Adv. Engineering Math Homework 7

Ian Sinclair
ENGR 3621-1

March 30, 2022
Dr. Margareta Stefanovic

Problem 1:

The MATLAB script below generates the real data linearly modeled as Ax, and the noise corrupted measured data b modeled as $b = Ax + n$

```
x = 3;  % true slope
A = [0: .25: 5]';
b = A*x + 1*randn(size(A));  %add noise
plot(A, A*x, 'k') % true relationship
hold on
plot(A,b, 'rx')
```

Compute and the least squares approximation of SVD Superimpose the plot of the result onto the previous plot. Attach the matlab script you wrote and the plot.

Let $Ax = b^*$ be the true relationship be the signal, and $Ax + n = b$ be the noisy signal data.
Desired is a least square approximation of x, being $\tilde{x}$ that minimizes the square different between b and b^*,

$$Ax \cong b = \min_{\tilde{x}} ||b - b^*||_2^2 = \min_{\tilde{x}} ||b - A\tilde{x}||_2^2$$

Has a well known solution,

$$\hat{b} = A\tilde{x} = P_A b$$

Or is the projection of b onto the span of A, which implies that,

$$\tilde{x} = (A^T A)^{-1} A^T b$$

As a result, the error, $e = b - \hat{b}$ between the noisy data and the approximation is,

$$e = b - \hat{b} = b - A(A^T A)^{-1} A^T b = b - P_A b = (I - P_A)b = P_{A\perp} b$$

Or is the projection of b onto the space normal to the span of A.
Next, note that A can be written in terms of its economy singular value decomposition,

$$A = U\Sigma V^T = \tilde{U}\Sigma V^T$$

Where,

$$U = \left[\tilde{U} \;\middle|\; U_\perp\right]$$

Now substituting for A in the projection, $\tilde{x}$,

$$\tilde{x} = (A^T A)^{-1} A^T b = (V\Sigma\tilde{U}^T\tilde{U}\Sigma V^T)^{-1} V\Sigma^T\tilde{U}^T b = (V\Sigma V^T)^{-1} V\Sigma^T\tilde{U}^T b = V\Sigma^{-1}\tilde{U}^T b$$

Interestingly, now $\hat{b}$ can be found by,

$$\hat{b} = P_A b = A(A^T A)^{-1} A^T b = \tilde{U}\Sigma V^T (V\Sigma^{-1}\tilde{U}^T)b = \tilde{U}\tilde{U}^T b$$

Which implies,

$$span\{A\} = span\{\tilde{U}\}$$

Now examining the error,

$$P_{A\perp} b = b - \hat{b} = (I - P_A b) = (I - \tilde{U}\tilde{U}^T)b \longrightarrow P_{A\perp} = (I - \tilde{U}\tilde{U}^T)$$

And now note by the construction of U,

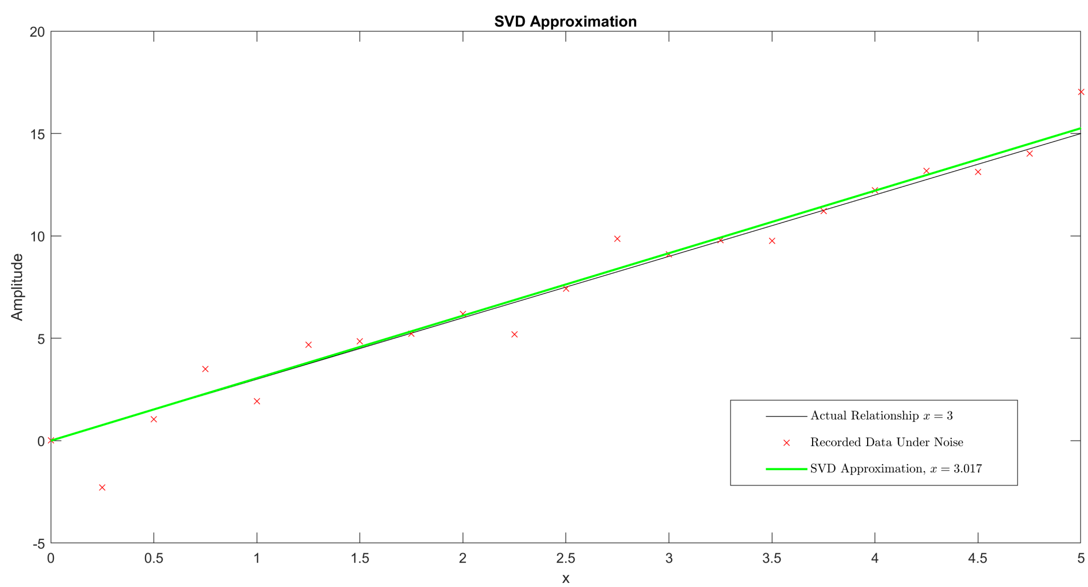
$$U = [\tilde{U} \mid U_\perp]$$

It follows that,

$$I = UU^T = [\tilde{U} \mid U_\perp] \frac{[\tilde{U}^T]}{[U_\perp^T]} = \tilde{U}\tilde{U}^T + U_\perp U_\perp^T \longrightarrow P_{A\perp} = (I - \tilde{U}\tilde{U}^T)) = U_\perp U_\perp^T$$

Ultimately, this results in an approximation generated from $\tilde{x}$ by,

```
[U, S, V] = svd(A,0); % Returns the economy SVD
Xtilde = V*(inv(S))*U'*b;
plot(A, A*Xtilde, 'r');
```



Problem 2:

A ray of light having amplitude α travels through space in the unit direction $d \in \mathbb{R}^3$ and falls on 4 photosensors located on a solar panel. Unit vectors $q \in \mathbb{R}^3$, $i = 1, 2, .., 4$ are outward normal vectors to the surface of each photosensor, representing the direction in which the sensor is pointed. Assume that $\{q_1, q_2, q_3, q_4\}$ span $\mathbb{R}^3$ (that is, not all sensors point in the same plane). Each photosensor generates a scalar output signal,

$$p_i = \alpha\sigma\cos(\theta_i + v_i)$$

where θ_i is the angle between the ray direction d and the sensor direction q_i and $\sigma \in \mathbb{R}$ is the photosensor sensitivity (same for all sensors). The scalar numbers v_i are small measurement errors (noise). We are given the following data: the photosensor direction vectors $q \in \mathbb{R}^3$, $i = 1, 2, .., 4$ the photosensor sensitivity σ and the noisy photosensor outputs, $p_i \in \mathbb{R}$, $i = 1, 2, .., 4$ Your task is to estimate the light ray direction $d \in \mathbb{R}^3$ (also a unit vector), and α the ray amplitude, knowing that v_i is small. The unit vectors q_i are defined in terms of the azimuth angles θ_i and elevation angles ϕ_i as follows:

$$q_i = \begin{bmatrix} \cos(\phi_i)\cos(\theta_i) \\ \cos(\phi_i)\sin(\phi_i) \\ \sin(\phi_i) \end{bmatrix}, i = 1, 2, .., 4$$

Similarly,

$$d = \begin{bmatrix} \cos(\phi_d)\cos(\theta_d) \\ \cos(\phi_d)\sin(\phi_d) \\ \sin(\phi_d) \end{bmatrix}$$

Implement your method in Matlab using the data given in $hw7_p2_data.m$. This file defines P the vector of photosensor outputs, the vector Θ which gives the azimuth angles of the photosensors directions, and the vector Φ which gives the elevation angles of the photosensor directions for the four sensors. Both angle vectors are given in degrees. Give your estimate of the ray amplitude α and ray direction d (in azimuth and elevation, in degrees). Attach the matlab code you wrote.

Let w be a ray of light with direction d and magnitude α; such that, $w = \alpha d$.
Then take 4 photosensors located on solar panels such that q_i is the outward normal vectors to the surface of each photosensor. Next, assume that $\{q_i\}$ is non-planar any so spans $\mathbf{R}^3$. Take $\sigma \in \mathbb{R}$ be the photosensor sensitivity consistent for all sensors. Lastly, let v_i be noise in the signal.
And so, the scalar output signal is,

$$p_i = \alpha\sigma\cos(\theta_i) + v_i$$

And for small v_i

$$p_i \approx \alpha\sigma\cos(\theta_i) = \sigma q_i^T(\alpha d) = \sigma q_i^T(w)$$

As a result, desired in a vector w that corresponds to the projection of p onto the basis generated by $[q]$.
And so define q_i in terms of ϕ and θ,

$$q_i = \begin{bmatrix} \cos(\phi_i)\cos(\theta_i) \\ \cos(\phi_i)\sin(\phi_i) \\ \sin(\phi_i) \end{bmatrix}, i = 1, 2, .., 4$$

Where ϕ_i and θ_i are known.
Then let q be the column wise adjoinment of each q_i, $q = [q_1, .., q_4]$.
Then by a symmetric argument to problem 1, the optimal $\hat{w}$ that finds the least square projection of p onto $span(q)$ is,

$$\hat{w} = V\Sigma^{-1}\tilde{U}^T p$$

Where $q = \tilde{U}\Sigma V^T$ is the economy SVD of q.
Next, because $\{q_1, ..., q_4\}$ spans $\mathbb{R}^3$ matrix q is consistent and so,

$$\hat{w} = \sigma w$$

And so,

$$\alpha = \frac{||\hat{w}||}{\sigma}$$

And d is the normalized direction vector of w expressed in azimuth and elevation angles.

```
%% Problem 2 {Ray Detection}
clear
% Data for HW 7, P2 (estimate light-ray amplitude and direction)
m = 4;
sigma = 0.5;
% theta is azimuth angle
theta =[ 3
 10
 80
 150
];
%  phi is elevation angle
phi =[ 88
 34
 30
 20
];
p = [  1.58
 1.50
 2.47
 1.10
];


q = [];
for i = 1:length(theta)
    n = [cosd(theta(i))*cosd(phi(i)); sind(theta(i))*cosd(phi(i)); sind(phi(i))];
    q = [q n];
end


[U, S, V] = svd(q', 0); %Returns the economy SVD
w = V'*inv(S)*U'*p;

w = w/sigma;
[TH, PHI, alpha] = cart2sph(w(1),w(2),w(3));
```

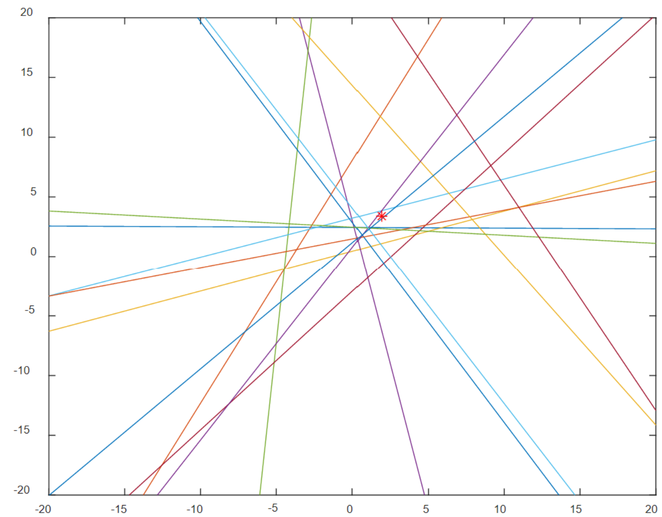
Solution

Which results in,

$$\alpha = 5.0107, \qquad d = \begin{bmatrix} \cos(\phi_d)\cos(\theta_d) \\ \cos(\phi_d)\sin(\phi_d) \\ \sin(\phi_d) \end{bmatrix} = \begin{bmatrix} \cos(-18.6^\circ)\cos(98.66^\circ) \\ \cos(-18.6^\circ)\sin(98.66^\circ) \\ \sin(-18.6^\circ) \end{bmatrix} = \begin{bmatrix} -0.1427 \\ 0.9369 \\ -0.32 \end{bmatrix}$$

Problem 3:

In this problem we find the point in $\mathbb{R}^n$ ((the position vector z^* associated with the point) that is closest (in Euclidean sense) to the given set of m lines in $\mathbb{R}^n$. The end result should look similar to the sample figure below, which shows 15



lines in $\mathbb{R}^2$. and the point (marked by a red asterisk) that is closest (among all points in $\mathbb{R}^2$ to the given lines. Each line is defined by a vector equation:

$$\mathscr{L}_i : r = P_i + tV_i$$

Where $t \in \mathbb{R}$ is a scalar parameter and $r, P_i, V_i \in \mathbb{R}^n$. Vectors V_i are unit length.
For example, for $\mathbb{R}^2$

$$\mathscr{L} : \begin{bmatrix} r_x \\ r_y \end{bmatrix} = \begin{bmatrix} P_x \\ P_y \end{bmatrix} + t \begin{bmatrix} V_x \\ V_y \end{bmatrix}$$

Next, the distance from a point, z, to a line is calculated as follows,

$$dist(z, \mathscr{L}) := \min_{u \in \mathscr{L}} ||z - u||$$

That is, the closest Euclidean distance between z and some point in $\mathscr{L}$. Additionally, note that,

$$||z - u|| = \sqrt{(z-u)^T(z-u)}$$

Find the point $z^* \in \mathbb{R}^n$ that minimizes the sum of the squares of the distances to all m given lines. In this way, we find the point that is closest to all lines simultaneously.

$$z^* = \arg\min_{z \in \mathbb{R}^n} \sum_{i=1}^{n} (dist(z, \mathscr{L}_i))^2$$

If you need to assume that some condition holds (such as some matrix being full rank), state this explicitly.

Consider the set of lines, $\mathscr{L}_i$ defined by,

$$r_i = P_i + tV_i$$

For matrices $r, P, V \in \mathbb{R}^n$, where n is the diminsionality of the lines.
Then, let z be a point, $z \in \mathbb{R}^n$,
then the minimum distance from any line $\mathscr{L}_i$ to z can be found by,

$$||z - u|| = \sqrt{(z-u)^T(z-u)} = \sqrt{(z - P - tV)^T(z - P - tV)}$$

And so take,

$$J_i = \sqrt{(z - P - tV)^T(z - P - tV)}$$

to be the cost function describing the distance from z to $\mathscr{L}_i$, which is optimized by,

$$\nabla_{t^*} J_i = 0$$

For optimal parameter t^*.
Additionally, note that the optimum for J_i is incident with the optimum for $J_i = (z - P - tV)^T(z - P - tV)$ and so redefine the cost function,

$$J_i = (z - P - tV)^T(z - P - tV)$$

Then,

$$0 = \nabla_t J_i = 2(z - P - tV)^T(-V)$$

Which implies,

$$(z - P)^T v = tV^T V$$

And so the t^* that gives an optimum for J_i is,

$$t^* = \frac{(z - P)^T v}{V^T V} = \frac{(z - P)^T v}{||v||}$$

And so the point closest to z is,

$$u^* = P + \frac{(z - P)^T V}{||V||} V$$

Next, assume that V is normalized such that, $||V|| = 1$; then,

$$u^* = P + VVT^(z - P)$$

Therefore, the minimum distance between z and $\mathscr{L}_i$ is,

$$dist(z, \mathscr{L}_i) = \min_{u \in \mathscr{L}_i} ||z - u|| = ||(I - VV^T)(z - P)||$$

Now, desired is a $z^* \in \mathbb{R}^n$ that minimizes the distance to all of m lines $\mathscr{L}$, and so define the cost function,

$$J := \sum_{i=1}^{m} (dist(z, \mathscr{L}_i))^2$$

in terms of vectors P_i and V_i.
Where,

$$z^* = \arg\min_z \{\sum_{i=1}^{m} (dist(z, \mathscr{L}_i))^2\}$$

Again note the distance function to each line,

$$J := \sum_{i=1}^{m} dist(z, \mathscr{L}_i) = \sum_{i=1}^{m} ||(I - VV^T)(z - P)|| = \sum_{i=1}^{m} \sqrt{(z - P)^T(I - VV^T)^2(z - P)}$$

Has a minimum incident with,

$$J := \sum_{i=1}^{m} (z - P)^T(I - VV^T)^2(z - P)$$

Now, because V is known, take,

$$Q_i = (I - V_i V_i^T)^2$$

And assume Q_i has full rank. Also, note that Q is symmetric because $V_i V_i^T$ is symmetric, and so, $(I - V_i V_i^T)$ is symmetric, lastly this means that, $Q = (I - V_i V_i^T)^2$ is symmetric.
Now note the cost function,

$$J = \sum_{i=1}^{m} (z-P)^T Q (z-P) = (z-P_1)^T Q_1 (z-P_1) + \cdots + (z-P_m)^T Q_m (z-P_m)$$

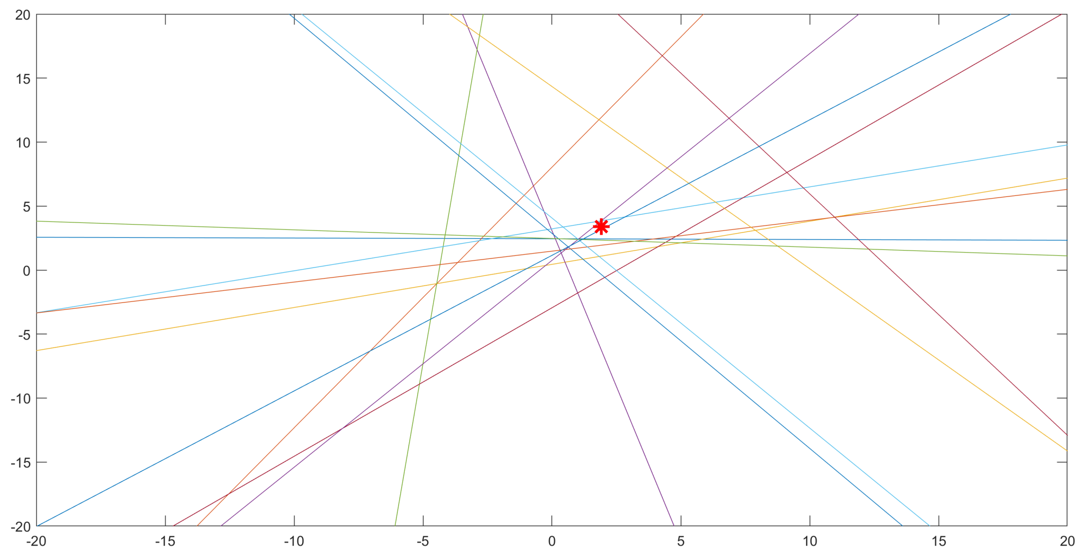
And now optimizing,

$$\begin{aligned} 0 = \nabla_z J &= Q_1(z-P_1) + \cdots + Q_m(z-P_m) = Q_1 z - Q_1 P_1 + \cdots + Q_m z - Q_m P_m \\ &= \underbrace{(Q_1 + \cdots + Q_m)}_{\mathcal{Q}} z - \underbrace{(Q_1 P_1 + \cdots + Q_m P_m)}_{\mathcal{W}} \end{aligned}$$

And so,

$$0 = \mathcal{Q} z - \mathcal{W} \longrightarrow z = \mathcal{Q}^{-1} \mathcal{W}$$

Assuming $\mathcal{Q}$ is invertible. Otherwise, select z from the $kern(\mathcal{Q}z - \mathcal{W})$.



Which is implemented by,

```
%% Problem 3 {Closest Point}
% Data for HW 7, P3 (point closest to the given lines)
clear
n = 2;
m = 15;
P = [ -0.7930, -6.1337, 6.5463, 1.3785, -4.0749, 1.9969, 3.0305, 1.7315, -0.2904, 4.6367, 1.3919, 0.0103, 3.1012, 10.9183, 4.6092;
        2.4518, -4.4494, 2.6530, -3.6324, 3.5452, 3.8776, 0.5708, -0.0375, 1.4084, 7.7409, 3.0078, 2.4676, -1.0208, 4.2206, 6.0331;
        ];
V = [ -1.0000, -0.4409, 0.9477, -0.2011, -0.0852, 0.9502, -0.6529, -0.5113, -0.9721,0.5746, -0.5262, 0.9977, -0.5202, -0.4683, -0.6865;
```

```
        0.0058, -0.8976, 0.3193, 0.9796, -0.9964, 0.3115, -0.7575, 0.8594, -0.2347, -0.8184,
            -0.8504, -0.0676, 0.8540, 0.8836, -0.7272;
        ];


Q = [0, 0 ; 0, 0];
for i = 1:length(V)
    Qn = (eye(2) - (V(:,i:i)*V(:,i:i)'))^2;
    Q = Q + Qn;
end

W = [0 ; 0];
for i = 1:length(P)
    PP = ((eye(2) - (V(:,i:i)*V(:,i:i)'))^2)*(P(:,i:i));
    W = W + PP;
end

zstar = inv(Q)*W;

t=-50:0.1:50;
for i=1:m
    cor=V(:,i)*t+P(:,i)*ones(1,length(t));
    plot(cor(1,:),cor(2,:));
    hold on
end
plot(zstar(1),zstar(2),'*r')
axis([-20 20 -20 20])
```