Math 128A: Programming Assignment 2

Michael Pham

Summer 2024

Problems

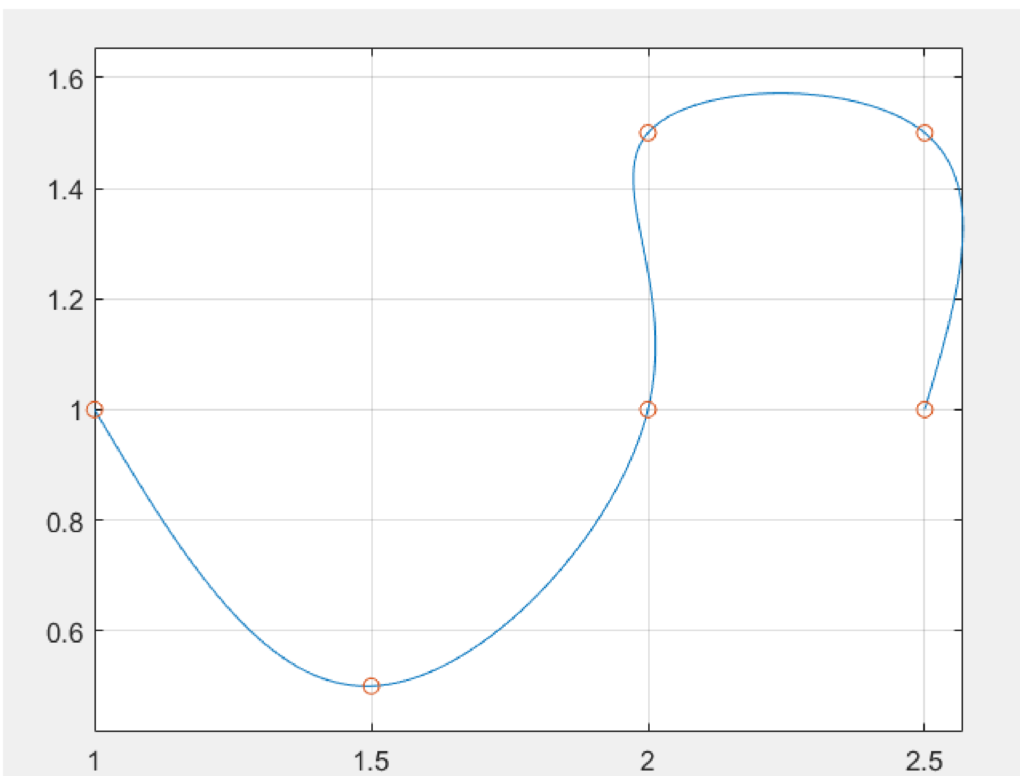
Problem 1.1. Approximate the curve by fitting natural cubic splines to the given data, independently for $x(t)$ and $y(t)$. Plot the curve in MATLAB by `plot(xx,yy, x,y,'o'), axis equal, grid on`, where `x,y` contain the given values and `xx,yy` contain the spline data evaluated for a large number of parameter values between 0 and 5.

Solution. Below, we include the commands used to construct the approximation of the curve using natural cubic splines:

Code: MATLAB Commands

```
>>> ax = [1, 1.5, 2, 2, 2.5, 2.5];
ay = [1, 0.5, 1, 1.5, 1.5, 1];
t = [0, 1, 2, 3, 4, 5];
[bx, cx, dx] = ncspline(t, ax);
[by, cy, dy] = ncspline(t, ay);
s = 0:0.01:5;
xx = splineeval(t, ax, bx, cx, dx, s);
yy = splineeval(t, ay, by, cy, dy, s);
plot(xx, yy, ax, ay, 'o'), axis equal, grid on
```

The following is the image of the plot created:



We also provide the values coefficients for both $x(t)$ and $y(t)$ in the provided order:

Code: Values for $x(t)$

```
>> ax

ax =

```

```
Columns 1 through 5

1.000000000000000   1.500000000000000   2.000000000000000   2.000000000000000   2.500000000000000

Column 6

2.500000000000000

>> bx

bx =

0.452153110047847   0.595693779904306   0.165071770334928   0.244019138755981   0.358851674641148

>> cx

cx =

0   0.143540669856459  -0.574162679425837   0.653110047846890  -0.538277511961722

>> dx

dx =

0.047846889952153  -0.239234449760766   0.409090909090909  -0.397129186602871   0.179425837320574
```

Code: Values for $y(t)$

```
>> ay

ay =

Columns 1 through 5

1.000000000000000   0.500000000000000   1.000000000000000   1.500000000000000   1.500000000000000

Column 6

1.000000000000000

>> by

by =

-0.760765550239234   0.021531100478469   0.674641148325359   0.279904306220096  -0.294258373205742

>> cy

cy =

0   0.782296650717703  -0.129186602870813  -0.265550239234450  -0.308612440191388

>> dy

dy =

0.260765550239234  -0.303827751196172  -0.045454545454545  -0.014354066985646   0.102870813397129
```

■

Problem 1.2. Use Newton's method to find the parameter values t_1 and t_2 where the curve intersects the line $y = 1.2$. Use the provided MATLAB functions `splineeval.m` and `diffsplineeval.m`. Give a short justification for your choice of initial guess for each of the two intersections.

Solution. First, we provide the MATLAB commands below:

Code: MATLAB Commands

```
>> f = @(p0) splineeval(t, ay, by, cy, dy, p0) - 1.2;
df = @(p0) diffsplineeval(t, ay, by, cy, dy, p0);
t1 = newton(f, df, 2.5, 0.000000001);
t2 = newton(f, df, 4.5, 0.000000001);

>> fprintf("%.8f", t1)
2.31798217

>> fprintf("%.8f", t2)
4.66164416
```

We chose the initial points $p_0 = 2.5$ and $q_0 = 4.5$. Looking at the data, we see that $y(t)$ goes from 1.0 to 1.5 between $t = 2$ and $t = 3$. So, we used the average of these endpoints.

Similarly, we used $q_0 = 4.5$ since at $t = 4$, $y(t) = 1.5$, and at $t = 5$, we have $y(t) = 1.0$. So, since the curve is continuous, we know that at some point between those two times, $y(t) = 1.2$ by the Mean Value Theorem. ■

Problem 1.3. Compute the length of the segment of the curve above $y = 1.2$, by numerically evaluating the integral

$$L = \int_{t_1}^{t_2} \sqrt{x'(t)^2 + y'(t)^2}\mathrm{dt}$$

using the composite trapezoidal rule. Compute a series of approximations L_{16}, L_{32}, L_{64}, and L_{128} using $n = 16, 32, 64, 128$, respectively. Also compute a highly accurate value L using $n = 10,000$.

Plot the errors $|L_n - L|$ versus $h = (t_2 - t_1)/n$ in a log-log plot and estimate its slope.

Solution. To begin with, we provide the code for the function `comp_trap` below:

Code: Code for `comp_trap` function

```
function val = comp_trap(a, b, n, f)

h = (b - a)/n;
i1 = a + h;
in = a + (n - 1)*h;
sum = 0;
for i = i1:h:in
sum = sum + f(i);
end
sum = 2*sum;
sum = sum + f(a) + f(b);
val = (h/2)*sum;
end
```

Next, we provide the results:

Code: Results for L_n

```
>> l16 = comp_trap(t1, t2, 16, g)
l32 = comp_trap(t1, t2, 32, g)
l64 = comp_trap(t1, t2, 64, g)
l128 = comp_trap(t1, t2, 128, g)

l16 =

```

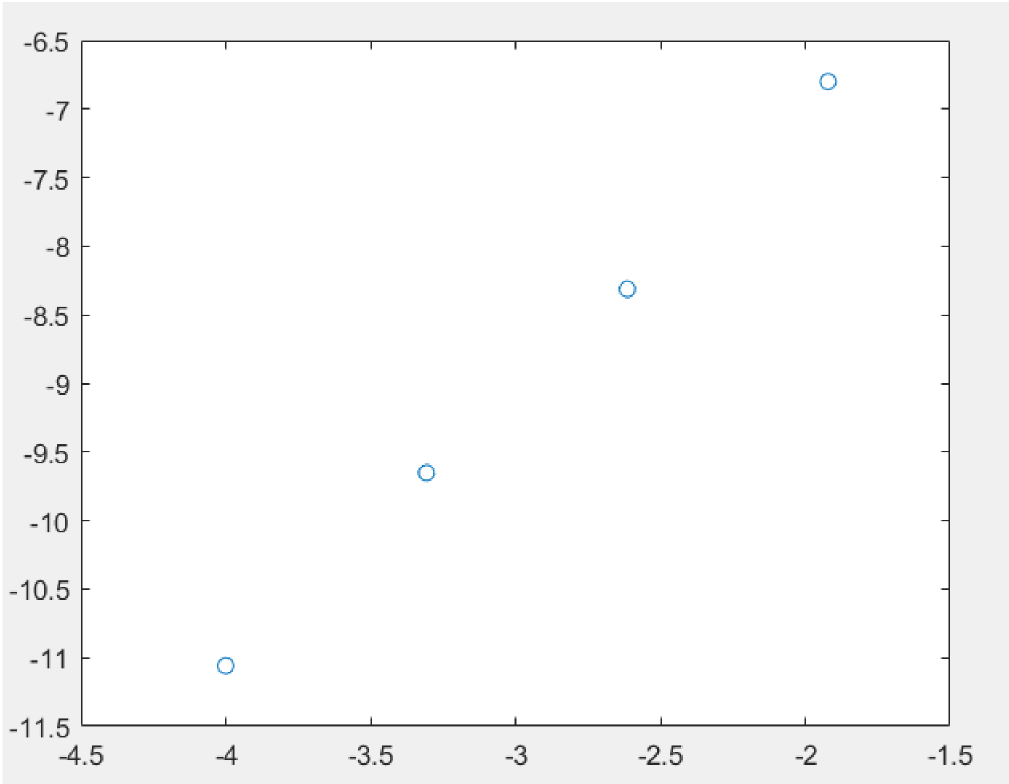
```
1.162654862462451


l32 =

1.161785809250850


l64 =

1.161604753231803


l128 =

1.161556258136053

>> l10000 = comp_trap(t1, t2, 10000, g)

l10000 =

1.161540504195514
```

Finally, we include the log-log plot below, generated with the following code:

Code: Plotting errors

```
>> yl16 = log(abs(l16 - l10000));
yl32 = log(abs(l32 - l10000));
yl64 = log(abs(l64 - l10000));
yl128 = log(abs(l128 - l10000));
plot([log((t2-t1)/16), log((t2-t1)/32), log((t2-t1)/64), log((t2-t1)/128)], [yl16, yl32, yl64, yl128], 'o')
```

And the following graph is generated, with the h values in the x-axis and the error values in the y-axis:



Using the middle two points, we then calculate the following slope:

$$\frac{-8.31301-(-9.65274)}{-2.61402-(-3.30717)} \approx 1.933$$

So, we see that the slope is around 2. ■