Report Lab 6

Nguyen Tien Duc - ITITIU18029

December 20, 2019

1/ Newton's interpolating polynomial

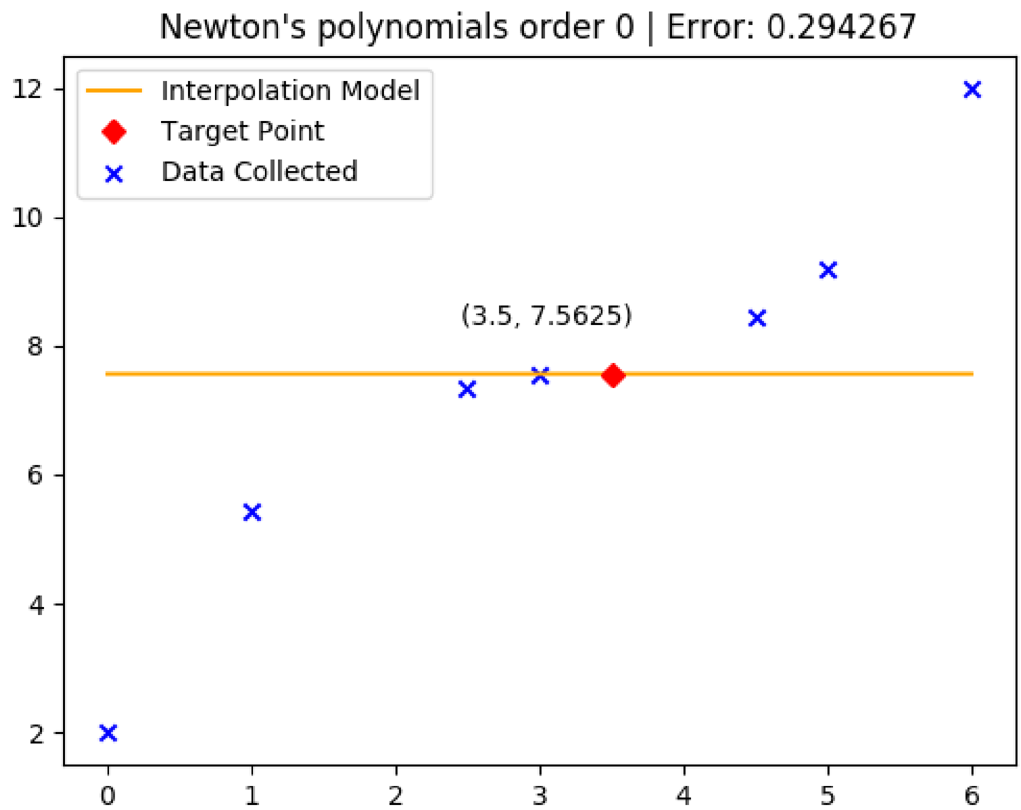
Code

```
from matplotlib import pyplot as plt
from numpy import arange


class DataPoint:
  def __init__(self, x, y):
    self.x = x
    self.y = y


def diff(data):
  if len(data) == 1:
    return data[0].y
  elif len(data) == 2:
    numerator = data[1].y - data[0].y
    denominator = data[1].x - data[0].x
    return numerator / denominator
  else:
    numerator = diff(data[1:]) - diff(data[:-1])
    denominator = data[-1].x - data[0].x
    return numerator / denominator


def newton_interpolation_coefficients(data):
  coefficient = []
  for i in range(1, len(data) + 1):
    coefficient.append(diff(data[:i]))
  return coefficient


def newton_interpolation_value(data, target):
  coefficients = newton_interpolation_coefficients(data)
  result = 0
  for i in range(len(data)):
    product = 1
    for j in range(i):
      product *= target - data[j].x
    result += coefficients[i] * product
  return result
```

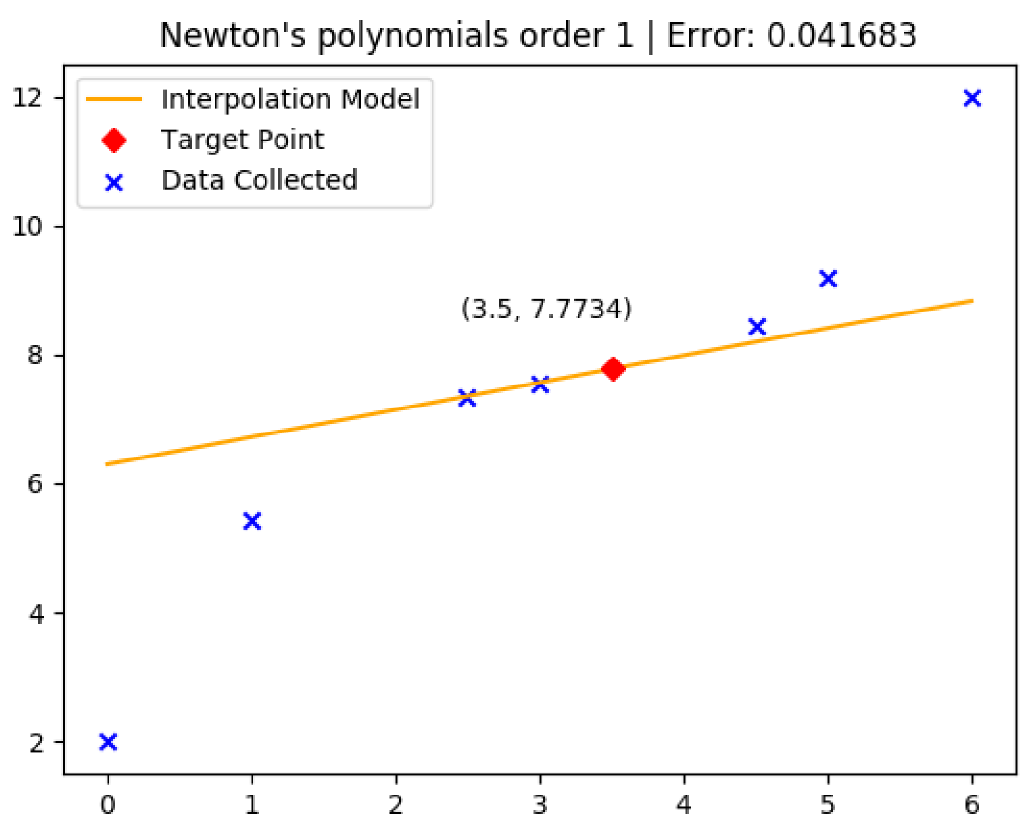
```
def newton_interpolation_error(data, target):
  product = 1
  for i in range(len(data) - 1):
    product *= target - data[i].x
  return abs(diff(data) * product)


def newton_interpolation_graph(best_data, data, data_graph_x, data_graph_y,
                                           value_graph_x, value_graph_y):
  x_min, x_max = data[0].x, data[-1].x
  for k in range(len(data)):
    if data[k].x > x_max:
      x_max = data[k].x
    if data[k].x < x_min:
      x_min = data[k].x
  for k in arange(x_min, x_max, 0.001):
    value_graph_x.append(k)
    value_graph_y.append(newton_interpolation_value(best_data, k))
  for k in range(len(data)):
    data_graph_x.append(data[k].x)
    data_graph_y.append(data[k].y)


def newton_interpolation_value_by_degree(data, target, order, data_graph_x,
                                           data_graph_y, value_graph_x,
                                           value_graph_y):
  final_data = data
  error = "N/A (Highest order)"
  if order < len(data) - 1:
    start = 0
    end = order + 2
    best_data = data[:end]
    while start + end <= len(data):
      tmp_data = data[start:start + end]
      if newton_interpolation_error(tmp_data, target) < newton_interpolation_error(
                                           best_data, target):
        best_data = tmp_data
      start += 1
    error = str(newton_interpolation_error(best_data, target).__round__(6))
    final_data = best_data[:-1]
  newton_interpolation_graph(final_data, data, data_graph_x, data_graph_y,
                                           value_graph_x, value_graph_y)
  return newton_interpolation_value(final_data, target), error


data_input = [
  DataPoint(0, 2),
  DataPoint(1, 5.4375),
  DataPoint(2.5, 7.3516),
  DataPoint(3, 7.5625),
  DataPoint(4.5, 8.4453),
  DataPoint(5, 9.1875),
  DataPoint(6, 12)
]

graph_values_y = []
graph_values_x = []
```

```
data_values_x = []
data_values_y = []
chosen_order = 6
chosen_target = 3.5
res = newton_interpolation_value_by_degree(
  data=data_input, target=chosen_target, order=chosen_order,
  data_graph_x=data_values_x, data_graph_y=data_values_y,
  value_graph_y=graph_values_y, value_graph_x=graph_values_x)

plt.title("Newton's polynomials order " + str(chosen_order) + " | Error: " + res[1])
plt.scatter(data_values_x, data_values_y, marker="x", c="blue", label="Data Collected
                                            ")
plt.plot(graph_values_x, graph_values_y, c="orange", label="Interpolation Model")
plt.plot(chosen_target, res[0], "D", c="red", label="Target Point")
plt.annotate('(' + str(chosen_target) + ', ' + str(res[0].__round__(6)) + ')', xy=(
                                            chosen_target * 0.7, res[0] * 1.1))
plt.legend()
plt.savefig('Order' + str(chosen_order) + '.png')
plt.show()
```

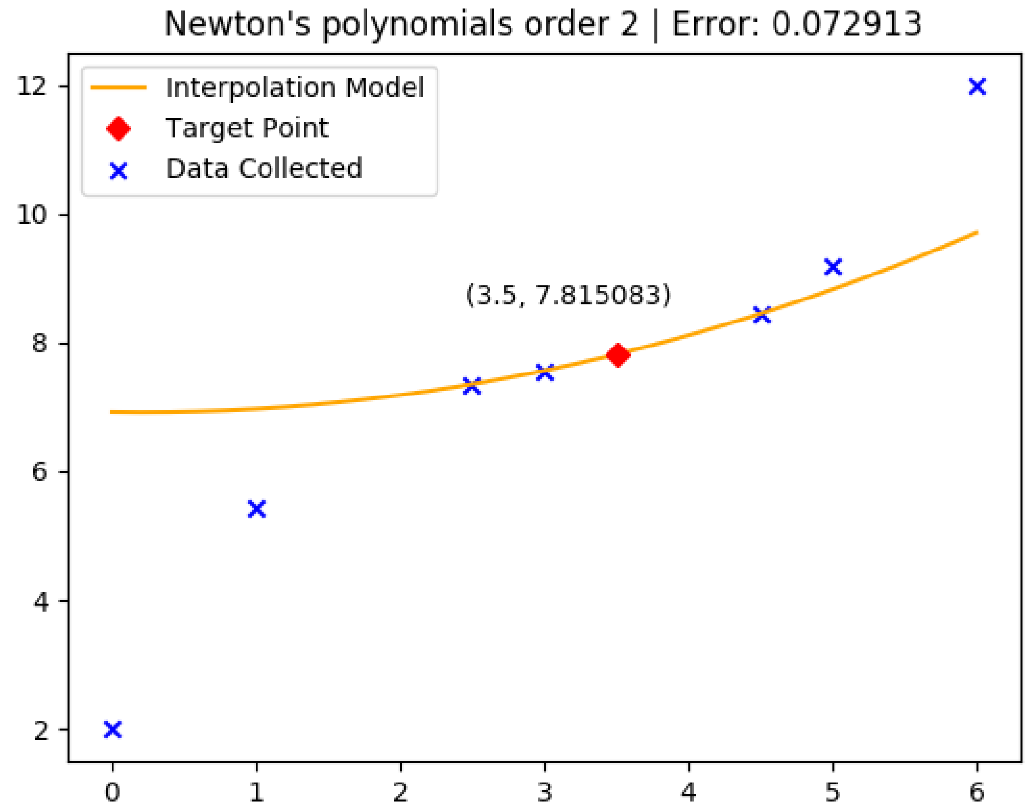
Running

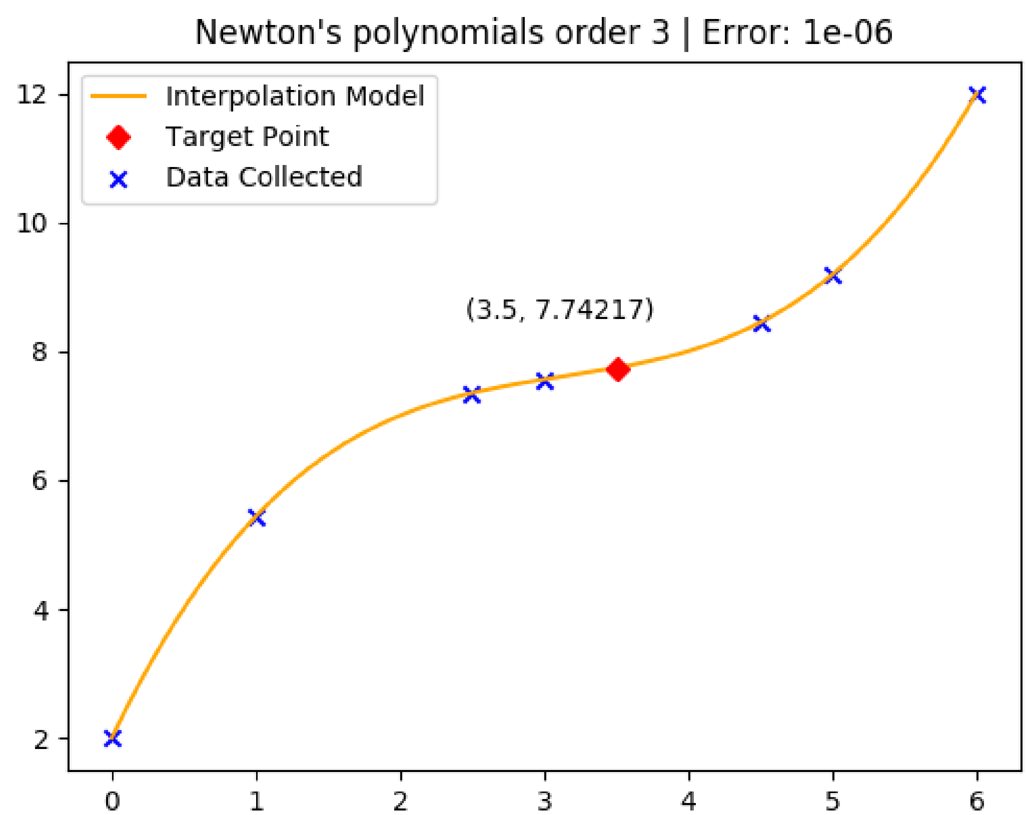


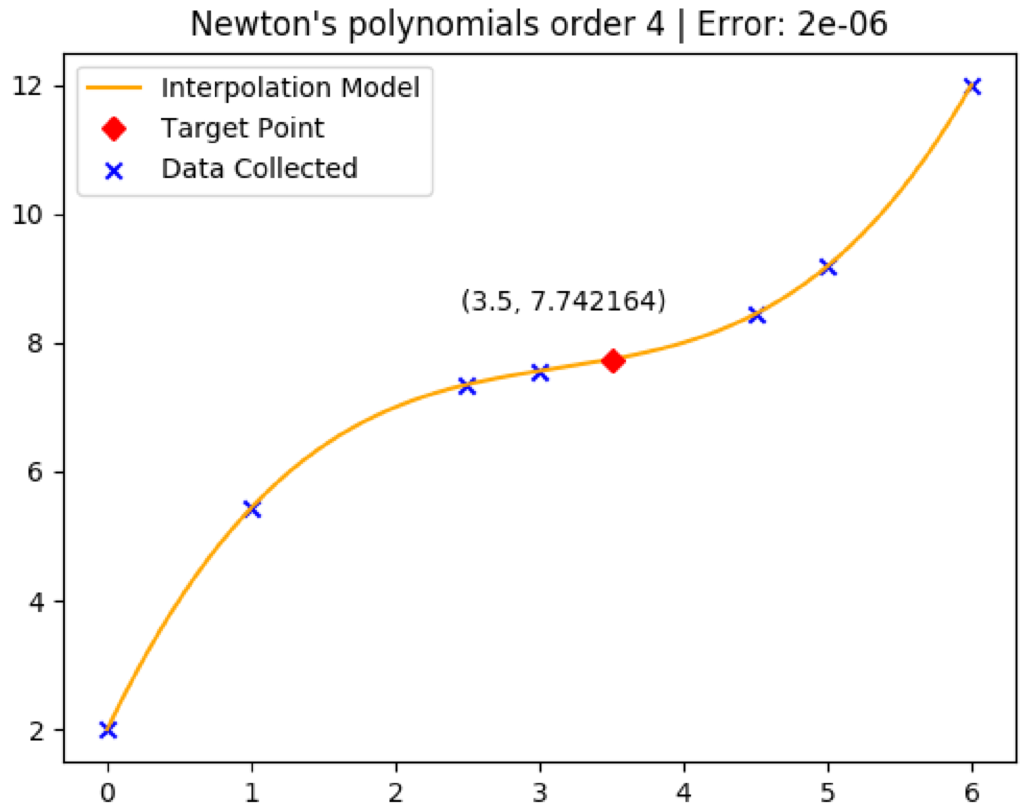
Newton's polynomials order 0 | Error: 0.294267
Interpolation Model
Target Point
Data Collected
(3.5, 7.5625)
12
10
8
6
4
2
0
1
2
3
4
5
6

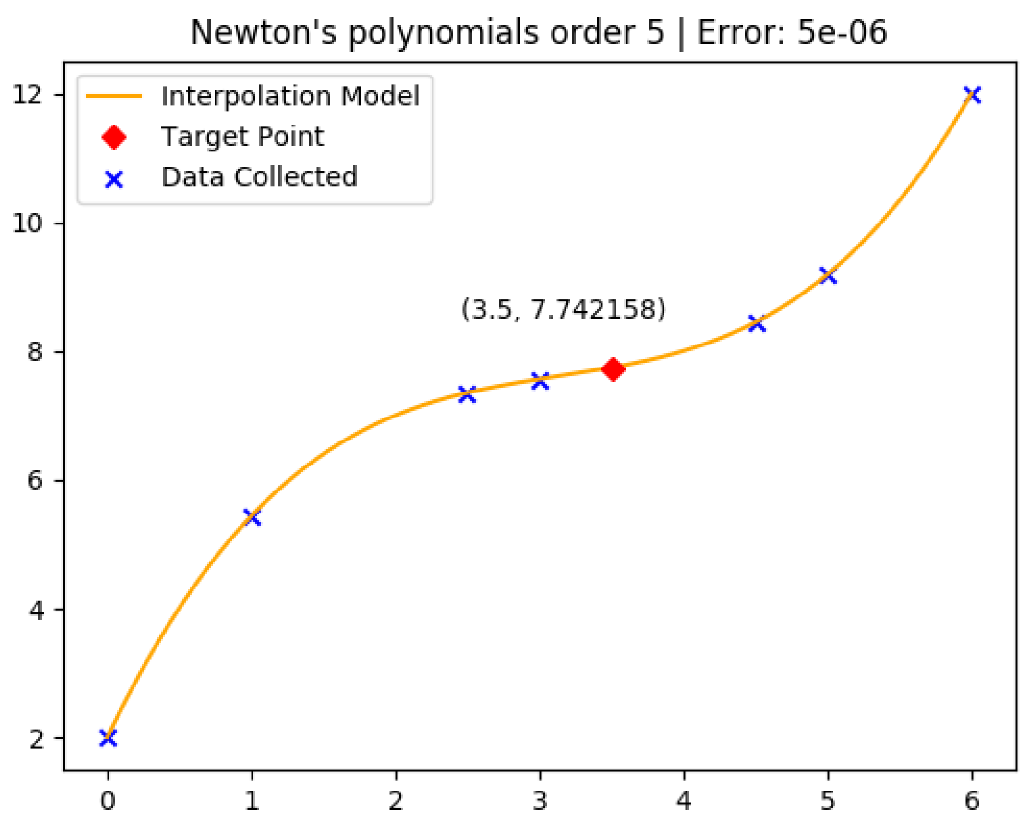


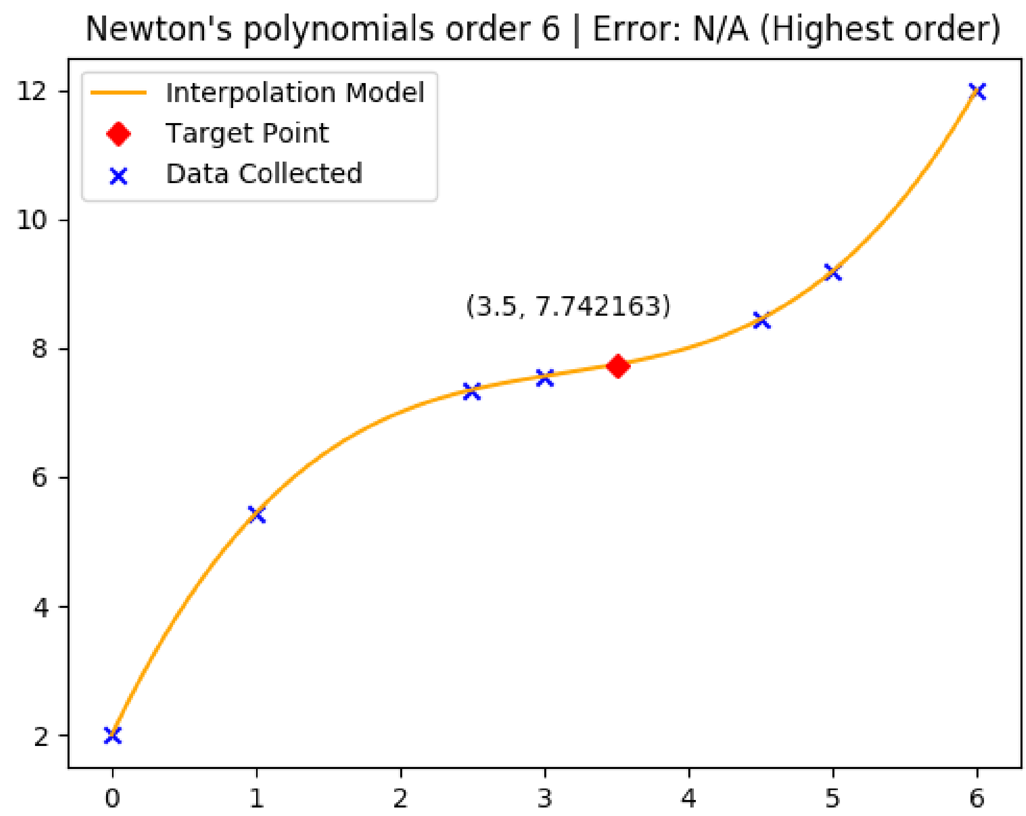
Newton's polynomials order 1 | Error: 0.041683
Interpolation Model
Target Point
Data Collected
(3.5, 7.7734)
12
10
8
6
4
2
0
1
2
3
4
5
6


Newton's polynomials order 2 | Error: 0.072913
Interpolation Model
Target Point
Data Collected
(3.5, 7.815083)
12
10
8
6
4
2
0
1
2
3
4
5
6


Newton's polynomials order 3 | Error: 1e-06
Interpolation Model
Target Point
Data Collected
(3.5, 7.74217)
12
10
8
6
4
2
0
1
2
3
4
5
6


Newton's polynomials order 4 | Error: 2e-06
Interpolation Model
Target Point
Data Collected
(3.5, 7.742164)
0
1
2
3
4
5
6
2
4
6
8
10
12


Newton's polynomials order 5 | Error: 5e-06
Interpolation Model
Target Point
Data Collected
(3.5, 7.742158)
0
1
2
3
4
5
6
2
4
6
8
10
12



2/ Conclusion

Through developing the algorithm to find interpolation model in various orders using Newton's interpolating polynomials, things that I learned are:

- The higher the order, the better fit the polynomials model, this is backed by the decrease in truncation error as the order increases.
- The lower, not maximum order can still give a fairly accurate estimation given that we choose the data sets with the least squared error. In this case, starting from 3^{th} order polynomials the truncation error has already been 10^{-6}, which is small enough to ignore and consider the model accurate.
- Newton's interpolating polynomials is harder to develop as an algorithm compared to Lagrange's interpolating polynomials.