Differential Equations Coding Project

Cam Walter

April 24, 2024

1 Questions

1. (a) $f(x) = -5 + 0.5x^2 + x^5$

   (b) The solution would not work, since we would need a degree 6 polynomial now. This makes sense, since if we added a row to multiply, there would not be enough columns in A to match up with x's rows.

   (c) $det(V) = 0 \iff \exists x_i, x_j(x_i = x_j, i \neq j)$

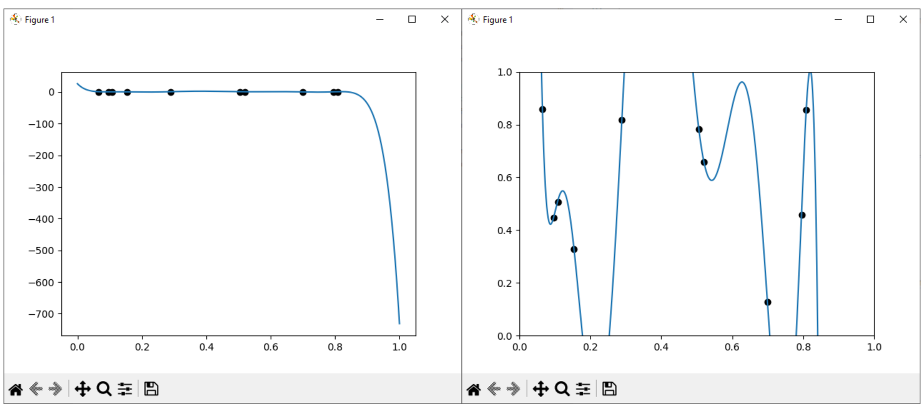

Figure 1: Zoomed out and in views of the data set

   (d) See Figure 1 and section 3.

   (e) We can use spline interpretation to form a piecewise function comprised ofseveral small polynomials to make the graph much better reflect the data points.

2. (a) $\lambda_1 = 5.652818837645601$

   (b) $\lambda_2 = 1.3215438145110312$

2 Code

```
import numpy as np
import numpy.matlib as ml
import numpy.linalg as linalg
import matplotlib.pyplot as plt
import matplotlib.colors as mcolors
import math
import numpy.typing as typing

def interpolate(x_in: list[float], y_in: list[float]) -> typing.NDArray[np.float64]:
  a: list[list[float]] = []
  for i in range(len(x_in)):
    a.append([])
    for j in range(len(x_in)):
      a[i].append(pow(x_in[i], j))

  a_inverse = linalg.inv(np.asmatrix(a, dtype=np.float64))

  output: typing.NDArray[np.float64] = np.matmul(
    a_inverse,
    ml.transpose(y_in)
  )

  return output.round(2)

def power_method(AtA: typing.NDArray[np.float64], initial_vec: list[float]) -> float:
  x = initial_vec
  eigenvalue = 0.
  for _ in range(20):
    y_i = np.matmul(AtA, x)

    y_transpose_y: float = np.matmul(ml.transpose(y_i), y_i)
    eigenvalue = math.sqrt(y_transpose_y)
    x = y_i / eigenvalue

  return eigenvalue

def main():
  # Question 1
  one_a = interpolate(
    x_in = [0., -1., 1., 2., 4., -2.],
    y_in = [-5., -5.5, -3.5, 29., 1027., -35.]
  )
  print(f"1a. {one_a}")
```

```python
# x_in = np.random.uniform(size=10)
# y_in = np.random.uniform(size=10)
# f = open("points.txt", "w")
# for i in range(10):
#   f.write(str(x_in[i]))
#   f.write(" ")
#   f.write(str(y_in[i]))
#   f.write("\n")
# f.close()

x_in: list[float] = []
y_in: list[float] = []

f = open("points.txt", "r")
lines = f.read().splitlines()
for line in lines:
  [x, y] = line.split(" ")
  x_in.append(float(x))
  y_in.append(float(y))
f.close()

one_d = interpolate(x_in, y_in)
print(f"1d. {one_d}")

def formula(x: typing.NDArray[np.float64]) -> typing.NDArray[np.float64]:
  y = 0
  for i in range(len(one_d[0])):
    y += one_d[0][i] * pow(x, i)
  return y

xs = np.linspace(0, 1, 1000, dtype=np.float64)
plt.plot(xs, formula(xs))

for i in range(len(x_in)):
  plt.scatter(x_in[i], y_in[i], color=mcolors.BASE_COLORS["k"])

# plt.axis([0, 1, 0, 1])
plt.show()

# Question 2
a: list[list[float]] = []
for i in range(10):
  a.append([x_in[i], y_in[i]])

AtA: typing.NDArray[np.float64] = np.matmul(ml.transpose(a), a)
eigen1 = power_method(AtA, [1., 1.])
```

```
  eigen2 = power_method(linalg.inv(AtA), [1., 1.])
  print(f"2a. eigenvalue={eigen1}")
  print(f"2b. eigenvalue={eigen2}")

main()
```

3 Points

0.6989859734239012 0.12752984504762666
0.5195413482072768 0.6578639788898996
0.7954693426170987 0.456846155415475
0.2890740668207832 0.8179469664047259
0.15382314665773744 0.3273088107653156
0.09682971271188445 0.4469241830463365
0.5054607786477722 0.7822612701272607
0.06480610406322695 0.8592441876541691
0.8091135851989284 0.8567062900354686
0.10807818056142182 0.5076911390833739