```
% sym02.m  20-jan-02
% space resection (angles only) of star photograph
% try matlab symbolic problem solution
% input data

% focal length
% xo
% yo
% k1
% k2
% k3
% omega (dec. deg.)
% phi (dec. deg.))
% kappa (dec. deg.)
% mm, number of data points
% x1 y1 az1 elev1
% x2 y2 az2 elev2
% ...
% xn yn azn elevn

degrad=45.0/atan(1.0);

% read in numerical values for parameter initial approximations
% followed by photo coordinates & az/el values

fidr=fopen('sym02.inp');
inpn=fscanf(fidr,'%f',10);
foc=inpn(1);
xo=inpn(2);
yo=inpn(3);
k1=inpn(4);
k2=inpn(5);
k3=inpn(6);
omegad=inpn(7);
phid=inpn(8);
kappad=inpn(9);
npts=inpn(10); % number of data points
nval=4; % number of entries per point (x,y,az,el)
dat=fscanf(fidr,'%f',[nval,npts]);
fclose(fidr);
tdat=dat';

% define the parameter include(1) / exclude(0) vector
% order is foc,xo,yo,k1,k2,k3,om,ph,kp

incl=[0 0 0 0 0 0 1 1 1];

% initialize an observation vector
l=zeros(npts*2,1);
l0=zeros(npts*2,1);
for i=1:npts
  ii=i*2-1;
  l(ii)=tdat(i,1);
  l(ii+1)=tdat(i,2);
  l0(ii)=l(ii);
  l0(ii+1)=l(ii+1);
  end

om=omegad/degrad;
ph=phid/degrad;
kp=kappad/degrad;

disp('initial parameters');
foc
xo
yo
k1
k2
k3
om
ph
kp

% define low level symbolic variables to represent the parameters
% and the observations (for a single photo)

par(1)=sym('foc');
par(2)=sym('xo');
par(3)=sym('yo');
```

```
par(4)=sym('k1');
par(5)=sym('k2');
par(6)=sym('k3');
par(7)=sym('om');
par(8)=sym('ph');
par(9)=sym('kp');

obs(1)=sym('x');
obs(2)=sym('y');

% define symbolic intermediate expressions

mw=sym('[1,0,0;0,cos(om),sin(om);0,-sin(om),cos(om)]');
mp=sym('[cos(ph),0,-sin(ph);0,1,0;sin(ph),0,cos(ph)]');
mk=sym('[cos(kp),sin(kp),0;-sin(kp),cos(kp),0;0,0,1]');
t=mp*mw;
m=mk*t;
c=sym('[cx;cy;cz]');
uvw=m*c;
u=uvw(1);
v=uvw(2);
w=uvw(3);

xp=sym('x-xo');
yp=sym('y-yo');
r=sym('sqrt(xp^2 + yp^2)');
dr=sym('k1*r^3 + k2*r^5 + k3*r^7');
dx=sym('dr*(xp/r)');
dy=sym('dr*(yp/r)');

% now define the condition equations and substitute in the
% just defined intemediate expressions

Fx=sym('xp + dx + foc*(u/w)');
Fx=subs(Fx,'dx',dx);
Fx=subs(Fx,'dr',dr);
Fx=subs(Fx,'r',r);
Fx=subs(Fx,'xp',xp);
Fx=subs(Fx,'yp',yp);
Fx=subs(Fx,'u',u);
Fx=subs(Fx,'w',w);

% same for Fy

Fy=sym('yp + dy + foc*(v/w)');
Fy=subs(Fy,'dy',dy);
Fy=subs(Fy,'dr',dr);
Fy=subs(Fy,'r',r);
Fy=subs(Fy,'xp',xp);
Fy=subs(Fy,'yp',yp);
Fy=subs(Fy,'v',v);
Fy=subs(Fy,'w',w);

% now create, symbolically, the matrices of partial derivatives
% A and B

Bsymjac=jacobian([Fx;Fy],par);
Asymjac=jacobian([Fx;Fy],obs);

% force exactly 6 iterations

for iter=1:6

  % A,B,f are regular numeric
 variables
  B=zeros(2*npts,9);
  A=zeros(2*npts,2*npts);
  f=zeros(2*npts,1);

  for i=1:npts
    ii=(i*2)-1;
    x=l0(ii);
    y=l0(ii+1);
    azd=tdat(i,3);
    eld=tdat(i,4);

    % correct for refraction
    % from "sight reduction" p.280 nautical almanac 2002
    den_arg=eld + 7.31/(eld + 4.4);
```

```
    r0=0.0167/tan(den_arg/degrad);
    el=(eld-r0)/degrad;
    az=azd/degrad;

    % compute c's from az,el
    cx=sin(az)*cos(el);
    cy=cos(az)*cos(el);
    cz=sin(el);
    %x
    %y
    %cx
    %cy
    %cz

    % substitute numbers into symbolic expressions for contributions to
    % A, B, and f

    Bcon=eval(Bsymjac);
    Acon=eval(Asymjac);
    fcon=eval([Fx;Fy]);

    % now everything is in numeric form

    B(ii:ii+1,:)=Bcon;
    A(ii:ii+1,ii:ii+1)=Acon;
    f(ii:ii+1)=-fcon;
    end

  f=f - A*(l-l0);
  Qe=A*A';
  We=inv(Qe);
  N=B'*We*B;
  t=B'*We*f;

  % process the include/exclude request
  for i=1:9
    if incl(i) == 0
      for j=1:9
        N(j,i)=0.0;
        N(i,j)=0.0;
        end;
      t(i)=0.0;
      N(i,i)=1.0;
      end
    end

  del=inv(N)*t;
  vv=A'*We*(f-B*del);
  l0=l+vv;
  %A
  %B
  %f
  del
  %vv

  % update the parameters

  foc=foc+del(1);
  xo=xo+del(2);
  yo=yo+del(3);
  k1=k1+del(4);
  k2=k2+del(5);
  k3=k3+del(6);
  om=om+del(7);
  ph=ph+del(8);
  kp=kp+del(9);
  end

vv
nn=npts*2;
npar=sum(incl);
red=nn-npar;
red
sig0=sqrt((vv'*vv)/red);
sig0
incl
foc
xo
yo
```

```
k1
k2
k3
om
ph
kp


mm=eval(m);
tt=acos(mm(3,3)); % note: two choices here unless force t-pos
aa=atan2(mm(3,1)/sin(tt),-mm(3,2)/sin(tt));
k2=atan2(mm(1,3)/sin(tt),mm(2,3)/sin(tt));

disp('azimuthm, tilt, kappa (deg) of camera boresight');
ad=aa*degrad
td=tt*degrad
k2d=k2*degrad
```