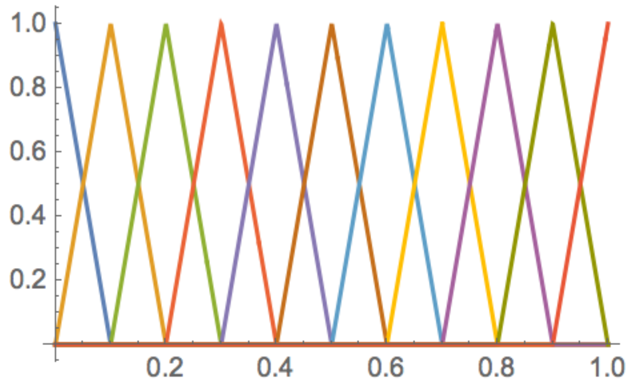
Caleb Logemann
MATH 561 Numerical Analysis I
Homework 2

#1 (a)



(b) If $j = k$, then $\pi_j(k/n) = \pi_j j/n = 1$. If $j \neq k$, then $\pi_j(k/n) = 0$, because k/n is outside of the division that is the support of π_j

(c) Let $c_0, c_1, \dots, c_n \in \mathbb{R}$ be given such that $\sum_{i=0}^{n}(c_i\pi_i(t)) = 0$ for all $t \in (0,1)$ Consider $t = k/n$ for some $k \in \{0, 1, \dots, n\}$, then $\sum_{i=0}^{n}(c_i\pi_i(k/n)) = c_k\pi_k(k/n)$, because $\pi_j k/n = 0$ for all $j \neq k$. Also $\pi_k(k/n) = 1$, so $c_k\pi_k(k/n) = c_k$. However this sum must be equal to 0 at $t = k/n$, so $c_k = 0$. This implies that $c_0 = c_1 = \dots = c_n = 0$. Thus the $\{\pi_j\}_{j=0}^{n}$ is linearly independent over the interval $(0,1)$. This also implies that $\{\pi_j\}_{j=0}^{n}$ is linearly independent over the points $\left\{0, 1/n, \dots, \frac{n-1}{n}, 1\right\}$, because at these points only one of the functions contributes to the overall sum.

(d) For $|i - j| > 1$, $\pi_i(t)\pi_j(t) = 0$ for $t \in (0,1)$. Therefore $\int_0^1 \pi_i(t)\pi_j(t)\,\mathrm{d}t = 0$ and $a_{ij} = 0$ for $|i - j| > 1$.

For $|i - j| = 1$, without loss of generality assume $j = i+1$. Note that $\operatorname{supp}(\pi_i(t)\pi_j(t)) = (i/n, i + 1/n) = (i/n, j/n)$

$$\int_0^1 \pi_i(t)\pi_j(t)\,\mathrm{d}t = \int_0^1 \pi_i(t)\pi_{i+1}(t)\,\mathrm{d}t$$

Since $\operatorname{supp}(\pi_i(t)\pi_{i+1}(t)) = (i/n, i + 1/n)$

$$= \int_{i/n}^{(i+1)/n} \pi_i(t)\pi_{i+1}(t)\,\mathrm{d}t$$

On this interval $\pi_i(t) = -nt + i + 1$ and $\pi_{i+1}(t) = nt - i$

$$
\begin{aligned}
&= \int_{i/n}^{(i+1)/n} (-nt + i + 1)(nt - i)\,\mathrm{d}t \\
&= \int_{i/n}^{(i+1)/n} -n^2t^2 + int + int + nt - i^2 - i\,\mathrm{d}t \\
&= \int_{i/n}^{(i+1)/n} -n^2t^2 + (2i+1)nt - i^2 - i\,\mathrm{d}t \\
&= -\frac{n^2}{3}t^3 + \frac{(2i+1)n}{2}t^2 - (i^2+i)t\bigg|_{t=i/n}^{(i+1)/n} \\
&= -\frac{n^2}{3}t^3 + \frac{(2i+1)n}{2}t^2 - (i^2+i)t\bigg|_{t=i/n}^{(i+1)/n} \\
&= -\frac{n^2}{3}\left(\frac{i+1}{n}\right)^3 + \frac{(2i+1)n}{2}\left(\frac{i+1}{n}\right)^2 - (i^2+i)\frac{i+1}{n} \\
&+ \frac{n^2}{3}\left(\frac{i}{n}\right)^3 - \frac{(2i+1)n}{2}\left(\frac{i}{n}\right)^2 + (i^2+i)\frac{i}{n} \\
&= -\frac{(i+1)^3}{3n} + \frac{(2i+1)(i+1)^2}{2n} - \frac{(i^2+i)(i+1)}{n} + \frac{i^3}{3n} - \frac{(2i+1)i^2}{2n} + \frac{i^3+i^2}{n} \\
&= -\frac{3i^2+3i+1}{3n} + \frac{(2i+1)^2}{2n} - \frac{(i^2+i)}{n} \\
&= \frac{1}{6n}
\end{aligned}
$$

For $|i-j| = 0$, that is $i = j$. Note that $\operatorname{supp}(\pi_i(t)^2) = (i - 1/n, i + 1/n)$.

$$\int_0^1 \pi_i(t)^2\,\mathrm{d}t = \int_{(i-1)/n}^{(i+1)/n} \pi_i(t)^2\,\mathrm{d}t$$

This can be split into two subintervals

$$= \int_{(i-1)/n}^{i/n} \pi_i(t)^2\,\mathrm{d}t + \int_{i/n}^{(i+1)/n} \pi_i(t)^2\,\mathrm{d}t$$

On $((i-1)/n, i)$, $\pi_i(t) = nt - i + 1$ and on $(i/n, (i+1)/n$, $\pi_i(t) = -nt + i + 1$

$$= \int_{(i-1)/n}^{i/n} (nt - i + 1)^2\,\mathrm{d}t + \int_{i/n}^{(i+1)/n} (-nt + i + 1)^2\,\mathrm{d}t$$

Because of symmetry

$$
\begin{aligned}
&= 2\int_{(i-1)/n}^{i/n} n^2t^2 - 2n(i-1)t + (i-1)^2\,\mathrm{d}t \\
&= 2\left(\frac{n^2}{3}t^3 - n(i-1)t^2 + (i-1)^2 t\right)\Bigg|_{t=(i-1)/n}^{i/n} \\
&= 2\left(\frac{i^3}{3n} - \frac{i^3-i^2}{n} + \frac{i(i-1)^2}{n} - \frac{(i-1)^3}{3n} + \frac{(i-1)^3}{n} - \frac{(i-1)^3}{n}\right) \\
&= 2\left(\frac{-i^2+i}{n} + \frac{3i^2-3i+1}{3n}\right) \\
&= \frac{2}{3n}
\end{aligned}
$$

If $i = j = 0$ or $i = j = n$, then symmetry does not apply, that is there is no other half. So $a_{00} = a_{nn} = \frac{1}{2}a_{ii} = \frac{1}{3n}$ for $1 \le i \le n-1$. Therefore

$$
a_{ij} = \begin{cases} 0 & |i-j| > 1 \\ \dfrac{1}{6n} & |i-j| = 1 \\ \dfrac{2}{3n} & |i-j| = 0 \text{ and } 1 \le i \le n-1 \\ \dfrac{1}{3n} & i = j = 0 \text{ or } i = j = n \end{cases}
$$

#2 (a) Find an upper bound for $\|f - p_2(f;\cdot)\|_\infty$ on equally spaced points. By the error of the interpolation problem $f(x) - p_2(f;x) = (x-x_0)(x-x_1)(x-x_2)\frac{f'''(\xi)}{6}$ Thus $\|f - p_2(f;\cdot)\|_\infty = \|(x-x_0)(x-x_1)(x-x_2)\|_\infty \frac{\|f'''(x)\|_\infty}{6}$. Using calculus, $\|(x-x_0)(x-x_1)(x-x_2)\|_\infty$ can be found.

$$
\begin{aligned}
g(x) &= (x-x_0)(x-x_1)(x-x_2) \\
g'(x) &= (x-x_1)(x-x_2) + (x-x_0)(x-x_2) + (x-x_0)(x-x_1) \\
&= 3x^2 - 2(x_0+x_1+x_2)x + x_0x_1 + x_0x_2 + x_1x_2 \\
&= 3x^2 - 6(x_0+h)x + 3x_0^2 + 6hx_0 + 2h^2
\end{aligned}
$$

Using the quadractic formula to solve for when $g'(x) = 0$

$$\begin{aligned}
x &= \frac{6(x_0+h) \pm \sqrt{36(x_0+h)^2 - 4\cdot 3(3x_0^2+6hx_0+2h^2)}}{6} \\
&= \frac{6(x_0+h) \pm 2\sqrt{3\big(3(x_0+h)^2 - (3x_0^2+6hx_0+2h^2)\big)}}{6} \\
&= \frac{3(x_0+h) \pm \sqrt{3\big(3x_0^2+6hx_0+3h^2 - (3x_0^2+6hx_0+2h^2)\big)}}{3} \\
&= \frac{3(x_0+h) \pm \sqrt{3h^2}}{3} \\
&= x_1 \pm \frac{\sqrt{3}}{3}h
\end{aligned}$$

So $g(x)$ achieves its maximum at $x = x_1 \pm \frac{\sqrt{3}}{3}h$. So $|g(x)| \leq g(x_1 + \frac{\sqrt{3}}{3}h)$.

$$\begin{aligned}
g(x_1 - \frac{\sqrt{3}}{3}h) &= (x_1 - \frac{\sqrt{3}}{3}h - x_0)(x_1 - \frac{\sqrt{3}}{3}h - x_1)(x_1 - \frac{\sqrt{3}}{3}h - x_2) \\
&= \frac{3-\sqrt{3}}{3}h \times \frac{-\sqrt{3}}{3}h \times \frac{-\sqrt{3}-3}{3}(h) \\
&= \frac{-3\sqrt{3}+3}{9}h^2 \times \frac{-\sqrt{3}-3}{3}(h) \\
&= \frac{6\sqrt{3}}{27}h^3 \\
&= \frac{2\sqrt{3}}{9}h^3
\end{aligned}$$

Therefore $\|f - p_2(f;\cdot)\|_\infty = \frac{2\sqrt{3}}{9}h^3\frac{\|f'''(x)\|_\infty}{6} = \frac{1}{9\sqrt{3}}\|f'''(x)\|_\infty h^3$.

(b) Let t_0, t_1, t_2 be the Chebyshev nodes on $[-1, 1]$. Then we know that $t_i = \cos\left(\frac{2i+1}{6}\pi\right)$, so $t_0 = \cos(()\pi/6) = \sqrt{3}/2$, $t_1 = \cos(()\pi/2) = 0$ and $\cos(()5\pi/6) = -\sqrt{3}/2$. In order to interpolate on $[x_0, x_2]$, these nodes need to be mapped to that domain. Let the map be $x = x_1 + ht$. The $c_0 = x_1 + \sqrt{3}/2h, c_1 = x_1, c_2 = x_1 - \sqrt{3}/2h$, will be the Chebyshev nodes on $[x_0, x_2]$.

Then we know that the error will be $f(x) - p_2(f;x) = (x-c_0)(x-c_1)(x-c_2)\frac{f'''(\xi)}{6}$.

$$\begin{aligned}
(x-c_0)(x-c_1)(x-c_2) &= (x - (x_1 + \sqrt{3}/2h))(x - x_1)(x - (x_1 - \sqrt{3}/2h) \\
&= (t-t_0)h(t-t_1)h(t-t_2)h \\
&= \frac{1}{4}T_2(t)h^3
\end{aligned}$$

Therefore $\|f(x) - p_2(f;x)\|_\infty = \frac{1}{4}h^3\frac{\|f'''(x)\|_\infty}{6} = \frac{1}{24}\|f'''(x)\|_\infty h^3$. This is a much better bound than for the equally spaced nodes.

#4

#6 (a) Table of divided differences

x_i	f_i				
-1	0				
-1	0	0			
0	1	1	1		
1	0	-1	-1	-1	
1	0	0	1	1	1

Using this table of divided differences the Hermite interpolation polynomial is $p(x) = (x+1)^2 - (x+1)^2x + (x+1)^2x(x-1)$. This can be simplified to $p(x) = x^4 - 2x^2 + 1$.

(b)(b.1) The error can be expressed as $e(x) = f(x) - p(x) = (x+1)^2x(x-1)^2\frac{f^{(5)}(\xi)}{5!}$ for $x \in [-1,1]$. This can be simplified by noting that $(x+1)(x-1) = (x^2-1)$. Therefore $e(x) = x(x^2-1)^2\frac{f^{(5)}(\xi)}{5!}$

(b.2) This can be bounded on $x \in [-1,1]$ by noting that $f^{(5)}(x) = -\frac{1}{2}\pi^5\sin(\pi x)$. Therefore $\left|f^{(5)}(\xi)\right| \le \frac{\pi^5}{2}$. So $|e(x)| \le \left|x(x^2-1)^2\frac{\pi^5}{2\cdot 5!}\right|$

(b.3) Consider $g(x) = x(x^2-1)^2$, where does this achieve its maxima on $[-1,1]$. These maxima can be found using calculus.

$$\begin{aligned} g'(x) &= 2x(x^2-1)2x + (x^2-1)^2 \\ &= (x^2-1)(4x^2 + x^2 - 1) \\ &= (x^2-1)(5x^2-1) \end{aligned}$$

Therefore $g'(x) = 0$ at $x = 1, -1, \frac{1}{\sqrt{5}}, \frac{1}{\sqrt{5}}$. Since $g(1) = 0$ and $g(-1) = 0$, these are not maxima. However $g(\pm\frac{1}{\sqrt{5}}) = \pm\frac{1}{\sqrt{5}}\frac{16}{25}$ are relative extrema. So $g(\frac{1}{\sqrt{5}}) = \frac{16}{25\sqrt{5}}$ is the maximum on $[-1,1]$. Thus $\max_{-1\le x\le 1}(e(x)) \le \frac{16}{25\sqrt{5}}\frac{\pi^5}{2\cdot 5!} \approx .36495$

#8 (a) Code for solving a tridiagonal system using Gaussian Elimination

```
function [y] = tridiag(n, a, b, c, v);
    % solve a tridiagonal system with Gaussian elimination
    % (a_1 c_1   0   0                   0  )(   y_1   )  (   v_1   )
    % (b_1 a_2 c_2   0                   0  )(   y_2   )  (   v_2   )
```

```
% (  0 b_2 a_3 c_3                  0  )(   y_3   )= (   v_3   )
% (  0                              0  )(         )= (         )
% (  .                 a_{n-1} c_{n-1})( y_{n-1} )  ( v_{n-1} )
% (  0                 b_{n-1}    a_n )(   y_n   )  (   v_n   )

p = inputParser;
p.addRequired('n', @Utils.checkPositiveInteger);
p.addRequired('a', @Utils.checkNumericVector);
p.addRequired('b', @Utils.checkNumericVector);
p.addRequired('c', @Utils.checkNumericVector);
p.addRequired('v', @Utils.checkNumericVector);
p.parse(n, a, b, c, v);

% error checking
if(length(a) ≠ n)
    disp('The vector a should have length n');
    return;
end

if(length(v) ≠ n)
    disp('The vector v should have length n')
    return;
end

if(length(b) ≠ n-1)
    disp('The vector b should have length n-1')
    return;
end

if(length(c) ≠ n-1)
    disp('The vector c should have length n-1')
    return;
end

% create array zero to store solutions
y = zeros(size(v));

% eliminate b_i's
for(i=1:n-1)
    a(i+1) = a(i+1) + c(i)*(-b(i)/a(i));
    v(i+1) = v(i+1) + v(i)*(-b(i)/a(i));
end

% solve for y_n
y(n) = v(n)/a(n);

for(i=(n-1):-1:1)
    y(i) = (v(i) - c(i)*y(i+1))/a(i);
end
```

```
end
```

(b) Code for creating natural cubic spline and displaying errormax

```
function [c0, c1, c2, c3] = cubicSpline(x, f)
    p = inputParser;
    p.addRequired('x', @Utils.checkNumericVector);
    p.addRequired('f', @Utils.checkNumericVector);
    p.parse(x, f);

    % error checking
    if(length(x) ≠ length(f))
        error('The lengths of x and f do not match');
    end

    % find ΔX and first divided differences
    ΔX = diff(x); % for 1 ≤ 1 ≤ n-1, ΔX(i) = x_{i+1} - x_i
    ddf = diff(f)./ΔX; % ddf(i) = [x_i, x_{i+1}]f = (f_{i+1} - ...
        f_i)(x_{i+1} - x_i)

    n = length(x);
    % cubicSpline results in tridiagonal system, so need to find ...
        a, b, c, and v for
    % m = tridiag(n, a, b, c, v), where m is coefficients for ...
        Hermite interpolation on
    % each interval
    a = zeros(n, 1);
    b = zeros(n-1,1);
    c = b;

    % common parts determined by continuity of second derivative ...
        at x_i for 2 ≤ i ≤ n-1
    % center diagonal
    a(2:n-1) = 2*(ΔX(2:n-1) + ΔX(1:n-2));
    b(1:n-2) = ΔX(2:n-1);
    c(2:n-1) = ΔX(1:n-2);
    v(2:n-1) = 3*(ΔX(1:n-2).*ddf(2:n-1) + ΔX(2:n-1).*ddf(1:n-2));

    % special by type of cubic spline
    % TODO: create more options than just natural cubic spline and
    % allow use to select type of cubic spline in input

    % natural cubic spline
    % first point p_1''(x_1) = 0
    a(1) = 2;
    c(1) = 1;
    v(1) = 3*ddf(1);
    % second point p_{n-1}''(x_n) = 0
    a(n) = 2;
```

```
    b(n-1) = 1;
    v(n) = 3*ddf(n-1);

    % tridiagonal system fully formed
    m = NumericalAnalysis.tridiag(n, a, b, c, v);

    % now that the values of m have been found, the coefficients ...
        of the interpolating
    % polynomials can be found.
    c0 = f(1:n-1);
    c1 = m(1:n-1);
    c2 = (3*ddf(1:n-1) - 2*m(1:n-1) - m(2:n))./ΔX(1:n-1);
    c3 = (m(2:n) + m(1:n-1) - 2*ddf(1:n-1))./(ΔX(1:n-1).^2);
end
```

```
% # 8
n = 11;
N = 51;
i = 1:n;

% part (1)
disp('part (1)');
x = (i-1)/(n-1);
f = exp(-x);
[c0, c1, c2, c3] = NumericalAnalysis.cubicSpline(x, f);
disp('e^(-x)');
printErrmax(c0, c1, c2, c3, x, @(x) exp(-x), N);

f = x.^(5/2);
[c0, c1, c2, c3] = NumericalAnalysis.cubicSpline(x, f);
disp('x^(5/2)');
printErrmax(c0, c1, c2, c3, x, @(x) x.^(5/2), N);

% part (2)
disp('part (2)');
x = x.^2;
f = x.^(5/2);
[c0, c1, c2, c3] = NumericalAnalysis.cubicSpline(x, f);
disp('x^(5/2)');
printErrmax(c0, c1, c2, c3, x, @(x) x.^(5/2), N);
```

(c)
```
>> H02
part (1)
e^(-x)
       1   4.9030e-04
       2   1.3163e-04
       3   3.5026e-05
```

```
         4   9.5467e-06
         5   2.2047e-06
         6   4.2094e-07
         7   3.4559e-06
         8   1.2809e-05
         9   4.8441e-05
        10   1.8036e-04
x^(5/2)
         1   2.0524e-04
         2   5.3392e-05
         3   1.6192e-05
         4   2.7607e-06
         5   1.2880e-06
         6   9.8059e-06
         7   3.4951e-05
         8   1.3252e-04
         9   4.9310e-04
        10   1.8416e-03
part (2)
x^(5/2)
         1   6.6901e-07
         2   2.3550e-07
         3   1.1749e-06
         4   1.6950e-06
         5   1.5853e-06
         6   1.6441e-05
         7   6.8027e-05
         8   3.2950e-04
         9   1.4755e-03
        10   6.5448e-03
```

This output shows the maximum error of these spline interpolations. The maximum error shows how the natural cubic spline interpolates better on the inner subdivisions. The natural cubic spline has larger errors near the endpoints, because the second derivative is forced to be zero at the endpoints. Note in part two because the subdivisions are no longer equal, the interpolation is better at the beggining where the subdivisions are smaller.