THE COLLEGE OF WILLIAM AND MARY

# Programming Assignment #3

Alexander Powell

November 10, 2014

## 1

The natural cubic spline of the parametric curve was found using the following MATLAB commands. The functions *ncspline.m* and *splineeval.m* were used.

```
>> t = [0,1,2,3,4,5];
>> x = [1,1.5,2,2,2.5,2.5];
>> y = [1,0.5,1,1.5,1.5,1];
>> temp = 0:0.01:5;
>> [b1,c1,d1] = ncspline(t,x);
>> xx = splineeval(t,x,b1,c1,d1,temp);
>> [b2,c2,d2] = ncspline(t,y);
>> yy = splineeval(t,y,b2,c2,d2,temp);
>> figure
>> plot(xx,yy,x,y,'o'), axis equal, grid on
>> axis([.9,2.7,.4,1.65])
```

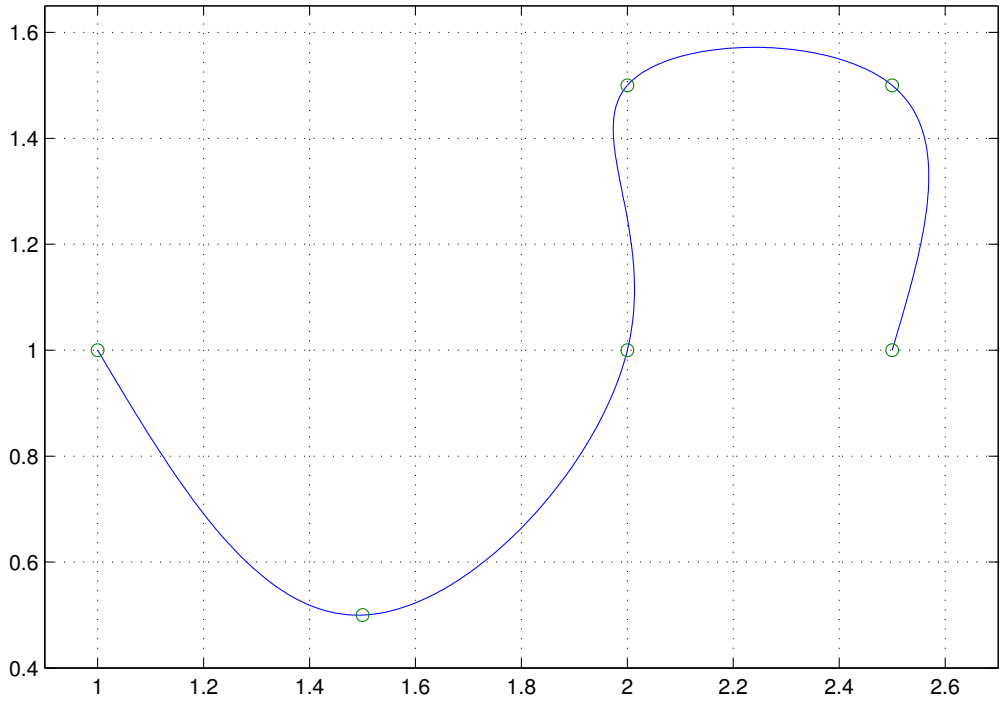
The following are tables of the natural cubic spline coeffients $a$, $b$, $c$, and $d$ for $t, x(t)$ and $t, y(t)$. .

$a_j, b_j, c_j$ and $d_j$ values for $x(t)$

| i | $a_i$ | $b_i$ | $c_i$ | $d_i$ |
|---|---|---|---|---|
| 0 | 1 | 0.45215311 | 0 | 0.04784689 |
| 1 | 1.5 | 0.59569378 | 0.14354067 | -0.23923445 |
| 2 | 2 | 0.16507177 | -0.57416268 | 0.40909090 |
| 3 | 2 | 0.24401914 | 0.65311005 | -0.39712918 |
| 4 | 2.5 | 0.35885167 | -0.53827751 | 0.17942583 |

$a_j, b_j, c_j$ and $d_j$ values for $y(t)$

| i | $a_i$ | $b_i$ | $c_i$ | $d_i$ |
|---|---|---|---|---|
| 0 | 1 | -0.76076555 | 0 | 0.26076555 |
| 1 | 0.5 | 0.02153110 | 0.78229665 | -0.30382775 |
| 2 | 1 | 0.67464114 | -0.12918660 | -0.04545454 |
| 3 | 1.5 | 0.27990430 | -0.26555023 | -0.01435406 |
| 4 | 1.5 | -0.29425837 | -0.30861244 | 0.10287081 |



## 2

Newton's method was used to estimate the parameter values $t_1$ and $t_2$ where the curve intersects the line $y = 1.2$. The following MATLAB code was used to make the calculations:

```
t = [0,1,2,3,4,5];
x = [1,1.5,2,2,2.5,2.5];
y = [1,0.5,1,1.5,1.5,1];
temp = 0:0.01:5;
[b1,c1,d1] = ncspline(t,x);
[b2,c2,d2] = ncspline(t,y);
CountMax = 100;
guess = 2; % 2 was the guess for the left, 5 for the right
x_p = guess;
tol = 1e-8;
for i=1:CountMax
        guess = guess - (splineeval(t,y,b2,c2,d2,guess)-1.2) /
    diffsplineeval(t,y,b2,c2,d2,guess);
    % the above two lines are actually one line but
    % are separated for readability.

        if (splineeval(t,y,b2,c2,d2,guess) < tol & abs(guess-x_p) < tol)
            fprintf( 'x%d = %.15f\n', i, guess );
            disp(['It took ' num2str(i) ' steps to converge.' ])
            return;
        else
            fprintf( 'x%d = %.15f\n', i, guess );
            x_p = guess;
        end
end
disp('Reach the maximum number of iterations!');
```

The above code generated the following values:

$$t_1 = 2.31798217, t_2 = 4.66164416$$

## 3

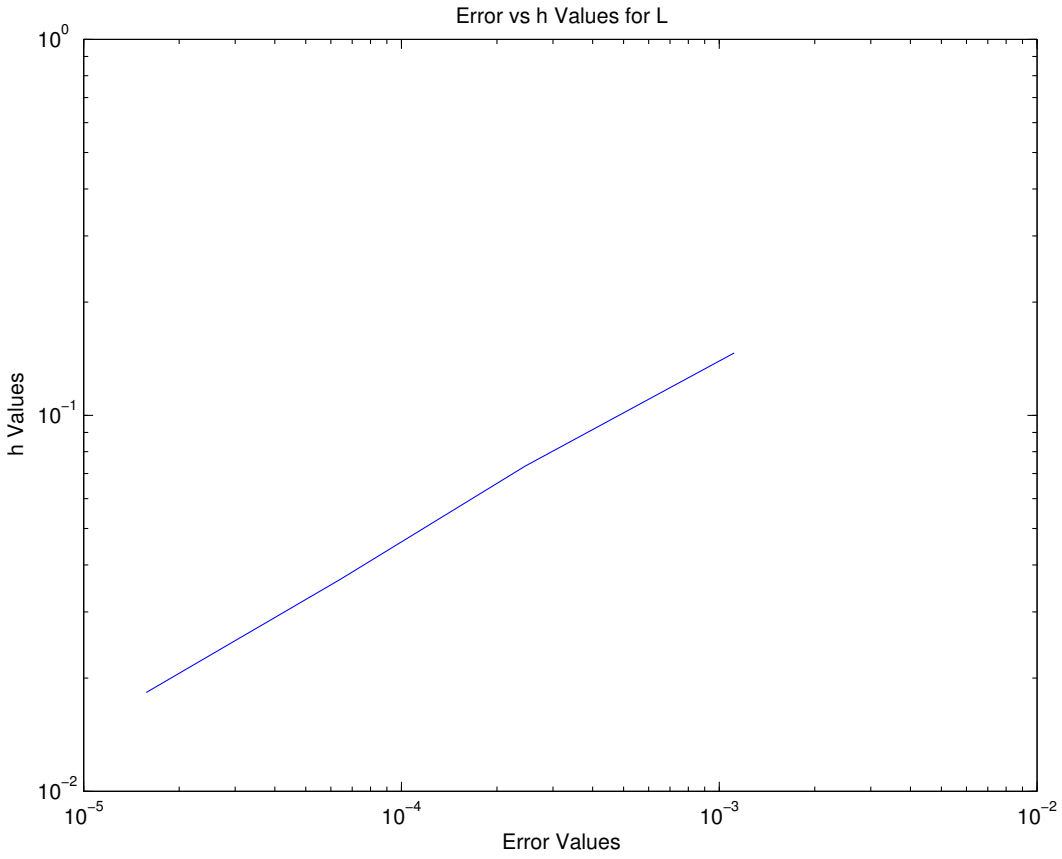
The following code computes the length of the curve by using the composite trapezoid rule with increasingly greater values of $n$.

```
ns = [16, 32, 64, 128, 10000];
t = [0,1,2,3,4,5];
x1 = [1,1.5,2,2,2.5,2.5];
y1 = [1,0.5,1,1.5,1.5,1];
[b1,c1,d1] = ncspline(t,x1);
[b2,c2,d2] = ncspline(t,y1);

f = @(x) sqrt(((diffsplineeval(t,x1,b1,c1,d1,x)).^2
+ ((diffsplineeval(t,y1,b2,c2,d2,x)).^2));
% ^these are one line in MATLAB but were separated for clarity

a = 2.31798217; b = 4.66164416;
for n = ns
    x = linspace(a, b, n+1);
    h = (b-a)/n;
    int = (h/2)*(2*sum(f(x)) -f(a) - f(b))
end
```

The following is a table displaying the different approximations:

| | |
|---|---|
| $L_{16}$ | 1.16265486 |
| $L_{32}$ | 1.16178581 |
| $L_{64}$ | 1.16160475 |
| $L_{128}$ | 1.16155626 |
| $L_{10000}$ | 1.16154051 |

The following is a log-log plot of the errors $|L_n - L|$ versus $h = (t_1 - t_2)/n$.




Applying the log-log plot changes the figure almost to a line. Using the ordered pairs $(10^{-3}, 10^{-0.9})$ and $(10^{-4}, 10^{-1.5})$ gives us:

$$\frac{10^{-1.5} - 10^{-0.9}}{10^{-4} - 10^{-3}} \approx 100$$

So, the slope of the log-log plot is approximately 100.