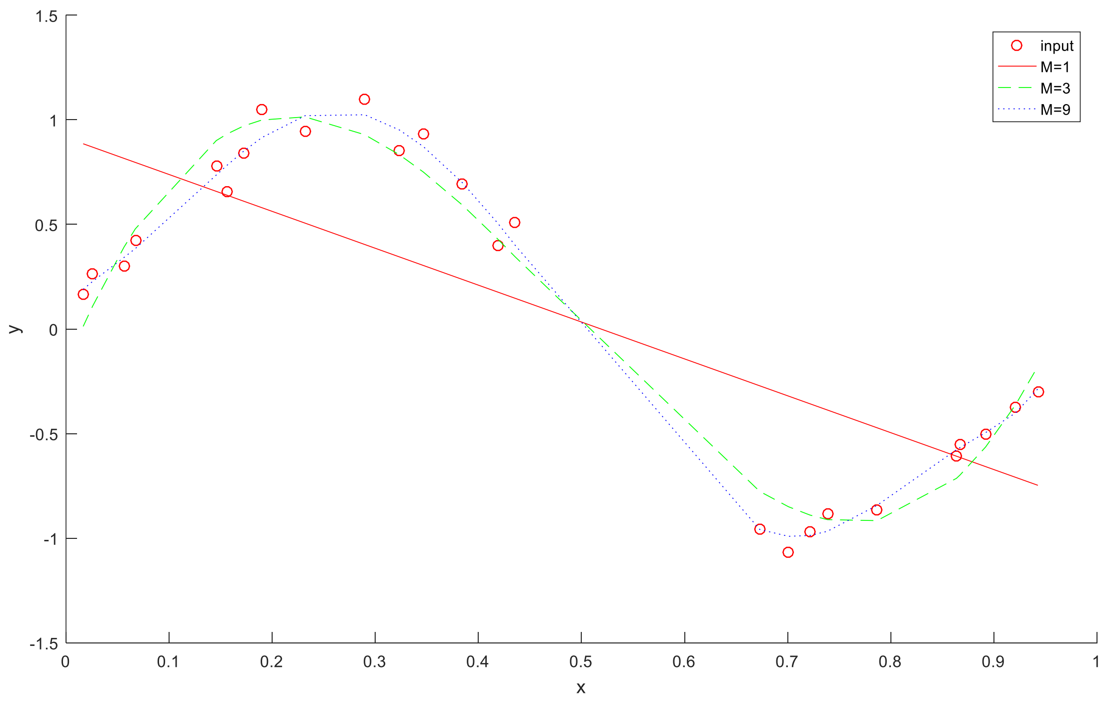
Polynomial curve fitting

b) Matlab code:

```
clear;
close all;
fclose all;

% Generate data
rand('seed', 2);
randn('seed', 2);
n = 25;
x = rand(n, 1);
x = sort(x);
noise = 0.1 * randn(n, 1);
y = sin(2 * pi * x) + noise;

%% problem b) without regularization term
% M = 1
M = 1;
m = M + 1;
X1 = zeros(n, m);
Y1 = zeros(n, 1);
for i = 1:n
    for j = 1:m
        X1(i, j) = x(i) ^ (j - 1);
    end
end
for i = 1:n
    Y1(i) = y(i);
end
% use CVX to solve
cvx_begin
    variable w1(m)
    minimize (norm(X1 * w1 - Y1, 2))
cvx_end

% M = 3
M = 3;
m = M + 1;
X2 = zeros(n, m);
Y2 = zeros(n, 1);
for i = 1:n
    for j = 1:m
        X2(i, j) = x(i) ^ (j - 1);
    end
end
for i = 1:n
    Y2(i) = y(i);
end
% use CVX to solve
cvx_begin
    variable w2(m)
    minimize (norm(X2 * w2 - Y2, 2))
cvx_end

% M = 9
M = 9;
m = M + 1;
X3 = zeros(n, m);
```

```
Y3 = zeros(n, 1);
for i = 1:n
    for j = 1:m
        X3(i, j) = x(i) ^ (j - 1);
    end
end
for i = 1:n
    Y3(i) = y(i);
end
% use CVX to solve
cvx_begin
    variable w3(m)
    minimize (norm(X3 * w3 - Y3, 2))
cvx_end

% plot
y_est1 = X1 * w1;
y_est2 = X2 * w2;
y_est3 = X3 * w3;
hold on;
axis on;
xlabel('x');ylabel('y');
plot( x(:), y(:), 'ro');
plot( x(:), y_est1(:), 'r-' );
plot( x(:), y_est2(:), 'g--' );
plot( x(:), y_est3(:), 'b:' );
legend('input', 'M=1', 'M=3', 'M=9')
hold off;
```

result:



A little overfitting seems existed in M = 9;

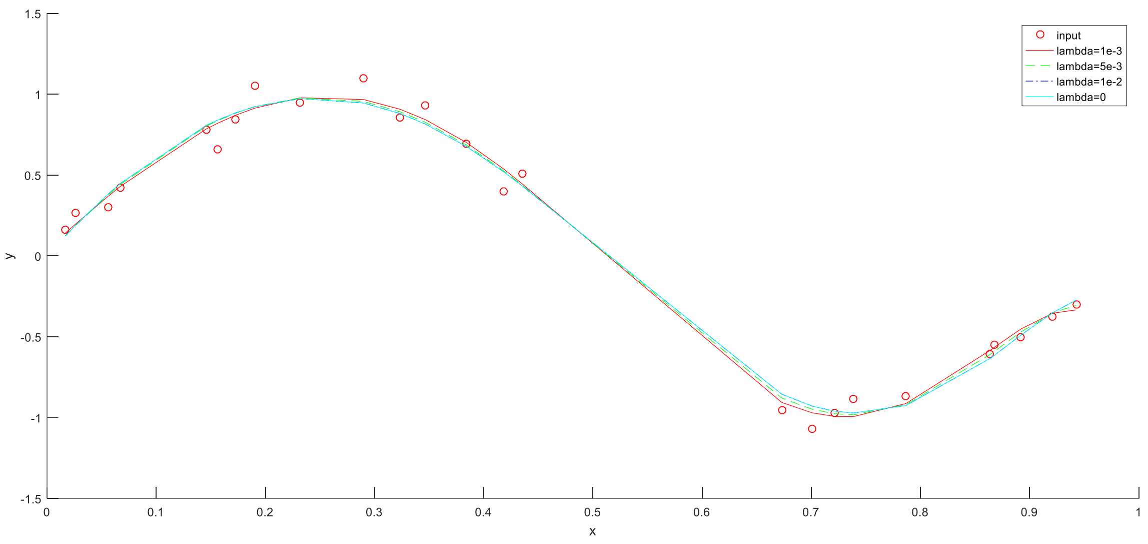
c) add regularization term

Matlab code:

```
clear;
close all;
fclose all;

% Generate data
rand('seed', 2);
randn('seed', 2);
n = 25;
x = rand(n, 1);
x = sort(x);
noise = 0.1 * randn(n, 1);
y = sin(2 * pi * x) + noise;

%% problem c) with regularization term
% set lambda
lambda1 = 1e-3;
lambda2 = 5e-3;
lambda3 = 1e-2;
lambda4 = 0;
% M = 9
M = 9;
m = M + 1;
X3 = zeros(n, m);
Y3 = zeros(n, 1);
for i = 1:n
    for j = 1:m
        X3(i, j) = x(i) ^ (j - 1);
    end
end
for i = 1:n
    Y3(i) = y(i);
end
% use CVX to solve
cvx_begin
    variable w1(m)
    minimize (norm(X3 * w1 - Y3, 2) + lambda1 * norm(w1, 2))
cvx_end

cvx_begin
    variable w2(m)
    minimize (norm(X3 * w2 - Y3, 2) + lambda2 * norm(w2, 2))
cvx_end

cvx_begin
    variable w3(m)
    minimize (norm(X3 * w3 - Y3, 2) + lambda3 * norm(w3, 2))
cvx_end

cvx_begin
    variable w4(m)
    minimize (norm(X3 * w4 - Y3, 2) + lambda4 * norm(w3, 2))
cvx_end

% plot
y_est1 = X3 * w1;
y_est2 = X3 * w2;
```

```
y_est3 = X3 * w3;
y_est4 = X3 * w4;
hold on;
axis on;
xlabel('x');ylabel('y');
plot( x(:), y(:), 'ro');
plot( x(:), y_est1(:), 'r-' );
plot( x(:), y_est2(:), 'g--' );
plot( x(:), y_est3(:), 'b-.' );
plot( x(:), y_est3(:), 'c-' );
legend('input', 'lambda=1e-3', 'lambda=5e-3', 'lambda=1e-2', 'lambda=0')
hold off;
```

result:



The green curve and blue curve are nearly overlapped with each other. They are smoother than the others.

Effect of regularization term: encourage small weight values, make the model variance smaller, but with large bias.

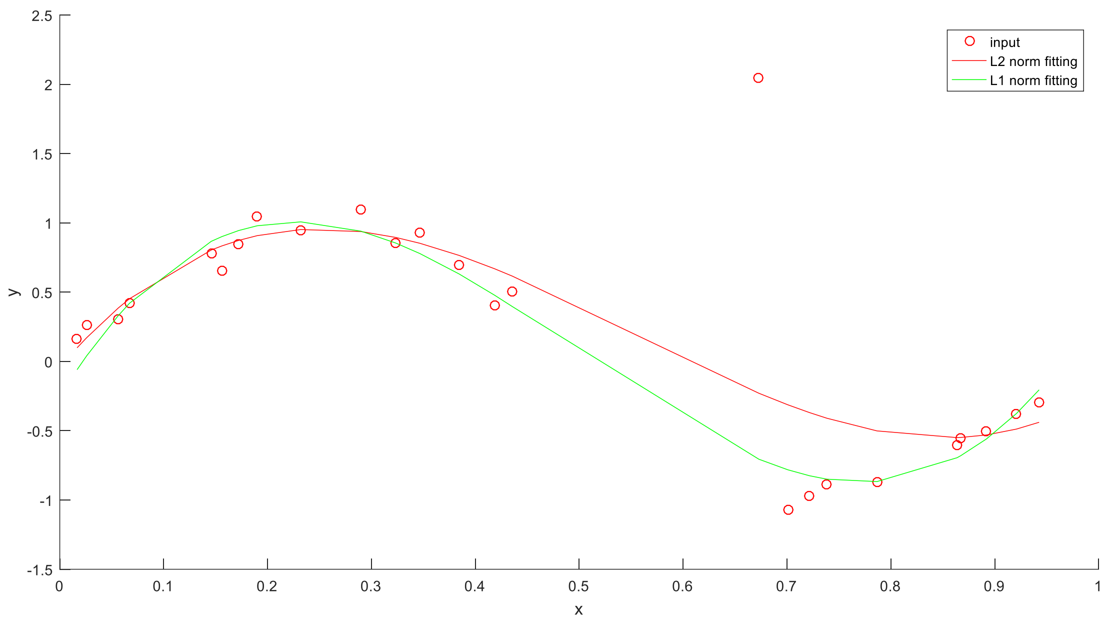
3) compare L2 norm and L1 norm

Matlab code:

```
clear;
close all;
fclose all;

% Generate data
rand('seed', 2);
randn('seed', 2);
n = 25;
x = rand(n, 1);
x = sort(x);
noise = 0.1 * randn(n, 1) + 3 * [zeros(15, 1); 1; zeros(9, 1)];
```

```
y = sin(2 * pi * x) + noise;

%% problem c) with regularization term
% set lambda
% M = 9
M = 3;
m = M + 1;
X3 = zeros(n, m);
Y3 = zeros(n, 1);
for i = 1:n
    for j = 1:m
        X3(i, j) = x(i) ^ (j - 1);
    end
end
for i = 1:n
    Y3(i) = y(i);
end
% use CVX to solve
cvx_begin
    variable w1(m)
    minimize (norm(X3 * w1 - Y3, 2))
cvx_end

cvx_begin
    variable w2(m)
    minimize (norm(X3 * w2 - Y3, 1))
cvx_end

% plot
y_est1 = X3 * w1;
y_est2 = X3 * w2;
hold on;
axis on;
xlabel('x');ylabel('y');
plot( x(:), y(:), 'ro');
plot( x(:), y_est1(:), 'r-' );
plot( x(:), y_est2(:), 'g-' );
legend('input', 'L2 norm fitting', 'L1 norm fitting')
hold off;
```

result:



We can see that there is a very large noisy point (outliers) in the input data. L2 fitting is very sensitive to outliers, while L1 fitting is better.

Minimum Volume Covering Ellipsoid

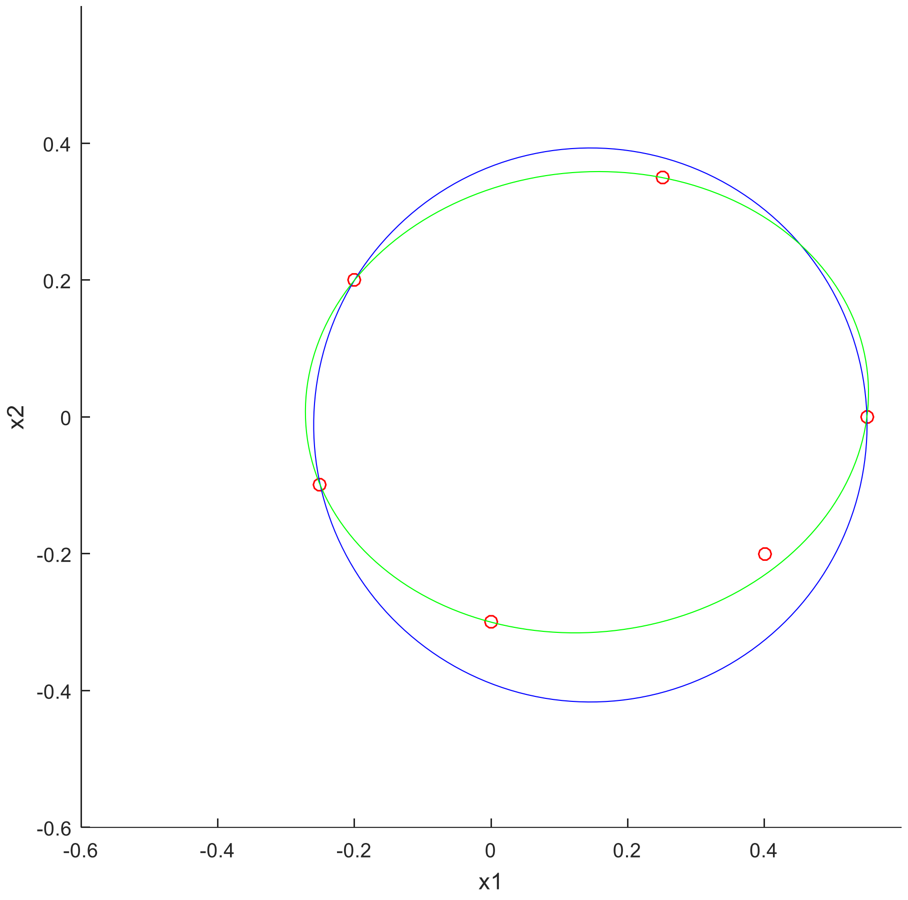
Matlab code:

```
clear;
close all;
fclose all;

% Generate data
x = [ 0.55  0.0;
      0.25  0.35
     -0.2   0.2
     -0.25 -0.1
     -0.0  -0.3
      0.4  -0.2 ]';
[n,m] = size(x);
eyemat = eye(2, 2);

% Create and solve the model for question a
cvx_begin
    variable A1
    variable b1(n)
    maximize(A1)
    subject to
        norms( A1 * eyemat * x + b1 * ones( 1, m ), 2 ) <= 1;
cvx_end

% Create and solve the model for question b
cvx_begin
    variable A2(n, n)
    variable b2(n)
    maximize(det_rootn(A2))
    subject to
        norms( A2 * x + b2 * ones( 1, m ), 2 ) <= 1;
cvx_end

% Plot the results
clf;
figure(1);

noangles = 200;
angles   = linspace( 0, 2 * pi, noangles );
ellipse1  = A1 * eyemat \ [ cos(angles) - b1(1) ; sin(angles) - b1(2) ];
ellipse2  = A2 \ [ cos(angles) - b2(1) ; sin(angles) - b2(2) ];
hold on;
axis on;
axis([-0.6 0.6 -0.6 0.6]); xlabel('x1');ylabel('x2');pbaspect([1, 1, 1])
plot( x(1,:), x(2,:), 'ro', ellipse1(1,:), ellipse1(2,:), 'b-' );
plot(ellipse2(1,:), ellipse2(2,:), 'g-' );
hold off;
```

result:

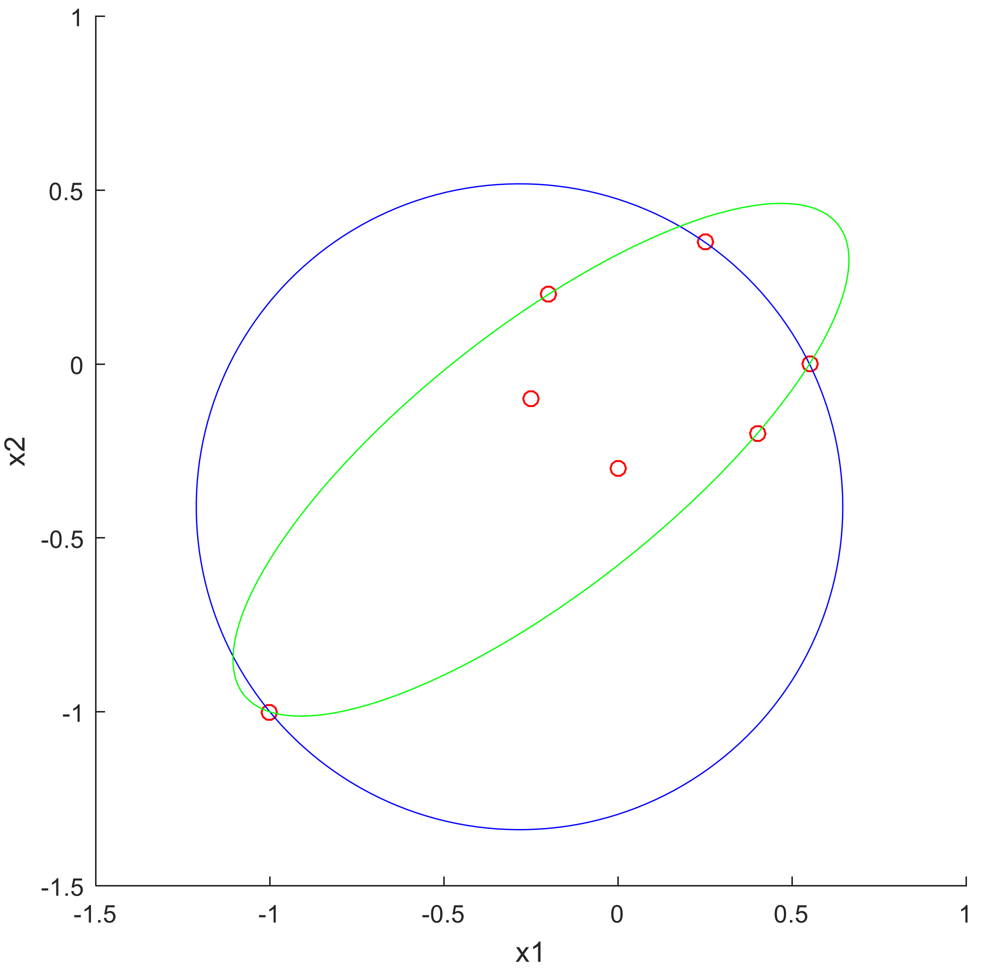
the blue curve (a circle) is the result of a).

the green curve (a ellipsoid) is the result of b).



c) add an extra point (-1, -1), repeat a) and b).

the result becomes:



the parameter of a) is:

Ball center (-0.2829, -0.4103), radius is 0.9284

The parameter of b) is:

Ellipsoid center (-0.2213, -0.2752), shape matrix $\begin{pmatrix} 0.8135 & 0.3473 \\ 0.3473 & 0.6501 \end{pmatrix}$

The outlier makes the result far from the original one. Because the new outlier point is far from the original points, both the ball and the ellipsoid must go through the outlier. This problem is not robust to outliers.