```
clear variables;
close all;
clc;
```

Problem

Load data and initiate constants

```
data = load("radarmeasdata_missile_new.mat");
t = data.thist;
rho_a = data.rhoahist;
rho_b = data.rhobhist;
theta_a = data.thetaahist;
theta_b = data.thetabhist;
% Range only
z = zeros([2*length(t), 1]);
z(1:2:end) = rho_a;
z(2:2:end) = rho_b;
% Combined
z_comb = zeros([4*length(t), 1]);
z_comb(1:4:end) = rho_a;
z_comb(2:4:end) = rho_b;
z_comb(3:4:end) = theta_a;
z_comb(4:4:end) = theta_b;
% Bearing Only
z_bear = theta_a;
Ri = [10^2, 0; 0 30^2];
Ri_comb = diag([10^2, 30^2, 0.01^2, 0.03^2]);
Ri_bear = [0.01^2];
R = kron(eye(length(t)), Ri);
R_comb = kron(eye(length(t)), Ri_comb);
R_bear = kron(eye(length(t)), Ri_bear);

x_g = [8.1547e4; 1.2842e3; 1.2856e3; 1.2842e3]; % Initial guess

% Set up function handles
Hprime = @(x) Hprime_func(x, t);
Hprime_comb = @(x) Hprime_comb_func(x, t);
Hprime_bear = @(x) Hprime_bear_func(x, t);
hprime = @(x) hprime_func(x, t);
hprime_comb = @(x) hprime_comb_func(x, t);
hprime_bear = @(x) hprime_bear_func(x, t);
J = @(z, x, h) J_func(z, x, h);

% Set up GN params
threshold = 1e-9;
adjustStep = true;

% Run GN  - just range
[x_sol, P_xx_sol] = gauss_newton(J, Hprime, hprime, R, z, x_g, threshold,
adjustStep);
```

```
x_sol
P_xx_sol

% Run GN - combined SOMETHING IS WRONG HERE
[x_sol_comb, P_xx_sol_comb] = gauss_newton(J, Hprime_comb, hprime_comb,
R_comb, z_comb, x_g, threshold, adjustStep);


x_sol_comb
P_xx_sol_comb

% This being bigger in some directions doesn't make sense
u_improvement_perc = (abs(P_xx_sol) - abs(P_xx_sol_comb))./abs(P_xx_sol) *
100


% Run GN - just bearing from a
x_g_bear = x_sol_comb;
[x_sol_bear, P_xx_sol_bear] = gauss_newton(J, Hprime_bear, hprime_bear,
R_bear, z_bear, x_g_bear, threshold, adjustStep);


x_sol_bear
P_xx_sol_bear
```

Functions

```
function [H] = Hprime_func(x, t)
    % Constants
    la = 4.1e5; % m
    lb = 4.4e5; % m
    g = 9.81; % m/s^2

    H = zeros([2*length(t), 4]); % Initialize the H matrix

    % Break out states
    x0 = x(1);
    y0 = x(2);
    vx0 = x(3);
    vy0 = x(4);

    for j=1:length(t)
        tj = t(j);
        % Compute elements for each time step
        h1x1 = (vx0*tj+x0-la) / sqrt((vx0*tj+x0-la)^2 +
(y0+vy0*tj-0.5*g*tj^2)^2);
        h1x2 = (y0+vy0*tj-0.5*g*tj^2) / sqrt((vx0*tj+x0-la)^2 +
(y0+vy0*tj-0.5*g*tj^2)^2);
        h1x3 = tj*h1x1;
        h1x4 = tj*h1x2;

        h2x1 = (x0+vx0*tj-lb) / sqrt((x0+vx0*tj-lb)^2 +
(y0+vy0*tj-0.5*g*tj^2)^2);
```

```
        h2x2 = (y0+vy0*tj-0.5*g*tj^2) / sqrt((x0+vx0*tj-lb)^2 +
(y0+vy0*tj-0.5*g*tj^2)^2);
        h2x3 = tj*h2x1;
        h2x4 = tj*h2x2;

        % Insert computed values into H
        H((j-1)*2 + 1, :) = [h1x1, h1x2, h1x3, h1x4];
        H((j-1)*2 + 2, :) = [h2x1, h2x2, h2x3, h2x4];
    end
end

function [H] = Hprime_comb_func(x, t)
    % Constants
    la = 4.1e5; % m
    lb = 4.4e5; % m
    g = 9.81; % m/s^2

    H = zeros([4*length(t), 4]); % Initialize the H matrix

    % Break out states
    x0 = x(1);
    y0 = x(2);
    vx0 = x(3);
    vy0 = x(4);

    for j=1:length(t)
        tj = t(j);
        % Compute elements for each time step
        h1x1 = (vx0*tj+x0-la) / sqrt((vx0*tj+x0-la)^2 +
(y0+vy0*tj-0.5*g*tj^2)^2);
        h1x2 = (y0+vy0*tj-0.5*g*tj^2) / sqrt((vx0*tj+x0-la)^2 +
(y0+vy0*tj-0.5*g*tj^2)^2);
        h1x3 = tj*h1x1;
        h1x4 = tj*h1x2;

        h2x1 = (x0+vx0*tj-lb) / sqrt((x0+vx0*tj-lb)^2 +
(y0+vy0*tj-0.5*g*tj^2)^2);
        h2x2 = (y0+vy0*tj-0.5*g*tj^2) / sqrt((x0+vx0*tj-lb)^2 +
(y0+vy0*tj-0.5*g*tj^2)^2);
        h2x3 = tj*h2x1;
        h2x4 = tj*h2x2;

        h3x1 = (y0+vy0*tj-0.5*g*tj^2) / ((x0+vx0*tj-la)^2 +
(y0+vy0*tj-0.5*g*tj^2)^2);
        h3x2 = -1 / ((((y0+vy0*tj-0.5*g*tj^2)^2) / (x0+vx0*tj-la)) +
(x0+vx0*tj-la));
        h3x3 = tj*(y0+vy0*tj-0.5*g*tj^2) / ((x0+vx0*tj-la)^2 +
(y0+vy0*tj-0.5*g*tj^2)^2);
        h3x4 = -tj / (((y0+vy0*tj-0.5*g*tj^2)^2) / (x0+vx0*tj-la) +
(x0+vx0*tj-la));

        h4x1 = (y0+vy0*tj-0.5*g*tj^2) / ((y0+vy0*tj-0.5*g*tj^2)^2 +
(x0+vx0*tj-lb)^2);
        h4x2 = 1 / ((lb - vx0*tj - x0) + (((y0+vy0*tj-0.5*g*tj^2)^2) / (lb -
```

```
vx0*tj - x0)));
        h4x3 = tj*(y0+vy0*tj-0.5*g*tj^2) / ((y0+vy0*tj-0.5*g*tj^2)^2 +
(x0+vx0*tj-lb)^2);
        h4x4 = tj / ((lb - vx0*tj - x0) + (((y0+vy0*tj-0.5*g*tj^2)^2) / (lb
- vx0*tj - x0)));

        % Insert computed values into H
        H((j-1)*4 + 1, :) = [h1x1, h1x2, h1x3, h1x4];
        H((j-1)*4 + 2, :) = [h2x1, h2x2, h2x3, h2x4];
        H((j-1)*4 + 3, :) = [h3x1, h3x2, h3x3, h3x4];
        H((j-1)*4 + 4, :) = [h4x1, h4x2, h4x3, h4x4];
    end
end

function [H] = Hprime_bear_func(x, t)
    % Constants
    la = 4.1e5; % m
    g = 9.81; % m/s^2

    H = zeros([length(t), 4]); % Initialize the H matrix

    % Break out states
    x0 = x(1);
    y0 = x(2);
    vx0 = x(3);
    vy0 = x(4);

    for j=1:length(t)
        tj = t(j);
        % Compute elements for each time step
        h1x1 = (y0+vy0*tj-0.5*g*tj^2) / ((x0+vx0*tj-la)^2 +
(y0+vy0*tj-0.5*g*tj^2)^2);
        h1x2 = -1 / (((y0+vy0*tj-0.5*g*tj^2)^2) / (x0+vx0*tj-la) +
(x0+vx0*tj-la));
        h1x3 = tj*(y0+vy0*tj-0.5*g*tj^2) / ((x0+vx0*tj-la)^2 +
(y0+vy0*tj-0.5*g*tj^2)^2);
        h1x4 = -tj / (((y0+vy0*tj-0.5*g*tj^2)^2) / (x0+vx0*tj-la) +
(x0+vx0*tj-la));

        % Insert computed values into H
        H(j, :) = [h1x1, h1x2, h1x3, h1x4];
    end
end

function [h] = hprime_func(x, t)
    % Constants
    la = 4.1e5; % m
    lb = 4.4e5; % m
    g = 9.81; % m/s^2

    h = zeros([2*length(t), 1]); % Initialize the h vector

    % Break out states
    x0 = x(1);
```

```
    y0 = x(2);
    vx0 = x(3);
    vy0 = x(4);

    % Compute h for each
    for j=1:length(t)
        tj = t(j);
        h1 = sqrt((vx0*tj+x0-la)^2 + (y0+vy0*tj-0.5*g*tj^2)^2);
        h2 = sqrt((x0+vx0*tj-lb)^2 + (y0+vy0*tj-0.5*g*tj^2)^2);

        h((j-1)*2 + 1) = h1;
        h((j-1)*2 + 2) = h2;


    end

end

function [h] = hprime_comb_func(x, t)
    % Constants
    la = 4.1e5; % m
    lb = 4.4e5; % m
    g = 9.81; % m/s^2

    h = zeros([4*length(t), 1]); % Initialize the h vector

    % Break out states
    x0 = x(1);
    y0 = x(2);
    vx0 = x(3);
    vy0 = x(4);

    % Compute h for each
    for j=1:length(t)
        tj = t(j);
        h1 = sqrt((vx0*tj+x0-la)^2 + (y0+vy0*tj-0.5*g*tj^2)^2);
        h2 = sqrt((x0+vx0*tj-lb)^2 + (y0+vy0*tj-0.5*g*tj^2)^2);
        h3 = pi - atan2((y0+vy0*tj-0.5*g*tj^2), (x0+vx0*tj-la));
        h4 = atan2((y0+vy0*tj-0.5*g*tj^2), (lb - (x0+vx0*tj)));

        h((j-1)*4 + 1) = h1;
        h((j-1)*4 + 2) = h2;
        h((j-1)*4 + 3) = h3;
        h((j-1)*4 + 4) = h4;

    end
end

function [h] = hprime_bear_func(x, t)
    % Constants
    la = 4.1e5; % m
    g = 9.81; % m/s^2

    h = zeros([length(t), 1]); % Initialize the h vector
```

```
    % Break out states
    x0 = x(1);
    y0 = x(2);
    vx0 = x(3);
    vy0 = x(4);

    % Compute h for each
    for j=1:length(t)
        tj = t(j);
        h1 = pi - atan((y0+vy0*tj-0.5*g*tj^2) / (x0+vx0*tj-la));

        h(j) = h1;
    end
end

function [x_star, P_xx_star] = gauss_newton(J, Hprime, hprime, Rprime,
zprime, x_g, threshold, adjustStep)
    dx = 1;

    % Perform normalization
    Ra = chol(Rprime);
    Rait = inv(Ra.');

    % Define normalized quantities
    H = @(x) Rait*Hprime(x);
    h = @(x) Rait*hprime(x);
    z = Rait*zprime;




    iter = 0;
    while norm(dx) > threshold && iter <= 100

        Hx = H(x_g);
        hx = h(x_g);
        dx = (Hx.'*Hx)\Hx.'*(z-hx);

        % Calculate step size
        alpha = 1;
        if adjustStep == true
            current_cost = J(z, x_g, h);
            new_cost = J(z, x_g + alpha * dx, h);
            while current_cost < new_cost
                alpha = alpha / 2;
                new_cost = J(z, x_g + alpha * dx, h);
            end
        end
        x_g = x_g + alpha * dx; % Update the estimate
        % cost = J(z, x_g, h); % Calculate the cost at each step

        % Increment iterations
        iter = iter + 1;
```

```
    end
    % Pass out final results
    x_star = x_g;
    P_xx_star = inv(Hx.'*Hx);
    % cost  = J(z, x_g, h);

end

function [cost] = J_func(z, x, h)
    cost = norm(z - h(x))^2;
end
```

```
x_sol =

   1.0e+03 *

    1.2963
    1.8026
    0.8993
    1.9004


P_xx_sol =

   1.0e+03 *

    0.0303    0.0157   -0.0005   -0.0005
    0.0157    4.0190    0.0203    0.0059
   -0.0005    0.0203    0.0001    0.0000
   -0.0005    0.0059    0.0000    0.0000


x_sol_comb =

   1.0e+03 *

    1.2964
    1.7982
    0.8993
    1.9004


P_xx_sol_comb =

   1.0e+03 *

    0.0302    0.0172   -0.0005   -0.0005
    0.0172    3.9043    0.0196    0.0056
   -0.0005    0.0196    0.0001    0.0000
   -0.0005    0.0056    0.0000    0.0000


u_improvement_perc =
```

```
    0.0850   -9.8513    1.8167    0.6623
   -9.8513    2.8526    3.1392    3.8554
    1.8167    3.1392    3.0304    2.8638
    0.6623    3.8554    2.8638    1.9055


x_sol_bear =

   1.0e+06 *

    0.4100
   -3.6286
   -0.0171
    0.0029


P_xx_sol_bear =

   1.0e+13 *

    0.0001   -0.0050   -0.0000    0.0000
   -0.0050    1.2283    0.0061   -0.0004
   -0.0000    0.0061    0.0000   -0.0000
    0.0000   -0.0004   -0.0000    0.0000
```

Published with MATLAB® R2025a