Author: Shantanu Tyagi

Date: 10-03-2021

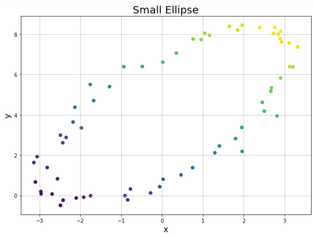
ID: 201801015

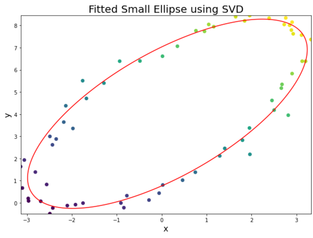
In [1]:
```
import pandas as pd
from matplotlib import pyplot as plt
import numpy as np
import math
from scipy import stats
```

In [9]:
```
def preprocess(data):
    x = []
    y = []
    for d in data:
        read = pd.read_csv(d)
        x.append(np.array(read['xn']))
        y.append(np.array(read['yn']))
    return x,y

def display_data(x,y,name):
    plt.figure(figsize=(10, 7))
    plt.scatter(x, y, c=x+y, cmap="viridis")
    plt.title(name, fontsize=20)
    plt.xlabel('x', fontsize=16)
    plt.ylabel('y', fontsize=16)
    plt.grid(alpha=0.75)
    plt.show()

def display_fit(x,y,x_plt,y_plt,X,name,method):
    plt.figure(figsize=(10, 7))
    plt.contour(x_plt,y_plt,X,levels=[0],colors='r')
    plt.scatter(x,y, c=x+y, cmap="viridis")
    plt.xlabel('x', fontsize=16)
    plt.ylabel('y', fontsize=16)
    plt.title('Fitted ' + name + ' using ' + method, fontsize=20)
    plt.show()

def rmse(x_new,x,y):
    rmse = 0
    for i in range(len(x)):
        rmse = rmse + (x_new[0]*x[i]*x[i] + x_new[1]*x[i]*y[i] + x_new[2]*y[i]*y[i] + x_new[3]*x[i] + x_new[4]*y[i]
+ x_new[5])**2
        rmse = rmse/len(x)
        rmse = math.sqrt(rmse)
    return rmse

def do_SVD(x,y,name):
    x_2 = x*x
    y_2 = y*y
    xy = x*y
    on = np.zeros(len(x)) + 1
    b = np.zeros(len(x))
    mat = np.column_stack((x_2,xy,y_2,x,y,on))
    U , s , V = np.linalg.svd(mat)
    index = np.argmin(abs(s))
    vt = np.transpose(V)
    x_new = vt[:,index]
    x_plt = np.linspace(np.min(x),np.max(x),300)
    y_plt = np.linspace(np.min(y),np.max(y),300)
    x_plt,y_plt = np.meshgrid(x_plt,y_plt)
    X = x_new[0]*x_plt*x_plt + x_new[1]*x_plt*y_plt + x_new[2]*y_plt*y_plt + x_new[3]*x_plt + x_new[4]*y_plt + x_new
[5]
    display_fit(x,y,x_plt,y_plt,X,name,'SVD')
    print('RMSE = ' + str(rmse(x_new,x,y)))

def do_eigen(x,y,name):
    C = np.column_stack(([0,0,-2,0,0,0],[0,1,0,0,0,0],[-2,0,0,0,0,0],[0,0,0,0,0,0],[0,0,0,0,0,0],[0,0,0,0,0,0]))
    x_2 = x*x
    y_2 = y*y
    xy = x*y
    on = np.zeros(len(x)) + 1
    mat = np.column_stack((x_2,xy,y_2,x,y,on))
    s = np.transpose(mat) @mat
    w,v = np.linalg.eig(np.linalg.inv(s) @C)
    index = np.argmax(abs(w))
    x_new = v[:,index]
    x_plt = np.linspace(np.min(x),np.max(x),300)
    y_plt = np.linspace(np.min(y),np.max(y),300)
    x_plt,y_plt = np.meshgrid(x_plt,y_plt)
    X = x_new[0]*x_plt*x_plt + x_new[1]*x_plt*y_plt + x_new[2]*y_plt*y_plt + x_new[3]*x_plt + x_new[4]*y_plt + x_new
[5]
    display_fit(x,y,x_plt,y_plt,X,name,'Eigen Values')
    print('RMSE = ' + str(rmse(x_new,x,y)))
```

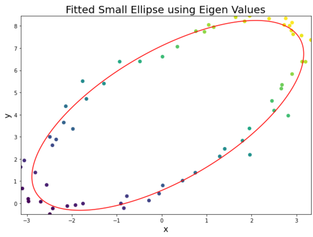
The singular value decomposition (SVD) of a matrix A is very useful in the context of least squares problems. Fitting circles and ellipses to given points in the plane is a problem that arises in many application areas, e.g. computer graphics, coordinate metrology, petroleum engineering, statistics. In the past, algorithms have been given which fit circles and ellipses in some least squares sense without minimizing the geometric distance to the given points. The two methods that we will see, compute the ellipse for which the sum of the squares of the distances to the given points is minimal.

In [10]:
```
data = ['ellipse-data.csv','ellipse-data_large.csv']
name= ['Small Ellipse','Large Ellipse']
x,y = preprocess(data)
for i in range(len(data)):
    display_data(x[i],y[i],name[i])
    do_SVD(x[i],y[i],name[i])
    do_eigen(x[i],y[i],name[i])
```

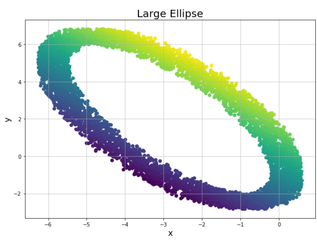


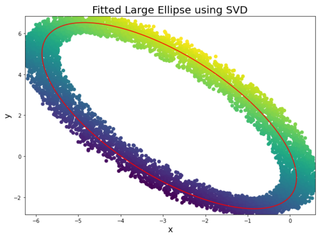


RMSE = 0.017213682616145604

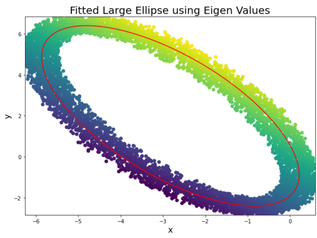


RMSE = 0.019455238548189983





RMSE = 0.0008534984321315696



RMSE = 0.0006145730547464993

We see that for the smaller ellipse, SVD lesser value for RMSE but on the other hand it gives a larger value of RMSE in case of larger Ellipse.

In [ ]: