problem set 5实验报告

16337183 孟衍璋

Part A

Problem 1

实现拟合曲线的函数，需要满足不同阶数的拟合。

```
def generate_models(x, y, degs):
    coefficients = []  # 存储系数
    for i in range(len(degs)):
        coefficients.append(pylab.polyfit(x,y,degs[i]))
    # print(coefficients)
    return coefficients
```

Problem 2

实现计算R^2的函数。

```
def r_squared(y, estimated):
    mean = y.mean()  # 计算y的平均值
    temp1 = (y - estimated) ** 2
    temp2 = (y - mean) ** 2
    return 1 - temp1.sum() / temp2.sum()
```

Problem 3

实现可视化的函数。

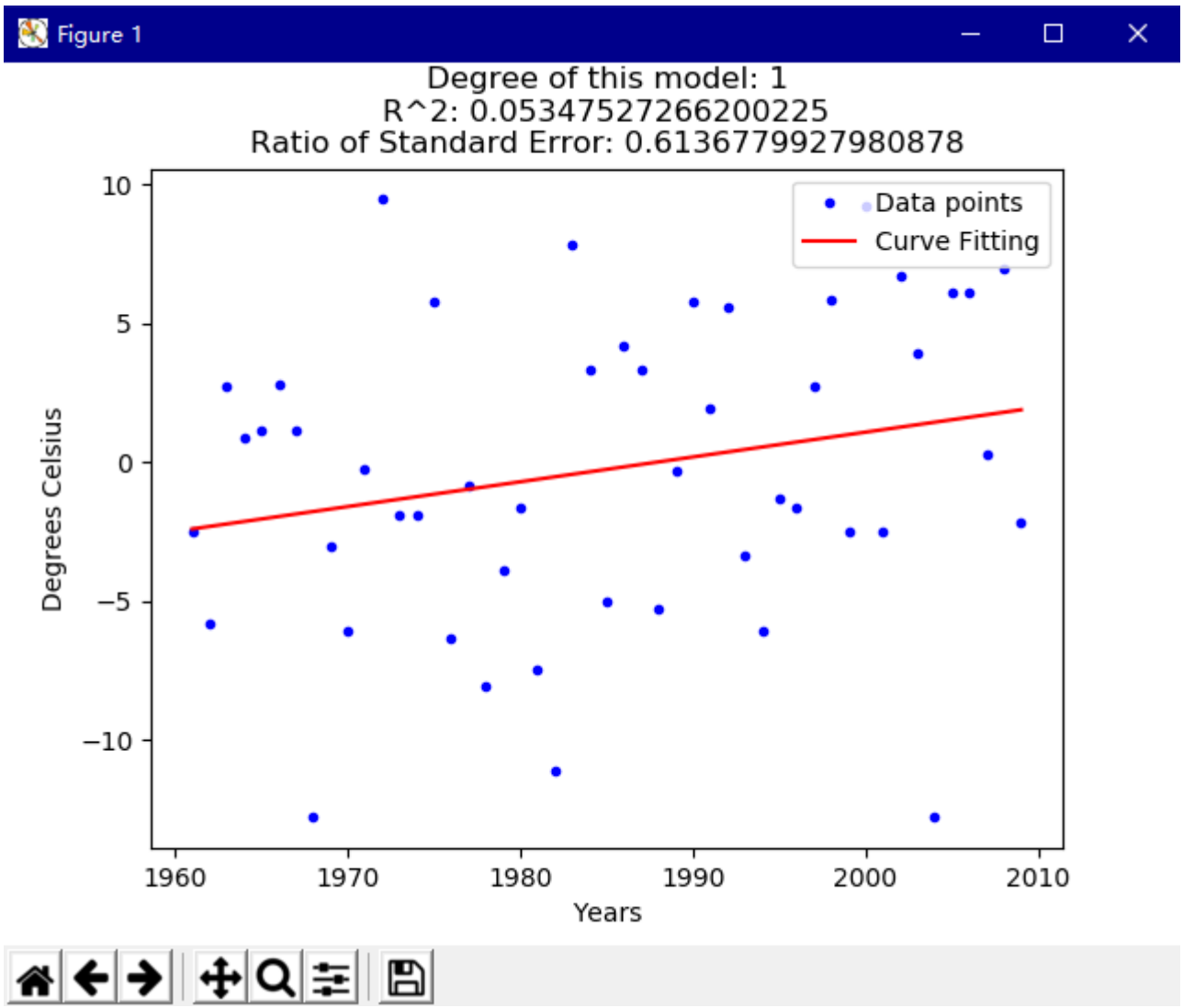
```
def evaluate_models_on_training(x, y, models):
    for model in models:
        p = pylab.poly1d(model)
        r_2 = r_squared(y, p(x))  # 计算r^2
        pylab.figure()
        pylab.plot(x, y, 'b.', label = 'Data points')  # 用蓝色的散点代表数据点
        pylab.plot(x, p(x), 'r-', label = 'Curve Fitting')  # 用红色的实线代表拟合的曲线
        pylab.legend()
        if len(model) == 2:  # 如果是一条直线，即最高项次数为1
```

```
            pylab.title('Degree of this model: ' + str(len(model) - 1) + '\n' + 'R^2: ' +
str(r_2) + '\n' + 'Ratio of Standard Error: ' + str(se_over_slope(x, y, p(x), model)))
        else:  # 如果最高项次数大于1
            pylab.title('Degree of this model: ' + str(len(model) - 1) + '\n' + 'R^2: ' +
str(r_2))
        pylab.xlabel('Years')
        pylab.ylabel('Degrees Celsius')
        pylab.show()
```

Problem 4

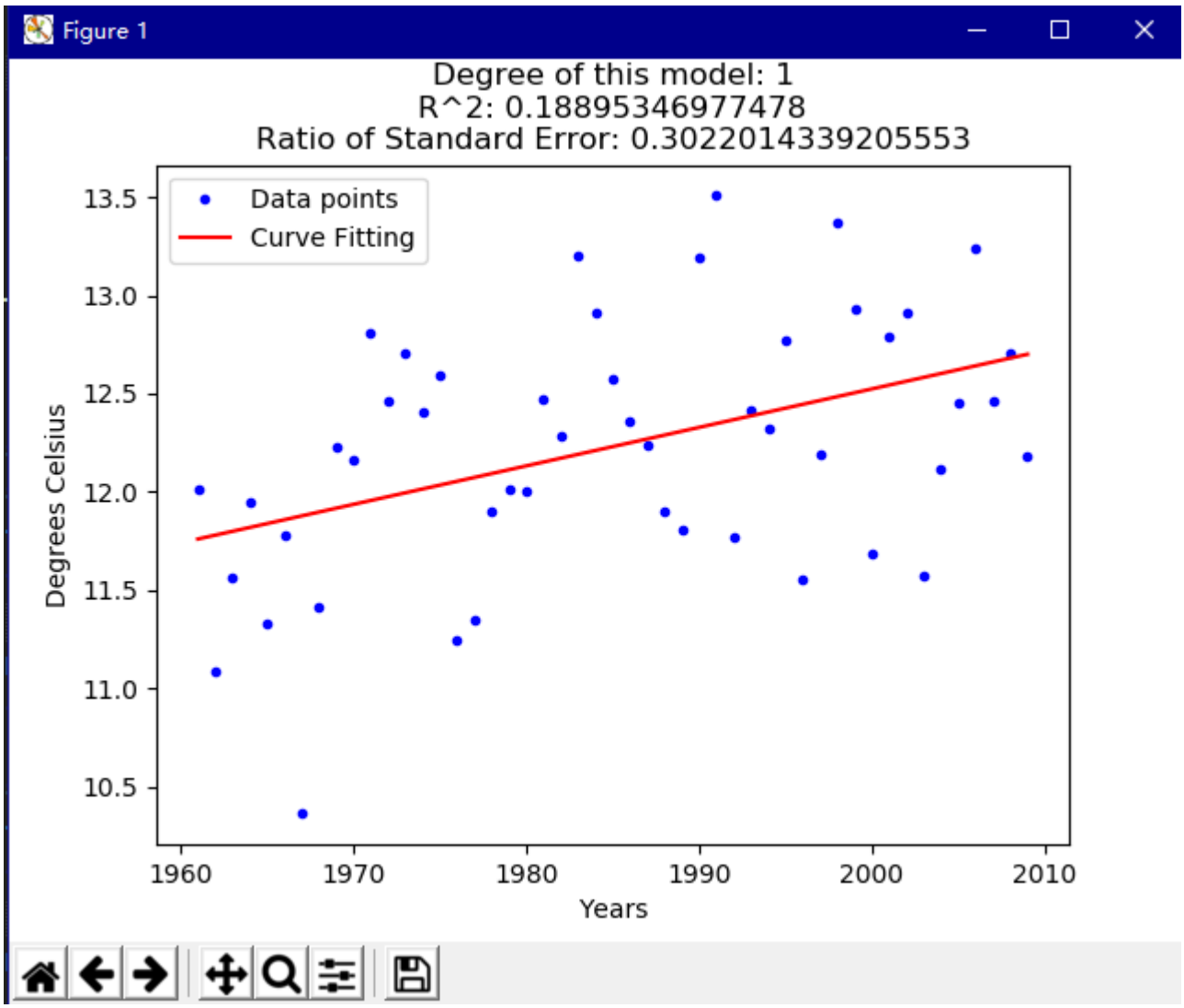
A.4 I

根据纽约每年1月10日的气温拟合曲线。



A.4 II

根据纽约每年的平均气温拟合曲线。



- What difference does choosing a specific day to plot the data for versus calculating the yearly average have on our graphs (i.e., in terms of the R^2 values and the fit of the resulting curves)? Interpret the results.

  第二幅图得到的R^2的值更大，意味着计算年平均值的拟合效果更好。

- Why do you think these graphs are so noisy? Which one is more noisy?

  因为拟合曲线的阶数只是1，无法很好地表达出数据的实际趋势。从图可以看出，第一幅图噪声更多。

- How do these graphs support or contradict the claim that global warming is leading to an increase in temperature? The slope and the standard error-to-slope ratio could be helpful in thinking about this.

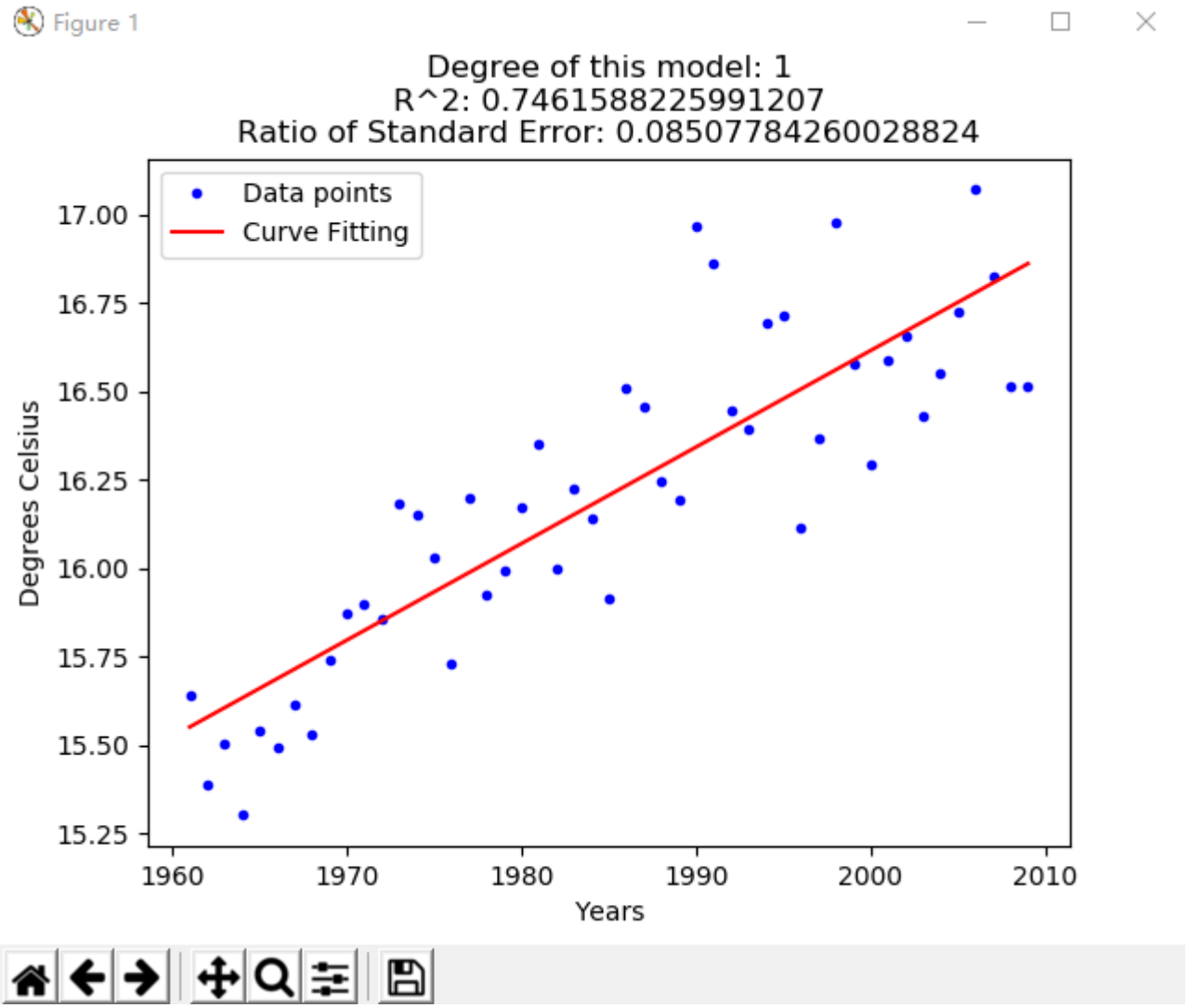
  因为拟合出的直线斜率是正数，所以这表明了多年来温度是呈上升趋势的。

Part B

实现函数`gen_cities_avg`，用来求某一年全部21座城市的平均气温。

```python
def gen_cities_avg(climate, multi_cities, years):
    temperature_data = []
    for year in years:
        total_temp = 0  # 某一年所有城市的总气温
        for city in multi_cities:
            total_temp += climate.get_yearly_temp(city, year).mean()
        avg_temp = total_temp / len(multi_cities)  # 某一年所有城市的平均气温
        temperature_data.append(avg_temp)
    return temperature_data
```

根据全部21座城市的平均气温拟合曲线。



- How does this graph compare to the graphs from part A (i.e., in terms of the R^2 values, t he fit of the resulting curves, and whether the graph supports/contradicts our claim about global warming)? Interpret the results.

与Part A部分得到的结果进行对比，该图R^2的值更大，曲线的拟合效果也更好，且直线斜率为正数，因此支持全球变暖的说法。

- Why do you think this is the case?

因为各个城市的年平均气温能更好地代表某年的温度情况。

- How would we expect the results to differ if we used 3 different cities? What about 100 different cities?

  如果使用3个城市的数据，得到的结果应该更分散；如果使用100个城市的数据，得到的结果应该更集中。

- How would the results have changed if all 21 cities were in the same region of the United States (for ex., New England)?

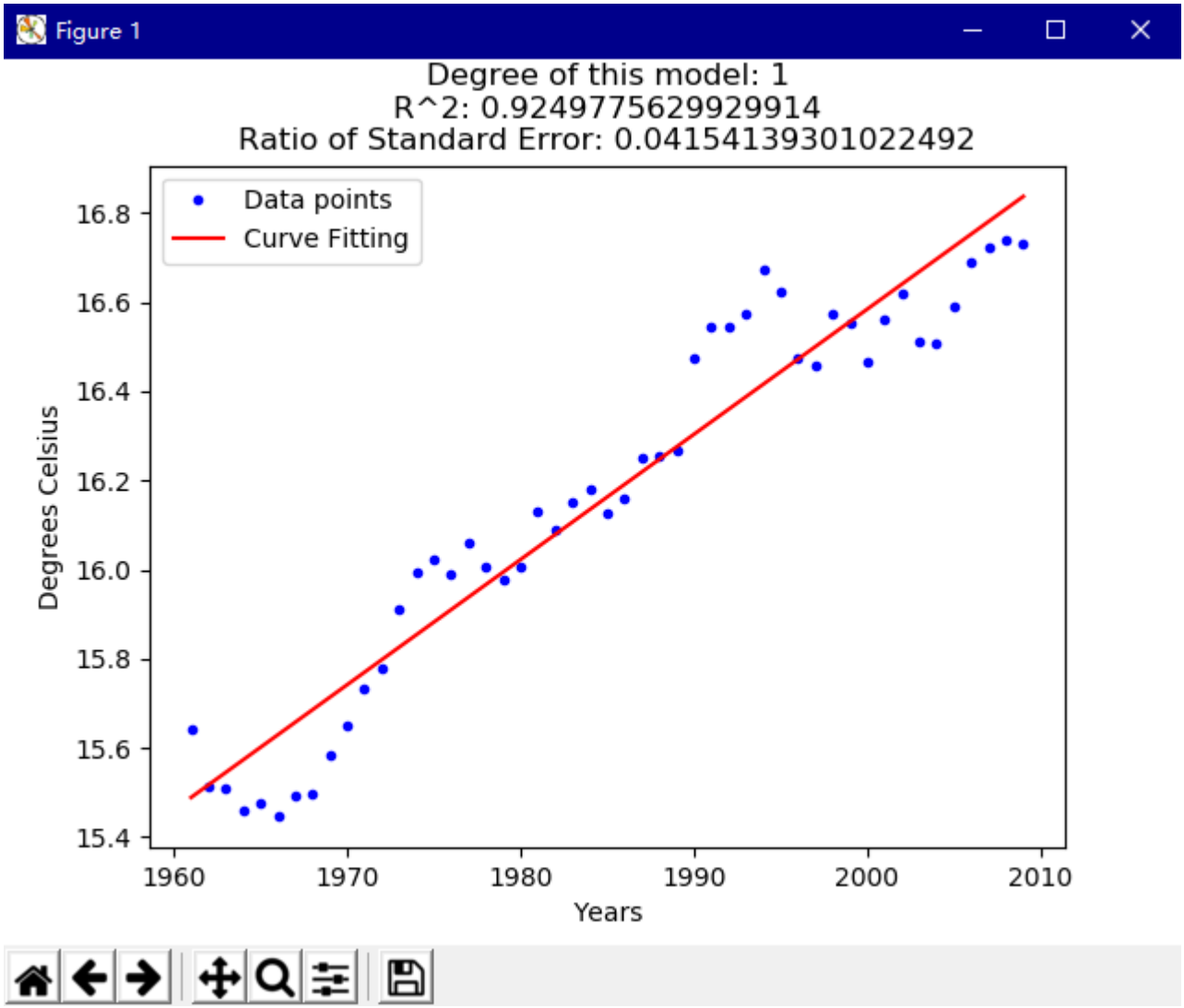
  数据的意义没有之前那么大，拟合的效果应该没有之前的好。

Part C

实现`moving_average`函数，用来求指定滑动窗口长度的平均值。

```
def moving_average(y, window_length):
    moving_avg = []
    for i in range(1, len(y) + 1):
        if i < window_length:  # 如果当前长度小于window_length，则只计算目前已经存在的数的平均数
            moving_avg.append(pylab.array(y[:i]).mean())
        else:  # 如果当前长度大于等于window_length，则选取window_length个数，求平均值
            moving_avg.append(pylab.array(y[i - window_length:i]).mean())
    return moving_avg
```

根据21座城市按照5年为滑动窗口长度的平均值拟合曲线。



- How does this graph compare to the graphs from part A and B (i.e., in terms of the R^2 values, the fit of the resulting curves, and whether the graph supports/contradicts our claim about global warming)? Interpret the results.

这幅图与Part A与Part B的图进行比较，R^2的值更大，拟合效果更好，且明显可以看出气温逐年升高，支持全球变暖的说法。

- Why do you think this is the case?

因为moving_average的数据更加具有普遍性，消除了特殊情况的影响。

Part D

Problem 1

实现计算均方根误差（Root Mean Square Error）的函数。

```
def rmse(y, estimated):
    return math.sqrt(((y - estimated) ** 2).sum() / len(y))
```

实现评估测试的模型。

```
def evaluate_models_on_testing(x, y, models):
    for model in models:
        p = pylab.poly1d(model)
        RMSE = rmse(y, p(x))  # 计算rmse
        pylab.figure()
        pylab.plot(x, y, 'b.', label = 'Data points')  # 用蓝色的散点代表数据点
        pylab.plot(x, p(x), 'r-', label = 'Curve Fitting')  # 用红色的实线代表拟合的曲线
        pylab.legend()
        pylab.title('Degree of this model: ' + str(len(model) - 1) + '\n' + 'RMSE: ' +
str(RMSE))
        pylab.xlabel('Years')
        pylab.ylabel('Degrees Celsius')
        pylab.show()
```

Problem 2

D2. I

生成阶数为1、2、20的模型，根据21座城市按照5年为滑动窗口长度的平均值拟合曲线。

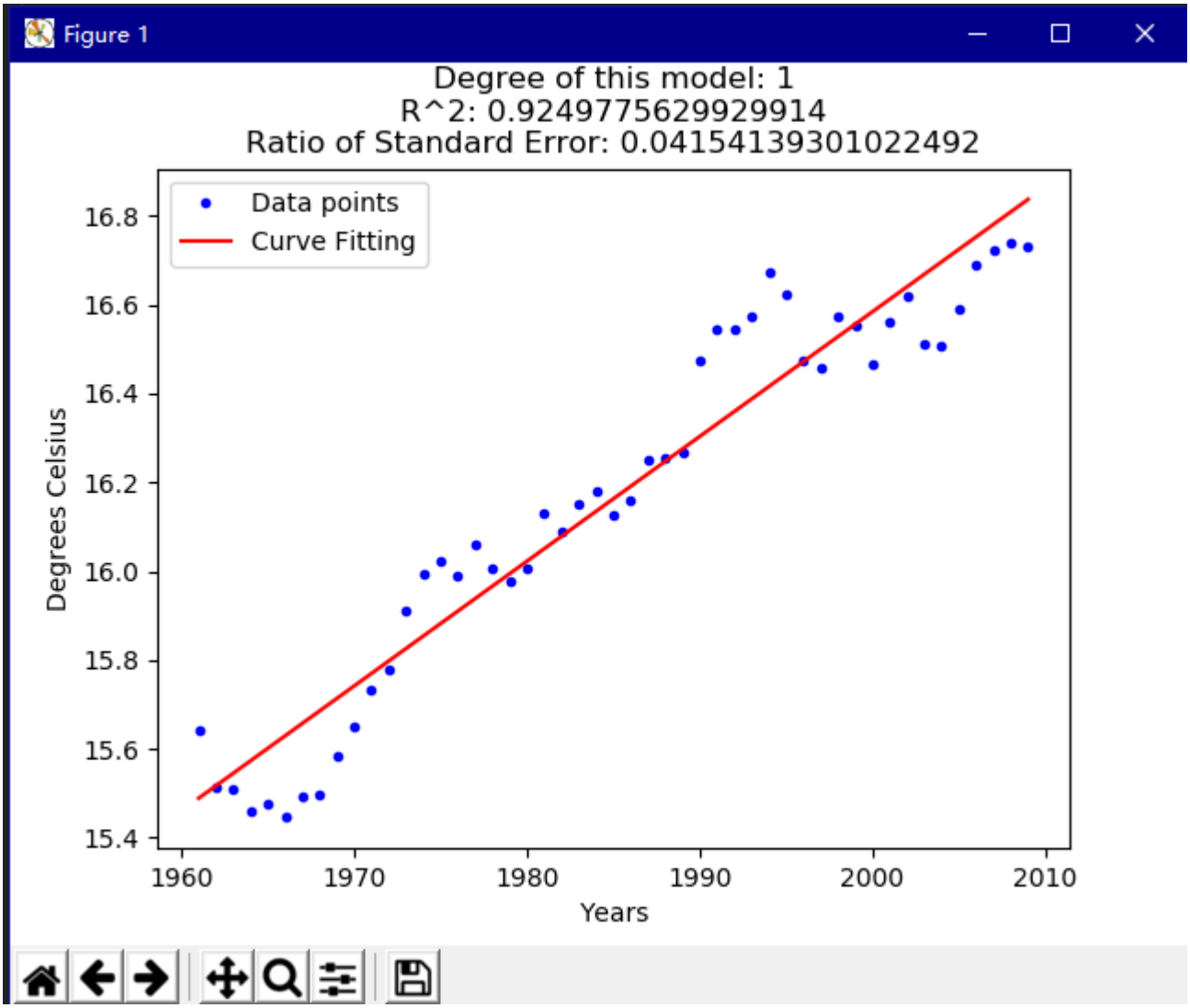


Figure 1
Degree of this model: 1
R^2: 0.9249775629929914
Ratio of Standard Error: 0.04154139301022492
Data points
Curve Fitting
Degrees Celsius
Years
16.8
16.6
16.4
16.2
16.0
15.8
15.6
15.4
1960
1970
1980
1990
2000
2010

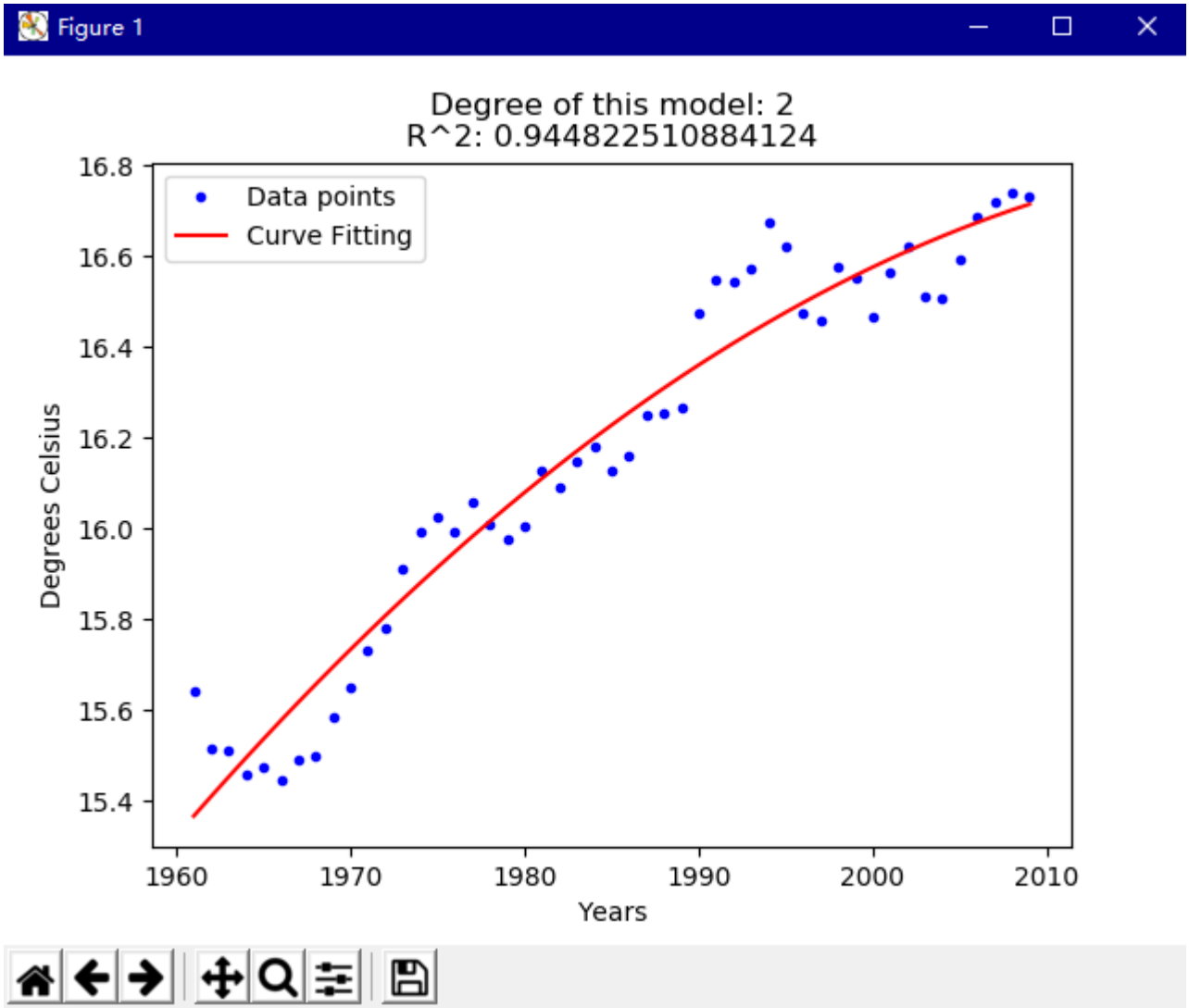
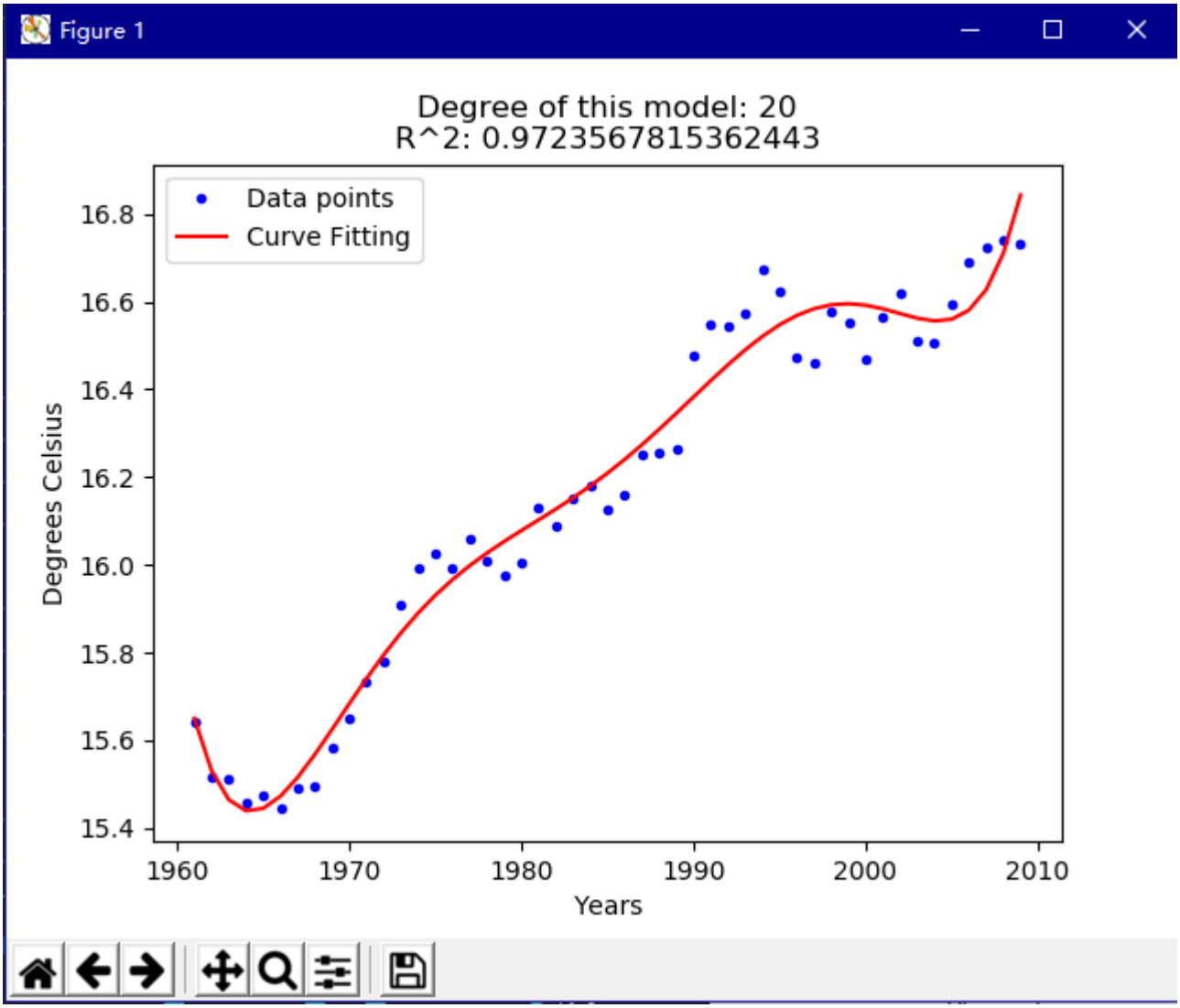


Figure 1
Degree of this model: 2
R^2: 0.944822510884124
Data points
Curve Fitting
Degrees Celsius
Years
16.8
16.6
16.4
16.2
16.0
15.8
15.6
15.4
1960
1970
1980
1990
2000
2010



- How do these models compare to each other?

  这些模型相比较，都很好地模拟了数据的趋势，但阶数为20的模型拟合得最好。

- Which one has the best R^2? Why?

  阶数为20的模型的R^2的值最好，因为其拟合数据点的程度最好。

- Which model best fits the data? Why?

  阶数为20的模型最好地拟合了数据，因为其R^2的值最好。

D2.II

根据21座城市按照5年为滑动窗口长度的平均值拟合曲线，来预测2010-2015年的数据。

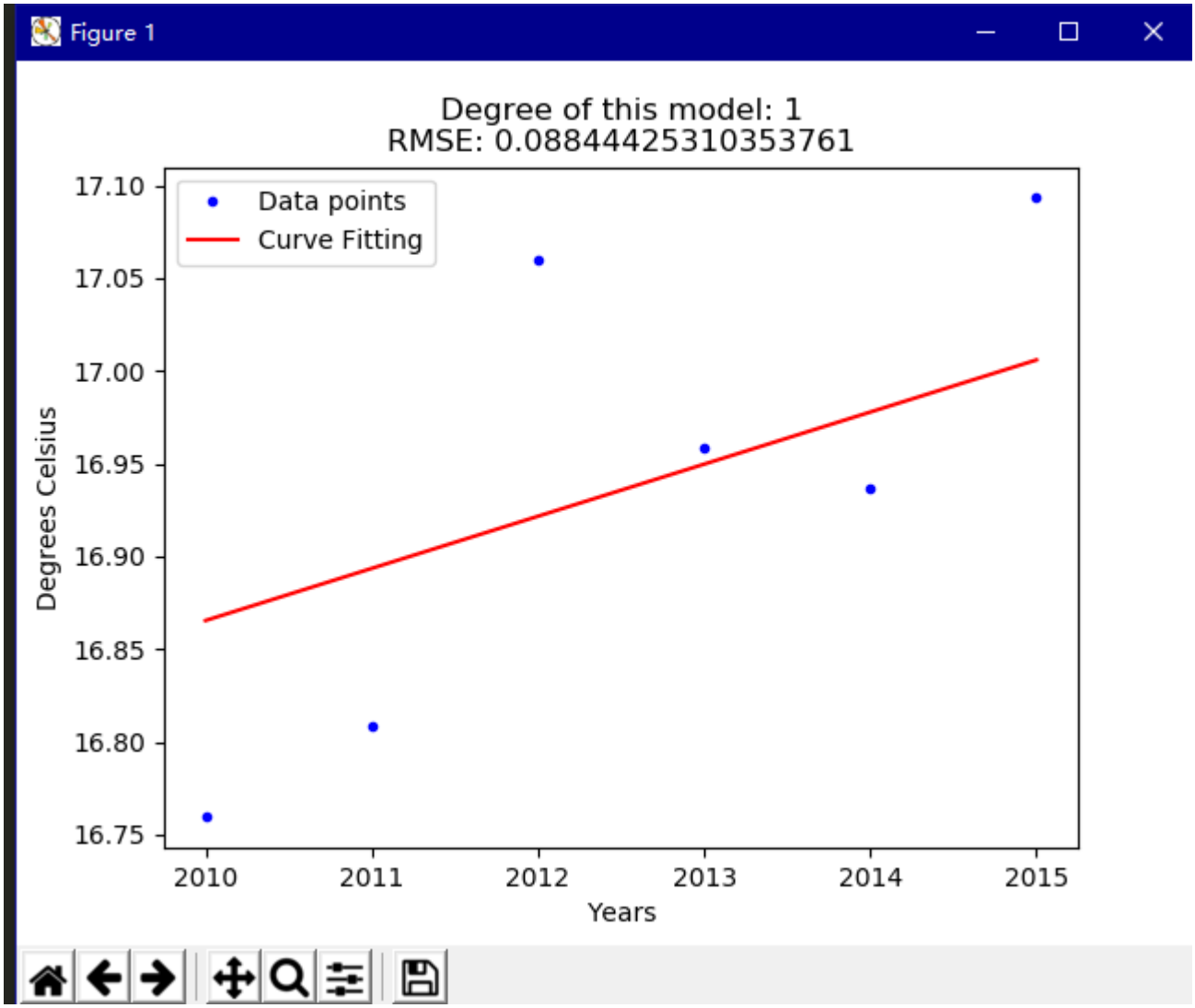


Figure 1
Degree of this model: 1
RMSE: 0.08844425310353761
Data points
Curve Fitting
17.10
17.05
17.00
16.95
16.90
16.85
16.80
16.75
Degrees Celsius
2010
2011
2012
2013
2014
2015
Years

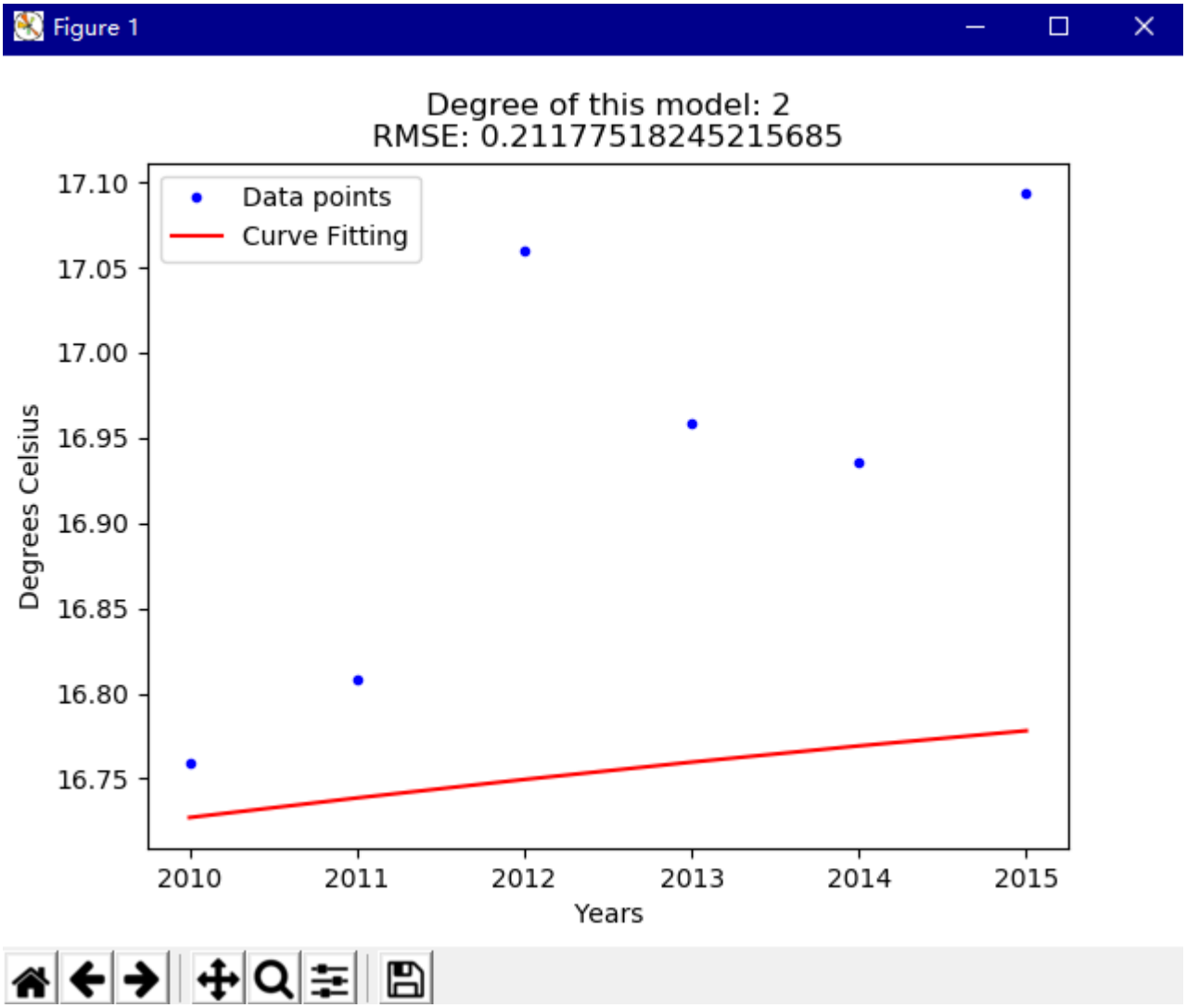
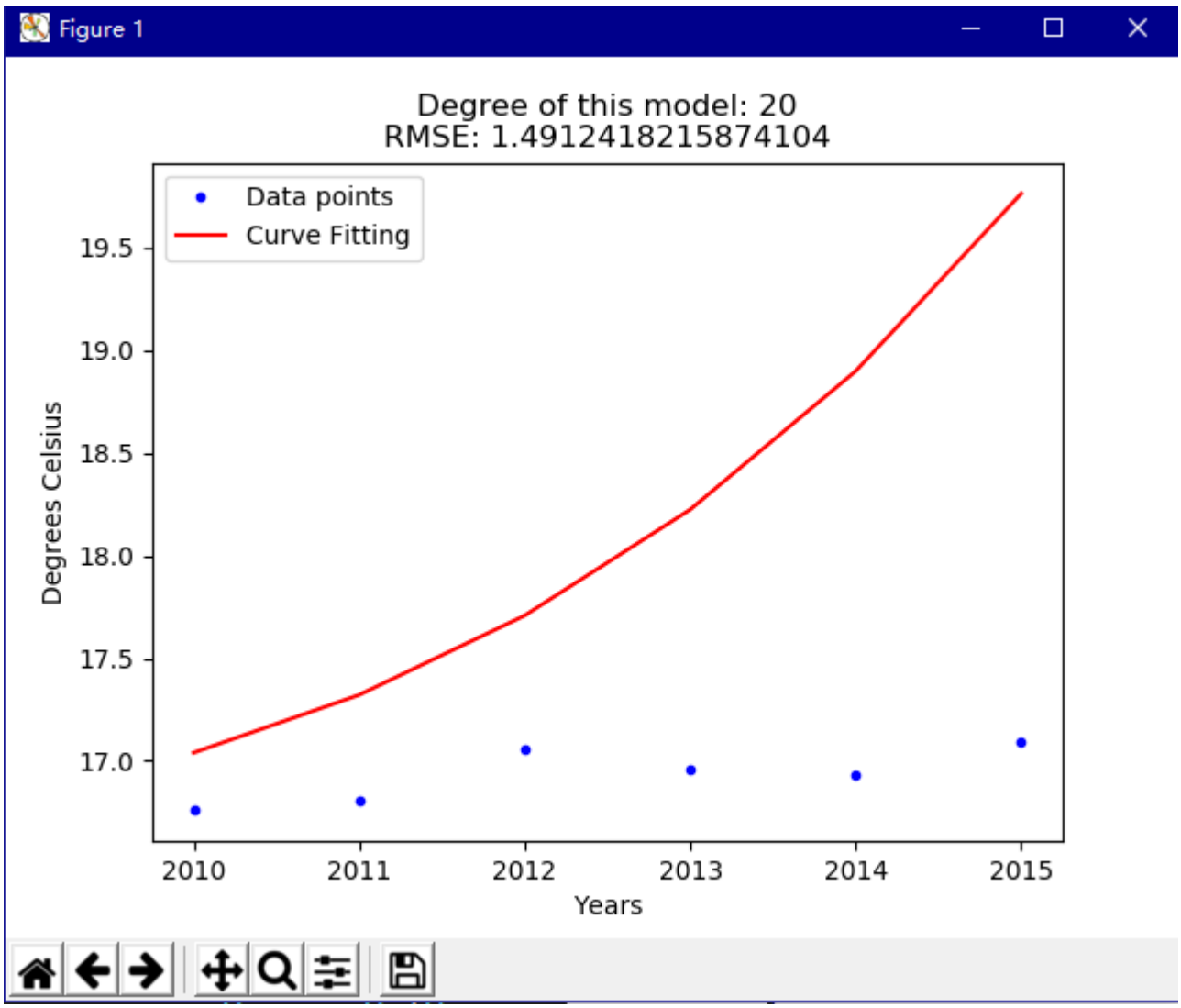


Figure 1
Degree of this model: 2
RMSE: 0.21177518245215685
Data points
Curve Fitting
17.10
17.05
17.00
16.95
16.90
16.85
16.80
16.75
Degrees Celsius
2010
2011
2012
2013
2014
2015
Years



- How did the different models perform? How did their RMSEs compare?

  阶数为1的模型计算出的RMSE的值最小。

- Which model performed the best? Which model performed the worst? Are they the same as those in part D.2.I? Why?

  阶数为1的模型表现得最好，阶数为20的模型表现得最差，这与D2.Ⅰ中的结果不同，因为数据太少。

- If we had generated the models using the A.4.II data (i.e. average annual temperature of New York City) instead of the 5-year moving average over 22 cities, how would the prediction results 2010-2015 have changed?

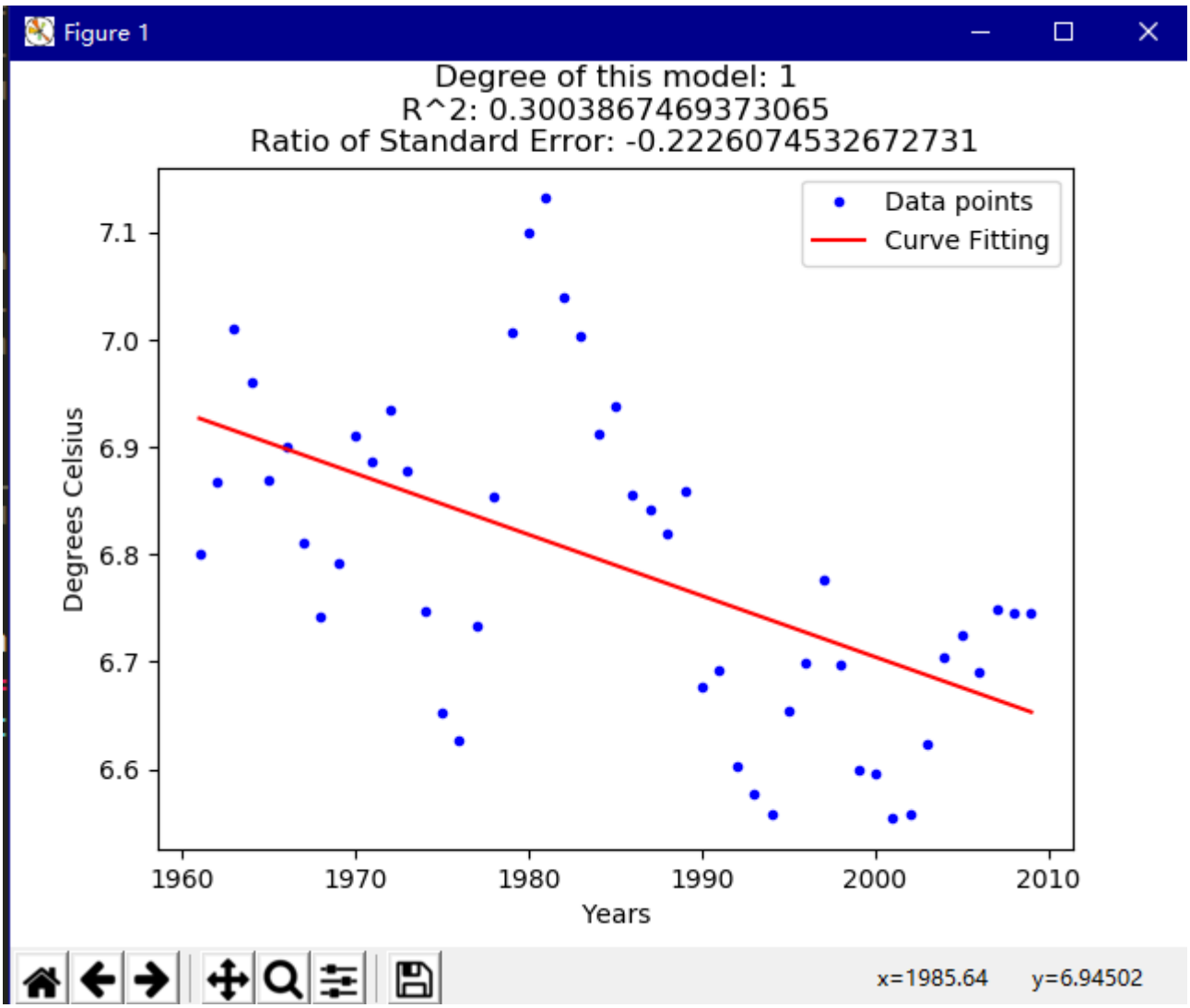
  预测结果的拟合情况应该会更差。

Part E

用标准差来衡量温度变化幅度大小，实现函数`gen_std_devs`来计算标准差。

```python
def gen_std_devs(climate, multi_cities, years):
    temperature_data = []
    for year in years:
        temperature_avg = []  # 存储各个城市每一天的平均气温
        for i in range(len(multi_cities)):
            if i == 0:  # 如果是第一个城市，就将本年的气温数据赋值给temperature_avg
                temperature_avg = climate.get_yearly_temp(multi_cities[i], year)
            else:  # 如果是之后的城市，将气温数据相加
                temperature_avg += climate.get_yearly_temp(multi_cities[i], year)
        temperature_avg /= len(multi_cities)  # 求出各个城市每一天的平均气温
        temperature_data.append(temperature_avg.std())  # 再将各个城市一年中每天的平均气温求标准差
    return temperature_data
```

根据21座城市每年气温的标准差来拟合曲线。



- Does the result match our claim (i.e., temperature variation is getting larger over these years)?

  结果与我们的判断不一致，从图表可以看出来，温度变化在逐年降低。

- Can you think of ways to improve our analysis?

  可以增加模型的阶数，或者增加moving_average函数中window_length的值。