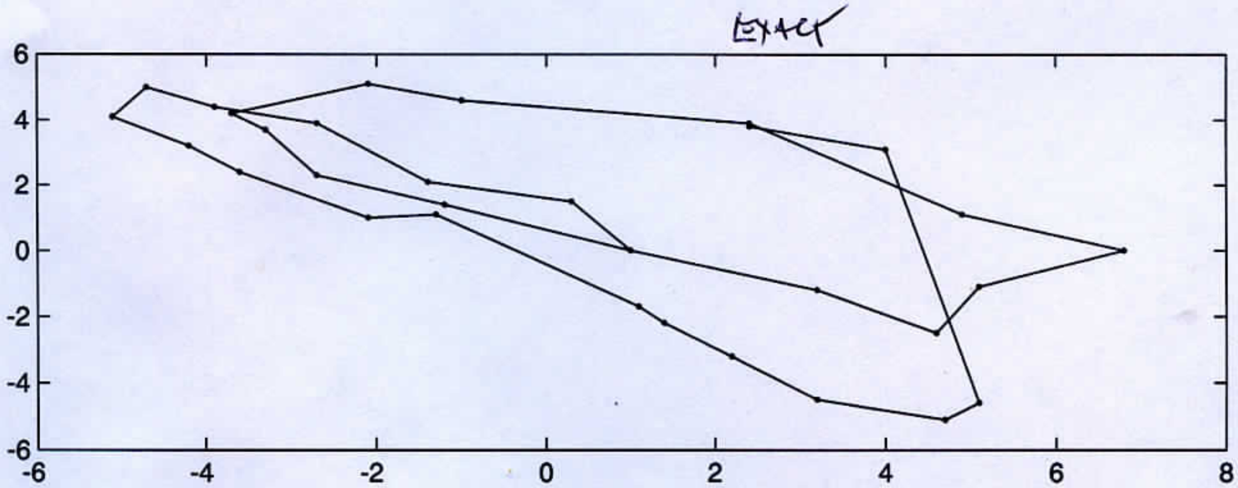
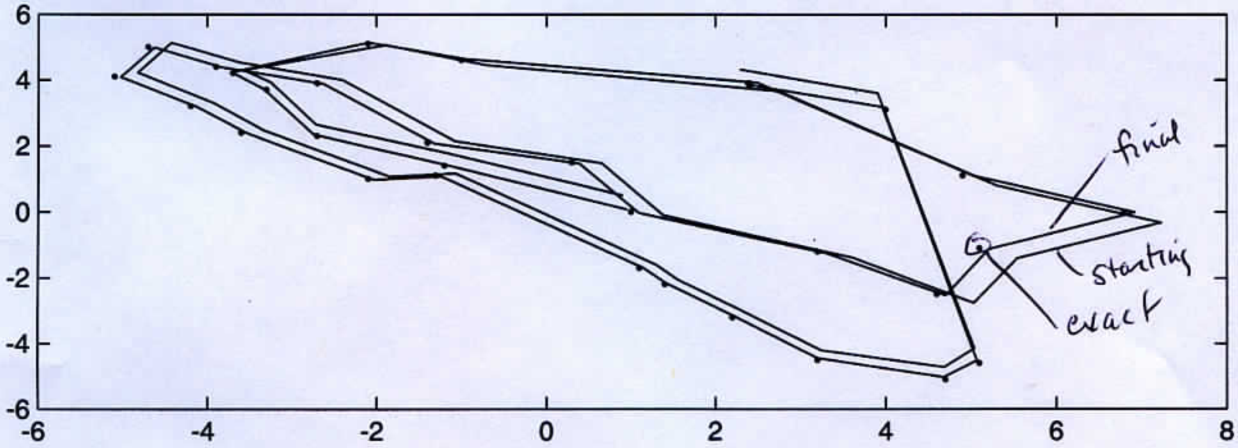
w/ Kori Newman
10-21-04
least-squares fitting of ship track given various navigational data

MRN041





```
clear

hold off;
subplots=2;

% weights
WR = 1;
WV = 1;
WH = 1;
WQ = 1;
epsilon2=100.0;
iterations=10;
smoothness=(-2);

% the true AUV track
dt=1.0;
n = [ 0.0,  1.4,  2.3,  3.7,  4.2,  5.1,  4.6,  3.9, 1.1,  0.0, -1.1, -2.5, -1.2,  0.0,  ↙
1.5,  2.1,  3.9,  4.4,  5.0,  4.1,  3.2,  2.4,  1.0,  1.1, -1.7, -2.2, -3.2, -4.5, -5.1,  ↙
-4.6, 3.1, 3.8]';
e = [ 1.0, -1.2, -2.7, -3.3, -3.7, -2.1, -1.0,  2.4,  4.9,  6.8,  5.1,  4.6,  3.2,  1.0,  ↙
 0.3, -1.4, -2.7, -3.9, -4.7, -5.1, -4.2, -3.6, -2.1, -1.3,  1.1,  1.4,  2.2,  3.2,  4.7, ↙
  5.1, 4.0, 2.4 ]';
N = length(n);
t=dt*[0:N-1]';

% the ship track; and the exact ship-to-AUV distance, R
tr = [   1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 2 ↙
1, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32 ]';
nr = [   1,  1,  1,  1,  1,  1,  2,  2,  2,  2,  2,  2,  3,  3,  3,  3,  3,  4,  4,  4,   ↙
4,  3,  3,  3,  3,  3,  3,  2,  2,  2,  2,  2 ]';
er = [   1,  1,  1,  1,  2,  2,  2,  2,  2,  2,  3,  3,  3,  2,  2,  3,  3,  4,  4,  4,   ↙
3,  2,  2,  1,  3,  3,  3,  2,  2,  2,  2,  2 ]';
Nr=length(tr);
R = zeros(Nr,1);
Tr = zeros(Nr,1);
for i=[1:Nr]
    j=tr(i);
    Tr(i)=t(j);
    R(i) = sqrt( (n(j)-nr(i)) * (n(j)-nr(i)) + (e(j)-er(i)) * (e(j)-er(i)) );
end

% plot the true AUV track
subplot(subplots,1,1);
plot(e,n,'r.');
hold on;
plot(e,n,'r');

% plot the true AUV track
subplot(subplots,1,2);
plot(e,n,'r.');
hold on;

% various constants related to fourier series
% the north(time) and east(time) position of the AUV are each expanded in
% fourier series
fny=1.0/(2.0*dt);
wny= 2*pi*fny;
```

```
df=fny/(N/2);
dw=2*pi*df;
w=[0:dw:wny]';

% matrices of sines and cosines that form the kernel of the fourier
% expansions; the A matrix gives position and the Av matrix gives velocity
A=zeros(N,N);
Av=zeros(N,N);
WM = zeros(N,1);
c = 1;
s = 0;
WM(1)=exp(smoothness*w(1));
A(:,1) = c;
Av(:,1) = s;
for i=[2:2:N-1]
    c = cos( w(i/2+1) .* t );
    s = sin( w(i/2+1) .* t );
    A(:,i)   = c;
    A(:,i+1) = s;
    Av(:,i)   = w(i/2+1)*(-s);
    Av(:,i+1) = w(i/2+1)*c;
    WM(i)=exp(smoothness*w(i/2+1));
    WM(i+1)=WM(i);
end
c = cos( w(N/2+1) .* t );
s = sin( w(N/2+1) .* t );
A(:,N) = c;
Av(:,N) = w(N/2+1)*(-s);
WM(N) = exp(smoothness*w(N/2+1));

% solve for the true fourier coefficeints of the AUV track
ATA = ones(N,1)*(N/2);
ATA(1)=N;
ATA(N)=N;
an = A' * n;
an = an ./ ATA;
ae = A' * e;
ae = ae ./ ATA;
% evaluate the fourier series; this is mainly for error-checking
% npre had better equal n, epre had better equal n
npre = A*an;
epre = A*ae;

% predicted velocity
vnpre=Av*an;
vepre=Av*ae;
vspre = sqrt( vnpre .* vnpre + vepre .* vepre );

% predicted heading
hpre = (180.0/pi)*atan2( vnpre, vepre );

% velocity data, the times at which the scaler velocity of the AUV is known
% the veloicity is initially set to the true velocity
tv = [   1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 2 ↙
1, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32 ]';
Nv=length(tv);
v = zeros(Nv,1);
```

```
Tv = zeros(Nv,1);
for i=[1:Nv]
    j=tv(i);
    Tv(i)=t(j);
    v(i) = vspre(j);
end

% heading data, the times at which the heading of the AUV is known
% the heading is initially set to the true heading
th = [  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 2 ↙
1, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32 ]';
Nh=length(tv);
h = zeros(Nh,1);
Th = zeros(Nh,1);
for i=[1:Nh];
    j=th(i);
    Th(i)=t(j);
    h(i) = hpre(j);
end

% generalized crossing data, a pair of times where the distance between the two
% positions of the AUV is knows (for a crossing, that distance would be
% zero)
tq1 = [  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, ↙
21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32 ]';
tq2 = [ 32, 31, 30, 29, 28, 27, 26, 25, 24, 23, 22, 21, 20, 19, 18, 17, 16, 15, 14, 13, ↙
12, 11, 10,  9,  8,  7,  6,  5,  4,  3,  2,  1 ]';
Nq=length(tq1);
q = zeros(Nq,1);
Tq1 = zeros(Nq,1);
Tq2 = zeros(Nq,1);
for i=[1:Nq]
    j1=tq1(i);
    j2=tq2(i);
    Tq1(i)=t(j1);
    Tq2(i)=t(j2);
    q(i)=sqrt( (n(j1)-n(j2))^2 + (e(j1)-e(j2))^2 );
end

% linearized least squares estimate of fourier coefficients of AUV
% track, given R, v, h, q data.  The equation is of the form is Gm=d
% d ordered R then v then h then q
% m ordered an then ae
NG = Nr+Nv+Nh+Nq;
MG = 2*N;
G=zeros(NG,MG);
d=zeros(NG,1);
m=zeros(MG,1);


% initial guess for (an, ae); slightly perturbed version of exact values
ang = an;
ang(1)=ang(1)+0.1;
ang(2)=ang(2)+0.2;
ang(3)=ang(3)-0.2;
ang(4)=ang(4)+0.2;
aeg = ae;
```

```
aeg(1)=aeg(1)+0.2;
aeg(2)=aeg(2)-0.2;
aeg(3)=aeg(3)+0.1;
aeg(4)=aeg(4)-0.1;

% iterative improvement; each iteration solves Gm=d to improve initial
% guess of fourier coefficients
for iter=[1:iterations]

% the consequent values of n, e, R and vs, given this initial guess (ang, aeg)
ng = A*ang;
eg = A*aeg;
if( iter==1 )
    plot(eg,ng,'b');
end
Rg = zeros(Nr,1);
for i=[1:Nr];
    j=tr(i);
    Rg(i) = sqrt( (ng(j)-nr(i)) * (ng(j)-nr(i)) + (eg(j)-er(i)) * (eg(j)-er(i)) );
end
vng=Av*ang;
veg=Av*aeg;
vsg = sqrt( vng .* vng + veg .* veg );
hg = (180.0/pi)*atan2(vng,veg);
qg=zeros(Nq,1);
for i=[1:Nq]
    j1=tq1(i);
    j2=tq2(i);
    qg(i) = sqrt( (ng(j1)-ng(j2))^2 + (eg(j1)-eg(j2))^2 );
end

% save initial values
if( iter==1 )
    ngo=ng;
    ego=eg;
    Rgo=Rg;
    vsgo=vsg;
    hgo=hg;
    qgo=qg;
end

% rows of G associated with R data
% R = [ (n-n0)^2 + (e-e0)^2 ] ^ 1/2, so
% dR/dan = (n-n0) dn/dan / R  and  dR/dae =(e-e0) de/dae / R
% and dn/dan and de/dae are just elements of A
for i=[1:Nr]
    k=tr(i);
    for j=[1:N]
        G(i,j)   = WM(j) * WR * (ng(k)-nr(i)) * A(k,j) / Rg(i);
        G(i,j+N) = WM(j) * WR * (eg(k)-er(i)) * A(k,j) / Rg(i);
    end
    d(i) = WR * (R(i)-Rg(i));
end

% rows of G associated with v data
% v = [ vn * vn + ve * dve ] ^ 1/2, so
% dv/dan = vn dvn/dan / v  and  dv/dae = ve dve/dae / v
```

```
% and dvn/dan and dve/dae are just elements of Av
for i=[1:Nv]
    k=tv(i);
    for j=[1:N]
        G(i+Nr,j)      = WM(j) * WV * vng(k) * Av(k,j) / vsg(k);
        G(i+Nr,j+N)    = WM(j) * WV * veg(k) * Av(k,j) / vsg(k);
    end
    d(i+Nr)=WV*(v(i)-vsg(k));
end

% rows of G associated with h data
% h = (180/pi) * atan( vn / ve ), so
% dh/dan = 1/(1+(vn/ve)^2) * 1/ve * dvn/dan
% dh/dae = 1/(1+(vn/ve)^2) * (-vn/ve^2) * dve/dae
% and dvn/dan and dve/dae are just elements of Av
for i=[1:Nh]
    k=th(i);
    for j=[1:N]
        tmp = WM(j)*WH*(180/pi)/(1+(vng(k)/veg(k))^2);
        G(i+Nr+Nv,j)      = tmp * (1/veg(k)) * Av(k,j);
        G(i+Nr+Nv,j+N)    = tmp * (-vng(k)/(veg(k)^2)) * Av(k,j);
    end
    d(i+Nr+Nv)=WH*(h(i)-hg(k));
end

% rows of G associated with q data
% q = [ (n(t1)-n(t2))^2 + (e(t1)-e(t2))^2 ] ^ 1/2
% dq/dan = (n(t1)-n(t2)) ( dn(t1)/dan - dn(t2)/dan ) / q
% dq/dae = (e(t1)-e(t2)) ( de(t1)/dan - de(t2)/dan ) / q
% and dn/dan and de/dae are just elements of A
for i=[1:Nq]
    k1=tq1(i);
    k2=tq2(i);
    for j=[1:N]
        tmp = WM(j)*WQ/qg(i);
        G(i+Nr+Nv+Nh,j)      = tmp * (ng(k1)-ng(k2)) * (A(k1,j)-A(k2,j));
        G(i+Nr+Nv+Nh,j+N)    = tmp * (eg(k1)-eg(k2)) * (A(k1,j)-A(k2,j));
    end
    d(i+Nr+Nv+Nh)=WQ*(q(i)-qg(i));
end

% damped least squares
GTG = G'*G;
for i=[1:MG]
    GTG(i,i) = GTG(i,i) + epsilon2;
end

% least sqaures solution and updating of initial guess
m = inv(GTG)*(G'*d);
ang = ang + WM .* m(1:N);
aeg = aeg + WM .* m(N+1:2*N);

% mean-squared error in data;
dres = d'*d;

end
```

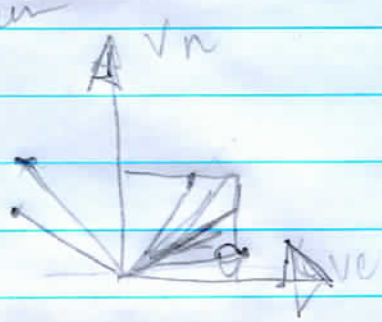
```
ng = A*ang;
eg = A*aeg;
plot(eg,ng,'r');

% mean-squared error in various things
% (we only ares, nres and eres because in this test we assume we know the true track)
ares = (an-ang)'*(an-ang)+(ae-aeg)'*(ae-aeg)
nres = (n-ng)'*(n-ng)
eres = (e-eg)'*(e-eg)
Rres = (d(1:Nr)'*d(1:Nr))/(WR^2)
vres = (d(Nr+1:Nr+Nv)'*d(Nr+1:Nr+Nv))/(WV^2)
hres = (d(Nr+Nv+1:Nr+Nv+Nh)'*d(Nr+Nv+1:Nr+Nv+Nh))/(WH^2)
qres = (d(Nr+Nv+Nh+1:Nr+Nv+Nh+Nq)'*d(Nr+Nv+Nh+1:Nr+Nv+Nh+Nq))/(WQ^2)
```

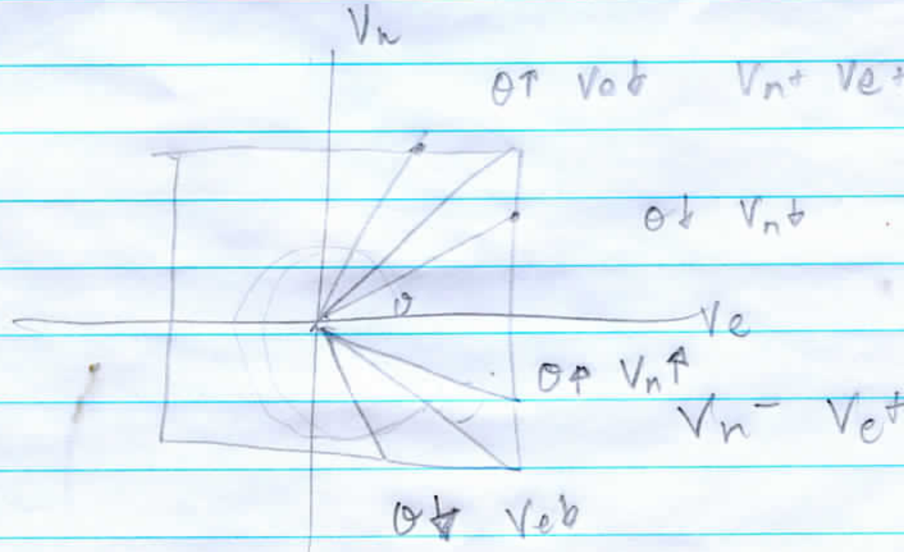
$$\theta = \tan^{-1}(v_n / v_e) \qquad \frac{d}{dx}\tan^{-1}u = \frac{1}{1+u^2}\frac{du}{dx}$$

$$\frac{\partial\theta}{\partial a_n} = \frac{1}{1+\left(\frac{v_n}{v_e}\right)^2}\,\frac{1}{v_e}\,\frac{dv_n}{da_n}$$

$$\frac{\partial\theta}{\partial a_e} = \frac{1}{1+\left(\frac{v_n}{v_e}\right)^2}\left(-\frac{v_n}{v_e^2}\right)\frac{\partial v_e}{\partial a_e}$$

at a certain time the

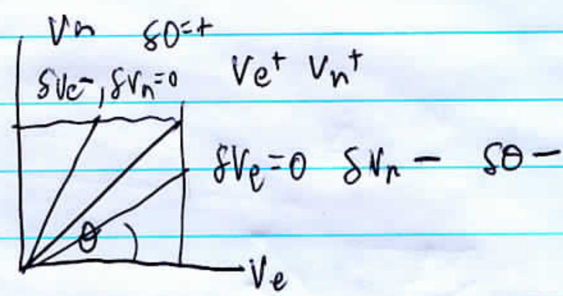


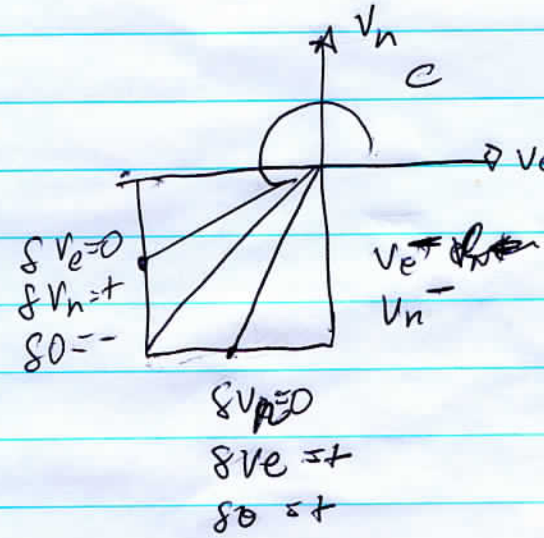


$$\frac{\partial\theta}{\partial a_n} \propto \frac{1}{V_e}\frac{\partial V_n}{\partial a_n} \qquad \frac{\partial\theta}{\partial a_e} \propto -V_n \frac{\partial V_e}{\partial a_e}$$

	$\frac{\partial\theta}{\partial a_n}$		$\frac{1}{V_e}$	$\frac{\partial V_n}{\partial a_n}$	$\frac{\partial\theta}{\partial a_e}$		$-$	V_n	$\frac{\partial V_e}{\partial a_e}$
A	−	✓	+	−	+	✓	−	+	−
B	+	✓	+	+	−	✓	−	−	−
C	−	✓	−	+	+	✓	−	[illegible]	+
D	+	✓	−	−	−	✓	−	+	+

A





B