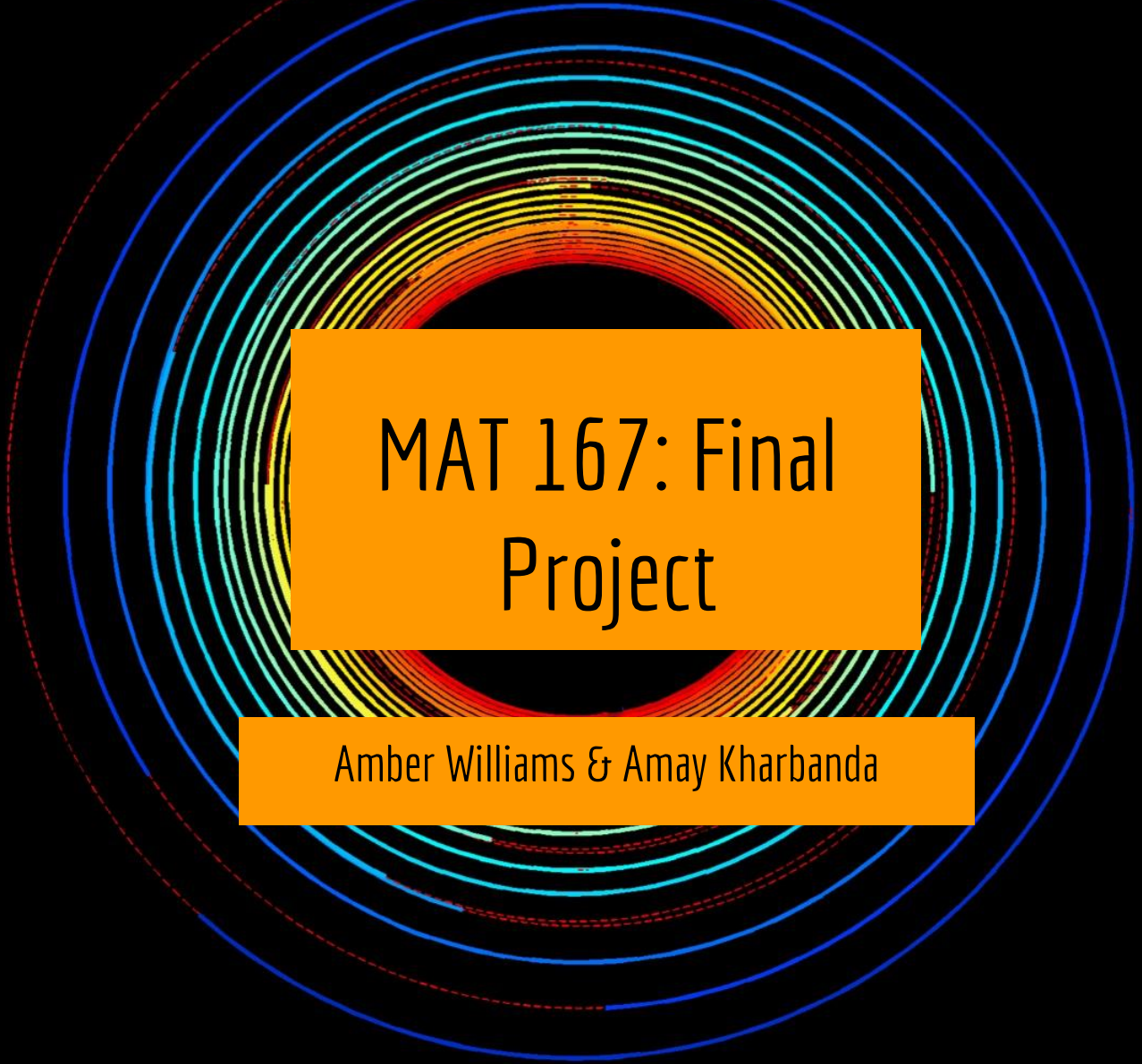


MAT 167: Final Project

Amber Williams & Amay Kharbanda

Statement of the Problem:

The following problem is given from Chapter 5 of Computing with MATLAB by Cleve Moler.

Problem 5.12 states that the orbital pattern of a planet can be represented by the quadratic form expression $z = ax^2 + bxy + cy^2 + dx + ey + f$. This expression takes the (x, y) plane data points and evaluates the expression where $z = 0$.

We are given the data points for a particular planet's elliptical orbit which are the following:

```
x = [1.02 .95 .87 .77 .67 .56 .44 .30 .16 .01]';
y = [0.39 .32 .27 .22 .18 .15 .13 .12 .13 .15]';
```

Statement of the Problem continued:

The problem is then engaged in two parts:

A) The first part of the problem asks for you to determine the coefficients (a,b,c,d, and e) that fits the data by solving a 10x5 overdetermined system of linear equations. This is assuming that the coefficients are in the quadratic form. Then plot the results.

B) The second part then asks for you to perturb the data by adding to each coordinate a uniform random number from the interval of [-0.0005, 0.0005]. Then you solve for the coefficients with the new data points and plot the results along with the previous plot.

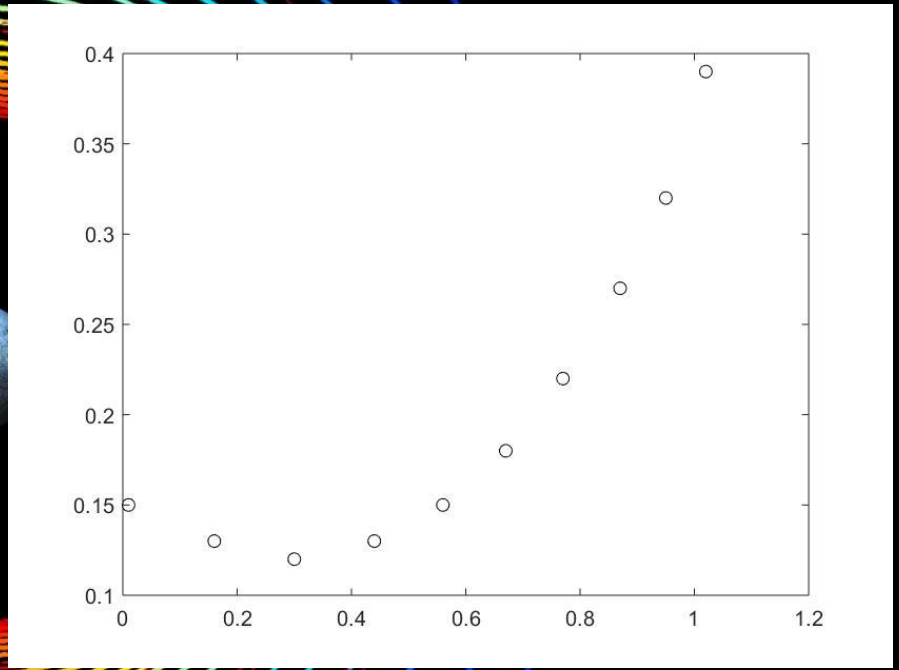


FIGURE 1

Figure 1 is the raw data points plotted.

Data Set and Orbit Equation

For this problem, we decided to create a new data set that was based on true planetary values. We made use of NASA's data set on Heliocentric Trajectories for Selected Spacecraft, Planets, and Comets. This data was converted using the orbit equation for a two-body system consisting of a central body of mass M and a much smaller, orbiting body of mass m in polar coordinate form. The equation is:

$$r = \frac{\ell^2}{m^2 \mu} \frac{1}{1 + e \cos \theta}$$

Where ***r*** is the separation distance between two bodies, ***θ*** is the angle made by ***r*** and the axis of periapsis, i.e. the smallest distance between the orbiter and its host body. ***e*** is the orbit eccentricity of the orbiter.

For simplicity, the term $\frac{\ell^2}{m^2 \mu}$ is considered to be 1.

Data Set and Orbit Equation

For good results, we chose a planet with high eccentricity. We initially decided on Pluto however the θ values were not spread out enough since we have not observed Pluto complete a full orbit. Therefore, Mercury was our best choice, with e = 0.2056. NASA's data set on Heliocentric Trajectories for Selected Spacecraft, Planets, and Comets was an appropriate and reliable source. It provided daily values of θ for many years ranging from 1965 to 2030. We chose 15 values starting from 1971 to 2027 with 4 year increments.

MERCURY (e = 0.2056)	
Year (January 1)	θ (Degrees)
1971	117.2
1975	303.6
1979	202.8
1983	23.4
1987	258.6
1991	137.4
1995	315.8
1999	214.6
2003	44
2007	268.7
2011	155.7
2015	329.1
2019	225.7
2023	66
2027	279

x =	y =
-0.5000	0.9800
0.5000	-0.7500
-1.1400	-0.4800
0.7700	0.3300
-0.2100	-1.0200
-0.8700	0.8000
0.6200	-0.6100
-0.9900	-0.6800
0.6300	0.6100
-0.0200	-1.0000
-1.1200	0.5100
0.7300	-0.4400
-0.8200	-0.8400
0.3800	0.8400
0.1500	-0.9600

This is what the data set looks like after conversion from polar to cartesian coordinates.

Rationale:

The path that results from the data points given tells us that the planet follows an elliptical orbit.

If you set $z = 0$, then you can arbitrarily choose one of your coefficients (usually it's the constant value f) to divide by all of the other coefficients in order to normalize the values. Doing this fixes the value of f at 1 and this reduces the amount of coefficients that you need to solve for to 5 which is $a,b,c,d,$ and e respectively.

However, our system of equations is **overdetermined** because we have 15 x-y pairs and only 5 coefficient values to solve for.

In an effort to satisfy the equation, we can use linear algebra to use a least squares approximation in order to solve for the coefficients.

Problem Solving:

The given quadratic form expression $z = ax^2 + bxy + cy^2 + dx + ey + f$ where $z = 0$, can be rewritten to form two separate matrices like the following:

$[x^2 \;\; xy \;\; y^2 \;\; x \;\; y\,]\,[a\ b\ c\ d\ e]^T = [\,-1\ \ldots.-1]^T \longrightarrow A\mathbf{x} = \mathbf{B}$

Applying all 10 *x-y* pairs in this orientation will create a 10x5 matrix A. We know that our vector **B** is the a 10x1 matrix of all -1 values. So, all we need to do is solve the equation A**x** = **B** for the vector **x** which will give us our coefficients *a,b,c,d,* and *e* respectively.

There are several ways to solve a least squares approximation problem. We chose to use a singular value decomposition (SVD) in order to satisfy the problem.

Singular Value Decomposition

If a matrix A has a matrix of eigenvectors P that is not invertible, then A does not have an eigen decomposition. However, if A is an m×n real matrix with m>n, then A can be written using a so-called singular value decomposition of the form A = USV'.

Starting from (assuming $m > n$ for simplicity)

$$A = U\Sigma V^T = \begin{pmatrix} | & & | \\ u_1 & \cdots & u_m \\ | & & | \end{pmatrix} \begin{pmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_m \\ & & 0 \end{pmatrix} \begin{pmatrix} - & v_1 & - \\ & \vdots & \\ - & v_n & - \end{pmatrix}$$

Solution:

Our first step was to create our matrix A and set our vector **B** equal to -1's. We accomplished this by setting the quadratic equation equal to matrix A in MATLAB and this yielded the following result:

```
theta = [117.2, 303.6, 202.8, 23.4, 258.6, 137.4, 315.8, 214.6, 44, 268.7, 155.7, 329.1, 225.7, 66, 279];
cos_theta = cosd(theta); % Evaluates theta values in degrees
sin_theta = sind(theta);
e = 0.2056; % Eccentricity of Mercury
x = cos_theta./(1 + e*cos_theta); % Polar to cartesian coordinates formula
y = sin_theta./(1 + e*cos_theta);
x = x';
y = y';

m = length(x); % The number of data equations

B = -ones(size(x)); % Creates matrix B that is 15 x 1
A = [x.^2, x.*y, y.^2, x, y]; % Creates the matrix A that is 15 x 5
```

```
A =

    0.2545   -0.4953    0.9637   -0.5045    0.9817
    0.2469   -0.3716    0.5593    0.4969   -0.7478
    1.2938    0.5439    0.2286   -1.1374   -0.4781
    0.5961    0.2580    0.1116    0.7721    0.3341
    0.0424    0.2105    1.0441   -0.2060   -1.0218
    0.7523   -0.6918    0.6361   -0.8674    0.7976
    0.3904   -0.3796    0.3692    0.6248   -0.6076
    0.9817    0.6772    0.4672   -0.9908   -0.6835
    0.3927    0.3792    0.3662    0.6267    0.6052
    0.0005    0.0229    1.0089   -0.0228   -1.0044
    1.2579   -0.5680    0.2564   -1.1216    0.5064
    0.5320   -0.3184    0.1906    0.7294   -0.4365
    0.6651    0.6815    0.6984   -0.8155   -0.8357
    0.1409    0.3164    0.7107    0.3753    0.8430
    0.0230   -0.1450    0.9157    0.1516   -0.9569
```

Solution continued:

```
% Singular Value Decomposition (SVD) code
[U,S,V] = svd(A);
C = U' * B;
coefficient_y = [C(1)/S(1,1); C(2)/S(2,2); C(3)/S(3,3); C(4)/S(4,4); C(5)/S(5,5)];
coefficient_x = V * coefficient_y; % Solution

% Coefficients
a = coefficient_x(1)
b = coefficient_x(2)
c = coefficient_x(3)
d = coefficient_x(4)
e = coefficient_x(5)
f = 1

% Conic section function along with coefficients
F = @(x,y) a*x.^2 + b*x.*y + c*y.^2 + d*x + e*y + f;

plot(x,y, 'ok')
hold on % To combine plots
fimplicit(F) % Implicit function plot
```

To find the coefficients, we chose to apply the Singular Value Decomposition method.

We factorized A into the product of three matrices A = **USV'** where the columns of **U** and **V** are orthonormal and the matrix **S** is diagonal with positive real entries.

Coefficients are stored into variables and f is equated to 1.

F is the function definition and fimplicit() was applied to plot the equation.

Solution continued:

Coefficient values are as follows:

Norm of the least squares equation as follows:

Error term to compute the approximation error of matrix A is as follows:

```
coefficient_x =

   -0.9577
   -0.0000
   -1.0000
   -0.4112
    0.0000
```

```
>> norm(A*coefficient_x - B)

ans =

   1.1484e-15
```

```
>> E = norm(A*coefficient_x - B)/norm(A)

E =

   3.1922e-16
```

Solution continued:

Part B of the problem asked us to perturb the original x and y data by adding a random number from a uniform distribution. That is done here by creating the boundaries of the interval and setting it equal to ±0.0005. For this specific data set, we decided to amplify the perturbations (by 100) in order to visualize them better.

Then you call the rand() function in MATLAB to generate the new randomized *x-y* matrix values of mx1 size.

The perturbed A matrix is as follows:

```
% Part B (perturbation)

min_val = -0.0005;
max_val = -min_val;

 % Multiplied by 100 to make perturbations visible on plot
x_pert = x + min_val+rand(m,1)*(max_val-min_val)*100;
y_pert = y + min_val+rand(m,1)*(max_val-min_val)*100;

% Creates matrix B that is 15 x 1
B_pert = -ones(size(x_pert));
% Creates matrix A that is 15 x 5
A_pert = [x_pert.^2, x_pert.*y_pert, y_pert.^2, x_pert, y_pert];
```

```
A_pert =

    0.1794   -0.4216    0.9907   -0.4235    0.9954
    0.3445   -0.4145    0.4987    0.5869   -0.7062
    1.2662    0.4355    0.1498   -1.1253   -0.3871
    0.7446    0.3562    0.1704    0.8629    0.4128
    0.0205    0.1327    0.8581   -0.1433   -0.9263
    0.7364   -0.7403    0.7442   -0.8581    0.8627
    0.4253   -0.3943    0.3655    0.6522   -0.6045
    0.8773    0.5611    0.3589   -0.9366   -0.5991
    0.5212    0.5039    0.4873    0.7219    0.6981
    0.0054   -0.0686    0.8781    0.0732   -0.9371
    1.2239   -0.6435    0.3384   -1.1063    0.5817
    0.6822   -0.2996    0.1316    0.8259   -0.3627
    0.5188    0.5741    0.6352   -0.7203   -0.7970
    0.1793    0.3845    0.8246    0.4234    0.9081
    0.0534   -0.2173    0.8842    0.2311   -0.9403
```

Solution continued:

Following the same logic as above, we used SVD to solve for the following new perturbed coefficients:

```
% SVD
[U_pert,S_pert,V_pert] = svd(A_pert);
C_pert = U_pert' * B_pert;
coefficient_y_pert = [C_pert(1)/S_pert(1,1); C_pert(2)/S_pert(2,2);
                      C_pert(3)/S_pert(3,3); C_pert(4)/S_pert(4,4);
                      C_pert(5)/S_pert(5,5)];
coefficient_x_pert = V_pert * coefficient_y_pert;

% Coefficients
a_pert = coefficient_x_pert(1)
b_pert = coefficient_x_pert(2)
c_pert = coefficient_x_pert(3)
d_pert = coefficient_x_pert(4)
e_pert = coefficient_x_pert(5)
f_pert = 1

% Perturbed equation
F_pert = @(x_pert,y_pert) a_pert*x_pert.^2 + b_pert*x_pert.*y_pert
                          + c_pert*y_pert.^2 + d_pert*x_pert +
                          e_pert*y_pert + f_pert;

fimplicit(F_pert)
plot(x_pert,y_pert, 'xr') % plot the randomized data
hold off
```

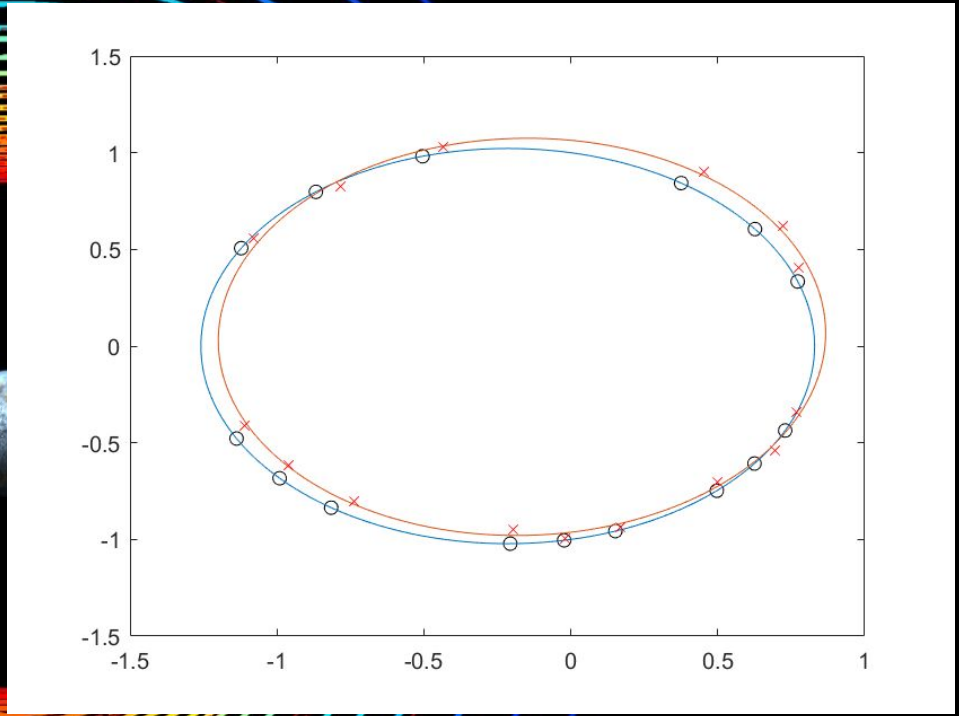
```
coefficient_x_pert =

   -0.8562
   -0.0309
   -1.0044
   -0.2529
    0.1129
```

*Note: because we used rand () the values will change slightly everytime the code is run.

Solution continued:

The graph of perturbed and normal data points and equations yielded the following:



The perturbed data resulted in visible displacement from the original data. This is because the random generator with the parameter of ±0.0005 yield a lot of noise (multiplied by 100).

```
min_val = -0.0005;
max_val = -min_val;

 % Multiplied by 100 to make perturbations visible on plot
x_pert = x + min_val+rand(m,1)*(max_val-min_val)*100;
y_pert = y + min_val+rand(m,1)*(max_val-min_val)*100;
```

Solution continued:

Error analysis

```
>> c_num = cond(A);
sing_ratio = S(1,1)/S(5,5);
percent_difference=abs(c_num - sing_ratio)./c_num*100

percent_difference =

   1.8032e-14

>> c_num_pert = cond(A_pert);
sing_ratio_pert = S_pert(1,1)/S_pert(5,5);
percent_difference_pert =abs(c_num_pert - sing_ratio_pert)./c_num_pert*100

percent_difference_pert =

   4.1294e-14
```

```
>> E = norm(A_pert*coefficient_x_pert - B_pert)/norm(A_pert)

E =

    0.0514
```

```
>> norm (A_pert*coefficient_x_pert - B_pert)

ans =

    0.1705
```

```
>> cond(A)

ans =

    2.4628

>> cond(A_pert)

ans =

    2.1509
```

Conclusion

- The low error values imply that the Single Value Decomposition method was able to approximate matrix A accurately.
- Moreover, the norm of the least squares equation had a significantly low value, suggesting that the analysis was valid.
- The precision of the equation plot to the data plot indicates a correct value for the coefficients and completes the equation of an ellipse.

References

- Mathworks
- Lectures 16, 17, 18, 19
- Fetter, Alexander; Walecka, John (2003). Theoretical Mechanics of Particles and Continua. Dover Publications. pp. 13–22.
- Moler Ch. 5
- NASA: Heliocentric Trajectories for Selected Spacecraft, Planets, and Comets https://omniweb.gsfc.nasa.gov/coho/helios/planet.html

```
theta = [117.2, 303.6, 202.8, 23.4, 258.6, 137.4, 315.8, 214.6,
44, 268.7, 155.7, 329.1, 225.7, 66, 279];
cos_theta = cosd(theta); % Evaluates theta values in degrees
sin_theta = sind(theta);
e = 0.2056; % Eccentricity of Mercury
x = cos_theta./(1 + e*cos_theta); % Polar to cartesian
coordinates formula
y = sin_theta./(1 + e*cos_theta);
x = x';
y = y';

m = length(x); % The number of data equations

B = -ones(size(x)); % Creates matrix B that is 15 x 1
A = [x.^2, x.*y, y.^2, x, y]; % Creates the matrix A that is 15
x 5

% Singular Value Decomposition (SVD) code
[U,S,V] = svd(A);
C = U' * B;
coefficient_y = [C(1)/S(1,1); C(2)/S(2,2); C(3)/S(3,3);
C(4)/S(4,4); C(5)/S(5,5)];
coefficient_x = V * coefficient_y; % Solution

% Error Analysis

c_num = cond(A);
sing_ratio = S(1,1)/S(5,5);
percent_difference = abs(c_num - sing_ratio)./c_num*100

% Coefficients
a = coefficient_x(1)
b = coefficient_x(2)
c = coefficient_x(3)
d = coefficient_x(4)
e = coefficient_x(5)
f = 1

% Conic section function along with coefficients
F = @(x,y) a*x.^2 + b*x.*y + c*y.^2 + d*x + e*y + f;

plot(x,y, 'ok')
hold on % To combine plots
fimplicit(F) % Implicit function plot

% Part B (perturbation)
```

```
min_val = -0.0005;
max_val = -min_val;

 % Multiplied by 100 to make perturbations visible on plot
x_pert = x + min_val+rand(m,1)*(max_val-min_val)*100;
y_pert = y + min_val+rand(m,1)*(max_val-min_val)*100;

% Creates matrix B that is 15 x 1
B_pert = -ones(size(x_pert));
% Creates matrix A that is 15 x 5
A_pert = [x_pert.^2, x_pert.*y_pert, y_pert.^2, x_pert, y_pert];

% SVD
[U_pert,S_pert,V_pert] = svd(A_pert);
C_pert = U_pert' * B_pert;
coefficient_y_pert = [C_pert(1)/S_pert(1,1);
C_pert(2)/S_pert(2,2); C_pert(3)/S_pert(3,3);
C_pert(4)/S_pert(4,4); C_pert(5)/S_pert(5,5)];
coefficient_x_pert = V_pert * coefficient_y_pert;

% Error Analysis
c_num_pert = cond(A_pert);
sing_ratio_pert = S_pert(1,1)/S_pert(5,5);
percent_difference_pert = abs(c_num_pert -
sing_ratio_pert)./c_num_pert*100

% Coefficients
a_pert = coefficient_x_pert(1)
b_pert = coefficient_x_pert(2)
c_pert = coefficient_x_pert(3)
d_pert = coefficient_x_pert(4)
e_pert = coefficient_x_pert(5)
f_pert = 1

% Perturbed equation
F_pert = @(x_pert,y_pert) a_pert*x_pert.^2 +
b_pert*x_pert.*y_pert + c_pert*y_pert.^2 + d_pert*x_pert +
e_pert*y_pert + f_pert;

fimplicit(F_pert)
plot(x_pert,y_pert, 'xr') % plot the randomized data
hold off
```