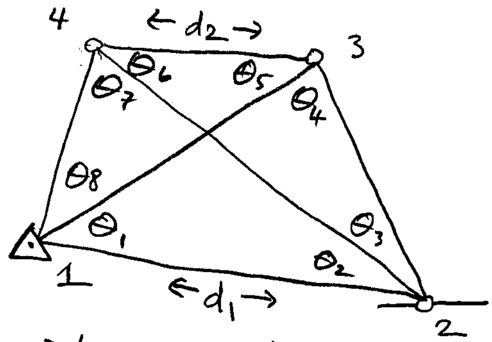
HW4 Solution



$(X_1, Y_1) = (362.0, 415.0)$
$Y_2 = 129.0$ } control points

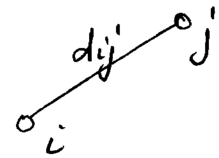
$n = 10$
$n_0 = 5$
$r = 5$

use <u>indirect</u> <u>observations</u>

$u = n_0 = 5 \quad (X_2, X_3, Y_3, X_4, Y_4)$

write a matlab function to evaluate distance condition equation

$$d_{ij} = \left[(x_j - x_i)^2 + (y_j - y_i)^2\right]^{1/2}, \quad \text{or}$$

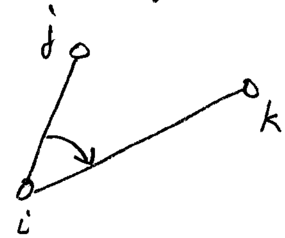


$$F_d = d_{ij} - \left[(x_j - x_i)^2 + (y_j - y_i)^2\right]^{1/2} = 0$$

write a matlab function to evaluate angle condition equation

$$\theta_{jik} = \alpha_{ik} - \alpha_{ij}$$

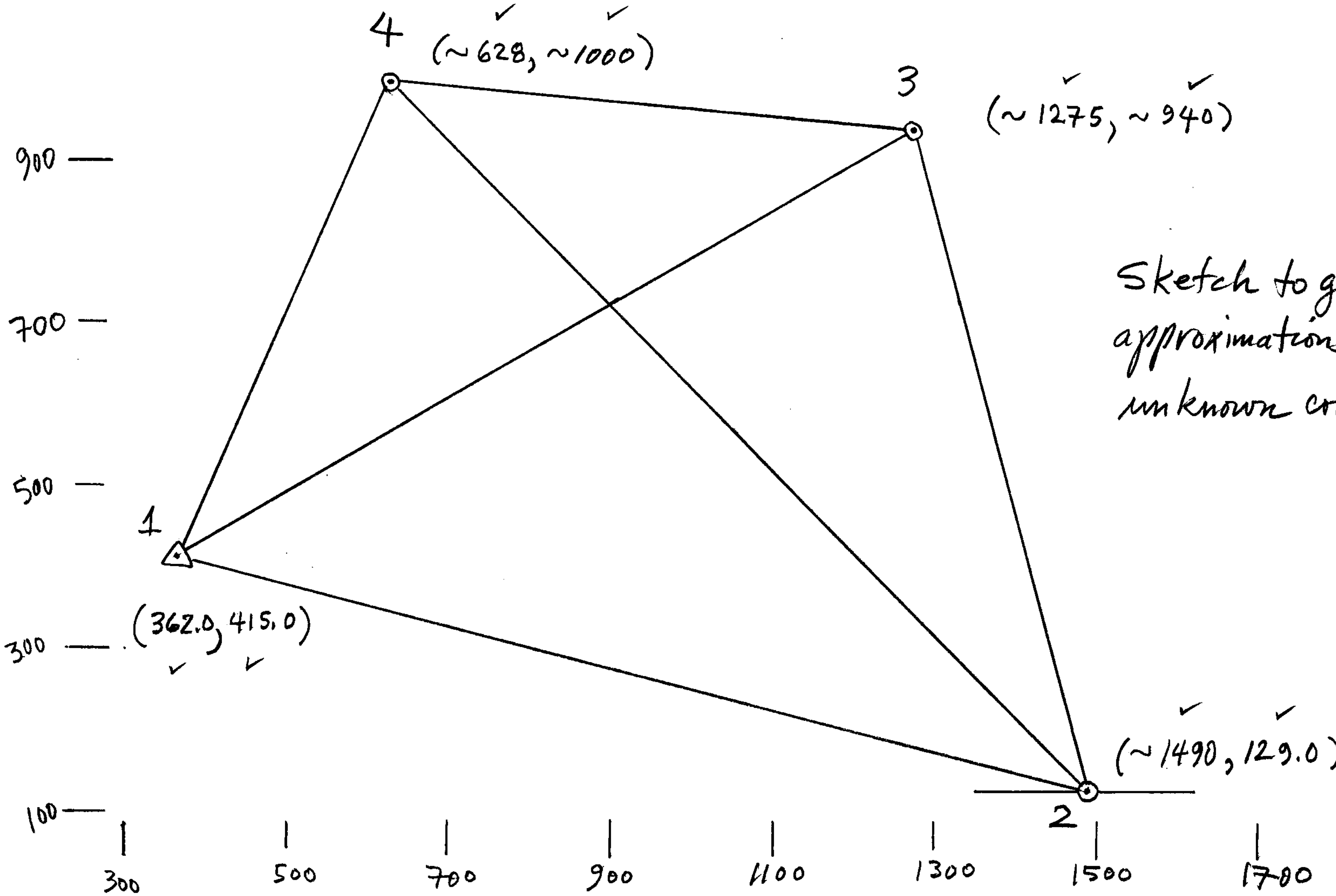
$$\alpha_{ij} = \text{azimuth from } i \rightarrow j$$



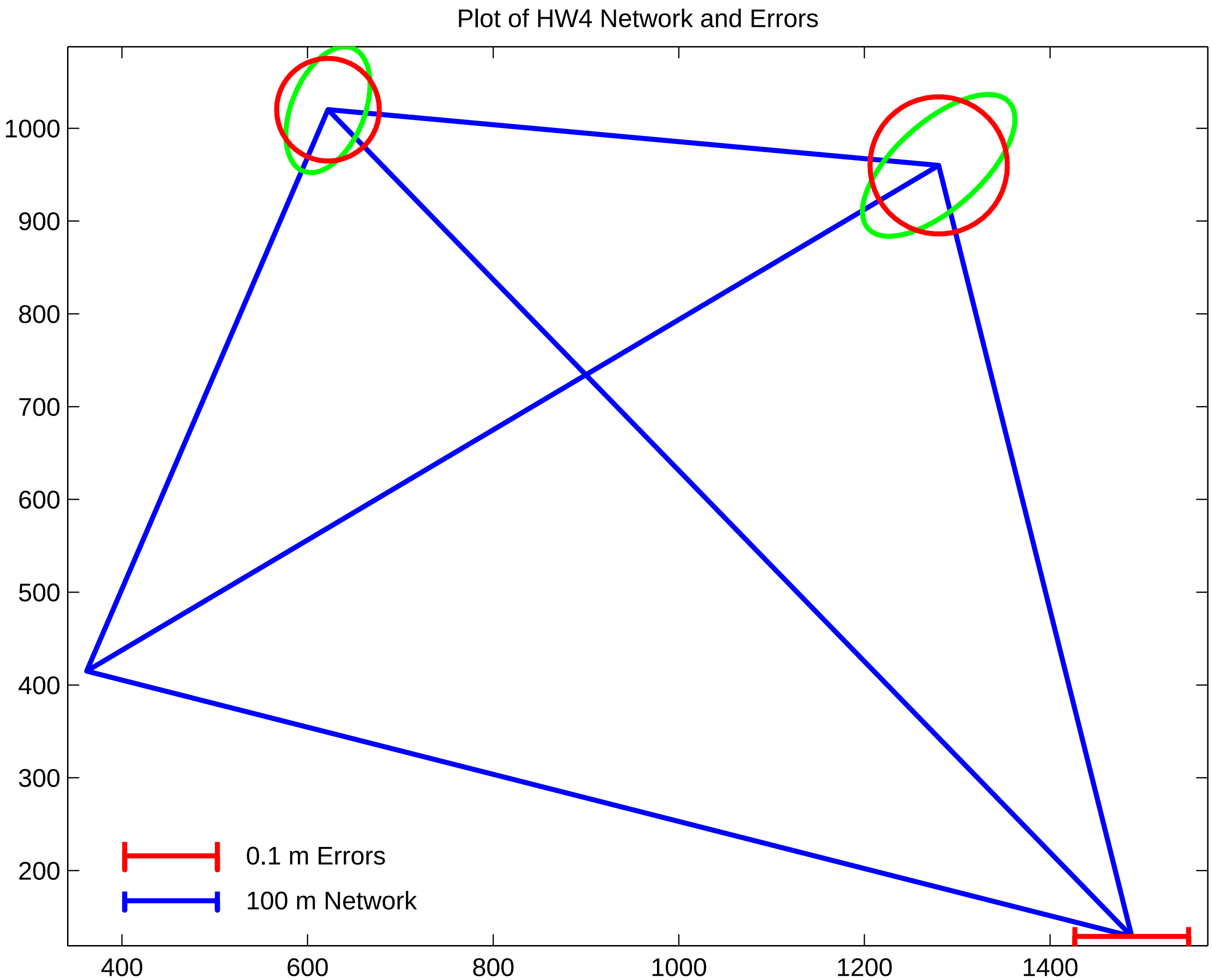
$$F_\theta = \theta_{jik} - (\alpha_{ik} - \alpha_{ij}) = 0$$

for both condition equations use partial derivatives as given in lecture <u>or</u> as given in chapter 10 of text (Same).

see accompanying Matlab <u>results</u> and <u>code</u>



Sketch to get initial approximations for the unknown coordinates


Plot of HW4 Network and Errors
1000
900
800
700
600
500
400
300
200
400
600
800
1000
1200
1400
0.1 m Errors
100 m Network

```
hw4_sol
iter =
     1
del2 =
         -1.89415684189726
        0.0706716863808907
          9.90762724808921
         0.160670064301669
          17.8901557133893
iter =
     2
del2 =
       -0.0770961849185834
       -0.0710482918856273
        0.0964279856432816
        -0.152823426638568
         0.118950398714267
iter =
     3
del2 =
     -8.0846609478936e-006
      -6.009063744709e-006
     9.78256042853996e-008
    -1.28880405901892e-005
     3.96893022911463e-006
iter =
     4
del2 =
    -2.15551351600252e-010
    -1.68691572267367e-010
    -6.29881741991511e-011
     3.10312010982249e-011
    -7.65696856632695e-011
convergence OK
v =
       -0.0182989896972755
         0.032176052428398
     1.77158384131451e-005
     3.36623489349303e-005
     1.59587444920282e-005
     3.44739411929667e-005
    -1.16140020500218e-005
     4.81454766477143e-006
    -1.24431114440428e-005
     -4.9981182258614e-006
vdms =
                         0
                         0
          3.65415397779224
vdms =
                         0
                         0
          6.94335788088555
vdms =
                         0
                         0
           3.2917273405951
vdms =
                         0
                         0
          7.11076080074107
vdms =
                         0
```

consistent with
StarNet v6 solution

```
                          0
         -2.39555988260112
vdms =
                          0
                          0
         0.993071741241489
vdms =
                          0
                          0
         -2.56657597111652
vdms =
                          0
                          0
         -1.03093588745738
final coordinates
P =
                        362
                        415
          1488.02873888831
                        129
          1279.99961738526
          960.004055331495
          622.007833749654
          1020.00911008096
post adj statistics - 2-sided F-test alpha 0.05
test_stat =
          1.51038001397748
cv1 =
         0.831211613486663
cv2 =
            12.83250199403
we pass
Sdd =
  Columns 1 through 3
                          0                          0                          0
                          0                          0                          0
                          0                          0        0.00825268523817354
                          0                          0                          0
                          0                          0        0.00500037010347704
                          0                          0        0.00535762814806167
                          0                          0       0.000831147666065795
                          0                          0        0.00453476129926905
  Columns 4 through 6
                          0                          0                          0
                          0                          0                          0
                          0        0.00500037010347704        0.00535762814806167
                          0                          0                          0
                          0        0.00487251922298519        0.00295652421177116
                          0        0.00295652421177116        0.00421964481275128
                          0        0.00161536406154951       0.000206593151309748
                          0        0.00332313980132341        0.00320959907218344
  Columns 7 through 8
      0.000831147666065796        0.00453476129926905
                          0                          0
                          0                          0
       0.00161536406154951        0.00332313980132341
      0.000206593151309748        0.00320959907218344
       0.00148438640298363       0.000895493931244048
      0.000895493931244048        0.00332362819454014
cov3 =
       0.00487251922298519        0.00295652421177116
       0.00295652421177116        0.00421964481275128
```

```
eigvec =
          0.744897834791554        -0.667178548608133
          0.667178548608133         0.744897834791554
eigval =
        0.00752057301023544                         0
                          0       0.00157159102550103
axlen3 =
          0.102106454039301                         0
axlen3 =
          0.102106454039301        0.0466764156361508
theta3 =
          0.730414629025776
CE_radius_3 =
         0.0739916387096958
eigvec =
         -0.926402324554458        -0.376535168424008
          0.376535168424008        -0.926402324554458
eigval =
        0.00112041391577531                         0
                          0       0.00368760068174846
axlen4 =
         0.0714989512592301                         0
axlen4 =
         0.0714989512592301        0.0394109565166791
theta4 =
          -1.95684967364084
CE_radius_4 =
         0.0553448800169284
half_intvl2 =
         0.0612735432579704
n =
    50
n =
    50
diary off
```

```
% hw4_sol.m  3-nov-08
% solve braced quad for hw4

X=[362;1490;1280;622];
Y=[415;129;950;1002];
P=[X(1); Y(1); X(2); Y(2); X(3); Y(3); X(4); Y(4)];
degrad=180/pi;
n=10;
n0=5;
r=5;
u=n0;
sigd=0.10;
siga=(10/3600)/degrad;
sd2=sigd^2;
sa2=siga^2;
sigma0_sqr=1;
w=[1/sd2 1/sd2 1/sa2 1/sa2 1/sa2 1/sa2 1/sa2 1/sa2 1/sa2 1/sa2];
W=diag(w);
d=[1161.80;660.69];
th=zeros(8,1);
th(1)=(44+56/60+50/3600)/degrad;
th(2)=(31+33/60+42/3600)/degrad;
th(3)=(30+07/60+48/3600)/degrad;
th(4)=(73+21/60+19/3600)/degrad;
th(5)=(35+54/60+30/3600)/degrad;
th(6)=(40+36/60+14/3600)/degrad;
th(7)=(67+26/60+31/3600)/degrad;
th(8)=(36+02/60+50/3600)/degrad;
% first 2 are distances, next 8 are angles
at=  [1 3 1 2 2 3 3 4 4 1];
from=[0 0 3 1 4 2 1 3 2 4];
to=  [2 4 2 4 3 1 4 2 1 3];
pidx=[1 3 5 7];

keep_going=1;
iter=0;
while(keep_going == 1)
  iter=iter+1
  B=zeros(n,u);
  f=zeros(n,1);
  rwidx=0;
  for i=1:2
    rwidx=rwidx+1;
    rs=distance2d(d(i),at(i),to(i),X,Y);
    B(rwidx,pidx(at(i)))  =rs(2);
    B(rwidx,pidx(at(i))+1)=rs(3);
    B(rwidx,pidx(to(i)))  =rs(4);
    B(rwidx,pidx(to(i))+1)=rs(5);
    f(rwidx)=-rs(1);
    end

  for i=1:8
    rwidx=rwidx+1;
    rs=angle2d(th(i),at(i+2),from(i+2),to(i+2),X,Y);
    B(rwidx,pidx(at(i+2)))    =rs(2);
    B(rwidx,pidx(at(i+2))+1)  =rs(3);
    B(rwidx,pidx(from(i+2)))  =rs(4);
    B(rwidx,pidx(from(i+2))+1)=rs(5);
    B(rwidx,pidx(to(i+2)))    =rs(6);
    B(rwidx,pidx(to(i+2))+1)  =rs(7);
    f(rwidx)=-rs(1);
    end
```

```
  B2=elim_col(B,[1 2 4]);
  N2=B2'*W*B2;
  t2=B2'*W*f;
  del2=inv(N2)*t2
  del=ins_zerv(del2,[1 2 4]);
  P=P+del;
  Ni2=inv(N2);
  Qdd2=Ni2;
  Qdd=ins_zerm(Qdd2,[1 2 4]);
  X=[P(1) P(3) P(5) P(7)];
  Y=[P(2) P(4) P(6) P(8)];
  if(all(abs(del2) < 0.00001))
    keep_going=0;
    disp('convergence OK');
    end
  if(iter > 10)
    keep_going=0;
    disp('failed to converge');
    end
  end

v=f-B2*del2
for i=3:10
  vdms=raddms(v(i))
  end

disp('final coordinates');
P

% ok now post-adjustment statistics

disp('post adj statistics - 2-sided F-test alpha 0.05');
test_stat=v'*W*v/1.0
cv1=icdf('chi2',0.025,r)
cv2=icdf('chi2',0.975,r)
if((test_stat > cv1) & (test_stat < cv2))
  pass=1;
  disp('we pass');
  Sdd=sigma0_sqr*Qdd
  % first point 3
  cov3=Sdd(5:6,5:6)
  [eigvec,eigval]=eig(cov3)
  major_axis=1;
  minor_axis=2;
  lambda1=eigval(1,1);
  lambda2=eigval(2,2);
  if(lambda2 > lambda1)
    major_axis=2;
    minor_axis=1;
    lambda1=eigval(2,2);
    lambda2=eigval(1,1);
    end
  axlen3=zeros(1,2);
  P=0.5;
  axlen3(1)=sqrt(eigval(major_axis,major_axis)*icdf('chi2',P,2))
  axlen3(2)=sqrt(eigval(minor_axis,minor_axis)*icdf('chi2',P,2))
  majr3=eigvec(:,major_axis)*axlen3(1);
  minr3=eigvec(:,minor_axis)*axlen3(2);
  theta3=atan2(majr3(2),majr3(1))
  % get 50% CE
  CE_radius_3=cep2(P,cov3)
  % next point 4
  cov4=Sdd(7:8,7:8);
```

```
  [eigvec,eigval]=eig(cov4)
  major_axis=1;
  minor_axis=2;
  lambda1=eigval(1,1);
  lambda2=eigval(2,2);
  if(lambda2 > lambda1)
    major_axis=2;
    minor_axis=1;
    lambda1=eigval(2,2);
    lambda2=eigval(1,1);
    end
  axlen4=zeros(1,2);
  P=0.5;
  axlen4(1)=sqrt(eigval(major_axis,major_axis)*icdf('chi2',P,2))
  axlen4(2)=sqrt(eigval(minor_axis,minor_axis)*icdf('chi2',P,2))
  majr4=eigvec(:,major_axis)*axlen4(1);
  minr4=eigvec(:,minor_axis)*axlen4(2);
  theta4=atan2(majr4(2),majr4(1))
  % get 50% CE
  CE_radius_4=cep2(P,cov4)
  % next confidence interval for X2
  varX2=Sdd(3,3);
  stdX2=sqrt(varX2);
  alpha=1-P;
  z=icdf('norm',1-alpha/2,0,1);
  half_intvl2=z*stdX2

else
  disp('we do not pass');
  pass=0;
  sigma0_sqr_hat=v'*W*v/r;
  Sdd=sigma0_sqr_hat*Qdd
  % first point 3
  cov3=Sdd(5:6,5:6)
  [eigvec,eigval]=eig(cov3)
  major_axis=1;
  minor_axis=2;
  lambda1=eigval(1,1);
  lambda2=eigval(2,2);
  if(lambda2 > lambda1)
    major_axis=2;
    minor_axis=1;
    lambda1=eigval(2,2);
    lambda2=eigval(1,1);
    end
  axlen3=zeros(1,2);
  P=0.5;
  axlen3(1)=sqrt(eigval(major_axis,major_axis)*2*icdf('f',P,2,r))
  axlen3(2)=sqrt(eigval(minor_axis,minor_axis)*2*icdf('f',P,2,r))
  majr3=eigvec(:,major_axis)*axlen3(1);
  minr3=eigvec(:,minor_axis)*axlen3(2);
  theta3=atan2(majr3(2),majr3(1))
  % get 50% CE
  CE_radius_3=cep2(P,cov3)
  % next point 4
  cov4=Sdd(7:8,7:8);
  [eigvec,eigval]=eig(cov4)
  major_axis=1;
  minor_axis=2;
  lambda1=eigval(1,1);
  lambda2=eigval(2,2);
  if(lambda2 > lambda1)
    major_axis=2;
```

```
    minor_axis=1;
    lambda1=eigval(2,2);
    lambda2=eigval(1,1);
    end
  axlen4=zeros(1,2);
  P=0.5;
  axlen4(1)=sqrt(eigval(major_axis,major_axis)*2*icdf('f',P,2,r))
  axlen4(2)=sqrt(eigval(minor_axis,minor_axis)*2*icdf('f',P,2,r))
  majr4=eigvec(:,major_axis)*axlen4(1);
  minr4=eigvec(:,minor_axis)*axlen4(2);
  theta4=atan2(majr4(2),majr4(1))
  % get 50% CE
  CE_radius_4=cep2(P,cov4)
  % next confidence interval for X2
  varX2=Sdd(3,3);
  stdX2=sqrt(varX2);
  alpha=1-P;
  tt=icdf('t',1-alpha/2,r);
  half_intvl2=ttstdX2
  end

% plot the network

plot([X(1) X(2) X(3) X(4) X(1) X(3)],[Y(1) Y(2) Y(3) Y(4) Y(1)
Y(3)],'b-','linewidth',2);
hold on
plot([X(2) X(4)],[Y(2) Y(4)],'b-','linewidth',2);
err_factor=1000;
% plot errors point 3
a=axlen3(1)*err_factor;
b=axlen3(2)*err_factor;
theta=theta3;
rd=CE_radius_3*err_factor;
rs=draw_ell(X(3),Y(3),a,b,theta);
rs=draw_cir(X(3),Y(3),rd);
% plot errors point 4
a=axlen4(1)*err_factor;
b=axlen4(2)*err_factor;
theta=theta4;
rd=CE_radius_4*err_factor;
rs=draw_ell(X(4),Y(4),a,b,theta);
rs=draw_cir(X(4),Y(4),rd);
halfbar=half_intvl2*err_factor;
tick=10;
barx1=X(2) - halfbar;
barx2=X(2) + halfbar;
bary1=Y(2);
bary2=Y(2);
pxvec=[barx1 barx1 barx1 barx2 barx2 barx2];
pyvec=[bary1+tick bary1-tick bary2 bary2 bary2-tick bary2+tick];
plot(pxvec,pyvec,'r-','linewidth',2);

% proper aspect ratio
axis equal

lmt=axis;
xrange=lmt(2)-lmt(1);
yrange=lmt(4)-lmt(3);
x5pct=0.05*xrange;
y5pct=0.05*yrange;
% network scale bar
barx1=lmt(1) + x5pct;
bary1=lmt(3) + y5pct;
```

```
barx2=barx1 + 100;
bary2=bary1;
tick=10;
pxvec=[barx1 barx1 barx1 barx2 barx2 barx2];
pyvec=[bary1+tick bary1-tick bary2 bary2 bary2-tick bary2+tick];
plot(pxvec,pyvec,'b-','linewidth',2);
textx=barx2 + 0.5*x5pct;
texty=bary2;
text(textx,texty,'100 m Network');
% error scale bar
barx1=lmt(1) + x5pct;
bary1=lmt(3) + 2*y5pct;
barx2=barx1 + 0.1*err_factor;
bary2=bary1;
tick=15;
pxvec=[barx1 barx1 barx1 barx2 barx2 barx2];
pyvec=[bary1+tick bary1-tick bary2 bary2 bary2-tick bary2+tick];
plot(pxvec,pyvec,'r-','linewidth',2);
textx=barx2 + 0.5*x5pct;
texty=bary2;
text(textx,texty,'0.1 m Errors');

title('Plot of HW4 Network and Errors');
```

```
% angle2d.m  3-nov-08
function result = angle2d(a,i,j,k,X,Y)
xi=X(i);
yi=Y(i);
xj=X(j);
yj=Y(j);
xk=X(k);
yk=Y(k);
Dij_sq=(xj-xi)^2 + (yj-yi)^2;
Dik_sq=(xk-xi)^2 + (yk-yi)^2;
dFdxi= (yk-yi)/Dik_sq - (yj-yi)/Dij_sq;
dFdyi=-(xk-xi)/Dik_sq + (xj-xi)/Dij_sq;
dFdxj=(yj-yi)/Dij_sq;
dFdyj=-(xj-xi)/Dij_sq;
dFdxk=-(yk-yi)/Dik_sq;
dFdyk=(xk-xi)/Dik_sq;
ac=atan2(xk-xi,yk-yi) - atan2(xj-xi,yj-yi);
if(ac < 0)
  ac=ac + 2*pi;
  end

% ac
% degrad=180/pi;
% ac*degrad

Fa=a - ac;
result=[Fa dFdxi dFdyi dFdxj dFdyj dFdxk dFdyk];
```

```
% cep2.m 11-nov-04
% for given 2x2 covariance and probability P,
% compute radius yielding P under bivariate normal
% syntax res=cep2(P,cov);
% original in d:\classes\ce603_03\

function res=cep2(P,cov)
sx2=cov(1,1);
sy2=cov(2,2);
sxy=cov(1,2);
sx=sqrt(sx2);
sy=sqrt(sy2);
long=max([sx sy]);
dr=long/50;
t1=2*pi*sqrt(det(cov));
term1=1/t1;
covi=inv(cov);
X=zeros(2,1);
degrad=180/(pi);
dth=1/degrad;
nth=180;
accumP=0;
rr=0;
while(accumP < 0.5*P)
rp=rr + 0.5*dr;
tt=0;
for j=1:nth
thp=tt + 0.5*dth;
X(1)=rp*cos(thp);
X(2)=rp*sin(thp);
term2=-0.5*(X'*covi*X);
f=term1*exp(term2);
dens=f;
%mu=[0 0];
%XX=[X(1) X(2)];
%dens=mvnpdf(XX,mu,cov);
da=rp*dth*dr;
accumP=accumP + da*dens;
tt=tt + dth;
end
rr=rr + dr;
end
res=rr;
```

```
% distance2d.m  3-nov-08
function result = distance2d(d,i,j,X,Y)
xi=X(i);
yi=Y(i);
xj=X(j);
yj=Y(j);
Dij=sqrt((xj-xi)^2 + (yj-yi)^2);
dFdxi=(xj-xi)/Dij;
dFdyi=(yj-yi)/Dij;
dFdxj=-dFdxi;
dFdyj=-dFdyi;
Fd=d - Dij;
result=[Fd dFdxi dFdyi dFdxj dFdyj];
```

```
% draw_cir.m  13-oct-08
function result=draw_cir(x0,y0,r)
xi=x0+r;
yi=y0;
n=50
degrad=180/pi;
dth=2*pi/n;
rth=0;
for i=1:n
  rth=rth+dth;
  costh=cos(rth);
  sinth=sin(rth);
  xip1=x0 + r*costh;
  yip1=y0 + r*sinth;
  plot([xi xip1],[yi yip1],'r','linewidth',2);
  if(i==2)
    hold on
    end
  xi=xip1;
  yi=yip1;
  end
result=0;
```

```
% draw_ell.m  22-oct-08
% function to draw ellipse

function result=draw_ell(xorg,yorg,a,b,theta)

th=theta;
x0=a;
y0=0;
nseg=50;
dalpha=2*pi/nseg;
for i=1:nseg
  alpha=i*dalpha;
  x1=a*cos(alpha);
  y1=b*sin(alpha);
  px0=xorg + cos(th)*x0 - sin(th)*y0;
  py0=yorg + sin(th)*x0 + cos(th)*y0;
  px1=xorg + cos(th)*x1 - sin(th)*y1;
  py1=yorg + sin(th)*x1 + cos(th)*y1;
  plot([px0 px1],[py0 py1],'-g','linewidth',2);
  if(i == 1)
    hold on
    end
  x0=x1;
  y0=y1;
  end
result=0;
```

```
% 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_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
```

```
% 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
```

```
% raddms.m  25-sep-08
% function to convert an angle in radians
% to degrees, minutes, and seconds

function dms=raddms(angrad)
degrad=180/pi;
angdeg=angrad*degrad;
deg=fix(angdeg);
frac=angdeg-deg;
rmin=frac*60;
min=fix(rmin);
frac=rmin-min;
sec=frac*60;
dms=[deg;min;sec];
```