553.481/681 Numerical Analysis

Homework 3 Solutions

Lowen Peng
Problem 1. Define the Vandermonde matrix by its elements:

\[ V_{ij}[x_0, ..., x_n] = x_i^j, \quad 0 \leq i, j \leq n, \]

for any set of \((n + 1)\) real numbers \(x_0, ..., x_n\).

(a) Show that

\[ \det V[x_0, ..., x_n] = \prod_{i=0}^{n-1} (x_n - x_i) \cdot \det V[x_0, ..., x_{n-1}]. \]

**Hint:** Show that the determinant on the left is a polynomial of degree \(n\) in \(x_n\) and find its roots and the coefficient of its highest-order term.

(b) Use part (a) and induction to show that

\[ \det V[x_0, ..., x_n] = \prod_{0 \leq i < j \leq n} (x_j - x_i) \]

(a) Define \(f(x) := \det V[x_0, ..., x_{n-1}, x]\) and note that if \(x = x_i\) for some \(i = 0, ..., n - 1\) the rows of the Vandermonde matrix are linearly dependent and thus its determinant \(f\) vanishes. Note \(f\) is a polynomial of degree \(n\), which by the Fundamental Theorem of Algebra implies it has \(n\) roots. Thus the roots of \(f\) are precisely \(x_0, ..., x_{n-1}\), that is, \(f(x) = \alpha \prod_{i=1}^{n-1} (x - x_i)\) for some \(\alpha\). Note \(\alpha\) must be the leading coefficient associated with the \(x^n\) term. To find it, begin the determinant expansion for \(f(x)\) along the bottom row, from right to left:

\[
\det V[x_0, ..., x_{n-1}, x] = x^n \det \begin{bmatrix}
1 & x_0 & \cdots & x_0^{n-1} \\
1 & x_1 & \cdots & x_1^{n-1} \\
\vdots & \vdots & \ddots & \vdots \\
1 & x_{n-1} & \cdots & x_{n-1}^{n-1}
\end{bmatrix} + \cdots.
\]

Thus \(\alpha = \det V[x_0, ..., x_{n-1}]\) and we’re done.

(b) For the base case \(n = 0\) note that \(\det V[x_0] = 1\). Suppose now that for \(n - 1\) we have \(\det V[x_0, ..., x_{n-1}] = \prod_{0 \leq i < j \leq n-1} (x_j - x_i)\). Applying our result in (a) we have

\[
\det V[x_0, ..., x_n] = \det V[x_0, ..., x_{n-1}] \prod_{i=1}^{n-1} (x_n - x_i) = \prod_{0 \leq i < j \leq n} (x_j - x_i).
\]
Problem 2. Consider the following eight points:

\[(x_1, y_1) = (1, 8),\ (x_2, y_2) = (2, 26),\ (x_3, y_3) = (3, 76),\ (x_4, y_4) = (4, 44),\]

\[(x_5, y_5) = (5, 128),\ (x_6, y_6) = (6, 718),\ (x_7, y_7) = (7, 14516),\ (x_8, y_8) = (8, -4864).\]

For this data, find the 7th-degree interpolating polynomial (a) in the monomial basis, (b) in the barycentric Lagrange form, and (c) in the Newtonian form with divided-differences. Compare the wall clock times to compute the coefficients in the monomial basis, the barycentric weights, and the divided-differences. Which form of the interpolating polynomial is computed the fastest in this example? Plot the eight points along with the interpolating polynomial over the interval \([0, 9]\).

(a) Interpolating polynomial:

\[p(x) = -19x^7 + 549x^6 - 6476x^5 + 40243x^4 - 141340x^3 + 277926x^2 - 279443x + 108568.\]
xlabel('x','FontSize',15)
ylabel('y','FontSize',15)
legend('data','polynomial','Location','best')
title(fgmn,'FontSize',18)

(b) Barycentric weights:

\[-\frac{1}{5400},\frac{1}{720},\frac{1}{240},\frac{1}{144},\frac{1}{144},\frac{1}{240},\frac{1}{720},\frac{1}{5040}\).

\%
weights in barycentric representation
w=baryctrwt(x);
ww=abs(round(1./w));
wgtmn=['w=[');
for k=1:n-1
    if w(k)>0
        wgtmn=[wgtmn,'1/','num2str(ww(k))',','];
    else
        wgtmn=[wgtmn,'-1/','num2str(ww(k))',','];
    end
end
if w(n)>0
    wgtmn=[wgtmn,'1/','num2str(ww(n))','] '];
else
    wgtmn=[wgtmn,'-1/','num2str(ww(n))','] '];
end
disp(' ')
disp('barycentric weights: ')
disp(wgtmn)

(c) The divided differences are

\[(8, 18, 16, -19, 13, -1, 17, -19)\].

\%
coefficients of Newton basis
disp(' ')
disp('divided differences: ')
d=newtondif(x,y)

We also time:

timV=0;
timL=0;
timN=0;
NIT=10000;
for kk=1:NIT;
    tic
    A=vander(x);
    c=A\y';
    timV=timV+toc;
    tic
    w=baryctrwt(x);
    timL=timL+toc;
    tic
    d=newtondif(x,y);
    timN=timN+toc;
end

timV=timV/NIT
timL=timL/NIT
timN=timN/NIT
timVtoN=timV/timN
timLtoN=timL/timN
timLtoN

to find that \[\text{timVtoN} = 12.933649413788673\] so that interpolating with the monomial basis takes about 13 times longer than with the Lagrange basis, and \[\text{timLtoN} = 1.669106929613677\] so that interpolating with the Lagrange basis takes about 67% times longer than with the Newton basis.
Problem 3. Use the expression

\[ f[x_0, \ldots, x_n] = \sum_{i=0}^{n} w_i f(x_i), \quad w_i = 1/\Psi'_n(x_i) \]

for the divided-difference, with \( \Psi_n(x) = \prod_{j=0}^{n}(x - x_j) \), to verify that

\[ f[x_0, \ldots, x_n] = \frac{f[x_1, \ldots, x_n] - f[x_0, \ldots, x_{n-1}]}{x_n - x_0}. \]

(b) Show that the polynomial \( p_n(x) \) interpolating \( f(x) \) can be written as

\[ p_n(x) = \sum_{j=0}^{n} \frac{w_j f(x_j)}{x - x_j} \]

provided \( x \) is not a node point.

(a) Since

\[ f[x_0, \ldots, x_{n-1}] = \sum_{i=0}^{n-1} f(x_i) \prod_{0 \leq j \leq n-1, j \neq i}(x_i - x_j), \quad f[x_1, \ldots, x_n] = \sum_{i=1}^{n} f(x_i) \prod_{1 \leq j \leq n, j \neq i}(x_i - x_j), \]

note that

\[
\begin{align*}
\frac{f[x_1, \ldots, x_n] - f[x_0, \ldots, x_{n-1}]}{x_n - x_0} &= \sum_{i=1}^{n} \frac{f(x_i)}{f(x_n)} \prod_{1 \leq j \leq n, j \neq i}(x_i - x_j) - \sum_{i=0}^{n-1} \frac{f(x_i)}{f(x_0)} \prod_{0 \leq j \leq n-1, j \neq i}(x_i - x_j) \\
&= \prod_{0 \leq j \leq n-1}(x_n - x_j) - \prod_{1 \leq j \leq n}(x_0 - x_j) \\
&\quad + \sum_{i=1}^{n-1} f(x_i) \left[ \prod_{1 \leq j \leq n, j \neq i}(x_i - x_j) - \prod_{0 \leq j \leq n-1, j \neq i}(x_i - x_j) \right] \\
&= \frac{f(x_n) - f(x_0)}{\prod_{0 \leq j \leq n-1}(x_n - x_j) - \prod_{1 \leq j \leq n}(x_0 - x_j)} + \sum_{i=1}^{n-1} \frac{f(x_i)(x_n - x_0)}{\prod_{0 \leq j \leq n-1, j \neq i}(x_i - x_j)}. 
\end{align*}
\]

It’s immediate that

\[ \frac{f[x_1, \ldots, x_n] - f[x_0, \ldots, x_{n-1}]}{x_n - x_0} = \sum_{i=0}^{n} \frac{f(x_i)}{\prod_{0 \leq j \leq n, j \neq i}(x_n - x_j)} = f[x_0, \ldots, x_n]. \]

(b) Fix nodes \( x_0, \ldots, x_n \) and let \( q_n(x) \) be an interpolating polynomial for the constant function \( x \mapsto 1 \). Note that \( p_n(x)/q_n(x) \) remains an interpolating polynomial for \( f(x) \). We recall that the barycentric form for the Lagrangian interpolation of \( f \) is

\[ p_n(x) = \Psi_n(x) \sum_{i=0}^{n} f(x_i)w_i/(x - x_i). \]

Correspondingly we also have 1 = \( q_n(x) = \Psi_n(x) \sum_{i=0}^{n} w_i/(x - x_i) \). Thus we take

\[ p_n(x) := \frac{p_n(x)}{q_n(x)} = \frac{\sum_{i=0}^{n} f(x_i)w_i/(x - x_i)}{\sum_{i=0}^{n} w_i/(x - x_i)}. \]
Problem 4. Another example of the Runge phenomenon is provided by the function

\[ f(x) = \frac{1}{(1 + (5x)^6)^{1/6}}, \quad x \in [-1, 1]. \]

(a) Define the \( n \)-point Chebyshev grid in the interval \([-1, 1]\) by

\[ x_k = \cos \left( \frac{(2k-1)\pi}{2n} \right), \quad k = 1, \ldots, n. \]

Interpolate the above function with a polynomial on the \( n \)-point Chebyshev grid for \( n = 10, 20, 30, \ldots, 110, 120 \) and plot the results. What would you conjecture about the limit \( n \to \infty \) of the interpolating polynomial on the basis of these results?

(b) Compare the results in (a) with those obtained by polynomial interpolation on the uniform grid

\[ x_k = \frac{2k - (n + 1)}{n + 1}, \quad k = 1, \ldots, n \]

for \( n = 10, 20, 30, \ldots, 110, 120 \).

(a) Code:

```matlab
f = @(x) 1./(1+(5*x).^6).^(1/6);
u = -1:0.01:1;
uu = u;
w = f(u);

'Chebyshev grid'
n = 12;
for ii = 1:n
    kk = 1:10*ii;
x = cos((2.*kk-1).*pi./20/ii);
y = f(x);
v = baryctrint(x,y,u);
plot(x,y,'o',u,v,'-b',u,w,'-r');
axis([-1 1 0 1.5])
fgnm = ['Lagrange interpolant on Chebyshev grid with n=', num2str(10*ii), ' points '];
title(fgnm)
xlabel('x')
ylabel('y')
pause
plot(u,v-w,'-b');
fgnm = ['Error on Chebyshev grid with n=', num2str(10*ii), ' points '];
title(fgnm)
xlabel('x')
ylabel('err')
pause
end
```

The numerical results presented on the following page suggest that the interpolating polynomials \( p_n(x) \) on the Chebyshev grid converge in the limit \( n \to \infty \) to the function \( f(x) \).
The numerical results on the following page suggest for a real value $a \doteq 0.6$ that $\lim_{n \to \infty} p_n(x) = f(x)$ when $|x| < a$ but, because of the observed wild oscillations, that $\limsup_{n \to \infty} p_n(x) = +\infty$ and $\liminf_{n \to \infty} p_n(x) = -\infty$ when $|x| > a$. 

(b) Code:

```matlab
'uniform grid'
for ii=1:n
    kk=1:10*ii;
    xx=(2*kk-(10*ii+1))/(10*ii+1);
    yy=f(xx);
    vv=baryctrint(xx,yy,uu);
    plot(xx,yy,'o',uu,vv,-b',u,\,w,;-r')
    axis([-1 1 0 1.5])
    fgnm=['Lagrange interpolant on uniform grid with n=', num2str(10*ii), ' points '];
    title(fgnm)
    xlabel('x')
    ylabel('y')
    pause
end
```
Problem 5. In the root-finding method of inverse quadratic interpolation a quadratic polynomial \( x = p_2(y) \) is fit to the three points \((f_a, a), (f_b, b), (f_c, c)\).

(a) Use the Lagrange form of the interpolating quadratic \( p_2(y) \) to show that it crosses the \( x \)-axis at the unique point

\[
x = a \frac{f_b f_c}{(f_a - f_b)(f_a - f_c)} + b \frac{f_a f_c}{(f_b - f_a)(f_b - f_c)} + c \frac{f_a f_b}{(f_c - f_a)(f_c - f_b)}.
\]

(b) Use the result in (a) to derive the expression used in \texttt{fzerotx.m}, i.e. show that if

\[
\begin{align*}
    r &= f_b / f_a, \quad s = f_b / f_c, \quad t = f_c / f_a \\
    p &= s[(a - b)t(t - r) - (b - c)(r - 1)], \quad q = (r - 1)(s - 1)(t - 1),
\end{align*}
\]

then \( \hat{x} = b - p / q \). [\textit{Hint:} In the latter expression for \( \hat{x} \) gather together all of the terms proportional to \( a, b, \) and \( c \).]

(c) Generalize the formula in (a) to give an approximate root \( x \) by inverse cubic interpolation, using a cubic polynomial \( x = p_3(y) \) to fit four points \((f_a, a), (f_b, b), (f_c, c), (f_d, d)\).

(a) Write

\[
p_2(y) = \frac{a(y - f_b)(y - f_c)}{(f_a - f_b)(f_a - f_c)} + \frac{b(y - f_a)(y - f_c)}{(f_b - f_a)(f_b - f_c)} + \frac{c(y - f_a)(y - f_b)}{(f_c - f_a)(f_c - f_b)}.
\]

Note \( x = p_2(0) \) is of the desired form.

(b) In the expression from (a) for \( x \) divide the numerators and denominators of each term by \( f_a^2 \) and note \( s = r / t \) to write

\[

p_2(0) = \frac{ar}{(r - 1)(t - 1)} + \frac{bt}{(t - r)(1 - r)} + \frac{cr}{(1 - t)(r - t)}
\]

\[
= \frac{ar}{(r - 1)(t - 1)} + \frac{bt}{(1 - s)(1 - r)} + \frac{cr}{(1 - t)(s - 1)}
\]

\[
= \frac{ar(s - 1) + b(t - 1) - cs(r - 1)}{(r - 1)(s - 1)(t - 1)}
\]

\[
= b - \frac{-ar(s - 1) + b(r - 1)(s - 1)(t - 1) - b(t - 1) + cs(r - 1)}{(r - 1)(s - 1)(t - 1)}
\]

\[
= b - \frac{-ar(t - r) + b(r^2 - rt - rs + s) + cs(r - 1)}{(r - 1)(s - 1)(t - 1)}
\]

\[
= b - \frac{ast(t - r) - bst(t - r) - hs(r - 1) + cs(r - 1)}{(r - 1)(s - 1)(t - 1)}
\]

\[
= b - \frac{s[(a - b)t(t - r) - (b - c)(r - 1)]}{(r - 1)(s - 1)(t - 1)}.
\]

The numerator and denominator of the fraction are respectively \( p \) and \( q \) so we are done.

(c) A similar exercise to (a) will show that

\[
x = p_3(0) = \frac{a f_b f_c f_d}{(f_a - f_b)(f_a - f_c)(f_a - f_d)} + \frac{b f_a f_c f_d}{(f_b - f_a)(f_b - f_c)(f_b - f_d)} + \frac{c f_a f_b f_d}{(f_c - f_a)(f_c - f_b)(f_c - f_d)} + \frac{d f_a f_b f_c}{(f_d - f_a)(f_d - f_b)(f_d - f_c)}.
\]
Problem 6*. Given \( n \) distinct points \( x_i, \, i = 1, \ldots, n \) define in terms of the standard Lagrange polynomials \( \ell_i(x), \, i = 1, \ldots, n \) the new set of \( 2n \) polynomials

\[
h_i(x) = [1 - 2\ell'_i(x_i)(x - x_i)](\ell_i(x))^2, \quad i = 1, \ldots, n
\]

and

\[
\tilde{h}_i(x) = (x - x_i)(\ell_i(x))^2, \quad i = 1, \ldots, n
\]

which are each of degree \( 2n - 1 \).

(a) Show that for all \( i, j \),

\[
h_i(x_j) = \delta_{ij}, \quad h'_i(x_j) = 0
\]

\[
\tilde{h}_i(x_j) = 0, \quad \tilde{h}'_i(x_j) = \delta_{ij},
\]

where \( \delta_{ij} \) is the Kronecker delta function.

(b) Use this result to show that the Hermite interpolating polynomial \( H_n(x) \) of degree at most \( 2n - 1 \) which satisfies

\[
H_n(x_i) = y_i, \quad H'_n(x_i) = y'_i, \quad i = 1, \ldots, n
\]

can be written in a “Lagrange form” as

\[
H_n(x) = \sum_{i=1}^{n} \left[ y_i h_i(x) + y'_i \tilde{h}_i(x) \right].
\]

(a) Since \( \ell_i(x_j) = \delta_{ij} \) and \( (x_j - x_i)\delta_{ij} = 0 \) for all \( i, j \), then

\[
h_i(x_j) = [1 - 2\ell'_i(x_i)(x_j - x_i)]\delta_{ij} = \delta_{ij}
\]

and

\[
\tilde{h}_i(x_j) = (x_j - x_i)\delta_{ij} = 0
\]

for all \( i, j \).

Since \( h'_i(x) = 2\ell'_i(x)\ell_i(x)[1 - 2\ell'_i(x_i)(x - x_i)] - 2\ell'_i(x)\ell_i(x)^2 \),

\[
h'_i(x_j) = 2\delta_{ij}\ell'_i(x_j)[1 - 2\ell'_i(x_i)(x_j - x_i)] - 2\ell'_i(x_j)\delta_{ij} = -4[\ell'_i(x_j)]^2(x_j - x_i)\delta_{ij} = 0
\]

for all \( i, j \).

Similarly \( \tilde{h}'_i(x) = \ell_i(x)^2 + 2(x - x_i)\ell'_i(x)\ell_i(x) \) so that

\[
\tilde{h}'_i(x_j) = \delta_{ij} + 2(x_j - x_i)\ell'_i(x_j)\delta_{ij} = \delta_{ij}.
\]

(b) We have

\[
H_n(x) = \sum_{i=1}^{n} \left[ y_i h_i(x) + y'_i \tilde{h}_i(x) \right] = \sum_{i=1}^{n} y_i \delta_{ij} = y_j,
\]

Similarly, since \( H'_n(x) = \sum_{i=1}^{n} \left[ y_i h'_i(x) + y'_i \tilde{h}'_i(x) \right] \),

\[
H'_n(x_j) = \sum_{i=1}^{n} \left[ y_i h'_i(x_j) + y'_i \tilde{h}'_i(x_j) \right] = \sum_{i=1}^{n} y'_i \delta_{ij} = y'_j.
\]
(a) By modifying the code `hermitedif.m`, write a script `hermite2dif.m` to find the coefficients of the 2nd-order Hermite interpolating polynomial satisfying

\[ H_n(x_i) = y_i, \quad H'_n(x_i) = y'_i, \quad H''_n(x_i) = y''_i, \quad i = 1, \ldots, n \]

for general data \( x = (x_1, \ldots, x_n) \), \( y = (y_1, \ldots, y_n) \), \( y' = (y'_1, \ldots, y'_n) \), and \( y'' = (y''_1, \ldots, y''_n) \). Hint: If the vector is tripled to \( X = (x_1, x_1, x_2, \ldots, x_n, x_n, x_n) \), then only the loops over \( i = 1 \) and \( i = 2 \) must be modified.

(b) Similarly, write a script `hermite2int.m` to evaluate the 2nd-order Hermite interpolating polynomial, by modifying `hermiteint.m`.

(c) Apply the code in (b) to Hermite interpolate the function \( f(x) = e^{-x} \sin x \) on the points \( x = 1, 2, 3, \ldots, 7 \). Plot the interpolating polynomial on the interval \([-1, 9]\) along with the original function \( f(x) \).

(d) Compare the results in (c) with the Lagrange interpolating polynomial and the 1st-order Hermite interpolating polynomial for the same points and function. Plot all three interpolating polynomials together with the function itself on the interval \([-1, 9]\) and plot also on this interval the absolute error of each interpolant in log-scale. Order the interpolants in terms of accuracy. Comment also on the relative accuracy of extrapolation versus interpolation.

---

**Problem 7**. (a) The function `hermite2dif.m`:

```matlab
function d=hermite2dif(x,y,yp,ypp)
    n=length(x);
    nn=3*n;
    for i=1:n
        xx(3*i-2)=x(i);
        xx(3*i-1)=x(i);
        xx(3*i)=x(i);
    end
    d(1)=y(1);
    d(2)=yp(1);
    for j=1:n
        d(3*j)=ypp(j)/2;
    end
    for j=2:n
        fd(j)=(y(j)-y(j-1))/(x(j)-x(j-1));
        d(3*j-2)=(fd(j)-yp(j-1))/(x(j)-x(j-1));
        d(3*j-1)=(yp(j)-fd(j))/(x(j)-x(j-1));
    end
    for i=3:nn-1
        for j=nn:-1:i+1
            d(j)=(d(j)-d(j-1))/(xx(j)-xx(j-1));
        end
    end
```

(b) The function `hermite2int.m`:

```matlab
function v=hermite2int(x,y,yp,ypp,u)
    n=length(x);
    nn=3*n;
    d=hermite2dif(x,y,yp,ypp);
    for i=1:n
        xx(3*i-2)=x(i);
        xx(3*i-1)=x(i);
        xx(3*i)=x(i);
    end
    v=d(nn).*ones(size(u));
    for k=nn:-1:2
        v=d(k-1)+(u-xx(k-1)).*v;
    end
```

(c) See:
The numerical results on the following page show that the methods are ordered from least to most accurate as Lagrange, 1st-order Hermite, and 2nd-order Hermite. The results show also that the errors are smallest within the interpolating interval [1, 7] from which the data points $x_i$ are selected, whereas errors are larger when extrapolating outside this interval.