**Section 4.3**

13. Prove that the error term in $E_N(x) = f(x) - P_N(x)$ in the polynomial interpolation of a function $f(x)$ has the form

$$E_N(x) = (x-x_0)(x-x_1)\cdots(x-x_N) \frac{f^{(N+1)}(c)}{(N+1)!}.$$ 

Proof. Let $f$ be a $C^{N+1}$ function on $[a, b]$, and let $P_N(x)$ be the polynomial of degree $N$ that interpolates $f(x)$ at the $N+1$ points $x_0, \ldots, x_N$ in $[a, b]$. Fix $x$ in $[a, b]$ and define the following function of $t$:

$$g(t) = f(t) - P_N(t) - E_N(x) \frac{(t-x_0)(t-x_1)\cdots(t-x_N)}{(x-x_0)(x-x_1)\cdots(x-x_N)}.$$ 

Then $g(x_k) = 0$ for $k = 0, \ldots, N$. Since $g$ has $N+1$ zeros in $[a, b]$, by the Generalized Rolle Theorem, $g^{(N+1)}(c) = 0$ for some $c$ in $(a, b)$. Note that

$$\frac{(t-x_0)(t-x_1)\cdots(t-x_N)}{(x-x_0)(x-x_1)\cdots(x-x_N)} \frac{1}{(x-x_0)(x-x_1)\cdots(x-x_N)} = t^{N+1} + \cdots,$$

where the ellipses in the final term represent powers of $t$ less than $N+1$. The $N+1$ derivative of these terms is therefore zero. The $N+1$ derivative of $t^{N+1}$ is $(N+1)!$, and the $(N+1)$st derivative of an $N$ degree polynomial is zero, so

$$g^{(N+1)}(t) = f^{(N+1)}(t) - 0 - E_N(x) \frac{1}{(x-x_0)(x-x_1)\cdots(x-x_N)} (N+1)!.$$ 

This must be zero at some point $c$. Therefore, setting $g^{(N+1)}(c) = 0$, and solving for $E_N(x)$, we obtain the form of $E_N(x)$ required.
Algorithms and Programs

1. Repeat problem AP2 from 4.2 using the Lagrange formulation of the interpolating polynomial. Simply redo the calculations using a code that calculates the coefficients via the Lagrange formula. Note that if you use the code lagran.m in the text, the input vector Y must be a row vector, or else it will not work. You should obtain identical results to those obtained in 4.2 AP2.

2. The measured temperatures during a 5-hour period in a suburb of Los Angeles on November 8 are given in the table. (a) Construct an interpolating polynomial P(x) using the Lagrange formula for the data. (b) Estimate the average temperature. (c) Graph the data and interpolating polynomial and discuss any error that may result from the polynomial interpolation.

<table>
<thead>
<tr>
<th>Time</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deg F</td>
<td>66</td>
<td>66</td>
<td>65</td>
<td>64</td>
<td>63</td>
<td>63</td>
</tr>
</tbody>
</table>

(a) The interpolating polynomial is
P(x) = 0.0167 x^5 - 0.2917 x^4 + 2.0000 x^3 - 6.7083 x^2 + 9.9833 x + 61.0000.

(b) The average value of P(x) over 1 to 6 is \( \frac{1}{6-1} \int_{1}^{6} P(x)dx = 64.5 \), and the mean of the measurements is also 64.5. That is, the average of the interpolating function agrees precisely with the mean of the measured values.

(c) The graph of P(x) and measured values is shown below. The mean of P(x) agrees with the mean of values because the portions of the integral above and below the graph of lines connecting the points cancel exactly. This may lead to errors when these portions vary in different ways.
Section 4.4

5.7. (a) Compute the divided-difference table for the tabulated function. (b) Write down the Newton polynomials \( P_1(x), \ldots, P_4(x) \). (c) Evaluate the Newton polynomials at the given values of \( x \). (d) Compare the values in (c) with the actual values of \( f(x) \).

5. \( f(x) = x^{1/2} \)
   
   \( P_4(x) = 2.0 + 2.3607(x-4) - 0.01132(x-4)(x-5) + 0.0091(x-4)(x-5)(x-6) \)
   
   \( - 0.00008(x-4)(x-5)(x-6)(x-7) \)
   
   \( P_1(4.5) = 2.11804, \ P_2(4.5) = 2.12086, \ P_3(4.5) = 2.12121, \ P_4(4.5) = 2.12128 \)
   
   \( f(4.5) = 2.1213 \)

7. \( f(x) = 3 \sin^2(\pi x/6) \)
   
   \( P_4(x) = 0.0 + 0.75(x-0) \)
   
   \(+ 0.375(x-0)(x-1) \)
   
   \(- 0.25(x-0)(x-1)(x-2) \)
   
   \(- 0.03125(x-0)(x-1)(x-2)(x-3) \)
   
   \( P_1(1.5) = 1.125, \ P_2(1.5) = 1.40625, \ P_3(1.5) = 1.5, \ P_4(1.5) = 1.51758 \)
   
   \( f(1.5) = 1.5 \)

Algorithms and Programs

1. Repeat 4.2 AP2 using the Newton polynomial interpolation. Using the Newton polynomial, e.g., newpoly.m, one should obtain identical results as in 4.2 AP2.

2. The modification of Program 4.2 seen below is an equivalent way to compute the Newton polynomial. It uses the fact that, in the Newton form, once we have \( P_N(x) \),
   
   \( P_{N+1}(x) = P_N(x) + a_{N+1}(x-x_0)(x-x_1)\ldots(x-x_N), \)
   
   so \(a_{N+1} = \frac{f(x_{N+1}) - P_N(x_{N+1})}{(x_{N+1}-x_0)(x_{N+1}-x_1)\ldots(x_{N+1}-x_N)} \).

```matlab
function [C,A]=newpoly2(X,Y)   %Input - X is a vector that contains a list of abscissas %      - Y is a vector that contains a list of ordinates %Output - A  - coefficients in Newton form %       - C  - coefficients in standard form   n=length(X); A = Y; for j=2:n    for k=n:-1:j       A(k) =(A(k)-A(k-1))/(X(k)-X(k-j+1));    end end
C=A(n); for k=(n-1):-1:1    C=conv(C,poly(X(k)));    m=length(C);    C(m)=C(m)+A(k); end
```
Section 4.6

2. (a) Find the Pade approximation \( R_{1,1}(x) \) for \( f(x) = \ln(1+x)/x \).
Using the code below, we compute
\[
[p, q] = \text{padeR(@(x)log(1+x)/x,1,1)}
\]
which returns
\[
p = [1.1667]
q = [1.667]
\]
so \( R_{1,1}(x) = \frac{1 + 0.1667x}{1 + 0.667x} = \frac{6 + x}{6 + 4x} \)

(b) Establish the approximation \( \ln(1 + x) \approx R_{2,1}(x) = \frac{6x + x^2}{6 + 4x} \)
We can use the code or compute the coefficients by hand, or we can use the fact that
\[
\ln(1 + x) = \frac{\ln(1 + x)}{x} \approx R_{1,1}(x) = \frac{6 + x}{6 + 4x}
\]

3. (a) Find \( R_{1,1}(x) \) for \( f(x) = \tan(x^{1/2})/x^{1/2} \).
Using the code below, we compute
\[
[p, q] = \text{padeR(@(x)tan(sqrt(x))/sqrt(x),1,1)}
\]
which returns
\[
p = [1.0000 -0.0667]
q = [1.0000 -0.4000]
\]
so \( R_{1,1}(x) = \frac{1 - 0.0667x}{1 - 0.4x} = \frac{15 - x}{15 - 6x} \)

(b) Establish the approximation \( \tan(x) \approx R_{3,2}(x) = \frac{15x - x^3}{15 - 6x^2} \)
\[
\tan(x) = \frac{\tan\left(\sqrt{x^2}\right)}{\sqrt{x^2}} \approx R_{1,1}(x^2) = \frac{15 - x^2}{15 - 6x^2} \]

Algorithms and Programs

2. Compare the following approximations to \( f(x) = \ln(1+x) \).
   Taylor: \( T_5(x) = x - x^2/2 + x^3/3 - x^4/4 + x^5/5 \)
   Pade: \( R_{3,2}(x) = \frac{30x + 21x^2 + x^3}{30 + 36x + 9x^2} \)
Plot the function and the approximations on the same graph and compare the errors in the approximations.
Using the code below we find the Pade and Taylor approximations. We plot them on the interval \([-0.9, 0.1]\), and find the maximal errors in the approximations. We see that the Pade approximation gives a much smaller error (by a factor of 6).
function \([p, q] = \text{padeR}(f, N, M)\)

\[
\text{padeR}(f, N, M)
\]

\[
% \text{Finds the Pade approximating rational function}
% \quad \text{R}_{N,M} = P_N(x)/Q_M(x)
% \text{Input: } f - \text{function}
% \quad N - \text{degree of numerator}
% \quad M - \text{degree of denominator}
% \text{Output } p = \{p_0 \ p_1 \ldots \ p_N\} \text{ coefficients of } P(x)
% \quad q = \{q_0 \ q_1 \ldots \ q_M\} \text{ coefficients of } Q(x)
\]

\[
s\text{ym} x\text{x}
\]

\[
a = \text{zeros}(1, N+M+1);
a(1) = \text{limit}(f(xx), xx, 0);
\]

\[
\text{for } i = 1: (N+M+1)
\quad a(i+1) = \text{limit}(\text{diff}(f(xx), i), xx, 0)/\text{factorial}(i);
\text{end}
\]

\[
% <<< \text{find coefficients of } Q(x) >>>
A_q = \text{zeros}(M);
\text{for } i = 1: M
\quad A_q(i,:) = a(N-M+1+i:N+i);
\text{end}
q = -A_q\backslash a(N+2:N+M+1)';
q = [1 q(M:-1:1)'];
\]

\[
% <<< \text{find coefficients of } P(x) >>>
\]

\[
p(1) = a(1);
pq = \text{zeros}(1, N+1);
pq(1:M+1) = q;
\text{for } i = 1: N
\quad p(i+1) = \text{sum}(a(1:i+1).*pq(i+1:-1:1));
\text{end}
\]