,1)*pgamma(1,1)-gamma(1,1)*pgamma(3,1))/
     &     (gamma(3,1)*gamma(3,1))
      pyc=f*(gamma(3,1)*pgamma(2,1)-gamma(2,1)*pgamma(3,1))/
     &     (gamma(3,1)*gamma(3,1))
      end
c
c subroutine to compute higher order IAU terms for alpha,delta and W
c
      subroutine iau(body,jdate,delalf,deldel,w0,delw)
      implicit double precision (a-h,o-z)
      character*10 body
      double precision jdate,j1,j2,j3,j4,j5,j6,j7,j8
      common/const/rad,f
      d=jdate-2451545.d0
```

```
t=d/36525.d0
e1=(125.045- 0.052992*d)/rad
e2=(249.390- 0.105984*d)/rad
e3=(196.694-13.012000*d)/rad
e4=(176.630+13.340716*d)/rad
e5=(358.219- 0.985600*d)/rad
ce1=dcos(e1)
se1=dsin(e1)
ce2=dcos(e2)
se2=dsin(e2)
ce3=dcos(e3)
se3=dsin(e3)
ce4=dcos(e4)
se4=dsin(e4)
se5=dsin(e5)
j1=( 73.32+91472.9*t)/rad
j2=(198.54+44243.8*t)/rad
j3=(283.90+ 4850.7*t)/rad
j4=(355.80+ 1191.3*t)/rad
j5=(119.90+  262.1*t)/rad
j6=(229.80+   64.3*t)/rad
j7=(352.25+ 2382.6*t)/rad
j8=(113.35+ 6070.0*t)/rad
cj1=dcos(j1)
sj1=dsin(j1)
s2j1=dsin(2.d0*j1)
cj2=dcos(j2)
sj2=dsin(j2)
c2j2=dcos(2.d0*j2)
s2j2=dsin(2.d0*j2)
cj3=dcos(j3)
sj3=dsin(j3)
cj4=dcos(j4)
sj4=dsin(j4)
cj5=dcos(j5)
sj5=dsin(j5)
cj6=dcos(j6)
sj6=dsin(j6)
```

```
cj7=dcos(j7)
sj7=dsin(j7)
cj8=dcos(j8)
sj8=dsin(j8)
   if(body.eq.'MOON      ')then
   delalf=(-3.878*se1-0.120*se2+0.070*se3-0.017*se4)/rad
   deldel=( 1.543*ce1+0.024*ce2-0.028*ce3+0.007*ce4)/rad
   w0=38.314/rad
   delw=(3.558*se1+0.121*se2-0.064*se3+0.016*se4+0.025*se5)/rad
   elseif(body.eq.'GASPRA    ')then
   delalf=0.d0
   deldel=0.d0
   w0=0.d0
   delw=0.d0
   elseif(body.eq.'METIS     ')then
   delalf=(-0.009*t)/rad
   deldel=( 0.003*t)/rad
   w0=302.24/rad
   delw=0.d0
   elseif(body.eq.'ADRASTEA  ')then
   delalf=(-0.009*t)/rad
   deldel=( 0.003*t)/rad
   w0=5.75/rad
   elseif(body.eq.'AMALTHEA  ')then
   delalf=(-0.009*t-0.84*sj1+0.01*s2j1)/rad
   deldel=( 0.003*t-0.36*cj1)/rad
   w0=231.67/rad
   delw=(0.76*sj1-0.01*s2j1)/rad
   elseif(body.eq.'THEBE     ')then
   delalf=(-0.009*t-2.12*sj2+0.04*s2j2)/rad
   deldel=( 0.003*t-0.91*cj2+0.01*c2j2)/rad
   w0=9.91/rad
   delw=(1.91*sj2-0.04*s2j2)/rad
   elseif(body.eq.'IO        ')then
   delalf=(-0.009*t+0.094*sj3+0.024*sj4)/rad
   deldel=( 0.003*t+0.040*cj3+0.011*cj4)/rad
   w0=200.39/rad
   delw=(-0.085*sj3-0.022*sj4)/rad
```

```
          elseif(body.eq.'EUROPA    ')then
          delalf=(-0.009*t+1.086*sj4+0.060*sj5+0.015*sj6+0.009*sj7)/rad
          deldel=( 0.003*t+0.468*cj4+0.026*cj5+0.007*cj6+0.002*cj7)/rad
          w0=35.72/rad
          delw=(-0.980*sj4-0.054*sj5-0.014*sj6-0.008*sj7)/rad
          elseif(body.eq.'GANYMEDE  ')then
          delalf=(-0.009*t-0.037*sj4+0.431*sj5+0.091*sj6)/rad
          deldel=( 0.003*t-0.016*cj4+0.186*cj5+0.039*cj6)/rad
          w0=43.14/rad
          delw=(0.033*sj4-0.389*sj5-0.082*sj6)/rad
          elseif(body.eq.'CALLISTO  ')then
          delalf=(-0.009*t-0.068*sj5+0.590*sj6+0.010*sj8)/rad
          deldel=( 0.003*t-0.029*cj5+0.254*cj6-0.004*cj8)/rad
          w0=259.67/rad
          delw=(0.061*sj5-0.533*sj6-0.009*sj8)/rad
          endif
       end
c
c subroutine to calculate dot product of two n-dimensional vectors
c
       subroutine dot2(temp,x,y)
       real*16 temp
       double precision x(munk),y(munk)
       common/dims/mpoi,mpic,munk,mzero
       common/outs/npoi,npic,nmea,nunk,nonzero,il
       temp=0.q0
          do 100 i=1,nunk
  100     temp=temp+x(i)*y(i)
       end
c
c subroutine to calculate product of packed-matrix and vector
c
       subroutine mult2(y,x,ltl,irow,icol)
       double precision ltl(mzero),x(munk),y(munk)
       integer irow(mzero),icol(mzero)
       common/dims/mpoi,mpic,munk,mzero
       common/outs/npoi,npic,nmea,nunk,nonzero,il
          do 100 i=1,nunk
```

```
100     y(i)=ltl(i)*x(i)
        do 110 i=nunk+1,nonzero
        i1=irow(i)
        i2=icol(i)
        y(i2)=y(i2)+ltl(i)*x(i1)
110     y(i1)=y(i1)+ltl(i)*x(i2)
    end
```

## REFERENCES

Davies, M. E., V. K. Abalakin, M. Bursa, G. E. Hunt, J. H. Lieske, B. Morando, R. H. Rapp, P. K. Seidelmann, A. T. Sinclair, and Y. S. Tjuflin, "Report of the IAU/IAG/COSPAR Working Group on Cartographic Coordinates and Rotational Elements of the Planets and Satellites: 1988," *Celestial Mechanics and Dynamical Astronomy,* 46: 187-204, 1989.

Dongarra, J. J., C. B. Moler, J. R. Bunch, and G. W. Stewart, *LINPACK Users' Guide*, Society for Industrial and Applied Mathematics, 1979.