EN.553.481/681 Numerical Analysis – Homework 3 Solutions

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

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

Thus $\alpha = \det \mathbf{V}[x_0, \ldots, x_{n-1}]$ and we're done.

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

$$\det \mathbf{V}[x_0, \ldots, x_n] = \det \mathbf{V}[x_0, \ldots, x_{n-1}] \prod_{i=1}^{n-1}(x_n - x_i) = \prod_{0 \le i < j \le n}(x_j - x_i).$$

Problem 2. Implemented as `interp.m` in Program 1. Plot in Figure 1. The monomial basis computes in time $4.139508333333372 \times 10^{-5}$, barycentric representation in $1.144091666666599 \times 10^{-6}$, and Newton divided differences in $7.850166666664648 \times 10^{-7}$. The Newton divided difference method is fastest. The monomial basis is roughly 52 times slower, and barycentric representation is roughly 1.457 times slower.

(a) We find the interpolating polynomial to be $-5x^6 + 101x^5 - 798x^4 + 3160x^3 - 6604x^2 + 6840x - 2698$.

(b) The barycentric weghts are $w = [1/720, -1/120, 1/48, -1/36, 1/48, -1/120, 1/720]$.

(c) The divided differences are $d = (-4, -6, -9, -5, 17, -4, -5)$.

Problem 3. (a) Since by the induction hypothesis

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

then

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

where in the first term of the last line we used $(x_i - x_0) - (x_i - x_n) = (x_n - x_0)$. It is then immediate that

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

(b) Fix nodes $x_0, \ldots, x_n$ and let $q_n(x) \equiv 1$ 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 $p_n(x) = \Psi_n(x) \sum_{i=0}^{n} \frac{f(x_i) w_i}{(x - x_i)}$. Correspondingly we also have $1 = q_n(x) = \Psi_n(x) \sum_{i=0}^{n} \frac{w_i}{(x - x_i)}$. Thus we take

$$p_n(x) := \frac{p_n(x)}{q_n(x)} = \frac{\sum_{i=0}^{n} \frac{f(x_i) w_i}{(x - x_i)}}{\sum_{i=0}^{n} \frac{w_i}{(x - x_i)}}.$$

Problem 4. (a) By inspection the data could have been generated by a strictly monotonic function f, which would admit a functional inverse $g = f^{-1}$. In that case, the data have been consistently generated both by $(x_i, f(x_i))$ and also by $(g(y_i), y_i)$.

(b) Implemented as `dvi.m` in Program 2. The divided differences for the direct method are

$$d \approx (1.0000, 0.2623, -0.0364, 0.0050, -0.0005, 0.0)$$

For the inverse method we find $d \approx 17.0000, 3.8124, 3.7504, 0.9995, 0, -2.1261$.

(c) Plot in Figure 2. By inspection it's clear Q_6 is not one-to-one and cannot be the functional inverse of P_6, and vice versa. We only have a guarantee for the nodes (x_i, y_i) that for $i = 0, \ldots, 6$ we have $Q_6(y_i) = x_i$ and $y_i = P_6(x_i)$.

Problem 5. (a) Implemented as `chebygrid.m` in Program 3. See Figure 3. The interpolating polynomials appear to be converging to the true function in a uniform fashion.

(b) Implemented as `unigrid.m` in Program 4. See Figure 4. The interpolating polynomials now appear to converge as $n \to \infty$ only for about $|x| < 0.7$ and oscillate more wildly with increasing n for larger $|x|$.

Problem 6.* (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 $p_2(0)$ is of the desired form.

(b) In the above expression for $p_2(0)$ divide f_a^2 into both the numerators and denominators of each term and note $s = r/t$ to write

$$\begin{aligned}
p_2(0) &= \frac{art}{(r-1)(t-1)} + \frac{bt}{(t-r)(1-r)} + \frac{cr}{(1-t)(r-t)} \\
&= \frac{art}{(r-1)(t-1)} + \frac{b}{(1-s)(1-r)} + \frac{cs}{(1-t)(s-1)} \\
&= \frac{art(s-1) + b(t-1) - cs(r-1)}{(r-1)(s-1)(t-1)} \\
&= b - \frac{-art(s-1) + b(r-1)(s-1)(t-1) - b(t-1) + cs(r-1)}{(r-1)(s-1)(t-1)} \\
&= b - \frac{-art(s-1) + b(t-1)(rs - r - s) + cs(r-1)}{(r-1)(s-1)(t-1)}
\end{aligned}$$

$$
\begin{aligned}
&= b - \frac{-ar(t-r) + b(r^2 - rt - rs + s) + cs(r-1)}{(r-1)(s-1)(t-1)} \\
&= b - \frac{ast(t-r) - bst(t-r) - bs(r-1) + cs(r-1)}{(r-1)(s-1)(t-1)} \\
&= b - \frac{s[(a-b)t(t-r) - (b-c)(r-1)]}{(r-1)(s-1)(t-1)}.
\end{aligned}
$$

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

$$
\begin{aligned}
p_3(0) = &\frac{af_bf_cf_d}{(f_a-f_b)(f_a-f_c)(f_a-f_d)} + \frac{bf_af_cf_d}{(f_b-f_a)(f_b-f_c)(f_b-f_d)} \\
&+ \frac{cf_af_bf_d}{(f_c-f_a)(f_c-f_b)(f_c-f_d)} + \frac{df_af_bf_c}{(f_d-f_a)(f_d-f_b)(f_d-f_c)}.
\end{aligned}
$$

Problem 7.* (a) First, note that

$$h_i(x_j) = [1 - 2\ell_i'(x_i)(x_j - x_i)](\delta_{ij})^2 = \delta_{ij}$$

Applying the product and chain rules,

$$h_i'(x) = 2\ell_i(x)\ell_i'(x)[1 - 2\ell_i'(x_i)(x - x_i)] - 2\ell_i'(x_i)[\ell_i(x)]^2$$

Therefore

$$
\begin{aligned}
h_i'(x_j) &= 2\delta_{ij}\ell_i'(x_i)[1 - 2\ell_i'(x_i)(x_j - x_i)] - 2\ell_i'(x_i)[\delta_{ij}]^2 \\
&= 2\delta_{ij}\ell_i'(x_i) - 2\ell_i'(x_i)\delta_{ij} \\
&= 0
\end{aligned}
$$

Next, we can check

$$
\begin{aligned}
\tilde{h}_i(x_j) &= (x_j - x_i)(\delta_{ij})^2 \\
&= (x_j - x_i)\delta_{ij} \\
&= 0
\end{aligned}
$$

Applying the product rule:

$$\tilde{h}_i'(x) = (\ell_i(x))^2 + 2(x - x_i)\ell_i'(x)\ell_i(x)$$

Finally

$$
\begin{aligned}
\tilde{h}_i'(x_j) &= (\delta_{ij})^2 + 2(x_j - x_i)\ell_i'(x_j)\delta_{ij} \\
&= \delta_{ij}
\end{aligned}
$$

since $2(x_j - x_i)\ell_i'(x_j) = 0$. Therefore we see that all four properties are satisfied.

(b) Taking

$$H_n(x) = \sum_{i=1}^{n} [\, y_i\, h_i(x) + y_i'\, \tilde{h}_i(x)\,]$$

we can then check the two properties. First, using properties from part a):

$$H_n(x_i) = \sum_{k=1}^{n} [\, y_k\, h_k(x_i) + y_k'\, \tilde{h}_k(x_i)\,]$$

$$= \sum_{k=1}^{n} [\, y_k \, \delta_{ki} + y'_k \, \cdot 0 \,]$$
$$= y_i$$

In the above, we switched the index of the sum to "k" instead of "i" simply for clarity of notation. The above works because $\delta_{ki} = 1$ only if $k = i$ which plucks out the y_i term in the sum. Next,

$$H'_n(x) = \sum_{k=1}^{n} [\, y_k \, h'_k(x) + y'_k \, \tilde{h}'_k(x) \,]$$

Therefore

$$H'_n(x_i) = \sum_{k=1}^{n} [\, y_k \, h'_k(x_i) + y'_k \, \tilde{h}'_k(x_i) \,]$$
$$= \sum_{k=1}^{n} [\, y_k \, \cdot 0 + y'_k \, \delta_{ki} \,]$$
$$= y'_i$$

Since the Hermite interpolating polynomial is unique (as proved in lecture) and our "Lagrange form" satisfies the two properties, we have proved the claim.

Problem 8.* (a) Implemented in Program 5.

(b) Implemented in Program 6.

(c) Implemented in Program 7. Plotted in Figure 5.

(d) All three interpolating polynomials are plotted together with the function itself in Figure 6. The respective errors of Hermite and Lagrangian interpolation are plotted in Figure 7. The method of Hermite is more accurate for interpolation. However, for extrapolation we see in Figure 7 that the Lagrange polynomial is actually closer to the exact answer than Hermite is on the far ends. The moral is that more accurate interpolation does not guarantee better extrapolation.
By inspection it is clear that interpolation is more accurate than extrapolation.

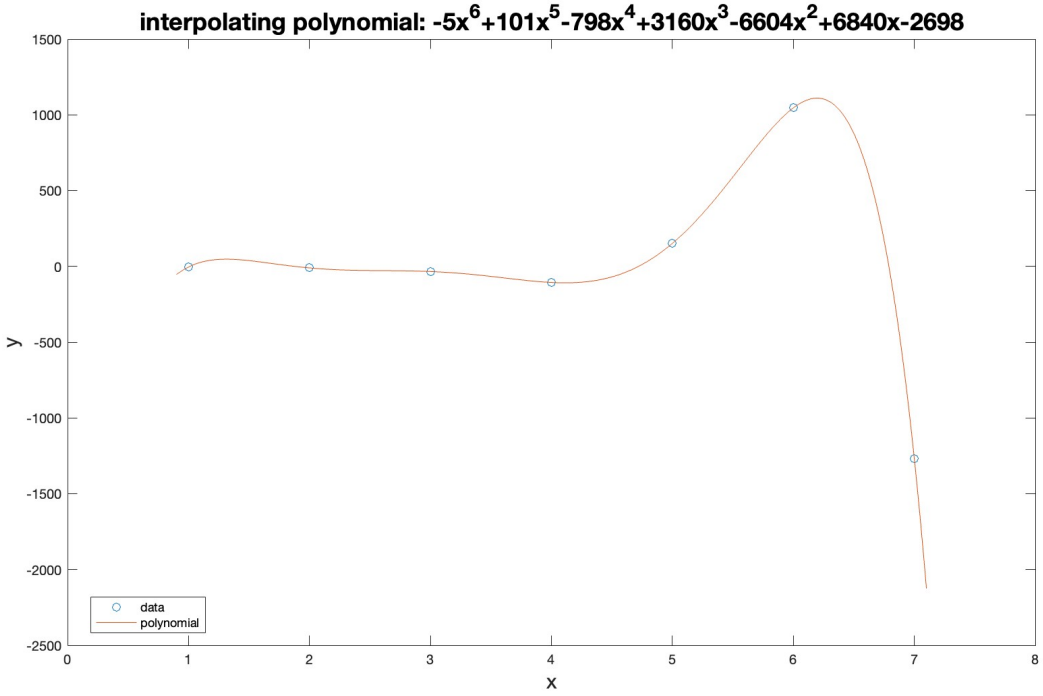

Figure 1. The interpolating polynomial along with the 7 points.

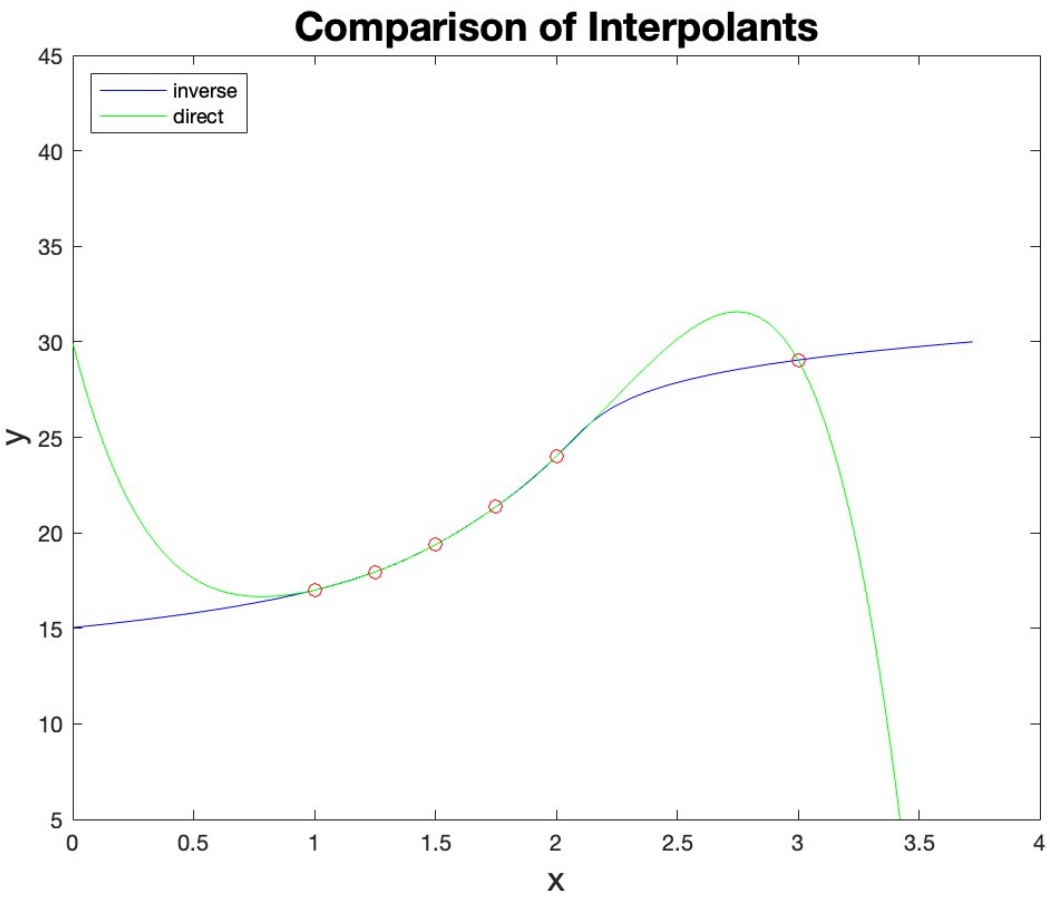

Figure 2. Direct vs. inverse interpolation.

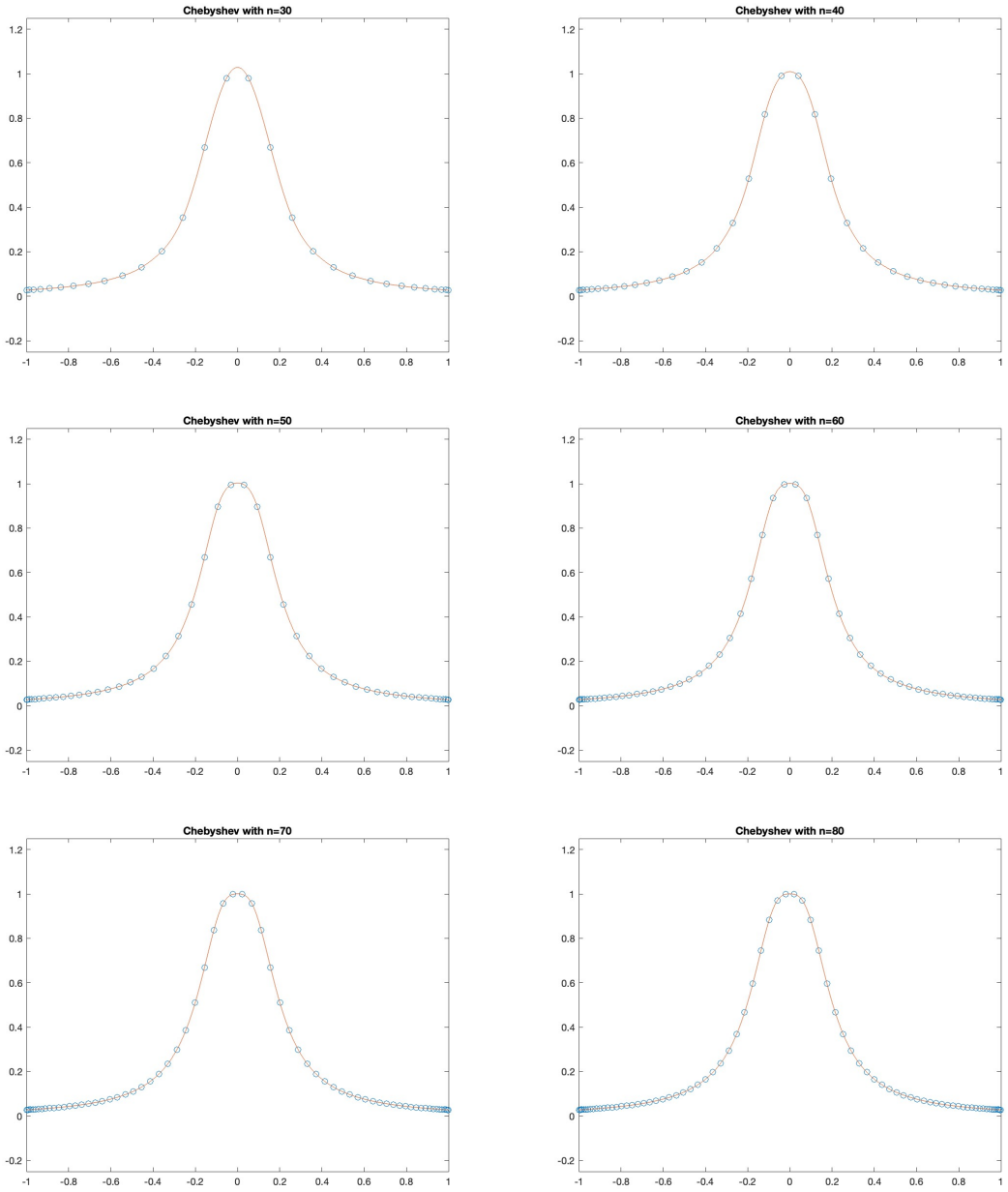

Figure 3. Increasingly fine interpolations on a Chebyshev grid.

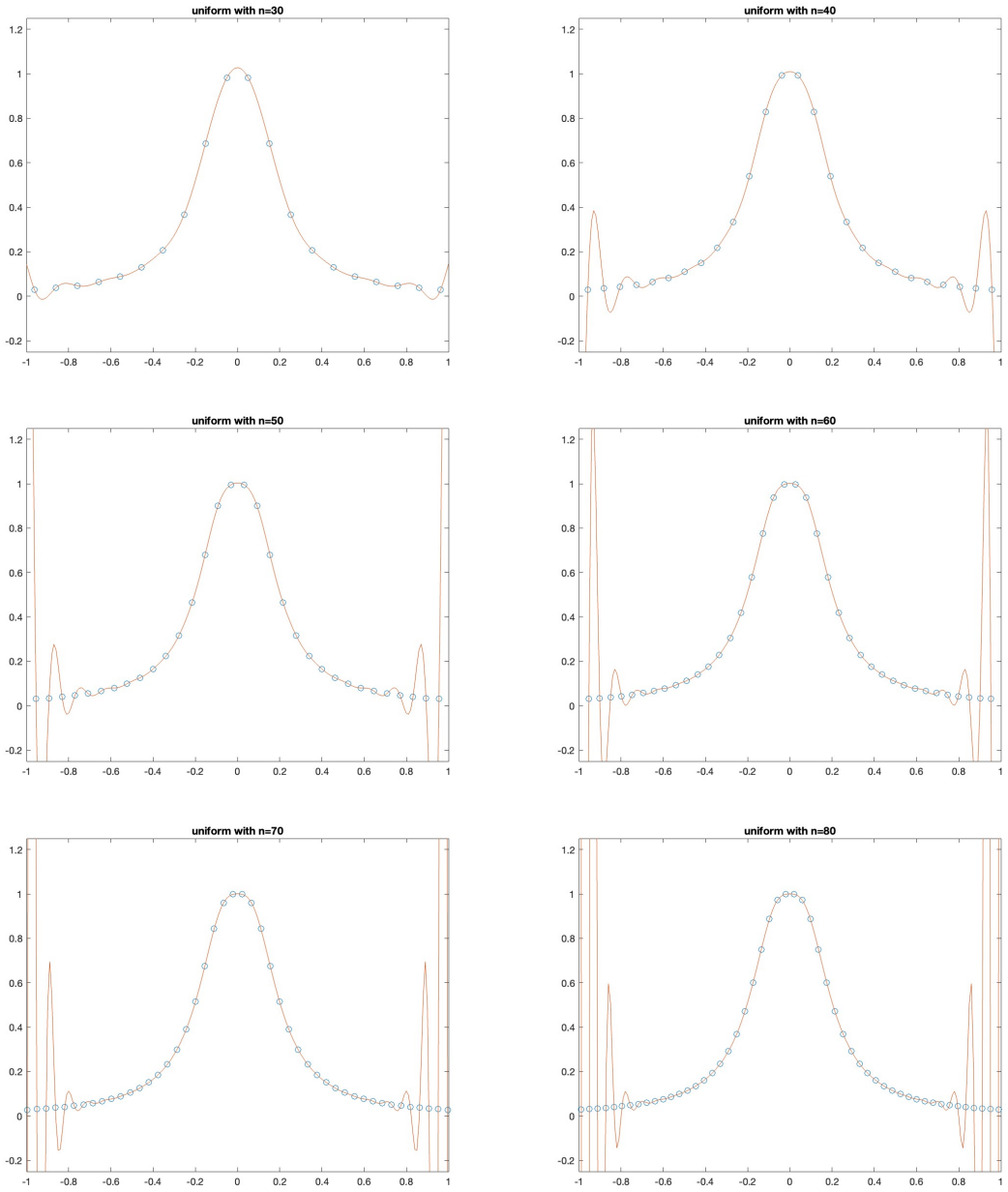

Figure 4. Increasingly fine interpolations on a uniform grid.

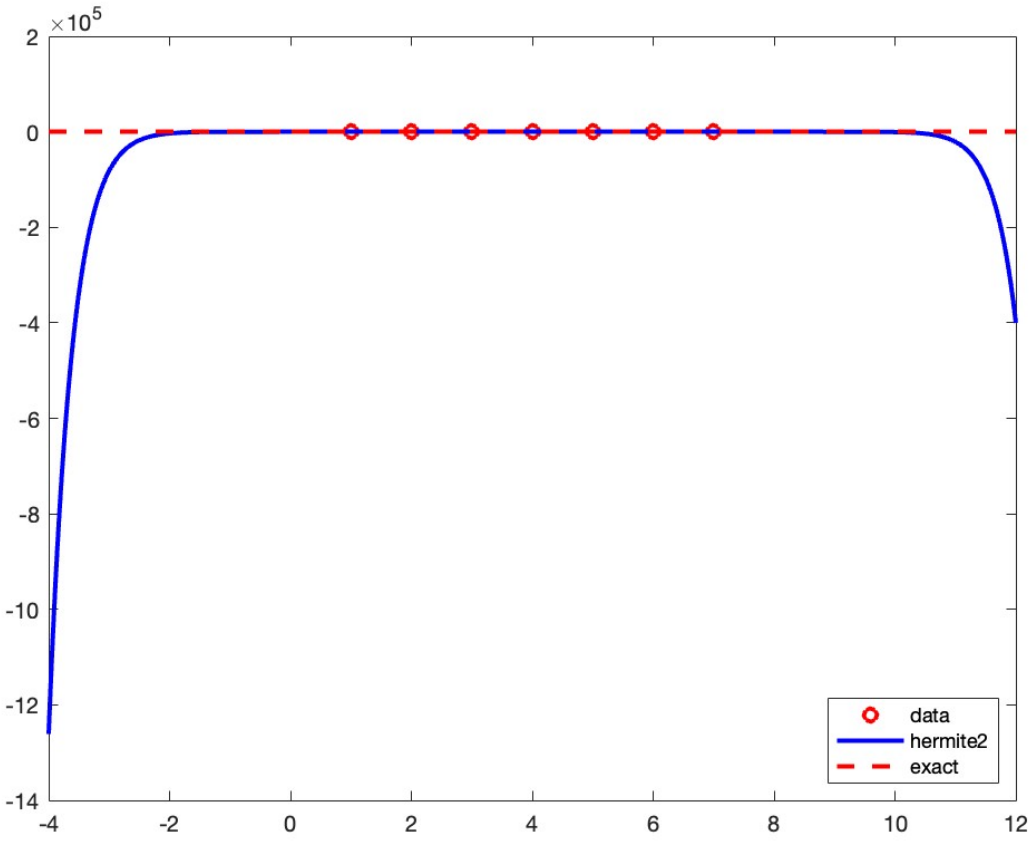

Figure 5. Hermit interpolation of $\ln|x|\cos(x)$ and the original function.

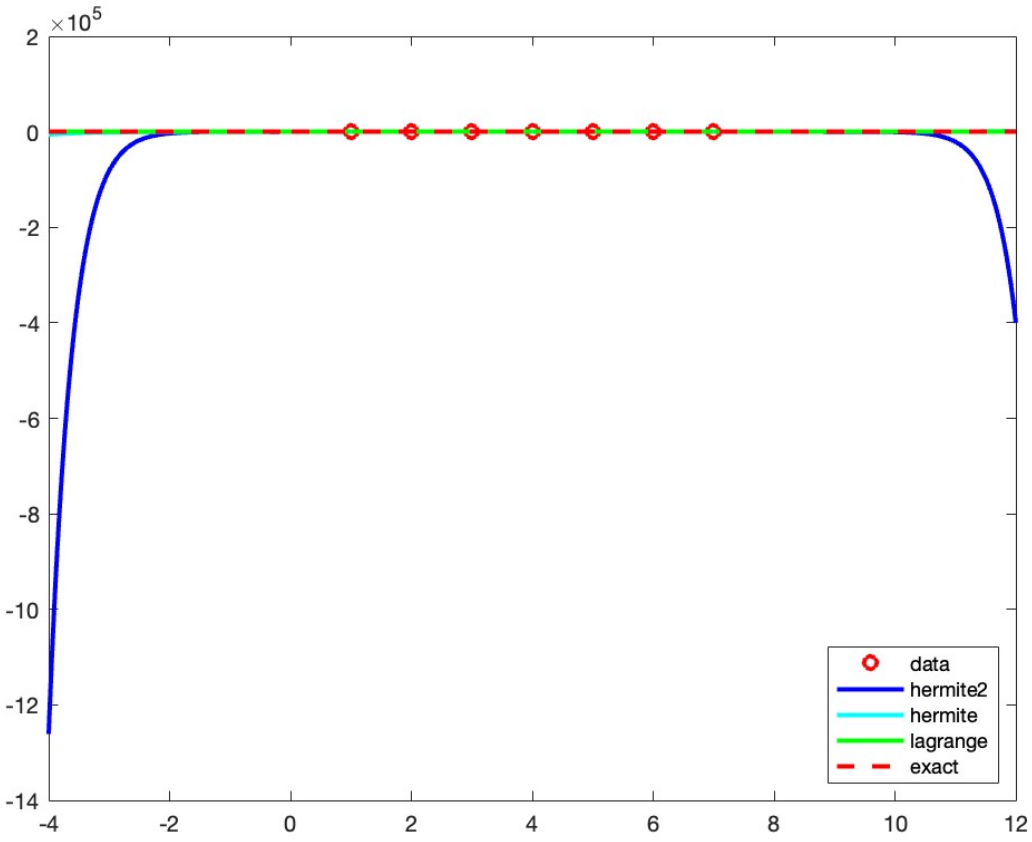

Figure 6. This plot contains all three interpolating polynomials together with the function itself on the interval $[-4, 12]$.

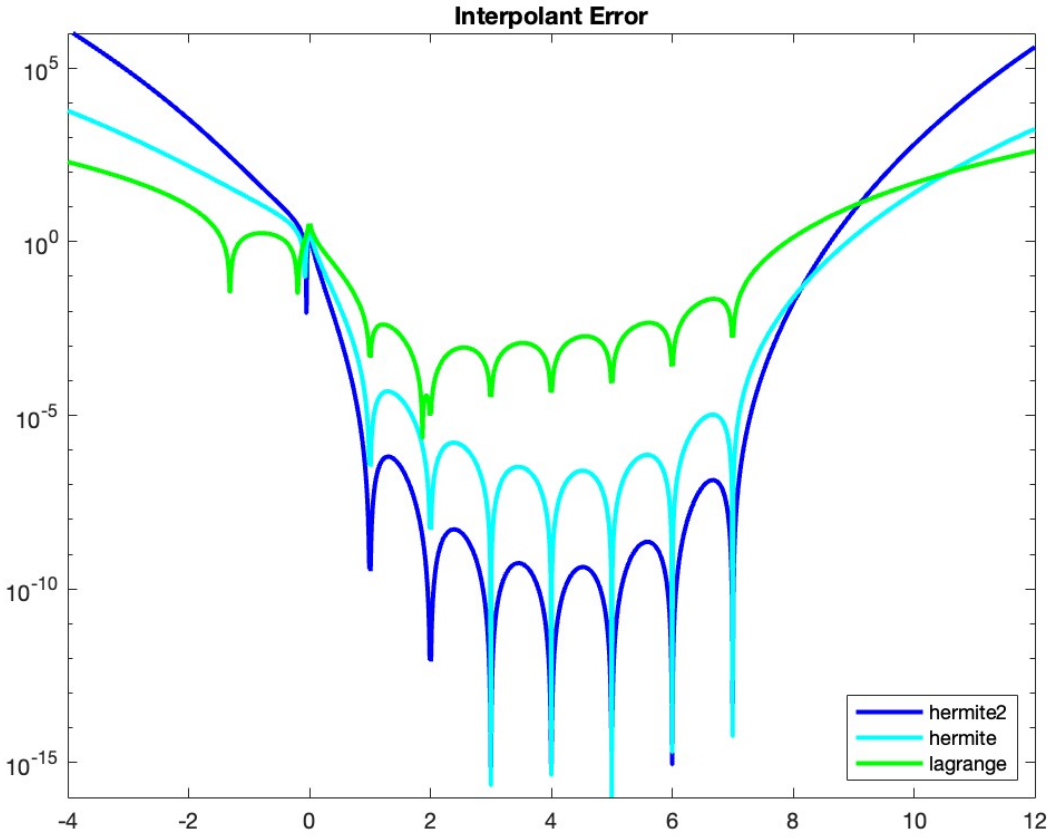

Figure 7. Absolute errors of each of the interplants.

MATLAB Code

Program 1. interp.m.

```
clear all

x=1:7;
y=[-4  -10  -34  -106  152  1046 -1270];
u=0.9:0.01:7.1;


% coefficients of monomials
A=vander(x);
c=A\y';
c=round(c).'
n=length(c);
for i=1:n
    if c(i)>0
       op(i)='+';
    else
       op(i)='-';
    end
end
fgnm=['interpolating polynomial: '];
for i=1:n-2
fgnm=[fgnm,op(i),num2str(abs(c(i))),'x^',num2str(n-i)];
end
fgnm=[fgnm,op(n-1),num2str(abs(c(n-1))),'x'];
fgnm=[fgnm,op(n),num2str(abs(c(n)))];
disp(fgnm)
pause

v=polyval(c,u);
plot(x,y,'o',u,v,'-')
xlabel('x','FontSize',15)
ylabel('y','FontSize',15)
legend('data','polynomial','Location','best')
title(fgnm,'FontSize',18)

pause

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

```
disp('barycentric weights:')
disp(wgtnm)
pause

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

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
```

Program 2. dvi.m.

```
x=[1  1.25  1.5  1.75  2  3];
y=[17.0000   17.9531   19.3750   21.3594   24.0000   29.0458];

w=0:0.1:max(y)+1;
c=newtondif(y,x)
z=newtonint(y,x,w);

u=0:0.01:4;
d=newtondif(x,y)
v=newtonint(x,y,u);

plot(z,w,'-b',u,v,'-g',x,y,'ro')
xlabel('x','FontSize',15)
ylabel('y','FontSize',15)
title('Comparison of Interpolants','FontSize',18)
legend('inverse','direct','Location','NorthWest')
axis([0 4 5 45])
```

Program 3. chebygrid.m.

```
f=@(x)(1+abs(6*x).^3).^(-2/3);

```

```
'Chebyshev grid'


kk=1:30;
x=cos((2.*kk-1).*pi./60);
y=f(x);
u=-1:0.01:1;
v=baryctrint(x,y,u);
plot(x,y,'o',u,v,'-')
title(['Chebyshev with n=',num2str(max(kk))])
axis([-1 1 -.25 1.25])

pause

kk=1:40;
x=cos((2.*kk-1).*pi./80);
y=f(x);
u=-1:0.01:1;
v=baryctrint(x,y,u);
plot(x,y,'o',u,v,'-')
title(['Chebyshev with n=',num2str(max(kk))])
axis([-1 1 -.25 1.25])

pause

kk=1:50;
x=cos((2.*kk-1).*pi./100);
y=f(x);
u=-1:0.01:1;
v=baryctrint(x,y,u);
plot(x,y,'o',u,v,'-')
title(['Chebyshev with n=',num2str(max(kk))])
axis([-1 1 -.25 1.25])

pause

kk=1:60;
x=cos((2.*kk-1).*pi./120);
y=f(x);
u=-1:0.01:1;
v=baryctrint(x,y,u);
plot(x,y,'o',u,v,'-')
title(['Chebyshev with n=',num2str(max(kk))])
axis([-1 1 -.25 1.25])

pause

kk=1:70;
x=cos((2.*kk-1).*pi./140);
y=f(x);
u=-1:0.01:1;
v=baryctrint(x,y,u);
plot(x,y,'o',u,v,'-')
title(['Chebyshev with n=',num2str(max(kk))])
axis([-1 1 -.25 1.25])

pause

kk=1:80;
```

```
x=cos((2.*kk-1).*pi./160);
y=f(x);
u=-1:0.01:1;
v=baryctrint(x,y,u);
plot(x,y,'o',u,v,'-')
title(['Chebyshev␣with␣n=',num2str(max(kk))])
axis([-1 1 -.25 1.25])
```

Program 4. unigrid.m.

```
'uniform␣grid'

kk=1:30;
x=(2.*kk-31).*pi./62;
y=f(x);
u=-1:0.01:1;
v=baryctrint(x,y,u);
plot(x,y,'o',u,v,'-')
title(['uniform␣with␣n=',num2str(max(kk))])
axis([-1 1 -.25 1.25])

pause

kk=1:40;
x=(2.*kk-41).*pi./82;
y=f(x);
u=-1:0.01:1;
v=baryctrint(x,y,u);
plot(x,y,'o',u,v,'-')
title(['uniform␣with␣n=',num2str(max(kk))])
axis([-1 1 -.25 1.25])

pause

kk=1:50;
x=(2.*kk-51).*pi./102;
y=f(x);
u=-1:0.01:1;
v=baryctrint(x,y,u);
plot(x,y,'o',u,v,'-')
title(['uniform␣with␣n=',num2str(max(kk))])
axis([-1 1 -.25 1.25])

pause

kk=1:60;
x=(2.*kk-61).*pi./122;
y=f(x);
u=-1:0.01:1;
v=baryctrint(x,y,u);
plot(x,y,'o',u,v,'-')
title(['uniform␣with␣n=',num2str(max(kk))])
axis([-1 1 -.25 1.25])

pause

kk=1:70;
```

```
x=(2.*kk-71).*pi./142;
y=f(x);
u=-1:0.01:1;
v=baryctrint(x,y,u);
plot(x,y,'o',u,v,'-')
title(['uniform␣with␣n=',num2str(max(kk))])
axis([-1 1 -.25 1.25])

pause

kk=1:80;
x=(2.*kk-81).*pi./162;
y=f(x);
u=-1:0.01:1;
v=baryctrint(x,y,u);
plot(x,y,'o',u,v,'-')
title(['uniform␣with␣n=',num2str(max(kk))])
axis([-1 1 -.25 1.25])

pause

'true␣function␣and␣interpolant␣on␣Chebyshev␣grid'

kk=1:80;
x=cos((2.*kk-1).*pi./160);
y=f(x);
u=-1:0.01:1;
v=baryctrint(x,y,u);
plot(x,y,'o',u,v,'-')
title(['Chebyschev␣with␣n=',num2str(max(kk))])
axis([-1 1 -.25 1.25])
```

Program 5. hermite2dif.m.

```
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
```

```
24
25
26 for i=3:nn-1
27    for j=nn:-1:i+1
28       d(j)=(d(j)-d(j-1))/(xx(j)-xx(j-i));
29    end
30 end
```

Program 6. hermite2int.m.

```
1  function v=hermite2int(x,y,yp,ypp,u)
2
3  tic
4
5  n=length(x);
6  nn=3*n;
7
8  d=hermite2dif(x,y,yp,ypp);
9
10 for i=1:n
11   xx(3*i-2)=x(i);
12   xx(3*i-1)=x(i);
13   xx(3*i)=x(i);
14 end
15
16 v=d(nn).*ones(size(u));
17
18 for k=nn:-1:2
19   v=d(k-1)+(u-xx(k-1)).*v;
20 end
21
22 toc
```

Program 7.

```
1  x=1:1:7;
2
3  y=log(abs(x)).*cos(x);
4  yp=cos(x)./x-log(abs(x)).*sin(x);
5  ypp=-cos(x)./x.^2-2*sin(x)./x-log(abs(x)).*cos(x);
6
7  u=-4:0.01:12;
8  v=hermite2int(x,y,yp,ypp,u);
9  t=log(abs(u)).*cos(u);
10 plot(x,y,'ro',u,v,'b-',u,t,'r--','LineWidth',2)
11 legend('data','hermite2','exact','Location','SouthEast')
12 pause
13
14 z=baryctrint(x,y,u);
15 w=hermiteint(x,y,yp,u);
16 figure;
17 plot(x,y,'ro',u,v,'b-',u,w,'c-',u,z,'g-',u,t,'r--','LineWidth',2)
18 legend('data','hermite2','hermite','lagrange','exact','Location','SouthEast')
19 pause
20
21 figure; semilogy(u,abs(v-t),'b-',u,abs(w-t),'c-',u,abs(z-t),'g-','LineWidth',2)
```

```
22 title('Interpolant␣Error')
23 legend('hermite2','hermite','lagrange','Location','SouthEast')
24 axis([-4 12 1e-16 1e6])
```