PHOTOGRAMMETRIC EVALUATION OF

SPACE LINEAR ARRAY IMAGERY

FOR

MEDIUM SCALE TOPOGRAPHIC MAPPING

BY

MOHAMMAD JAVAD VALADAN ZOEJ

VOLUME II

A Thesis Submitted for the Degree of Doctor of Philosophy (Ph.D.)
in Photogrammetry and Remote Sensing
in the Faculty of Science at the University of Glasgow

Topographic Science, January 1997

PAGE
NUMBERS
CUT OFF
IN THE
ORIGINAL

VOLUME II

APPENDIX A : TRANSFORMATION BETWEEN UTM AND ELLIPSOIDAL GEODETIC (EG) COORDINATE SYSTEMS

(i) Introduction - Relationship Between UTM and TM Systems

Before undertaking the transformation between the latitude (ϕ) and longitude (λ) values of the EG system and the easting (X_{UTM}) and northing (Y_{UTM}) values of the UTM system, it is necessary to consider first the relationship between the TM and UTM projections. This is essential since the transformations and reverse transformations are all carried out in a two stage procedure via the intermediate stage of the general or basic TM system. Essentially the computation in the UTM system is done using the established equations for the Transverse Mercator projection (TM), while taking into account the special characteristics of the UTM system, whereby:

$$X_{UTM} = 500{,}000 + k_0\, x_{TM}$$

$$Y_{UTM} = k_0\, y_{TM}$$

or

$$x_{TM} = (1 / k_0)\,(X_{UTM} - 500{,}000)$$

$$y_{TM} = Y_{UTM} / k_0$$

where

x_{TM} : Easting in TM

y_{TM} : Northing in TM

k_0 : the central meridian projection scale factor in UTM

The following terms are required to be defined before computation in the UTM system can be undertaken:

ϕ' : latitude of the foot of the perpendicular from the point to the central meridian

S : true meridian distance on the ellipsoid from the Equator

a : semi-major axis of the ellipsoid

b : semi-minor axis of the ellipsoid

e : first eccentricity of the ellipsoid and is equal to $(a^2 - b^2) / a^2$

$\acute{e}$: second eccentricity of the ellipsoid and is equal to $(a^2 - b^2) / b^2$

N : radius of curvature in the prime vertical and is equal to: $a / (1 - e^2 \sin^2 \phi)^{-1/2}$

$\acute{N}$: is the same as N related to ϕ'

and from there:

$$(I) = S$$

$$(II) = \frac{1}{2} N \sin\phi \sin 1'' k_0$$

$$(III) = \frac{1}{24} N \sin 1'' \sin\phi \cos^3\phi (5 - \tan^2\phi + 9e'^2 \cos^2\phi + 4e'^4 \cos^4\phi)$$

$$(IV) = N \cos\phi \sin 1''$$

$$(V) = \frac{1}{6} \sin^3 1'' N \cos^3\phi (1 - \tan^2\phi + e'^2 \cos^2\phi)$$

$$(VI) = \frac{\tan\phi'}{2 N'^2 \sin 1''} (1 + e'^2 \cos^2\phi')$$

$$(VII) = \frac{\tan\phi'}{24 N'^4 \sin 1''} (5 + 3\tan^2\phi' + 6e'^2 \cos^2\phi' - 6e'^2 \sin^2\phi' - 3e'^4 \cos^4\phi' - 9e'^4 \cos_2\phi' \sin^2\phi')$$

$$(VIII) = \frac{\sec\phi'}{N' \sin 1''}$$

$$(IX) = \frac{\sec\phi'}{6 N'^3 \sin 1''} (1 + 2\tan^2\phi' + e'^2 \cos^2\phi')$$

$$\bar{A} = \frac{1}{720} \lambda^6 \sin^6 1'' N \sin\phi \cos^5\phi (61 - 58\tan^2\phi + \tan^4\phi + 270 e'^2 \cos^2\phi - 330 e'^2 \sin^2\phi)$$

$$\bar{B} = \frac{1}{120} \lambda^5 \sin^5 1'' N \cos^5\phi (5 - 18\tan^2\phi + \tan^4\phi + 14 e'^2 \cos^2\phi - 58 e'^2 \sin^2\phi)$$

$$\bar{C} = x^6 \frac{\tan\phi'}{720 N'^6 \sin 1''} (61 + 90\tan^2\phi' + 45\tan^4\phi' + 107 e'^2 \cos^2\phi' - 162 e'^2 \sin^2\phi' - 45 e'^2 \tan^2\phi' \sin^2\phi')$$

$$\bar{D} = x^5 \frac{\sec\phi'}{120 N'^5 \sin 1''} (5 + 28\tan^2\phi' + 24\tan^4\phi' + 6 e'^2 \cos^2\phi' + 8 e'^2 \sin^2\phi')$$

(ii) UTM to Ellipsoidal Geodetic (EG) System

Then the computation of the EG coordinates from the UTM values can be summarised as follows:

(a) UTM to TM

$$x_{TM} = (\frac{1}{k_0})\,(X_{UTM} - 500{,}000)$$

$$y_{TM} = \frac{Y_{UTM}}{k_0}$$

(b) TM to EG

$$\phi = \phi' - (VI)\,x_{TM}^{2} + (VII)\,x_{TM}^{4} - \bar{C}$$

$$\lambda = (VIII)\,x_{TM} - (IX)\,x_{TM}^{3} + \bar{D}$$

(iii) Ellipsoidal Geodetic (EG) to UTM system:

According to the definitions described in the direct transformation from the UTM to the EG coordinate system, the following formulas can be derived:

(a) EG to TM

$$x_{TM} = (IV)\,\lambda + (V)\,\lambda^{3} + \bar{B}$$

$$y_{TM} = (I) + (II)\,\lambda^{2} + (III)\,\lambda^{4} + \bar{A}$$

(b) TM to UTM

$$X_{UTM} = 500{,}000 + k_0\,x_{TM}$$

$$Y_{UTM} = k_0\,y_{TM}$$

APPENDIX B: COMPUTATION OF KEPLERIAN ELEMENTS USING THE EPHEMERIS DATA (POSITION AND VELOCITY VALUES) OF THE SPACECRAFT

This Appendix deals with the determination of approxime values of the Keplerian elements using the position and velocity vector of the spacecraft given as ephemeris data in the header data of the satellite image. The position and velocity of the satellite are given normally in the WGS 1984 coordinate system and for a certain time period of the satellite's movements in space. For example, in the case of SPOT, these values are given for every minute. The time of imaging the centre of the scene is also known approximately and is given in the header data. Then, in order to compute these values with respect to the centre of the scene, the Lagrange polynomial given in Chapter 6, i.e. equation (6.4) can be used. Usually the approximate values of some Keplerian elements are already known. These include a (semi-major axis of the orbit), i (orbital inclination) and e (orbital eccentricity). Using the position vector of the satellite (X_s, Y_s, Z_s) at the time of imaging the centre of the scene, the geocentric distance to the satellite (r) can be computed as follows:

$$r = \sqrt{X_s^2 + Y_s^2 + Z_s^2}$$

Now using the computed parameter r and the known parameters a and e, the eccentric anomaly E can be computed from the following formula:

$$r = a\,(1 - e\cos E) \quad then \quad E = \arccos(\frac{a - r}{a\,e})$$

Now the true anomaly (f) can be computed as follows:

$$f = \arccos(\frac{\cos E - e}{1 - e\cos E})$$

The position of the satellite in the orbital plane is given by the following formula:

$$x = r\cos(f + \omega_P)$$
$$y = r\sin(f + \omega_P)$$

where ω_p is the argument of perigee and is unknown. This element can be computed from the following formula:

$$Z_s = r\sin(f + \omega_P)\sin i \quad \textit{then} \quad \omega_P = \arcsin\left(\frac{Z_s}{r\sin i}\right) - f$$

Now x and y defined above can be computed. Using these values and the following equations, the right ascension of the ascending node (Ω) can be computed:

$$X_s = x\cos\Omega - y\sin\Omega\cos i$$
$$Y_s = x\sin\Omega + y\cos\Omega\cos i$$

or

$$\Omega = \arccos\left(\frac{X_s x + Y_s y\cos i}{x^2 + y^2\cos^2 i}\right)$$

In the case if a, e, and i are unknown, these values can be also computed through the following formulae. First the norm of the velocity vector of the satellite can be computed as follows:

$$v = \sqrt{v_x^2 + v_y^2 + v_z^2}$$

Now the semi-major axis of the orbit (a) can be computed as follows:

$$v = \sqrt{u\left(\frac{2}{r} - \frac{1}{a}\right)}$$

where u is equal to 3986005×10^8 m^3 s^{-2}.

The eccentricity e can be computed using the following two equations:

$$r \,.\, v = \sqrt{u\,a}\; e\sin E$$
$$r = a(1 - e\cos E)$$

as below:

$$e = \sqrt{\frac{r^2 v^2}{u a} + \frac{(a - r)^2}{a^2}}$$

To compute the orbital inclination *i*, the following equations have to be computed:

$$H_1 = Y_s V_z - Z_s V_y$$
$$H_2 = Z_s V_x - X_s V_z$$
$$H_3 = X_s V_y - Y_s V_x$$

Then *i* (orbital inclination) can be determined using the following equation:

$$i = \arctan\left(\frac{\sqrt{H_1^2 + H_2^2}}{H_3}\right)$$

APPENDIX C: DERIVATION OF COLLINEARITY EQUATIONS WITH RESPECT TO THE EXTERIOR ORIENTATION PARAMETERS AND GROUND COORDINATES OF THE CONTROL POINTS (GCPS)

The elements of matrix B_e which has been defined in Chapter 7, Section 7.3.3, equation 7.19, are given in this Appendix. The following assumptions have been made:

Be	= B_e;
RS	= R_s (defined in Chapter 5, equation 5.40);
RA	= R_A (defined in Chapter 5, equation 5.41);
RAS	= R_{AS} (defined in Chapter 5, equation 5.42);
R	: defined in Chapter 5, equation 5.45;
F	: is the true anomaly;
a_omega	: is the argument of perigee;
inclin	: is orbital inclination;
phi	= φ;
omega	= ω;
kapa	= κ;
pixelc	: is the pixel located in the centre of the image line;
pixel_xy	: is the x (row) coordinate of each image point;
pixel_y	: is the y (column) coordinate of each image point;
C_OMEGA	: is the right ascension of the ascending node;
r	: is the geocentric distance to the satellite;
view_angle	: is the mirror angle in the case of cross-track linear array stereo images;
f	: is focal length;
DX	= D_1 (defined in Chapter 5, equation 5.47);
DY	= D_2 (defined in Chapter 5, equation 5.47);
DZ	= D_3 (defined in Chapter 5, equation 5.47);
m_param	= m (defined in Chapter 7, equation 7.7);
n_param	= n (defined in Chapter 7, equation 7.7);
q_param	= q (defined in Chapter 7, equation 7.7);
CONST	= focal/pow(q_param,2), where pow(x,i) means x^i;

The elements of matrix B_e are as follows:

```
Be[0][0]=CONST*
    (q_param*((-RA[0][0]*RS[2][0]+RA[0][2]*RS[0][0])*DX+
            (-RA[0][0]*RS[2][1]+RA[0][2]*RS[0][1])*DY+
            (-RA[0][0]*RS[2][2]+RA[0][2]*RS[0][2])*DZ-
            r*e*sin(F)/(1+e*cos(F))*(R[0][0]*RS[2][0]+
              R[0][1]*RS[2][1]+R[0][2]*RS[2][2])-
              r*(R[0][0]*RS[0][0]+R[0][1]*RS[0][1]+R[0][2]*RS[0][2]))-
      m_param*(((RA[1][0]*RS[2][0]-RA[1][2]*RS[0][0])*sin(view_angle)+
              (-RA[2][0]*RS[2][0]+RA[2][2]*RS[0][0])*cos(view_angle))*DX+
              ((RA[1][0]*RS[2][1]-RA[1][2]*RS[0][1])*sin(view_angle)+
              (-RA[2][0]*RS[2][1]+RA[2][2]*RS[0][1])*cos(view_angle))*DY+
              ((RA[1][0]*RS[2][2]-RA[1][2]*RS[0][2])*sin(view_angle)+
              (-RA[2][0]*RS[2][2]+RA[2][2]*RS[0][2])*cos(view_angle))*DZ-
```

```
r*e*sin(F)/(1+e*cos(F))*(R[2][0]*RS[2][0]+R[2][1]*RS[2][1]+R[2][2]*
RS[2][2])-r*(R[2][0]*RS[0][0]+R[2][1]*RS[0][1]+R[2][2]*RS[0][2])));


Be[0][1]=CONST*(q_param*(-R[0][1]*DX+R[0][0]*DY+r*(RS[2][1]*R[0][0]-RS[2][
0]*R[0][1]))-m_param*(-R[2][1]*DX+R[2][0]*DY+r*(RS[2][1]*R[2][0]-RS[2][0]*R[
2][1])));

double pqr=q_param*(
    (cos(F+a_omega)*sin(inclin)*sin(C_OMEGA)*RA[0][0]+
    cos(inclin)*sin(C_OMEGA)*RA[0][1]+
    sin(inclin)*sin(C_OMEGA)*sin(F+a_omega)*RA[0][2])*DX+
    (-cos(F+a_omega)*sin(inclin)*cos(C_OMEGA)*RA[0][0]+
    (-cos(inclin)*cos(C_OMEGA))*RA[0][1]+
    (-sin(inclin)*cos(C_OMEGA)*sin(F+a_omega))*RA[0][2])*DY+
    (cos(F+a_omega)*cos(inclin)*RA[0][0]+
    (-sin(inclin))*RA[0][1]+cos(inclin)*sin(F+a_omega)*RA[0][2])*DZ+
r*sin(F+a_omega)*(-sin(C_OMEGA)*sin(inclin)+cos(C_OMEGA)*sin(inclin)-cos(inclin)));

double mno=-m_param*(
    ((-cos(F+a_omega)*sin(inclin)*sin(C_OMEGA)*RA[1][0]-
    cos(inclin)*sin(C_OMEGA)*RA[1][1]-
    sin(inclin)*sin(C_OMEGA)*sin(F+a_omega)*RA[1][2])*sin(view_angle)+
    (cos(F+a_omega)*sin(inclin)*sin(C_OMEGA)*RA[2][0]+
    cos(inclin)*sin(C_OMEGA)*RA[2][1]+
    sin(inclin)*sin(C_OMEGA)*sin(F+a_omega)*RA[2][2])*cos(view_angle))*DX+
    ((cos(F+a_omega)*sin(inclin)*cos(C_OMEGA)*RA[1][0]+
    cos(inclin)*cos(C_OMEGA)*RA[1][1]+
    sin(inclin)*cos(C_OMEGA)*sin(F+a_omega)*RA[1][2])*sin(view_angle)+
    (-cos(F+a_omega)*sin(inclin)*cos(C_OMEGA)*RA[2][0]-
    cos(inclin)*cos(C_OMEGA)*RA[2][1]-
    sin(inclin)*cos(C_OMEGA)*sin(F+a_omega)*RA[2][2])*cos(view_angle))*DY+
    ((-cos(F+a_omega)*cos(inclin)*RA[1][0]+
    sin(inclin)*RA[1][1]-
    cos(inclin)*sin(F+a_omega)*RA[1][2])*sin(view_angle)+
    (cos(F+a_omega)*cos(inclin)*RA[2][0]-
    sin(inclin)*RA[2][1]+
    cos(inclin)*sin(F+a_omega)*RA[2][2])*cos(view_angle))*DZ-
    r*sin(F+a_omega)*(sin(C_OMEGA)*sin(inclin)*R[2][0]-
    cos(C_OMEGA)*sin(inclin)*R[2][1]+cos(inclin)*R[2][2]));
Be[0][2]=CONST*(pqr+mno);

Be[0][3]=CONST*(pow(e,2)-1)/(1+e*cos(F))*
    (q_param*(RS[2][0]*R[0][0]+RS[2][1]*R[0][1]+RS[2][2]*R[0][2])-
    m_param*(RS[2][0]*R[2][0]+RS[2][1]*R[2][1]+RS[2][2]*R[2][2]));

Be[0][4]=CONST*(
    q_param*(
        (-RS[1][0]*RA[0][2]+RS[2][0]*RA[0][1])*DX+
```

```
            (-RS[1][1]*RA[0][2]+RS[2][1]*RA[0][1])*DY+
            (-RS[1][2]*RA[0][2]+RS[2][2]*RA[0][1])*DZ)-
        m_param*(
            ((RS[1][0]*RA[1][2]-RS[2][0]*RA[1][1])*sin(view_angle)+
            (-RS[1][0]*RA[2][2]+RS[2][0]*RA[2][1])*cos(view_angle))*DX+
            ((RS[1][1]*RA[1][2]-RS[2][1]*RA[1][1])*sin(view_angle)+
            (-RS[1][1]*RA[2][2]+RS[2][1]*RA[2][1])*cos(view_angle))*DY+
            ((RS[1][2]*RA[1][2]-RS[2][2]*RA[1][1])*sin(view_angle)+
            (-RS[1][2]*RA[2][2]+RS[2][2]*RA[2][1])*cos(view_angle))*DZ));

Be[0][5]=CONST*(
        q_param*(
          (RS[0][0]*(-sin(phi)*cos(kapa))+
           RS[1][0]*(sin(omega)*cos(phi)*cos(kapa))+
           RS[2][0]*(-cos(omega)*cos(phi)*cos(kapa)))*DX+
          (RS[0][1]*(-sin(phi)*cos(kapa))+
           RS[1][1]*(sin(omega)*cos(phi)*cos(kapa))+
           RS[2][1]*(-cos(omega)*cos(phi)*cos(kapa)))*DY+
          (RS[0][2]*(-sin(phi)*cos(kapa))+
           RS[1][2]*(sin(omega)*cos(phi)*cos(kapa))+
           RS[2][2]*(-cos(omega)*cos(phi)*cos(kapa)))*DZ)-
        m_param*(
         ((-RS[0][0]*(sin(phi)*sin(kapa))-
            RS[1][0]*(-sin(omega)*cos(phi)*sin(kapa))-
            RS[2][0]*(cos(omega)*cos(phi)*sin(kapa)))*sin(view_angle)+
          (RS[0][0]*(cos(phi))+
           RS[1][0]*(sin(omega)*sin(phi))+
           RS[2][0]*(-cos(omega)*sin(phi)))*cos(view_angle))*DX+
         ((-RS[0][1]*(sin(phi)*sin(kapa))-
            RS[1][1]*(-sin(omega)*cos(phi)*sin(kapa))-
            RS[2][1]*(cos(omega)*cos(phi)*sin(kapa)))*sin(view_angle)+
          (RS[0][1]*(cos(phi))+
           RS[1][1]*(sin(omega)*sin(phi))+
           RS[2][1]*(-cos(omega)*sin(phi)))*cos(view_angle))*DY+
         ((-RS[0][2]*(sin(phi)*sin(kapa))-
            RS[1][2]*(-sin(omega)*cos(phi)*sin(kapa))-
            RS[2][2]*(cos(omega)*cos(phi)*sin(kapa)))*sin(view_angle)+
          (RS[0][2]*(cos(phi))+
           RS[1][2]*(sin(omega)*sin(phi))+
           RS[2][2]*(-cos(omega)*sin(phi)))*cos(view_angle))*DZ));

Be[0][6] =CONST*(
            q_param*(RAS[1][0]*DX+RAS[1][1]*DY+RAS[1][2]*DZ)-
            m_param*(m_param*sin(view_angle)));

Be[0][7] =((pixel_xy-pixelc)*size)*Be[0][0];

Be[0][8] =((pixel_xy-pixelc)*size)*Be[0][1];
```

```
Be[0][9] =((pixel_xy-pixelc)*size)*Be[0][4];

Be[0][10]=((pixel_xy-pixelc)*size)*Be[0][5];

Be[0][11]=((pixel_xy-pixelc)*size)*Be[0][6];

if(the image is SPOT Level 1B){
     Be[0][12]=pow(((pixel_y-centre)*size),2)*Be[0][4];
     Be[0][13]=pow(((pixel_y-centre)*size),2)*Be[0][5];
     }
else{
Be[0][12]=pow(((pixel_xy-pixelc)*size),2)*Be[0][4];
Be[0][13]=pow(((pixel_xy-pixelc)*size),2)*Be[0][5];
     }
Be[0][14]=pow(((pixel_xy-pixelc)*size),2)*Be[0][6];

Be[1][0]=CONST*
     (q_param*(((-RA[1][0]*RS[2][0]+RA[1][2]*RS[0][0])*cos(view_angle)+
                (-RA[2][0]*RS[2][0]+RA[2][2]*RS[0][0])*sin(view_angle))*DX+
               ((-RA[1][0]*RS[2][1]+RA[1][2]*RS[0][1])*cos(view_angle)+
                (-RA[2][0]*RS[2][1]+RA[2][2]*RS[0][1])*sin(view_angle))*DY+
               ((-RA[1][0]*RS[2][2]+RA[1][2]*RS[0][2])*cos(view_angle)+
                (-RA[2][0]*RS[2][2]+RA[2][2]*RS[0][2])*sin(view_angle))*DZ-
              r*e*sin(F)/(1+e*cos(F))*(R[1][0]*RS[2][0]+
                R[1][1]*RS[2][1]+R[1][2]*RS[2][2])-
                r*(R[1][0]*RS[0][0]+R[1][1]*RS[0][1]+R[1][2]*RS[0][2]))-
       n_param*(((RA[1][0]*RS[2][0]-RA[1][2]*RS[0][0])*sin(view_angle)+
                (-RA[2][0]*RS[2][0]+RA[2][2]*RS[0][0])*cos(view_angle))*DX+
                ((RA[1][0]*RS[2][1]-RA[1][2]*RS[0][1])*sin(view_angle)+
                (-RA[2][0]*RS[2][1]+RA[2][2]*RS[0][1])*cos(view_angle))*DY+
                ((RA[1][0]*RS[2][2]-RA[1][2]*RS[0][2])*sin(view_angle)+
                (-RA[2][0]*RS[2][2]+RA[2][2]*RS[0][2])*cos(view_angle))*DZ-
     r*e*sin(F)/(1+e*cos(F))*(R[2][0]*RS[2][0]+R[2][1]*RS[2][1]+R[2][2]*RS[2][2])-
                r*(R[2][0]*RS[0][0]+R[2][1]*RS[0][1]+R[2][2]*RS[0][2])));

Be[1][1]=CONST*(q_param*(-R[1][1]*DX+R[1][0]*DY+r*(RS[2][1]*R[1][0]-RS[2][
     0]*R[1][1]))-
     n_param*(-R[2][1]*DX+R[2][0]*DY+r*(RS[2][1]*R[2][0]-RS[2][0]*R[2][1])));

     pqr=q_param*(
                ((cos(F+a_omega)*sin(inclin)*sin(C_OMEGA)*RA[1][0]+
                  cos(inclin)*sin(C_OMEGA)*RA[1][1]+
               sin(inclin)*sin(C_OMEGA)*sin(F+a_omega)*RA[1][2])*cos(view_angle)+
                 (cos(F+a_omega)*sin(inclin)*sin(C_OMEGA)*RA[2][0]+
                  cos(inclin)*sin(C_OMEGA)*RA[2][1]+
     sin(inclin)*sin(C_OMEGA)*sin(F+a_omega)*RA[2][2])*sin(view_angle))*DX+
                ((-cos(F+a_omega)*sin(inclin)*cos(C_OMEGA)*RA[1][0]-
                  cos(inclin)*cos(C_OMEGA)*RA[1][1]-
               sin(inclin)*cos(C_OMEGA)*sin(F+a_omega)*RA[1][2])*cos(view_angle)+
```

```
(-cos(F+a_omega)*sin(inclin)*cos(C_OMEGA)*RA[2][0]-
cos(inclin)*cos(C_OMEGA)*RA[2][1]-
sin(inclin)*cos(C_OMEGA)*sin(F+a_omega)*RA[2][2])*sin(view_angle))*DY+
((cos(F+a_omega)*cos(inclin)*RA[1][0]-
sin(inclin)*RA[1][1]+
cos(inclin)*sin(F+a_omega)*RA[1][2])*cos(view_angle)+
(cos(F+a_omega)*cos(inclin)*RA[2][0]-
sin(inclin)*RA[2][1]+
cos(inclin)*sin(F+a_omega)*RA[2][2])*sin(view_angle))*DZ-
r*sin(F+a_omega)*(sin(C_OMEGA)*sin(inclin)*R[1][0]-cos(C_OMEGA)*sin(inclin)*R
[1][1]+cos(inclin)*R[1][2]));
mno=n_param*(
((-cos(F+a_omega)*sin(inclin)*sin(C_OMEGA)*RA[1][0]-
cos(inclin)*sin(C_OMEGA)*RA[1][1]-
sin(inclin)*sin(C_OMEGA)*sin(F+a_omega)*RA[1][2])*sin(view_angle)+
(cos(F+a_omega)*sin(inclin)*sin(C_OMEGA)*RA[2][0]+
cos(inclin)*sin(C_OMEGA)*RA[2][1]+
sin(inclin)*sin(C_OMEGA)*sin(F+a_omega)*RA[2][2])*cos(view_angle))*DX+
((cos(F+a_omega)*sin(inclin)*cos(C_OMEGA)*RA[1][0]+
cos(inclin)*cos(C_OMEGA)*RA[1][1]+
sin(inclin)*cos(C_OMEGA)*sin(F+a_omega)*RA[1][2])*sin(view_angle)+
(-cos(F+a_omega)*sin(inclin)*cos(C_OMEGA)*RA[2][0]-
cos(inclin)*cos(C_OMEGA)*RA[2][1]-
sin(inclin)*cos(C_OMEGA)*sin(F+a_omega)*RA[2][2])*cos(view_angle))*DY+
((-cos(F+a_omega)*cos(inclin)*RA[1][0]+
sin(inclin)*RA[1][1]-
cos(inclin)*sin(F+a_omega)*RA[1][2])*sin(view_angle)+
(cos(F+a_omega)*cos(inclin)*RA[2][0]-
sin(inclin)*RA[2][1]+
cos(inclin)*sin(F+a_omega)*RA[2][2])*cos(view_angle))*DZ-
r*sin(F+a_omega)*(sin(C_OMEGA)*sin(inclin)*R[2][0]-
cos(C_OMEGA)*sin(inclin)*R[2][1]+cos(inclin)*R[2][2]));

Be[1][2]=CONST*(pqr-mno);

Be[1][3]=CONST*(pow(e,2)-1)/(1+e*cos(F))*
(q_param*(RS[2][0]*R[1][0]+RS[2][1]*R[1][1]+RS[2][2]*R[1][2])-
n_param*(RS[2][0]*R[2][0]+RS[2][1]*R[2][1]+RS[2][2]*R[2][2]));

Be[1][4]=CONST*(
q_param*(
((-RS[1][0]*RA[1][2]+RS[2][0]*RA[1][1])*cos(view_angle)+
(-RS[1][0]*RA[2][2]+RS[2][0]*RA[2][1])*sin(view_angle))*DX+
((-RS[1][1]*RA[1][2]+RS[2][1]*RA[1][1])*cos(view_angle)+
(-RS[1][1]*RA[2][2]+RS[2][1]*RA[2][1])*sin(view_angle))*DY+
((-RS[1][2]*RA[1][2]+RS[2][2]*RA[1][1])*cos(view_angle)+
(-RS[1][2]*RA[2][2]+RS[2][2]*RA[2][1])*sin(view_angle))*DZ)-
n_param*(
((RS[1][0]*RA[1][2]-RS[2][0]*RA[1][1])*sin(view_angle)+
```

```
                (-RS[1][0]*RA[2][2]+RS[2][0]*RA[2][1])*cos(view_angle))*DX+
                ((RS[1][1]*RA[1][2]-RS[2][1]*RA[1][1])*sin(view_angle)+
                (-RS[1][1]*RA[2][2]+RS[2][1]*RA[2][1])*cos(view_angle))*DY+
                ((RS[1][2]*RA[1][2]-RS[2][2]*RA[1][1])*sin(view_angle)+
                (-RS[1][2]*RA[2][2]+RS[2][2]*RA[2][1])*cos(view_angle))*DZ));

Be[1][5]=CONST*(
          q_param*(
           ((RS[0][0]*(sin(phi)*sin(kapa))+
             RS[1][0]*(-sin(omega)*cos(phi)*sin(kapa))+
             RS[2][0]*(cos(omega)*cos(phi)*sin(kapa)))*cos(view_angle)+
           (RS[0][0]*(cos(phi))+
            RS[1][0]*(sin(omega)*sin(phi))+
            RS[2][0]*(-cos(omega)*sin(phi)))*sin(view_angle))*DX+
           ((RS[0][1]*(sin(phi)*sin(kapa))+
             RS[1][1]*(-sin(omega)*cos(phi)*sin(kapa))+
             RS[2][1]*(cos(omega)*cos(phi)*sin(kapa)))*cos(view_angle)+
           (RS[0][1]*(cos(phi))+
            RS[1][1]*(sin(omega)*sin(phi))+
            RS[2][1]*(-cos(omega)*sin(phi)))*sin(view_angle))*DY+
           ((RS[0][2]*(sin(phi)*sin(kapa))+
             RS[1][2]*(-sin(omega)*cos(phi)*sin(kapa))+
             RS[2][2]*(cos(omega)*cos(phi)*sin(kapa)))*cos(view_angle)+
           (RS[0][2]*(cos(phi))+
            RS[1][2]*(sin(omega)*sin(phi))+
            RS[2][2]*(-cos(omega)*sin(phi)))*sin(view_angle))*DZ)-
          n_param*(
           ((-RS[0][0]*(sin(phi)*sin(kapa))-
             RS[1][0]*(-sin(omega)*cos(phi)*sin(kapa))-
             RS[2][0]*(cos(omega)*cos(phi)*sin(kapa)))*sin(view_angle)+
           (RS[0][0]*(cos(phi))+
            RS[1][0]*(sin(omega)*sin(phi))+
            RS[2][0]*(-cos(omega)*sin(phi)))*cos(view_angle))*DX+
           ((-RS[0][1]*(sin(phi)*sin(kapa))-
             RS[1][1]*(-sin(omega)*cos(phi)*sin(kapa))-
             RS[2][1]*(cos(omega)*cos(phi)*sin(kapa)))*sin(view_angle)+
           (RS[0][1]*(cos(phi))+
            RS[1][1]*(sin(omega)*sin(phi))+
            RS[2][1]*(-cos(omega)*sin(phi)))*cos(view_angle))*DY+
           ((-RS[0][2]*(sin(phi)*sin(kapa))-
             RS[1][2]*(-sin(omega)*cos(phi)*sin(kapa))-
             RS[2][2]*(cos(omega)*cos(phi)*sin(kapa)))*sin(view_angle)+
           (RS[0][2]*(cos(phi))+
            RS[1][2]*(sin(omega)*sin(phi))+
            RS[2][2]*(-cos(omega)*sin(phi)))*cos(view_angle))*DZ));

Be[1][6] =CONST*(
                q_param*(-m_param*cos(view_angle))-
                n_param*(m_param*sin(view_angle)));
```

```
Be[1][7] =((pixel_xy-pixelc)*size)*Be[1][0];
Be[1][8] =((pixel_xy-pixelc)*size)*Be[1][1];

Be[1][9] =((pixel_xy-pixelc)*size)*Be[1][4];
Be[1][10]=((pixel_xy-pixelc)*size)*Be[1][5];
Be[1][11]=((pixel_xy-pixelc)*size)*Be[1][6];

else if (the image is SPOT Level 1B) {
Be[1][12]=pow(((pixel_y-centre)*size),2)*Be[1][4];
Be[1][13]=pow(((pixel_y-centre)*size),2)*Be[1][5];
}
else {
Be[1][12]=pow(((pixel_xy-pixelc)*size),2)*Be[1][4];
Be[1][13]=pow(((pixel_xy-pixelc)*size),2)*Be[1][5];
      }

Be[1][14]=pow(((pixel_xy-pixelc)*size),2)*Be[1][6];
```

The elements of matrix B_g for the Case 3 of bundle adjustment program are as follows:

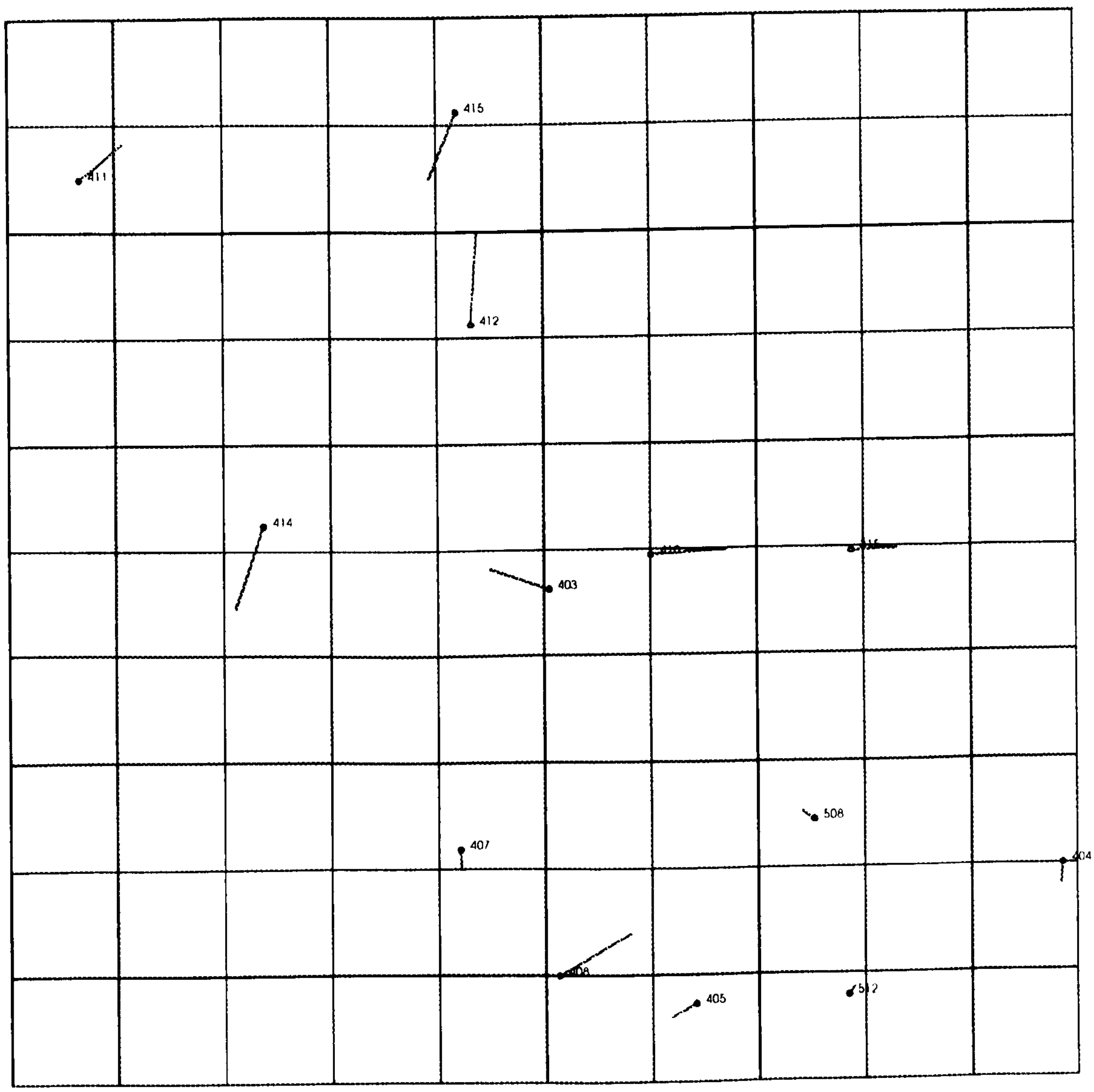
```
for(int j=0; j<3 ;++j) {

Bg[0][j]=CONST*(q_param*R[0][j]-m_param*R[2][j]);

Bg[1][j]=CONST*(q_param*R[1][j]-n_param*R[2][j]);

}
```

where:

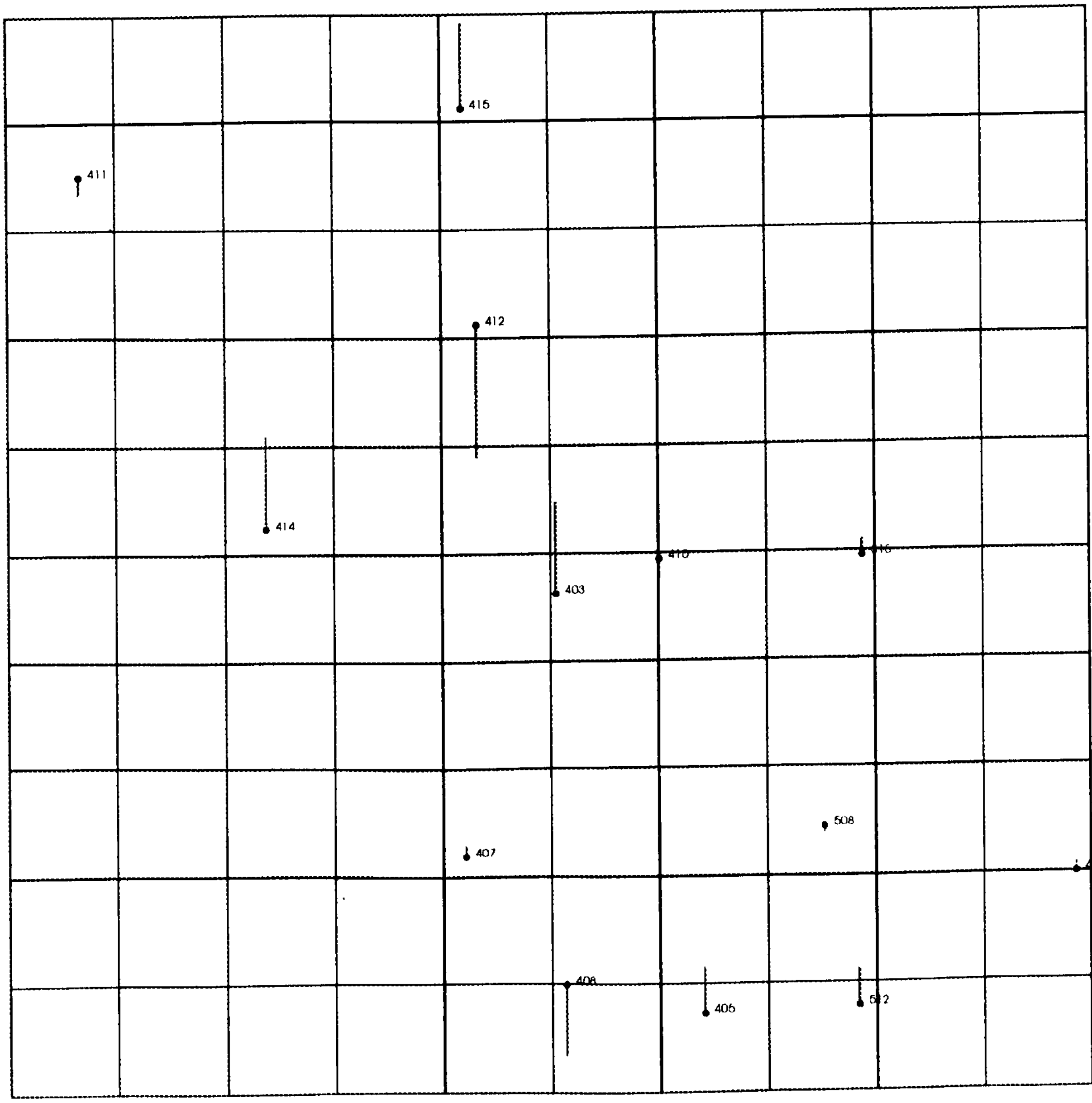
Bg = B_g.

APPENDIX D: VECTOR PLOTS OF XY AND Z ERRORS AT CONTROL POINTS FOR THE JORDANIAN TEST FIELD WITH SPOT LEVEL 1B STEREO-PAIRS

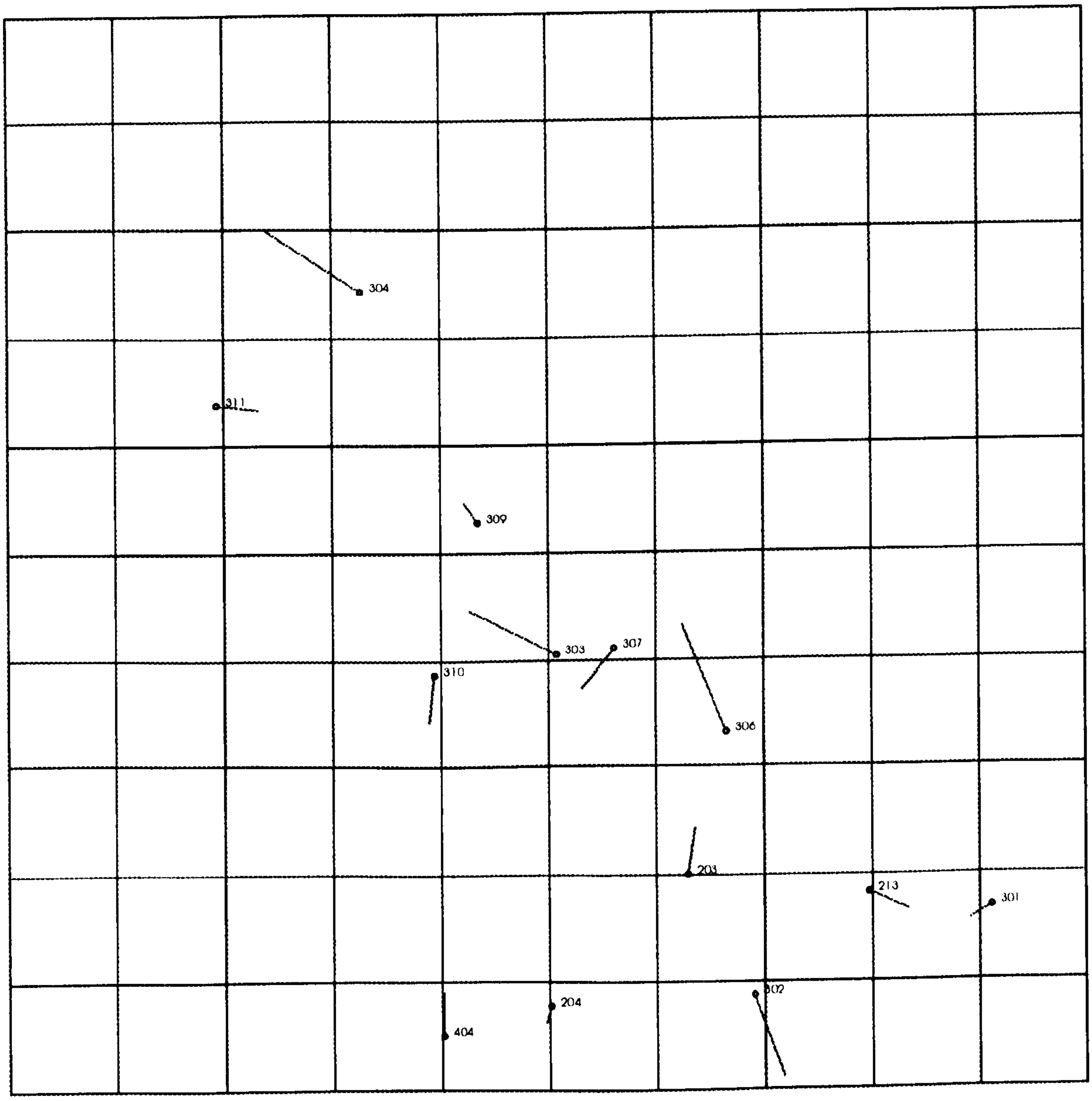
The following diagrams are the vector plots of the residual errors in the X, Y and Z directions at the control points for the Jordanian test area with four SPOT Level 1B stereo-pairs.



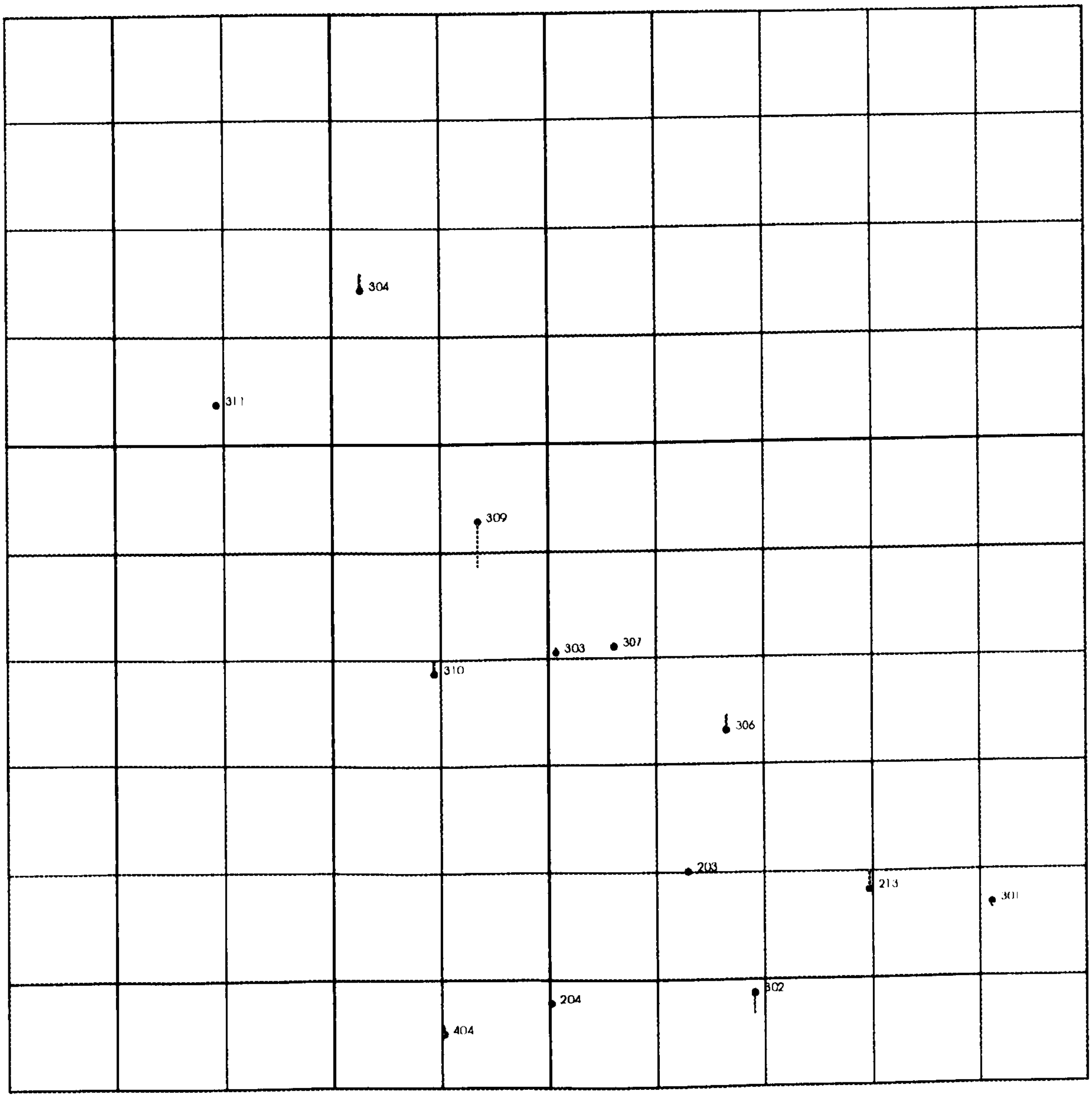
Vector Plot of XY errors at control points for the Jordanian test field with SPOT Level 1B stereo-pair (124-285)



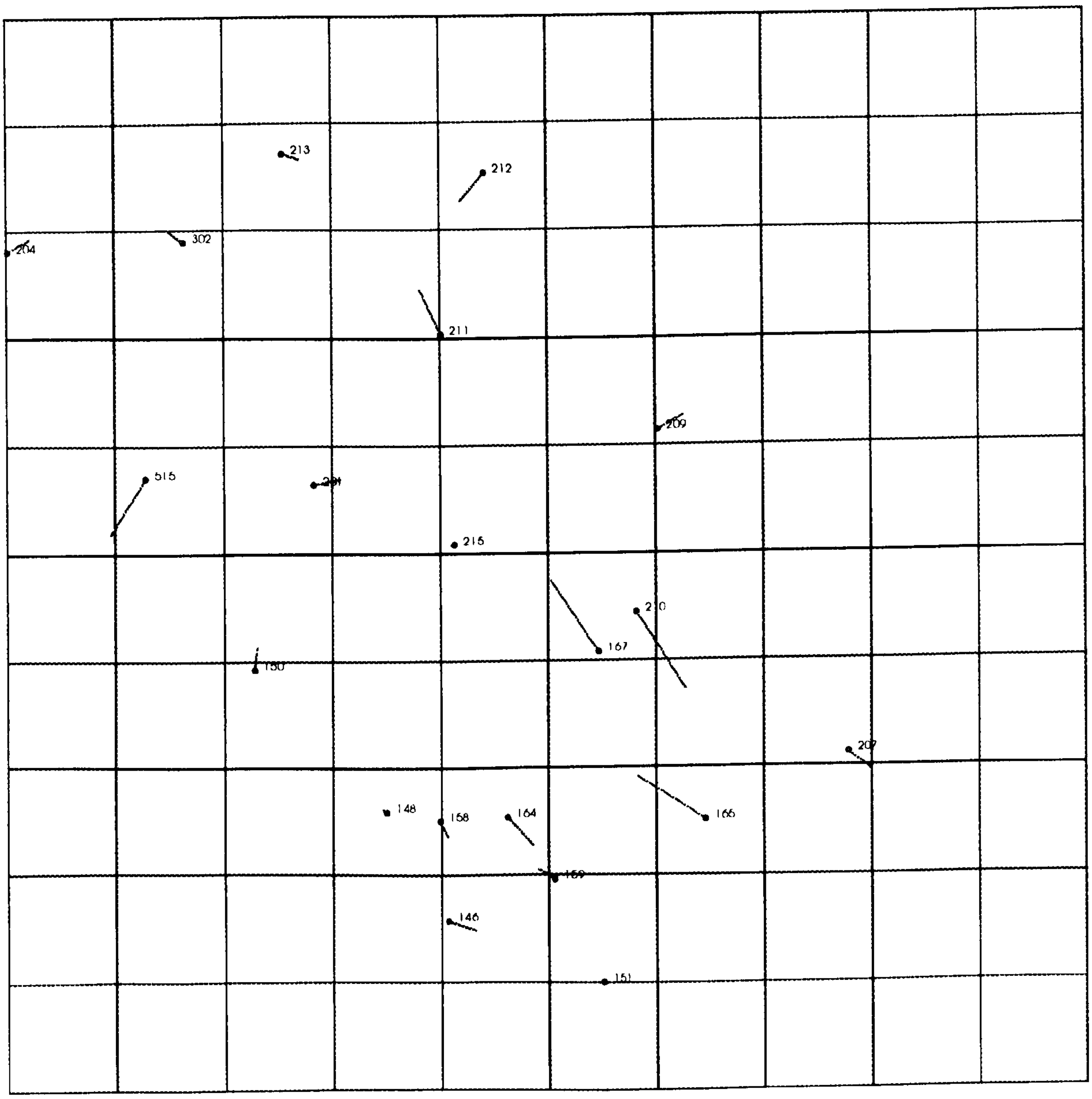
Vector Plot of Z errors at control points for the Jordanian test field with SPOT Level 1B stereo-pair (124-285)



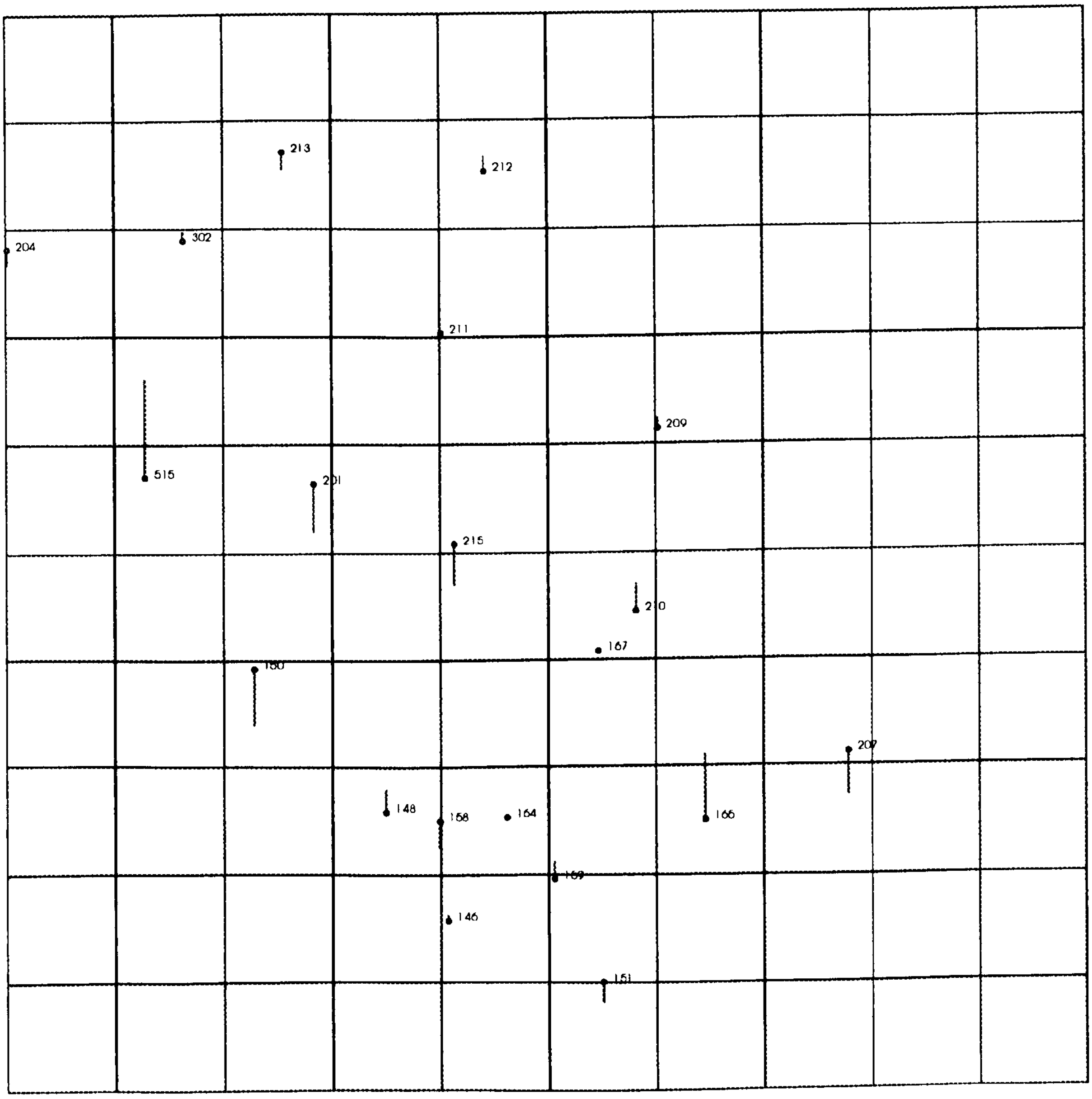
Vector Plot of XY errors at control points for the Jordanian test field with SPOT Level 1B stereo-pair (124-286)



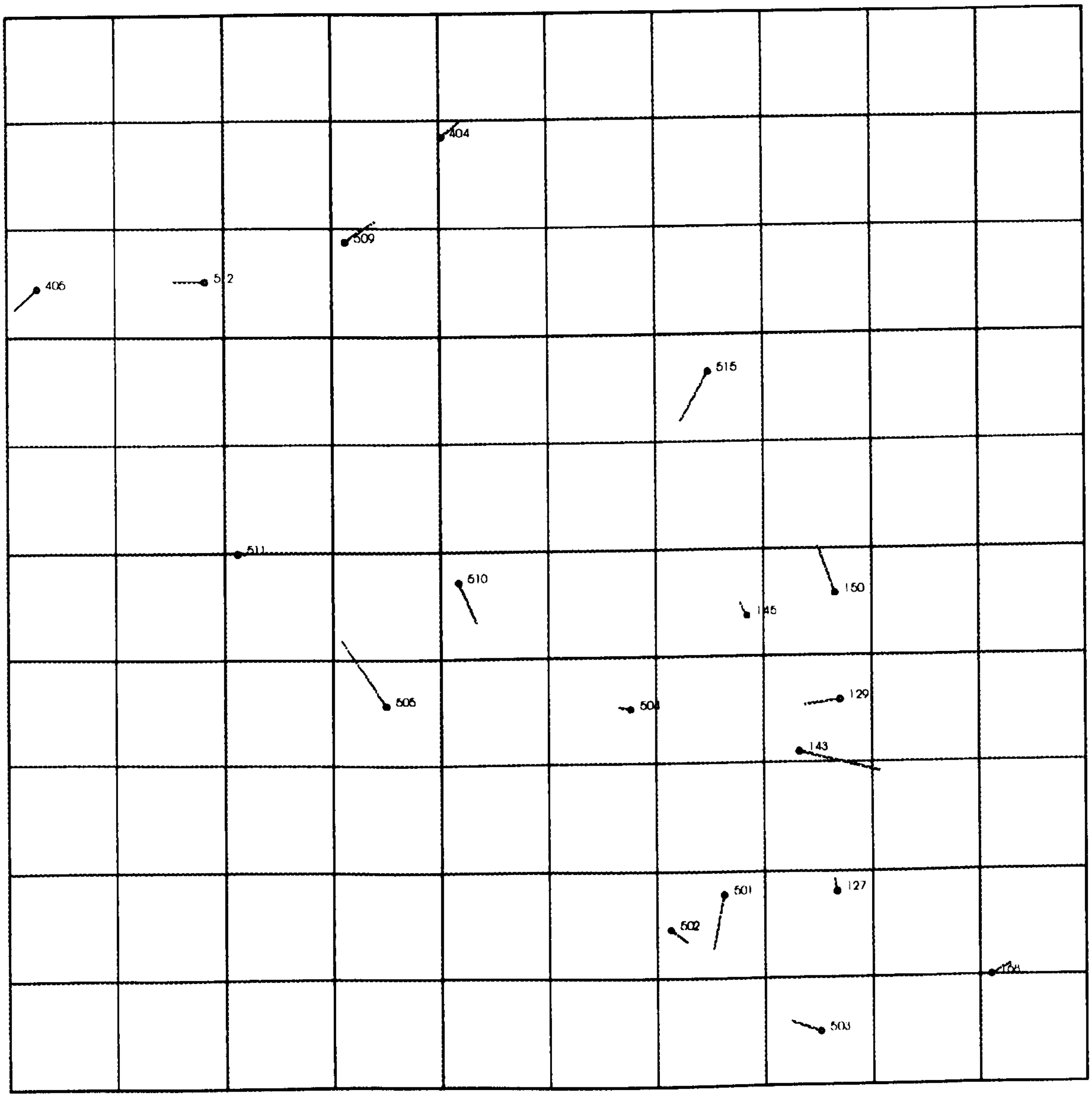
Vector Plot of Z errors at control points for the Jordanian test field with SPOT Level 1B stereo-pair (124-286)



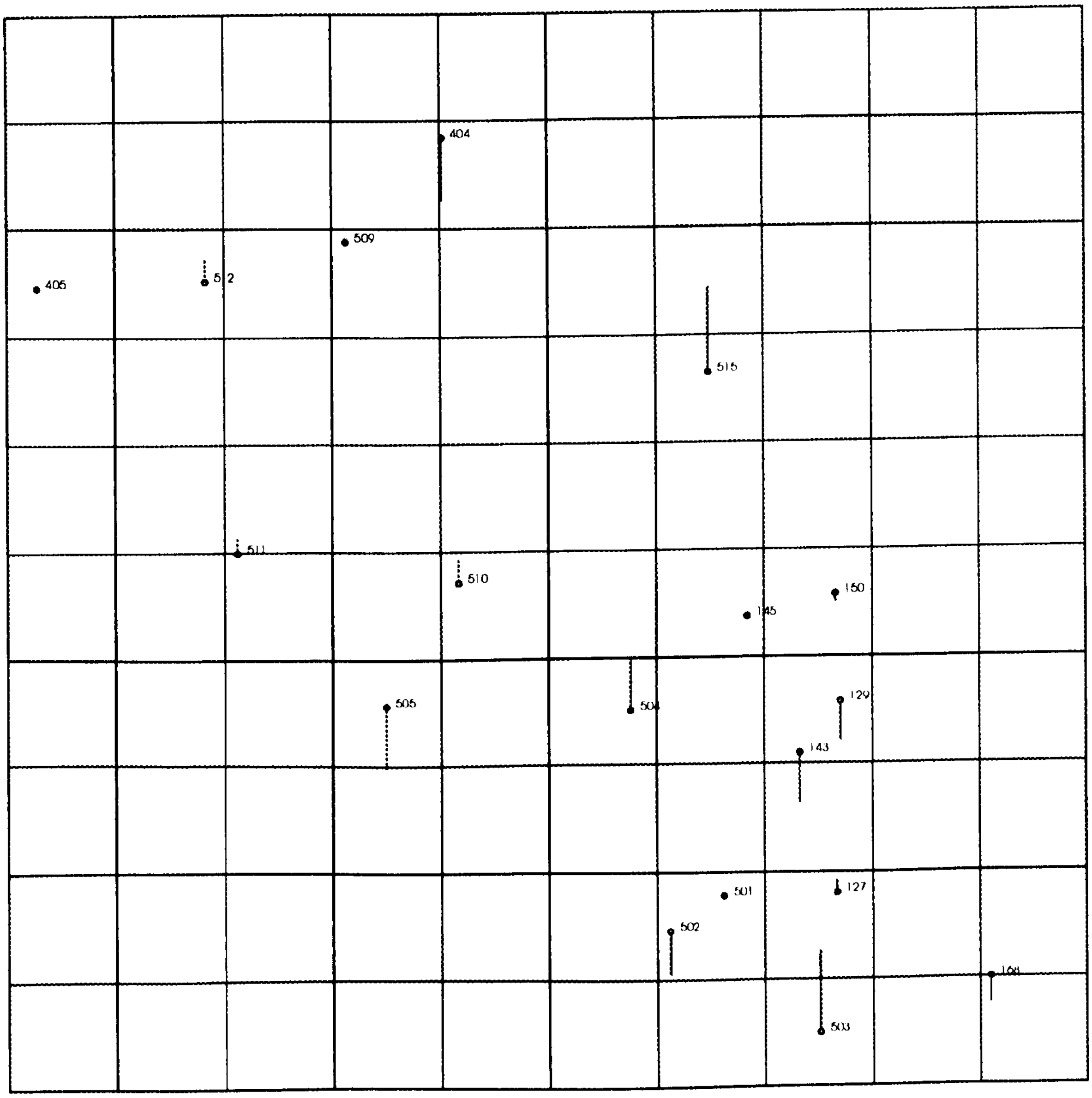
Vector Plot of XY errors at control points for the Jordanian test field with SPOT Level 1B stereo-pair (123-286)



Vector Plot of Z errors at control points for the Jordanian test field with SPOT Level 1B stereo-pair (123-286)



Vector Plot of XY errors at control points for the Jordanian test field with SPOT Level 1B stereo-pair (123-285)



Vector Plot of Z errors at control points for the Jordanian test field with SPOT Level 1B stereo-pair (123-285)

APPENDIX E: LISTING OF THE MAIN ADJUSTMENT PROGRAM AND SAMPLE INPUT/OUTPUT

```
/*---------------------------------------------------------------------------------
                              MAIN PROGRAM
         POBALAT.CPP -- A Polynomial & Bundle Adjustment Program
                        for Space Linear Array Imagery
                          M. J. Valadan Zoej, 1996
 ---------------------------------------------------------------------------------*/

#include <windows.h>
#include <commdlg.h>
#include <stdlib.h>
#include "pobalat.h"
#include "pushbrom.h"
#include "gcp.h"
#include "poly.h"

#define EDITID   1
#define UNTITLED "(untitled)"

static WORD CrossTrack;
static WORD AlongTrack;
static WORD no_of_terms;

static int _idd_ngcps;
char                    szGcpx[128];
char                    szGcpy[128];
int igcp;

long FAR PASCAL _export WndProc     (HWND, UINT, UINT, LONG) ;

BOOL CALLBACK _export AboutDlgProc (HWND, UINT, WPARAM, LPARAM) ;
BOOL _export FAR PASCAL D3DlgProc(HWND , unsigned , WORD, LONG);
BOOL _export FAR PASCAL D2DlgProc(HWND , unsigned , WORD, LONG);

void pushbroom();
void poly25();

        // Functions in POBFILE.CPP

void PopFileInitialize (HWND) ;
BOOL PopFileOpenDlg   (HWND, LPSTR, LPSTR) ;
BOOL PopFileSaveDlg   (HWND, LPSTR, LPSTR) ;
BOOL PopFileRead      (HWND, LPSTR) ;
BOOL PopFileWrite     (HWND, LPSTR) ;
```

```
// Functions in POBFIND.CPP

HWND PopFindFindDlg    (HWND) ;
HWND PopFindReplaceDlg  (HWND) ;
BOOL PopFindFindText    (HWND, int *, LPFINDREPLACE) ;
BOOL PopFindReplaceText (HWND, int *, LPFINDREPLACE) ;
BOOL PopFindNextText    (HWND, int *) ;
BOOL PopFindValidFind   (void) ;

        // Functions in POBFONT.CPP

void PopFontInitialize   (HWND) ;
BOOL PopFontChooseFont   (HWND) ;
void PopFontSetFont      (HWND) ;
void PopFontDeinitialize (void) ;

        // Functions in POBPRNT.CPP

BOOL PopPrntPrintFile (HANDLE, HWND, HWND, LPSTR) ;

        // Global variables

static char szAppName [] = "POBALAT" ;
static HWND hDlgModeless ;

int PASCAL WinMain (HANDLE hInstance, HANDLE hPrevInstance,
            LPSTR lpszCmdLine, int nCmdShow)
   {
   MSG      msg;
   HWND     hwnd ;
   HANDLE   hAccel ;
   WNDCLASS wndclass ;

   if (!hPrevInstance)
      {
      wndclass.style         = CS_HREDRAW | CS_VREDRAW ;
      wndclass.lpfnWndProc   = WndProc ;
      wndclass.cbClsExtra    = 0 ;
      wndclass.cbWndExtra    = 0 ;
      wndclass.hInstance     = hInstance ;
      wndclass.hIcon         = LoadIcon (hInstance, szAppName) ;
      wndclass.hCursor       = LoadCursor (NULL, IDC_ARROW) ;
      wndclass.hbrBackground = GetStockObject (WHITE_BRUSH) ;
      wndclass.lpszMenuName  = szAppName ;
      wndclass.lpszClassName = szAppName ;

      RegisterClass (&wndclass) ;
      }
```

```
     hwnd = CreateWindow (szAppName, NULL,
                    WS_OVERLAPPEDWINDOW,
                    CW_USEDEFAULT, CW_USEDEFAULT,
                    CW_USEDEFAULT, CW_USEDEFAULT,
                    NULL, NULL, hInstance, lpszCmdLine) ;

     ShowWindow (hwnd, nCmdShow) ;
     UpdateWindow (hwnd);

     hAccel = LoadAccelerators (hInstance, szAppName) ;

     while (GetMessage (&msg, NULL, 0, 0))
          {
          if (hDlgModeless == NULL || !IsDialogMessage (hDlgModeless, &msg))
               {
               if (!TranslateAccelerator (hwnd, hAccel, &msg))
                    {
                    TranslateMessage (&msg) ;
                    DispatchMessage (&msg) ;
                    }
               }
          }
     return msg.wParam ;
     }

void DoCaption (HWND hwnd, char *szTitleName)
     {
     char szCaption [64 + _MAX_FNAME + _MAX_EXT] ;

     wsprintf (szCaption, "%s - %s", (LPSTR) szAppName,
               (LPSTR) (szTitleName [0] ? szTitleName : UNTITLED)) ;

     SetWindowText (hwnd, szCaption) ;
     }

void OkMessage (HWND hwnd, char *szMessage, char *szTitleName)
     {
     char szBuffer [64 + _MAX_FNAME + _MAX_EXT] ;

     wsprintf (szBuffer, szMessage,
               (LPSTR) (szTitleName [0] ? szTitleName : UNTITLED)) ;

     MessageBox (hwnd, szBuffer, szAppName, MB_OK | MB_ICONEXCLAMATION) ;
     }

short AskAboutSave (HWND hwnd, char *szTitleName)
     {
     char  szBuffer [64 + _MAX_FNAME + _MAX_EXT] ;
     short nReturn ;
```

```
    wsprintf (szBuffer, "Save current changes in %s?",
          (LPSTR) (szTitleName [0] ? szTitleName : UNTITLED)) ;

    nReturn = MessageBox (hwnd, szBuffer, szAppName,
                    MB_YESNOCANCEL | MB_ICONQUESTION) ;

    if (nReturn == IDYES)
       if (!SendMessage (hwnd, WM_COMMAND, IDM_SAVE, 0L))
          nReturn = IDCANCEL ;

    return nReturn ;
    }

long FAR PASCAL _export WndProc (HWND hwnd, UINT message, UINT wParam,
                                        LONG lParam)
    {
    static BOOL    bNeedSave = FALSE ;
    static char    szFileName  [_MAX_PATH] ;
    static char    szTitleName [_MAX_FNAME + _MAX_EXT] ;

    //************************
    DLGPROC lpfnAboutDlgProc;
    DLGPROC lpfnBeginDlgProc;
    DLGPROC lpfnD3DlgProc;
    DLGPROC lpfnD2DlgProc;
    HCURSOR hCursor,hOldCursor;
    //************************
    static HANDLE  hInst ;
    static HWND    hwndEdit ;
    static int     iOffset ;
    static UINT    messageFindReplace ;
    LONG           lSelect ;
    LPFINDREPLACE  lpfr ;
    WORD           wEnable ;

    switch (message)
       {
         case WM_CREATE:

hInst=(HINSTANCE)GetWindowWord(hwnd,GWW_HINSTANCE);
                  lpfnBeginDlgProc = (DLGPROC)MakeProcInstance (
                           (FARPROC) AboutDlgProc,hInst) ;
                  DialogBox (hInst, "BeginBox", hwnd,
                              lpfnBeginDlgProc);
                  FreeProcInstance((FARPROC)lpfnBeginDlgProc);

              // Get About dialog instance address

         hInst = ((LPCREATESTRUCT) lParam)->hInstance ;
```

```
    lpfnAboutDlgProc = MakeProcInstance ((FARPROC) AboutDlgProc,
                                hInst) ;

          // Create the edit control child window

    hwndEdit = CreateWindow ("edit", NULL,
          WS_CHILD | WS_VISIBLE | WS_HSCROLL | WS_VSCROLL |
             WS_BORDER | ES_LEFT | ES_MULTILINE |
             ES_NOHIDESEL | ES_AUTOHSCROLL | ES_AUTOVSCROLL,
          0, 0, 0, 0,
          hwnd, EDITID, hInst, NULL) ;

      SendMessage (hwndEdit, EM_LIMITTEXT, 6000000, 0L) ;

          // Initialize common dialog box stuff

    PopFileInitialize (hwnd) ;
    PopFontInitialize (hwndEdit) ;

    messageFindReplace = RegisterWindowMessage (FINDMSGSTRING) ;

          // Process command line

    lstrcpy (szFileName, (LPSTR)
         (((LPCREATESTRUCT) lParam)->lpCreateParams)) ;

    if (lstrlen (szFileName) > 0)
       {
       GetFileTitle (szFileName, szTitleName,
               sizeof (szTitleName)) ;

       if (!PopFileRead (hwndEdit, szFileName))
          OkMessage (hwnd, "File %s cannot be read!",
                   szTitleName) ;
       }

    DoCaption (hwnd, szTitleName) ;
    return 0 ;

case WM_SETFOCUS:
    SetFocus (hwndEdit) ;
    return 0 ;

case WM_SIZE:
    MoveWindow (hwndEdit, 0, 0, LOWORD (lParam),
                   HIWORD (lParam), TRUE) ;
    return 0 ;

case WM_INITMENUPOPUP:
```

```
switch (lParam)
   {
   case 1:      // Edit menu

         // Enable Undo if edit control can do it

      EnableMenuItem (wParam, IDM_UNDO,
         SendMessage (hwndEdit, EM_CANUNDO, 0, 0L) ?
            MF_ENABLED : MF_GRAYED) ;

         // Enable Paste if text is in the clipboard

      EnableMenuItem (wParam, IDM_PASTE,
         IsClipboardFormatAvailable (CF_TEXT) ?
            MF_ENABLED : MF_GRAYED) ;

         // Enable Cut, Copy, and Del if text is selected

      lSelect = SendMessage (hwndEdit, EM_GETSEL, 0, 0L) ;
      wEnable = HIWORD (lSelect) != LOWORD (lSelect) ?
               MF_ENABLED : MF_GRAYED ;

      EnableMenuItem (wParam, IDM_CUT,  wEnable) ;
      EnableMenuItem (wParam, IDM_COPY, wEnable) ;
      EnableMenuItem (wParam, IDM_DEL,  wEnable) ;
      break ;

   case 2:      // Search menu

         // Enable Find, Next, and Replace if modeless
         //   dialogs are not already active

      wEnable = hDlgModeless == NULL ?
               MF_ENABLED : MF_GRAYED ;

      EnableMenuItem (wParam, IDM_FIND,    wEnable) ;
      EnableMenuItem (wParam, IDM_NEXT,    wEnable) ;
      EnableMenuItem (wParam, IDM_REPLACE, wEnable) ;
      break ;
   }
return 0 ;

case WM_COMMAND :
         // Messages from edit control

   if (LOWORD (lParam) && wParam == EDITID)
      {
      switch (HIWORD (lParam))
         {
```

```
        case EN_UPDATE:
           bNeedSave = TRUE ;
           return 0 ;

        case EN_ERRSPACE:
        case EN_MAXTEXT:
           MessageBox (hwnd, "Edit control out of space.",
                 szAppName, MB_OK | MB_ICONSTOP) ;
           return 0 ;
        }
    break ;
    }

switch (wParam)
    {
          // Messages from File menu
case IDM_ABOUT:
        hInst=(HINSTANCE)GetWindowWord(hwnd,GWW_HINSTANCE);
              lpfnAboutDlgProc = (DLGPROC)MakeProcInstance (
                             (FARPROC) AboutDlgProc,hInst) ;
        DialogBox (hInst, "AboutBox", hwnd,
                             lpfnAboutDlgProc);
              FreeProcInstance((FARPROC)lpfnAboutDlgProc);
break;

case IDM_2D:
       hInst=(HINSTANCE)GetWindowWord(hwnd,GWW_HINSTANCE);
       lpfnD2DlgProc = (DLGPROC)MakeProcInstance (
                             (FARPROC) D2DlgProc,hInst) ;

if(!(DialogBox (hInst, "D2DLG", hwnd,
                             lpfnD2DlgProc)))
return 0;
       FreeProcInstance((FARPROC)lpfnD2DlgProc);
       switch(no_of_terms){
              case IDD_T3:

                 if(IDOK==MessageBox(hwnd,
              "You are now selecting 3 terms to be run in the\n"
              "        polynomial adjustment program.\n"
              "                         Is this OK?",
                 szAppName, MB_ICONQUESTION | MB_OKCANCEL)){
                 if(_idd_ngcps < 3){
                 MessageBox(hwnd,
                 "Number of GCPs are less than the selected terms!\n"
                 "        Please increase the number of GCPs.",
                 szAppName, MB_ICONINFORMATION | MB_OK);
                 parameter=0;}
                 else
```

```
                parameter=3;}
              else if(IDCANCEL)
                    break;
break;
        case IDD_T4:

              if(IDOK==MessageBox(hwnd,
        "You are now selecting 4 terms to be run in the\n"
        "         polynomial adjustment program.\n"
        "                    Is this OK?",
                szAppName, MB_ICONQUESTION | MB_OKCANCEL)){
                if(_idd_ngcps < 4){
                MessageBox(hwnd,
                "Number of GCPs are less than the selected terms!\n"
                "         Please increase the number of GCPs\n"
                "           or decrease the number of terms.",
                szAppName, MB_ICONINFORMATION | MB_OK);
                parameter=0;}
                else
                parameter=4;}
              else if(IDCANCEL)
                    break;
break;
        case IDD_T5:

              if(IDOK==MessageBox(hwnd,
        "You are now selecting 5 terms to be run in the\n"
        "         polynomial adjustment program.\n"
        "                    Is this OK?",
                szAppName, MB_ICONQUESTION | MB_OKCANCEL)){
                if(_idd_ngcps < 5){
                MessageBox(hwnd,
                "Number of GCPs are less than the selected terms!\n"
                "         Please increase the number of GCPs\n"
                "           or decrease the number of terms.",
                szAppName, MB_ICONINFORMATION | MB_OK);
                parameter=0;}
                else
                parameter=5;}
              else if(IDCANCEL)
                    break;
        break;
        case IDD_T6:

              if(IDOK==MessageBox(hwnd,
        "You are now selecting 6 terms to be run in the\n"
        "         polynomial adjustment program.\n"
        "                    Is this OK?",
                szAppName, MB_ICONQUESTION | MB_OKCANCEL)){
```

```
        if(_idd_ngcps < 6){
        MessageBox(hwnd,
        "Number of GCPs are less than the selected terms!\n"
        "          Please increase the number of GCPs\n"
        "             or decrease the number of terms.",
        szAppName, MB_ICONINFORMATION | MB_OK);
        parameter=0;}
        else
        parameter=6;}
      else if(IDCANCEL)
              break;
   break;
   case IDD_T7:

      if(IDOK==MessageBox(hwnd,
  "You are now selecting 7 terms to be run in the\n"
  "        polynomial adjustment program.\n"
  "                     Is this OK?",
        szAppName, MB_ICONQUESTION | MB_OKCANCEL)){
        if(_idd_ngcps < 7){
        MessageBox(hwnd,
        "Number of GCPs are less than the selected terms!\n"
        "          Please increase the number of GCPs\n"
        "             or decrease the number of terms.",
        szAppName, MB_ICONINFORMATION | MB_OK);
        parameter=0;}
        else
        parameter=7;}
      else if(IDCANCEL)
              break;
   break;
   case IDD_T8:

      if(IDOK==MessageBox(hwnd,
  "You are now selecting 8 terms to be run in the\n"
  "        polynomial adjustment program.\n"
  "                     Is this OK?",
        szAppName, MB_ICONQUESTION | MB_OKCANCEL)){
        if(_idd_ngcps < 8){
        MessageBox(hwnd,
        "Number of GCPs are less than the selected terms!\n"
        "          Please increase the number of GCPs\n"
        "             or decrease the number of terms.",
        szAppName, MB_ICONINFORMATION | MB_OK);
        parameter=0;}
        else
        parameter=8;}
      else if(IDCANCEL)
              break;
```

```
 break;
 case IDD_T9:

   if(IDOK==MessageBox(hwnd,
"You are now selecting 9 terms to be run in the\n"
"        polynomial adjustment program.\n"
"                        Is this OK?",
     szAppName, MB_ICONQUESTION | MB_OKCANCEL)){
     if(_idd_ngcps < 9){
     MessageBox(hwnd,
     "Number of GCPs are less than the selected terms!\n"
     "            Please increase the number of GCPs\n"
     "               or decrease the number of terms.",
     szAppName, MB_ICONINFORMATION | MB_OK);
     parameter=0;}
     else
     parameter=9;}
   else if(IDCANCEL)
          break;
 break;
 case IDD_T10:

   if(IDOK==MessageBox(hwnd,
"You are now selecting 10 terms to be run in the\n"
"        polynomial adjustment program.\n"
"                        Is this OK?",
     szAppName, MB_ICONQUESTION | MB_OKCANCEL)){
     if(_idd_ngcps < 10){
     MessageBox(hwnd,
     "Number of GCPs are less than the selected terms!\n"
     "            Please increase the number of GCPs\n"
     "               or decrease the number of terms.",
     szAppName, MB_ICONINFORMATION | MB_OK);
     parameter=0;}
     else
     parameter=10;}
   else if(IDCANCEL)
          break;
 break;
 case IDD_T11:

   if(IDOK==MessageBox(hwnd,
"You are now selecting 11 terms to be run in the\n"
"        polynomial adjustment program.\n"
"                        Is this OK?",
     szAppName, MB_ICONQUESTION | MB_OKCANCEL)){
     if(_idd_ngcps < 11){
     MessageBox(hwnd,
     "Number of GCPs are less than the selected terms!\n"
```

```
      "           Please increase the number of GCPs\n"
      "              or decrease the number of terms.",
      szAppName, MB_ICONINFORMATION | MB_OK);
      parameter=0;}
      else
      parameter=11;}
    else if(IDCANCEL)
            break;
 break;
 case IDD_T12:

    if(IDOK==MessageBox(hwnd,
"You are now selecting 12 terms to be run in the\n"
"       polynomial adjustment program.\n"
"                     Is this OK?",
      szAppName, MB_ICONQUESTION | MB_OKCANCEL)){
      if(_idd_ngcps < 12){
      MessageBox(hwnd,
      "Number of GCPs are less than the selected terms!\n"
      "           Please increase the number of GCPs\n"
      "              or decrease the number of terms.",
      szAppName, MB_ICONINFORMATION | MB_OK);
      parameter=0;}
      else
      parameter=12;}
    else if(IDCANCEL)
            break;
 break;
 case IDD_T13:

    if(IDOK==MessageBox(hwnd,
"You are now selecting 13 terms to be run in the\n"
"       polynomial adjustment program.\n"
"                     Is this OK?",
      szAppName, MB_ICONQUESTION | MB_OKCANCEL)){
      if(_idd_ngcps < 13){
      MessageBox(hwnd,
      "Number of GCPs are less than the selected terms!\n"
      "           Please increase the number of GCPs\n"
      "              or decrease the number of terms.",
      szAppName, MB_ICONINFORMATION | MB_OK);
      parameter=0;}
      else
      parameter=13;}
    else if(IDCANCEL)
            break;
 break;
 case IDD_T14:
```

```
   if(IDOK==MessageBox(hwnd,
"You are now selecting 14 terms to be run in the\n"
"        polynomial adjustment program.\n"
"                       Is this OK?",
     szAppName, MB_ICONQUESTION | MB_OKCANCEL)){
     if(_idd_ngcps < 14){
     MessageBox(hwnd,
     "Number of GCPs are less than the selected terms!\n"
     "           Please increase the number of GCPs\n"
     "               or decrease the number of terms.",
     szAppName, MB_ICONINFORMATION | MB_OK);
     parameter=0;}
     else
     parameter=14;}
   else if(IDCANCEL)
          break;
 break;
 case IDD_T15:

   if(IDOK==MessageBox(hwnd,
"You are now selecting 15 terms to be run in the\n"
"        polynomial adjustment program.\n"
"                       Is this OK?",
     szAppName, MB_ICONQUESTION | MB_OKCANCEL)){
     if(_idd_ngcps < 15){
     MessageBox(hwnd,
     "Number of GCPs are less than the selected terms!\n"
     "           Please increase the number of GCPs\n"
     "               or decrease the number of terms.",
     szAppName, MB_ICONINFORMATION | MB_OK);
     parameter=0;}
     else
     parameter=15;}
   else if(IDCANCEL)
          break;
 break;
 case IDD_T16:

   if(IDOK==MessageBox(hwnd,
"You are now selecting 16 terms to be run in the\n"
"        polynomial adjustment program.\n"
"                       Is this OK?",
     szAppName, MB_ICONQUESTION | MB_OKCANCEL)){
     if(_idd_ngcps < 16){
     MessageBox(hwnd,
     "Number of GCPs are less than the selected terms!\n"
     "           Please increase the number of GCPs\n"
     "               or decrease the number of terms.",
     szAppName, MB_ICONINFORMATION | MB_OK);
```

```
    parameter=0;}
    else
    parameter=16;}
  else if(IDCANCEL)
        break;
 break;
 case IDD_T17:

  if(IDOK==MessageBox(hwnd,
"You are now selecting 17 terms to be run in the\n"
"      polynomial adjustment program.\n"
"                  Is this OK?",
    szAppName, MB_ICONQUESTION | MB_OKCANCEL)){
    if(_idd_ngcps < 17){
    MessageBox(hwnd,
    "Number of GCPs are less than the selected terms!\n"
    "         Please increase the number of GCPs\n"
    "           or decrease the number of terms.",
    szAppName, MB_ICONINFORMATION | MB_OK);
    parameter=0;}
    else
    parameter=17;}
  else if(IDCANCEL)
        break;
 break;
 case IDD_T18:

  if(IDOK==MessageBox(hwnd,
"You are now selecting 18 terms to be run in the\n"
"      polynomial adjustment program.\n"
"                  Is this OK?",
    szAppName, MB_ICONQUESTION | MB_OKCANCEL)){
    if(_idd_ngcps < 18){
    MessageBox(hwnd,
    "Number of GCPs are less than the selected terms!\n"
    "         Please increase the number of GCPs\n"
    "           or decrease the number of terms.",
    szAppName, MB_ICONINFORMATION | MB_OK);
    parameter=0;}
    else
    parameter=18;}
  else if(IDCANCEL)
        break;
 break;
 case IDD_T19:

  if(IDOK==MessageBox(hwnd,
"You are now selecting 19 terms to be run in the\n"
"      polynomial adjustment program.\n"
```

```
"                       Is this OK?",
        szAppName, MB_ICONQUESTION | MB_OKCANCEL)){
        if(_idd_ngcps < 19){
        MessageBox(hwnd,
        "Number of GCPs are less than the selected terms!\n"
        "           Please increase the number of GCPs\n"
        "              or decrease the number of terms.",
        szAppName, MB_ICONINFORMATION | MB_OK);
        parameter=0;}
        else
        parameter=19;}
      else if(IDCANCEL)
             break;
 break;
 case IDD_T20:

      if(IDOK==MessageBox(hwnd,
"You are now selecting 20 terms to be run in the\n"
"        polynomial adjustment program.\n"
"                       Is this OK?",
        szAppName, MB_ICONQUESTION | MB_OKCANCEL)){
        if(_idd_ngcps < 20){
        MessageBox(hwnd,
        "Number of GCPs are less than the selected terms!\n"
        "           Please increase the number of GCPs\n"
        "              or decrease the number of terms.",
        szAppName, MB_ICONINFORMATION | MB_OK);
        parameter=0;}
        else
        parameter=20;}
      else if(IDCANCEL)
             break;
 break;
 case IDD_T21:

      if(IDOK==MessageBox(hwnd,
"You are now selecting 21 terms to be run in the\n"
"        polynomial adjustment program.\n"
"                       Is this OK?",
        szAppName, MB_ICONQUESTION | MB_OKCANCEL)){
        if(_idd_ngcps < 21){
        MessageBox(hwnd,
        "Number of GCPs are less than the selected terms!\n"
        "           Please increase the number of GCPs\n"
        "              or decrease the number of terms.",
        szAppName, MB_ICONINFORMATION | MB_OK);
        parameter=0;}
        else
        parameter=21;}
```

```
   else if(IDCANCEL)
         break;
 break;
 case IDD_T22:

   if(IDOK==MessageBox(hwnd,
"You are now selecting 22 terms to be run in the\n"
"       polynomial adjustment program.\n"
"                    Is this OK?",
    szAppName, MB_ICONQUESTION | MB_OKCANCEL)){
    if(_idd_ngcps < 22){
    MessageBox(hwnd,
    "Number of GCPs are less than the selected terms!\n"
    "          Please increase the number of GCPs\n"
    "            or decrease the number of terms.",
    szAppName, MB_ICONINFORMATION | MB_OK);
    parameter=0;}
    else
    parameter=22;}
   else if(IDCANCEL)
         break;
 break;
 case IDD_T23:

   if(IDOK==MessageBox(hwnd,
"You are now selecting 23 terms to be run in the\n"
"       polynomial adjustment program.\n"
"                    Is this OK?",
    szAppName, MB_ICONQUESTION | MB_OKCANCEL)){
    if(_idd_ngcps < 23){
    MessageBox(hwnd,
    "Number of GCPs are less than the selected terms!\n"
    "          Please increase the number of GCPs\n"
    "            or decrease the number of terms.",
    szAppName, MB_ICONINFORMATION | MB_OK);
    parameter=0;}
    else
    parameter=23;}
   else if(IDCANCEL)
         break;
 break;
 case IDD_T24:

   if(IDOK==MessageBox(hwnd,
"You are now selecting 24 terms to be run in the\n"
"       polynomial adjustment program.\n"
"                    Is this OK?",
    szAppName, MB_ICONQUESTION | MB_OKCANCEL)){
    if(_idd_ngcps < 24){
```

```
                  MessageBox(hwnd,
                  "Number of GCPs are less than the selected terms!\n"
                  "          Please increase the number of GCPs\n"
                  "             or decrease the number of terms.",
                  szAppName, MB_ICONINFORMATION | MB_OK);
                  parameter=0;}
                  else
                  parameter=24;}
                 else if(IDCANCEL)
                        break;
              break;
              case IDD_T25:

                 if(IDOK==MessageBox(hwnd,
                  "You are now selecting 25 terms to be run in the\n"
              "        polynomial adjustment program.\n"
              "                    Is this OK?",
                  szAppName, MB_ICONQUESTION | MB_OKCANCEL)){
                  if(_idd_ngcps < 25){
                  MessageBox(hwnd,
                  "Number of GCPs are less than the selected terms!\n"
                  "          Please increase the number of GCPs\n"
                  "             or decrease the number of terms.",
                  szAppName, MB_ICONINFORMATION | MB_OK);
                  parameter=0;}
                  else
                  parameter=25;}
                 else if(IDCANCEL)
                        break;
              break;
}
break;

case IDM_3D:
     hInst=(HINSTANCE)GetWindowWord(hwnd,GWW_HINSTANCE);
     lpfnD3DlgProc = (DLGPROC)MakeProcInstance (
                              (FARPROC) D3DlgProc,hInst) ;

     if(!(DialogBox (hInst, "D3DLG", hwnd,
                              lpfnD3DlgProc)))
return 0;
     FreeProcInstance((FARPROC)lpfnD3DlgProc);

     switch(CrossTrack){
             case IDD_SPOT1A:

                 if(IDOK==MessageBox(hwnd,
             "You are now selecting stereo SPOT Level 1A\n"
             "  to be run in the bundle adjustment program.\n"
```

```
"                         Is this OK?",
        szAppName, MB_ICONQUESTION | MB_OKCANCEL))
        image_case=1;
       else
        break;
    break;

    case IDD_SPOT1B:
       if(IDOK == MessageBox(hwnd,
     "You are now selecting stereo SPOT Level 1B\n"
     "  to be run in the bundle adjustment program.\n"
     "                         Is this OK?",
          szAppName,MB_ICONQUESTION | MB_OKCANCEL))
     image_case=2;
 else
     break;
    break;

    case IDD_IRS1C:
       if(IDOK == MessageBox(hwnd,
     "You are now selecting stereo image Indian IRS-1C\n"
     "     to be run in the bundle adjustment program.\n"
     "                         Is this OK?",
          szAppName,MB_ICONQUESTION | MB_OKCANCEL))
     image_case=6;
 else
     break;

    break;

    case IDD_MOMS1:
        if(IDOK == MessageBox(hwnd,
       "You are now selecting three-fold stereo MOMS-02 Mode 1\n"
       "          to be run in the bundle adjustment program.\n"
       "                               Is this OK?",
            szAppName,MB_ICONQUESTION | MB_OKCANCEL))
            image_case=4;
        else
            break;
break;

    case IDD_MOMS3:
        if(IDOK == MessageBox(hwnd,
       "You are now selecting stereo MOMS-02 Mode 3\n"
       "    to be run in the bundle adjustment program.\n"
       "                          Is this OK?",
           szAppName, MB_ICONQUESTION | MB_OKCANCEL))
       image_case=3;
    else
```

```
                    break;
            break;

            case IDD_OPS:
                if(IDOK == MessageBox(hwnd,
                "You are now selecting stereo image Japanese OPS\n"
                "       to be run in the bundle adjustment program.\n"
                "                          Is this OK?",
                    szAppName,MB_ICONQUESTION | MB_OKCANCEL))
                image_case=5;
            else
                    break;
            break;
    }

break;
    case IDM_RUN3D:
      if(image_case==0){
      MessageBox(hwnd,
      "You have not selected the image or the correct image.\n"
      "             Please select the image name again.",
      szAppName,MB_OK);
  return 0;}

    switch(CrossTrack){
      case IDD_SPOT1A:

      if(IDOK == MessageBox(hwnd,
      "You are now running the bundle adjustment program\n"
  "     for the stereo SPOT Level 1A. Is this OK?",
      szAppName,MB_ICONQUESTION | MB_OKCANCEL)){
      hCursor = LoadCursor(NULL, IDC_WAIT);
      hOldCursor = SetCursor(hCursor);
      pushbroom();
      SetCursor(hOldCursor);
      MessageBox(hwnd,
      "The program has been run successfully for SPOT Level 1A\n"
      "      You can see the final result in pobalat.out file in the\n"
      "                 directory c:\\valadan\\thesis\\pobalat\\"
      ,szAppName,MB_OK);
  image_case=0;
      }
      else{
  image_case=0;
      return 0;}
      break;

      case IDD_SPOT1B:
```

```
  if(IDOK == MessageBox(hwnd,
  "You are now running the bundle adjustment program\n"
"       for the stereo SPOT Level 1B. Is this OK?",
  szAppName,MB_ICONQUESTION | MB_OKCANCEL)){
  hCursor = LoadCursor(NULL, IDC_WAIT);
  hOldCursor = SetCursor(hCursor);
  pushbroom();
  SetCursor(hOldCursor);
  MessageBox(hwnd,
  "The program has been run successfully for SPOT Level 1B\n"
  "       You can see the final result in pobalat.out file in the\n"
  "                directory c:\\valadan\\thesis\\pobalat\\"
  ,szAppName,MB_OK);
image_case=0;
  }
  else{
image_case=0;
  return 0;}
  break;

case IDD_IRS1C:

  if(IDOK == MessageBox(hwnd,
  "You are now running the bundle adjustment program\n"
"         for an IRS-1C stereo pair. Is this OK?",
  szAppName,MB_ICONQUESTION | MB_OKCANCEL)){
  hCursor = LoadCursor(NULL, IDC_WAIT);
  hOldCursor = SetCursor(hCursor);
  pushbroom();
  SetCursor(hOldCursor);
  MessageBox(hwnd,
  "The program has been run successfully for the IRS-1C stereo pair\n"
  "         You can see the final result in pobalat.out file in the\n"
  "                directory c:\\valadan\\thesis\\pobalat\\"
  ,szAppName,MB_OK);
image_case=0;
  }
  else{
  image_case=0;
  return 0;}
break;
 // }

// switch(AlongTrack){
case IDD_MOMS3:

  if(IDOK == MessageBox(hwnd,
  "You are now running the bundle adjustment program\n"
"     for the stereo MOMS-02 mode 3. Is this OK?",
```

```
  szAppName,MB_ICONQUESTION | MB_OKCANCEL)){
  hCursor = LoadCursor(NULL, IDC_WAIT);
  hOldCursor = SetCursor(hCursor);
  pushbroom();
  SetCursor(hOldCursor);
  MessageBox(hwnd,
  "The program has been run successfully for the stereo MOMS-02 mode 3\n"
  "           You can see the final result in pobalat.out file in the\n"
  "                   directory c:\\valadan\\thesis\\pobalat\\"
  ,szAppName,MB_OK);
image_case=0;
  }
  else{
image_case=0;
  return 0;}
  break;

  case IDD_MOMS1:

  if(IDOK == MessageBox(hwnd,
  "You are now running the bundle adjustment program\n"
  "        for the three-fold stereo MOMS-02 mode 1.\n"
  "                              Is this OK?",
  szAppName,MB_ICONQUESTION | MB_OKCANCEL)){
  hCursor = LoadCursor(NULL, IDC_WAIT);
  hOldCursor = SetCursor(hCursor);
  pushbroom();
  SetCursor(hOldCursor);
  MessageBox(hwnd,
  "The program has been run for three-fold stereo MOMS-02 mode 1\n"
  "        You can see the final result in pobalat.out file in the\n"
  "                  directory c:\\valadan\\thesis\\pobalat\\"
  ,szAppName,MB_OK);
image_case=0;
  }
  else{
image_case=0;
  return 0;}
  break;

case IDD_OPS:

  if(IDOK == MessageBox(hwnd,
  "You are now running the bundle adjustment program\n"
"         for an OPS stereo pair. Is this OK?",
  szAppName,MB_ICONQUESTION | MB_OKCANCEL)){
  hCursor = LoadCursor(NULL, IDC_WAIT);
  hOldCursor = SetCursor(hCursor);
  pushbroom();
```

```
  SetCursor(hOldCursor);
  MessageBox(hwnd,
  "The program has been run successfully for the OPS stereo pair\n"
  "       You can see the final result in pobalat.out file in the\n"
  "                directory c:\\valadan\\thesis\\pobalat\\"
  ,szAppName,MB_OK);
  image_case=0;
  }
  else{
image_case=0;
  return 0;}
break;
  }

  break;

 case IDM_RUN2D:
 if(parameter==0){
  MessageBox(hwnd,
  "You have not selected the image or the correct image.\n"
  "             Please select the image name again.",
  szAppName,MB_OK);
return 0;}

 switch (no_of_terms){
     case IDD_T3 :
       case IDD_T4 :
       case IDD_T5 :
       case IDD_T6 :
       case IDD_T7 :
       case IDD_T8 :
       case IDD_T9 :
       case IDD_T10:
       case IDD_T11:
       case IDD_T12:
       case IDD_T13:
       case IDD_T14:
       case IDD_T15:
       case IDD_T16:
       case IDD_T17:
       case IDD_T18:
       case IDD_T19:
       case IDD_T20:
       case IDD_T21:
       case IDD_T22:
       case IDD_T23:
       case IDD_T24:
       case IDD_T25:
 if(IDOK == MessageBox(hwnd,
```

```
  "You are now running the polynomial\n"
  "      adjustment program. Is this OK?",
  szAppName,MB_ICONQUESTION | MB_OKCANCEL)){
            hCursor = LoadCursor(NULL, IDC_WAIT);
            hOldCursor = SetCursor(hCursor);
            poly25();
            SetCursor(hOldCursor);
            MessageBox(hwnd,
  "The program has been run successfully for the polynomial adjustment\n"
  "       You can see the final result in residual.out file in the\n"
  "                directory c:\\valadan\\thesis\\pobalat\\"
  ,szAppName,MB_OK);
parameter=0;
  }
  else {
  parameter=0;
return 0;}
  break;
 }


  break;

   case IDM_NEW:
      if (bNeedSave && IDCANCEL ==
            AskAboutSave (hwnd, szTitleName))
         return 0 ;

      SetWindowText (hwndEdit, "\0") ;
      szFileName [0]  = '\0' ;
      szTitleName [0] = '\0' ;
      DoCaption (hwnd, szTitleName) ;
      bNeedSave = FALSE ;
      return 0 ;

   case IDM_OPEN:
      if (bNeedSave && IDCANCEL ==
            AskAboutSave (hwnd, szTitleName))
         return 0 ;

      if (PopFileOpenDlg (hwnd, szFileName, szTitleName))
         {
         if (!PopFileRead (hwndEdit, szFileName))
            {
            OkMessage (hwnd, "Could not read file %s!",
                     szTitleName) ;
            szFileName  [0] = '\0' ;
            szTitleName [0] = '\0' ;
            }
```

```
        }

    DoCaption (hwnd, szTitleName) ;
    bNeedSave = FALSE ;
    return 0 ;

case IDM_SAVE:
    if (szFileName [0])
        {
        if (PopFileWrite (hwndEdit, szFileName))
            {
            bNeedSave = FALSE ;
            return 1 ;
            }
        else
            OkMessage (hwnd, "Could not write file %s",
                        szTitleName) ;
        return 0 ;
        }
                    // fall through
case IDM_SAVEAS:
    if (PopFileSaveDlg (hwnd, szFileName, szTitleName))
        {
        DoCaption (hwnd, szTitleName) ;

        if (PopFileWrite (hwndEdit, szFileName))
            {
            bNeedSave = FALSE ;
            return 1 ;
            }
        else
            OkMessage (hwnd, "Could not write file %s",
                        szTitleName) ;
        }
    return 0 ;

case IDM_PRINT:
    if (!PopPrntPrintFile (hInst, hwnd, hwndEdit,
                    szTitleName))
        OkMessage (hwnd, "Could not print file %s",
                szTitleName) ;
    return 0 ;

case IDM_EXIT:
    SendMessage (hwnd, WM_CLOSE, 0, 0L) ;
    return 0 ;

        // Messages from Edit menu
```

```
case IDM_UNDO:
    SendMessage (hwndEdit, WM_UNDO, 0, 0L) ;
    return 0 ;

case IDM_CUT:
    SendMessage (hwndEdit, WM_CUT, 0, 0L) ;
    return 0 ;

case IDM_COPY:
    SendMessage (hwndEdit, WM_COPY, 0, 0L) ;
    return 0 ;

case IDM_PASTE:
    SendMessage (hwndEdit, WM_PASTE, 0, 0L) ;
    return 0 ;

case IDM_DEL:
    SendMessage (hwndEdit, WM_CLEAR, 0, 0L) ;
    return 0 ;

case IDM_SELALL:
    SendMessage (hwndEdit, EM_SETSEL, 0,
                MAKELONG (0, 32767)) ;
    return 0 ;

        // Messages from Search menu

case IDM_FIND:
    iOffset = HIWORD (
        SendMessage (hwndEdit, EM_GETSEL, 0, 0L)) ;
    hDlgModeless = PopFindFindDlg (hwnd) ;
    return 0 ;

case IDM_NEXT:
    iOffset = HIWORD (
        SendMessage (hwndEdit, EM_GETSEL, 0, 0L)) ;

    if (PopFindValidFind ())
        PopFindNextText (hwndEdit, &iOffset) ;
    else
        hDlgModeless = PopFindFindDlg (hwnd) ;

    return 0 ;

case IDM_REPLACE:
    iOffset = HIWORD (
        SendMessage (hwndEdit, EM_GETSEL, 0, 0L)) ;

    hDlgModeless = PopFindReplaceDlg (hwnd) ;
```

```
            return 0 ;

        case IDM_FONT:
            if (PopFontChooseFont (hwnd))
                PopFontSetFont (hwndEdit) ;

            return 0 ;

                // Messages from Help menu

        case IDM_HELP:
            OkMessage (hwnd, "Help not yet implemented!", NULL) ;
            return 0 ;

        }
    break ;

case WM_CLOSE:
        if (!bNeedSave || IDCANCEL != AskAboutSave (hwnd, szTitleName)){
        if( MessageBox(hwnd, "Terminate the program?",
                    szAppName, MB_YESNO | MB_ICONQUESTION) == IDYES
)

        DestroyWindow (hwnd) ;}

    return 0 ;

case WM_QUERYENDSESSION:
    if (!bNeedSave || IDCANCEL != AskAboutSave (hwnd, szTitleName))
        return 1L ;

    return 0 ;

case WM_DESTROY:
    PopFontDeinitialize () ;
    PostQuitMessage (0) ;
    return 0 ;

default:
        // Process "Find-Replace" messages

    if (message == messageFindReplace)
        {
        lpfr = (LPFINDREPLACE) lParam ;

        if (lpfr->Flags & FR_DIALOGTERM)
            hDlgModeless = NULL ;

        if (lpfr->Flags & FR_FINDNEXT)
```

```
                if (!PopFindFindText (hwndEdit, &iOffset, lpfr))
                    OkMessage (hwnd, "Text not found!", NULL) ;

            if (lpfr->Flags & FR_REPLACE ||
               lpfr->Flags & FR_REPLACEALL)
                if (!PopFindReplaceText (hwndEdit, &iOffset, lpfr))
                    OkMessage (hwnd, "Text not found!", NULL) ;

            if (lpfr->Flags & FR_REPLACEALL)
                while (PopFindReplaceText (hwndEdit, &iOffset, lpfr));

            return 0 ;
            }
        break ;
    }
  return DefWindowProc (hwnd, message, wParam, lParam) ;
  }


BOOL CALLBACK _export AboutDlgProc (HWND hDlg, UINT message, WPARAM
wParam,
                                    LPARAM lParam)
  {
  switch (message)
     {
     case WM_INITDIALOG:
        return TRUE ;

     case WM_COMMAND:
        switch (wParam)
            {
                case IDOK:
                case IDCANCEL:
               EndDialog (hDlg, 0) ;
                     return TRUE ;
            }
        break ;
     }
  return FALSE ;
  }


BOOL _export FAR PASCAL D3DlgProc(HWND hDlg, unsigned message,WORD
wParam, LONG lParam)
{
 static char szString[132];
 //**********************
 static HBRUSH hBrush;
 POINT point;
```

```
int x,y;
int xlparam,ylparam;
MSG msg;
//***********************

        switch (message)
        {
        case WM_INITDIALOG:
              hBrush=CreateSolidBrush(RGB(192,192,192));

              CrossTrack=IDD_SPOT1A;
              CheckRadioButton( hDlg, IDD_SPOT1A, IDD_IRS1C, IDD_SPOT1A );
              AlongTrack=IDD_MOMS1;
              CheckRadioButton( hDlg, IDD_MOMS1, IDD_OPS, IDD_SPOT1A );
        break;

        case WM_COMMAND:
                switch (wParam)
                {
                case IDOK:

                            EndDialog(hDlg, TRUE);
                            break;
                case IDCANCEL:
                            EndDialog(hDlg, FALSE);
                            break;
                case IDD_SPOT1A :
                case IDD_SPOT1B :
                case IDD_IRS1C  :
                case IDD_MOMS1  :
                case IDD_MOMS3  :
                case IDD_OPS    :
                       CrossTrack=wParam;
                       CheckRadioButton( hDlg, IDD_SPOT1A, IDD_IRS1C,wParam );
                       AlongTrack=wParam;
                       CheckRadioButton( hDlg, IDD_MOMS1, IDD_OPS, wParam );
                break;


                default:
                            return FALSE;
                }
                break;
        default:
                return FALSE;
        }
        return TRUE;
}
//*********************************************************************
```

```
BOOL _export FAR PASCAL D2DlgProc(HWND hDlg, unsigned message,WORD
wParam, LONG lParam)
{
 static char szString[132];
 //***********************
 static HBRUSH hBrush;
 POINT point;
 int x,y;
 int xlparam,ylparam;
 MSG msg;
 //***********************

        switch (message)
        {
        case WM_INITDIALOG:
              hBrush=CreateSolidBrush(RGB(192,192,192));

              no_of_terms=IDD_T3;
              CheckRadioButton( hDlg, IDD_T3, IDD_T25, IDD_T3);
              SetDlgItemText( hDlg, IDD_PIXELSIZE, "0.0" );
              SetDlgItemText( hDlg, IDD_NGCPS, "0" );
               break;

        //****************
        //****************
        case WM_COMMAND:
               switch (wParam)
               {
               case IDOK:
                GetDlgItemText( hDlg, IDD_PIXELSIZE, szString, sizeof( szString ) );
                 _idd_pixelsize = atof( szString );
                GetDlgItemText( hDlg, IDD_NGCPS, szString, sizeof( szString ) );
                          _idd_ngcps = atof( szString );
                          EndDialog(hDlg, TRUE);
                          break;
               case IDCANCEL:
                          EndDialog(hDlg, FALSE);
                          break;
               case IDD_T3 :
               case IDD_T4 :
               case IDD_T5 :
               case IDD_T6 :
               case IDD_T7 :
               case IDD_T8 :
               case IDD_T9 :
               case IDD_T10:
               case IDD_T11:
               case IDD_T12:
               case IDD_T13:
```

```
            case IDD_T14:
            case IDD_T15:
            case IDD_T16:
            case IDD_T17:
            case IDD_T18:
            case IDD_T19:
            case IDD_T20:
            case IDD_T21:
            case IDD_T22:
            case IDD_T23:
            case IDD_T24:
            case IDD_T25:
                no_of_terms=wParam;
                CheckRadioButton( hDlg, IDD_T3, IDD_T25,wParam );
            break;


            default:
                        return FALSE;
            }
            break;
        default:
            return FALSE;
        }
        return TRUE;
}
//***************************************************************************
```

```
/***************************************************************************
                File Manipulation Sub-Module of Edit Text Module
***************************************************************************/


#include <windows.h>
#include <commdlg.h>
#include <stdlib.h>

static OPENFILENAME ofn ;

void PopFileInitialize (HWND hwnd)
   {
   static char *szFilter[] = { "TEXT Files (*.TXT)",  "*.txt",
                         "OUTPUT Files (*.OUT)",  "*.out",
                              "INPUT Files (*.INP)", "*.inp",
                         "All Files (*.*)",     "*.*",
                         "" } ;

   ofn.lStructSize       = sizeof (OPENFILENAME) ;
   ofn.hwndOwner         = hwnd ;
   ofn.hInstance         = NULL ;
   ofn.lpstrFilter       = szFilter [0] ;
   ofn.lpstrCustomFilter = NULL ;
   ofn.nMaxCustFilter    = 0 ;
   ofn.nFilterIndex      = 0 ;
   ofn.lpstrFile         = NULL ;        // Set in Open and Close functions
   ofn.nMaxFile          = _MAX_PATH ;
   ofn.lpstrFileTitle    = NULL ;        // Set in Open and Close functions
   ofn.nMaxFileTitle     = _MAX_FNAME + _MAX_EXT ;
   ofn.lpstrInitialDir   = NULL ;
   ofn.lpstrTitle        = NULL ;
   ofn.Flags             = 0 ;           // Set in Open and Close functions
   ofn.nFileOffset       = 0 ;
   ofn.nFileExtension    = 0 ;
   ofn.lpstrDefExt       = "txt" ;
   ofn.lCustData         = 0L ;
   ofn.lpfnHook          = NULL ;
   ofn.lpTemplateName    = NULL ;
   }

 BOOL PopFileOpenDlg (HWND hwnd, LPSTR lpstrFileName, LPSTR lpstrTitleName)
   {
   ofn.hwndOwner         = hwnd ;
   ofn.lpstrFile         = lpstrFileName ;
   ofn.lpstrFileTitle    = lpstrTitleName ;
   ofn.Flags             = OFN_CREATEPROMPT ;

   return GetOpenFileName (&ofn) ;
```

```
    }

BOOL PopFileSaveDlg (HWND hwnd, LPSTR lpstrFileName, LPSTR lpstrTitleName)
    {
    ofn.hwndOwner       = hwnd ;
    ofn.lpstrFile       = lpstrFileName ;
    ofn.lpstrFileTitle  = lpstrTitleName ;
    ofn.Flags           = OFN_OVERWRITEPROMPT ;

    return GetSaveFileName (&ofn) ;
    }

static long PopFileLength (int hFile)
    {
    long lCurrentPos = _llseek (hFile, 0L, 1) ;
    long lFileLength = _llseek (hFile, 0L, 2) ;

    _llseek (hFile, lCurrentPos, 0) ;

    return lFileLength ;
    }

BOOL PopFileRead (HWND hwndEdit, LPSTR lpstrFileName)
    {
    long   lLength ;
    HANDLE hBuffer ;
    int    hFile ;
    LPSTR  lpstrBuffer ;

    if (-1 == (hFile = _lopen (lpstrFileName, OF_READ | OF_SHARE_DENY_WRITE)))
        return FALSE ;

    if ((lLength = PopFileLength (hFile)) >= 6000000)
        {
        _lclose (hFile) ;
        return FALSE ;
        }

    if (NULL == (hBuffer = GlobalAlloc (GHND, lLength + 1)))
        {
        _lclose (hFile) ;
        return FALSE ;
        }

    lpstrBuffer = GlobalLock (hBuffer) ;
    _lread (hFile, lpstrBuffer, (WORD) lLength) ;
    _lclose (hFile) ;
    lpstrBuffer [(WORD) lLength] = '\0' ;
```

```
    SetWindowText (hwndEdit, lpstrBuffer) ;
    GlobalUnlock (hBuffer) ;
    GlobalFree (hBuffer) ;

    return TRUE ;
    }

BOOL PopFileWrite (HWND hwndEdit, LPSTR lpstrFileName)
    {
    HANDLE hBuffer ;
    int    hFile ;
    LPSTR  lpstrBuffer ;
    WORD   wLength ;

    if (-1 == (hFile = _lopen (lpstrFileName, OF_WRITE | OF_SHARE_EXCLUSIVE)))
        if (-1 == (hFile = _lcreat (lpstrFileName, 0)))
            return FALSE ;

    wLength = GetWindowTextLength (hwndEdit) ;
    hBuffer = (HANDLE) SendMessage (hwndEdit, EM_GETHANDLE, 0, 0L) ;
    lpstrBuffer = (LPSTR) LocalLock (hBuffer) ;

    if (wLength != _lwrite (hFile, lpstrBuffer, wLength))
        {
        _lclose (hFile) ;
        return FALSE ;
        }

    _lclose (hFile) ;
    LocalUnlock (hBuffer) ;

    return TRUE ;
    }


/**************************************************************************
            Input/Output Manipulation Sub-Module of Edit Text Module
**************************************************************************/


#include <windows.h>
#include <commdlg.h>
#include <string.h>
#define MAX_STRING_LEN   256

static char szFindText [MAX_STRING_LEN] ;
static char szReplText [MAX_STRING_LEN] ;

HWND PopFindFindDlg (HWND hwnd)
```

```
   {
   static FINDREPLACE fr ;      // must be static for modeless dialog!!!

   fr.lStructSize     = sizeof (FINDREPLACE) ;
   fr.hwndOwner        = hwnd ;
   fr.hInstance       = NULL ;
        fr.Flags                    = FR_HIDEUPDOWN | FR_HIDEMATCHCASE |
FR_HIDEWHOLEWORD ;
   fr.lpstrFindWhat    = szFindText ;
   fr.lpstrReplaceWith = NULL ;
   fr.wFindWhatLen     = sizeof (szFindText) ;
   fr.wReplaceWithLen  = 0 ;
   fr.lCustData        = 0 ;
   fr.lpfnHook         = NULL ;
   fr.lpTemplateName   = NULL ;

   return FindText (&fr) ;
   }

HWND PopFindReplaceDlg (HWND hwnd)
   {
   static FINDREPLACE fr ;      // must be static for modeless dialog!!!

   fr.lStructSize     = sizeof (FINDREPLACE) ;
   fr.hwndOwner        = hwnd ;
   fr.hInstance       = NULL ;
        fr.Flags                    = FR_HIDEUPDOWN | FR_HIDEMATCHCASE |
FR_HIDEWHOLEWORD ;
   fr.lpstrFindWhat    = szFindText ;
   fr.lpstrReplaceWith = szReplText ;
   fr.wFindWhatLen     = sizeof (szFindText) ;
   fr.wReplaceWithLen  = sizeof (szReplText) ;
   fr.lCustData        = 0 ;
   fr.lpfnHook         = NULL ;
   fr.lpTemplateName   = NULL ;

   return ReplaceText (&fr) ;
   }

BOOL PopFindFindText (HWND hwndEdit, int *piSearchOffset, LPFINDREPLACE lpfr)
   {
   int          iPos ;
   LOCALHANDLE hLocal ;
   LPSTR        lpstrDoc, lpstrPos ;

              // Get a pointer to the edit document

   hLocal   = (HWND) SendMessage (hwndEdit, EM_GETHANDLE, 0, 0L) ;
   lpstrDoc = (LPSTR) LocalLock (hLocal) ;
```

```
          // Search the document for the find string

   lpstrPos = _fstrstr (lpstrDoc + *piSearchOffset, lpfr->lpstrFindWhat) ;
   LocalUnlock (hLocal) ;

          // Return an error code if the string cannot be found

   if (lpstrPos == NULL)
      return FALSE ;

          // Find the position in the document and the new start offset

   iPos = lpstrPos - lpstrDoc ;
   *piSearchOffset = iPos + _fstrlen (lpfr->lpstrFindWhat) ;

          // Select the found text

   SendMessage (hwndEdit, EM_SETSEL, 0,
             MAKELONG (iPos, *piSearchOffset)) ;

   return TRUE ;
   }

BOOL PopFindNextText (HWND hwndEdit, int *piSearchOffset)
   {
   FINDREPLACE fr ;

   fr.lpstrFindWhat = szFindText ;

   return PopFindFindText (hwndEdit, piSearchOffset, &fr) ;
   }

BOOL PopFindReplaceText (HWND hwndEdit, int *piSearchOffset, LPFINDREPLACE
lpfr)
   {
          // Find the text

   if (!PopFindFindText (hwndEdit, piSearchOffset, lpfr))
      return FALSE ;

          // Replace it

   SendMessage (hwndEdit, EM_REPLACESEL, 0, (long) lpfr->lpstrReplaceWith) ;

   return TRUE ;
   }

BOOL PopFindValidFind (void)
   {
```

```
   return *szFindText != '\0' ;
   }


/**********************************************************************
   Changing the Fonts Program of File Manipulation Sub-Module of Edit Text Module
**********************************************************************/

#include <windows.h>
#include <commdlg.h>

static LOGFONT logfont ;
static HFONT   hFont ;

BOOL PopFontChooseFont (HWND hwnd)
   {
   CHOOSEFONT cf ;

   cf.lStructSize     = sizeof (CHOOSEFONT) ;
   cf.hwndOwner       = hwnd ;
   cf.hDC             = NULL ;
   cf.lpLogFont       = &logfont ;
   cf.iPointSize      = 0 ;
   cf.Flags           = CF_INITTOLOGFONTSTRUCT | CF_SCREENFONTS
                              | CF_EFFECTS ;
   cf.rgbColors       = 0L ;
   cf.lCustData       = 0L ;
   cf.lpfnHook        = NULL ;
   cf.lpTemplateName  = NULL ;
   cf.hInstance       = NULL ;
   cf.lpszStyle       = NULL ;
   cf.nFontType       = 0 ;          // Returned from ChooseFont
   cf.nSizeMin        = 0 ;
   cf.nSizeMax        = 0 ;

   return ChooseFont (&cf) ;
   }

void PopFontInitialize (HWND hwndEdit)
   {
   GetObject (GetStockObject (SYSTEM_FONT), sizeof (LOGFONT),
                              (LPSTR) &logfont) ;
   hFont = CreateFontIndirect (&logfont) ;
   SendMessage (hwndEdit, WM_SETFONT, hFont, 0L) ;
   }

void PopFontSetFont (HWND hwndEdit)
   {
   HFONT hFontNew ;
```

```
    hFontNew = CreateFontIndirect (&logfont) ;
    SendMessage (hwndEdit, WM_SETFONT, hFontNew, 0L) ;
    DeleteObject (hFont) ;
    hFont = hFontNew ;
    }

void PopFontDeinitialize (void)
    {
    DeleteObject (hFont) ;
    }


/************************************************************************
              Input/Output Printing Sub-Module of Edit Text Module
************************************************************************/

#include <windows.h>
#include <commdlg.h>
#include <string.h>
#include "pobalat.h"

BOOL bUserAbort ;
HWND hDlgPrint ;

BOOL FAR PASCAL _export PrintDlgProc (HWND hDlg, UINT message, UINT wParam,
                                         LONG lParam)
    {
    switch (message)
        {
        case WM_INITDIALOG:
            EnableMenuItem (GetSystemMenu (hDlg, FALSE), SC_CLOSE,
                                         MF_GRAYED) ;
            return TRUE ;

        case WM_COMMAND:
            bUserAbort = TRUE ;
            EnableWindow (GetParent (hDlg), TRUE) ;
            DestroyWindow (hDlg) ;
            hDlgPrint = 0 ;
            return TRUE ;
        }
    return FALSE ;
    }

BOOL FAR PASCAL _export AbortProc (HDC hPrinterDC, short nCode)
    {
    MSG msg ;

    while (!bUserAbort && PeekMessage (&msg, NULL, 0, 0, PM_REMOVE))
```

```
     {
     if (!hDlgPrint || !IsDialogMessage (hDlgPrint, &msg))
          {
          TranslateMessage (&msg) ;
          DispatchMessage (&msg) ;
          }
     }
   return !bUserAbort ;
   }

BOOL PopPrntPrintFile (HANDLE hInst, HWND hwnd, HWND hwndEdit,
                       LPSTR szTitleName)
   {
   static PRINTDLG pd ;
   BOOL        bSuccess ;
   char        szJobName [40] ;
   FARPROC     lpfnAbortProc, lpfnPrintDlgProc ;
   NPSTR       npstrBuffer ;
   short       yChar, nCharsPerLine, nLinesPerPage, nTotalLines,
               nTotalPages, nPage, nLine, nLineNum ;
   TEXTMETRIC  tm ;
   WORD        nColCopy, nNonColCopy ;

   pd.lStructSize       = sizeof (PRINTDLG) ;
   pd.hwndOwner         = hwnd ;
   pd.hDevMode          = NULL ;
   pd.hDevNames         = NULL ;
   pd.hDC               = NULL ;
   pd.Flags             = PD_ALLPAGES | PD_COLLATE | PD_RETURNDC ;
   pd.nFromPage         = 0 ;
   pd.nToPage           = 0 ;
   pd.nMinPage          = 0 ;
   pd.nMaxPage          = 0 ;
   pd.nCopies           = 1 ;
   pd.hInstance         = NULL ;
   pd.lCustData         = 0L ;
   pd.lpfnPrintHook     = NULL ;
   pd.lpfnSetupHook     = NULL ;
   pd.lpPrintTemplateName = NULL ;
   pd.lpSetupTemplateName = NULL ;
   pd.hPrintTemplate    = NULL ;
   pd.hSetupTemplate    = NULL ;

   if (!PrintDlg (&pd))
        return TRUE ;

   nTotalLines = (short) SendMessage (hwndEdit, EM_GETLINECOUNT, 0, 0L) ;

   if (nTotalLines == 0)
```

```
    return TRUE ;

GetTextMetrics (pd.hDC, &tm) ;
yChar = tm.tmHeight + tm.tmExternalLeading ;

nCharsPerLine = GetDeviceCaps (pd.hDC, HORZRES) / tm.tmAveCharWidth ;
nLinesPerPage = GetDeviceCaps (pd.hDC, VERTRES) / yChar ;
nTotalPages   = (nTotalLines + nLinesPerPage - 1) / nLinesPerPage ;

npstrBuffer = (NPSTR) LocalAlloc (LPTR, nCharsPerLine + 1) ;

EnableWindow (hwnd, FALSE) ;

bSuccess   = TRUE ;
bUserAbort = FALSE ;

lpfnPrintDlgProc = MakeProcInstance ((FARPROC) PrintDlgProc, hInst) ;
hDlgPrint = CreateDialog (hInst, "PrintDlgBox", hwnd, lpfnPrintDlgProc) ;
SetDlgItemText (hDlgPrint, IDD_FNAME, szTitleName) ;

lpfnAbortProc = MakeProcInstance ((FARPROC) AbortProc, hInst) ;
Escape (pd.hDC, SETABORTPROC, 0, (LPSTR) lpfnAbortProc, NULL) ;

GetWindowText (hwnd, szJobName, sizeof (szJobName)) ;

if (Escape (pd.hDC, STARTDOC, strlen (szJobName), szJobName, NULL) > 0)
    {
    for (nColCopy = 0 ;
        nColCopy < (pd.Flags & PD_COLLATE ? pd.nCopies : 1) ;
        nColCopy++)
        {
        for (nPage = 0 ; nPage < nTotalPages ; nPage++)
            {
            for (nNonColCopy = 0 ;
                nNonColCopy < (pd.Flags & PD_COLLATE ? 1 : pd.nCopies);
                nNonColCopy++)
                {
                for (nLine = 0 ; nLine < nLinesPerPage ; nLine++)
                    {
                    nLineNum = nLinesPerPage * nPage + nLine ;

                    if (nLineNum > nTotalLines)
                        break ;

                    * (short *) npstrBuffer = nCharsPerLine ;

                    TextOut (pd.hDC, 0, yChar * nLine, npstrBuffer,
                        (short) SendMessage (hwndEdit, EM_GETLINE,
                        nLineNum, (LONG) (LPSTR) npstrBuffer)) ;
```

```
                }

            if (Escape (pd.hDC, NEWFRAME, 0, NULL, NULL) < 0)
                {
                bSuccess = FALSE ;
                break ;
                }

            if (bUserAbort)
                break ;
            }

        if (!bSuccess || bUserAbort)
            break ;
        }

    if (!bSuccess || bUserAbort)
        break ;
    }
  }
else
  bSuccess = FALSE ;

if (bSuccess)
  Escape (pd.hDC, ENDDOC, 0, NULL, NULL) ;

if (!bUserAbort)
  {
  EnableWindow (hwnd, TRUE) ;
  DestroyWindow (hDlgPrint) ;
  }

LocalFree ((LOCALHANDLE) npstrBuffer) ;
FreeProcInstance (lpfnPrintDlgProc) ;
FreeProcInstance (lpfnAbortProc) ;
DeleteDC (pd.hDC) ;

return bSuccess && !bUserAbort ;
}
```

```
/**************************************************************************
                    Polynomial Adjustment Module
**************************************************************************/

#include <memory.h>
#include <math.h>           //for memset()
#include <stdlib.h>
#include <fstream.h>
#include <iomanip.h>
#include "pushbrom.h"
int parameter;
float  _idd_pixelsize;

void poly25() {
double huge(*coords)[5]=new double huge[MAX][5];
double huge(*pt_matrix_a)[MAX]=new double huge[MAX][MAX];
double huge(*pt_matrix_b)[MAX]=new double huge[MAX][MAX];
double huge(*ATA)[MAX]=new double huge[MAX][MAX];
double huge(*ATL)=new double huge[MAX];
double huge(*pt_vect_l)=new double huge[MAX];
double huge(*pt_vect_x)=new double huge[MAX];
double huge(*pt_vect_c)=new double huge[MAX];
double huge(*pt_mat_r)[2]=new double huge[MAX][2];

ifstream fin;
ofstream fout;
ofstream final;

//Enter the ground coordinates of the image points in (m)a
//and their image coordinates in pixel

 fin.open("coords.inp");
 fout.open("coords.out");

   fin >> ngp >> ncp;  //reading the number of ground control points.
   for(int i=0; i< (ngp+ncp); ++i)
     for(int j=0; j<5; ++j){
          fin >> coords[i][j];}

   fout << "Number of GCPs is:" << ngp << "\t"
         << "Number of check points is:" << ncp <<"\t"
         << "Pixel size is:" << _idd_pixelsize << "\n\n";
   fout << setiosflags(ios::left);

 for( i=0;i<ngp;++i){
         for(int j=0;j<5;++j){
          if(j==0)
           fout  << coords[i][j] << "\t";
          else if(j==1)
```

```
        fout << "x = " <<  setprecision(8)<< setw(16) << coords[i][j];
       else if(j==2)
        fout << "y = " << setw(16) << coords[i][j];
       else if(j==3)
        fout << "X = " << setw(11) << coords[i][j];
       else if(j==4)
        fout << "Y = " << setw(11) << coords[i][j];
                          }
                          fout << "\n\n";
                          }
fin.close();
fout.close();
int ii;
int param;
int p_case;
//To form the matrix of coefficients
final.open("residual.out");
fout.open("param.out");
for(param=3; param<=parameter; ++param){ //open aculad number 1

for(i=0;i<(ngp+ncp);++i){

 ii=i;
  if(param>=3){
     p_case=3;
     pt_matrix_a[ii][0]=1.0;
     pt_matrix_a[ii][1]=coords[i][1];
     pt_matrix_a[ii][2]=coords[i][2]; }
  if(param>=4){
     p_case=4;
     pt_matrix_a[ii][3]=coords[i][1]*coords[i][2];}
  if(param>=5){
     p_case=5;
     pt_matrix_a[ii][4]=pow(coords[i][1],2);}
  if(param>=6){
     p_case=6;
     pt_matrix_a[ii][5]=pow(coords[i][2],2);}
  if(param>=7){
     p_case=7;
     pt_matrix_a[ii][6]=pow(coords[i][1],2)*coords[i][2];}
  if(param>=8){
     p_case=8;
     pt_matrix_a[ii][7]=coords[i][1]*pow(coords[i][2],2);}
  if(param>=9){
     p_case=9;
     pt_matrix_a[ii][8]=pow(coords[i][1],2)*pow(coords[i][2],2);}
  if(param>=10){
     p_case=10;
     pt_matrix_a[ii][9]=pow(coords[i][1],3);}
```

```
if(param>=11){
  p_case=11;
  pt_matrix_a[ii][10]=pow(coords[i][2],3);}
if(param>=12){
  p_case=12;
  pt_matrix_a[ii][11]=coords[i][1]*pow(coords[i][2],3);}
if(param>=13){
  p_case=13;
  pt_matrix_a[ii][12]=pow(coords[i][1],3)*coords[i][2];}
if(param>=14){
  p_case=14;
  pt_matrix_a[ii][13]=pow(coords[i][1],2)*pow(coords[i][2],3); }
if(param>=15){
  p_case=15;
  pt_matrix_a[ii][14]=pow(coords[i][1],3)*pow(coords[i][2],2);}
if(param>=16){
  p_case=16;
  pt_matrix_a[ii][15]=pow(coords[i][1],3)*pow(coords[i][2],3);}
if(param>=17){
  p_case=17;
  pt_matrix_a[ii][16]=pow(coords[i][1],4);}
if(param>=18){
  p_case=18;
  pt_matrix_a[ii][17]=pow(coords[i][2],4);}
if(param>=19){
  p_case=19;
  pt_matrix_a[ii][18]=pow(coords[i][1],4)*coords[i][2]; }
if(param>=20){
  p_case=20;
  pt_matrix_a[ii][19]=coords[i][3]*pow(coords[i][2],4); }
if(param>=21){
  p_case=21;
  pt_matrix_a[ii][20]=pow(coords[i][1],4)*pow(coords[i][2],2);}
if(param>=22){
  p_case=22;
  pt_matrix_a[ii][21]=pow(coords[i][1],2)*pow(coords[i][2],4);}
if(param>=23){
  p_case=23;
  pt_matrix_a[ii][22]=pow(coords[i][1],4)*pow(coords[i][2],3);}
if(param>=24){
  p_case=24;
  pt_matrix_a[ii][23]=pow(coords[i][1],3)*pow(coords[i][2],4);}
if(param>=25){
  p_case=25;
  pt_matrix_a[ii][24]=pow(coords[i][1],4)*pow(coords[i][2],4);}
}

for(int id=1; id<=2;++id){    // open aculad number 2
 for(i=0;i<(ngp+ncp);++i){
```

```
        << setw(10) << pt_mat_r[i][0] << "\t"
        << "DN(" << setw(3) << int(coords[i][0]) << ") = "
        << setw(10) << pt_mat_r[i][1] << "\n";}
       sum1=sum1+pow(pt_mat_r[i][0],2);
       sum2=sum2+pow(pt_mat_r[i][1],2);
}

final << "\nRMSE in (m) for the E and N and in (pixel) for x and y coordinates\n"
    << "for the control points are as follows:\n\n";
final << "RMSE_E = " << setw(10) << sqrt(sum1/(ngp-1)) << "\t"
    << "RMSE_x = " << setw(10) << (sqrt(sum1/(ngp-1)))/_idd_pixelsize << "\n"
    << "RMSE_N = " << setw(10) << sqrt(sum2/(ngp-1)) << "\t"
    << "RMSE_y = " << setw(10) << (sqrt(sum2/(ngp-1)))/_idd_pixelsize << "\n";

if(ncp>0){
if(param==parameter)
final << "\nThe residual errors in E and N for the Check Points are as follows:  \n\n";
sum1=0.0;
sum2=0.0;
for(i=0;i<ncp;++i){
   if(param==parameter){
   final << "DE(" << setw(3) << int(coords[i+ngp][0]) << ") = "
         << setw(10) << pt_mat_r[i+ngp][0] << "\t"
         << "DN(" << setw(3) << int(coords[i+ngp][0]) << ") = "
         << setw(10) << pt_mat_r[i+ngp][1] << "\n"; }
        sum1=sum1+pow(pt_mat_r[i+ngp][0],2);
        sum2=sum2+pow(pt_mat_r[i+ngp][1],2);
}

final << "\nRMSE in (m) for the E and N and in (pixel) for x and y coordinates\n"
    << "for the check points are as follows:\n\n";
final << "RMSE_E = " << setw(10) << sqrt(sum1/(ncp-1)) << "\t"
    << "RMSE_x = " << setw(10) << (sqrt(sum1/(ncp-1)))/_idd_pixelsize << "\n"
    << "RMSE_N = " << setw(10) << sqrt(sum2/(ncp-1)) << "\t"
    << "RMSE_y = " << setw(10) << (sqrt(sum2/(ncp-1)))/_idd_pixelsize << "\n";
final << "\n**************************************************************\n";
}// close the aculad related to if
} // close aculad number 1
fout.close();
final.close();
fout.open("vector.out");
 fout << ngp+ncp << "\n";
 for(i=0;i<ngp+ncp;++i){
     fout << setprecision(3) << setw(3) << int(coords[i][0])<< "\t"
                   << setw(15) << coords[i][1]
                         << setw(15) << coords[i][2]
                         << setw(15) << pt_mat_r[i][0]
                   << setw(15) << pt_mat_r[i][1] << "\n";
 }
```

```
  if(id==1)
    pt_vect_l[i]=coords[i][3]/1e+6;
  else
    pt_vect_l[i]=coords[i][4]/1e+6;
 }
 for(i=0;i<(ngp+ncp);++i)
  for(int j=0; j<param;++j)
       pt_matrix_a[i][j]=pt_matrix_a[i][j]/1e+6;

 for(i=0;i<ngp;++i)
  for(int j=0; j<param;++j){
       pt_matrix_b[j][i]=pt_matrix_a[i][j];}

multiply(pt_matrix_b,pt_matrix_a,ATA,param,param,ngp);  //AT*A;
amulti(pt_matrix_b,pt_vect_l,ATL,param,ngp);  //AT*L;
choleski(ATA,ATL,pt_vect_x,param);

//printing the coefficient parameters:
fout << "\n**************************************************************\n";
fout << "\n******** In the case of " << param << "parameters ********\n";
if(id==1)
   fout << "the coefficient parameters for the X are as follows:\n";
else
   fout << "the coefficient parameters for the Y are as follows:\n";

for(int ix=0;ix<param;++ix){
 fout << pt_vect_x[ix] << "\n";}

 // computing the X and Y coordinates using the computed coefficients

  amulti(pt_matrix_a,pt_vect_x,pt_vect_c,(ngp+ncp),param);
  for(i=0;i<(ngp+ncp);++i){
    if(id==1)
         pt_mat_r[i][0]=(pt_vect_l[i]-pt_vect_c[i])*1e+6;
    else
         pt_mat_r[i][1]=(pt_vect_l[i]-pt_vect_c[i])*1e+6;
  }
} // close aculad number 2

//printing the residuals for the GCPs:

final << "\n******** In the case of " << param << " parameters ********\n";
if(param==parameter)
final << "The residual errors in E and N for the Control Points are as follows:  \n\n";
double sum1=0.0;
double sum2=0.0;
for(i=0;i<ngp;++i){
   if(param==parameter){
   final << setprecision(3) << "DE(" << setw(3) << int(coords[i][0]) << ") = "
```

```
fout.close();
delete pt_matrix_a;
delete pt_matrix_b;
delete pt_vect_x;
delete pt_vect_l;
delete pt_vect_c;
delete pt_mat_r;
delete ATA;
delete ATL;
delete coords;

}
```

```
/****************************************************************************
                    The Main Bundle Adjustment Module
****************************************************************************/

#include <memory.h>
#include <math.h>
#include <stdlib.h>
#include <fstream.h>
#include <iomanip.h>
#include "pushbrom.h"
int np1,np2;
int NPO;
int image_case;
int no_of_ext_p;
float angle_l,angle_r;
float focal_b,focal_f;
float pixelc;
int ngp,ncp;

void pushbroom() {

double huge(*ground_xyz)[7]=new double huge[max_i][7];
double huge(*pixel1_xy)[2]=new double huge[max_i][2];
double huge(*pixel2_xy)[2]=new double huge[max_i][2];
float huge(*image1)[4]=new float huge[max_i][4];
float huge (*image2)[4]=new float huge[max_i][4];
double huge(*imago1)[3]=new double huge[30][3];
double huge(*imago2)[3]=new double huge[30][3];

ifstream fin;
ofstream fout;
int l_c1,l_c3,p_c1,p_c2;
int l1_c1,l1_c3,p1_c1,p1_c2;
int l2_c1,l2_c3,p2_c1,p2_c2;
float temp;

no_of_ext_p=15;
// Enter the interior orientation parameters and constant parameters which come
// from the previous calibrations, then print the entered information to be
//checked:

  fin.open("iop.inp");
  fout.open("iop.out");

  fin >> focal_b >> angle_l
     >> focal_f >> angle_r
     >> l1_c1 >> l1_c3 >> p1_c1 >> p1_c2
     >> l2_c1 >> l2_c3 >> p2_c1 >> p2_c2 >>NPO;
```

```
  fout << "Left lens" << "\n\n" << "principal distance = "
           << focal_b << "\t" << "view angle = "
           << angle_l
           << "\n\n\n"
           << "Right lens" << "\n\n" << "principal distance = "
           << focal_f << "\t" << "view angle = "
           << angle_r
           << "\n" << "Number of images= " << NPO;
  angle_r=angle_r*M_PI/180.0;
  angle_l=angle_l*M_PI/180.0;
fin.close();
fout.close();

//Enter the ground coordinates of image points and their precisions in (m)
//and then print the entered data to be checked:

 fin.open("ground.inp");
 fout.open("ground.out");

   fin >> ngp >> ncp;  //reading the number of ground control points.
   for(int i=0; i< (ngp+ncp); ++i)
     for(int j=0; j<7; ++j){
          fin >> ground_xyz[i][j];}

   fout << "Number of GCPs is:" << ngp << "\t"
       << "Number of check points is:" << ncp << "\n\n";
   fout << setiosflags(ios::left);
   for( i=0;i<ngp;++i){
         for(int j=0;j<7;++j){
          if(j==0)
           fout  << ground_xyz[i][j] << "\t";
          else if(j==1)
           fout << "X = " <<  setprecision(8)<< setw(16) << ground_xyz[i][j];
          else if(j==2)
           fout << "Y = " << setw(16) << ground_xyz[i][j];
          else if(j==3)
           fout << "Z = " << setw(16) << ground_xyz[i][j]<< "\n\t";
          else if(j==4)
           fout << "SIGMAX = " << setw(11) << ground_xyz[i][j];
          else if(j==5)
           fout << "SIGMAY = " << setw(11) << ground_xyz[i][j];
          else
           fout << "SIGMAZ = " << setw(11) << ground_xyz[i][j] ;
                         }
                         fout << "\n\n";
                         }
 fin.close();
 fout.close();
```

```
//Enter the pixel coordinates of the image points of the first image
//and then print the entered data:

fin.open("image1.inp");
fout.open("image.out");

   fin >> np1;  //number of points in image 1

   for(i=0;i<(np1+ncp);++i)
    for(int j=0;j<4;++j){
      fin >> image1[i][j];
  }
//printing the entered data:
  fout   << "Total number of image points is:" << np1
          << "\n\n\n"
       << "The pixel coordinates of the first image are as follows:\n\n"
          << "Image no." << "\t" << "Point no." << "\t"
          << "x" << "\t\t" << "y" << "\n\n";

  for(i=0;i<np1;++i){
    for(int j=0;j<4;++j){
          fout << setprecision(8)<< image1[i][j] << "\t\t";
    }
    fout << "\n\n";
  }
  fout << "\n*************************************\n\n";

//converting the pixel coordinates in the digital image (x direction left to
//right, y direction up to down with the origin in the upper left corner of
// the image) into the satellite image pixel coordinate system where x is in the
//direction of motion up to down of the screen and y is perpendicular to it
//in the direction of the lines and the origin in the upper left corner of the image:

for(i=0;i<(np1+ncp);++i){
  temp=image1[i][2];
  image1[i][2]=image1[i][3];
  image1[i][3]=temp;
}

l_c1=l1_c1;
l_c3=l1_c3;
p_c1=p1_c1;
p_c2=p1_c2;
l_b_to_a(image1,pixel1_xy,l_c1,l_c3,p_c1,p_c2);

fin.close();

//Enter the pixel coordinates of the image points of the second image
//and then print the entered data:
```

```
fin.open("image2.inp");

fin >> np2;

for(i=0;i<(np2+ncp);++i)
  for(int j=0;j<4;++j){
     fin >> image2[i][j];}

//printing the entered data:
  fout  << "Total number of image points is:" << np2
          << "\n\n\n"
        << "The pixel coordinates of the second image are as follows:\n\n"
          << "Image no." << "\t" << "Point no." << "\t"
          << "x" << "\t\t" << "y" << "\n\n";

  for(i=0;i<np2;++i){
    for(int j=0;j<4;++j){
          fout << image2[i][j] << "\t\t";
    }
    fout << "\n\n";
  }
  fout << "\n************************************\n\n";

//converting the pixel coordinates in the digital image (x direction left to
//right, y direction up to down with the origin in the upper left corner of
// the image) into the satellite image pixel coordinate system where x is in the
//direction of flight up to down of the screen and y is perpendicular to it
//in the direction of the lines and the origin in the upper left corner of the image:

for(i=0;i<(np2+ncp);++i){
  temp=image2[i][2];
  image2[i][2]=image2[i][3];
  image2[i][3]=temp;
}


l_c1=l2_c1;
l_c3=l2_c3;
p_c1=p2_c1;
p_c2=p2_c2;

l_b_to_a(image2,pixel2_xy,l_c1,l_c3,p_c1,p_c2);

fin.close();
for(i=0;i<np1;++i){
  for(int j=0; j<4;++j){
    fout << image1[i][j] << "\t";}
    fout << "\n";}
for(i=0;i<np2;++i){
```

```
  for(int j=0; j<4;++j){
    fout << image2[i][j] << "\t";}
    fout << "\n";}
for(i=0;i<np1;++i){
  for(int j=0; j<2;++j){
    fout << pixel1_xy[i][j] << "\t";}
    fout << "\n";}
for(i=0;i<np2;++i){
  for(int j=0; j<2;++j){
    fout << pixel2_xy[i][j] << "\t";}
    fout << "\n";}

fout.close();

//Enter the approximation values for the exterior orientaion parameters for
//the orientation images of each image:

fin.open("eop1.inp");
  for(i=0;i<22;++i)
   for(int j=0;j<3;++j)
     fin >> imago1[i][j];
   imago1[19][0]=imago1[19][0]*M_PI/180;        //omega0
   imago1[19][1]=imago1[19][1]*M_PI/180;       //phi0
   imago1[19][2]=imago1[19][2]*M_PI/180;      //kappa0
   imago1[20][0]=imago1[20][0]*M_PI/180;     //omega1
   imago1[20][1]=imago1[20][1]*M_PI/180;    //phi1
   imago1[20][2]=imago1[20][2]*M_PI/180;   //kappa1
   imago1[21][0]=imago1[21][0]*M_PI/180;  //omega2
   imago1[21][1]=imago1[21][1]*M_PI/180; //phi2
   imago1[21][2]=imago1[21][2]*M_PI/180; //kappa2
fin.close();

fin.open("eop2.inp");

  for(i=0;i<22;++i)
   for(int j=0;j<3;++j)
     fin >> imago2[i][j];

   imago2[19][0]=imago2[19][0]*M_PI/180;        //omega0
   imago2[19][1]=imago2[19][1]*M_PI/180;       //phi0
   imago2[19][2]=imago2[19][2]*M_PI/180;      //kappa0
   imago2[20][0]=imago2[20][0]*M_PI/180;     //omega1
   imago2[20][1]=imago2[20][1]*M_PI/180;    //phi1
   imago2[20][2]=imago2[20][2]*M_PI/180;   //kappa1
   imago2[21][0]=imago2[21][0]*M_PI/180;  //omega2
   imago2[21][1]=imago2[21][1]*M_PI/180; //phi2
   imago2[21][2]=imago2[21][2]*M_PI/180; //kappa2

  fin.close();
```

```
//Printing the above entered data:

fout.open("eop.out");

  fout << "For the first image: ";
  fout << "\n\n";

  for(i=0;i<8;++i){
  fout << "X0(" << i+1 << ")= " << imago1[i][0] << "\t"
     << "Y0(" << i+1 << ")= " << imago1[i][1] << "\t"
     << "Z0(" << i+1 << ")= " << imago1[i][2] << "\n";}
  for(i=0;i<8;++i){
  fout << "VX0(" << i+1 << ")= " << imago1[i+8][0] << "\t"
     << "VY0(" << i+1 << ")= " << imago1[i+8][1] << "\t"
     << "VZ0(" << i+1 << ")= " << imago1[i+8][2] << "\n";}
  fout << "\n";
  fout << "t0(1)= " << imago1[16][0] << "\t"
     << "t0(2)= " << imago1[16][1] << "\t"
     << "t0(3)= " << imago1[16][2] << "\n"
     << "t0(4)= " << imago1[17][0] << "\t"
     << "t0(5)= " << imago1[17][1] << "\t"
     << "t0(6)= " << imago1[17][2] << "\n"
     << "t0(7)= " << imago1[18][0] << "\t"
     << "t0(8)= " << imago1[18][1] << "\t"
     << "t0(c)= " << imago1[18][2] << "\n";
  fout <"\n";
  fout << "omega0 = " << imago1[19][0] << "\tphi0 = " << imago1[19][1]
     << "\tkappa0 = " << imago1[19][2] << "\n";

  fout << "omega1 = " << imago1[20][0] << "\tphi1 = " << imago1[20][1]
     << "\tkappa1 = " << imago1[20][2] << "\n";

  fout << "omega2 = " << imago1[21][0] << "\tphi2 = " << imago1[21][1]
     << "\tkappa2 = " << imago1[21][2] << "\n\n\n\n";

  fout << "For the second image: ";
  fout << "\n\n";

  for(i=0;i<8;++i){
  fout << "X0(" << i+1 << ")= " << imago2[i][0] << "\t"
     << "Y0(" << i+1 << ")= " << imago2[i][1] << "\t"
     << "Z0(" << i+1 << ")= " << imago2[i][2] << "\n";}
  for(i=0;i<8;++i){
  fout << "VX0(" << i+1 << ")= " << imago2[i+8][0] << "\t"
     << "VY0(" << i+1 << ")= " << imago2[i+8][1] << "\t"
     << "VZ0(" << i+1 << ")= " << imago2[i+8][2] << "\n";}
  fout << "\n";
  fout << "t0(1)= " << imago2[16][0] << "\t"
     << "t0(2)= " << imago2[16][1] << "\t"
```

```
    << "t0(3)= " << imago2[16][2] << "\n"
    << "t0(4)= " << imago2[17][0] << "\t"
    << "t0(5)= " << imago2[17][1] << "\t"
    << "t0(6)= " << imago2[17][2] << "\n"
    << "t0(7)= " << imago2[18][0] << "\t"
    << "t0(8)= " << imago2[18][1] << "\t"
    << "t0(c)= " << imago2[18][2] << "\n";
 fout <"\n";
 fout << "omega0 = " << imago2[19][0] << "\tphi0 = " << imago2[19][1]
    << "\tkappa0 = " << imago2[19][2] << "\n";

 fout << "omega1 = " << imago2[20][0] << "\tphi1 = " << imago2[20][1]
    << "\tkappa1 = " << imago2[20][2] << "\n";

 fout << "omega2 = " << imago2[21][0] << "\tphi2 = " << imago2[21][1]
    << "\tkappa2 = " << imago2[21][2] << "\n\n\n\n";

fout.close();

adjust(image1,image2,ground_xyz,imago1,imago2,pixel1_xy,pixel2_xy);

//*********************************************************************
//*********************************************************************

delete image1;
delete image2;
delete imago1;
delete imago2;
delete ground_xyz;
delete pixel1_xy;
delete pixel2_xy;

}

/*************************************************************************
    Image Coordinate Transformation Sub-Modules of Bundle Adjustment Module
*************************************************************************/

#include <math.h>
#include <fstream.h>
#include <iomanip.h>
#include "pushbrom.h"

void l_b_to_a(float huge (*image_xy)[4],double huge (*pixel_xy)[2],
            int l_c1,int l_c3,
            int p_c1,int p_c2)
{

ofstream level;
```

```
level.open("level.out");

//converting the pixel coordinate in Level 1B to the pixel coordinate in Level 1A

level << "\n" << p_c1 << "\t" << p_c2 << "\t" << l_c1 << "\t" << l_c3 << "\n";
double coef_p=6000.0/(p_c2-p_c1+1);
double coef_l=6000.0/(l_c3);

level << coef_p << "\t" << coef_l << "\n";

 for(int i=0;i<(np1+ncp);++i){
  level << image_xy[i][2] << "\t" << image_xy[i][3] << "\n";


  if(image_case==2){                    // SPOT Level 1B
  pixel_xy[i][0]=image_xy[i][2]*coef_l;
  pixel_xy[i][1]=(image_xy[i][3]-(l_c3-image_xy[i][2]+1)*(p_c1/l_c3))*coef_p;
  }
  else if(image_case==1){               //SPOT Level 1A
  pixel_xy[i][0]=image_xy[i][2];
  pixel_xy[i][1]=image_xy[i][3];
  }
  else if(image_case==3 || image_case==4){
  pixel_xy[i][0]=image_xy[i][2]-l_c1;
  pixel_xy[i][1]=image_xy[i][3]-p_c1;
  }

  level << pixel_xy[i][0] << "\t" << pixel_xy[i][1] << "\n";

//converting the pixel coordinates in Level 1A to the image coordinate system
//with its origin in the projection centre of each line which is the
//centre of each scan line, the x direction is in the direction of flight,
//and the y direction is perpendicular to x in the plane of the scan line:
float pixel_size;
float p_y_c;
if(image_case==4){          // MOMS Mode 1
  pixel_size=0.010;
  p_y_c=1488.5;}

else if(image_case==3){          //MOMS Mode 3
  pixel_size=0.010;
  p_y_c=2900.5; }


else{
  pixel_size=0.013;
  p_y_c=3000;}

 image_xy[i][2]=((pixel_xy[i][0]-int(pixel_xy[i][0])-0.5)*pixel_size)/1000.0;
```

```
 image_xy[i][3]=((pixel_xy[i][1]-p_y_c)*pixel_size)/1000.0;
 level << image_xy[i][2] << "\t" << image_xy[i][3] <<"\n";
 level << "\n************************\n";
 }
level.close();
}

/**************************************************************************
      The Bundle Adjustment Program (Case 2) including the Space Resection and
        Intersection Procedure (A Sub-Module of the Bundle Adjustment Module)
**************************************************************************/


#include <fstream.h>
#include <iomanip.h>
#include <math.h>
#include <memory.h>
#include "pushbrom.h"


double UU,VV,WW;
double X_prj,Y_prj,Z_prj;

double e;                    // a is the semi-major axis and e is the eccentricity
double Cx,Cy;
float focal;
float view_angle;


void adjust(float huge(*image1)[4], float huge(*image2)[4], double huge(*ground_xyz)[7],
            double huge(*imago1)[3],double huge (*imago2)[3],
         double huge(*pixel1_xy)[2], double huge (*pixel2_xy)[2])
{

 double huge(*Be)[15]=new double huge[2][15];
 double huge(*eop_obs)[3]=new double huge[max_i][3];
 double huge(*eop_it)[3]=new double huge[max_i][3];
 double huge(*image_eo)[3]=new double huge[30][3];
 float huge(*image_xy)[4]=new float huge[max_i][4];
 double huge(*pixel_xy)[2]=new double huge[max_i][2];

 double huge (*pt_vector_v)= new double huge[MAX];
 double huge (*pt_vector_x)= new double huge[MAX];

 double huge (*pt_matrix_a)[MAX]= new double huge[MAX][MAX];
 double huge (*pt_matrix_b)[MAX]= new double huge[MAX][MAX];
 double huge (*pt_matrix_c)[MAX]= new double huge[MAX][MAX];

 double huge (*omega)[1]=new double[max_i][1];
 double huge (*phi)[1]=new double[max_i][1];
```

```
double huge (*kapa)[1]=new double huge [max_i][1];
double huge (*F)[1]=new double[max_i][1];
double huge (*C_OMEGA)[1]=new double huge [max_i][1];
double huge (*r)[1]=new double huge [max_i][1];
float huge (*X0)[1]=new float huge [max_i][1];
float huge (*Y0)[1]=new float huge [max_i][1];
float huge (*Z0)[1]=new float huge [max_i][1];

double huge (*BBAR_11)[MAX]=new double huge [MAX][MAX];
double huge (*mat_u1)=new double huge [MAX];
double huge (*BBAR_21)[MAX]=new double huge [MAX][MAX];
double huge (*BTWB1)[MAX]=new double huge [MAX][MAX];
double huge (*BTWB2)[MAX]=new double huge [MAX][MAX];
double huge (*BTWE1)=new double huge [MAX];
double huge (*BTWE2)=new double huge [MAX];
double huge (*BTWE3)=new double huge [MAX];
double huge (*EBAR1)=new double huge [MAX];
double huge (*EBAR2)=new double huge [MAX];
double huge (*EBARg)=new double huge [MAX];
double huge (*normal_11)[MAX]=new double huge [MAX][MAX];
double huge (*mat_x1)=new double huge [MAX];
double huge (*weight)=new double huge [MAX];
double huge (*XCP)=new double huge [max_i];
double huge (*YCP)=new double huge [max_i];
double huge (*ZCP)=new double huge [max_i];
double huge (*DXX)=new double huge [max_i];
double huge (*DYY)=new double huge [max_i];
double huge (*DZZ)=new double huge [max_i];
double huge (*landa_r)=new double huge [max_i];
double huge (*xpix_l)=new double huge [max_i];
double huge (*ypix_l)=new double huge [max_i];
double huge (*xpix_r)=new double huge [max_i];
double huge (*ypix_r)=new double huge [max_i];


double XR,YR,ZR;
double UR,VR,WR;
double XL,YL,ZL;
double UL,VL,WL;
int pixely;
ofstream final;
final.open("final.out");
ofstream fout;

float XP;
float YP;
double a_of_p; //argument of preigee

lagrange(imago1);
```

```
fout.open("imago1.out");
fout << "\n" << imago1[0][0] ;
fout << "\n" << imago1[0][1] << "\t" << imago1[0][2] << "\t" << imago1[1][0];
fout << "\n" << imago1[1][1] << "\t" << imago1[1][2] << "\t" << imago1[2][0];
fout << "\n" << imago1[2][1] << "\t" << imago1[2][2] << "\t" << imago1[3][0];
fout << "\n" << imago1[3][1] << "\t" << imago1[3][2] << "\t" << imago1[4][0];
fout << "\n" << imago1[4][1] << "\t" << imago1[4][2] << "\n";
fout << "\n" << imago1[5][0] << "\t" << imago1[5][1] << "\t" << imago1[5][2]
   << "\n" << imago1[6][0];
double    a_of_p1=imago1[5][2];  // argument of prigee
double    e1=imago1[6][0];       // eccentricity
imago1[8][0]=imago1[1][1];
imago1[8][1]=imago1[1][2];
imago1[1][1]=imago1[2][0];  //imago1[1][1] becomes omega(0)
imago1[1][2]=imago1[2][1];  //imago1[1][2] becomes phi(0)
imago1[2][0]=imago1[2][2];  //imago1[2][0] becomes kapa(0)
imago1[2][1]=imago1[8][0];  //imago1[2][1] becomes F(1)
imago1[2][2]=imago1[8][1];  //imago1[2][2] becomes common_mega(1)
fout.close();

fout.open("imago2.out");
lagrange(imago2);

fout << "\n" << imago2[0][0] ;
fout << "\n" << imago2[0][1] << "\t" << imago2[0][2] << "\t" << imago2[1][0];
fout << "\n" << imago2[1][1] << "\t" << imago2[1][2] << "\t" << imago2[2][0];
fout << "\n" << imago2[2][1] << "\t" << imago2[2][2] << "\t" << imago2[3][0];
fout << "\n" << imago2[3][1] << "\t" << imago2[3][2] << "\t" << imago2[4][0];
fout << "\n" << imago2[4][1] << "\t" << imago2[4][2] << "\n";
fout << "\n" << imago2[5][0] << "\t" << imago2[5][1] << "\t" << imago2[5][2]
   << "\n" << imago2[6][0];
fout.close();
double    a_of_p2=imago2[5][2];  // argument of prigee
double    e2=imago2[6][0];       // eccentricity
imago2[8][0]=imago2[1][1];
imago2[8][1]=imago2[1][2];
imago2[1][1]=imago2[2][0];  //imago2[1][1] becomes omega(0)
imago2[1][2]=imago2[2][1];  //imago2[1][2] becomes phi(0)
imago2[2][0]=imago2[2][2];  //imago2[2][0] becomes kapa(0)
imago2[2][1]=imago2[8][0];  //imago2[2][1] becomes F(1)
imago2[2][2]=imago2[8][1];  //imago2[2][2] becomes common_mega(1)

if(no_of_ext_p==8){
eop_obs[0][0]=imago1[0][0];
eop_obs[0][1]=imago1[0][1];
eop_obs[0][2]=imago1[0][2];
eop_obs[1][0]=imago1[1][0];
eop_obs[1][1]=imago1[1][1];
eop_obs[1][2]=imago1[1][2];
```

```
eop_obs[2][0]=imago1[2][0];
eop_obs[2][1]=imago1[3][1];
eop_obs[2][2]=imago2[0][0];
eop_obs[3][0]=imago2[0][1];
eop_obs[3][1]=imago2[0][2];
eop_obs[3][2]=imago2[1][0];
eop_obs[4][0]=imago2[1][1];
eop_obs[4][1]=imago2[1][2];
eop_obs[4][2]=imago2[2][0];
eop_obs[5][0]=imago2[3][1];}
else if(no_of_ext_p==7){
eop_obs[0][0]=imago1[0][0];
eop_obs[0][1]=imago1[0][1];
eop_obs[0][2]=imago1[0][2];
eop_obs[1][0]=imago1[1][0];
eop_obs[1][1]=imago1[1][1];
eop_obs[1][2]=imago1[1][2];
eop_obs[2][0]=imago1[3][1];
eop_obs[2][1]=imago2[0][0];
eop_obs[2][2]=imago2[0][1];
eop_obs[3][0]=imago2[0][2];
eop_obs[3][1]=imago2[1][0];
eop_obs[3][2]=imago2[1][1];
eop_obs[4][0]=imago2[1][2];
eop_obs[4][1]=imago2[3][1];}

else{
 for(int ieo=0; ieo<(no_of_ext_p)*2/3; ++ieo){
  for(int jeo=0; jeo<3; ++jeo){
    if(ieo<(no_of_ext_p)/3)
      eop_obs[ieo][jeo]=imago1[ieo][jeo];
    else
      eop_obs[ieo][jeo]=imago2[ieo-(no_of_ext_p)/3][jeo];
  }
 }
}

//************************************************************************
//************************************************************************
//****************** Start of the iteration *********************************
//************************************************************************
//************************************************************************
float size;
for(int it=1; it<=5;++it){   //open the iteration acolad

for (int i=0; i<MAX; ++i){
  for (int j=0; j<MAX; ++j){
    BBAR_11[i][j]=0.0;
    BBAR_21[i][j]=0.0;}}
```

```
int np=np1;
final << "\n\n";
for(i=0;i<np;++i) {
  for(int j=0;j<4;++j){
    image_xy[i][j]=image1[i][j];
    final << image_xy[i][j] << "\t";}
    final << "\n";}

for(i=0;i<np;++i)
  for(int j=0;j<2;++j)
    pixel_xy[i][j]=pixel1_xy[i][j];


//Placing the elements of the matrices imago1, imago2, and imago3
//in a handle matrix as image_eo. id is the parameter which indicates the
//number of photographs (NPO). NPO is two in SPOT and MOMS-02, mode 3 and
//is equal to three in the case of MOMS-02 mode 1.

for(int id=1; id<=NPO; ++id){//open the acolad number 0.
  if(id==2){
    np=np2;
    for(int i9=1; i9<=np; ++i9)
      for(int j9=1; j9<=4; ++j9)
           image_xy[i9-1][j9-1]=image2[i9-1][j9-1];

     for(i9=1; i9<=np; ++i9)
      for(int j9=1; j9<=2; ++j9)
           pixel_xy[i9-1][j9-1]=pixel2_xy[i9-1][j9-1];

            // in the SPOT case, we consider just the centre line.
     final << "\nfor the second image\n";
    if(no_of_ext_p==8){
     for(int io=0; io<3; ++io) {
       for(int jo=0; jo<3; ++jo){
          image_eo[io][jo]=imago2[io][jo];}}
          image_eo[3][0]=imago2[3][1];
          eop_it[2][2]=imago2[0][0];
       eop_it[3][0]=imago2[0][1];
       eop_it[3][1]=imago2[0][2];
       eop_it[3][2]=imago2[1][0];
       eop_it[4][0]=imago2[1][1];
       eop_it[4][1]=imago2[1][2];
          eop_it[4][2]=imago2[2][0];
          eop_it[5][0]=imago2[3][1];
          }
    else if(no_of_ext_p==7){
      for(int io=0; io<3; ++io) {
        for(int jo=0; jo<3; ++jo){
          image_eo[io][jo]=imago2[io][jo];}}
```

```
      image_eo[3][0]=imago2[3][1];
      eop_it[2][1]=imago2[0][0];
    eop_it[2][2]=imago2[0][1];
    eop_it[3][0]=imago2[0][2];
    eop_it[3][1]=imago2[1][0];
    eop_it[3][2]=imago2[1][1];
    eop_it[4][0]=imago2[1][2];
    eop_it[4][1]=imago2[3][1];
  }
 else {
  for(int io=0; io<(no_of_ext_p)/3; ++io) {
    for(int jo=0; jo<3; ++jo){
      image_eo[io][jo]=imago2[io][jo];
      eop_it[io+(no_of_ext_p)/3][jo]=imago2[io][jo]; }}
 }

      a_of_p=a_of_p2;  // argument of perigee
    e=e2;       // eccentricity
      view_angle = angle_r;
 if(image_case==3){
  size=0.010;
  pixelc=4060.5;
  pixely=2900.5;
 }
 else if(image_case==4){
  size=0.010;
  pixelc=4060.5;
  pixely=1488.5;
 }
 else{
 size=0.013;
 pixelc=3000;//pixel_cr;
 pixely=3000;//3762;
  }
}

else if(id==1){
 if(image_case==3){
   size=0.010;
   pixelc=4060.5;
   pixely=2900.5;
  }
  else if(image_case==4){
   size=0.010;
   pixelc=4060.5;
   pixely=1488.5;
  }

  else{
```

```
size=0.013;
pixelc=3000;//pixel_cl;
pixely=3000;//3865;
}

final << "\nfor the first image\n";
if(no_of_ext_p==8){
  for(int io=0; io<3; ++io){
    for(int jo=0; jo<3; ++jo){
       image_eo[io][jo]=imago1[io][jo];}}
       image_eo[3][0]=imago1[3][1];
       eop_it[0][0]=imago1[0][0];
    eop_it[0][1]=imago1[0][1];
    eop_it[0][2]=imago1[0][2];
    eop_it[1][0]=imago1[1][0];
    eop_it[1][1]=imago1[1][1];
    eop_it[1][2]=imago1[1][2];
       eop_it[2][0]=imago1[2][0];
       eop_it[2][1]=imago1[3][1];
 }
 else if(no_of_ext_p==7){
  for(int io=0; io<3; ++io){
    for(int jo=0; jo<3; ++jo){
       image_eo[io][jo]=imago1[io][jo];}}
       eop_it[0][0]=imago1[0][0];
    eop_it[0][1]=imago1[0][1];
    eop_it[0][2]=imago1[0][2];
    eop_it[1][0]=imago1[1][0];
    eop_it[1][1]=imago1[1][1];
    eop_it[1][2]=imago1[1][2];
    eop_it[2][0]=imago1[3][1];
 }
 else {
  for(int io=0; io<(no_of_ext_p)/3; ++io){
    for(int jo=0; jo<3; ++jo){
       image_eo[io][jo]=imago1[io][jo];
       eop_it[io][jo]=imago1[io][jo]; }}
 }
       a_of_p=a_of_p1;  // argument of perigee
    e=e1;       // eccentricity
       view_angle = angle_l;}

       // initializing the focal length

        if(id==1)
         focal=focal_b/1000.0;
    else if (id==2)
         focal=focal_f/1000.0;
```

```
//*****************************************************************
//*****************************************************************
 //Placing the elements of matrix image_eo related to the exterior
 //orientation parameters in 18 different vectors as follows:


  for(int ii=1; ii<=np; ++ii){   //open the acolad number 1 related to spot.

       XP=0.0;   //x-x0
       YP=image_xy[ii-1][3];    //y-y0

   F[ii-1][0]=image_eo[0][0]+image_eo[2][1]*((pixel_xy[ii-1][0]-pixelc)*size);
    C_OMEGA[ii-1][0]=image_eo[0][1]+image_eo[2][2]*((pixel_xy[ii-1][0]-pixelc)*size);
   r[ii-1][0]=image_eo[1][0]*(1-pow(e,2))/(1+e*cos(F[ii-1][0]));
 if(no_of_ext_p == 15){
    if(image_case==1 || image_case==3 || image_case==4){
       omega[ii-1][0]=image_eo[1][1]+image_eo[3][0]*((pixel_xy[ii-1][0]-pixelc)*size)+
                   image_eo[4][0]*pow(((pixel_xy[ii-1][0]-pixelc)*size),2);
       phi[ii-1][0]=image_eo[1][2]+image_eo[3][1]*((pixel_xy[ii-1][0]-pixelc)*size)+
                   image_eo[4][1]*pow(((pixel_xy[ii-1][0]-pixelc)*size),2);}
       else if(image_case==2){
       omega[ii-1][0]=image_eo[1][1]+image_eo[3][0]*((pixel_xy[ii-1][0]-pixelc)*size)+
                   image_eo[4][0]*pow(((pixel_xy[ii-1][1]-pixely)*size),2);
       phi[ii-1][0]=image_eo[1][2]+image_eo[3][1]*((pixel_xy[ii-1][0]-pixelc)*size)+
                   image_eo[4][1]*pow(((pixel_xy[ii-1][1]-pixely)*size),2);}

       kapa[ii-1][0]=image_eo[2][0]+image_eo[3][2]*((pixel_xy[ii-1][0]-pixelc)*size)+
                   image_eo[4][2]*pow(((pixel_xy[ii-1][0]-pixelc)*size),2);
   }
   else if(no_of_ext_p == 12){
       omega[ii-1][0]=image_eo[1][1]+image_eo[3][0]*((pixel_xy[ii-1][0]-pixelc)*size);
       phi[ii-1][0]=image_eo[1][2]+image_eo[3][1]*((pixel_xy[ii-1][0]-pixelc)*size);
       kapa[ii-1][0]=image_eo[2][0]+image_eo[3][2]*((pixel_xy[ii-1][0]-pixelc)*size);
   }
   else if(no_of_ext_p == 9 ){
       omega[ii-1][0]=image_eo[1][1];
       phi[ii-1][0]=image_eo[1][2];
       kapa[ii-1][0]=image_eo[2][0];
   }
    else if(no_of_ext_p == 8){
       omega[ii-1][0]=image_eo[1][1];
       phi[ii-1][0]=image_eo[1][2]+image_eo[3][0]*((pixel_xy[ii-1][0]-pixelc)*size);
       kapa[ii-1][0]=image_eo[2][0];
   }
    else if(no_of_ext_p == 7){
       omega[ii-1][0]=image_eo[1][1];
       phi[ii-1][0]=image_eo[1][2]+image_eo[3][0]*((pixel_xy[ii-1][0]-pixelc)*size);
     if(id==1)
       kapa[ii-1][0]=-0.004;
```

```
      if(id==2)
      kapa[ii-1][0]= 0.003;
    }
    be_bg(Be,phi[ii-1][0],omega[ii-1][0],kapa[ii-1][0],F[ii-1][0],
          C_OMEGA[ii-1][0],ground_xyz[ii-1][1],
          ground_xyz[ii-1][2],ground_xyz[ii-1][3],pixel_xy[ii-1][0],
          pixel_xy[ii-1][1],pixely,r[ii-1][0],a_of_p,image_eo[0][2],0,0,size);

//***********************************************************************
   //Placing the elements of matrix Be into the general matrix BBAR_11
   //and BBAR_21 for the first image and second image respectively:
//***********************************************************************

    int kk;
    if(id==1){
       kk=ii*2-1;
       int ik=no_of_ext_p;
       final << "ii=" << ii << "\t and id=" << id << "\n";
       for(int i5=1; i5<=no_of_ext_p; ++i5){
         BBAR_11[kk-1][i5-1]=Be[0][i5-1];
       BBAR_11[kk-1][i5+ik-1]=0.0;
         BBAR_11[kk][i5-1]=Be[1][i5-1];
         BBAR_11[kk][i5+ik-1]=0.0;

       }
    }
    else if(id==2){
       kk=ii*2-1;
       int ik=no_of_ext_p;
       final << "ii=" << ii << "\t and id=" << id << "\n";
       for(int i5=1; i5<=no_of_ext_p; ++i5){
      BBAR_21[kk-1][i5-1]=0.0;
        BBAR_21[kk-1][i5+ik-1]=Be[0][i5-1];
        BBAR_21[kk][i5+ik-1]=Be[1][i5-1];
        BBAR_21[kk][i5-1]=0.0;

     }
    }


    //Form the matrix EBAR:

    if(id==1){
       EBAR1[kk-1]=XP-Cx;
       EBAR1[kk]=YP-Cy;
       final << "yp1-cy1=" << EBAR1[kk] << "\t"
            << "xp1-cx1=" << EBAR1[kk-1]<< "\n";}

    else if(id==2){
```

```
        EBAR2[kk-1]=XP-Cx;
        EBAR2[kk]=YP-Cy;

        final << "yp2-cy2=" << EBAR2[kk]<< "\t"
              << "xp2-cx2=" << EBAR2[kk-1]<< "\n";}


  }  // close acolad number 1.
 }  //close acolad number 0.

//*****************************************************************
//*****************************************************************


//Adding to matrix EBAR, the elements related to the quasi-observations
//of EOPs:

fout.open("deop.out");
int kj8;
if(no_of_ext_p==7){
 for(int i8=1; i8<=4; ++i8){
  for(int j8=1; j8<=3; ++j8){
    kj8=(i8-1)*3+j8;
   EBARg[kj8-1]=eop_obs[i8-1][j8-1]-eop_it[i8-1][j8-1];}}
   EBARg[12]=eop_obs[4][0]-eop_it[4][0];
   EBARg[13]=eop_obs[4][1]-eop_it[4][1];
}

else if(no_of_ext_p==8){
 for(int i8=1; i8<=5; ++i8){
  for(int j8=1; j8<=3; ++j8){
    kj8=(i8-1)*3+j8;
   EBARg[kj8-1]=eop_obs[i8-1][j8-1]-eop_it[i8-1][j8-1];}}
   EBARg[15]=eop_obs[5][0]-eop_it[5][0];
}
else{
 for(int i8=1; i8<=(no_of_ext_p)*2/3; ++i8)
  for(int j8=1; j8<=3; ++j8){
   int kj8=(i8-1)*3+j8;
   EBARg[kj8-1]=eop_obs[i8-1][j8-1]-eop_it[i8-1][j8-1];
   fout << eop_obs[i8-1][j8-1] << "\t" << eop_it[i8-1][j8-1] << "\n";
   fout << EBARg[kj8-1]<< "\n";
  }
}
fout.close();


//computing of the matrix N which has been divided into four submatrix
//N11, N12, N21, N22. In this case:
```

```
//N11=normal_11=BBART_11*W1_bar*BBAR_11+BBART_21*W2_bar*BBAR_21+
//          BBART_31*W3_bar*BBAR_31.

int ncb11,ncb21,ncb31;
int nrb11,nrb21,nrb31;

  ncb11=(no_of_ext_p)*NPO;
  ncb21=(no_of_ext_p)*NPO;
  ncb31=(no_of_ext_p)*NPO;
  nrb11=np1*2;
  nrb21=np2*2;
  nrb31=(no_of_ext_p)*NPO;

fout.open("bbar11.out");

for(int ix=0; ix<nrb11; ++ix) {
 for(int jx=0; jx<ncb11; ++jx) {
 pt_matrix_a[jx][ix]=BBAR_11[ix][jx];
 pt_matrix_b[ix][jx]=BBAR_11[ix][jx];}}

multiply(pt_matrix_a,pt_matrix_b,BTWB1,ncb11,ncb11,nrb11);

for(ix=0; ix<nrb11; ++ix) {
 for( int jx=0; jx<ncb11; ++jx) {
  fout << BBAR_11[ix][jx] << "\t";}
  fout << "\n"; }

fout.close();


for(ix=0; ix<nrb21; ++ix) {
 for(int jx=0; jx<ncb21; ++jx) {
  pt_matrix_b[jx][ix]=BBAR_21[ix][jx];
  pt_matrix_a[ix][jx]=BBAR_21[ix][jx];}}

multiply(pt_matrix_b,pt_matrix_a,BTWB2,ncb21,ncb21,nrb21);

fout.open("bbar21.out");
fout <<"row=" << nrb21 << "\t" << "column=" << ncb21 << "\n";
for( ix=0; ix<ncb21; ++ix) {
 for( int jx=0; jx<ncb21; ++jx) {
 double test=BTWB2[ix][jx]-BTWB2[jx][ix];
 if(test!=0.0)
  fout << ix << "\t" << jx << "\t" << test << "\n";}}

fout.close();

for(ix=1; ix<=2;++ix){
 weight[(ix-1)*(no_of_ext_p)]=1.0/pow(0.02,2);  //F(0)
```

```
 weight[(ix-1)*(no_of_ext_p)+1]=1.0/pow(0.02,2); //C_OMEGA(0)
 weight[(ix-1)*(no_of_ext_p)+2]=1.0/pow(0.001,2);//i
 weight[(ix-1)*(no_of_ext_p)+3]=1.0/pow(10,2);  //a
 weight[(ix-1)*(no_of_ext_p)+4]=1.0/pow(0.09,2); //omega(0)
 weight[(ix-1)*(no_of_ext_p)+5]=1.0/pow(0.09,2); //phi(0)

 weight[(ix-1)*(no_of_ext_p)+6]=1.0/pow(0.09,2); //kapa(0)
if(no_of_ext_p == 8)
 weight[(ix-1)*(no_of_ext_p)+7]=1.0/pow(0.001,2);
if(no_of_ext_p == 15 || no_of_ext_p == 12 || no_of_ext_p == 9){
 weight[(ix-1)*(no_of_ext_p)+7]=1.0/pow(0.0001,2); //F(1)
 weight[(ix-1)*(no_of_ext_p)+8]=1.0/pow(0.0001,2); //C_OMEGA(1)
 }
}

if((no_of_ext_p)== 15 || (no_of_ext_p)== 12 ){
for(ix=9; ix<12;++ix){
 weight[ix]=1.0/pow(0.0001,2);
 weight[ix+(no_of_ext_p)]=1.0/pow(0.0001,2);}
}
if((no_of_ext_p)== 15){
for(ix=12; ix<(no_of_ext_p);++ix){
 weight[ix]=1.0/pow(0.0001,2);
 weight[ix+(no_of_ext_p)]=1.0/pow(0.0001,2);}
}

double sigma=pow(0.00001,2);

fout.open("weight.out");
for(ix=0; ix<(no_of_ext_p)*2;++ix){
 weight[ix]=weight[ix]*sigma;
 fout << weight[ix] << "\n"; }
fout.close();

//Computation of matrix normal
//    N=BTWB1+BTWB2+The diagonal weight matrix related to exterior orientation
parameters

for(ix=0; ix<ncb11; ++ix){
 for(int jx=0; jx<ncb11; ++jx) {
normal_11[ix][jx]=BTWB1[ix][jx]+BTWB2[ix][jx];}}

for(ix=0;ix<ncb11;++ix)
 normal_11[ix][ix]=normal_11[ix][ix]+weight[ix];

fout.open("norm11.out");
for( ix=0; ix<ncb11; ++ix) {
 for( int jx=0; jx<ncb11; ++jx) {
 double test=normal_11[ix][jx]-normal_11[jx][ix];
```

```
  if(test!=0.0){
   fout << ix << "\t" << jx << "\t" << test << "\n";
  }
 }}

 for( ix=0; ix<ncb11; ++ix) {
  for( int jx=0; jx<ncb11; ++jx) {
   fout << normal_11[ix][jx] << "\t";}
   fout << "\n"; }
 fout << "\n*****************End of normal_11 in iteration " << it <<
"******************************\n";
 fout.close();

//*****************************************************************
//*****************************************************************

//computation of matrices U1=mat_u1:
//mat_u1=BBART_11*W1_bar*EBAR1+BBART_21*W2_bar*EBAR2+
//          BBART_31*W3_bar*EBAR3.


for(ix=0; ix<nrb11; ++ix)
 for(int jx=0; jx<ncb11; ++jx) {
 pt_matrix_b[jx][ix]=BBAR_11[ix][jx];
}
amulti(pt_matrix_b,EBAR1,BTWE1,ncb11,nrb11);  //BBART_11*W1_bar*EBAR1;

for(ix=0; ix<nrb21; ++ix)
 for(int jx=0; jx<ncb21; ++jx)
 pt_matrix_b[jx][ix]=BBAR_21[ix][jx];

amulti(pt_matrix_b,EBAR2,BTWE2,ncb21,nrb21);  //BBART_21*W2_bar*EBAR2;

for(ix=0; ix<ncb11; ++ix){
  BTWE3[ix]=weight[ix]*EBARg[ix];
  BTWE3[ix]=-BTWE3[ix];}

//Computation of mterix mat_u1=BTWE1+BTWE2+BTWE3

for(ix=0; ix<ncb11; ++ix)
 mat_u1[ix]=BTWE1[ix]+BTWE2[ix]+BTWE3[ix];

//*****************************************************************
fout.open("mat_u.out");
for(ix=0;ix<(no_of_ext_p)*2;++ix)
 fout << mat_u1[ix] << "\n";
fout.close();

int row_n=(no_of_ext_p)*2;
```

```
choleski(normal_11,mat_u1,mat_x1,row_n);

fout.open("mat_x.out");
for(ix=0;ix<(no_of_ext_p)*2;++ix){
 mat_x1[ix]=-mat_x1[ix];
 fout << mat_x1[ix] << "\n";}
fout.close();

//*************************************************************
//*************************************************************

final << "\n\n\n        ************************************************"
      << "\n        *************************************************"
      << "\n        *************************************************\n"
      << "        ***                                          ***\n";

//Printing the results:

//print the iteration number:

final << "        ***  In iteration (" << it << ") the corrections are as follows: ***\n"
    << "        ***                                          ***";

//printing the satellite image number:

for(int ic=0; ic<=(no_of_ext_p); ic=ic+(no_of_ext_p)){
  final << "\n*************************************************************\n"
      << "\nCorrections of the exterior orientation parameters"
      << " \n          for the satellite image number" << ic/(no_of_ext_p)+1 << "\n\n";

//printing the corrections for the exterior orientation parameters:

    final << "DF0 = " << mat_x1[ic] << "\t"
         << "D[common-omega0] = " << mat_x1[ic+1]  << "\n"
         << "Di = " << mat_x1[ic+2]  << "\t"
         << "Da = " << mat_x1[ic+3]<< "\n\n\n"
         << "Domega(0) = " << mat_x1[ic+4]<< "\t"
         << "Dphi(0) = " << mat_x1[ic+5]  << "\t";
         if(no_of_ext_p == 7)
   final << "\nDphi(1) = " << mat_x1[ic+6]  << "\n";
         else
   final << "Dkappa(0) = " << mat_x1[ic+6] << "\n";
         if(no_of_ext_p == 8)
   final << "Dphi(1) = " << mat_x1[ic+7]  << "\n";
         if(no_of_ext_p == 15 || no_of_ext_p == 12 || no_of_ext_p == 9){
   final << "DF(1) = " << mat_x1[ic+7]<< "\t"
         << "D[common-omega](1) = " << mat_x1[ic+8]  << "\n";
      }
         if(no_of_ext_p == 15 || no_of_ext_p == 12){
```

```
    final << "Domega(1) = " << mat_x1[ic+9]<< "\t"
            << "Dphi(1) = " << mat_x1[ic+10]  << "\t"
            << "Dkappa(1) = " << mat_x1[ic+11] << "\n";
        }
            if((no_of_ext_p)==15){
    final << "Domega(2) = " << mat_x1[ic+12]<< "\t"
            << "Dphi(2) = " << mat_x1[ic+13]  << "\t"
            << "Dkappa(2) = " << mat_x1[ic+14] << "\n";
        }

     final << "\n-------------------------------\n";
  }

//up date the exterior orientation parameters:
if(no_of_ext_p==8){
   for(int je=0; je<2;++je){
     for(int ie=0;ie<3;++ie){
         imago1[je][ie]=imago1[je][ie]+mat_x1[ie+je*3];
         imago2[je][ie]=imago2[je][ie]+mat_x1[ie+je*3+(no_of_ext_p)];}}
         imago1[2][0]=imago1[2][0]+mat_x1[6];
         imago2[2][0]=imago2[2][0]+mat_x1[14];
         imago1[3][1]=imago1[3][1]+mat_x1[7];
         imago2[3][1]=imago2[3][1]+mat_x1[15];}
else if(no_of_ext_p==7){
   for(int je=0; je<2;++je){
     for(int ie=0;ie<3;++ie){
         imago1[je][ie]=imago1[je][ie]+mat_x1[ie+je*3];
         imago2[je][ie]=imago2[je][ie]+mat_x1[ie+je*3+(no_of_ext_p)];}}
         imago1[3][1]=imago1[3][1]+mat_x1[6];
         imago2[3][1]=imago2[3][1]+mat_x1[13];
}
else {
    for(int je=0; je<(no_of_ext_p)/3;++je){
     for(int ie=0;ie<3;++ie){
         imago1[je][ie]=imago1[je][ie]+mat_x1[ie+je*3];
         imago2[je][ie]=imago2[je][ie]+mat_x1[ie+je*3+(no_of_ext_p)];}}
}

}  //close the iteration acolad.

fout.open("inv.out");
double sigma=pow(0.00001,2);
int row_n=(no_of_ext_p)*2;
inverse(normal_11,pt_matrix_b,row_n);
for(int ix=0;ix<row_n;++ix){
  pt_matrix_b[ix][ix]=pt_matrix_b[ix][ix]*sigma;
  fout << pt_matrix_b[ix][ix] << "\n";
}
fout.close();
```

```
final << "\n                    ***********                    \n";
final << "\n          ******************************           \n";
final << "\n************************************************************\n";

final << "---------The final result for the first image----------\n\n\n";
final << "True anomaly = " << imago1[0][0] << "\n";
  if(no_of_ext_p == 15 || no_of_ext_p == 12 || no_of_ext_p == 9){
  final << "First rate of the true anomaly = " << imago1[2][1] << "\n";
  }
  final << "Right ascension of the ascending node = " << imago1[0][1] << "\n";
  if(no_of_ext_p == 15 || no_of_ext_p == 12 || no_of_ext_p == 9){
  final << "First rate of the right ascension of the ascending node = " << imago1[2][2] <<
"\n";
  }
final << "Inclination = " << imago1[0][2] << "\n"
    << "Semi major axis of the orbit = " << imago1[1][0] << "\n";

final << "Omega = " << imago1[1][1] << "\t"
    << "Phi = " << imago1[1][2] << "\t";
    if(no_of_ext_p == 7)
final << "\nFirst rate of the Phi = " << imago1[3][1] << "\n";
    else
final << "Kappa = " << imago1[2][0] << "\n";
    if(no_of_ext_p == 8)
final << "First rate of the Phi = " << imago1[3][1] << "\n";
    if(no_of_ext_p == 15 || no_of_ext_p == 12){
final << "First rate of the omega = " << imago1[3][0] << "\n"
    << "First rate of the Phi = " << imago1[3][1] << "\n"
    << "First rate of the kappa = " << imago1[3][2] << "\n";
    }
    if((no_of_ext_p)==15){
 final<< "Second rate of the omega = " << imago1[4][0] << "\n"
    << "Second rate of the Phi = " << imago1[4][1] << "\n"
    << "Second rate of the kappa = " << imago1[4][2] << "\n";
    }
final << "\n----------The final result for the second image----------\n\n";

final << "True anomaly = " << imago2[0][0] << "\n";
  if(no_of_ext_p == 15 || no_of_ext_p == 12 || no_of_ext_p == 9){
  final << "First rate of the true anomaly = " << imago2[2][1] << "\n";
  }
final << "Right ascension of the ascending node = " << imago2[0][1] << "\n";
  if(no_of_ext_p == 15 || no_of_ext_p == 12 || no_of_ext_p == 9){
  final << "First rate of the right ascension of the ascending node = " << imago2[2][2] <<
"\n";
  }
final << "Inclination = " << imago2[0][2] << "\n"
    << "Semi major axis of the orbit = " << imago2[1][0] << "\n";
```

```
final << "Omega = " << imago2[1][1] << "\t"
    << "Phi = " << imago2[1][2] << "\t";
    if(no_of_ext_p == 7)
final << "\nFirst rate of the Phi = " << imago1[3][1] << "\n";
    else
final << "Kappa = " << imago2[2][0] << "\n";
    if(no_of_ext_p == 8)
final << "First rate of the Phi = " << imago2[3][1] << "\n";

    if(no_of_ext_p == 15 || no_of_ext_p == 12){
final << "First rate of the omega = " << imago2[3][0] << "\n"
    << "First rate of the Phi = " << imago2[3][1] << "\n"
    << "First rate of the kappa = " << imago2[3][2] << "\n";
    }
    if((no_of_ext_p)==15){
 final<< "Second rate of the omega = " << imago2[4][0] << "\n"
    << "Second rate of the Phi = " << imago2[4][1] << "\n"
    << "Second rate of the kappa = " << imago2[4][2] << "\n";
    }

final << "*************************************************************\n";
final << "\n          ******************************          \n";
final << "\n                 ***********                 \n";
final.close();

fout.open("pobalat.out");
fout << "\n                 ***********                 \n";
fout << "\n          ******************************          \n";
fout << "\n*************************************************************\n";

fout << "---------The final result for the first image----------\n\n";
fout << "True anomaly = " << imago1[0][0] << "\n";
  if(no_of_ext_p == 15 || no_of_ext_p == 12 || no_of_ext_p == 9){
  fout << "First rate of the true anomaly = " << imago1[2][1] << "\n";
  }
  fout << "Right ascension of the ascending node = " << imago1[0][1] << "\n";
  if(no_of_ext_p == 15 || no_of_ext_p == 12 || no_of_ext_p == 9){
  fout << "First rate of the right ascension of the ascending node = " << imago1[2][2] <<
"\n";
  }
fout << "Inclination = " << imago1[0][2] << "\n"
    << "Semi major axis of the orbit = " << imago1[1][0] << "\n";

fout << "Omega = " << imago1[1][1] << "\t"
    << "Phi = " << imago1[1][2] << "\t";
    if(no_of_ext_p == 7)
fout << "\nFirst rate of the Phi = " << imago1[3][1] << "\n";
    else
fout << "Kappa = " << imago1[2][0] << "\n";
```

```
    if(no_of_ext_p == 8)
fout << "First rate of the Phi = " << imago1[3][1] << "\n";
    if(no_of_ext_p == 15 || no_of_ext_p == 12){
fout << "First rate of the omega = " << imago1[3][0] << "\n"
    << "First rate of the Phi = " << imago1[3][1] << "\n"
    << "First rate of the kappa = " << imago1[3][2] << "\n";
    }
    if((no_of_ext_p)==15){
 fout<< "Second rate of the omega = " << imago1[4][0] << "\n"
    << "Second rate of the Phi = " << imago1[4][1] << "\n"
    << "Second rate of the kappa = " << imago1[4][2] << "\n";
    }
fout << "\n----------The final result for the second image----------\n\n";

fout << "True anomaly = " << imago2[0][0] << "\n";
   if(no_of_ext_p == 15 || no_of_ext_p == 12 || no_of_ext_p == 9){
   fout << "First rate of the true anomaly = " << imago2[2][1] << "\n";
   }
fout << "Right ascension of the ascending node = " << imago2[0][1] << "\n";
   if(no_of_ext_p == 15 || no_of_ext_p == 12 || no_of_ext_p == 9){
   fout << "First rate of the right ascension of the ascending node = " << imago2[2][2] <<
"\n";
   }
fout << "Inclination = " << imago2[0][2] << "\n"
    << "Semi major axis of the orbit = " << imago2[1][0] << "\n";

fout << "Omega = " << imago2[1][1] << "\t"
    <<  "Phi = " << imago2[1][2] << "\t";
    if(no_of_ext_p == 7)
fout << "\nFirst rate of the Phi = " << imago1[3][1] << "\n";
    else
fout << "Kappa = " << imago2[2][0] << "\n";
    if(no_of_ext_p == 8)
fout << "First rate of the Phi = " << imago2[3][1] << "\n";

    if(no_of_ext_p == 15 || no_of_ext_p == 12){
fout << "First rate of the omega = " << imago2[3][0] << "\n"
    << "First rate of the Phi = " << imago2[3][1] << "\n"
    << "First rate of the kappa = " << imago2[3][2] << "\n";
    }
    if((no_of_ext_p)==15){
 fout<< "Second rate of the omega = " << imago2[4][0] << "\n"
    << "Second rate of the Phi = " << imago2[4][1] << "\n"
    << "Second rate of the kappa = " << imago2[4][2] << "\n";
    }

fout << "\n*****************************************************************\n";
fout << "\n          ********************************          \n";
fout << "\n                 ***********                  \n";
```

```
for(int ii=1; ii<=ngp+ncp; ++ii){  //open the acolad number 0.
 for(int id=1; id<=NPO; ++id){//open the acolad number 1.
  if(id==2){
    int np=np2+ncp;
    for(int i9=1; i9<=np; ++i9)
     for(int j9=1; j9<=4; ++j9)
          image_xy[i9-1][j9-1]=image2[i9-1][j9-1];

     for(i9=1; i9<=np; ++i9)
     for(int j9=1; j9<=2; ++j9)
          pixel_xy[i9-1][j9-1]=pixel2_xy[i9-1][j9-1];

            // in the SPOT case, we consider just the centre line.
  if(no_of_ext_p==8 || no_of_ext_p==7){
     for(int io=0; io<3; ++io) {
      for(int jo=0; jo<3; ++jo){
         image_eo[io][jo]=imago2[io][jo];}}
         image_eo[3][0]=imago2[3][1];
  }
  else{
     for(int io=0; io<(no_of_ext_p)/3; ++io) {
      for(int jo=0; jo<3; ++jo){
         image_eo[io][jo]=imago2[io][jo];}}
  }
         a_of_p=a_of_p2;  // argument of perigee
      e=e2;      // eccentricity
         view_angle = angle_r;
   if(image_case==3){
     size=0.010;
     pixelc=4060.5;
     pixely=2900.5;
   }
   else if(image_case==4){
     size=0.010;
     pixelc=4060.5;
     pixely=1488.5;
   }

   else{
     size=0.013;
     pixelc=3000;//pixel_cr;
     pixely=3000;//3762;
   }
   focal=focal_f/1000.0;
  }

  else if(id==1){
 int np=np1+ncp;
 for(int i=0;i<np;++i) {
```

```
  for(int j=0;j<4;++j){
   image_xy[i][j]=image1[i][j];}}

for(i=0;i<np;++i)
  for(int j=0;j<2;++j)
   pixel_xy[i][j]=pixel1_xy[i][j];
   if(image_case==3){
    size=0.010;
    pixelc=4060.5;
    pixely=2900.5;
   }
   else if(image_case==4){
    size=0.010;
    pixelc=4060.5;
    pixely=1488.5;
   }

   else{
    size=0.013;
    pixely=3000;//3865;
    pixelc=3000;//pixel_cl;
   }

if(no_of_ext_p==8 || no_of_ext_p==7){
   for(int io=0; io<3; ++io) {
    for(int jo=0; jo<3; ++jo){
       image_eo[io][jo]=imago1[io][jo];}}
       image_eo[3][0]=imago1[3][1];
}
else{
   for(int io=0; io<(no_of_ext_p)/3; ++io){
    for(int jo=0; jo<3; ++jo){
       image_eo[io][jo]=imago1[io][jo];}}
}

      a_of_p=a_of_p1;  // argument of perigee
      e=e1;        // eccentricity
      view_angle = angle_l;
      focal=focal_b/1000.0;}


      XP=0.0;  //x-x0
      YP=image_xy[ii-1][3];//-Ypp;  //y-y0

      F[ii-1][0]=image_eo[0][0]+image_eo[2][1]*((pixel_xy[ii-1][0]-pixelc)*size);

C_OMEGA[ii-1][0]=image_eo[0][1]+image_eo[2][2]*((pixel_xy[ii-1][0]-pixelc)*size);
      r[ii-1][0]=image_eo[1][0]*(1-pow(e,2))/(1+e*cos(F[ii-1][0]));
   if((no_of_ext_p)==15){
```

```
if(image_case==1 || image_case==3 || image_case==4){
  omega[ii-1][0]=image_eo[1][1]+image_eo[3][0]*((pixel_xy[ii-1][0]-pixelc)*size)+
            image_eo[4][0]*pow(((pixel_xy[ii-1][0]-pixelc)*size),2);
  phi[ii-1][0]=image_eo[1][2]+image_eo[3][1]*((pixel_xy[ii-1][0]-pixelc)*size)+
            image_eo[4][1]*pow(((pixel_xy[ii-1][0]-pixelc)*size),2);}
  else if(image_case==2){
  omega[ii-1][0]=image_eo[1][1]+image_eo[3][0]*((pixel_xy[ii-1][0]-pixelc)*size)+
            image_eo[4][0]*pow(((pixel_xy[ii-1][1]-pixely)*size),2);
  phi[ii-1][0]=image_eo[1][2]+image_eo[3][1]*((pixel_xy[ii-1][0]-pixelc)*size)+
            image_eo[4][1]*pow(((pixel_xy[ii-1][1]-pixely)*size),2);}

  kapa[ii-1][0]=image_eo[2][0]+image_eo[3][2]*((pixel_xy[ii-1][0]-pixelc)*size)+
            image_eo[4][2]*pow(((pixel_xy[ii-1][0]-pixelc)*size),2);
}
else if((no_of_ext_p)==12){
   omega[ii-1][0]=image_eo[1][1]+image_eo[3][0]*((pixel_xy[ii-1][0]-pixelc)*size);
   phi[ii-1][0]=image_eo[1][2]+image_eo[3][1]*((pixel_xy[ii-1][0]-pixelc)*size);
   kapa[ii-1][0]=image_eo[2][0]+image_eo[3][2]*((pixel_xy[ii-1][0]-pixelc)*size);
}
else if((no_of_ext_p)==9 ){
   omega[ii-1][0]=image_eo[1][1];
   phi[ii-1][0]=image_eo[1][2];
   kapa[ii-1][0]=image_eo[2][0];
}
 else if(no_of_ext_p == 8 ){
   omega[ii-1][0]=image_eo[1][1];
   phi[ii-1][0]=image_eo[1][2]+image_eo[3][0]*((pixel_xy[ii-1][0]-pixelc)*size);
   kapa[ii-1][0]=image_eo[2][0];
 }
 else if(no_of_ext_p == 7 ){
   omega[ii-1][0]=image_eo[1][1];
   phi[ii-1][0]=image_eo[1][2]+image_eo[3][0]*((pixel_xy[ii-1][0]-pixelc)*size);
   if(id==1)
   kapa[ii-1][0]=-0.004;
   if(id==2)
   kapa[ii-1][0]= 0.003;

 }

     be_bg(Be,phi[ii-1][0],omega[ii-1][0],kapa[ii-1][0],F[ii-1][0],
     C_OMEGA[ii-1][0],ground_xyz[ii-1][1],
     ground_xyz[ii-1][2],ground_xyz[ii-1][3],pixel_xy[ii-1][0],
     pixel_xy[ii-1][1],pixely,r[ii-1][0],a_of_p,image_eo[0][2],YP,1,size);

 if(id==1){
    xpix_l[ii-1]=(-Cx)*1e+06/13;
    ypix_l[ii-1]=(image1[ii-1][3]-Cy)*1e+06/13;
    XL=X_prj;
    YL=Y_prj;
```

```
        ZL=Z_prj;
        UL=UU;
        VL=VV;
        WL=WW;}
    else{
        xpix_r[ii-1]=(-Cx)*1e+06/13;
        ypix_r[ii-1]=(image2[ii-1][3]-Cy)*1e+06/13;
      XR=X_prj;
        YR=Y_prj;
        ZR=Z_prj;
        UR=UU;
        VR=VV;
        WR=WW;}
} // close acolad number 1;

landa_r[ii-1]=((XR-XL)*VL-(YR-YL)*UL)/(VR*UL-UR*VL);
XCP[ii-1]=XR+landa_r[ii-1]*UR;
YCP[ii-1]=YR+landa_r[ii-1]*VR;
ZCP[ii-1]=ZR+landa_r[ii-1]*WR;

DXX[ii-1]=XCP[ii-1]-ground_xyz[ii-1][1];
DYY[ii-1]=YCP[ii-1]-ground_xyz[ii-1][2];
DZZ[ii-1]=ZCP[ii-1]-ground_xyz[ii-1][3];

} // close acolad number 0;


double sum1=0.0;
double sum2=0.0;
double sum3=0.0;
double sum4=0.0;
fout << "\n***************************************************************\n";
fout << "\nThe residuals (in pixels) for the image coordinates of CPs are:\n\n";
fout << "           LEFT IMAGE " << "            " << "RIGHT IMAGE \n" ;
fout << "no.      Dx        Dy           Dx         Dy\n";

for(int i=0; i<ngp; ++i){
 fout << setiosflags(ios::right);
 fout << setw(3) << ground_xyz[i][0] << setprecision(3) << setw(10)
      << xpix_l[i] << setw(10)
      << ypix_l[i] << setw(14)
      << xpix_r[i] << setw(10)
      << ypix_r[i] << "\n";
      sum1=sum1+pow(xpix_l[i],2);
      sum2=sum2+pow(ypix_l[i],2);
      sum3=sum3+pow(xpix_r[i],2);
      sum4=sum4+pow(ypix_r[i],2);
 }
 fout << "\n***************************************************************\n";
```

```
fout << "The RMSE (in pixels) of x and y for the GCPs in left and right images are as
follows:\n\n";
fout << "RMSE(x_left)= " << sqrt(sum1/(ngp-1))<< "          "
    << "RMSE(y_left)= " << sqrt(sum2/(ngp-1))<< "\n"
    << "RMSE(x_right)= " << sqrt(sum3/(ngp-1))<< "          "
    << "RMSE(y_right)= " << sqrt(sum4/(ngp-1))<< "\n";
if(ncp>0.0){
sum1=0.0;
sum2=0.0;
sum3=0.0;
sum4=0.0;
fout << "****************************************************************\n";
fout << "The residuals (in pixels) for the image coordinates of the Check Points are:\n\n";
fout << "          LEFT IMAGE " << "          " << "RIGHT IMAGE \n" ;
fout << "no.     Dx       Dy         Dx       Dy\n";

for(int i=0; i<ncp; ++i){
 fout << setiosflags(ios::right);
 fout << setw(3) << ground_xyz[i+ngp][0] << setprecision(3) << setw(10)
     << xpix_l[i+ngp] << setw(10)
     << ypix_l[i+ngp] << setw(14)
     << xpix_r[i+ngp] << setw(10)
     << ypix_r[i+ngp] << "\n";
     sum1=sum1+pow(xpix_l[ngp+i],2);
     sum2=sum2+pow(ypix_l[ngp+i],2);
     sum3=sum3+pow(xpix_r[ngp+i],2);
     sum4=sum4+pow(ypix_r[ngp+i],2);
}
fout << "\n****************************************************************\n";
fout << "The RMSE (in pixels) of x and y for the GCPs in left and right images are as
follows:\n\n";
fout << "RMSE(x_left)= " << sqrt(sum1/(ncp-1))<< "          "
    << "RMSE(y_left)= " << sqrt(sum2/(ncp-1))<< "\n"
    << "RMSE(x_right)= " << sqrt(sum3/(ncp-1))<< "          "
    << "RMSE(y_right)= " << sqrt(sum4/(ncp-1))<< "\n";
}

sum1=0.0;
sum2=0.0;
sum3=0.0;

fout << "****************************************************************\n";
fout << "The residuals (in metres)for the ground coordinates of " << ngp <<
      " selected CPs are:\n\n";
fout << "no.          DX              DY              DZ\n";
for(i=0; i<ngp; ++i){
 fout << setw(3) << ground_xyz[i][0] << setprecision(6) << setw(16)
     << DXX[i] << setw(16)
     << DYY[i] << setw(16)
```

```
    << DZZ[i] << "\n";
    sum1=sum1+pow(DXX[i],2);
    sum2=sum2+pow(DYY[i],2);
    sum3=sum3+pow(DZZ[i],2);
}
fout << "\n*********************************************************\n";
fout << "The RMSE (in meteres) of X, Y and Z for the CPs are as follows:\n\n";
fout << "RMSE(X)= " << sqrt(sum1/(ngp-1))<< "       "
   << "RMSE(Y)= " << sqrt(sum2/(ngp-1))<< "       "
   << "RMSE(Z)= " << sqrt(sum3/(ngp-1))<< "\n";
if(ncp>0.0){
sum1=0.0;
sum2=0.0;
sum3=0.0;
fout << "*********************************************************\n";
fout << "The residuals (in metres) for the ground coordinates of " << ncp <<
       "\nselected Check Pts are:\n\n";
fout << "no.        DX            DY            DZ\n";
for(i=0; i<ncp; ++i){
 fout << setw(3) << ground_xyz[i+ngp][0] << setprecision(6) << setw(16)
     << DXX[i+ngp] << setw(16)
     << DYY[i+ngp] << setw(16)
     << DZZ[i+ngp] << "\n";

     sum1=sum1+pow(DXX[ngp+i],2);
     sum2=sum2+pow(DYY[ngp+i],2);
     sum3=sum3+pow(DZZ[ngp+i],2);
 }
 fout << "\n*********************************************************\n";
 fout << "The RMSE (in metres) of X, Y and Z for the check points are as follows:\n\n";
 fout << "RMSE(X)= " << sqrt(sum1/(ncp-1))<< "       "
     << "RMSE(Y)= " << sqrt(sum2/(ncp-1))<< "       "
     << "RMSE(Z)= " << sqrt(sum3/(ncp-1))<< "\n";

 }

 fout.close();

 delete pt_matrix_a;
 delete pt_matrix_b;
 delete pt_matrix_c;
 delete BBAR_11;
 delete Be;
 delete eop_obs;
 delete eop_it;
 delete image_xy;
 delete image_eo;
 delete pixel_xy;
 delete BBAR_21;
```

```
delete EBAR1;
delete EBAR2;
delete EBARg;
delete normal_11;
delete weight;
delete mat_u1;
delete mat_x1;
delete BTWB1;
delete BTWB2;
delete BTWE1;
delete BTWE2;
delete BTWE3;
delete omega;
delete phi;
delete kapa;
delete X0;
delete Y0;
delete Z0;
delete pt_vector_v;
delete pt_vector_x;
delete XCP;
delete YCP;
delete ZCP;
delete landa_r;
delete DXX;
delete DYY;
delete DZZ;
}


/*************************************************************************
            Derivations Sub-Module of Bundle Adjustment Module (Case 2)
*************************************************************************/

#include <math.h>
#include <fstream.h>
#include <iomanip.h>
#include <memory.h>
#include "pushbrom.h"

void be_bg(double huge(*Be)[15],double phi,double omega,double kapa,double F,
        double C_OMEGA,double ground_x,double ground_y,
        double ground_z,double pixel_xy,double pixel_y,int centre, double r,
        double a_omega,double inclin,double y_image,int intersect, float size)
{
double huge (*RS)[3]=new double huge [3][3];
double huge (*RA)[3]=new double huge [3][3];
double huge (*RAS)[3]=new double huge [3][3];
double huge (*R)[3]=new double huge [3][3];
```

```
ofstream fout;
fout.open("bebg.out");
        fout << F << "\t" << a_omega << "\t" << inclin << "\n";
        fout << phi << "\t" << omega << "\t" << kappa << "\n";
        fout << C_OMEGA << "\t" << r << "\n"
            << ground_x << "\t" << ground_y << "\t" << ground_z;
        fout << "\n" << pixel_xy << "\t" << pixelc << "\n";
        fout << view_angle << "\n" << focal << "\n" << e << "\n";
double DX;
double DY;
double DZ;
double m_param;
double n_param;
double q_param;
double CONST;


        //the element of matrix RS:
        RS[0][0]=-sin(F+a_omega)*cos(C_OMEGA)-
                    cos(F+a_omega)*cos(inclin)*sin(C_OMEGA);
        RS[0][1]=-sin(F+a_omega)*sin(C_OMEGA)+
                    cos(F+a_omega)*cos(inclin)*cos(C_OMEGA);
        RS[0][2]=cos(F+a_omega)*sin(inclin);
        RS[1][0]=sin(C_OMEGA)*sin(inclin);
        RS[1][1]=-cos(C_OMEGA)*sin(inclin);
        RS[1][2]=cos(inclin);
        RS[2][0]=cos(F+a_omega)*cos(C_OMEGA)-
                    sin(F+a_omega)*cos(inclin)*sin(C_OMEGA);
        RS[2][1]=cos(F+a_omega)*sin(C_OMEGA)+
                    sin(F+a_omega)*cos(inclin)*cos(C_OMEGA);

        RS[2][2]=sin(F+a_omega)*sin(inclin);

        //compute the differences between the GCPs and coordinates of the
        //projection centres of each line related to that GCP:
        X_prj=r*RS[2][0];
        Y_prj=r*RS[2][1];
        Z_prj=r*RS[2][2];

        //Forming the rotational matrix R composed of the rotational
        //elements related to each line:

        RA[0][0]=cos(phi)*cos(kapa);
        RA[0][1]=cos(omega)*sin(kapa)+
                    sin(omega)*sin(phi)*cos(kapa);
        RA[0][2]=sin(omega)*sin(kapa)-
                    cos(omega)*sin(phi)*cos(kapa);
        RA[1][0]=(-cos(phi)*sin(kapa));
        RA[1][1]=cos(omega)*cos(kapa)-
```

```
            sin(omega)*sin(phi)*sin(kapa);
    RA[1][2]=sin(omega)*cos(kapa)+
              cos(omega)*sin(phi)*sin(kapa);
    RA[2][0]=sin(phi);
    RA[2][1]=(-sin(omega)*cos(phi));
    RA[2][2]= cos(omega)*cos(phi);


 RAS[0][0]= RA[0][0]*RS[0][0]+RA[0][1]*RS[1][0]+RA[0][2]*RS[2][0];
 RAS[0][1]= RA[0][0]*RS[0][1]+RA[0][1]*RS[1][1]+RA[0][2]*RS[2][1];
 RAS[0][2]= RA[0][0]*RS[0][2]+RA[0][1]*RS[1][2]+RA[0][2]*RS[2][2];
 RAS[1][0]= RA[1][0]*RS[0][0]+RA[1][1]*RS[1][0]+RA[1][2]*RS[2][0];
 RAS[1][1]= RA[1][0]*RS[0][1]+RA[1][1]*RS[1][1]+RA[1][2]*RS[2][1];
 RAS[1][2]= RA[1][0]*RS[0][2]+RA[1][1]*RS[1][2]+RA[1][2]*RS[2][2];
 RAS[2][0]= RA[2][0]*RS[0][0]+RA[2][1]*RS[1][0]+RA[2][2]*RS[2][0];
 RAS[2][1]= RA[2][0]*RS[0][1]+RA[2][1]*RS[1][1]+RA[2][2]*RS[2][1];
 RAS[2][2]= RA[2][0]*RS[0][2]+RA[2][1]*RS[1][2]+RA[2][2]*RS[2][2];

 // initialising the view angle in the case of SPOT (Cross-Track Imagery)

 R[0][0]= RAS[0][0];
 R[0][1]= RAS[0][1];
 R[0][2]= RAS[0][2];
 R[1][0]= RAS[1][0]* cos(view_angle) + RAS[2][0]* sin(view_angle);
 R[1][1]= RAS[1][1]* cos(view_angle) + RAS[2][1]* sin(view_angle);
 R[1][2]= RAS[1][2]* cos(view_angle) + RAS[2][2]* sin(view_angle);
 R[2][0]= RAS[2][0]* cos(view_angle) - RAS[1][0]* sin(view_angle);
 R[2][1]= RAS[2][1]* cos(view_angle) - RAS[1][1]* sin(view_angle);
 R[2][2]= RAS[2][2]* cos(view_angle) - RAS[1][2]* sin(view_angle);

if(intersect==1){
    UU=R[1][0]*y_image+R[2][0]*(-focal);
    VV=R[1][1]*y_image+R[2][1]*(-focal);
    WW=R[1][2]*y_image+R[2][2]*(-focal);
}


      DX=ground_x-X_prj;
      DY=ground_y-Y_prj;
      DZ=ground_z-Z_prj;

   //computing the parameters:m_param, n_param, q_param:

   m_param=R[0][0]*DX+R[0][1]*DY+R[0][2]*DZ;
   n_param=R[1][0]*DX+R[1][1]*DY+R[1][2]*DZ;
   q_param=R[2][0]*DX+R[2][1]*DY+R[2][2]*DZ;

  fout << "\nm="<<m_param<<"\t" << "n="<<n_param<<"\t"  << "q="<<q_param<<"\n";
```

```
//computing the Cx=x-x0 and Cy=y-y0 by calculation. This will
//be later compared with those values that have been got by
//observations.

Cx=-focal*m_param/q_param;
double xpixel=Cx*1e+06/13;
Cy=-focal*n_param/q_param;
double ypixel=Cy*1e+06/13;

fout << "DX=" << DX << "\t" << "DY=" << DY << "\t" << "DZ=" << DZ << "\n";
fout << "cx=" << Cx << "\t" << "cy=" << Cy << "\n";
fout << "pixel_x= " <<xpixel << "\t" << "pixel_y= " << ypixel << "\n";

CONST=focal/pow(q_param,2);
fout << "\CONST = " << CONST << "\n";

//computing the elements of matrix Be:

if(intersect==0){
  Be[0][0]=CONST*
  (q_param*((-RA[0][0]*RS[2][0]+RA[0][2]*RS[0][0])*DX+
          (-RA[0][0]*RS[2][1]+RA[0][2]*RS[0][1])*DY+
          (-RA[0][0]*RS[2][2]+RA[0][2]*RS[0][2])*DZ-
           r*e*sin(F)/(1+e*cos(F))*(R[0][0]*RS[2][0]+
             R[0][1]*RS[2][1]+R[0][2]*RS[2][2])-
             r*(R[0][0]*RS[0][0]+R[0][1]*RS[0][1]+R[0][2]*RS[0][2]))-
     m_param*(((RA[1][0]*RS[2][0]-RA[1][2]*RS[0][0])*sin(view_angle)+
             (-RA[2][0]*RS[2][0]+RA[2][2]*RS[0][0])*cos(view_angle))*DX+
             ((RA[1][0]*RS[2][1]-RA[1][2]*RS[0][1])*sin(view_angle)+
             (-RA[2][0]*RS[2][1]+RA[2][2]*RS[0][1])*cos(view_angle))*DY+
             ((RA[1][0]*RS[2][2]-RA[1][2]*RS[0][2])*sin(view_angle)+
             (-RA[2][0]*RS[2][2]+RA[2][2]*RS[0][2])*cos(view_angle))*DZ-

r*e*sin(F)/(1+e*cos(F))*(R[2][0]*RS[2][0]+R[2][1]*RS[2][1]+R[2][2]*RS[2][2])-
              r*(R[2][0]*RS[0][0]+R[2][1]*RS[0][1]+R[2][2]*RS[0][2])));


Be[0][1]=CONST*(q_param*(-R[0][1]*DX+R[0][0]*DY+r*(RS[2][1]*R[0][0]-RS[2][
0]*R[0][1]))-

m_param*(-R[2][1]*DX+R[2][0]*DY+r*(RS[2][1]*R[2][0]-RS[2][0]*R[2][1])));

   double pqr=q_param*(
              (cos(F+a_omega)*sin(inclin)*sin(C_OMEGA)*RA[0][0]+
                     cos(inclin)*sin(C_OMEGA)*RA[0][1]+
                     sin(inclin)*sin(C_OMEGA)*sin(F+a_omega)*RA[0][2])*DX+
                    (-cos(F+a_omega)*sin(inclin)*cos(C_OMEGA)*RA[0][0]+
                     (-cos(inclin)*cos(C_OMEGA))*RA[0][1]+
                     (-sin(inclin)*cos(C_OMEGA)*sin(F+a_omega))*RA[0][2])*DY+
```

```
q_param=R[2][0]*DX+R[2][1]*DY+R[2][2]*DZ;


//computing the Cx=x-x0 and Cy=y-y0 by calculation. This will
//be later compared with these values that have been got by
//observations.

Cx=-focal*m_param/q_param;
Cy=-focal*n_param/q_param;

CONST=focal/pow(q_param,2);



//computing the elements of matrix Be

Be[0][0]=CONST*
    (q_param*((-RA[0][0]*RS[2][0]+RA[0][2]*RS[0][0])*DX+
             (-RA[0][0]*RS[2][1]+RA[0][2]*RS[0][1])*DY+
             (-RA[0][0]*RS[2][2]+RA[0][2]*RS[0][2])*DZ-
             r*e*sin(F)/(1+e*cos(F))*(R[0][0]*RS[2][0]+
               R[0][1]*RS[2][1]+R[0][2]*RS[2][2])-
               r*(R[0][0]*RS[0][0]+R[0][1]*RS[0][1]+R[0][2]*RS[0][2]))-
      m_param*(((RA[1][0]*RS[2][0]-RA[1][2]*RS[0][0])*sin(view_angle)+
               (-RA[2][0]*RS[2][0]+RA[2][2]*RS[0][0])*cos(view_angle))*DX+
               ((RA[1][0]*RS[2][1]-RA[1][2]*RS[0][1])*sin(view_angle)+
               (-RA[2][0]*RS[2][1]+RA[2][2]*RS[0][1])*cos(view_angle))*DY+
               ((RA[1][0]*RS[2][2]-RA[1][2]*RS[0][2])*sin(view_angle)+
               (-RA[2][0]*RS[2][2]+RA[2][2]*RS[0][2])*cos(view_angle))*DZ-

r*e*sin(F)/(1+e*cos(F))*(R[2][0]*RS[2][0]+R[2][1]*RS[2][1]+R[2][2]*RS[2][2])-
               r*(R[2][0]*RS[0][0]+R[2][1]*RS[0][1]+R[2][2]*RS[0][2])));


Be[0][1]=CONST*(q_param*(-R[0][1]*DX+R[0][0]*DY+r*(RS[2][1]*R[0][0]-RS[2][
0]*R[0][1]))-

m_param*(-R[2][1]*DX+R[2][0]*DY+r*(RS[2][1]*R[2][0]-RS[2][0]*R[2][1])));

    double pqr=q_param*(
               (cos(F+a_omega)*sin(inclin)*sin(C_OMEGA)*RA[0][0]+
                   cos(inclin)*sin(C_OMEGA)*RA[0][1]+
                   sin(inclin)*sin(C_OMEGA)*sin(F+a_omega)*RA[0][2])*DX+
                  (-cos(F+a_omega)*sin(inclin)*cos(C_OMEGA)*RA[0][0]+
                   (-cos(inclin)*cos(C_OMEGA))*RA[0][1]+
                   (-sin(inclin)*cos(C_OMEGA)*sin(F+a_omega))*RA[0][2])*DY+
                  (cos(F+a_omega)*cos(inclin)*RA[0][0]+
```

```
                                   (cos(F+a_omega)*cos(inclin)*RA[0][0]+

(-sin(inclin))*RA[0][1]+cos(inclin)*sin(F+a_omega)*RA[0][2])*DZ+

r*sin(F+a_omega)*(-sin(C_OMEGA)*sin(inclin)+cos(C_OMEGA)*sin(inclin)-cos(inclin)));
    double mno=-m_param*(
                                   ((-cos(F+a_omega)*sin(inclin)*sin(C_OMEGA)*RA[1][0]-
                                      cos(inclin)*sin(C_OMEGA)*RA[1][1]-

sin(inclin)*sin(C_OMEGA)*sin(F+a_omega)*RA[1][2])*sin(view_angle)+
                                    (cos(F+a_omega)*sin(inclin)*sin(C_OMEGA)*RA[2][0]+
                                     cos(inclin)*sin(C_OMEGA)*RA[2][1]+

sin(inclin)*sin(C_OMEGA)*sin(F+a_omega)*RA[2][2])*cos(view_angle))*DX+
                                   ((cos(F+a_omega)*sin(inclin)*cos(C_OMEGA)*RA[1][0]+
                                      cos(inclin)*cos(C_OMEGA)*RA[1][1]+

sin(inclin)*cos(C_OMEGA)*sin(F+a_omega)*RA[1][2])*sin(view_angle)+
                                     (-cos(F+a_omega)*sin(inclin)*cos(C_OMEGA)*RA[2][0]-
                                      cos(inclin)*cos(C_OMEGA)*RA[2][1]-

sin(inclin)*cos(C_OMEGA)*sin(F+a_omega)*RA[2][2])*cos(view_angle))*DY+
                                   ((-cos(F+a_omega)*cos(inclin)*RA[1][0]+
                                      sin(inclin)*RA[1][1]-
                                      cos(inclin)*sin(F+a_omega)*RA[1][2])*sin(view_angle)+
                                    (cos(F+a_omega)*cos(inclin)*RA[2][0]-
                                      sin(inclin)*RA[2][1]+

cos(inclin)*sin(F+a_omega)*RA[2][2])*cos(view_angle))*DZ-
                                   r*sin(F+a_omega)*(sin(C_OMEGA)*sin(inclin)*R[2][0]-

cos(C_OMEGA)*sin(inclin)*R[2][1]+cos(inclin)*R[2][2]));
      Be[0][2]=CONST*(pqr+mno);
      Be[0][3]=CONST*(pow(e,2)-1)/(1+e*cos(F))*
            (q_param*(RS[2][0]*R[0][0]+RS[2][1]*R[0][1]+RS[2][2]*R[0][2])-

m_param*(RS[2][0]*R[2][0]+RS[2][1]*R[2][1]+RS[2][2]*R[2][2]));

      Be[0][4]=CONST*(
            q_param*(
                          (-RS[1][0]*RA[0][2]+RS[2][0]*RA[0][1])*DX+
                          (-RS[1][1]*RA[0][2]+RS[2][1]*RA[0][1])*DY+
                          (-RS[1][2]*RA[0][2]+RS[2][2]*RA[0][1])*DZ)-
              m_param*(
                          ((RS[1][0]*RA[1][2]-RS[2][0]*RA[1][1])*sin(view_angle)+
                          (-RS[1][0]*RA[2][2]+RS[2][0]*RA[2][1])*cos(view_angle))*DX+
                          ((RS[1][1]*RA[1][2]-RS[2][1]*RA[1][1])*sin(view_angle)+
                          (-RS[1][1]*RA[2][2]+RS[2][1]*RA[2][1])*cos(view_angle))*DY+
                          ((RS[1][2]*RA[1][2]-RS[2][2]*RA[1][1])*sin(view_angle)+
```

```
(-RS[1][2]*RA[2][2]+RS[2][2]*RA[2][1])*cos(view_angle))*DZ));
    Be[0][5]=CONST*(
              q_param*(
               (RS[0][0]*(-sin(phi)*cos(kapa))+
                RS[1][0]*(sin(omega)*cos(phi)*cos(kapa))+
                RS[2][0]*(-cos(omega)*cos(phi)*cos(kapa)))*DX+
               (RS[0][1]*(-sin(phi)*cos(kapa))+
                RS[1][1]*(sin(omega)*cos(phi)*cos(kapa))+
                RS[2][1]*(-cos(omega)*cos(phi)*cos(kapa)))*DY+
               (RS[0][2]*(-sin(phi)*cos(kapa))+
                RS[1][2]*(sin(omega)*cos(phi)*cos(kapa))+
                RS[2][2]*(-cos(omega)*cos(phi)*cos(kapa)))*DZ)-
              m_param*(
               ((-RS[0][0]*(sin(phi)*sin(kapa))-
                 RS[1][0]*(-sin(omega)*cos(phi)*sin(kapa))-
                 RS[2][0]*(cos(omega)*cos(phi)*sin(kapa)))*sin(view_angle)+
                (RS[0][0]*(cos(phi))+
                 RS[1][0]*(sin(omega)*sin(phi))+
                 RS[2][0]*(-cos(omega)*sin(phi)))*cos(view_angle))*DX+
               ((-RS[0][1]*(sin(phi)*sin(kapa))-
                 RS[1][1]*(-sin(omega)*cos(phi)*sin(kapa))-
                 RS[2][1]*(cos(omega)*cos(phi)*sin(kapa)))*sin(view_angle)+
                (RS[0][1]*(cos(phi))+
                 RS[1][1]*(sin(omega)*sin(phi))+
                 RS[2][1]*(-cos(omega)*sin(phi)))*cos(view_angle))*DY+
               ((-RS[0][2]*(sin(phi)*sin(kapa))-
                 RS[1][2]*(-sin(omega)*cos(phi)*sin(kapa))-
                 RS[2][2]*(cos(omega)*cos(phi)*sin(kapa)))*sin(view_angle)+
                (RS[0][2]*(cos(phi))+
                 RS[1][2]*(sin(omega)*sin(phi))+
                 RS[2][2]*(-cos(omega)*sin(phi)))*cos(view_angle))*DZ));
     Be[0][6] =CONST*(
                   q_param*(RAS[1][0]*DX+RAS[1][1]*DY+RAS[1][2]*DZ)-
                   m_param*(m_param*sin(view_angle)));
    if(no_of_ext_p==7)
     Be[0][6] =((pixel_xy-pixelc)*size)*Be[0][5];
    if(no_of_ext_p==8)
     Be[0][7] =((pixel_xy-pixelc)*size)*Be[0][5];
    if(no_of_ext_p == 15 || no_of_ext_p == 12 || no_of_ext_p == 9){
     Be[0][7] =((pixel_xy-pixelc)*size)*Be[0][0];
     Be[0][8] =((pixel_xy-pixelc)*size)*Be[0][1];
    }
    if(no_of_ext_p == 15 || no_of_ext_p == 12){
     Be[0][9] =((pixel_xy-pixelc)*size)*Be[0][4];
     Be[0][10]=((pixel_xy-pixelc)*size)*Be[0][5];
     Be[0][11]=((pixel_xy-pixelc)*size)*Be[0][6];
    }
    if(no_of_ext_p == 15){
```

```
   if(image_case==1 || image_case==3 || image_case==4){
   Be[0][12]=pow(((pixel_xy-pixelc)*size),2)*Be[0][4];
   Be[0][13]=pow(((pixel_xy-pixelc)*size),2)*Be[0][5];
   }
   else if(image_case==2){
   Be[0][12]=pow(((pixel_y-centre)*size),2)*Be[0][4];
   Be[0][13]=pow(((pixel_y-centre)*size),2)*Be[0][5];
   }
   Be[0][14]=pow(((pixel_xy-pixelc)*size),2)*Be[0][6];
  }

   Be[1][0]=CONST*
   (q_param*(((-RA[1][0]*RS[2][0]+RA[1][2]*RS[0][0])*cos(view_angle)+
                  (-RA[2][0]*RS[2][0]+RA[2][2]*RS[0][0])*sin(view_angle))*DX+
                 ((-RA[1][0]*RS[2][1]+RA[1][2]*RS[0][1])*cos(view_angle)+
                  (-RA[2][0]*RS[2][1]+RA[2][2]*RS[0][1])*sin(view_angle))*DY+
                 ((-RA[1][0]*RS[2][2]+RA[1][2]*RS[0][2])*cos(view_angle)+
                  (-RA[2][0]*RS[2][2]+RA[2][2]*RS[0][2])*sin(view_angle))*DZ-
               r*e*sin(F)/(1+e*cos(F))*(R[1][0]*RS[2][0]+
                  R[1][1]*RS[2][1]+R[1][2]*RS[2][2])-
                  r*(R[1][0]*RS[0][0]+R[1][1]*RS[0][1]+R[1][2]*RS[0][2]))-
       n_param*(((RA[1][0]*RS[2][0]-RA[1][2]*RS[0][0])*sin(view_angle)+
                  (-RA[2][0]*RS[2][0]+RA[2][2]*RS[0][0])*cos(view_angle))*DX+
                 ((RA[1][0]*RS[2][1]-RA[1][2]*RS[0][1])*sin(view_angle)+
                  (-RA[2][0]*RS[2][1]+RA[2][2]*RS[0][1])*cos(view_angle))*DY+
                 ((RA[1][0]*RS[2][2]-RA[1][2]*RS[0][2])*sin(view_angle)+
                  (-RA[2][0]*RS[2][2]+RA[2][2]*RS[0][2])*cos(view_angle))*DZ-

r*e*sin(F)/(1+e*cos(F))*(R[2][0]*RS[2][0]+R[2][1]*RS[2][1]+R[2][2]*RS[2][2])-
                  r*(R[2][0]*RS[0][0]+R[2][1]*RS[0][1]+R[2][2]*RS[0][2])));


Be[1][1]=CONST*(q_param*(-R[1][1]*DX+R[1][0]*DY+r*(RS[2][1]*R[1][0]-RS[2][
0]*R[1][1]))-

n_param*(-R[2][1]*DX+R[2][0]*DY+r*(RS[2][1]*R[2][0]-RS[2][0]*R[2][1])));

    pqr=q_param*(
                  ((cos(F+a_omega)*sin(inclin)*sin(C_OMEGA)*RA[1][0]+
                   cos(inclin)*sin(C_OMEGA)*RA[1][1]+

sin(inclin)*sin(C_OMEGA)*sin(F+a_omega)*RA[1][2])*cos(view_angle)+
                   (cos(F+a_omega)*sin(inclin)*sin(C_OMEGA)*RA[2][0]+
                   cos(inclin)*sin(C_OMEGA)*RA[2][1]+

sin(inclin)*sin(C_OMEGA)*sin(F+a_omega)*RA[2][2])*sin(view_angle))*DX+
                  ((-cos(F+a_omega)*sin(inclin)*cos(C_OMEGA)*RA[1][0]-
                   cos(inclin)*cos(C_OMEGA)*RA[1][1]-
```

```
sin(inclin)*cos(C_OMEGA)*sin(F+a_omega)*RA[1][2])*cos(view_angle)+
                         (-cos(F+a_omega)*sin(inclin)*cos(C_OMEGA)*RA[2][0]-
                          cos(inclin)*cos(C_OMEGA)*RA[2][1]-

sin(inclin)*cos(C_OMEGA)*sin(F+a_omega)*RA[2][2])*sin(view_angle))*DY+
                         ((cos(F+a_omega)*cos(inclin)*RA[1][0]-
                          sin(inclin)*RA[1][1]+
                          cos(inclin)*sin(F+a_omega)*RA[1][2])*cos(view_angle)+
                         (cos(F+a_omega)*cos(inclin)*RA[2][0]-
                          sin(inclin)*RA[2][1]+
                          cos(inclin)*sin(F+a_omega)*RA[2][2])*sin(view_angle))*DZ-

r*sin(F+a_omega)*(sin(C_OMEGA)*sin(inclin)*R[1][0]-cos(C_OMEGA)*sin(inclin)*R
[1][1]+cos(inclin)*R[1][2]));
    mno=n_param*(
                         ((-cos(F+a_omega)*sin(inclin)*sin(C_OMEGA)*RA[1][0]-
                          cos(inclin)*sin(C_OMEGA)*RA[1][1]-

sin(inclin)*sin(C_OMEGA)*sin(F+a_omega)*RA[1][2])*sin(view_angle)+
                         (cos(F+a_omega)*sin(inclin)*sin(C_OMEGA)*RA[2][0]+
                          cos(inclin)*sin(C_OMEGA)*RA[2][1]+

sin(inclin)*sin(C_OMEGA)*sin(F+a_omega)*RA[2][2])*cos(view_angle))*DX+
                         ((cos(F+a_omega)*sin(inclin)*cos(C_OMEGA)*RA[1][0]+
                          cos(inclin)*cos(C_OMEGA)*RA[1][1]+

sin(inclin)*cos(C_OMEGA)*sin(F+a_omega)*RA[1][2])*sin(view_angle)+
                         (-cos(F+a_omega)*sin(inclin)*cos(C_OMEGA)*RA[2][0]-
                          cos(inclin)*cos(C_OMEGA)*RA[2][1]-

sin(inclin)*cos(C_OMEGA)*sin(F+a_omega)*RA[2][2])*cos(view_angle))*DY+
                         ((-cos(F+a_omega)*cos(inclin)*RA[1][0]+
                          sin(inclin)*RA[1][1]-
                          cos(inclin)*sin(F+a_omega)*RA[1][2])*sin(view_angle)+
                         (cos(F+a_omega)*cos(inclin)*RA[2][0]-
                          sin(inclin)*RA[2][1]+
                          cos(inclin)*sin(F+a_omega)*RA[2][2])*cos(view_angle))*DZ-

r*sin(F+a_omega)*(sin(C_OMEGA)*sin(inclin)*R[2][0]-cos(C_OMEGA)*sin(inclin)*R
[2][1]+cos(inclin)*R[2][2]));
    Be[1][2]=CONST*(pqr-mno);
    Be[1][3]=CONST*(pow(e,2)-1)/(1+e*cos(F))*
           (q_param*(RS[2][0]*R[1][0]+RS[2][1]*R[1][1]+RS[2][2]*R[1][2])-
                n_param*(RS[2][0]*R[2][0]+RS[2][1]*R[2][1]+RS[2][2]*R[2][2]));

    Be[1][4]=CONST*(
           q_param*(
                    ((-RS[1][0]*RA[1][2]+RS[2][0]*RA[1][1])*cos(view_angle)+
                    (-RS[1][0]*RA[2][2]+RS[2][0]*RA[2][1])*sin(view_angle))*DX+
```

```
            ((-RS[1][1]*RA[1][2]+RS[2][1]*RA[1][1])*cos(view_angle)+
             (-RS[1][1]*RA[2][2]+RS[2][1]*RA[2][1])*sin(view_angle))*DY+
            ((-RS[1][2]*RA[1][2]+RS[2][2]*RA[1][1])*cos(view_angle)+
             (-RS[1][2]*RA[2][2]+RS[2][2]*RA[2][1])*sin(view_angle))*DZ)-
        n_param*(
            ((RS[1][0]*RA[1][2]-RS[2][0]*RA[1][1])*sin(view_angle)+
             (-RS[1][0]*RA[2][2]+RS[2][0]*RA[2][1])*cos(view_angle))*DX+
            ((RS[1][1]*RA[1][2]-RS[2][1]*RA[1][1])*sin(view_angle)+
             (-RS[1][1]*RA[2][2]+RS[2][1]*RA[2][1])*cos(view_angle))*DY+
            ((RS[1][2]*RA[1][2]-RS[2][2]*RA[1][1])*sin(view_angle)+

(-RS[1][2]*RA[2][2]+RS[2][2]*RA[2][1])*cos(view_angle))*DZ));
   Be[1][5]=CONST*(
        q_param*(
          ((RS[0][0]*(sin(phi)*sin(kapa))+
            RS[1][0]*(-sin(omega)*cos(phi)*sin(kapa))+
            RS[2][0]*(cos(omega)*cos(phi)*sin(kapa)))*cos(view_angle)+
           (RS[0][0]*(cos(phi))+
            RS[1][0]*(sin(omega)*sin(phi))+
            RS[2][0]*(-cos(omega)*sin(phi)))*sin(view_angle))*DX+
          ((RS[0][1]*(sin(phi)*sin(kapa))+
            RS[1][1]*(-sin(omega)*cos(phi)*sin(kapa))+
            RS[2][1]*(cos(omega)*cos(phi)*sin(kapa)))*cos(view_angle)+
           (RS[0][1]*(cos(phi))+
            RS[1][1]*(sin(omega)*sin(phi))+
            RS[2][1]*(-cos(omega)*sin(phi)))*sin(view_angle))*DY+
          ((RS[0][2]*(sin(phi)*sin(kapa))+
            RS[1][2]*(-sin(omega)*cos(phi)*sin(kapa))+
            RS[2][2]*(cos(omega)*cos(phi)*sin(kapa)))*cos(view_angle)+
           (RS[0][2]*(cos(phi))+
            RS[1][2]*(sin(omega)*sin(phi))+
            RS[2][2]*(-cos(omega)*sin(phi)))*sin(view_angle))*DZ)-
        n_param*(
          ((-RS[0][0]*(sin(phi)*sin(kapa))-
            RS[1][0]*(-sin(omega)*cos(phi)*sin(kapa))-
            RS[2][0]*(cos(omega)*cos(phi)*sin(kapa)))*sin(view_angle)+
           (RS[0][0]*(cos(phi))+
            RS[1][0]*(sin(omega)*sin(phi))+
            RS[2][0]*(-cos(omega)*sin(phi)))*cos(view_angle))*DX+
          ((-RS[0][1]*(sin(phi)*sin(kapa))-
            RS[1][1]*(-sin(omega)*cos(phi)*sin(kapa))-
            RS[2][1]*(cos(omega)*cos(phi)*sin(kapa)))*sin(view_angle)+
           (RS[0][1]*(cos(phi))+
            RS[1][1]*(sin(omega)*sin(phi))+
            RS[2][1]*(-cos(omega)*sin(phi)))*cos(view_angle))*DY+
          ((-RS[0][2]*(sin(phi)*sin(kapa))-
            RS[1][2]*(-sin(omega)*cos(phi)*sin(kapa))-
            RS[2][2]*(cos(omega)*cos(phi)*sin(kapa)))*sin(view_angle)+
           (RS[0][2]*(cos(phi))+
```

```
                    RS[1][2]*(sin(omega)*sin(phi))+
                    RS[2][2]*(-cos(omega)*sin(phi)))*cos(view_angle))*DZ));
  Be[1][6] =CONST*(
                       q_param*(-m_param*cos(view_angle))-
                       n_param*(m_param*sin(view_angle)));
 if(no_of_ext_p==7)
  Be[1][6] =((pixel_xy-pixelc)*size)*Be[1][5];

 if(no_of_ext_p==8)
  Be[1][7] =((pixel_xy-pixelc)*size)*Be[1][5];

 if(no_of_ext_p == 15 || no_of_ext_p == 12 || no_of_ext_p == 9){
  Be[1][7] =((pixel_xy-pixelc)*size)*Be[1][0];
  Be[1][8] =((pixel_xy-pixelc)*size)*Be[1][1];
 }
 if(no_of_ext_p == 15 || no_of_ext_p == 12){
  Be[1][9] =((pixel_xy-pixelc)*size)*Be[1][4];
  Be[1][10]=((pixel_xy-pixelc)*size)*Be[1][5];
  Be[1][11]=((pixel_xy-pixelc)*size)*Be[1][6];
 }
 if(no_of_ext_p == 15){
  if(image_case==1 || image_case==3 || image_case==4){
  Be[1][12]=pow(((pixel_xy-pixelc)*size),2)*Be[1][4];
  Be[1][13]=pow(((pixel_xy-pixelc)*size),2)*Be[1][5];
  }
  else if(image_case==2){
  Be[1][12]=pow(((pixel_y-centre)*size),2)*Be[1][4];
  Be[1][13]=pow(((pixel_y-centre)*size),2)*Be[1][5];
  }
  Be[1][14]=pow(((pixel_xy-pixelc)*size),2)*Be[1][6];
 }

} // this is related to if.

fout.close();
delete R;
delete RA;
delete RS;
delete RAS;
}

/*************************************************************************
        Header Data Manipulation Sub-Module of Bundle Adjustment Module
*************************************************************************/

#include <math.h>
#include <fstream.h>
#include <iomanip.h>
#include "pushbrom.h"
```

```
double huge (*imago)[3]=new double huge [30][3];
void lagrange(double huge(*imago)[3])
{

ofstream fout("XC.out");
double XC[3];
double VC[3];
double t[8];
double tc;
double X[8][3];
double V[8][3];

 for(int i=0; i<8; ++i){
   X[i][0]=imago[i][0];
   X[i][1]=imago[i][1];
   X[i][2]=imago[i][2];
   V[i][0]=imago[i+8][0];
   V[i][1]=imago[i+8][1];
   V[i][2]=imago[i+8][2];
 }

 for(i=1; i<=3; ++i){
   t[(i-1)*3]=imago[i+15][0];
   t[(i-1)*3 +1]=imago[i+15][1];
   t[(i-1)*3+2]=imago[i+15][2];}
   tc=t[8];

  for(int ii=0;ii<3;++ii){
    XC[ii]=0.0;
    VC[ii]=0.0;
  }

  for(ii=0;ii<3;++ii){
   for(i=0;i<8;++i){
     double multiple=1.0;
     for(int j=0;j<8;++j){
       if(i!=j)
         multiple=multiple*(tc-t[j])/(t[i]-t[j]);}
            XC[ii]=XC[ii]+X[i][ii]*multiple;
            VC[ii]=VC[ii]+V[i][ii]*multiple;
  }}

 if(image_case==3 || image_case==4){
   XC[0]=XC[0]*0.3048;
   XC[1]=XC[1]*0.3048;
   XC[2]=XC[2]*0.3048;
   VC[0]=VC[0]*0.3048;
   VC[1]=VC[1]*0.3048;
   VC[2]=VC[2]*0.3048;
```

```
}

fout << "XC=" << XC[0] << "\t" << "YC=" << XC[1] << "\t" << "ZC=" << XC[2] <<
"\n";
fout << "VXC=" << VC[0] << "\t" << "VYC=" << VC[1] << "\t" << "VZC=" << VC[2]
<< "\n";


imago[5][0]=sqrt(pow(XC[0],2)+pow(XC[1],2)+pow(XC[2],2));
                //imago[5][0] becomes the distance of the satellite
                //from the Earth's centre at the time of imaging of the
                //centre of the scene.


//*****************
double ee,E,f_plus_omega;
double xs,ys;
if(image_case==1 || image_case==2){     //in the case of SPOT
    imago[1][0]=7200000.0;
    imago[0][2]=98.7*M_PI;
    ee=0.001;
}

else if(image_case==3 || image_case==4){   //in the case of MOMS-02
    imago[1][0]=6678000.0;
    imago[0][2]=28.5*M_PI/180.0;
    ee=0.001;
}

 E=acos((imago[1][0]-imago[5][0])/(imago[1][0]*ee));
 imago[0][0]=acos((cos(E)-e)/(1-e*cos(E)));  // imago[0][0]=true anomaly
 f_plus_omega=asin(XC[2]/(imago[5][0]*sin(imago[0][2])));
if(XC[2]<0)
 f_plus_omega=M_PI+(f_plus_omega*(-1));  //third quarter
else
 f_plus_omega=M_PI-(f_plus_omega);    //second quarter

imago[5][2]=f_plus_omega-imago[0][0]; //imago[5][2]=omega=argument of perigee
xs=imago[5][0]*cos(f_plus_omega);
ys=imago[5][0]*sin(f_plus_omega);
imago[0][1]=acos((XC[0]*xs+XC[1]*ys*cos(imago[0][2]))/
              (pow(xs,2)+pow(ys,2)*pow(cos(imago[0][2]),2)));
if(imago[0][1]>0)
 imago[0][1]=2*M_PI-imago[0][1];
else
 imago[0][1]=M_PI+(imago[0][1]*(-1));


//*****************
```

```
//****************

imago[6][0]=ee;
imago[2][0]=imago[19][0]; //omega(0)
imago[2][1]=imago[19][1]; //phi(0)
imago[2][2]=imago[19][2]; //kappa(0)
imago[1][1]=0.0; //F(1)
imago[1][2]=0.0; //longitude of the ascending node(1)
for( i=3; i<5;++i)
  for(int j=0; j<3;++j)
    imago[i][j]=imago[i+17][j]; //omega(1),phi(1),kappa(1)
                                //omega(2),phi(2),kappa(2)
fout << "\n" << imago[0][0] ;
fout << "\n" << imago[0][1] << "\t" << imago[0][2] << "\t" << imago[1][0];
fout << "\n" << imago[1][1] << "\t" << imago[1][2];
fout << "\n" << imago[2][0] << "\t" << imago[2][1] << "\t" << imago[2][2];
fout << "\n" << imago[3][0] << "\t" << imago[3][1] << "\t" << imago[3][2];
fout << "\n" << imago[4][0] << "\t" << imago[4][1] << "\t" << imago[4][2];
fout << "\n" << imago[5][0] << "\t" << imago[5][1] << "\t" << imago[5][2]
    << "\n" << imago[6][0];
fout.close();

}

/*************************************************************************
  Matrix Manipulation Sub-Module of Bundle Adjustment and Polynomial Adjustment
                                   Modules
*************************************************************************/

#include <math.h>
#include <bcd.h>
#include <memory.h>
#include <fstream.h>
#include <iomanip.h>
#include "pushbrom.h"

void amulti(double huge(*pt_matrix_a)[MAX],double huge(*pt_vector_v),
         double huge(*pt_vector_x),int column_x,int column_v)
{
ofstream fout;
fout.open("amulti.out");

 for(int i=0; i<column_x; ++i)
         pt_vector_x[i]=0.0;

 for( i=0; i<column_x; ++i){
   double sum=0.0;
   for(int j=0; j<column_v; ++j)
           sum=sum+pt_matrix_a[i][j]*pt_vector_v[j];
```

```
  pt_vector_x[i]=sum;}

 for( int jx=0; jx<column_x; ++jx)
  fout << pt_vector_x[jx] << "\n";

fout.close();
}

void multiply(double huge(*pt_matrix_a)[MAX],double huge(*pt_matrix_b)[MAX],
         double huge(*pt_matrix_c)[MAX],int row_c,int column_c,int column_a)
{
ofstream fout;
fout.open("multiple.out");

 for(int i=0; i<row_c; ++i)
  for(int j=0; j<column_c; ++j)
        pt_matrix_c[i][j]=0.0;

 for( i=0; i<row_c; ++i)
  for(int j=0; j<column_c; ++j){
      double sum=0.0;
        for(int k=0; k<column_a; ++k)
           sum+=pt_matrix_a[i][k]*pt_matrix_b[k][j];
           pt_matrix_c[i][j]=sum;}

 fout.close();
 }


 void choleski(double huge(*pt_matrix_a)[MAX],double huge (*pt_vector_v),
          double huge(*pt_vector_x),int row)
 {

 ofstream fout;
 fout.open("choleski.out");

   // matrix t is triangular matrix of original matrix.
   double huge(*pt_matrix_t)[MAX]= new double huge[MAX][MAX];
   double huge(*pt_matrix_at)[MAX]= new double huge[MAX][MAX];
   // vector r.
   double huge(*pt_vector_r)= new double huge[MAX];

    for(int ix=0; ix<MAX; ++ix) {
     pt_vector_x[ix]=0.0;
     pt_vector_r[ix]=0.0;
     for(int jx=0; jx<MAX; ++jx)
                 pt_matrix_t[ix][jx]=0.0;
    }
```

```
// matrix t calculation:

     pt_matrix_t[0][0]=sqrtl(pt_matrix_a[0][0]);

    for(int j=1; j<row; ++j)
      pt_matrix_t[0][j]=pt_matrix_a[0][j]/pt_matrix_t[0][0];

    for(int i=1;i<row;++i){
       double  st=0.0;
       for(int k=0; k<i; ++k)
               st+=pt_matrix_t[k][i]*pt_matrix_t[k][i];
                     double qq=pt_matrix_a[i][i]-st;

       pt_matrix_t[i][i]=sqrt(qq);

       fout <<"t(" << i << "," << i << ")=" << pt_matrix_t[i][i]
              <<"\n" << "qq= " << qq << "\n\n\n\n";

       for(j=i+1; j<row;++j){
     double s2=0.0;
             for(int kk=0; kk<i; ++kk)
                    s2+=pt_matrix_t[kk][i]*pt_matrix_t[kk][j];
             pt_matrix_t[i][j]=(pt_matrix_a[i][j]-s2)/pt_matrix_t[i][i];
       }
    }
     fout << "\n" << "%%% matrix T %%%" << "\n";
     for(i=0; i<row; ++i){
        for(int j=0; j<row; ++j){
              if(i>j)
                pt_matrix_t[i][j]=0.0;
                fout << pt_matrix_t[i][j] << "\t";}
   fout << "\n";
        }
for( i=0; i<row; ++i)
  for(int j=0; j<row; ++j)
        pt_matrix_at[i][j]=0.0;

 for( i=0; i<row; ++i)
  for(int j=0; j<row; ++j){
      double sum=0.0;
        for(int k=0; k<row; ++k)
           sum+=pt_matrix_t[k][i]*pt_matrix_t[k][j];
           pt_matrix_at[i][j]=sum;}

          // calculation of vector r:
             pt_vector_r[0]=pt_vector_v[0]/pt_matrix_t[0][0];

          for(i=1; i<row; ++i){
     double s=0.0;
```

```
                for(int k=0; k<i; ++k)
                   s+=pt_matrix_t[k][i]*pt_vector_r[k];
                pt_vector_r[i]=(pt_vector_v[i]-s)/pt_matrix_t[i][i];
               }
for( i=0; i<row; ++i)
       pt_vector_x[i]=0.0;

for( i=0; i<row; ++i){
  double sum=0.0;
  for(int j=0; j<row; ++j)
           sum+=pt_matrix_t[j][i]*pt_vector_r[j];
  pt_vector_x[i]=sum;}

          //calculation of vector x:
               pt_vector_x[row-1]=pt_vector_r[row-1]/pt_matrix_t[row-1][row-1];
               for(i=2; i<=row; ++i){
                 double s1=0.0;
                 for(int k=row-i+1; k<row; ++k)
                   s1+=pt_matrix_t[row-i][k]*pt_vector_x[k];
                 pt_vector_x[row-i]=(pt_vector_r[row-i]-s1)/pt_matrix_t[row-i][row-i];
               }
for( i=0; i<row; ++i){
  double sum=0.0;
  for(int j=0; j<row; ++j)
           sum+=pt_matrix_a[i][j]*pt_vector_x[j];
  pt_vector_r[i]=sum;}

fout.close();

                 delete pt_matrix_t;
                 delete pt_vector_r;
                 delete pt_matrix_at;
         }

void inverse(double (*pt_matrix_a)[MAX],double (*pt_matrix_c)[MAX], int row)
{
  double  huge(*pt_matrix_t)[MAX]= new double huge[MAX][MAX];
  double huge (*pt_matrix_l)[MAX]= new double huge[MAX][MAX];
  double huge(*pt_matrix_at)[MAX]= new double huge[MAX][MAX];
  double huge(*pt_matrix_b)[MAX]= new double huge[MAX][MAX];

 ofstream fout;
 fout.open("invers.out");
 double s1,s2,s;

                 pt_matrix_t[0][0]=sqrt(pt_matrix_a[0][0]);

                 for(int j=1; j<row; ++j)
                    pt_matrix_t[0][j]=pt_matrix_a[0][j]/pt_matrix_t[0][0];
```

```
for(int i=1;i<row;++i){
   s1=0.0;
   for(int k=0; k<i; ++k){

         s1=s1+pt_matrix_t[k][i]*pt_matrix_t[k][i];
         }
   pt_matrix_t[i][i]=sqrt(pt_matrix_a[i][i]-s1);
   for(j=i+1; j<row;++j){
        s2=0.0;
        for(int kk=0; kk<i; ++kk)
    s2=s2+pt_matrix_t[kk][i]*pt_matrix_t[kk][j];
pt_matrix_t[i][j]=(pt_matrix_a[i][j]-s2)/pt_matrix_t[i][i];}}

for(i=0; i<row; ++i)
   for(int j=0; j<row; ++j){
        if(i>j)
          pt_matrix_t[i][j]=0.0;
   }


// calculation of matrix l(inverse of matrix t).

for(i=0; i<row; ++i)
  pt_matrix_l[i][i]=1/pt_matrix_t[i][i];

for(int it=1;it<row;++it)
      for(i=0; i<(row-it);++i){
        int j=i+it;
          s=0.0;
          for(int k=i+1; k<=j; ++k)
                s=s+pt_matrix_t[i][k]*pt_matrix_l[k][j];
          pt_matrix_l[i][j]=-pt_matrix_l[i][i]*s;
        }

   for(i=0; i<row; ++i)
      for(int j=0; j<row; ++j){
           if(i>j)
             pt_matrix_l[i][j]=0.0;
      }

   // calculation of a_inv(original inverse matrix).

   for(i=0; i<row; ++i){
      double s1=0.0;
      for(int k=i; k<row; ++k)
            s1=s1+pt_matrix_l[i][k]*pt_matrix_l[i][k];
      pt_matrix_c[i][i]=s1;
   }
```

```
        for(i=0; i<row; ++i)
          for(int j=i+1; j<row; ++j){
                double s2=0.0;
                for(int k=j; k<row; ++k)
                   s2=s2+pt_matrix_l[i][k]*pt_matrix_l[j][k];
                pt_matrix_c[i][j]=s2;
          }

        for(i=0; i<row; ++i)
           for(int j=0; j<row; ++j){
                if(i>j)
                   pt_matrix_c[i][j]=pt_matrix_c[j][i];
           }

fout.close();

        delete pt_matrix_t;
        delete pt_matrix_l;
        delete pt_matrix_b;
        delete pt_matrix_at;

}
```

```
/*************************************************************************
    The Resource File Program (POBALAT.RC) for the Main Adjustment Program
*************************************************************************/

#include <windows.h>
#include "pobalat.h"
#include "gcp.h"
#include "poly.h"


POBALAT ICON "pobalat.ico"

POBALAT MENU
   {
   POPUP "&File"
      {
      MENUITEM "&New",            IDM_NEW
      MENUITEM "&Open...",        IDM_OPEN
      MENUITEM "&Save",           IDM_SAVE
      MENUITEM "Save &As...",     IDM_SAVEAS
      MENUITEM SEPARATOR
      MENUITEM "&Print...",       IDM_PRINT
      MENUITEM SEPARATOR
      MENUITEM "E&xit",           IDM_EXIT
      }
   POPUP "&Edit"
      {
      MENUITEM "&Undo\tCtrl+Z",   IDM_UNDO
      MENUITEM SEPARATOR
      MENUITEM "Cu&t\tCtrl+X",    IDM_CUT
      MENUITEM "&Copy\tCtrl+C",   IDM_COPY
      MENUITEM "&Paste\tCtrl+V",  IDM_PASTE
      MENUITEM "De&lete\tDel",    IDM_DEL
      MENUITEM SEPARATOR
      MENUITEM "&Select All",     IDM_SELALL
      }
   POPUP "&Search"
      {
      MENUITEM "&Find...",        IDM_FIND
      MENUITEM "Find &Next\tF3",  IDM_NEXT
      MENUITEM "&Replace...",     IDM_REPLACE
      }
   POPUP "&Character"
      {
      MENUITEM "&Font...",        IDM_FONT
       }
    POPUP "&Poly/Bundle Adj. Program"
    BEGIN
          MENUITEM "&Polynomial (2-D) Adj.", IDM_2D
```

```
            MENUITEM "&Bundle (3-D) Adj.", IDM_3D
    END

      POPUP "R&UN"
      BEGIN
            MENUITEM "&Run Polynomial Adj.", IDM_RUN2D
            MENUITEM "&Run Bundle Adj.", IDM_RUN3D
      END

   POPUP "&Help"
     {
     MENUITEM "&Help",          IDM_HELP
     MENUITEM "&About PopPad...",  IDM_ABOUT
     }
   }


AboutBox DIALOG  20, 20, 160, 130
   STYLE WS_POPUP | WS_DLGFRAME
   {
   CTEXT "\" PABALAT \""                         -1,  0, 12, 160,  8
   ICON  "POBALAT"                         -1,  8,  8,  0,  0
   CTEXT "About  \" PABALAT \"  Program"             -1,  0, 36, 160,  8
   CTEXT "A Polynomial and Bundle Adjustment Program" -1,  0, 48, 160,  8
   CTEXT "for Space Linear Array Technology "         -1,  0, 60, 160,  8
   CTEXT "M. J. Valadan Zoej, 1996"          -1,  0, 82, 160,  8
   CTEXT "University of Glasgow"           -1,  0, 94, 160,  8
   DEFPUSHBUTTON "OK"             IDOK,  64, 110,  32, 14, WS_GROUP
   }
BeginBox DIALOG  20, 20, 160, 120
CAPTION "PABALAT"
   STYLE WS_POPUP | WS_DLGFRAME
   {
   CTEXT "\" WELCOME TO PABALAT \""             -1,  0, 12, 160,  8
   ICON  "POBALAT"                         -1,  8,  8,  0,  0
   CTEXT "A Polynomial and Bundle Adjustment Program" -1,  0, 36, 160,  8
   CTEXT "for Space Linear Array Technology "         -1,  0, 48, 160,  8
   CTEXT "M. J. Valadan Zoej, 1996"          -1,  0, 70, 160,  8
   CTEXT "University of Glasgow"           -1,  0, 82, 160,  8
   DEFPUSHBUTTON "OK"             IDOK,  64, 100,  32, 14, WS_GROUP
   }

D3DLG DIALOG DISCARDABLE LOADONCALL PURE MOVEABLE 28, 17, 232,
115
STYLE WS_POPUP | WS_CAPTION | WS_SYSMENU | 0x80L
CAPTION "Space Linear Array Stereo Systems"
BEGIN
      CONTROL  "OK",  1,  "button",  BS_DEFPUSHBUTTON  |  WS_GROUP  |
WS_TABSTOP | WS_CHILD, 18, 95, 30, 14
```

```
CONTROL "Cancel", 2, "button", BS_PUSHBUTTON | WS_TABSTOP | WS_CHILD, 58, 95, 30, 14
ICON "POBALAT" -1, 205, 95, 0, 0
CONTROL "Stereo SPOT Level 1A", IDD_SPOT1A, "button", BS_RADIOBUTTON | WS_GROUP | WS_TABSTOP | WS_CHILD, 11, 25, 85, 12
CONTROL "Stereo SPOT Level 1B", IDD_SPOT1B, "button", BS_RADIOBUTTON | WS_TABSTOP | WS_CHILD, 11, 39, 85, 12
CONTROL "Stereo IRS-1C", IDD_IRS1C, "button", BS_RADIOBUTTON | WS_TABSTOP | WS_CHILD, 11, 53, 85, 12
CONTROL "Cross-Track Systems", 103, "button", BS_GROUPBOX | WS_CHILD, 5, 10, 100, 67
CONTROL "Stereo MOMS-02 Mode 1", IDD_MOMS1, "button", BS_RADIOBUTTON | WS_GROUP | WS_TABSTOP | WS_CHILD, 128, 25, 85, 12
CONTROL "Stereo MOMS-02 Mode 3", IDD_MOMS3, "button", BS_RADIOBUTTON | WS_TABSTOP | WS_CHILD, 128, 39, 87, 12
CONTROL "Stereo OPS", IDD_OPS, "button", BS_RADIOBUTTON | WS_TABSTOP | WS_CHILD, 128, 53, 85, 12
CONTROL "Along-Track Systems", 103, "button", BS_GROUPBOX | WS_CHILD, 120, 10, 105, 67
END

D2DLG DIALOG DISCARDABLE LOADONCALL PURE MOVEABLE 28, 17, 272, 160
STYLE WS_POPUP | WS_CAPTION | WS_SYSMENU | 0x80L
CAPTION "Selection of Number of Terms in Polynomial"
BEGIN
CONTROL "OK", 1, "button", BS_DEFPUSHBUTTON | WS_GROUP | WS_TABSTOP | WS_CHILD, 18, 115, 30, 14
CONTROL "Cancel", 2, "button", BS_PUSHBUTTON | WS_TABSTOP | WS_CHILD, 18, 140, 30, 14
ICON "POBALAT" -1, 250, 135, 0, 0
CONTROL "3 Terms", IDD_T3, "button", BS_RADIOBUTTON | WS_GROUP | WS_TABSTOP | WS_CHILD, 15, 25, 38, 12
CONTROL "4 Terms", IDD_T4, "button", BS_RADIOBUTTON | WS_TABSTOP | WS_CHILD, 15, 39, 38, 12
CONTROL "5 Terms", IDD_T5, "button", BS_RADIOBUTTON | WS_TABSTOP | WS_CHILD, 15, 53, 38, 12
CONTROL "6 Terms", IDD_T6, "button", BS_RADIOBUTTON | WS_TABSTOP | WS_CHILD,15 , 67, 38, 12
CONTROL "7 Terms", IDD_T7, "button", BS_RADIOBUTTON | WS_TABSTOP | WS_CHILD, 15, 81, 38, 12
CONTROL "8 Terms", IDD_T8, "button", BS_RADIOBUTTON | WS_TABSTOP | WS_CHILD, 65, 25, 38, 12
CONTROL "9 Terms", IDD_T9, "button", BS_RADIOBUTTON | WS_TABSTOP | WS_CHILD, 65, 39, 38, 12
CONTROL "10 Terms", IDD_T10, "button", BS_RADIOBUTTON | WS_TABSTOP | WS_CHILD, 65, 53, 38, 12
CONTROL "11 Terms", IDD_T11, "button", BS_RADIOBUTTON | WS_TABSTOP | WS_CHILD, 65, 67, 38, 12
```

```
   CONTROL "12 Terms", IDD_T12, "button", BS_RADIOBUTTON | WS_TABSTOP
| WS_CHILD, 65, 81, 38, 12
   CONTROL "13 Terms", IDD_T13, "button", BS_RADIOBUTTON | WS_TABSTOP
| WS_CHILD, 115, 25, 38, 12
   CONTROL "14 Terms", IDD_T14, "button", BS_RADIOBUTTON | WS_TABSTOP
| WS_CHILD, 115, 39, 38, 12
   CONTROL "15 Terms", IDD_T15, "button", BS_RADIOBUTTON | WS_TABSTOP
| WS_CHILD, 115, 53, 38, 12
   CONTROL "16 Terms", IDD_T16, "button", BS_RADIOBUTTON | WS_TABSTOP
| WS_CHILD, 115, 67, 38, 12
   CONTROL "17 Terms", IDD_T17, "button", BS_RADIOBUTTON | WS_TABSTOP
| WS_CHILD, 115, 81, 38, 12
   CONTROL "18 Terms", IDD_T18, "button", BS_RADIOBUTTON | WS_TABSTOP
| WS_CHILD, 165, 25, 38, 12
   CONTROL "19 Terms", IDD_T19, "button", BS_RADIOBUTTON | WS_TABSTOP
| WS_CHILD, 165, 39, 38, 12
   CONTROL "20 Terms", IDD_T20, "button", BS_RADIOBUTTON | WS_TABSTOP
| WS_CHILD, 165, 53, 38, 12
   CONTROL "21 Terms", IDD_T21, "button", BS_RADIOBUTTON | WS_TABSTOP
| WS_CHILD, 165, 67, 38, 12
   CONTROL "22 Terms", IDD_T22, "button", BS_RADIOBUTTON | WS_TABSTOP
| WS_CHILD, 165, 81, 38, 12
   CONTROL "23 Terms", IDD_T23, "button", BS_RADIOBUTTON | WS_TABSTOP
| WS_CHILD, 215, 25, 38, 12
   CONTROL "24 Terms", IDD_T24, "button", BS_RADIOBUTTON | WS_TABSTOP
| WS_CHILD, 215, 39, 38, 12
   CONTROL "25 Terms", IDD_T25, "button", BS_RADIOBUTTON | WS_TABSTOP
| WS_CHILD, 215, 53, 38, 12
    CONTROL "Number of Terms in Polynomial", 53, "button", BS_GROUPBOX |
WS_CHILD, 10, 10, 250, 90
   CONTROL "", IDD_PIXELSIZE, "edit", ES_LEFT | WS_BORDER | WS_GROUP |
WS_TABSTOP | WS_CHILD, 180, 115,25, 14
    CONTROL "", IDD_NGCPS, "edit", ES_LEFT | WS_BORDER | WS_TABSTOP |
WS_CHILD, 180, 140, 25, 14
    CONTROL "            Pixel Size\n(in the object space)", 113, "static", SS_LEFT |
WS_CHILD, 100, 115, 75, 20
   CONTROL "          Number of \nGround Control Points", 114, "static", SS_LEFT |
WS_CHILD, 100, 140, 75, 20

END

PrintDlgBox DIALOG 20, 20, 100, 76
   STYLE WS_POPUP | WS_CAPTION | WS_SYSMENU | WS_VISIBLE
   CAPTION "POBALAT"
   {
   CTEXT "Sending",             -1,  0, 10, 100,  8
   CTEXT "",              IDD_FNAME,  0, 20, 100,  8
   CTEXT "to print spooler.",     -1,  0, 30, 100,  8
   DEFPUSHBUTTON  "Cancel",   IDCANCEL, 34, 50,  32, 14, WS_GROUP}
```

```
/**************************************************************************
     The Header Data File (PUSHBROOM.H) for the Bundle Adjustment Program
**************************************************************************/

const MAX=120;
const max_i=100;
extern int no_of_ext_p;
extern int image_case;
extern int parameter;
extern int np1;
extern int np2;
extern int ngp;
extern int ncp;
extern float focal_b;
extern float focal_f;
extern float focal;
extern int NPO;
extern float pixelc;
extern float view_angle;
extern float angle_l;
extern float angle_r;
extern float  _idd_pixelsize;
extern double e;
extern double Cx;
extern double Cy;
extern double UU;
extern double VV;
extern double WW;
extern double X_prj;
extern double Y_prj;
extern double Z_prj;


void adjust(float huge(*image1)[4], float huge(*image2)[4], double huge(*ground_xyz)[7],
          double huge(*imago1)[3],double huge (*imago2)[3],
        double huge(*pixel1_xy)[2], double huge (*pixel2_xy)[2]);

void amulti(double huge(*pt_matrix_a)[MAX],double huge(*pt_vector_v),
        double huge(*pt_vector_x),int column_x,int column_v);

void multiply(double huge(*pt_matrix_a)[MAX],double huge(*pt_matrix_b)[MAX],
              double huge(*pt_matrix_c)[MAX],int,int,int);

void choleski(double huge(*pt_matrixc_a)[MAX],double huge(*pt_vector_v),
          double huge(*pt_vector_x),int);
void inverse(double (*pt_matrix_a)[MAX],double (*pt_matrix_c)[MAX], int row);

void l_b_to_a(float huge (*image_xy)[4],double huge (*pixel_xy)[2],
```

```
        int l_c1,int l_c2,
        int p_c1,int p_c2);

void be_bg(double huge(*Be)[15],double phi,double omega,double kapa,double F,
        double C_OMEGA,double ground_x,double ground_y,
        double ground_z,double pixel_xy,double pixel_y,int centre,
        double r,double a_omega,double inclin,double y_image,
        int intersect, float size);

void lagrange(double huge(*imago)[3]);
```

```
/*************************************************************************
  The Bundle Adjustment Program (Case 3) including Space Resection and Intersection
              Procedure (A Sub-Module of Bundle Adjustment Module)
*************************************************************************/

#include <fstream.h>
#include <iomanip.h>
#include <math.h>
#include <memory.h>
#include "pushbroom.h"

double UU,VV,WW;
double X_prj,Y_prj,Z_prj;
double e;
double Cx,Cy;
float focal;
float view_angle;

void adjust(float huge(*image1)[4], float huge(*image2)[4], double huge(*ground_xyz)[7],
         double huge(*imago1)[3],double huge (*imago2)[3],
       double huge(*pixel1_xy)[2], double huge (*pixel2_xy)[2])
{
double huge(*image_eo)[3]=new double huge[30][3];
float huge(*image_xy)[4]=new float huge[max_i][4];
double huge(*pixel_xy)[2]=new double huge[max_i][2];

double huge(*Be)[15]=new double huge[2][15];
double huge(*Bg)[3]=new double huge[2][3];
double huge (*pt_vector_v)= new double huge[MAX];
double huge (*pt_vector_x)= new double huge[MAX];

double huge (*pt_matrix_a)[MAX1]= new double huge[MAX1][MAX1];
double huge (*pt_matrix_b)[MAX]= new double huge[MAX][MAX];
double huge (*pt_matrix_c)[MAX]= new double huge[MAX][MAX];

double huge(*eop_obs)[3]=new double huge[max_i][3];
double huge(*xyz_obs)[7]=new double huge[max_i][7];
double huge(*eop_it)[3]=new double huge[max_i][3];

float huge (*X0)[1]=new float huge [max_i][1];
float huge (*Y0)[1]=new float huge [max_i][1];
float huge (*Z0)[1]=new float huge [max_i][1];

double huge (*BBAR_11)[MAX]=new double huge [MAX][MAX];
double huge (*BBAR_12)[MAX]=new double huge [MAX][MAX];
double huge (*BBAR_21)[MAX]=new double huge [MAX][MAX];
double huge (*BBAR_22)[MAX]=new double huge [MAX][MAX];

double huge (*mat_u1)=new double huge [MAX];
```

```
double huge (*mat_u2)=new double huge [MAX];
double huge (*mat_x1)=new double huge [MAX];
double huge (*mat_x2)=new double huge [MAX];
double huge (*BTWB1)[MAX]=new double huge [MAX][MAX];
double huge (*BTWB2)[MAX]=new double huge [MAX][MAX];
double huge (*BTWE1)=new double huge [MAX];
double huge (*BTWE2)=new double huge [MAX];
double huge (*BTWE3)=new double huge [MAX];
double huge (*EBAR1)=new double huge [MAX];
double huge (*EBAR2)=new double huge [MAX];
double huge (*EBARg)=new double huge [MAX];
double huge (*EBARe)=new double huge [MAX];

double huge (*normal_11)[MAX]=new double huge [MAX][MAX];
double huge (*normal_12)[MAX]=new double huge [MAX][MAX];
double huge (*normal_21)[MAX]=new double huge [MAX][MAX];
double huge (*normal_22)[MAX]=new double huge [MAX][MAX];

double huge (*weight)=new double huge [MAX];
double huge (*XCP)=new double huge [max_i];
double huge (*YCP)=new double huge [max_i];
double huge (*ZCP)=new double huge [max_i];
double huge (*DXX)=new double huge [max_i];
double huge (*DYY)=new double huge [max_i];
double huge (*DZZ)=new double huge [max_i];
double huge (*landa_r)=new double huge [max_i];

double XR,YR,ZR;
double UR,VR,WR;
double XL,YL,ZL;
double UL,VL,WL;

ofstream final;
final.open("final.out");
ofstream fout;

double XP;
double YP;
double a_of_p; //argument of prigee

//Put the values of the matrix ground_xyz including the object coordinates
//in the matrix xyz_obs which will be used as observations in the iteration:

for(int i=0; i<ngp; ++i) {
  for(int j=0; j<7; ++j){
    xyz_obs[i][j]=ground_xyz[i][j];}}

lagrange(imago1);
```

```
double    a_of_p1=imago1[5][2];  // argument of perigee
double    e1=imago1[6][0];       // eccentricity
lagrange(imago2);
imago2[0][0]=83.31537595*M_PI/180.0; //this is for Sudan

double    a_of_p2=imago2[5][2];  // argument of perigee
double    e2=imago2[6][0];       // eccentricity

for(int ieo=0; ieo<10; ++ieo){
 for(int jeo=0; jeo<3; ++jeo){
  if(ieo<5)
   eop_obs[ieo][jeo]=imago1[ieo][jeo];
  else
   eop_obs[ieo][jeo]=imago2[ieo-5][jeo];
 }
}


//**************************************************************
//**************************************************************
//****************** Start of the iteration ************************
//**************************************************************
//**************************************************************
float size;
for(int it=1; it<=30;++it){   //open the iteration acolad

for (int i=0; i<MAX; ++i){
 for (int j=0; j<MAX; ++j){
   BBAR_11[i][j]=0.0;
   BBAR_21[i][j]=0.0;
   BBAR_12[i][j]=0.0;
   BBAR_22[i][j]=0.0;}}

int np=np1;

for(i=0;i<np;++i) {
 for(int j=0;j<4;++j){
  image_xy[i][j]=image1[i][j];} }

for(i=0;i<np;++i)
 for(int j=0;j<2;++j)
  pixel_xy[i][j]=pixel1_xy[i][j];


//Placing the elements of the matrices imago1, imago2, and imago3
//in a handle matrix as image_eo. id is the parameter which indicates the
//number of images (NPO). NPO is two in SPOT and MOMS-02, mode 3 and
//is equal to three in the case of MOMS-02 mode 1.
```

```
for(int id=1; id<=NPO; ++id){//open the acolad number 0.
 if(id==2){
   np=np2;
   for(int i9=1; i9<=np; ++i9)
     for(int j9=1; j9<=4; ++j9)
          image_xy[i9-1][j9-1]=image2[i9-1][j9-1];

    for(i9=1; i9<=np; ++i9)
     for(int j9=1; j9<=2; ++j9)
          pixel_xy[i9-1][j9-1]=pixel2_xy[i9-1][j9-1];

    for(int io=0; io<5; ++io) {
      for(int jo=0; jo<3; ++jo){
        image_eo[io][jo]=imago2[io][jo];
        eop_it[io+5][jo]=imago2[io][jo]; }}

        a_of_p=a_of_p2;  // argument of perigee
     e=e2;       // eccentricity
        view_angle = angle_r;
   if(image_case==3){
     size=0.010;
     pixelc=4060;
     pixelr=2900;
   }

  else if(image_case==4){
     size=0.010;
     pixelc=4060;
     pixelr=1488.5;
   }
   else{
     size=0.013;
     pixelc=3000;
     pixelr=3000;
   }
  }

  else if(id==1){
   if(image_case==3){
     size=0.010;
     pixelc=4060;
     pixelr=2900;
   }
   else if(image_case==4){
     size=0.010;
     pixelc=4060;
     pixelr=1488.5;
    }
    else{
```

```
  size=0.013;
  pixelc=3000;
  pixelr=3000;
 }

  for(int io=0; io<5; ++io){
   for(int jo=0; jo<3; ++jo){
     image_eo[io][jo]=imago1[io][jo];
     eop_it[io][jo]=imago1[io][jo]; }}
     a_of_p=a_of_p1;  // argument of perigee
     e=e1;       // eccentricity
     view_angle = angle_1;}

     // initializing the focal length

      if(id==1)
       focal=focal_b/1000.0;
   else if (id==2)
       focal=focal_f/1000.0;


//*************************************************************
//*************************************************************
 //Placing the elements of matrix image_eo related to the exterior
 //orientation parameters in 18 different vectors as follows:


 for(int ii=1; ii<=np; ++ii){   //open the acolad number 1.

      XP=0.0;   //x-x0
    YP=image_xy[ii-1][3];//-Ypp;   //y-y0

   be_bg(ground_xyz[ii-1][1],ground_xyz[ii-1][2],ground_xyz[ii-1][3],
         pixel_xy[ii-1][0],pixel_xy[ii-1][1],YP,
         a_of_p,image_eo[0][2],0,size);

//*******************************************************************
   //Placing the elements of matrix Be into the general matrices BBAR_11
   //and BBAR_21 for the first image and second image respectively:
//*******************************************************************

    int kk;
    if(id==1){
        kk=ii*2-1;
        int ik=15;

        for(int i5=1; i5<=15; ++i5){
          BBAR_11[kk-1][i5-1]=Be[0][i5-1];
        BBAR_11[kk-1][i5+ik-1]=0.0;
```

```
        BBAR_11[kk][i5-1]=Be[1][i5-1];
        BBAR_11[kk][i5+ik-1]=0.0;

      }
   }
   else if(id==2){
      kk=ii*2-1;
      int ik=15;
      final << "ii=" << ii << "\t and id=" << id << "\n";
      for(int i5=1; i5<=15; ++i5){
     BBAR_21[kk-1][i5-1]=0.0;
        BBAR_21[kk-1][i5+ik-1]=Be[0][i5-1];
        BBAR_21[kk][i5+ik-1]=Be[1][i5-1];
        BBAR_21[kk][i5-1]=0.0;

    }
   }

   //Placing the elements of matrix Bg in the general matrix BBAR:
   int kg;
   for(int i6=1; i6<=2; ++i6)
       for(int j6=1; j6<=3; ++j6){
         int ig=(ii-1)*3+j6;
         if(id==1){
           kg=ii*2-2+i6;
           BBAR_12[kg-1][ig-1]=Bg[i6-1][j6-1];}
         else if(id==2) {
           kg=ii*2-2+i6;
           BBAR_22[kg-1][ig-1]=Bg[i6-1][j6-1];}
       }

   //Form the matrix EBAR:

   if(id==1){
       EBAR1[kk-1]=XP-Cx;
       EBAR1[kk]=YP-Cy;
       final << "yp1-cy1=" << EBAR1[kk] << "\t"
            << "xp1-cx1=" << EBAR1[kk-1]<< "\n";}

   else if(id==2){
     EBAR2[kk-1]=XP-Cx;
       EBAR2[kk]=YP-Cy;

       final << "yp2-cy2=" << EBAR2[kk]<< "\t"
           << "xp2-cx2=" << EBAR2[kk-1]<< "\n";}


 }   // close acolad number 1.
} //close acolad number 0.
```

```
//**************************************************************
//**************************************************************

//Adding to matrix EBAR, the elements related to the quasi-observations
//of GCPs:

for(int i8=1; i8<=ngp; ++i8)
  for(int j8=1; j8<=3; ++j8){
    int kj8=(i8-1)*3+j8;
    EBARg[kj8-1]=xyz_obs[i8-1][j8]-ground_xyz[i8-1][j8];

  }

//Adding to matrix EBAR, the elements related to the quasi-observations
//of GCPs:


for(i8=1; i8<=10; ++i8)
  for(int j8=1; j8<=3; ++j8){
    int kj8=(i8-1)*3+j8;
    EBARe[kj8-1]=eop_obs[i8-1][j8-1]-eop_it[i8-1][j8-1];

  }

//computation of the matrix N which has been divided into four submatrix
//N11, N12, N21, N22. In our case:
//N11=normal_11=BBART_11*W1_bar*BBAR_11+BBART_21*W2_bar*BBAR_21+
//          BBART_31*W3_bar*BBAR_31+BBART_41*Wg_bar*BBAR_41. and,
//N12=normal_12=BBART_11*W1_bar*BBAR_12+BBART_21*W2_bar*BBAR_22+
//          BBART_31*W3_bar*BBAR_32+BBART_41*Wg_bar*BBAR_42. and,
//N21=normal_21=BBART_12*W1_bar*BBAR_11+BBART_22*W2_bar*BBAR_21+
//          BBART_32*W3_bar*BBAR_31+BBART_42*Wg_bar*BBAR_41. and,
//N22=normal_22=BBART_12*W1_bar*BBAR_12+BBART_22*W2_bar*BBAR_22+
//          BBART_32*W3_bar*BBAR_32+BBART_42*Wg_bar*BBAR_42.

int ncb11,ncb12,ncb21,ncb22,ncb31,nrb31;
int nrb11,nrb12,nrb21,nrb22,nrb41,nrb42;

  ncb11=15*NPO;
  ncb21=15*NPO;
  ncb31=15*NPO;
  ncb12=3*ngp;
  ncb22=3*ngp;
  nrb11=np1*2;
  nrb21=np2*2;
  nrb22=np2*2;
  nrb31=15*NPO;
  nrb41=3*ngp;
  nrb42=3*ngp;
```

```
//computation of BBART*mat_wb where:
//(mat_wb11=W1_bar*BBAR_11 to mat_wb42=Wg_bar*BBAR_42)

multiply(BBAR_11,BBAR_11,BTWB1,ncb11,ncb11,nrb11);
//BBART_11*W1_bar*BBAR_11;

multiply(BBAR_21,BBAR_21,BTWB2,ncb21,ncb21,nrb21);
//BBART_11*W1_bar*BBAR_11;

for(int ix=1; ix<=2;++ix){
  weight[(ix-1)*15]=1.0/pow(0.02,2);
  weight[(ix-1)*15+1]=1.0/pow(0.02,2);
  weight[(ix-1)*15+2]=1.0/pow(0.001,2);
  weight[(ix-1)*15+3]=1.0/pow(100,2);
  weight[(ix-1)*15+4]=1.0/pow(0.001,2);
  weight[(ix-1)*15+5]=1.0/pow(0.001,2); }
for(ix=6; ix<9;++ix){
  weight[ix]=1.0/pow(0.01,2);
  weight[ix+15]=1.0/pow(0.01,2);}
for(ix=9; ix<12;++ix){
  weight[ix]=1.0/pow(0.00001,2);
  weight[ix+15]=1.0/pow(0.00001,2);}
for(ix=12; ix<15;++ix){
  weight[ix]=1.0/pow(0.00001,2);
  weight[ix+15]=1.0/pow(0.00001,2);}


double sigma=pow(0.00001,2);

for(ix=0; ix<30;++ix){
  weight[ix]=weight[ix]*sigma;}

//Computation of matrix normal
//   N=BTWB1+BTWB2+The diagonal weight matrix related to exterior orientation
parameters

for(ix=0; ix<ncb11; ++ix){
  for(int jx=0; jx<ncb11; ++jx) {
normal_11[ix][jx]=BTWB1[ix][jx]+BTWB2[ix][jx];}}

for(ix=0;ix<ncb11;++ix)
  normal_11[ix][ix]=normal_11[ix][ix]+weight[ix];


//*********************************************************************
********************
//*********************************************************************
********************
```

```
multiply(BBAR_11,BBAR_12,BTWB1,ncb11,ncb12,nrb11);
//BBART_11*W1_bar*BBAR_11;

multiply(BBAR_21,BBAR_22,BTWB2,ncb21,ncb22,nrb21);
//BBART_11*W1_bar*BBAR_11;

//Computation of mterix normal_12=BTWB1+BTWB2+BTWB3+BTWB4

for(ix=0; ix<ncb11; ++ix){
 for(int jx=0; jx<ncb12; ++jx) {
normal_12[ix][jx]=BTWB1[ix][jx]+BTWB2[ix][jx];}}

//*********************************************************************
******************
//*********************************************************************
******************
multiply(BBAR_12,BBAR_11,BTWB1,ncb12,ncb11,nrb11);
//BBART_11*W1_bar*BBAR_11;

multiply(BBAR_22,BBAR_21,BTWB2,ncb22,ncb21,nrb21);
//BBART_11*W1_bar*BBAR_11;


//Computation of mterix normal_21=BTWB1+BTWB2+BTWB3+BTWB4

for(ix=0; ix<ncb12; ++ix){
 for(int jx=0; jx<ncb11; ++jx) {
normal_21[ix][jx]=BTWB1[ix][jx]+BTWB2[ix][jx];}}


//*******************************************************************
//*********************************************************************


multiply(BBAR_12,BBAR_12,BTWB1,ncb12,ncb12,nrb12);
//BBART_11*W1_bar*BBAR_11;

multiply(BBAR_22,BBAR_22,BTWB2,ncb22,ncb22,nrb22);
//BBART_11*W1_bar*BBAR_11;

//Computation of mterix normal_22=BTWB1+BTWB2+BTWB3+BTWB4

double sigz=sigma/64; //10m accuracy for the GCPs
for(ix=0; ix<ncb12; ++ix){
 for(int jx=0; jx<ncb12; ++jx) {
  normal_22[ix][jx]=BTWB1[ix][jx]+BTWB2[ix][jx];}}

for(ix=0; ix<ncb12; ++ix){
  normal_22[ix][ix]=normal_22[ix][ix]+sigz;}
```

```
//**********************************************************************
//**********************************************************************
//**********************************************************************

for(ix=0; ix<(ncb11+ncb12);++ix){
 for(int jx=0; jx<(ncb11+ncb12); ++jx){
        pt_matrix_a[ix][jx]=0.0;}}
for(ix=0; ix<(ncb11+ncb12);++ix){
 for(int jx=0; jx<(ncb11+ncb12); ++jx){
  if(ix<30 & jx<30)
   pt_matrix_a[ix][jx]=normal_11[ix][jx];
  else if(ix<30 & jx>=30)
   pt_matrix_a[ix][jx]=normal_12[ix][jx-30];
  else if(ix>=30 & jx<30)
   pt_matrix_a[ix][jx]=normal_21[ix-30][jx];
  else
   pt_matrix_a[ix][jx]=normal_22[ix-30][jx-30];}}

//*****************************************************************
//*****************************************************************

//computation of matrices U1=mat_u1 and U2=mat_u2:
//mat_u1=BBART_11*W1_bar*EBAR1+BBART_21*W2_bar*EBAR2+
//       BBART_31*W3_bar*EBAR3+BBART_41*Wg_bar*EBARg; and
//mat_u2=BBART_12*W1_bar*EBAR1+BBART_22*W2_bar*EBAR2+
//       BBART_32*W3_bar*EBAR3+BBART_42*Wg_bar*EBARg.


 amulti(BBAR_11,EBAR1,BTWE1,ncb11,nrb11);  //BBART_11*W1_bar*EBAR1;

 amulti(BBAR_21,EBAR2,BTWE2,ncb21,nrb21);  //BBART_21*W2_bar*EBAR2;

 for(ix=0; ix<ncb11; ++ix){
  BTWE3[ix]=weight[ix]*EBARe[ix];
  BTWE3[ix]=-BTWE3[ix];}

 //Computation of mterix mat_u1=BTWE1+BTWE2+BTWE3

 for(ix=0; ix<ncb11; ++ix)
  mat_u1[ix]=BTWE1[ix]+BTWE2[ix]+BTWE3[ix];

 amulti(BBAR_12,EBAR1,BTWE1,ncb12,nrb11);  //BBART_12*W1_bar*EBAR1;

 amulti(BBAR_22,EBAR2,BTWE2,ncb22,nrb21);  //BBART_22*W2_bar*EBAR2;

 for(ix=0; ix<nrb41; ++ix){
  BTWE3[ix]=sigz*EBARg[ix];
  BTWE3[ix]=-BTWE3[ix];}
```

```
//Computation of mterix mat_u2=BTWE1+BTWE2+BTWE3+BTWE4

for(ix=0; ix<ncb12; ++ix)
  mat_u2[ix]=BTWE1[ix]+BTWE2[ix]+BTWE3[ix];

for(ix=0;ix<(ncb11+ncb12);++ix){
 if(ix<30)
   mat_x2[ix]=mat_u1[ix];
 else
   mat_x2[ix]=mat_u2[ix-30];}

//*******************************************************************
//*******************************************************************
//*******************************************************************
int row_n=(ncb11+ncb12);
choleski(pt_matrix_a,mat_x2,mat_x1,row_n);

for(ix=0;ix<(ncb11+ncb12);++ix){
 mat_x1[ix]=-mat_x1[ix];}

//**************************************************************
//**************************************************************

final << "\n\n\n
**********************************************************"
   << "\n        ***********************************************************"
   << "\n        ***********************************************************\n"
   << "        ***                                        ***\n";

//Printing the results:

//print the iteration number:

final << "        *** In iteration (" << it << ") the corrections are as follows: ***\n"
   << "        ***                                         ***";

//printing the satellite image number:

for(int ic=0; ic<=15; ic=ic+15){
                                  final                          <<
"\n*******************************************************************
********\n"
      << "\nCorrections of the exterior orientation parameters"
        << " \n          for the satellite image number" << ic/15+1 << "\n\n";

//printing the corrections for the exterior orientation parameters:

     final << "DF0 = " << mat_x1[ic] << "\t"
          << "D[common-omega0] = " << mat_x1[ic+1]  << "\n"
```

```
          << "Di = " << mat_x1[ic+2]  << "\t"
          << "Da = " << mat_x1[ic+3]<< "\n\n\n"
          << "DF(1) = " << mat_x1[ic+4]<< "\t"
          << "D[common-omega](1) = " << mat_x1[ic+5]  << "\n"
          << "Domega(0) = " << mat_x1[ic+6] << "\t"
          << "Dphi(0) = " << mat_x1[ic+7]<< "\t"
          << "Dkappa(0) = " << mat_x1[ic+8]  << "\n";

    final << "Domega(1) = " << mat_x1[ic+9]<< "\t"
          << "Dphi(1) = " << mat_x1[ic+10]  << "\t"
          << "Dkapa(1) = " << mat_x1[ic+11] << "\n"
          << "Domega(2) = " << mat_x1[ic+12]<< "\t"
          << "Dphi(2) = " << mat_x1[ic+13]  << "\t"
          << "Dkapa(2) = " << mat_x1[ic+14] << "\n";


    final << "\n--------------------------------\n";
  }

//printing the corrections for the GCPs:

final << "\n*****************************************\n"
    << "\ncorrections for the GCPs are as follows:\n"
    << "\n*****************************************\n\n";

for(int ig=1; ig<=ngp; ++ig){
  int ikg=(ig-1)*3+30;
  final << "DXGCP(" << ig << ")=" << mat_x1[ikg]<< "\t"
        << "DYGCP(" << ig << ")=" << mat_x1[ikg+1]  << "\t"
        << "DZGCP(" << ig << ")=" << mat_x1[ikg+2]<< "\n\n";
}
final << "\n************************************************************\n";

//up date the values. First GCPs:

for(int i9=1; i9<=ngp; ++i9){
  for(int j9=1; j9<=3; ++j9){
    int ku=(i9-1)*3;
    ground_xyz[i9-1][j9]=ground_xyz[i9-1][j9]+mat_x1[ku+j9-1+30];
  }
 }


//up date the exterior orientation parameters:

    for(int je=0; je<5; ++je){
      for(int ie=0;ie<3;++ie){
          imago1[je][ie]=imago1[je][ie]+mat_x1[ie+je*3];
          imago2[je][ie]=imago2[je][ie]+mat_x1[ie+je*3+15];}}
```

```
} //close the iteration acolad.

final << "\n                    ***********                    \n";
final << "\n          ********************************          \n";
final << "\n**************************************************************\n";

final << "---------The final result for the first image----------\n\n\n";
final << "True anomaly = " << imago1[0][0] << "\n"
    << "First rate of the true anomaly = " << imago1[1][1] << "\n"
    << "Right ascension of the ascending node = " << imago1[0][1] << "\n"
    << "First rate of the right ascension of the ascending node = " << imago1[1][2] << "\n"
    << "Inclination = " << imago1[0][2] << "\n"
    << "Semi major axis of the orbit = " << imago1[1][0] << "\n"
    << "Omega = " << imago1[2][0] << "\t"
    << "First rate of the omega = " << imago1[3][0] << "\t"
    << "Second rate of the omega = " << imago1[4][0] << "\n";
 final<<  "Phi = " << imago1[2][1] << "\t"
    << "First rate of the Phi = " << imago1[3][1] << "\t"
    << "Second rate of the Phi = " << imago1[4][1] << "\n"
    << "Kappa = " << imago1[2][2] << "\t"
    << "First rate of the kappa = " << imago1[3][2] << "\t"
    << "Second rate of the kappa = " << imago1[4][2] << "\n";
final << "\n----------The final result for the second image----------\n\n";

final << "True anomaly = " << imago2[0][0] << "\n"
    << "First rate of the true anomaly = " << imago2[1][1] << "\n"
    << "Right ascension of the ascending node = " << imago2[0][1] << "\n"
    << "First rate of the right ascension of the ascending node = " << imago2[1][2] << "\n"
    << "Inclination = " << imago2[0][2] << "\n"
    << "Semi major axis of the orbit = " << imago2[1][0] << "\n"
    << "Omega = " << imago2[2][0] << "\t"
    << "First rate of the omega = " << imago2[3][0] << "\t"
    << "Second rate of the omega = " << imago2[4][0] << "\n";
 final<< "Phi = " << imago2[2][1] << "\t"
    << "First rate of the Phi = " << imago2[3][1] << "\t"
    << "Second rate of the Phi = " << imago2[4][1] << "\n"
    << "Kappa = " << imago2[2][2] << "\t"
    << "First rate of the kappa = " << imago2[3][2] << "\t"
    << "Second rate of the kappa = " << imago2[4][2] << "\n";

final << "\n----------The final result for the GCPs----------\n\n";
for(i=0;i<ngp;++i){
  final << "X("<< i+1 << ")=" << ground_xyz[i][1] << "\t"
        << "Y("<< i+1 << ")=" << ground_xyz[i][2] << "\t"
        << "Z("<< i+1 << ")=" << ground_xyz[i][3] << "\n";
}

final << "\n**************************************************************\n";
final << "\n          ********************************          \n";
```

```
final << "\n                    ***********                    \n";


for(int ii=1; ii<=ngp+ncp; ++ii){   //open the acolad number 0.
 for(int id=1; id<=NPO; ++id){//open the acolad number 1.
  if(id==2){
    int np=np2+ncp;
    for(int i9=1; i9<=np; ++i9)
     for(int j9=1; j9<=4; ++j9)
          image_xy[i9-1][j9-1]=image2[i9-1][j9-1];

     for(i9=1; i9<=np; ++i9)
      for(int j9=1; j9<=2; ++j9)
          pixel_xy[i9-1][j9-1]=pixel2_xy[i9-1][j9-1];


     for(int io=0; io<5; ++io) {
      for(int jo=0; jo<3; ++jo){
        image_eo[io][jo]=imago2[io][jo];}}

        a_of_p=a_of_p2;  // argument of perigee
        e=e2;       // eccentricity
        view_angle = angle_r;
    if(image_case==3){
     size=0.010;
     pixelc=4060;
     pixelr=2900;
    }

    else if(image_case==4){
     size=0.010;
     pixelc=4060;
     pixelr=1488.5;
    }

    else{
     size=0.013;
     pixelc=3000;
     pixelr=3000;
    }
//  focal=focal_f/1000.0;
  }

  else if(id==1){
int np=np1+ncp;
for(int i=0;i<np;++i) {
 for(int j=0;j<4;++j){
   image_xy[i][j]=image1[i][j];}}
```

```
for(i=0;i<np;++i)
 for(int j=0;j<2;++j)
  pixel_xy[i][j]=pixel1_xy[i][j];

  if(image_case==3){
   size=0.010;
   pixelc=4060;
   pixelr=2900;
  }

  else if(image_case==4){
   size=0.010;
   pixelc=4060;
   pixelr=1488.5;
  }

  else{
   size=0.013;
   pixelc=3000;
   pixelr=3000;
  }

   for(int io=0; io<5; ++io){
    for(int jo=0; jo<3; ++jo){
      image_eo[io][jo]=imago1[io][jo];}}

      a_of_p=a_of_p1;  // argument of perigee
    e=e1;       // eccentricity
      view_angle = angle_l;}

      be_bg(ground_xyz[ii-1][1],ground_xyz[ii-1][2],ground_xyz[ii-1][3],
          pixel_xy[ii-1][0],pixel_xy[ii-1][1],image_xy[ii-1][3],
          a_of_p,image_eo[0][2],1,size);

    if(id==1){
       XL=X_prj;
       YL=Y_prj;
       ZL=Z_prj;
       UL=UU;
       VL=VV;
       WL=WW;}
    else{
     XR=X_prj;
       YR=Y_prj;
       ZR=Z_prj;
       UR=UU;
       VR=VV;
       WR=WW;}
  } // close acolad number 1;
```

```
 landa_r[ii-1]=((XR-XL)*VL-(YR-YL)*UL)/(VR*UL-UR*VL);
 XCP[ii-1]=XR+landa_r[ii-1]*UR;
 YCP[ii-1]=YR+landa_r[ii-1]*VR;
 ZCP[ii-1]=ZR+landa_r[ii-1]*WR;

 DXX[ii-1]=XCP[ii-1]-ground_xyz[ii-1][1];
 DYY[ii-1]=YCP[ii-1]-ground_xyz[ii-1][2];
 DZZ[ii-1]=ZCP[ii-1]-ground_xyz[ii-1][3];

} // close acolad number 0;

// final << ground_xyz[ii-1][1] << "\t" << ground_xyz[ii-1][2] << "\t"
//      << ground_xyz[ii-1][3] << "\n";

double sum1=0.0;
double sum2=0.0;
double sum3=0.0;
final << "\n************************************************************\n";
final << "************************************************************\n";
final << "\nThe residuals for the Ground Control Points are:\n";
for( i=0; i<ngp; ++i){
 final << "\nFor the GCP number " << i+1 << "\n";
 final << "Difference in X = " << DXX[i] << "\n"
     << "Difference in Y = " << DYY[i] << "\n"
     << "Difference in Z = " << DZZ[i] << "\n";
     sum1=sum1+pow(DXX[i],2);
     sum2=sum2+pow(DYY[i],2);
     sum3=sum3+pow(DZZ[i],2);
}
final << "\n************************************************************\n";
final << "\nThe RMSE in X, Y and Z for the GCPs are as follows:\n\n";
final << "RMSE(X)= " << sqrt(sum1/(ngp-1))<< "\t"
    << "RMSE(Y)= " << sqrt(sum2/(ngp-1))<< "\t"
    << "RMSE(Z)= " << sqrt(sum3/(ngp-1))<< "\n";
if(ncp>0.0){
sum1=0.0;
sum2=0.0;
sum3=0.0;
final << "\n************************************************************\n";
final << "************************************************************\n";
final << "\nThe residuals for the Check Points are:\n";
for(i=0; i<ncp; ++i){
 final << "\nFor the check point number " << i+1 << "\n";
 final << "Difference in X = " << DXX[ngp+i] << "\n"
     << "Difference in Y = " << DYY[ngp+i] << "\n"
     << "Difference in Z = " << DZZ[ngp+i] << "\n";
     sum1=sum1+pow(DXX[ngp+i],2);
     sum2=sum2+pow(DYY[ngp+i],2);
     sum3=sum3+pow(DZZ[ngp+i],2);
```

```
 }
final << "\n ***************************************************\n";
final << "\nThe RMSE in X, Y and Z for the check points are as follows:\n\n";
final << "RMSE(X)= " << sqrt(sum1/(ncp-1))<< "\t"
    << "RMSE(Y)= " << sqrt(sum2/(ncp-1))<< "\t"
    << "RMSE(Z)= " << sqrt(sum3/(ncp-1))<< "\n";
}

final.close();

delete pt_matrix_a;
delete pt_matrix_b;
delete pt_matrix_c;
delete BBAR_11;
delete BBAR_12;
delete BBAR_22;
delete BBAR_21;
delete Be;
delete Bg;
delete xyz_obs;
delete eop_obs;
delete eop_it;
delete image_xy;
delete image_eo;
delete pixel_xy;
delete EBAR1;
delete EBAR2;
delete EBARg;
delete EBARe;
delete normal_11;
delete normal_12;
delete normal_21;
delete normal_22;
delete weight;
delete mat_u1;
delete mat_u2;
delete mat_x1;
delete mat_x2;
delete BTWB1;
delete BTWB2;
delete BTWE1;
delete BTWE2;
delete BTWE3;
delete X0;
delete Y0;
delete Z0;
delete pt_vector_v;
delete pt_vector_x;
delete XCP;
```

```
delete YCP;
delete ZCP;
delete landa_r;
delete DXX;
delete DYY;
delete DZZ;
}


/**************************************************************************
        Derivations Sub-Module of Bundle Adjustment Module (Case 2)
**************************************************************************/

#include <math.h>
#include <fstream.h>
#include <memory.h>
#include "pushbroom.h"

void be_bg(double ground_x,double ground_y,double ground_z,
         double pixel_x,double pixel_y,double YP,
         double a_omega,double inclin,int intersect,float size)
{
double huge (*RS)[3]=new double huge [3][3];
double huge (*RA)[3]=new double huge [3][3];
double huge (*RAS)[3]=new double huge [3][3];
double huge (*R)[3]=new double huge [3][3];

ofstream fout;
fout.open("bebg.out");
      fout << ground_x << "\t" << ground_y << "\t" << ground_z;
      fout << "\n" << pixel_x << "\t" << pixel_y << "\n"
         << pixelc << "\t" << pixelr << "\t" << YP << "\n";
      fout << view_angle << "\n" << focal << "\n" << e << "\n";

for(int ix=0;ix<5;++ix){
 for(int jx=0;jx<3;++jx){
  fout << image_eo[ix][jx] << "\t";}
 fout<< "\n";}
double DX;
double DY;
double DZ;
double F,r,C_OMEGA,omega,phi,kapa;
double m_param;
double n_param;
double q_param;
double CONST;
float XP;
XP=0.0;  //x-x0
```

```
    F=image_eo[0][0]+image_eo[1][1]*((pixel_x-pixelc)*size);
    C_OMEGA=image_eo[0][1]+image_eo[1][2]*((pixel_x-pixelc)*size);
    r=image_eo[1][0]*(1-pow(e,2))/(1+e*cos(F));
if(image_case==1|| image_case==3 || image_case==4){
    omega=image_eo[2][0]+image_eo[3][0]*((pixel_x-pixelc)*size)+
            image_eo[4][0]*pow(((pixel_x-pixelc)*size),2);
    phi=image_eo[2][1]+image_eo[3][1]*((pixel_x-pixelc)*size)+
            image_eo[4][1]*pow(((pixel_x-pixelc)*size),2);
}

else if(image_case==2){
    omega=image_eo[2][0]+image_eo[3][0]*((pixel_x-pixelc)*size)+
            image_eo[4][0]*pow(((pixel_y-pixelr)*size),2);
    phi=image_eo[2][1]+image_eo[3][1]*((pixel_x-pixelc)*size)+
            image_eo[4][1]*pow(((pixel_y-pixelr)*size),2);
}

    kapa=image_eo[2][2]+image_eo[3][2]*((pixel_x-pixelc)*size)+
            image_eo[4][2]*pow(((pixel_x-pixelc)*size),2);


    //the element of matrix RS:
    RS[0][0]=-sin(F+a_omega)*cos(C_OMEGA)-
            cos(F+a_omega)*cos(inclin)*sin(C_OMEGA);
    RS[0][1]=-sin(F+a_omega)*sin(C_OMEGA)+
            cos(F+a_omega)*cos(inclin)*cos(C_OMEGA);
    RS[0][2]=cos(F+a_omega)*sin(inclin);
    RS[1][0]=sin(C_OMEGA)*sin(inclin);
    RS[1][1]=-cos(C_OMEGA)*sin(inclin);
    RS[1][2]=cos(inclin);
    RS[2][0]=cos(F+a_omega)*cos(C_OMEGA)-
            sin(F+a_omega)*cos(inclin)*sin(C_OMEGA);
    RS[2][1]=cos(F+a_omega)*sin(C_OMEGA)+
            sin(F+a_omega)*cos(inclin)*cos(C_OMEGA);

    RS[2][2]=sin(F+a_omega)*sin(inclin);

    //compute the differences between the coordinates of the GCPs and
    //the coordinates of the projection centres of each line related to that GCP:
    X_prj=r*RS[2][0];
    Y_prj=r*RS[2][1];
    Z_prj=r*RS[2][2];

    //Forming the rotational mtrix R composing of the rotational
    //elements related to each line:

    RA[0][0]=cos(phi)*cos(kapa);
    RA[0][1]=cos(omega)*sin(kapa)+
            sin(omega)*sin(phi)*cos(kapa);
```

```
    RA[0][2]=sin(omega)*sin(kapa)-
            cos(omega)*sin(phi)*cos(kapa);
    RA[1][0]=(-cos(phi)*sin(kapa));
    RA[1][1]=cos(omega)*cos(kapa)-
            sin(omega)*sin(phi)*sin(kapa);
    RA[1][2]=sin(omega)*cos(kapa)+
             cos(omega)*sin(phi)*sin(kapa);
    RA[2][0]=sin(phi);
    RA[2][1]=(-sin(omega)*cos(phi));
    RA[2][2]= cos(omega)*cos(phi);


  RAS[0][0]= RA[0][0]*RS[0][0]+RA[0][1]*RS[1][0]+RA[0][2]*RS[2][0];
  RAS[0][1]= RA[0][0]*RS[0][1]+RA[0][1]*RS[1][1]+RA[0][2]*RS[2][1];
  RAS[0][2]= RA[0][0]*RS[0][2]+RA[0][1]*RS[1][2]+RA[0][2]*RS[2][2];
  RAS[1][0]= RA[1][0]*RS[0][0]+RA[1][1]*RS[1][0]+RA[1][2]*RS[2][0];
  RAS[1][1]= RA[1][0]*RS[0][1]+RA[1][1]*RS[1][1]+RA[1][2]*RS[2][1];
  RAS[1][2]= RA[1][0]*RS[0][2]+RA[1][1]*RS[1][2]+RA[1][2]*RS[2][2];
  RAS[2][0]= RA[2][0]*RS[0][0]+RA[2][1]*RS[1][0]+RA[2][2]*RS[2][0];
  RAS[2][1]= RA[2][0]*RS[0][1]+RA[2][1]*RS[1][1]+RA[2][2]*RS[2][1];
  RAS[2][2]= RA[2][0]*RS[0][2]+RA[2][1]*RS[1][2]+RA[2][2]*RS[2][2];

  // initialising the view angle in the case of SPOT

  R[0][0]= RAS[0][0];
  R[0][1]= RAS[0][1];
  R[0][2]= RAS[0][2];
  R[1][0]= RAS[1][0]* cos(view_angle) + RAS[2][0]* sin(view_angle);
  R[1][1]= RAS[1][1]* cos(view_angle) + RAS[2][1]* sin(view_angle);
  R[1][2]= RAS[1][2]* cos(view_angle) + RAS[2][2]* sin(view_angle);
  R[2][0]= RAS[2][0]* cos(view_angle) - RAS[1][0]* sin(view_angle);
  R[2][1]= RAS[2][1]* cos(view_angle) - RAS[1][1]* sin(view_angle);
  R[2][2]= RAS[2][2]* cos(view_angle) - RAS[1][2]* sin(view_angle);

if(intersect==1){
    UU=R[1][0]*YP+R[2][0]*(-focal);
    VV=R[1][1]*YP+R[2][1]*(-focal);
    WW=R[1][2]*YP+R[2][2]*(-focal);
}

if(intersect==0){
     DX=ground_x-X_prj;
       DY=ground_y-Y_prj;
       DZ=ground_z-Z_prj;

    //computing the parametrs:m_param, n_param, q_param:

    m_param=R[0][0]*DX+R[0][1]*DY+R[0][2]*DZ;
    n_param=R[1][0]*DX+R[1][1]*DY+R[1][2]*DZ;
```

```
    q_param=R[2][0]*DX+R[2][1]*DY+R[2][2]*DZ;


    //computing the Cx=x-x0 and Cy=y-y0 by calculation. This will
    //be later compared with these values that have been got by
    //observations.

    Cx=-focal*m_param/q_param;
    Cy=-focal*n_param/q_param;

    CONST=focal/pow(q_param,2);



 //computing the elements of matrix Be

Be[0][0]=CONST*
     (q_param*((-RA[0][0]*RS[2][0]+RA[0][2]*RS[0][0])*DX+
             (-RA[0][0]*RS[2][1]+RA[0][2]*RS[0][1])*DY+
             (-RA[0][0]*RS[2][2]+RA[0][2]*RS[0][2])*DZ-
              r*e*sin(F)/(1+e*cos(F))*(R[0][0]*RS[2][0]+
                R[0][1]*RS[2][1]+R[0][2]*RS[2][2])-
                r*(R[0][0]*RS[0][0]+R[0][1]*RS[0][1]+R[0][2]*RS[0][2]))-
        m_param*(((RA[1][0]*RS[2][0]-RA[1][2]*RS[0][0])*sin(view_angle)+
                (-RA[2][0]*RS[2][0]+RA[2][2]*RS[0][0])*cos(view_angle))*DX+
               ((RA[1][0]*RS[2][1]-RA[1][2]*RS[0][1])*sin(view_angle)+
                (-RA[2][0]*RS[2][1]+RA[2][2]*RS[0][1])*cos(view_angle))*DY+
               ((RA[1][0]*RS[2][2]-RA[1][2]*RS[0][2])*sin(view_angle)+
                (-RA[2][0]*RS[2][2]+RA[2][2]*RS[0][2])*cos(view_angle))*DZ-

r*e*sin(F)/(1+e*cos(F))*(R[2][0]*RS[2][0]+R[2][1]*RS[2][1]+R[2][2]*RS[2][2])-
                 r*(R[2][0]*RS[0][0]+R[2][1]*RS[0][1]+R[2][2]*RS[0][2])));


Be[0][1]=CONST*(q_param*(-R[0][1]*DX+R[0][0]*DY+r*(RS[2][1]*R[0][0]-RS[2][
0]*R[0][1]))-

m_param*(-R[2][1]*DX+R[2][0]*DY+r*(RS[2][1]*R[2][0]-RS[2][0]*R[2][1])));

     double pqr=q_param*(
                  (cos(F+a_omega)*sin(inclin)*sin(C_OMEGA)*RA[0][0]+
                         cos(inclin)*sin(C_OMEGA)*RA[0][1]+
                         sin(inclin)*sin(C_OMEGA)*sin(F+a_omega)*RA[0][2])*DX+
                        (-cos(F+a_omega)*sin(inclin)*cos(C_OMEGA)*RA[0][0]+
                         (-cos(inclin)*cos(C_OMEGA))*RA[0][1]+
                         (-sin(inclin)*cos(C_OMEGA)*sin(F+a_omega))*RA[0][2])*DY+
                        (cos(F+a_omega)*cos(inclin)*RA[0][0]+
```

```
(-sin(inclin))*RA[0][1]+cos(inclin)*sin(F+a_omega)*RA[0][2])*DZ+

r*sin(F+a_omega)*(-sin(C_OMEGA)*sin(inclin)+cos(C_OMEGA)*sin(inclin)-cos(inclin)));
    double mno=-m_param*(
                    ((-cos(F+a_omega)*sin(inclin)*sin(C_OMEGA)*RA[1][0]-
                        cos(inclin)*sin(C_OMEGA)*RA[1][1]-

sin(inclin)*sin(C_OMEGA)*sin(F+a_omega)*RA[1][2])*sin(view_angle)+
                    (cos(F+a_omega)*sin(inclin)*sin(C_OMEGA)*RA[2][0]+
                     cos(inclin)*sin(C_OMEGA)*RA[2][1]+

sin(inclin)*sin(C_OMEGA)*sin(F+a_omega)*RA[2][2])*cos(view_angle))*DX+
                    ((cos(F+a_omega)*sin(inclin)*cos(C_OMEGA)*RA[1][0]+
                        cos(inclin)*cos(C_OMEGA)*RA[1][1]+

sin(inclin)*cos(C_OMEGA)*sin(F+a_omega)*RA[1][2])*sin(view_angle)+
                     (-cos(F+a_omega)*sin(inclin)*cos(C_OMEGA)*RA[2][0]-
                        cos(inclin)*cos(C_OMEGA)*RA[2][1]-

sin(inclin)*cos(C_OMEGA)*sin(F+a_omega)*RA[2][2])*cos(view_angle))*DY+
                    ((-cos(F+a_omega)*cos(inclin)*RA[1][0]+
                        sin(inclin)*RA[1][1]-
                        cos(inclin)*sin(F+a_omega)*RA[1][2])*sin(view_angle)+
                     (cos(F+a_omega)*cos(inclin)*RA[2][0]-
                        sin(inclin)*RA[2][1]+

cos(inclin)*sin(F+a_omega)*RA[2][2])*cos(view_angle))*DZ-
                    r*sin(F+a_omega)*(sin(C_OMEGA)*sin(inclin)*R[2][0]-

cos(C_OMEGA)*sin(inclin)*R[2][1]+cos(inclin)*R[2][2]));
     Be[0][2]=CONST*(pqr+mno);
     Be[0][3]=CONST*(pow(e,2)-1)/(1+e*cos(F))*
            (q_param*(RS[2][0]*R[0][0]+RS[2][1]*R[0][1]+RS[2][2]*R[0][2])-

m_param*(RS[2][0]*R[2][0]+RS[2][1]*R[2][1]+RS[2][2]*R[2][2]));
     fout << "Be(x/a)= " << Be[0][3] << "\n";
     Be[0][6]=CONST*(
          q_param*(
                  (-RS[1][0]*RA[0][2]+RS[2][0]*RA[0][1])*DX+
                  (-RS[1][1]*RA[0][2]+RS[2][1]*RA[0][1])*DY+
                  (-RS[1][2]*RA[0][2]+RS[2][2]*RA[0][1])*DZ)-
            m_param*(
                  ((RS[1][0]*RA[1][2]-RS[2][0]*RA[1][1])*sin(view_angle)+
                  (-RS[1][0]*RA[2][2]+RS[2][0]*RA[2][1])*cos(view_angle))*DX+
                  ((RS[1][1]*RA[1][2]-RS[2][1]*RA[1][1])*sin(view_angle)+
                  (-RS[1][1]*RA[2][2]+RS[2][1]*RA[2][1])*cos(view_angle))*DY+
                  ((RS[1][2]*RA[1][2]-RS[2][2]*RA[1][1])*sin(view_angle)+

(-RS[1][2]*RA[2][2]+RS[2][2]*RA[2][1])*cos(view_angle))*DZ));
```

```
Be[0][7]=CONST*(
        q_param*(
          (RS[0][0]*(-sin(phi)*cos(kapa))+
           RS[1][0]*(sin(omega)*cos(phi)*cos(kapa))+
           RS[2][0]*(-cos(omega)*cos(phi)*cos(kapa)))*DX+
          (RS[0][1]*(-sin(phi)*cos(kapa))+
           RS[1][1]*(sin(omega)*cos(phi)*cos(kapa))+
           RS[2][1]*(-cos(omega)*cos(phi)*cos(kapa)))*DY+
          (RS[0][2]*(-sin(phi)*cos(kapa))+
           RS[1][2]*(sin(omega)*cos(phi)*cos(kapa))+
           RS[2][2]*(-cos(omega)*cos(phi)*cos(kapa)))*DZ)-
        m_param*(
         ((-RS[0][0]*(sin(phi)*sin(kapa))-
           RS[1][0]*(-sin(omega)*cos(phi)*sin(kapa))-
           RS[2][0]*(cos(omega)*cos(phi)*sin(kapa)))*sin(view_angle)+
          (RS[0][0]*(cos(phi))+
           RS[1][0]*(sin(omega)*sin(phi))+
           RS[2][0]*(-cos(omega)*sin(phi)))*cos(view_angle))*DX+
         ((-RS[0][1]*(sin(phi)*sin(kapa))-
           RS[1][1]*(-sin(omega)*cos(phi)*sin(kapa))-
           RS[2][1]*(cos(omega)*cos(phi)*sin(kapa)))*sin(view_angle)+
          (RS[0][1]*(cos(phi))+
           RS[1][1]*(sin(omega)*sin(phi))+
           RS[2][1]*(-cos(omega)*sin(phi)))*cos(view_angle))*DY+
         ((-RS[0][2]*(sin(phi)*sin(kapa))-
           RS[1][2]*(-sin(omega)*cos(phi)*sin(kapa))-
           RS[2][2]*(cos(omega)*cos(phi)*sin(kapa)))*sin(view_angle)+
          (RS[0][2]*(cos(phi))+
           RS[1][2]*(sin(omega)*sin(phi))+
           RS[2][2]*(-cos(omega)*sin(phi)))*cos(view_angle))*DZ));
Be[0][8] =CONST*(
            q_param*(RAS[1][0]*DX+RAS[1][1]*DY+RAS[1][2]*DZ)-
        m_param*(m_param*sin(view_angle)));
Be[0][4] =((pixel_x-pixelc)*size)*Be[0][0];
Be[0][5] =((pixel_x-pixelc)*size)*Be[0][1];
Be[0][9] =((pixel_x-pixelc)*size)*Be[0][6];
Be[0][10]=((pixel_x-pixelc)*size)*Be[0][7];
Be[0][11]=((pixel_x-pixelc)*size)*Be[0][8];
if(image_case==1 || image_case==3 || image_case==4){
Be[0][12]=pow(((pixel_x-pixelc)*size),2)*Be[0][6];
Be[0][13]=pow(((pixel_x-pixelc)*size),2)*Be[0][7]; }
else if(image_case==2){
Be[0][12]=pow(((pixel_y-pixelr)*size),2)*Be[0][6];
Be[0][13]=pow(((pixel_y-pixelr)*size),2)*Be[0][7]; }

Be[0][14]=pow(((pixel_x-pixelc)*size),2)*Be[0][8];

Be[1][0]=CONST*
(q_param*(((-RA[1][0]*RS[2][0]+RA[1][2]*RS[0][0])*cos(view_angle)+
```

```
          (-RA[2][0]*RS[2][0]+RA[2][2]*RS[0][0])*sin(view_angle))*DX+
         ((-RA[1][0]*RS[2][1]+RA[1][2]*RS[0][1])*cos(view_angle)+
          (-RA[2][0]*RS[2][1]+RA[2][2]*RS[0][1])*sin(view_angle))*DY+
         ((-RA[1][0]*RS[2][2]+RA[1][2]*RS[0][2])*cos(view_angle)+
          (-RA[2][0]*RS[2][2]+RA[2][2]*RS[0][2])*sin(view_angle))*DZ-
       r*e*sin(F)/(1+e*cos(F))*(R[1][0]*RS[2][0]+
          R[1][1]*RS[2][1]+R[1][2]*RS[2][2])-
          r*(R[1][0]*RS[0][0]+R[1][1]*RS[0][1]+R[1][2]*RS[0][2]))-
  n_param*(((RA[1][0]*RS[2][0]-RA[1][2]*RS[0][0])*sin(view_angle)+
          (-RA[2][0]*RS[2][0]+RA[2][2]*RS[0][0])*cos(view_angle))*DX+
          ((RA[1][0]*RS[2][1]-RA[1][2]*RS[0][1])*sin(view_angle)+
          (-RA[2][0]*RS[2][1]+RA[2][2]*RS[0][1])*cos(view_angle))*DY+
          ((RA[1][0]*RS[2][2]-RA[1][2]*RS[0][2])*sin(view_angle)+
          (-RA[2][0]*RS[2][2]+RA[2][2]*RS[0][2])*cos(view_angle))*DZ-

r*e*sin(F)/(1+e*cos(F))*(R[2][0]*RS[2][0]+R[2][1]*RS[2][1]+R[2][2]*RS[2][2])-
          r*(R[2][0]*RS[0][0]+R[2][1]*RS[0][1]+R[2][2]*RS[0][2])));


Be[1][1]=CONST*(q_param*(-R[1][1]*DX+R[1][0]*DY+r*(RS[2][1]*R[1][0]-RS[2][
0]*R[1][1]))-

n_param*(-R[2][1]*DX+R[2][0]*DY+r*(RS[2][1]*R[2][0]-RS[2][0]*R[2][1])));
    //Be[1][2]=CONST*(
    pqr=q_param*(
           ((cos(F+a_omega)*sin(inclin)*sin(C_OMEGA)*RA[1][0]+
            cos(inclin)*sin(C_OMEGA)*RA[1][1]+

sin(inclin)*sin(C_OMEGA)*sin(F+a_omega)*RA[1][2])*cos(view_angle)+
            (cos(F+a_omega)*sin(inclin)*sin(C_OMEGA)*RA[2][0]+
            cos(inclin)*sin(C_OMEGA)*RA[2][1]+

sin(inclin)*sin(C_OMEGA)*sin(F+a_omega)*RA[2][2])*sin(view_angle))*DX+
           ((-cos(F+a_omega)*sin(inclin)*cos(C_OMEGA)*RA[1][0]-
            cos(inclin)*cos(C_OMEGA)*RA[1][1]-

sin(inclin)*cos(C_OMEGA)*sin(F+a_omega)*RA[1][2])*cos(view_angle)+
            (-cos(F+a_omega)*sin(inclin)*cos(C_OMEGA)*RA[2][0]-
            cos(inclin)*cos(C_OMEGA)*RA[2][1]-

sin(inclin)*cos(C_OMEGA)*sin(F+a_omega)*RA[2][2])*sin(view_angle))*DY+
           ((cos(F+a_omega)*cos(inclin)*RA[1][0]-
            sin(inclin)*RA[1][1]+
            cos(inclin)*sin(F+a_omega)*RA[1][2])*cos(view_angle)+
            (cos(F+a_omega)*cos(inclin)*RA[2][0]-
            sin(inclin)*RA[2][1]+
            cos(inclin)*sin(F+a_omega)*RA[2][2])*sin(view_angle))*DZ-

r*sin(F+a_omega)*(sin(C_OMEGA)*sin(inclin)*R[1][0]-cos(C_OMEGA)*sin(inclin)*R
```

```
[1][1]+cos(inclin)*R[1][2]));
    mno=n_param*(
                ((-cos(F+a_omega)*sin(inclin)*sin(C_OMEGA)*RA[1][0]-
                 cos(inclin)*sin(C_OMEGA)*RA[1][1]-

sin(inclin)*sin(C_OMEGA)*sin(F+a_omega)*RA[1][2])*sin(view_angle)+
                (cos(F+a_omega)*sin(inclin)*sin(C_OMEGA)*RA[2][0]+
                 cos(inclin)*sin(C_OMEGA)*RA[2][1]+

sin(inclin)*sin(C_OMEGA)*sin(F+a_omega)*RA[2][2])*cos(view_angle))*DX+
                ((cos(F+a_omega)*sin(inclin)*cos(C_OMEGA)*RA[1][0]+
                 cos(inclin)*cos(C_OMEGA)*RA[1][1]+

sin(inclin)*cos(C_OMEGA)*sin(F+a_omega)*RA[1][2])*sin(view_angle)+
                (-cos(F+a_omega)*sin(inclin)*cos(C_OMEGA)*RA[2][0]-
                 cos(inclin)*cos(C_OMEGA)*RA[2][1]-

sin(inclin)*cos(C_OMEGA)*sin(F+a_omega)*RA[2][2])*cos(view_angle))*DY+
                ((-cos(F+a_omega)*cos(inclin)*RA[1][0]+
                 sin(inclin)*RA[1][1]-
                 cos(inclin)*sin(F+a_omega)*RA[1][2])*sin(view_angle)+
                (cos(F+a_omega)*cos(inclin)*RA[2][0]-
                 sin(inclin)*RA[2][1]+
                 cos(inclin)*sin(F+a_omega)*RA[2][2])*cos(view_angle))*DZ-

r*sin(F+a_omega)*(sin(C_OMEGA)*sin(inclin)*R[2][0]-cos(C_OMEGA)*sin(inclin)*R
[2][1]+cos(inclin)*R[2][2]));
    Be[1][2]=CONST*(pqr-mno);
    Be[1][3]=CONST*(pow(e,2)-1)/(1+e*cos(F))*
         (q_param*(RS[2][0]*R[1][0]+RS[2][1]*R[1][1]+RS[2][2]*R[1][2])-
            n_param*(RS[2][0]*R[2][0]+RS[2][1]*R[2][1]+RS[2][2]*R[2][2]));

    Be[1][6]=CONST*(
         q_param*(
                 ((-RS[1][0]*RA[1][2]+RS[2][0]*RA[1][1])*cos(view_angle)+
                  (-RS[1][0]*RA[2][2]+RS[2][0]*RA[2][1])*sin(view_angle))*DX+
                 ((-RS[1][1]*RA[1][2]+RS[2][1]*RA[1][1])*cos(view_angle)+
                  (-RS[1][1]*RA[2][2]+RS[2][1]*RA[2][1])*sin(view_angle))*DY+
                 ((-RS[1][2]*RA[1][2]+RS[2][2]*RA[1][1])*cos(view_angle)+
                  (-RS[1][2]*RA[2][2]+RS[2][2]*RA[2][1])*sin(view_angle))*DZ)-
           n_param*(
                 ((RS[1][0]*RA[1][2]-RS[2][0]*RA[1][1])*sin(view_angle)+
                  (-RS[1][0]*RA[2][2]+RS[2][0]*RA[2][1])*cos(view_angle))*DX+
                 ((RS[1][1]*RA[1][2]-RS[2][1]*RA[1][1])*sin(view_angle)+
                  (-RS[1][1]*RA[2][2]+RS[2][1]*RA[2][1])*cos(view_angle))*DY+
                 ((RS[1][2]*RA[1][2]-RS[2][2]*RA[1][1])*sin(view_angle)+

(-RS[1][2]*RA[2][2]+RS[2][2]*RA[2][1])*cos(view_angle))*DZ));
    Be[1][7]=CONST*(
```

```
                q_param*(
                  ((RS[0][0]*(sin(phi)*sin(kapa))+
                    RS[1][0]*(-sin(omega)*cos(phi)*sin(kapa))+
                    RS[2][0]*(cos(omega)*cos(phi)*sin(kapa)))*cos(view_angle)+
                   (RS[0][0]*(cos(phi))+
                    RS[1][0]*(sin(omega)*sin(phi))+
                    RS[2][0]*(-cos(omega)*sin(phi)))*sin(view_angle))*DX+
                  ((RS[0][1]*(sin(phi)*sin(kapa))+
                    RS[1][1]*(-sin(omega)*cos(phi)*sin(kapa))+
                    RS[2][1]*(cos(omega)*cos(phi)*sin(kapa)))*cos(view_angle)+
                   (RS[0][1]*(cos(phi))+
                    RS[1][1]*(sin(omega)*sin(phi))+
                    RS[2][1]*(-cos(omega)*sin(phi)))*sin(view_angle))*DY+
                  ((RS[0][2]*(sin(phi)*sin(kapa))+
                    RS[1][2]*(-sin(omega)*cos(phi)*sin(kapa))+
                    RS[2][2]*(cos(omega)*cos(phi)*sin(kapa)))*cos(view_angle)+
                   (RS[0][2]*(cos(phi))+
                    RS[1][2]*(sin(omega)*sin(phi))+
                    RS[2][2]*(-cos(omega)*sin(phi)))*sin(view_angle))*DZ)-
                n_param*(
                  ((-RS[0][0]*(sin(phi)*sin(kapa))-
                    RS[1][0]*(-sin(omega)*cos(phi)*sin(kapa))-
                    RS[2][0]*(cos(omega)*cos(phi)*sin(kapa)))*sin(view_angle)+
                   (RS[0][0]*(cos(phi))+
                    RS[1][0]*(sin(omega)*sin(phi))+
                    RS[2][0]*(-cos(omega)*sin(phi)))*cos(view_angle))*DX+
                  ((-RS[0][1]*(sin(phi)*sin(kapa))-
                    RS[1][1]*(-sin(omega)*cos(phi)*sin(kapa))-
                    RS[2][1]*(cos(omega)*cos(phi)*sin(kapa)))*sin(view_angle)+
                   (RS[0][1]*(cos(phi))+
                    RS[1][1]*(sin(omega)*sin(phi))+
                    RS[2][1]*(-cos(omega)*sin(phi)))*cos(view_angle))*DY+
                  ((-RS[0][2]*(sin(phi)*sin(kapa))-
                    RS[1][2]*(-sin(omega)*cos(phi)*sin(kapa))-
                    RS[2][2]*(cos(omega)*cos(phi)*sin(kapa)))*sin(view_angle)+
                   (RS[0][2]*(cos(phi))+
                    RS[1][2]*(sin(omega)*sin(phi))+
                    RS[2][2]*(-cos(omega)*sin(phi)))*cos(view_angle))*DZ));
Be[1][8] =CONST*(
                    q_param*(-m_param*cos(view_angle))-
                    n_param*(m_param*sin(view_angle)));

Be[1][4] =((pixel_x-pixelc)*size)*Be[1][0];
Be[1][5] =((pixel_x-pixelc)*size)*Be[1][1];
Be[1][9] =((pixel_x-pixelc)*size)*Be[1][6];
Be[1][10]=((pixel_x-pixelc)*size)*Be[1][7];
Be[1][11]=((pixel_x-pixelc)*size)*Be[1][8];

if(image_case==1 || image_case==3 || image_case==4){
```

```
    Be[1][12]=pow(((pixel_x-pixelc)*size),2)*Be[1][6];
    Be[1][13]=pow(((pixel_x-pixelc)*size),2)*Be[1][7];}
    else if(image_case==2){
    Be[1][12]=pow(((pixel_y-pixelr)*size),2)*Be[1][6];
    Be[1][13]=pow(((pixel_y-pixelr)*size),2)*Be[1][7];}

    Be[1][14]=pow(((pixel_x-pixelc)*size),2)*Be[1][8];


    //Computing the elements of matrix Bg:

     for(int jb=0;jb<3;++jb){
          Bg[0][jb]=CONST*(q_param*R[0][jb]-m_param*R[2][jb]);
          fout << Bg[0][jb] << "\t";
          Bg[1][jb]=CONST*(q_param*R[1][jb]-n_param*R[2][jb]);
        fout << Bg[1][jb] << "\n";
        }
} // this is related to if.

fout.close();
delete R;
delete RA;
delete RS;
delete RAS;
}
```

****

INPUT DATA FOR POLYNOMIAL ADJUSTMENT PROGRAM

Format of the input is as follows:
number of control points   number of check points
point number   x (image coordinate) y(image coordinate)   Easting   Northing

Note: These are input coordinates for MOMS-02 mode 1, channel 6, in the case of the ETH data set.
****

18	33			
112	1586.375	5641.625	7625411.337	800117.134
205	694.825	5083.375	7616848.978	788165.707
200	1017.125	2646.125	7633107.009	758149.086
135	2154.400	7410.100	7623855.749	826157.624
61	2671.625	2316.625	7655844.367	762559.537
12	2216.333	584.867	7658672.510	737657.369
122	629.000	6353.900	7609684.659	804366.794
128	2765.667	6663.800	7635408.030	819669.727
45	1796.000	1557.667	7648463.359	748101.161
220	1555.175	477.625	7650783.426	732778.862
106	2704.875	5098.625	7642410.310	798929.138
201	1863.125	3336.625	7640443.041	771561.814
23	1012.967	1138.500	7640582.196	738519.036
83	2797.375	3822.925	7649936.418	782817.346
89	660.375	4303.375	7620294.079	777849.735
93	1913.375	4644.375	7634560.069	788846.277
116	2030.900	6277.900	7627921.267	810732.077
125	1374.125	6626.825	7617803.499	811829.604
28	1819.375	1111.375	7650978.533	742407.593
30	2066.375	1014.375	7654624.592	742447.255
32	2175.430	1415.650	7654009.100	748236.986
43	1231.000	1708.500	7640510.416	747081.425
49	2309.375	1822.125	7653687.246	754213.610
58	1606.875	2312.625	7642279.057	756901.066
60	2248.925	2738.625	7648346.646	765812.639
64	1152.375	3055.125	7632772.665	764169.198
66	1577.125	3405.125	7636447.905	770957.252
67	1668.667	3472.333	7637282.224	772302.599
69	2075.875	3016.875	7644742.318	768522.242
70	2174.400	3549.000	7643348.422	775966.204
71	2479.200	3316.000	7648402.084	774530.121
73	2723.250	2965.917	7653275.293	771274.167
76	985.625	3731.475	7627248.319	772153.101
82	2077.125	3916.625	7640274.007	780243.761
85	2398.225	3673.925	7645596.010	778761.542
86	2488.875	4204.875	7644094.765	786145.954
95	2560.875	4616.675	7642966.474	791896.408

96	2757.875	4601.625	7645573.823	792733.791
104	2012.625	5344.125	7632372.325	798474.224
105	2200.375	5573.125	7633600.520	802457.122
110	941.333	6021.000	7615310.641	801669.304
118	2682.125	5793.625	7638679.992	807864.600
119	2664.500	6033.667	7637243.299	810902.717
120	2734.333	6007.000	7638264.880	810920.334
121	2393.667	6216.667	7632874.207	811857.343
126	1430.933	6507.667	7619129.036	810566.842
127	2251.333	6669.400	7628789.427	817012.966
134	1804.000	7056.900	7621136.549	819701.421
206	890.125	5284.875	7618328.335	791805.774
207	1221.200	4603.700	7625943.791	784669.984
222	1335.625	829.925	7646222.292	736203.262

****

OUTPUT FOR POLYNOMIAL ADJUSTMENT PROGRAM

****

******** In the case of 3 parameters ********

RMSE in (m) for the E and N and in (pixel) for the x and y coordinates
for the control points are as follows:

RMSE_E = 17.776906 RMSE_x = 1.316808
RMSE_N = 11.97505 RMSE_y = 0.887041

RMSE in (m) for the E and N and in (pixel) for the x and y coordinates
for the check points are as follows:

RMSE_E = 19.99849 RMSE_x = 1.48137
RMSE_N = 14.668116 RMSE_y = 1.086527

****

******** In the case of 4 parameters ********

RMSE in (m) for the E and N and in (pixel) for the x and y coordinates
for the control points are as follows:

RMSE_E = 17.042744 RMSE_x = 1.262425
RMSE_N = 10.944792 RMSE_y = 0.810725

RMSE in (m) for the E and N and in (pixel) for the x and y coordinates
for the check points are as follows:

RMSE_E = 19.171641 RMSE_x = 1.420122
RMSE_N = 14.236745 RMSE_y = 1.054574

****

******** In the case of 5 parameters ********

RMSE in (m) for the E and N and in (pixel) for the x and y coordinates for the control points are as follows:

RMSE_E = 11.282815   RMSE_x = 0.835764
RMSE_N = 10.637042   RMSE_y = 0.787929

RMSE in (m) for the E and N and in (pixel) for the x and y coordinates for the check points are as follows:

RMSE_E = 13.452836   RMSE_x = 0.996506
RMSE_N = 13.789635   RMSE_y = 1.021454

****

******** In the case of 6 parameters ********

RMSE in (m) for the E and N and in (pixel) for the x and y coordinates for the control points are as follows:

RMSE_E = 8.840502   RMSE_x = 0.654852
RMSE_N = 5.627702   RMSE_y = 0.416867

RMSE in (m) for the E and N and in (pixel) for the x and y coordinates for the check points are as follows:

RMSE_E = 12.467274   RMSE_x = 0.923502
RMSE_N = 11.882974   RMSE_y = 0.88022

****

******** In the case of 7 parameters ********

RMSE in (m) for the E and N and in (pixel) for the x and y coordinates for the control points are as follows:

RMSE_E = 6.631787   RMSE_x = 0.491243
RMSE_N = 5.627701   RMSE_y = 0.416867

RMSE in (m) for the E and N and in (pixel) for the x and y coordinates for the check points are as follows:

RMSE_E = 10.910824   RMSE_x = 0.808209
RMSE_N = 11.882309   RMSE_y = 0.880171

****

******** In the case of 8 parameters ********

RMSE in (m) for the E and N and in (pixel) for the x and y coordinates for the control points are as follows:

RMSE_E = 6.309826 RMSE_x = 0.467394
RMSE_N = 3.576607 RMSE_y = 0.264934

RMSE in (m) for the E and N and in (pixel) for the x and y coordinates for the check points are as follows:

RMSE_E = 11.304934 RMSE_x = 0.837403
RMSE_N = 11.723722 RMSE_y = 0.868424

****

******** In the case of 9 parameters ********

RMSE in (m) for the E and N and in (pixel) for the x and y coordinates for the control points are as follows:

RMSE_E = 6.304191 RMSE_x = 0.466977
RMSE_N = 2.404191 RMSE_y = 0.178088

RMSE in (m) for the E and N and in (pixel) for the x and y coordinates for the check points are as follows:

RMSE_E = 11.309602 RMSE_x = 0.837748
RMSE_N = 11.852682 RMSE_y = 0.877976

****

******** In the case of 10 parameters ********

RMSE in (m) for the E and N and in (pixel) for the x and y coordinates for the control points are as follows:

RMSE_E = 5.371502 RMSE_x = 0.397889
RMSE_N = 2.368285 RMSE_y = 0.175428

RMSE in (m) for the E and N and in (pixel) for the x and y coordinates for the check points are as follows:

RMSE_E = 11.595439 RMSE_x = 0.858921
RMSE_N = 12.014853 RMSE_y = 0.889989

****

******** In the case of 11 parameters ********

RMSE in (m) for the E and N and in (pixel) for the x and y coordinates for the control points are as follows:

RMSE_E = 4.896819 RMSE_x = 0.362727
RMSE_N = 2.360278 RMSE_y = 0.174835

RMSE in (m) for the E and N and in (pixel) for the x and y coordinates for the check points are as follows:

RMSE_E = 11.489447 RMSE_x = 0.85107
RMSE_N = 12.114845 RMSE_y = 0.897396

****

******** In the case of 12 parameters ********

RMSE in (m) for the E and N and in (pixel) for the x and y coordinates for the control points are as follows:

RMSE_E = 4.730059 RMSE_x = 0.350375
RMSE_N = 2.288788 RMSE_y = 0.16954

RMSE in (m) for the E and N and in (pixel) for the x and y coordinates for the check points are as follows:

RMSE_E = 11.993893 RMSE_x = 0.888437
RMSE_N = 11.935498 RMSE_y = 0.884111

****

******** In the case of 13 parameters ********

RMSE in (m) for the E and N and in (pixel) for the x and y coordinates for the control points are as follows:

RMSE_E = 4.623088 RMSE_x = 0.342451
RMSE_N = 2.286719 RMSE_y = 0.169387

RMSE in (m) for the E and N and in (pixel) for the x and y coordinates for the check points are as follows:

RMSE_E = 12.480709 RMSE_x = 0.924497
RMSE_N = 11.93344 RMSE_y = 0.883959

****

******** In the case of 14 parameters ********

RMSE in (m) for the E and N and in (pixel) for the x and y coordinates for the control points are as follows:

RMSE_E = 4.560319 RMSE_x = 0.337801
RMSE_N = 1.60187 RMSE_y = 0.118657

RMSE in (m) for the E and N and in (pixel) for the x and y coordinates for the check points are as follows:

RMSE_E = 12.712455 RMSE_x = 0.941663
RMSE_N = 11.634542 RMSE_y = 0.861818

****

******** In the case of 15 parameters ********

RMSE in (m) for the E and N and in (pixel) for the x and y coordinates for the control points are as follows:

RMSE_E = 3.526798 RMSE_x = 0.261244
RMSE_N = 0.903794 RMSE_y = 0.066948

RMSE in (m) for the E and N and in (pixel) for the x and y coordinates for the check points are as follows:

RMSE_E = 15.416238 RMSE_x = 1.141944
RMSE_N = 12.301618 RMSE_y = 0.911231

****

******** In the case of 16 parameters ********

RMSE in (m) for the E and N and in (pixel) for the x and y coordinates for the control points are as follows:

RMSE_E = 3.474005 RMSE_x = 0.257334
RMSE_N = 0.148074 RMSE_y = 0.010968

RMSE in (m) for the E and N and in (pixel) for the x and y coordinates for the check points are as follows:

RMSE_E = 15.801076 RMSE_x = 1.17045
RMSE_N = 12.343451 RMSE_y = 0.91433

****

******** In the case of 17 parameters ********

RMSE in (m) for the E and N and in (pixel) for the x and y coordinates for the control points are as follows:

RMSE_E = 2.015104   RMSE_x = 0.149267
RMSE_N = 0.14807   RMSE_y = 0.010968

RMSE in (m) for the E and N and in (pixel) for the x and y coordinates for the check points are as follows:

RMSE_E = 18.407891   RMSE_x = 1.363548
RMSE_N = 12.344097   RMSE_y = 0.914378

****

******** In the case of 18 parameters ********
The residuals in E and N for the Control Points are as follows:

DE(112) = 5.266e-06	DN(112) = 3.662e-05
DE(205) = -4.996e-06	DN(205) = -3.85e-05
DE(200) = -9.058e-07	DN(200) = -3.369e-06
DE(135) = 1.413e-07	DN(135) = 8.153e-07
DE(61) = 1.959e-06	DN(61) = 1.036e-05
DE(12) = -9.918e-07	DN(12) = -5.767e-06
DE(122) = 1.111e-06	DN(122) = 9.416e-06
DE(128) = -5.428e-07	DN(128) = -2.967e-06
DE(45) = -1.228e-06	DN(45) = -6.703e-06
DE(220) = 1.403e-06	DN(220) = 9.369e-06
DE(106) = 3.228e-06	DN(106) = 1.706e-05
DE(201) = 3.522e-06	DN(201) = 1.913e-05
DE(23) = -6.305e-07	DN(23) = -6.205e-06
DE(83) = -3.086e-06	DN(83) = -1.573e-05
DE(89) = 3.965e-06	DN(89) = 2.901e-05
DE(93) = -6.204e-06	DN(93) = -3.826e-05
DE(116) = -7.372e-07	DN(116) = -3.982e-06
DE(125) = -1.275e-06	DN(125) = -1.034e-05

RMSE in (m) for the E and N and in (pixel) for the x and y coordinates for the control points are as follows:

RMSE_E = 2.995e-06   RMSE_x = 2.218e-07
RMSE_N = 1.967e-05   RMSE_y = 1.457e-06

The residuals in E and N for the Check Points are as follows:

DE(28) =	-45.019	DN(28) =	-5.234
DE(30) =	-102.195	DN(30) =	16.736
DE(32) =	-124.173	DN(32) =	14.494
DE(43) =	89.261	DN(43) =	-15.887
DE(49) =	-156.41	DN(49) =	13.903
DE(58) =	70.506	DN(58) =	-17.09
DE(60) =	-164.28	DN(60) =	16.011
DE(64) =	31.348	DN(64) =	-19.366

DE(66) =	68.665	DN(66) =	-11.946
DE(67) =	45.773	DN(67) =	-7.951
DE(69) =	-90.487	DN(69) =	-0.664
DE(70) =	-99.959	DN(70) =	3.04
DE(71) =	-178.925	DN(71) =	12.725
DE(73) =	-45.642	DN(73) =	16.123
DE(76) =	20.121	DN(76) =	-33.955
DE(82) =	-45.449	DN(82) =	-1.453
DE(85) =	-156.202	DN(85) =	21.165
DE(86) =	-126.565	DN(86) =	4.477
DE(95) =	-78.192	DN(95) =	3.702
DE(96) =	-3.581	DN(96) =	15.907
DE(104) =	-11.255	DN(104) =	32.779
DE(105) =	-18.459	DN(105) =	2.794
DE(110) =	-94.903	DN(110) =	10.15
DE(118) =	9.64	DN(118) =	14.959
DE(119) =	-0.148	DN(119) =	-1.669
DE(120) =	29.412	DN(120) =	-9.165
DE(121) =	-39.414	DN(121) =	10.568
DE(126) =	-7.18	DN(126) =	9.607
DE(127) =	-24.52	DN(127) =	3.098
DE(134) =	93.72	DN(134) =	-14.997
DE(206) =	-82.317	DN(206) =	11.023
DE(207) =	-41.937	DN(207) =	14.913
DE(222) =	68.419	DN(222) =	-18.642

RMSE in (m) for the E and N and in (pixel) for the x and y coordinates for the check points are as follows:

RMSE_E =	85.878	RMSE_x =	6.361
RMSE_N =	14.832	RMSE_y =	1.099

****

****

INPUT DATA FOR BUNDLE ADJUSTMENT PROGRAM

****

Note: These are input data for stereo MOMS-02 mode 1, in the case of the ETH data set.

****

IOP.INP (interior orientation data input)

Format of the input is as follows:
focal length for the left image (forward) its viewing angle
focal length for the right image (backward) its viewing angle
line number for the top left corner of the scene (left image)
line number for the bottom right corner of the scene (left image)
pixel number for the bottom left corner of the scene (left image)
pixel number for the top right corner of the scene (left image)
line number for the top left corner of the scene (right image)
line number for the bottom right corner of the scene (right image)
pixel number for the bottom left corner of the scene (right image)
pixel number for the top right corner of the scene (right image)
the number of scenes

****

```
237.16
0.0
237.2
0.0
0 8260 0 6100
0 8260 0 6100
2
```

****

EOP1.INP (exterior orientation data input for the left image)

Format of the input is as follows:

The first 8 sets of data (24 input data) are the spacecraft's 3D position (X, Y, and Z) in space for 8 different times.

The second 8 sets of data (24 input data) are the spacecraft's 3D velocity (V_X, V_Y, and V_Z) in space for 8 different times.

The third 9 values are the corresponding times for the above position and velocity values and the time of imaging the centre of the scene.

The fourth 9 sets of data are approximation values for the rotation angles (ω, φ, and κ), their first order terms, and their quadratic terms respectively.

****

```
-14656545.0 14315916.0 -7710282.0
-14682246.0 14281062.0 -7725884.0
-14707885.0 14246145.0 -7741447.0
```

-14733460.0 14211164.0 -7756972.0
-14758973.0 14176121.0 -7772459.0
-14784423.0 14141014.0 -7787907.0
-14809810.0 14105846.0 -7803316.0
-14835133.0 14070614.0 -7818686.0
-13402.180 -18136.781 -8135.871
-13369.594 -18169.738 -8115.914
-13336.957 -18202.629 -8095.918
-13304.262 -18235.430 -8075.879
-13271.508 -18268.160 -8055.805
-13238.699 -18300.797 -8035.688
-13205.836 -18333.367 -8015.531
-13172.914 -18365.840 -7995.332
140.0 1140.0 2140.0
3140.0 4140.0 5140.0
6140.0 7140.0 4200.0
0.0 -21.475 0.0
0.0 0.0 0.0
0.0 0.0 0.0

****

EOP2.INP (exterior orientation data input for the right image)
Format of the input is as follows:
The first 8 sets of data (24 input data) are the spacecraft's 3D position (X, Y, and Z) in space for 8 different times.

The second 8 sets of data (24 input data) are the spacecraft's 3D velocity (V_X, V_Y, and V_Z) in space for 8 different times.

The third 9 values are the corresponding times for the above position and velocity values and the time of imaging the centre of the scene.

The fourth 9 sets of data are approximation values for the rotation angles (ω, φ, and κ), their first order terms, and their quadratic terms respectively.
****

-15109494.0 13678994.0 -7985183.0
-15134052.0 13643026.0 -8000082.0
-15158545.0 13606997.0 -8014942.0
-15182973.0 13570908.0 -8029762.0
-15207337.0 13534759.0 -8044543.0
-15231635.0 13498550.0 -8059283.0
-15255869.0 13462281.0 -8073983.0
-15280038.0 13425953.0 -8088644.0
-12807.133 -18717.742 -7770.574
-12773.543 -18749.234 -7749.902
-12739.910 -18780.652 -7729.199
-12706.218 -18811.977 -7708.453
-12672.473 -18843.230 -7687.676

```
-12638.672  -18874.387  -7666.852
-12604.820  -18905.480  -7645.992
-12570.910  -18936.469  -7625.094
 140.0       1140.0      2140.0
3140.0       4140.0      5140.0
6140.0       7140.0      4200.0
0.0         21.475       0.0
0.0         0.0         0.0
0.0         0.0         0.0
```

****

IMAGE1.INP (image coordinate values of the GCPs for the left image)

Format of the input is as follows:

number of control points

image number point number x y

****

```
18
1 12 2216.333  584.867
1 23 1012.967 1138.500
1 45 1796.000 1557.667
1 61 2671.625 2316.625
1 83 2797.375 3822.925
1 89  660.375 4303.375
1 93 1913.375 4644.375
1 106 2704.875 5098.625
1 112 1586.375 5641.625
1 116 2030.900 6277.900
1 122  629.000 6353.900
1 125 1374.125 6626.825
1 128 2765.667 6663.800
1 135 2154.400 7410.100
1 200 1017.125 2646.125
1 201 1863.125 3336.625
1 205  694.825 5083.375
1 220 1555.175  477.625
1 28 1819.375 1111.375
1 30 2066.375 1014.375
1 32 2175.430 1415.650
1 43 1231.000 1708.500
1 49 2309.375 1822.125
1 58 1606.875 2312.625
1 60 2248.925 2738.625
1 64 1152.375 3055.125
1 66 1577.125 3405.125
1 67 1668.667 3472.333
1 69 2075.875 3016.875
1 70 2174.400 3549.000
1 71 2479.200 3316.000
1 73 2723.250 2965.917
```

1	76	985.625	3731.475
1	82	2077.125	3916.625
1	85	2398.225	3673.925
1	86	2488.875	4204.875
1	95	2560.875	4616.675
1	96	2757.875	4601.625
1	104	2012.625	5344.125
1	105	2200.375	5573.125
1	110	941.333	6021.000
1	118	2682.125	5793.625
1	119	2664.500	6033.667
1	120	2734.333	6007.000
1	121	2393.667	6216.667
1	126	1430.933	6507.667
1	127	2251.333	6669.400
1	134	1804.000	7056.900
1	206	890.125	5284.875
1	207	1221.200	4603.700
1	222	1335.625	829.925

****

IMAGE2.INP (image coordinate values of the GCPs for the right image)

Format of the input is as follows:

number of control points

image number point number x y

****

18

2	12	1664.827	182.155
2	23	413.066	697.119
2	45	1226.849	1142.312
2	61	2136.442	1930.485
2	83	2266.536	3442.351
2	89	41.262	3853.149
2	93	1345.660	4234.520
2	106	2169.642	4716.646
2	112	1005.412	5221.775
2	116	1467.500	5874.000
2	122	7.680	5903.759
2	125	783.861	6201.859
2	128	2232.823	6285.950
2	135	1594.806	7012.987
2	200	414.552	2205.344
2	201	1294.684	2922.738
2	205	76.369	4633.466
2	220	977.438	53.729
2	28	1251.270	696.151
2	30	1508.640	607.438
2	32	1621.695	1012.652

```
2 43    638.591 1275.280
2 49   1760.323 1423.616
2 58   1028.444 1890.141
2 60   1696.314 2337.946
2 64    555.374 2619.074
2 66    996.792 2982.720
2 67   1092.106 3052.611
2 69   1516.456 2610.560
2 70   1618.430 3146.301
2 71   1935.737 2923.462
2 73   2189.149 2581.367
2 76    380.870 3290.675
2 82   1516.760 3511.583
2 85   1851.448 3279.520
2 86   1945.231 3814.189
2 95   2019.627 4228.951
2 96   2224.528 4220.410
2 104   1449.016 4937.623
2 105   1645.408 5173.614
2 110    333.572 5579.870
2 118   2146.116 5411.040
2 119   2127.652 5650.709
2 120   2200.520 5626.708
2 121   1846.189 5824.786
2 126    843.018 6084.224
2 127   1697.773 6273.969
2 134   1230.715 6647.172
2 206    279.780 4841.583
2 207    626.000 4170.100
2 222    748.966  398.963
```

```
*************************************************************************
GROUND.INP (ground coordinate values of the GCPs)
Format of the input is as follows:
number of control points      number of check points
point number  X    Y    Z   accuracy of X   accuracy of Y   accuracy of Z
*************************************************************************
```

```
18 33
12  -4372704.6521  4036613.6097 -2287796.3868   1.0  1.0  1.0
23  -4368670.4330  4031360.9002 -2304645.8101   1.0  1.0  1.0
45  -4377196.9343  4026366.9024 -2297165.0569   1.0  1.0  1.0
61  -4388920.5410  4017652.2302 -2290058.9095   1.0  1.0  1.0
83  -4401133.6802  4001261.3960 -2295242.1424   1.0  1.0  1.0
89  -4390130.2469  3997268.2335 -2322928.1660   1.0  1.0  1.0
93  -4401263.6465  3992873.8386 -2309481.2300   1.0  1.0  1.0
106  -4410085.5486  3987461.4835 -2302000.2144   1.0  1.0  1.0
112  -4406503.5653  3982202.5428 -2317795.7619   1.0  1.0  1.0
116  -4414306.2109  3975023.5619 -2315276.8136   1.0  1.0  1.0
```

122	-4405289.6383	3974989.6078	-2332339.5123	1.0	1.0	1.0
125	-4412426.4810	3971592.8406	-2324661.1366	1.0	1.0	1.0
128	-4422255.4524	3970351.0463	-2308153.8725	1.0	1.0	1.0
135	-4423633.4085	3962578.9566	-2318772.2447	1.0	1.0	1.0
200	-4380093.9794	4015054.2965	-2311321.1783	1.0	1.0	1.0
201	-4391092.1488	4007108.1308	-2304291.3052	1.0	1.0	1.0
205	-4396228.5273	3988799.0380	-2325968.1908	1.0	1.0	1.0
220	-4367359.2158	4038161.0619	-2295214.4509	1.0	1.0	1.0
28	-4373971.1408	4031169.1286	-2294900.7120	1.0	1.0	1.0
30	-4374929.7312	4032071.8104	-2291501.5906	1.0	1.0	1.0
32	-4378713.3948	4027676.2416	-2291993.6803	1.0	1.0	1.0
43	-4374465.3603	4025060.5763	-2304585.1827	1.0	1.0	1.0
49	-4382694.4546	4023214.7392	-2292206.9335	1.0	1.0	1.0
58	-4381608.8002	4018331.8620	-2302801.6490	1.0	1.0	1.0
60	-4389207.8062	4013344.6320	-2297011.9054	1.0	1.0	1.0
64	-4384098.1942	4010555.2394	-2311542.9827	1.0	1.0	1.0
66	-4389645.9330	4006516.4176	-2308015.5290	1.0	1.0	1.0
67	-4390776.1499	4005746.3784	-2307219.6889	1.0	1.0	1.0
69	-4390127.2441	4010436.6984	-2300330.6121	1.0	1.0	1.0
70	-4394814.7843	4004611.6197	-2301512.3537	1.0	1.0	1.0
71	-4395139.5753	4006964.7209	-2296810.8761	1.0	1.0	1.0
73	-4394171.5270	4010595.2428	-2292335.0068	1.0	1.0	1.0
76	-4388054.7632	4003300.3011	-2316519.1525	1.0	1.0	1.0
82	-4396915.3066	4000672.2351	-2304303.4255	1.0	1.0	1.0
85	-4397278.6851	4003128.6811	-2299371.6465	1.0	1.0	1.0
86	-4401891.9258	3997315.1019	-2300632.9310	1.0	1.0	1.0
95	-4405481.2048	3992785.6206	-2301602.2064	1.0	1.0	1.0
96	-4406718.0229	3992840.0310	-2299161.3993	1.0	1.0	1.0
104	-4407197.4285	3985211.9982	-2311375.8503	1.0	1.0	1.0
105	-4410198.6056	3982594.1818	-2310138.5918	1.0	1.0	1.0
110	-4404938.1168	3978458.3734	-2327153.0529	1.0	1.0	1.0
118	-4415150.4825	3979913.1467	-2305318.8489	1.0	1.0	1.0
119	-4416826.0648	3977303.3352	-2306601.1411	1.0	1.0	1.0
120	-4417101.5743	3977553.7011	-2305647.8958	1.0	1.0	1.0
121	-4416346.6011	3975473.4937	-2310648.5464	1.0	1.0	1.0
126	-4411918.4987	3972867.3018	-2323451.0855	1.0	1.0	1.0
127	-4418760.0599	3970610.7654	-2314352.0099	1.0	1.0	1.0
134	-4418584.9141	3966641.8114	-2321417.7754	1.0	1.0	1.0
206	-4399066.0974	3986497.8587	-2324528.4084	1.0	1.0	1.0
207	-4396213.6799	3993720.6957	-2317565.5734	1.0	1.0	1.0
222	-4368531.0324	4034494.0836	-2299421.0230	1.0	1.0	1.0

****

OUTPUT FOR BUNDLE ADJUSTMENT PROGRAM

****

****

---------The final result for the first image----------

True anomaly = 0.547369
First rate of the true anomaly = 0.000267
Right ascension of the ascending node = 4.741162
First rate of the right ascension of the ascending node = -5.299907e-05
Inclination = 0.497432
Semi major axis of the orbit = 6678000.003327
Omega = 0.030547 Phi = -0.418319 Kappa = -0.019685
First rate of the omega = 0.000297
First rate of the Phi = 4.853686e-05
First rate of the kappa = 8.785534e-05
Second rate of the omega = -2.631989e-08
Second rate of the Phi = -2.016657e-08
Second rate of the kappa = -1.892339e-06

----------The final result for the second image----------

True anomaly = 0.472142
First rate of the true anomaly = 0.000257
Right ascension of the ascending node = 4.741953
First rate of the right ascension of the ascending node = -4.944761e-05
Inclination = 0.497415
Semi major axis of the orbit = 6677999.995324
Omega = 0.050907 Phi = 0.325521 Kappa = -0.018063
First rate of the omega = 0.000279
First rate of the Phi = -0.000104
First rate of the kappa = -0.000103
Second rate of the omega = -5.212754e-08
Second rate of the Phi = -5.331123e-08
Second rate of the kappa = -1.83714e-06

****
****

The residuals for the image cordinates of CPs are:

	LEFT IMAGE		RIGHT IMAGE	
no.	Dx	Dy	Dx	Dy
12	0.049	0.34	-0.089	0.189
23	-0.098	-0.033	0.05	0.499
45	-0.334	-0.417	0.126	-0.218
61	0.083	-0.361	0.235	-0.612
83	-0.048	0.056	-0.254	0.194
89	0.094	0.183	0.099	-0.248
93	-0.118	-0.045	0.261	-0.321
106	0.338	0.587	0.386	0.739
112	-0.262	0.216	-0.746	0.754
116	0.773	0.328	0.591	0.394
122	-0.146	-0.239	-0.45	0.152
125	-0.137	-0.149	0.173	0.402
128	-0.289	-0.502	-0.851	-0.169

135	-0.139	0.046	0.539	-0.912
200	0.055	-0.119	0.01	-0.473
201	-0.381	-0.236	-0.193	-0.187
205	0.282	0.154	0.243	-0.298
220	0.278	0.192	-0.119	0.117

****
The RMSE in x and y for the GCPs in the left and right images are as follows:

RMSE(x_left)= 0.285   RMSE(y_left)= 0.289
RMSE(x_right)= 0.394   RMSE(y_right)= 0.459
****

The residuals for the image coordinates of the Check Points are:

	LEFT IMAGE		RIGHT IMAGE	
no.	Dx	Dy	Dx	Dy
28	-0.138	0.487	-0.319	0.287
30	0.263	-0.298	0.326	-0.186
32	-0.011	-0.05	0.302	0.164
43	0.098	-0.206	0.242	-0.269
49	0.537	0.507	0.729	0.492
58	-0.225	0.127	-0.316	-0.424
60	0.136	0.185	-0.048	-0.095
64	-0.059	1.03	0.577	1.243
66	-0.409	0.05	-0.179	-0.089
67	0.021	0.112	0.267	0.082
69	-0.379	0.541	-0.146	0.715
70	-0.64	0.329	-0.465	0.388
71	0.414	-0.287	0.44	-0.247
73	-0.259	0.233	-0.75	-0.477
76	-0.076	-0	-0.195	-0.051
82	-0.715	0.127	-0.441	0.005
85	0.353	-0.082	0.711	0.184
86	0.703	-0.192	0.994	-0.12
95	-0.077	0.568	-0.087	0.336
96	0.518	0.037	0.366	-0.302
104	0.277	-0.597	-0.031	-0.442
105	-0.077	-0.352	-0.704	0.835
110	0.408	-0.621	0.139	0.067
118	0.994	-0.827	0.226	-0.345
119	0.643	0.14	-0.238	0.588
120	0.836	0.484	0.211	1.125
121	0.55	-0.748	0.084	0.217
126	0.749	-0.441	0.779	0.019
127	0.074	-0.39	0.082	0.489
134	-0.282	0.22	-0.004	0.055
206	0.664	0.219	0.458	-0.068
207	1.154	-0.09	0.462	0.514
222	-0.142	0.538	-0.279	0.878

****

The RMSE in x and y for the GCPs in the left and right images are as follows:

RMSE(x_left)= 0.497 RMSE(y_left)= 0.425
RMSE(x_right)= 0.436 RMSE(y_right)= 0.48

****

The residuals for the ground coordinates of 18 selected CPs are:

no.	DX	DY	DZ
12	-3.987575	2.351895	0.535106
23	2.902753	-3.233395	8.968297
45	9.361649	-4.964601	2.056651
61	2.486577	-2.587554	-8.00785
83	-1.133769	3.500335	2.28699
89	-2.688329	0.343019	-5.113005
93	3.602925	-4.234132	-2.812147
106	-6.696193	-4.753735	7.399655
112	-1.92241	9.128408	10.480693
116	-10.314736	-5.192242	-0.178556
122	1.250632	5.373264	3.050689
125	5.256982	-5.609575	9.087984
128	2.923151	11.249631	-0.919968
135	4.22744	-6.707832	-11.576425
200	-0.449062	1.32954	-7.319957
201	6.392119	0.631048	0.897568
205	-4.45925	-0.840155	-6.845142
220	-6.682199	4.048155	-1.932589

****

The RMSE in X, Y and Z for the CPs are as follows:

RMSE(X)= 5.177927 RMSE(Y)= 5.211186 RMSE(Z)= 6.346475

****

The residuals for the ground coordinates of 33 selected Check Pts are:

no.	DX	DY	DZ
28	-3.769047	5.205486	2.25508
30	0.363969	-4.202355	-2.67216
32	2.843663	-5.875232	4.231232
43	1.545199	-3.259891	-3.263725
49	-7.130748	-8.703075	3.291575
58	-0.317925	5.72971	-6.591086
60	-4.142646	2.645515	-4.009807
64	-1.947347	-10.67826	17.499785
66	4.653624	0.877134	1.319481
67	0.689303	-4.239706	1.618676
69	1.335056	-0.506116	11.845014
70	4.599495	3.54894	8.806067
71	-1.424043	-4.904669	-4.72048

73	-3.484976	13.381478	-9.858907
76	-0.269349	2.965308	-0.852841
82	7.117596	3.270125	4.373247
85	0.189807	-10.598501	3.040496
86	-2.868951	-12.459114	-3.592712
95	-3.738093	1.81299	2.8835
96	-6.347852	-1.648063	-8.845652
104	-0.085753	1.705034	-6.889005
105	0.534886	7.325955	13.297568
110	-0.076871	-2.086898	0.936059
118	-7.391138	0.85924	-10.224618
119	-11.97662	6.637589	1.47293
120	-14.192894	-0.294248	8.578357
121	-1.368037	-1.231089	2.539119
126	-2.510296	-9.931034	-1.248457
127	3.390453	-3.91219	9.772308
134	2.602402	-1.148552	2.399776
206	-9.213192	-2.5675	-6.0699
207	-13.513095	-2.463721	0.199328
222	-3.070935	2.688718	11.631937

****

The RMSE in X, Y and Z for the check points are as follows:

RMSE(X)= 5.512454 RMSE(Y)= 5.835594 RMSE(Z)= 7.014596