1.5 Lösningar till kapitel 7

1: Nej, det säger bara att $q(t) = p(t)r(t)$ där r är ett polynom. Nej, det säger att $q(t) = p(t)\alpha$ där α är ett reellt tal. Ja, ty nu gäller det att $p(r_k) - q(r_k) = 0$ och att $p(\rho) - q(\rho) = 0$, så att polynomet $p - q$, av grad n, har $n + 1$ nollställen. Det måste alltså vara nollpolynomet.

2: $p(t) = (t-r)^m q(t)$ där $q(r) \neq 0$. För att beräkna derivatorna kan man t.ex. använda Leibniz formel:

$$p^{(k)}(t) = \sum_{j=0}^{k} \left[\left(\begin{array}{c} k \\ j \end{array} \right) \pi_j (t-r)^{m-j} q^{(k-j)}(t) \right], \quad \pi_j = m(m-1)\cdots(m-j+1), \quad \pi_0 = 1$$

Vi noterar att det ingår en faktor $t - r$, i alla termerna i summan, så länge som $k < m$.
Svaret på den andra frågan är nej, ty:

$$p^{(m)}(t) = \sum_{j=0}^{m} \left[\left(\begin{array}{c} m \\ j \end{array} \right) \pi_j (t-r)^{m-j} q^{(m-j)}(t) \right] \Rightarrow p^{(m)}(r) = m!q(r)$$

Om $p^{(m)}(r) =$ så är således $q(r) = 0$ vilket strider mot förutsättningarna.

3: Om p inte är konstant så kan det skrivas $p(t) = a_n t^n + \cdots$ där $a_n \neq 0$ och $n > 0$. Då gäller tydligen $p(t) = a_n t^n \left[1 + a_{n-1}t^{-1} + \ldots + a_0 t^{-n}\right]$ så att p uppför sig som $a_n t^n$ för stora $|t|$.

4: Antag att $p(t) = c$ för alla $t \in [a, b]$ med $a < b$. Eftersom det finns oändligt många reella tal i $[a, b]$ så har $p(t) - c$ oändligt många distinkta nollställen. $p(t) - c$ kan då bara vara nollpolynomet varför p är konstant lika med c.

5: Tag $k = 1$. Enligt medelvärdessatsen så existerar $\xi \in (t_1, t_2)$ så att $p'(\xi) = (y_2 - y_1)/(t_2 - t_1) < 0$. Analogt för de övriga intervallen. Vi har alltså en uppsättning $\xi_k \in (t_k, t_{k+1}), k = 1, \ldots, n-1$ så att $\text{sign}(p'(\xi_k)) = (-1)^k$. Detta medför att derivatan, $p'(t)$, (som inte kan vara konstant noll) har minst $n - 2$ **distinkta** nollställen i (t_1, t_n). Alltså måste $\text{grad}(p') \geq n - 2$ varför $\text{grad}(p) \geq n - 1$.

6: a) Låt $p(t) = x_3 t^2 + x_2 t + x_1$. Vi får ekvationssystemet

$$\left[\begin{array}{ccc} 1 & -1 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \end{array} \right] \left[\begin{array}{c} x_1 \\ x_2 \\ x_3 \end{array} \right] = \left[\begin{array}{c} 2 \\ 3 \\ 6 \end{array} \right]$$

Om vi löser systemet får vi $\mathbf{x} = [3, 2, 1]^T$, så att $p(t) = t^2 + 2t + 3$.
b) För Lagranges form skriver vi:

$$p(t) = 2 \cdot \frac{(t-0)(t-1)}{(-1-0)(-1-1)} + 3 \cdot \frac{(t-(-1))(t-1)}{(0-(-1))(0-1)} + 6 \cdot \frac{(t-(-1))(t-0)}{(1-(-1))(1-0)}$$

Vilket efter förenkling blir samma som i a). Slutligen Newtons form.

$$\left[\begin{array}{ccc} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 2 & 2 \end{array} \right] \left[\begin{array}{c} x_1 \\ x_2 \\ x_3 \end{array} \right] = \left[\begin{array}{c} 2 \\ 3 \\ 6 \end{array} \right]$$

ger $\mathbf{x} = [2, 1, 1]^T$ så att polynomet ges av: $p(t) = 2 + 1 \cdot (t - (-1)) + 1 \cdot (t - (-1))(t - 0) = t^2 + 2t + 3$. Det vanligaste sättet att beräkna Newtons form är att använda dividerade differenser (se boken). Så här ser

---

standarduppställningen ut med **fyra** punkter:

$$\begin{array}{cc|ccc} t_1 & f_1 & & & \\ & & f[t_1,t_2] & & \\ t_2 & f_2 & & f[t_1,t_2,t_3] & \\ & & f[t_2,t_3] & & f[t_1,t_2,t_3,t_4] \\ t_3 & f_3 & & f[t_2,t_3,t_4] & \\ & & f[t_3,t_4] & & \\ t_4 & f_4 & & & \end{array}$$

och där

$$f[t_i,t_{i+1},\ldots,t_{k-1},t_k]=\frac{f[t_{i+1},\ldots,t_{k-1},t_k]-f[t_i,t_{i+1},\ldots,t_{k-1}]}{t_k-t_i}$$

Så, i exemplet

$$\begin{array}{cc|ccc} -1 & 2 & & & \\ & & 1=(3-2)/(0-(-1)) & & \\ 0 & 3 & & & 1=(3-1)/(1-(-1)) \\ & & 3=(6-3)/(1-0) & & \\ 1 & 6 & & & \end{array}$$

Så polynomet blir:

$$p(t)=f_1+f[t_1,t_2](t-t_1)+f[t_1,t_2,t_3](t-t_1)(t-t_2)=2+1\cdot(t-(-1))+1\cdot(t-(-1))(t-0)=t^2+2t+3$$

7: Jag föredrar att skriva konstantermen längst till vänster, så:

$$-2+7t-3t^2+5t^3=-2+t(7+t(-3+5t))$$

Ett litet program kan se ut som (detta skrivs givetvis normalt som en loop där koefficienterna lagras i en vektor).

```
p =  5
p = -3 + t * p
p =  7 + t * p
p = -2 + t * p
```

8: Låt oss betrakta specialfallet, $n=4$:

$$\mathbf{A}=\begin{bmatrix} 1 & t_1 & t_1^2 & t_1^3 \\ 1 & t_2 & t_2^2 & t_2^3 \\ 1 & t_3 & t_3^2 & t_3^3 \\ 1 & t_4 & t_4^2 & t_4^3 \end{bmatrix}$$

Man kan resonera utifrån interpolationspolynomets existens och entydighet. Om $p(t_k)=y_k, k=1,...,n$ så gäller att polynomets koefficienter ges av $\mathbf{Ax}=\mathbf{y}$. $\mathbf{A}$ måste därmed vara ickesingulär. Ett annat bevis går direkt på determinanten av $\mathbf{A}$. Man kan bevisa (se kursen i linjär algebra) att

$$\det(\mathbf{A})=(t_1-t_2)(t_1-t_3)(t_1-t_4)(t_2-t_3)(t_2-t_4)(t_3-t_4)$$

i exemplet. Eftersom $t_j\neq t_k$ om $j\neq k$ så är determinanten skild från noll.

9: Vi har andragradspolynom $p_k(t), k=1,\ldots,n-1$ och kräver att $p_1(t_1)=y_1$, $p_k(t_k)=p_{k+1}(t_k)=y_k, k=2,\ldots,n-2$ samt $p_{n-1}(t_n)=y_n$. $n-1$ andragradspolynom ger oss $3(n-1)$ parametrar (koefficienter). Kontinuiteten (interpolationen) ger $2(n-1)$ ekvationer. Om vi kräver kontinuerlig förstaderivata får vi dessutom de

---

$n-2$ ekvationerna $p'_{k-1}(t_k) = p'_k(t_k), k = 2, \ldots, n-1$, dvs. totalt $2(n-1)+n-2 = 3n-4$ ekvationer. Detta borde alltså gå att lösa, ty antalet ekvationer, $3n-4$, är mindre än antalet parametrar, $3n-3$. Om vi dessutom kräver kontinuerlig andraderivata får vi ytterligare $n-2$ ekvationer. Om inte $n = 2$ (**ett** polynom) får vi flera ekvationer än obekanta. Detta problem är, i allmänhet, inte lösbart. Givetvis existerar en lösning ibland. Om t.ex. $y_k = 0, k = 1, \ldots, n$, så kommer alla derivator att vara kontinuerliga (interpolanten är ju nollfunktionen).

10: Vi skriver polynomet på Lagranges form:

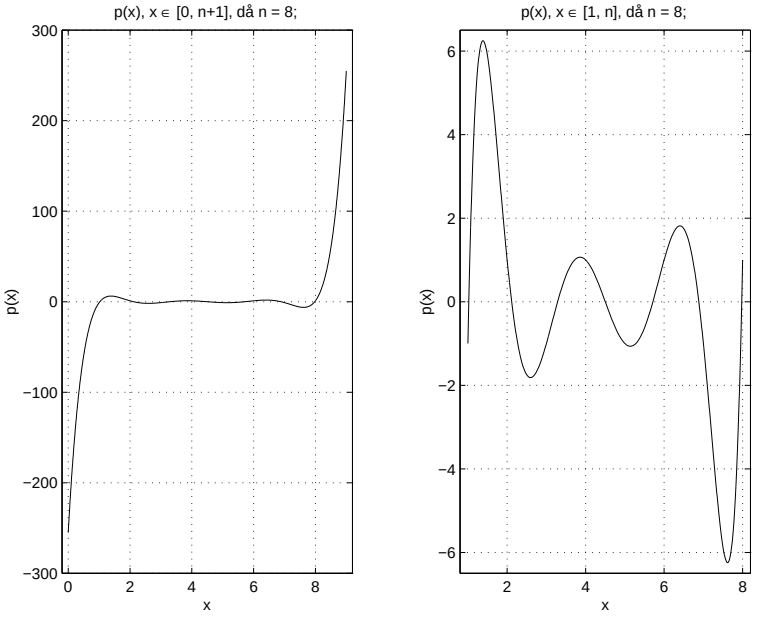
$$L_k(t) = \frac{\prod_{j\neq k}(t-t_j)}{\prod_{j\neq k}(t_k-t_j)}, \quad p(t) = \sum_{k=1}^{n}(-1)^k \epsilon L_k(t)$$

Eftersom ϵ multiplicerar alla termerna kan vi lika gärna studera fallet när $\epsilon = 1$, vilket vi antar från och med nu. Vill vi få med ϵ är det bara att multiplicera det slutgiltiga polynomet med ϵ. Vi beräknar nu $p(0)$:

$$p(0) = \sum_{k=1}^{n}(-1)^k \frac{(0-1)(0-2)\cdots(0-(k-1))(0-(k+1))\cdots(0-n)}{(k-1)(k-2)\cdots(k-(k-1))(k-(k+1))\cdots(k-n)} = \sum_{k=1}^{n} \frac{(-1)^k(-1)^{n-1}n!}{k(k-1)!(n-k)!(-1)^{n-k}} =$$

$$-\sum_{k=1}^{n}\binom{n}{k} = -(2^n-1)$$

Motsvarande gäller för $p(n+1)$ (förutom tecken om n är jämnt). Detta visar att polynomet ändrar sig mycket snabbt utanför intervallet, $[1, n]$. Det brukar vara farligt att extrapolera.



11: Matlabkoden består av två rutiner, den första, `inter_quad`, konstruerar koefficienterna i polynomen och den andra, `eval_quad`, evaluerar splinefunktionen.

```
function C = inter_quad(t, y, v)
%
% Given n points (t(k), y(k)), k = 1, ..., n, n >= 2,
% this functions computes a piecewise quadratic polynomial, q,
% consisting of the n-1 quadratic polynomials q_1, q_2, ...
% q interpolates the given data and has a continous first derivative.
% t should contain distinct values sorted in ascending order,
% so t(1) < t(2) < ... < t(n).
% v is used to specify the following end-point condition
% for the derivatives: v(1) q'(t(1)) + v(2) q'(t(n)) = v(3).
% v should be a non-zero array.
%
% The function returns a 3xn-matrix C where C(:, k) contains
% the three coefficients defining the quadratic polynomial
% on sub-interval number k (k = 1 is the left-most interval etc.).
% So, q_k(t) = C(1, k)*t^2 + C(2, k)*t + C(3, k)
%
% Thomas Ericsson, Nov. 15 2001.

% Some sanity checks..
if nargin < 2
  error('Not enough input arguments.')
end

if length(t) < 2 | length(y) < 2
  error('Not enough points.')
end

if length(t) ~= length(y)
  error('Unequal length of input arrays.')
end

if all(v == 0)
  error('v is the zero vector.')
end
%
% Number of polynomials = n - 1
%
% q_k(t) = a_k t^2 + b_k t + c_k, k = 1, ..., num_polys
%
% Equations of continuity (interpolation)
%
% q_1(t(1)) = y(1)   q_1(t(2)) = y(2)
% q_2(t(2)) = y(2)   q_2(t(3)) = y(3)
%   ...
% q_(n-1)(t(n-1)) = y(n-1)   q_(n-1)(t(n)) = y(n)
%
% This gives us 2(n-1) conditions
%
```

```
% Equations of continuity of q'(t)
%
% q_1'(t(2)) = q_2'(t(2))
% q_2'(t(3)) = q_3'(t(3))  etc., an additional n-2 conditions
%
% v(1) q'(t(1)) + v(2) q'(t(n)) = v(3) gives us the last condition.
% So, we have 2(n-1) + n-2 + 1 = 3(n-1) conditions, which equals
% the number of coefficients.
%
% Let x = [a_1 b_1 c_1  a_2 b_2 c_2 ...]' then the system,
% A x = b, is given by the following code:
%

n = length(t);
A = zeros(3 * (n - 1));
b = zeros(3 * (n - 1), 1);

% Continuity
% This can be coded in a MORE EFFICIENT way, but I have
% opted for clarity.
row = 1;
for k = 1:n-1                              % for each q_k ...
  cols            = 3 * (k - 1) + [1:3];  % columns in A
  A(row,   cols)  = [t(k)^2   t(k)   1];
  A(row+1, cols ) = [t(k+1)^2 t(k+1) 1];
  b(row)          = y(k);
  b(row + 1)      = y(k + 1);
  row             = row + 2;
end

% Continuity of q'. b(row) = 0 from above
for k = 1:n-2
  cols         = 3 * (k - 1) + [1:6];
  A(row, cols) = [2*t(k+1) 1 0 -2*t(k+1) -1 0];
  row          = row + 1;
end

% v(1) q'(t(1)) + v(2) q'(t(n)) = v(3).
v                 = v / norm(v); % for stability
A(end, 1:3)       = [2*v(1)*t(1)   v(1) 0];
A(end, end-2:end) = [2*v(2)*t(end) v(2) 0];
b(end)            = v(3);

x = A \ b;
cond_A = cond(A);
if cond_A > 1e10
  warning('A is ill-conditioned, cond(A) > 1e10. q may not be unique.')
  cond_A
end

% One column for each poly
C = reshape(x, 3, n-1);
```

Här följer `eval_quad`:

```
function qt = eval_quad(t, C, ts)
%
% Evalute a piecewise quadratic polynomial, in ts(1), ...,
% given the coefficient matrix C and the abscissas t.
% C and t must be unchanged from the call of inter_quad.
% If ts(k) < t(1) or t(end) < ts(k) extrapolation will be used.

qt = zeros(size(ts));

% Extrapolate if points < t(1)
i = find(ts < t(1));
if ~isempty(i)
  qt(i) = polyval(C(:, 1), ts(i));
end

% Extrapolate if points >= t(end)
i = find(ts >= t(end));
if ~isempty(i)
  qt(i) = polyval(C(:, end), ts(i));
end

% Interpolate if points in [t(k), t(k+1)]
for k = 2:length(t)
  i = find(t(k-1) <= ts & ts < t(k));
  if ~isempty(i)
    qt(i) = polyval(C(:, k-1), ts(i));
  end
end
```

12: Låt $e(t) = (t-t_1)(t-t_2)$ (för fixa t_1 och t_2). Funktionen är en parabel med $e(-1) \geq 0$ och $e(1) \geq 0$ eftersom t_1, t_2 skall ligga i intervallet $[-1,1]$. Derivatan av $e(t)$ är noll när då $t = m$, där $m = (t_1+t_2)/2$ och $e(m) = -(t_2-t_1)^2/4$. Det verkar mycket rimligt att kräva att $e(-1) = e(1) = -e(m)$ (notera minustecknet). Ty om t.ex. $e(-1) < e(1)$ borde man kunna få ett mindre max-värde genom att justera t_1, t_2. Den första likheten medför att $t_1 + t_2 = 0$. Detta i den andra likheten ger $t_2 = 1/\sqrt{2}$ och därmed $t_1 = -1/\sqrt{2}$. Kan man få något bättre? Nej, spännvidden i y-led år minst ett

$$(p(-1) - p(m)) + (p(1) - p(m)) = 2 + \frac{(t_1+t_2)^2}{4} \geq 2$$

Med våra värden blir just summan av dessa avstånd två.

13: Vi vet att

$$f(t) - p_n(t) = \frac{f^{(n)}(\theta)}{n!} \prod_{k=1}^{n} (t - t_k), \quad \theta \in \text{int}(t, t_1, \ldots, t_n)$$

Nu är n-te derivatan av e^t funktionen själv varför:

$$|e^t - p_n(t)| \leq \frac{e}{n!} \left| \prod_{k=1}^{n} (t - t_k) \right| \leq \frac{e}{n!} \quad (\dagger)$$

---

eftersom $|t - t_k| \leq 1$. Observera att det ger oss ett konvergensresultat för varje funktion vars alla derivator är begränsade på $[0,1]$, så $\left|f^{(n)}(t)\right| \leq M, 0 \leq t \leq 1$.
Om vi väljer t_k-punkterna på ett dåligt sätt får vi inte bättre konvergens än så här. Ett dåligt sätt är att låta värdena hopa sig i ändarna av intervallet (nära noll och ett). Man kan få snabbare konvergens med ett lämpligt val av punkter. Ett sådant är Chebyshevpunkterna, $c_k = -\cos((2k-1)\pi/(2n)), k = 1, \ldots n$. Vi vet att dessa minimerar $\Pi_{k=1}^n |t - t_k|$ när $|t| \leq 1$ och minvärdet är $1/2^{n-1}$. Nu har vi ju intervallet $[0,1]$ så vi får transformera punkterna c_k. Den linjära transformationen $2t - 1$ avbildar $[0,1]$ på $[-1,1]$. Så tag $t_k = (c_k + 1)/2$ (den inversa avbildningen). Vi får:

$$\max_{0\leq t\leq 1} \Pi_{k=1}^n |t - t_k| = \max_{0\leq t\leq 1} \Pi_{k=1}^n \left|t - \frac{c_k+1}{2}\right| = \max_{0\leq t\leq 1} \Pi_{k=1}^n \left|\frac{2t-1-c_k}{2}\right| = \frac{1}{2^n} \max_{-1\leq s\leq 1} \Pi_{k=1}^n |s - c_k| = \frac{1}{2^{2n-1}}$$

Detta tillsammans med (†) visar olikheten.

Låt $f(t)$ vara en funktion vars alla derivator begränsas av M då $a \leq t \leq b$. Vi bestämmer interpolationspolynomet, p_n, på $[a,b]$ som interpolerar $f(t)$ i punkterna $a = t_0 < t_1 < \ldots < t_n = b$. Oavsett hur vi väljer t_k-punkterna (i övrigt) så gäller att

$$\lim_{n\to\infty} \max_{a\leq t\leq b} |f(t) - p_n(t)| = 0$$

ty

$$f(t) - p_n(t) = \frac{f^{(n)}(\theta)}{n!} \prod_{k=1}^n (t - t_k), \quad \theta \in \text{int}(t, t_1, \ldots, t_n)$$

så att

$$|f(t) - p_n(t)| \leq \frac{M}{n!} \prod_{k=1}^n |t - t_k| \leq \frac{M(b-a)^n}{n!}$$