Homework 3

Ex.3.1

x	2.0	4.5	5.25	7.81	9.2	10.6
y	7.2	7.1	6.0	5.0	3.5	5.0

For fifth polynomial interpolation we choose the value of y given by

$$y(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + a_4 x^4 + a_5 x^5$$

Using the six points

$x_0 = 2.00$ $\quad y(x_0) = 7.2$
$x_1 = 4.25$ $\quad y(x_1) = 7.1$
$x_2 = 5.25$ $\quad y(x_2) = 6.0$
$x_3 = 7.81$ $\quad y(x_3) = 5.0$
$x_4 = 9.20$ $\quad y(x_4) = 3.5$
$x_5 = 10.60$ $\quad y(x_5) = 5.0$

gives:

$$y(2.00) = a_0 + a_1(2.00) + a_2(2.00)^2 + a_3(2.00)^3 + a_4(2.00)^4 + a_5(2.00)^5 = 7.2$$
$$y(4.25) = a_0 + a_1(4.25) + a_2(4.25)^2 + a_3(4.25)^3 + a_4(4.25)^4 + a_5(4.25)^5 = 7.1$$
$$y(5.25) = a_0 + a_1(5.25) + a_2(5.25)^2 + a_3(5.25)^3 + a_4(5.25)^4 + a_5(5.25)^5 = 6.0$$
$$y(7.81) = a_0 + a_1(7.81) + a_2(7.81)^2 + a_3(7.81)^3 + a_4(7.81)^4 + a_5(7.81)^5 = 5.0$$
$$y(9.20) = a_0 + a_1(9.20) + a_2(9.20)^2 + a_3(9.20)^2 + a_4(9.20)^4 + a_5(9.20)^5 = 3.5$$
$$y(10.60) = a_0 + a_1(10.60) + a_2(10.60)^2 + a_3(10.60)^3 + a_4(10.60)^4 + a_5(10.60)^5 = 5.0$$

Writing the six equations in matrix form:

$$\begin{bmatrix} 1 & 2.00 & 4 & 8 & 16 & 32 \\ 1 & 4.25 & 18.063 & 76.766 & 326.25 & 1386.6 \\ 1 & 5.25 & 27.563 & 144.70 & 769.69 & 3988.4 \\ 1 & 7.81 & 60.996 & 476.38 & 3720.5 & 29057 \\ 1 & 9.20 & 84.64 & 778.69 & 7163.9 & 65908 \\ 1 & 10.60 & 112.36 & 1191.0 & 12625 & 133820 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{bmatrix} = \begin{bmatrix} 7.2 \\ 7.1 \\ 6.0 \\ 5.0 \\ 3.5 \\ 5.0 \end{bmatrix}$$

Solving the above six equations gives

$a_0 = -30.898$
$a_1 = 41.344$
$a_2 = -15.855$
$a_3 = 2.7862$
$a_4 = -0.23091$
$a_5 = 0.0072923$

Hence

$$y(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + a_4 x^4 + a_5 x^5$$
$$= -30.898 + 41.344x - 15.855x^2 + 2.7862x^3 - 0.23091x^4 +$$
$$+ 0.0072923x^5, \qquad 2 \le x \le 16$$

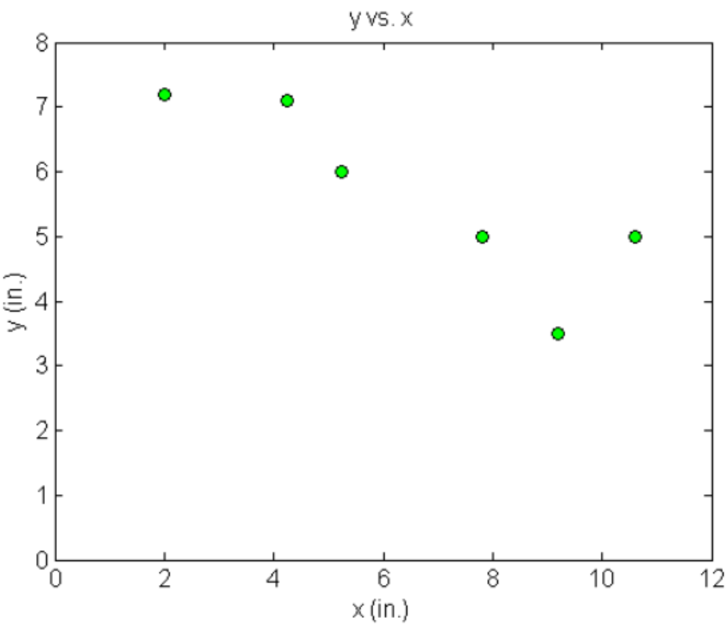


Figure 1 Location of holes on the rectangular plate.

Cubic Spline formula is

$$f(x) = \frac{(x_i - x)^3}{6h} M_{i-1} + \frac{(x - x_{i-1})^3}{6h} M_i + \frac{(x_i - x)}{h}\left(y_{i-1} - \frac{h^2}{6} M_{i-1}\right) + \frac{(x - x_{i-1})}{h} \cdot \left(y_i - \frac{h^2}{6} M_i\right)$$

We have $M_{i-1} + 4M_i + M_{i+1} = \frac{6}{h^2}(y_{i-1} - 2y - y_{i+1})$

Here $h = 2.5$ $n = 5$ $M_0 = 0$ $M_5 = 0$

Substitute $i = 1$ in equation

$$M_0 + 4M_1 + M_2 = \frac{6}{h^2}(y_0 - 2y_1 + y_2) \Rightarrow 4M_1 + M_2 = -0.96$$

$$M_1 + 4M_2 + M_3 = \frac{6}{h^2}(y_1 - 2y_2 + y_3) \Rightarrow M_1 + 4M_2 + M_3 = 0.096$$

$$M_2 + 4M_3 + M_4 = \frac{6}{h^2}(y_2 - 2y_3 + y_4) \Rightarrow M_2 + 4M_3 + M_4 = -0.48$$

$$M_3 + 4M_4 + M_5 = \frac{6}{h^2}(y_3 - 2y_4 + y_5) \Rightarrow M_3 + 4M_4 = 2.88$$

Substitute $i = 1$ in eq, we get cubic spline 1st interval [2.4 - 5]

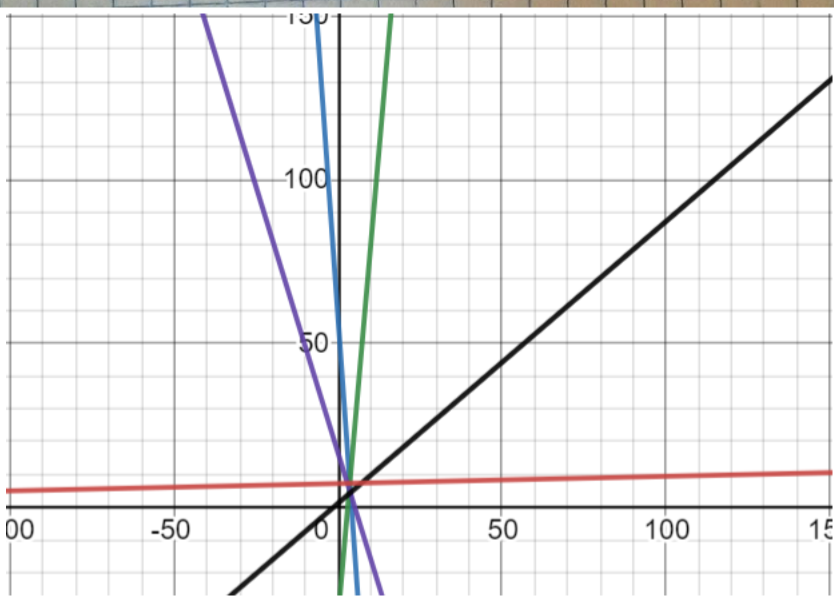
$f_1(x) = -0.0191x^3 + 0.1148x^2 - 0.1501x + 7.1939$ for $2.0 \le x \le 4.5$

$f_2(x) = 0.0317x^3 - 0.4709x^2 + 1.7072x + 1.776$ for $4.5 \le x \le 5.25$

$f_3(x) = -0.0372x^3 + 0.6828x^2 - 4.5051x + 16.3729$ for $5.25 \le x \le 7.81$

$f_4(x) = 0.0789x^3 - 1.9504x^2 + 15.087x - 33.4957$ for $7.81 \le x \le 9.2$

$f_5(x) = -0.0542x^3 + 1.7227x^2 - 17.3222x + 57.3723$ for $9.2 \le x \le 10.6$



Ex.3.2

a)

n	1	2	3	4	5
$\Gamma(n)$	1	1	2	6	24

```
import numpy as np
import matplotlib.pyplot as plt

plt.style.use('seaborn-poster')

%matplotlib inline
def divided_diff(x, y):
    '''
    function to calculate the divided
    differences table
    '''
    n = len(y)
    coef = np.zeros([n, n])
    # the first column is y
    coef[:,0] = y

    for j in range(1,n):
        for i in range(n-j):
            coef[i][j] = \
           (coef[i+1][j-1] - coef[i][j-1]) / (x[i+j]-x[i])

    return coef

def newton_poly(coef, x_data, x):
    '''
    evaluate the newton polynomial
    at x
    '''
    n = len(x_data) - 1
    p = coef[n]
    for k in range(1,n+1):
        p = coef[n-k] + (x -x_data[n-k])*p
    return p
x = np.array([1,2,3,4,5])
y = np.array([1,1,2,6,24])
# get the divided difference coef
a_s = divided_diff(x, y)[0, :]

# evaluate on new data points
#x_new = np.arange(1, 2,3,4,5)
#Y_new = newton_poly(a_s, x, x_new)

plt.figure(figsize = (12, 8))
plt.plot(x, y, 'bo')
#plt.plot(x_new, y_new)
```

$$
\begin{array}{cccccc}
x_0 & y_0 & & & & \\
 & & f[x_1,x_0] & & & \\
x_1 & y_1 & & f[x_2,x_1,x_0] & & \\
 & & f[x_2,x_1] & & f[x_3,x_2,x_1,x_0] & \\
x_2 & y_2 & & f[x_3,x_2,x_1] & & f[x_4,x_3,x_2,x_1,x_0] \\
 & & f[x_3,x_2] & & f[x_4,x_3,x_2,x_1] & \\
x_3 & y_3 & & f[x_4,x_3,x_2] & & \\
 & & f[x_4,x_3] & & & \\
x_4 & y_4 & & & &
\end{array}
$$

x	y	1^{st} order	2^{nd} order	3^{rd} order	4^{th} order
1	1				
		0			
2	1		0.5		
		1		0.333333	
3	2		1.5		-0.458333
		4		-1.5	
4	6		-3		
		-2			
5	4				

Newton's divided difference interpolation formula

$$f(x) = y_0 + \left(x - x_0\right)f\left[x_0, x_1\right] + \left(x - x_0\right)\left(x - x_1\right)f\left[x_0, x_1, x_2\right] + \left(x - x_0\right)\left(x - x_1\right)\left(x - x_2\right)f\left[x_0, x_1, x_2, x_3\right] + \left(x - x_0\right)\left(x - x_1\right)\left(x - x_2\right)\left(x - x_3\right)f\left[x_0, x_1, x_2, x_3, x_4\right]$$

$$f(x) = 1 + (x - 1) \times 0 + (x - 1)(x - 2) \times 0.5 + (x - 1)(x - 2)(x - 3) \times 0.333333 + (x - 1)(x - 2)(x - 3)(x - 4) \times - 0.458333$$

$$f(x) = 1 + (x - 1) \times 0 + \left(x^2 - 3x + 2\right) \times 0.5 + \left(x^3 - 6x^2 + 11x - 6\right) \times 0.333333 + \left(x^4 - 10x^3 + 35x^2 - 50x + 24\right) \times - 0.458333$$

$$f(x) = 1 + (0) + \left(0.5x^2 - 1.5x + 1\right) + \left(0.333333x^3 - 2x^2 + 3.666667x - 2\right) + \left(- 0.458333x^4 + 4.583333x^3 - 16.041667x^2 + 22.916667x - 11\right)$$

$$f(x) = - 0.458333x^4 + 4.916667x^3 - 17.541667x^2 + 25.083333x - 11$$

$$f(x) = \frac{(x_i - x)^3}{6h} M_{i-1} + \frac{(x - x_{i-1})^3}{6h} M_i + \frac{(x_i - x)}{h}\left(y_{i-1} - \frac{h^2}{6} M_{i-1}\right) + \frac{(x - x_{i-1})}{h}\left(y_i - \frac{h^2}{6} M_i\right)$$

b)

We have $M_{i-1} + 4M_i + M_{i+1} = \frac{6}{h^2}(y_{i-1} - 2y_i + y_{i+1})$

$h=1$ $\quad h=4 \quad M_0=0 \quad M_4=0$

$M_0 + 4M_1 + M_2 = \frac{6}{h^2}(y_0 - 2y_1 + y_2). \Rightarrow 4M_1 + M_2 = 6$

$M_1 + 4M_2 + M_3 = \frac{6}{h^2}(y_1 - 2y_2 + y_3) \Rightarrow M_1 + 4M_2 + M_3 = 18$

$M_2 + 4M_3 + M_4 = \frac{6}{h^2}(y_2 - 2y_3 + y_4) \Rightarrow M_2 + 4M_3 = 84$

$\cancel{M_3}$

$f_1(x) = 0.3036x^3 - 0.9107x^2 + 0.6071x + 1 \quad \text{for } 1 \le x \le 2$

$f_2(x) = -0.5178x^3 + 4.0178x^2 - 9.2498x + 7.5713 \quad \text{for } 2 \le x \le 3$

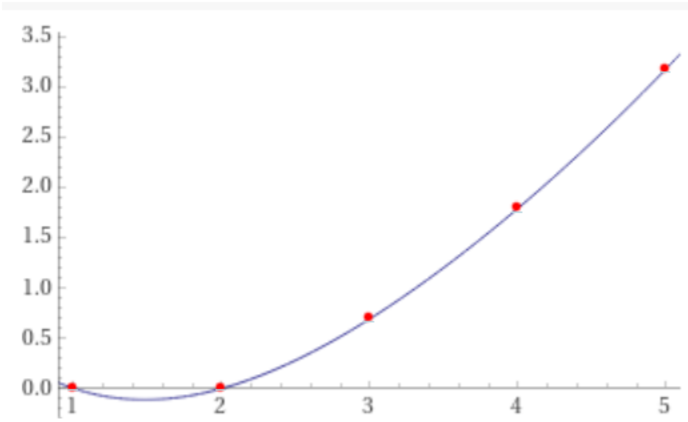
$f_3(x) = 3.7678x^3 - 34.5535x^2 + 106.4641x - 108.1426 \quad \text{for } 3 \le x \le 4$

$f_4(x) = -3.5536x^3 + 53.3035x^2 - 244.9639x + 360.428 \quad \text{for } 4 \le x \le 5$

c)

n	1	2	3	4	5
$\log\Gamma(n)$	0	0	$\log 2$	$\log 6$	$\log 24$

c) $\frac{1}{24}x^4\log 24 - \frac{1}{6}x^4\log 6 + \frac{1}{4}\log 2 - \frac{5}{12}x^3\log 24 + \frac{11}{6}x^3\log 6 -$
$- 3x^3\log 2 + \frac{35}{24}x^2\log(24) - \frac{41}{6}x^2\log 6 + \frac{49}{4}x^2\log 2 - \frac{25}{12}x\log 24 +$
$\frac{61}{6}x\log 6 - \frac{39}{2}x\log 2 + \log 24 - 5\log 6 + 10\log 2.$



$$Q_4(x) = 0.0071x^4 - 0.1187x^3 + 0.8820x^2 - 1.9211x + 1.1507$$

$$g(x) = e^{Q(x)}$$

Ex.3.3

$f(x) = \sqrt{x+1}$

interval $[-1, 1]$

```
>> pkg load symbolic
>> syms n x
```

```
>> C=chebyshevU(8,x)
C = (sym)

       8        6        4       2
  256*x  - 448*x  + 240*x  - 40*x  + 1
```

The function to approximate is $f(x) = \sqrt{1+x}$ and the approx polynomial is $p(x) = \sum_{i=0}^{d} p_i x^i$ of degree $d \geq 1$

Required that $p(0) = f(0) = 1$ and
$p(1) = f(1) = \sqrt{2}$

So we need to compute coefficients p_i that $1 \leq i \leq d-1$

An initial guess for $p(x)$ is $p(i/d) = f(i/d)$ for $1 \leq i \leq d-1$ in which case $g(x) = f(x) - p(x)$ is oscillatory

For degree 7, the conditions $p(0) = f(0)$

$p(1/7) = f(1/7)$

...

$p(1) = f(1)$

d	coefficients	d	coefficients
1	$p_0 = +1$ $p_1 = +4.1421356237309505 * 10^{-1}$ $e = +1.7766952966368793 * 10^{-2}$	2	$p_0 = +1$ $p_1 = +4.8563183076125260 * 10^{-1}$ $p_2 = -7.1418268388157458 * 10^{-2}$ $e = +1.1795695163108744 * 10^{-3}$
3	$p_0 = +1$ $p_1 = +4.9750045320242231 * 10^{-1}$ $p_2 = -1.0787308044477850 * 10^{-1}$ $p_3 = +2.4586189615451115 * 10^{-2}$ $e = +1.1309620116468910 * 10^{-4}$	4	$p_0 = +1$ $p_1 = +4.9955939832918816 * 10^{-1}$ $p_2 = -1.2024066151943025 * 10^{-1}$ $p_3 = +4.5461507257698486 * 10^{-2}$ $p_4 = -1.0566681694362146 * 10^{-2}$ $e = +1.2741170151556180 * 10^{-5}$
5	$p_0 = +1$ $p_1 = +4.9992197660031912 * 10^{-1}$ $p_2 = -1.2378506719245053 * 10^{-1}$ $p_3 = +5.6122776972699739 * 10^{-2}$ $p_4 = -2.3128836281145482 * 10^{-2}$ $p_5 = +5.0827122737047148 * 10^{-3}$ $e = +1.5725568940708201 * 10^{-6}$	6	$p_0 = +1$ $p_1 = +4.9998616695784914 * 10^{-1}$ $p_2 = -1.2470733323278438 * 10^{-1}$ $p_3 = +6.0388587356982271 * 10^{-2}$ $p_4 = -3.1692053551807930 * 10^{-2}$ $p_5 = +1.2856590305148075 * 10^{-2}$ $p_6 = -2.6183954624343642 * 10^{-3}$ $e = +2.0584155535630089 * 10^{-7}$
7	$p_0 = +1$ $p_1 = +4.9999754817809228 * 10^{-1}$ $p_2 = -1.2493243476353655 * 10^{-1}$ $p_3 = +6.1859954146370910 * 10^{-2}$ $p_4 = -3.6091595023208356 * 10^{-2}$ $p_5 = +1.9483946523450868 * 10^{-2}$ $p_6 = -7.5166134568007692 * 10^{-3}$ $p_7 = +1.4127567687864939 * 10^{-3}$ $e = +2.8072302919734948 * 10^{-8}$	8	$p_0 = +1$ $p_1 = +4.9999956583056759 * 10^{-1}$ $p_2 = -1.2498490369914350 * 10^{-1}$ $p_3 = +6.2318494667579216 * 10^{-2}$ $p_4 = -3.7982961896432244 * 10^{-2}$ $p_5 = +2.3642612312869460 * 10^{-2}$ $p_6 = -1.2529377587270574 * 10^{-2}$ $p_7 = +4.5382426960713929 * 10^{-3}$ $p_8 = -7.8810995273670414 * 10^{-4}$ $e = +3.9460605685825989 * 10^{-9}$

The numbers are coefficient pi for the polynomyal p(x), The table shows the maximum error for the approximation.

```
% construct N=8 data points
N=8;
xdata=linspace(-1,1,N);
```

```
ydata=sqrt(1+x)(xdata);

% construct many test points
xval=linspace(???,???,4001);
% construct the true test point values, for reference
yvalTrue=sinh(???);

% use Lagrange polynomial interpolation to evaluate
% the interpolant at the test points
yval=eval_lag(???,???,xval);

% plot reference values in thick green
plot(xval,yvalTrue,'g','linewidth',4);
hold on
% plot interpolation data points
plot(xdata,ydata,'k+');
% plot interpolant in thin black
plot(xval,yval,'k');
hold off

% estimate the approximation error of the interpolant
approximationError=max(abs(yvalTrue-yval))/max(abs(yvalTrue))
```

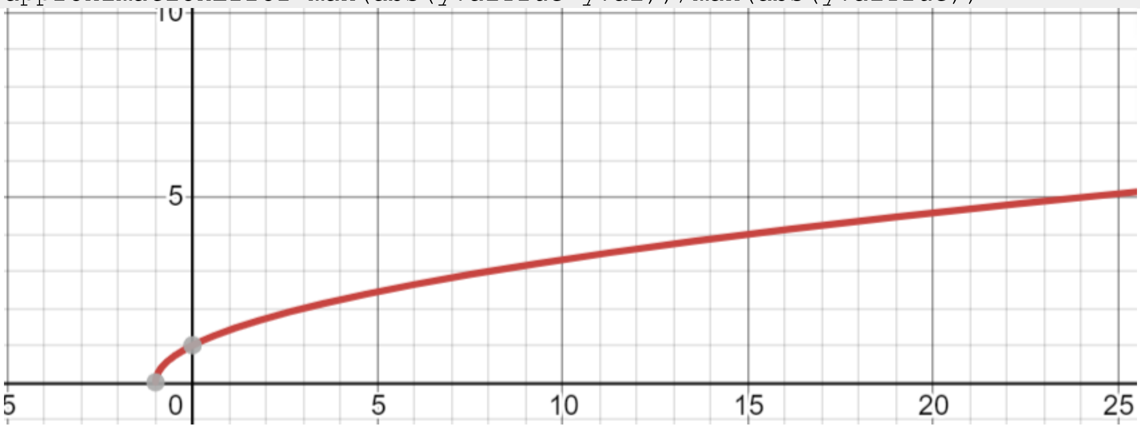


Figure 1 sqrt(x+1)