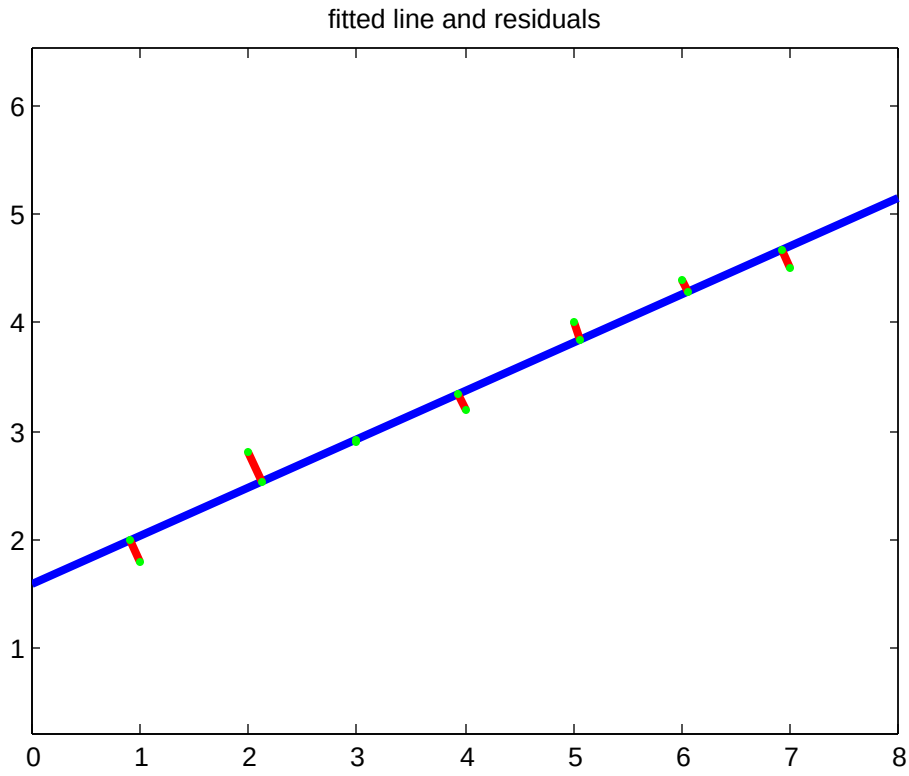
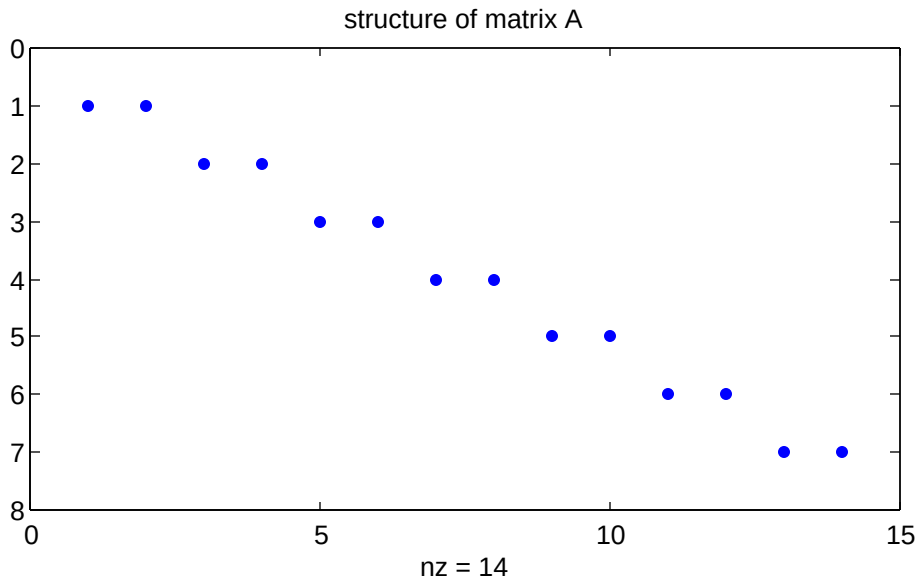
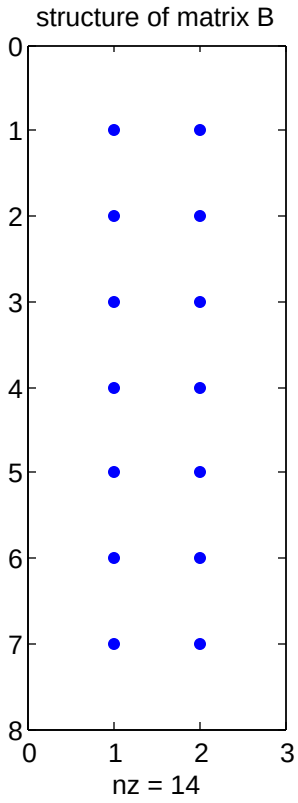
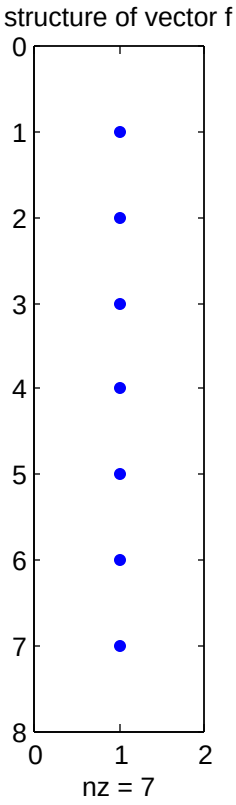

fitted line and residuals
6
5
4
3
2
1
0
1
2
3
4
5
6
7
8

```
hw5b
del =
    0.4429
    1.6000
v =
    0.2429
   -0.3143
    0.0286
    0.1714
   -0.1857
   -0.1429
    0.2000
ok now do the nonlinear GLS
magfdel =
    0.4863
magfdel =
    0.0238
magfdel =
    0.0015
magfdel =
  1.6722e-004
magfdel =
  1.0672e-005
magfdel =
  1.1775e-006
magfdel =
  7.5146e-008
m0 =
    0.4466
b0 =
    1.5849
v =
   -0.0862
    0.1930
    0.1198
   -0.2683
   -0.0092
    0.0207
   -0.0638
    0.1429
    0.0677
   -0.1517
    0.0504
   -0.1128
   -0.0787
    0.1762
ans =
    0.9138    2.1198    2.9908    3.9362    5.0677    6.0504    6.9213
ans =
    1.9930    2.5317    2.9207    3.3429    3.8483    4.2872    4.6762
diary off
```


structure of matrix A
0
1
2
3
4
5
6
7
8
0
5
10
15
nz = 14


structure of matrix B
0
1
2
3
4
5
6
7
8
0
1
2
3
nz = 14


structure of vector f
0
1
2
3
4
5
6
7
8
0
1
2
nz = 7

```
% hw5b.m  19-nov-09
% GLS line fit - first to I/O solution for parameter approx.

% I/O linear
n=7;
n0=2;
r=5;
u=2;
x=[1;2;3;4;5;6;7];
y=[1.8;2.8;2.9;3.2;4.0;4.4;4.5];

% more accurate data, convergence is quadratic here !!!
%x=[    0.9;    2.1;    2.9;    3.9;    5.0;    6.0;    6.9 ];
%y=[    1.9;    2.5;    2.9;    3.3;    3.8;    4.2;    4.6 ];

B=zeros(n,u);
f=zeros(n,1);
W=eye(n);
for i=1:n
  B(i,:)=[-x(i) -1];
  f(i)=-y(i);
  end
del=inv(B'*W*B)*B'*W*f
v=f-B*del
%slope & intercept
m0=del(1);
b0=del(2);

% ok now GLS non-linear
disp('ok now do the nonlinear GLS');

n=14;
n0=2 + 7; % =9
r=5; % same !!
u=2;
c=r+u; % 7 !!
l=[x(1);y(1);x(2);y(2);x(3);y(3);x(4);y(4);x(5);y(5);x(6);y(6);x(7);y(7)];
l0=l;
W=eye(n);
Q=inv(W);

iter=1;
keep_going=1;
while(keep_going == 1)
  x0=[l0(1);l0(3);l0(5);l0(7);l0(9);l0(11);l0(13)];
  y0=[l0(2);l0(4);l0(6);l0(8);l0(10);l0(12);l0(14)];
  A=zeros(c,n);
  B=zeros(c,u);
  f=zeros(c,1);
  for i=1:c
    a=[-m0 1];
    index=(i-1)*2 + 1;
    A(i,index:index+1)=a;
    B(i,:)=[-x0(i) -1];
    F=y0(i) - m0*x0(i) - b0;
```

```
    f(i)=-F - a*([x(i);y(i)] - [x0(i);y0(i)]);
    end
  if(iter==1)
    spy(A);
    title('structure of matrix A');
    figure(2);
    spy(B);
    title('structure of matrix B');
    figure(3);
    spy(f);
    title('structure of vector f');
    end
  Qe=A*Q*A';
  We=inv(Qe);
  N=B'*We*B;
  t=B'*We*f;
  del=inv(N)*t;
  m0=m0 + del(1);
  b0=b0 + del(2);
  k=We*(f - B*del);
  v=Q*A'*k;
  old_l0=l0;
  l0=l + v;
  delta_l=l0-old_l0;
  fulldel=[del;delta_l];
  magfdel=sqrt(fulldel'*fulldel)
  if( (all(abs(del) < 1.0e-08) && (iter >=2)) || (iter >= 10) )
    keep_going=0;
    end
  iter=iter+1;
  end

m0
b0
v
x0'
y0'

figure(4);
minx=x(1)-1;
maxx=x(7)+1;
plx=[minx maxx];
ply=[(m0*minx+b0)  (m0*maxx+b0)];
plot(plx,ply,'linewidth',3);
hold on
for i=1:7
  plx=[x(i) x0(i)];
  ply=[y(i) y0(i)];
  plot(plx,ply,'r-','linewidth',3);
  plot(x(i),y(i),'g.','linewidth',3);
  plot(x0(i),y0(i),'g.','linewidth',3);
  title('fitted line and residuals');
  end
axis equal
```

```
hnet3
number of observations
nobs =
     9
number of points
npnt =
     7
number of fixed coordinate components
ncpc =
     8
number of unknown parameters
nunkn =
     6
redundancy
red =
     3
show iteration number and vTWv

iteration & vtWv
ans =
                         1          8.84324759770352
iteration & vtWv
ans =
                         2          8.84330875008965
iteration & vtWv
ans =
                         3          8.84330875008079
we have converged
finished
parameters x y
ans =
                         1                     10300                      4800
                         2                     10600                      3900
                         3          11800.1196544927          4399.94388470638
                         4          12700.0816690733           2599.9457451043
                         5           11600.099174794          3000.01070396858
                         6                     10300                      1800
                         7                     12100                      1100
number residual obs-code
units: meters & arc-seconds
ans =
                         1           7.82147299758242                         2
                         2       -0.0911288243049075                          1
                         3          22.2070559075294                          2
                         4         -0.10747156373795                          1
                         5          3.78300914297971                          2
                         6         0.135740471355647                          1
                         7          2.12572434713202                          2
                         8        0.0507365617868507                          1
                         9         -22.3016741966046                          2
P and alpha associated with test statistic
test statistic    dof    P(test)    alpha(test)
ans =
         8.84330875008079                         3          0.96855161920902
0.0314483807909804
level of significance of the test     critical value (low)   critical value (high)
ans =
                    0.05         0.215795282623898          9.34840360449614
passed 2-sided global test at 0.05
```

```
% hnet3.m  19-nov-06
% general 2d network with distance and angle
% observations
% no plot no statistics
% hardwire sigmas
% const + ppm for distance sigma
% add post adjustment statistics copied from hw6a.m (04/05 ?)

%filnum=input('which file are we processing? ');
%disp('we are processing file number: ');
%filnum

degrad=180/pi;
dat=textread('obs.dat');
code=dat(:,1);
at=dat(:,2);
fr=dat(:,3);
to=dat(:,4);
obs1=dat(:,5);
obs2=dat(:,6);
obs3=dat(:,7);
% meters or DMS
% hardwired sigmas
ang_sig_sec=15;
dst_sig_const=0.1;
dst_sig_ppm=0;
ang_sig_rad=ang_sig_sec*(1/3600)*(1/degrad);

[m,n]=size(dat);
nobs=m;
disp('number of observations');
nobs

dat=textread('pnt.dat');
x=dat(:,1);
y=dat(:,2);
[m,n]=size(dat);
npnt=m;
disp('number of points');
npnt

dat=textread('control.dat');
cpn=dat(:,1);
cpc=dat(:,2);
[m,n]=size(dat);
ncpc=m;
disp('number of fixed coordinate components');
ncpc

nunkn=2*npnt - ncpc;
disp('number of unknown parameters');
nunkn
red=nobs-nunkn;
disp('redundancy');
red

% parameter order [x1 y1 x2 y2 ... xn yn]

W=eye(nobs);
s0=1.0;
keep_going=1;
old_phi=9.99e+09;
iter=1;
disp('show iteration number and vTWv');

while (keep_going == 1)
  BB=zeros(nobs,npnt*2);
  f=zeros(nobs,1);
  for i=1:nobs
    if(code(i) == 1)
      [b,F,comp_obs]=dist2d(x,y,at(i),to(i),obs1(i));
      d=comp_obs;
      sigma=dst_sig_const + (d/1000000.0)*dst_sig_ppm;
      if(iter == 1)
        disp('distance sigma for ith observation');
        [i sigma]
        end
      W(i,i)=1.0/(sigma^2);
      at_idx=(at(i)-1)*2 + 1;
      to_idx=(to(i)-1)*2 + 1;
      BB(i,at_idx)=b(1);
      BB(i,at_idx+1)=b(2);
      BB(i,to_idx)=b(3);
```

```
      BB(i,to_idx+1)=b(4);
      f(i)=-F;
      end
    if(code(i) == 2)
      [b,F,comp_obs]=ang(x,y,at(i),fr(i),to(i),obs1(i),obs2(i),obs3(i));
      sigma=ang_sig_rad;
      if(iter == 1)
        disp('distance sigma for ith observation');
        [i sigma]
      end
      W(i,i)=1.0/(sigma^2);
      at_idx=(at(i)-1)*2 + 1;
      fr_idx=(fr(i)-1)*2 + 1;
      to_idx=(to(i)-1)*2 + 1;
      BB(i,at_idx)=b(1);
      BB(i,at_idx+1)=b(2);
      BB(i,fr_idx)=b(3);
      BB(i,fr_idx+1)=b(4);
      BB(i,to_idx)=b(5);
      BB(i,to_idx+1)=b(6);
      f(i)=-F;
      end
    end

% !!!!!!!!!!
% keyboard

% save some intermediate results for student checking
%  if(iter == 1)
%    purge=[1 2 3 4 11 12 13 14]
%    B=elim_col(BB,purge);
%    save bwfxy B W f x y
%    clear B
%    end

  % now purge the control columns
  purge=zeros(ncpc,1);
  for i=1:ncpc
    purge(i)=(cpn(i)-1)*2 + cpc(i);
    end
  B=elim_col(BB,purge);
  N=B'*W*B;
  t=B'*W*f;
  del=inv(N)*t;
  v=f-B*del;
  phi=v'*W*v;

  %keyboard

  disp('iteration & vtWv');
  [iter phi]
  if(abs((phi-old_phi)/old_phi) < 1.0e-06)
    keep_going=0;
    converged=1;
    disp('we have converged');
    end
  if(iter > 10)
    keep_going=0;
    converged=0;
    disp('too many iterations');
    end
  old_phi=phi;

  % insert zeros into delta
  del2=ins_zerv(del,purge);
  delx=zeros(npnt,1);
  dely=zeros(npnt,1);
  ix=1;
  iy=2;
  for i=1:npnt
    delx(i)=del2(ix);
    dely(i)=del2(iy);
    ix=ix+2;
    iy=iy+2;
    end
  x=x + delx;
  y=y + dely;

%  save some intermediate results for student checking
%  if(iter == 1)
%    save iter1 purge del del2 delx dely delz x y z v
%    end
```

```
  iter=iter+1;
  end

disp('finished');
disp('parameters x y');
count=1:npnt;
count=count';
[count x y]

count=1:nobs;
count=count';
vdisp=v;
for i=1:nobs
  if (code(i) == 2)
    vdisp(i)=vdisp(i)*degrad*3600;
    end
  end
disp('number residual obs-code');
disp('units: meters & arc-seconds');
[count vdisp code]

% ********************************************
% post adjustment error analysis
% ********************************************

% following is the test statistic
glo_test=(v'*W*v)/s0^2;
dof=red;
Pstat=chi2cdf(glo_test,dof);
alpha=1-Pstat;
disp('P and alpha associated with test statistic');
disp('test statistic    dof    P(test)    alpha(test)');
[glo_test dof Pstat alpha]

% let's make 2-sided test
% level of significance
lev_sign=0.05;
crit_val_low=chi2inv(lev_sign/2,dof);
crit_val_high=chi2inv(1-lev_sign/2,dof);
disp('level of significance of the test     critical value (low)   critical value (high)');
[lev_sign    crit_val_low    crit_val_high]

Qdd=inv(N);
NewQ=ins_zerm(Qdd,purge);
Qdd=NewQ;
s0hat_sqr=(v'*W*v)/red;

if((glo_test < crit_val_low) || (glo_test > crit_val_high))
  disp('failed 2-sided global test at 0.05');
  disp('scale Qdd by the a posteriori estimate of sigma-nought squared');
  Sdd=Qdd*s0hat_sqr;
  passed=0;
else
  disp('passed 2-sided global test at 0.05');
  disp('scale Qdd by the a priori value of sigma-nought squared');
  Sdd=Qdd*s0^2;
  passed=1;
  end

for i=1:npnt
  idx=(i-1)*2 + 1;
  subm=Sdd(idx:idx+1,idx:idx+1);
  disp('covariance matrix for point');
  i
  subm
  end

% ok now we want 50% confidence interval for point 1-X
%figure(2);
%P=0.5;
%subm=Sdd(1:2,1:2);
%disp('2x2 covariance matrix for point 1');
%subm
%sigx=sqrt(subm(1,1));
%sigy=sqrt(subm(2,2));
%P_prime=1 - (1-P)/2;
%estm=[x(1); y(1)];
%if (passed == 1)
%  zz=norminv(P_prime,0,1);
%  half=zz*sigx;
%  disp('zz sigx half');
%  [zz sigx half]
%  intvl=[estm(1)-half; estm(1)+half];
```

```
%  disp('interval')
%  intvl
%else
  % (passed == 0)
%  tt=tinv(P_prime,dof);
%  half=tt*sigx;
%  disp('tt sigx half');
%  [tt sigx half]
%  intvl=[estm(1)-half; estm(1)+half];
%  disp('interval');
%  intvl
%  end
%disp('center of interval,   displacement +/-');
%[estm(1) half]
%len_intvl=half+half;

% ok now we want 95% confidence region for point 2

P=0.95;
subm=Sdd(3:4,3:4);
disp('2x2 covariance matrix for point 2');
subm
estm=[x(2); y(2)];
sigx=sqrt(subm(1,1));
sigy=sqrt(subm(2,2));

[eigvec,eigval]=eig(subm);
if(eigval(1,1) > eigval(2,2))
  lam1=eigval(1,1);
  lam2=eigval(2,2);
  evect1=eigvec(:,1);
  evect2=eigvec(:,2);
  theta=atan2(eigvec(2,1),eigvec(1,1));
  % theta is angle to major axis
else
  lam1=eigval(2,2);
  lam2=eigval(1,1);
  evect1=eigvec(:,2);
  evect2=eigvec(:,1);
  theta=atan2(eigvec(2,2),eigvec(1,2));
  % theta is angle to major axis
  end
theta_deg=theta*degrad;
disp('lambda_1   lambda_2    theta   theta_deg');
[lam1  lam2  theta  theta_deg]
disp('major axis vector');
evect1
disp('minor axis vector');
evect2

if(passed == 1)
  C=sqrt(chi2inv(P,2));
  disp('C from Chi-squared');
  C
else
  C=sqrt(2*finv(P,2,dof));
  disp('C from F');
  C
  end

maj_ax=C*sqrt(lam1);
min_ax=C*sqrt(lam2);
disp('lam1  sqrt(lam1)');
[lam1 sqrt(lam1)]
disp('lam2  sqrt(lam2)');
[lam2 sqrt(lam2)]
disp('C  maj_ax  min_ax');
[C  maj_ax  min_ax]

a=maj_ax;
b=min_ax;
px=zeros(101,1);
py=zeros(101,1);
for i=1:100
  alph=(i/100)*2*pi;
  xx=a*cos(alph);
  yy=b*sin(alph);
  px(i)= cos(-theta)*xx + sin(-theta)*yy;
  py(i)=-sin(-theta)*xx + cos(-theta)*yy;
  end
xx=a;
yy=0;
px(101)=px(1);
```

```
py(101)=py(1);
px=px + estm(1);
py=py + estm(2);

%tick=0.05*len_intvl;

%ppx=[intvl(1); intvl(1)];
%ppy=[estm(2)-tick; estm(2)+tick];
%plot(ppx,ppy,'linewidth',3);
%hold on

%ppx=[intvl(2); intvl(2)];
% same ppy
%plot(ppx,ppy,'linewidth',3);

%ppx=[intvl(1); intvl(2)];
%ppy=[estm(2); estm(2)];
%plot(ppx,ppy,'linewidth',3);

plot(px,py,'linewidth',3);
hold on
axis equal
scale_ax(0.9);
title('95% confidence ellipse ');

v=axis;
px=[estm(1); estm(1)];
py=[v(3); v(4)];
plot(px,py,'linewidth',2,'color','black');
px=[v(1); v(2)];
py=[estm(2); estm(2)];
plot(px,py,'linewidth',2,'color','black');

y_range=v(4)-v(3);
x_range=v(2)-v(1);
lin_spc=y_range/12;
locx=v(1) + 0.6*x_range;
locy=v(3) + 0.4*y_range;
str=[num2str(sigx) ' sig-x'];
text(locx,locy,str);
str=[num2str(sigy) ' sig-y'];
text(locx,locy-1*lin_spc,str);
str=[num2str(maj_ax) ' semi-maj axis'];
text(locx,locy-2*lin_spc,str);
str=[num2str(min_ax) ' semi-min axis'];
text(locx,locy-3*lin_spc,str);

figure(2)
axis(v);
px=[estm(1); estm(1)];
py=[v(3); v(4)];
plot(px,py,'linewidth',2,'color','black');
hold on
px=[v(1); v(2)];
py=[estm(2); estm(2)];
plot(px,py,'linewidth',2,'color','black');
half_width=(v(2)-v(1))/2.0;
half__height=(v(4)-v(3))/2.0;
% major axis
px=[estm(1)+evect1(1)*maj_ax estm(1)-evect1(1)*maj_ax];
py=[estm(2)+evect1(2)*maj_ax estm(2)-evect1(2)*maj_ax];
plot(px,py,'linewidth',2,'color','red');
% minor axis
px=[estm(1)+evect2(1)*min_ax estm(1)-evect2(1)*min_ax];
py=[estm(2)+evect2(2)*min_ax estm(2)-evect2(2)*min_ax];
plot(px,py,'linewidth',2,'color','red');
title('major and minor axes of ellipse scaled by probability');
axis equal
```

```
% dist2d.m.m  6-nov-02
% function to evaluate distance condition equation
% and return elements of B-matrix, F, and computed obs
% function [b,F,cmpobs]=dist2d(x,y,at,to,obs)
% F = obs - sqrt((xt-xa)^2 + (yt-ya)^2) = 0
% order of unkowns: xa,ya,xt,yt
% args
%  x : array of x-coords of network points
%  y : array of y-coords of network points
%  at : index of "at" point
%  to : index of "to" point
%  obs : the distance observation

function [b,F,comp_obs]=dist2d(x,y,at,to,obs)
b=zeros(1,4);
dobs=obs;
xa=x(at);
ya=y(at);
xt=x(to);
yt=y(to);

dx=xt-xa;
dy=yt-ya;
D0=sqrt(dx^2+dy^2);
b(1)=(xt-xa)/D0;
b(2)=(yt-ya)/D0;
b(3)=-(xt-xa)/D0;
b(4)=-(yt-ya)/D0;
F=dobs - D0;
comp_obs=D0;
```

```
% ang.m  25-oct-06
% function to evaluate angle condition equation
% and return elements of B-matrix, F, and computed obs
% function [b,F,comp_obs]=ang(x,y,z,at,to,degree,minute,second)
% F_ang = theta - (atan((xk-xi)/(yk-yi)) - atan((xj-xi)/(yj-yi)) = 0
% at = i, from = j, to = k
% order of unknowns: xi,yi,xj,yj,xk,yk or xat,yat,xfrom,yfrom,xto,yto
% args
%  x : array of x-coords of network points
%  y : array of y-coords of network points
%  at : index of "at"point
%  from : index of "from" point
%  to : index of "to" point
%  degree,minute,second: d,m,s of direction observation

function [b,F,comp_obs]=ang(x,y,at,from,to,degree,minute,second)
degrad=180/pi;
xi=x(at);
yi=y(at);
xj=x(from);
yj=y(from);
xk=x(to);
yk=y(to);

dx_ij=xj-xi;
dy_ij=yj-yi;
dx_ik=xk-xi;
dy_ik=yk-yi;
D2_ij=dx_ij^2 + dy_ij^2;
D2_ik=dx_ik^2 + dy_ik^2;

dF_dxi= dy_ik/D2_ik - dy_ij/D2_ij;
dF_dyi=-dx_ik/D2_ik + dx_ij/D2_ij;
dF_dxj= dy_ij/D2_ij;
dF_dyj=-dx_ij/D2_ij;
dF_dxk=-dy_ik/D2_ik;
dF_dyk= dx_ik/D2_ik;
b=[dF_dxi dF_dyi dF_dxj dF_dyj dF_dxk dF_dyk];

az_ij=atan2(dx_ij,dy_ij);
az_ik=atan2(dx_ik,dy_ik);
angle=az_ik-az_ij;
if(angle < 0.0)
  angle=angle + 2*pi;
  end
theta=angle;
comp_obs=theta;
aobs=(degree + minute/60.0 + second/3600.0)/degrad;
F=aobs - comp_obs;
```

```
% elim_col.m  8-nov-04
% eliminate a list of columns from a matrix

function Bnew = elim_col(B,col_list);
[m,n]=size(B);
[p,q]=size(col_list);
nelim=max([p q]);
newcol=n-nelim;
if(newcol<1)
  disp('trying to eliminate too many columns');
  pause
  end

Bnew=zeros(m,newcol);
ii=1;
for i=1:n
  ok=1;
  for j=1:nelim
    if(col_list(j) == i)
      ok=0;
      end
    end

  if(ok == 1)
    Bnew(:,ii)=B(:,i);
    ii=ii+1;
    end
  end
```

```
% ins_zerv.m  8-nov-04
% insert zeros into a vector

function del2 = ins_zerv(del,col_list);
[m,n]=size(del);
orig_size=max([m n]);
[p,q]=size(col_list);
nadd=max([p q]);
newdim=orig_size + nadd;

del2=zeros(newdim,1);
ii=1;
for i=1:newdim
  ins=0;
  for j=1:nadd
    if(col_list(j) == i)
      ins=1;
      end
    end

  if(ins == 1)
    del2(i)=0;
  else
    del2(i)=del(ii);
    ii=ii+1;
    end

  end
```

```
% ins_zerm.m  8-nov-04
% insert zero rows & cols into a square matrix

function Ni3 = ins_zerm(Ni,col_list);
[m,n]=size(Ni);
orig_size=m;
[p,q]=size(col_list);
nadd=max([p q]);
newdim=orig_size + nadd;

Ni2=zeros(newdim,orig_size);

% first the rows
ii=1;
for i=1:newdim
  ins=0;
  for j=1:nadd
    if(col_list(j) == i)
      ins=1;
      end
    end

  if(ins == 1)
    Ni2(i,:)=zeros(1,orig_size);
  else
    Ni2(i,:)=Ni(ii,:);
    ii=ii+1;
    end
  end

Ni3=zeros(newdim,newdim);

% now the cols
ii=1;
for i=1:newdim
  ins=0;
  for j=1:nadd
    if(col_list(j) == i)
      ins=1;
      end
    end

  if(ins == 1)
    Ni3(:,i)=zeros(newdim,1);
  else
    Ni3(:,i)=Ni2(:,ii);
    ii=ii+1;
    end
  end
```