Contents

set 4, prob 4

```
clear; clc

% loads rhoahist, rhobhist, and thist
% load radarmeasdata_missle.mat
load radarmeasdata_missile_new.mat

global la lb
la = 3.5e5;
lb = 4.0e5;

% meas error covariances
sigma_rhoa = 10;
sigma_rhob = 30;
sigma_thetaa = 0.01;
sigma_thetab = 0.03;

% measurement data case
meas_data = 'both';

switch meas_data

    case 'rho'

        R_j = [sigma_rhoa^2 0; 0 sigma_rhob^2];

        % Build full R matrix
        R = zeros(length(thist));
        for j = 1:length(thist)
            R(2*j-1 : 2*j, 2*j-1 : 2*j) = R_j;
        end

        % build full zhist
        zhist = [];
        for j = 1:length(thist)
            zhist = [ zhist; rhoahist(j); rhobhist(j) ];
        end

    case 'theta_a'

        R_aj = zeros(1);
        R_aj(1,1) = sigma_thetaa^2;

        % Build full R matrix
        R = zeros(length(thist));
        for j = 1:length(thist)
            R(j,j) = R_aj;
        end

        % build full zhist
        zhist = [];
        for j = 1:length(thist)
            zhist = [ zhist; thetaahist(j) ];
        end

    case 'both'

        R_aj = zeros(4);
        R_aj(1,1) = sigma_rhoa^2;
        R_aj(2,2) = sigma_thetaa^2;
        R_aj(3,3) = sigma_rhob^2;
        R_aj(4,4) = sigma_thetab^2;

        % Build full R matrix
        R = zeros(length(thist));
        for j = 1:length(thist)
            R(4*j-3 : 4*j, 4*j-3 : 4*j) = R_aj;
        end
        % build full zhist
        zhist = [];
        for j = 1:length(thist)
            zhist = [ zhist; rhoahist(j); thetaahist(j); rhobhist(j); thetabhist(j) ];
        end
```

```
end

Ra = chol(R);
```

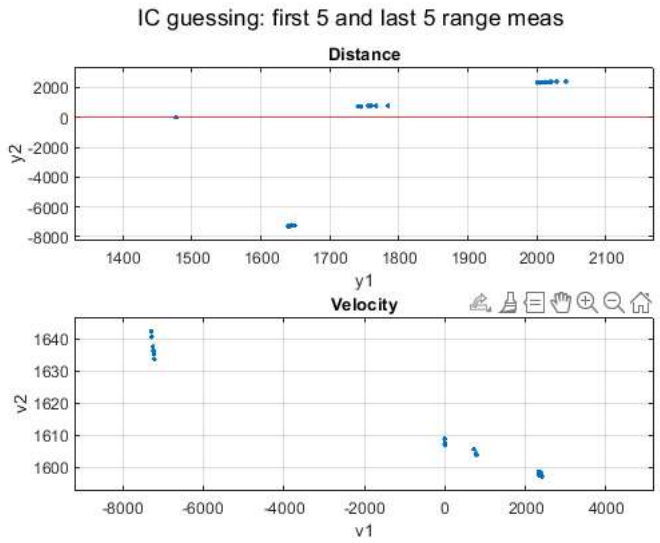
Initial condition guessing

```
xg0_arr = [];
% First guess
for i = 1:5
    for f = 23:28
        sprintf('i = %d, f = %d', i, f)
        xg0 = find_xg0(rhoahist, rhobhist, thist, i, f);
        xg0_arr = [xg0_arr; xg0'];
    end
end

y = [xg0_arr(:,1), xg0_arr(:,3)];
ftitle = 'IC guessing: first 5 and last 5 range meas';
figure('name', ftitle);
    subplot(2,1,1)
        plot(y(:,1), y(:,2),'.')
        grid on; hold on;
        yline(0, 'r')
        xlabel('y1'); ylabel('y2');
        bigger_ylim; bigger_xlim
        title('Distance')
    subplot(2,1,2)
        plot(xg0_arr(:,3), xg0_arr(:,4), '.');
        grid on; hold on;
        xlabel('v1'); ylabel('v2');
        bigger_ylim; bigger_xlim
        title('Velocity')
    sgtitle(ftitle);
```

```
ans =

    'i = 1, f = 23'


ans =

    'i = 1, f = 24'


ans =

    'i = 1, f = 25'


ans =

    'i = 1, f = 26'


ans =

    'i = 1, f = 27'


ans =

    'i = 1, f = 28'


ans =

    'i = 2, f = 23'


ans =

    'i = 2, f = 24'


ans =

    'i = 2, f = 25'


ans =

    'i = 2, f = 26'


ans =

    'i = 2, f = 27'
```

```
ans =

    'i = 2, f = 28'


ans =

    'i = 3, f = 23'


ans =

    'i = 3, f = 24'


ans =

    'i = 3, f = 25'


ans =

    'i = 3, f = 26'


ans =

    'i = 3, f = 27'


ans =

    'i = 3, f = 28'


ans =

    'i = 4, f = 23'


ans =

    'i = 4, f = 24'


ans =

    'i = 4, f = 25'


ans =

    'i = 4, f = 26'


ans =

    'i = 4, f = 27'


ans =

    'i = 4, f = 28'


ans =

    'i = 5, f = 23'


ans =

    'i = 5, f = 24'


ans =

    'i = 5, f = 25'


ans =

    'i = 5, f = 26'


ans =

    'i = 5, f = 27'
```

```
ans =

    'i = 5, f = 28'

Warning: Imaginary parts of complex X and/or Y arguments ignored.
Warning: Imaginary parts of complex X and/or Y arguments ignored.
```



this one looks good

```
xg0_OG = find_xg0(rhoahist, rhobhist, thist, 3, 25)
xg0 = xg0_OG;
```

```
xg0_OG =

          2019.29620390645
           899.18143927834
          2349.46128871635
          1597.49910053977
```

Jacobian H

```
x = sym('x', [4 1]);
syms la_sym lb_sym tj g

y1 = x(1) + x(2)*tj;
dy_1a = la_sym - y1;
dy_1b = lb_sym - y1;
dy_2 = x(3) + tj * x(4) - 4.9*tj^2;

h_rhoa   = sqrt( dy_1a^2 + dy_2^2 );
h_rhob   = sqrt( dy_1b^2 + dy_2^2 );
h_thetaa = atan2( dy_2, dy_1a );
h_thetab = atan2( dy_2, dy_1b );

% inputs: la, lb, tj, x1, x2, x3, x4

switch meas_data

    case 'rho'
        Hhist_j = matlabFunction( [ jacobian(h_rhoa, x); jacobian(h_rhob, x) ] );

    case 'theta_a'
        Hhist_j = matlabFunction( [ jacobian(h_thetaa, x) ] );

    case 'both'
        Hhist_j = matlabFunction( [ jacobian(h_rhoa, x); jacobian(h_thetaa, x); jacobian(h_rhob, x); jacobian(h_thetab, x) ] );

end

% if no symbolic toolbox - here is Hhist_j copied from comand window
% Hhist_j = @(la_sym,lb_sym,tj,x1,x2,x3,x4)reshape([(1.0./sqrt((-la_sym+x1+tj.*x2).^2+(x3+tj.*x4-tj.^2.*(4.9e+1./1.0e+1)).^2).*(la_sym.*-2.0+x1.*2.0+tj.*x2.*2.0))./
```

First cost function

```
[Jg, h, H, dx] = cost_fn(xg0, thist, zhist, Ra, Hhist_j, meas_data);
```

```
% first a step
a = 1;

% First step-size adjusted cost function
xg = xg0 + a * dx;
[Jgnew, h, H, ~] = cost_fn(xg, thist, zhist, Ra, Hhist_j, meas_data);

% Gauss-Newton dx
% dx = inv((H' * H)) * H' * (z - h);
```

The while loop: Jgnew > Jg

```
Jg_i = [];
while norm(dx) > 0.0000001
```

```
    while Jgnew >= Jg

        % Next a
        a = a/2;
        if a < 0.001
            break; end

        % Step size-adjusted guess and cost fn
        xg = xg0 + a * dx;
        [Jgnew, h, H, ~] = cost_fn(xg, thist, zhist, Ra, Hhist_j, meas_data);

    end
```

While loop: "New" first guess - saved from last iteration

```
%     if a < eps
%         break; end

    xg0 = xg;
    Jg  = Jgnew;

    % Gauss-Newton dx (H, z, and h saved from last iteration)
    z  = inv(Ra') * zhist;
    dx = inv((H' * H)) * H' * (z - h);

    % first a step
    a = 1;

    % "new" step-size adjusted guess
    xg = xg0 + a * dx;
    [Jgnew, h, H, ~] = cost_fn(xg, thist, zhist, Ra, Hhist_j, meas_data);

    Jg_i = [Jg_i; Jg];
```

```
end

xg0_sol = xg0;
```

output

```
% original initial guess
xg0_OG

% solution to initial guess
xg0_sol

% covariance
Pxx = inv(H' * H)

norm(Pxx)
```

```
xg0_OG =

          2019.29620390645
           899.18143927834
          2349.46128871635
          1597.49910053977


xg0_sol =

          2009.37317814612
          899.930080622043
          2250.33006742093
          1598.79191840315
```

```
Pxx =

  Columns 1 through 3

         31.8997213721097        -0.309085320893451          64.0179446060099
        -0.309085320893451        0.0211683191056671          2.16647104432628
         64.0179446060099          2.16647104432628          721.526969961692
        -0.520479035921291        0.0173132374010022         0.021697715758036

  Column 4

        -0.520479035921291
        0.0173132374010022
         0.021697715758036
        0.0211932737104801


ans =

         727.425626880099
```

subfunctions

```
function h = h_NL(x, t, meas_data)
% Nonlinear measurement h

global la lb

    % Initialize h
    h = [];

    for i = 1:length(t)

        % y1 = y10 + v10*t = x1 + x2*t
        y1 = x(1) + x(2)*t(i);
        dy_1a = la - y1;
        dy_1b = lb - y1;

        % y2 = y20 + v20*t - 0.5 * 9.8 * t^2
        dy_2 = x(3) + x(4)*t(i) - 4.9*t(i)^2;

        h_rhoa   = sqrt( dy_1a^2 + dy_2^2 );
        h_rhob   = sqrt( dy_1b^2 + dy_2^2 );
        h_thetaa = atan2( dy_2, dy_1a );
        h_thetab = atan2( dy_2, dy_1b );

        % Build nonlinear h from guess
        switch meas_data

            case 'rho'
                h = [h; h_rhoa; h_rhob];

            case 'theta_a'
                h = [h; h_thetaa];

            case 'both'
                h = [h; h_rhoa; h_thetaa; h_rhob; h_thetab];

        end

    end

end

function H = Hhist(x, thist, Hhist_j, meas_data)
% Full jacobian of h

    global la lb

    H = [];
    for j = 1:length(thist)

        switch meas_data

            case 'rho'
                H = [ H; Hhist_j(la, lb, thist(j), x(1), x(2), x(3), x(4)) ];

            case 'theta_a'
                H = [ H; Hhist_j(la, thist(j), x(1), x(2), x(3), x(4)) ];

            case 'both'
                H = [ H; Hhist_j(la, lb, thist(j), x(1), x(2), x(3), x(4)) ];

        end

    end

end
```

```
function [Jg, h, H, dx] = cost_fn(xg, thist, zhist, Ra, Hhist_j, meas_data)

    % Normalized NL at guess
    h = inv(Ra') * h_NL(xg, thist, meas_data);

    % Normalized jacobian at guess
    H = inv(Ra') * Hhist(xg, thist, Hhist_j, meas_data);

    % Normalized measurement
    z = inv(Ra') * zhist;

    % Gauss-Newton dx
    dx = inv((H' * H)) * H' * (z - h);

    % Cost function
    Jg = norm(z - h);

end

function xg0_OG = find_xg0(rhoahist, rhobhist, thist, i, f)

clear x

% "initial" measurements
% i = 3;
p_ai = rhoahist(i);
p_bi = rhobhist(i);

global la lb

y_1i = 1/( 2*lb - 2*la ) * ( p_ai^2 - la^2 - p_bi^2 + lb^2);
y_2i = sqrt( p_ai^2 - ( la - y_1i )^2 );

% last measurements
% f = 26;
p_af = rhoahist(f);
p_bf = rhobhist(f);

y_1f = 1/( 2*lb - 2*la ) * ( p_af^2 - la^2 - p_bf^2 + lb^2 );
y_2f = sqrt( p_af^2 - ( la - y_1f )^2 );

% guessing x1 (y10) and x2 (v10)
% y_1s = (1)*y10 + (ts)*v10
% y_1f = (1)*y10 + (tf)*v10
ti = thist(i); tf = thist(f);
x = pinv( [ 1 ti; 1 tf ] ) * [y_1i; y_1f];
y_10 = x(1);
v_10 = x(2);

% guessing x3 (y20) and x4 (v20)
% y_2s = (1)*y20 + (ts)*v20 - 4.9ts^2
% y_2f = (1)*y20 + (tf)*v20 - 4.9tf^2
x = pinv( [ 1 ti; 1 tf ] ) * ( [ y_2i; y_2f ] + 4.9 * [ ti^2; tf^2 ] );
y_20 = x(1);
v_20 = x(2);

% SANITY CHECK linear algebra
t = [ 0; -0.5 * 9.8 * ti^2; 0; -0.5 * 9.8 * tf^2 ];
y = [y_1i; y_2i; y_1f; y_2f];
A = [1, ti, 0, 0;
     0, 0, 1, ti;
     1, tf, 0, 0;
     0, 0, 1, tf ];
x = pinv( A ) * (y - t);

% First guess
xg0_OG = [y_10; v_10; y_20; v_20];

end
```

Published with MATLAB® R2020a