EN.553.481/681 Numerical Analysis – Homework 3 Solutions

Problem 1. (a) Define $f(x) := \det \mathbf{V}[x_0, \ldots, x_{n-1}, x]$ and note that if $x = x_i$ 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, 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, $f(x) = \alpha \prod_{i=1}^{n-1}(x - x_i)$ for some α. Note α 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:

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

Thus $\alpha = \det \mathbf{V}[x_0, \ldots, x_{n-1}]$ 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 $3.112310999999999 \times 10^{-5}$, barycentric representation in $2.259429999999981 \times 10^{-6}$, and Newton divided differences in $1.455899999999852 \times 10^{-6}$. The Newton divided difference method is fastest. The monomial basis is roughly 21 times slower, and barycentric representation is roughly 1.55 times slower.

(a) We find the interpolating polynomial to be $5x^6 - 103x^5 + 837x^4 - 3435x^3 + 7443x^2 - 7973x + 3209$.

(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 $d = (-17, -12, -7, -10, -8, 2, 5)$.

Problem 3. (a) Since

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

note that

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

It's immediate that

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

(b) Fix nodes $x_0,\dots,x_n$ 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, which would admit a functional inverse, which is consistent.

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

$$d\approx(17.0, 4.46, 5.76, 2.49, 0.2316, -0.014966, -3.48)$$

For the inverse method we find $d\approx(1.0, 2.2398\times10^{-1}, -3.3938\times10^{-2}, 4.892\times10^{-3}, -5.3345\times10^{-4}, 4.27527\times10^{-5}, 6.7592779\times10^{-6})$.

(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 $i=0,\dots,6$ 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 $p_2(0)$ multiply f_a^2 to both the numerators and denominators of each term and note $s=r/t$ to write

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

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

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) Implemented in Program 5.

(b) Implemented in Program 6.

(c) Implemented as `ive.m` in Program 7. Plotted in Figure 5. By inspection it is clear that interpolation is more accurate than extrapolation.

(d) The respective errors of Hermite and Lagrangian interpolation are plotted in Figure 6. The method of Hermite is more accurate for interpolation. However, for extrapolation we see in Figure 5 that the Lagrange polynomial is actually closer to the exact answer than Hermite is on the far right, while on the far left the reverse is true. The moral is that more accurate interpolation does not guarantee better extrapolation.

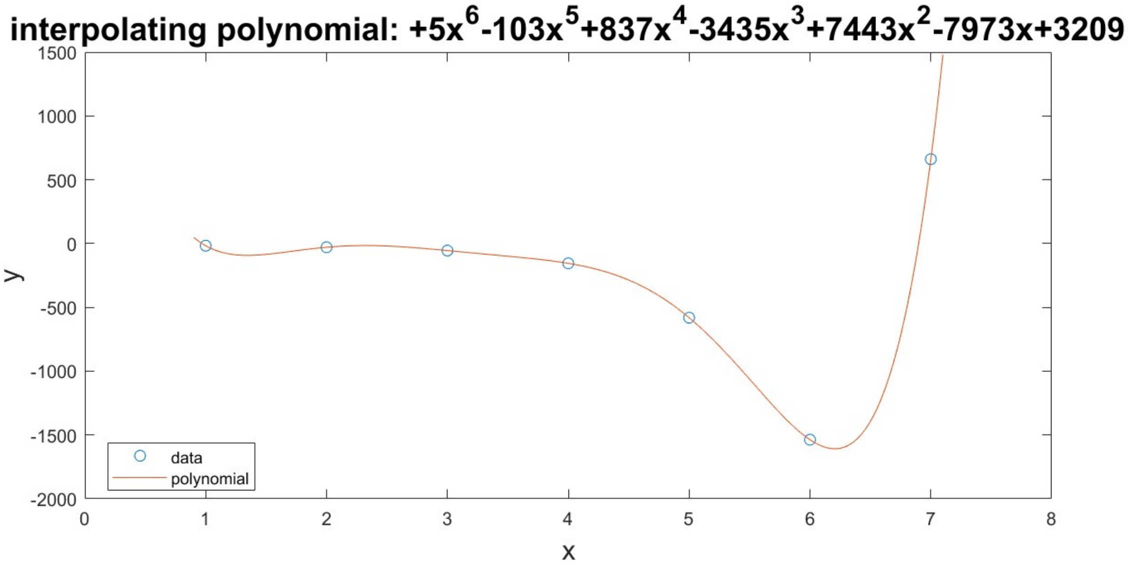

Figure 1. The interpolating polynomial along with the 7 points.

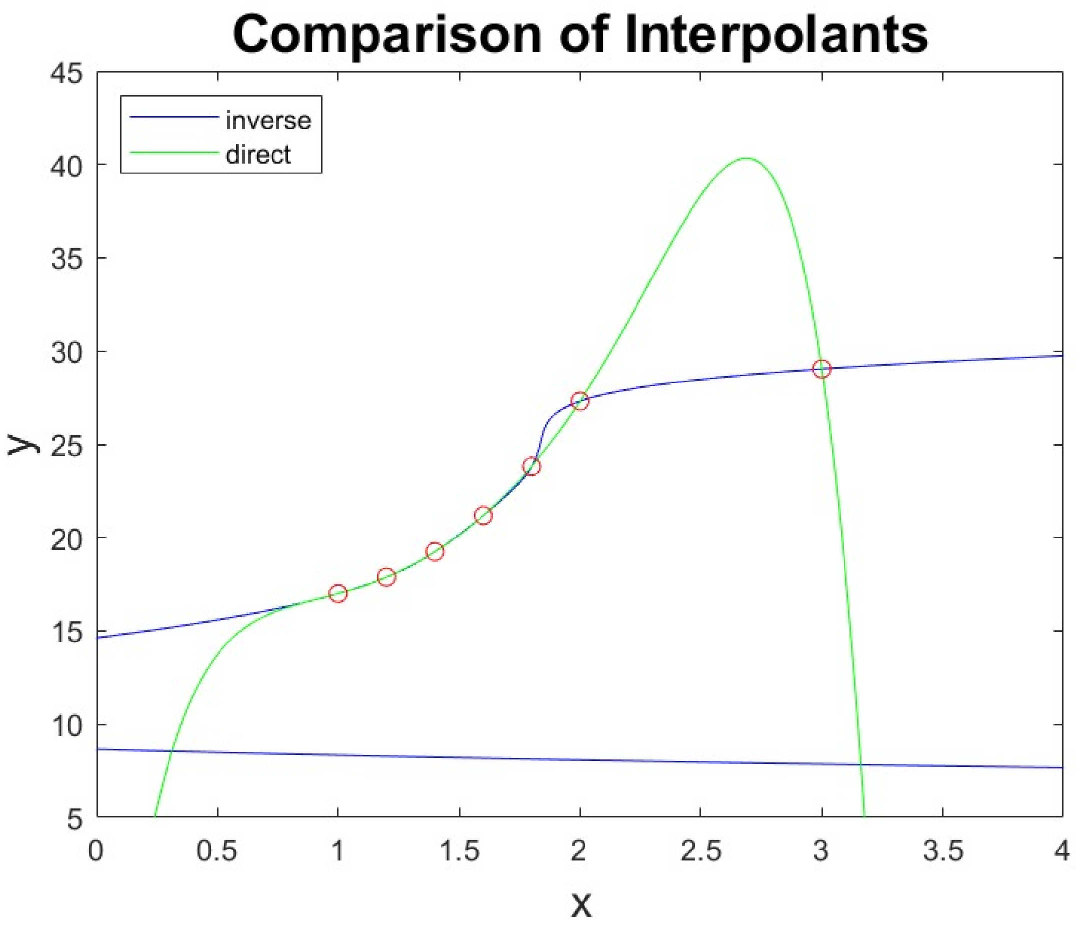

Figure 2. Direct vs. inverse interpolation.

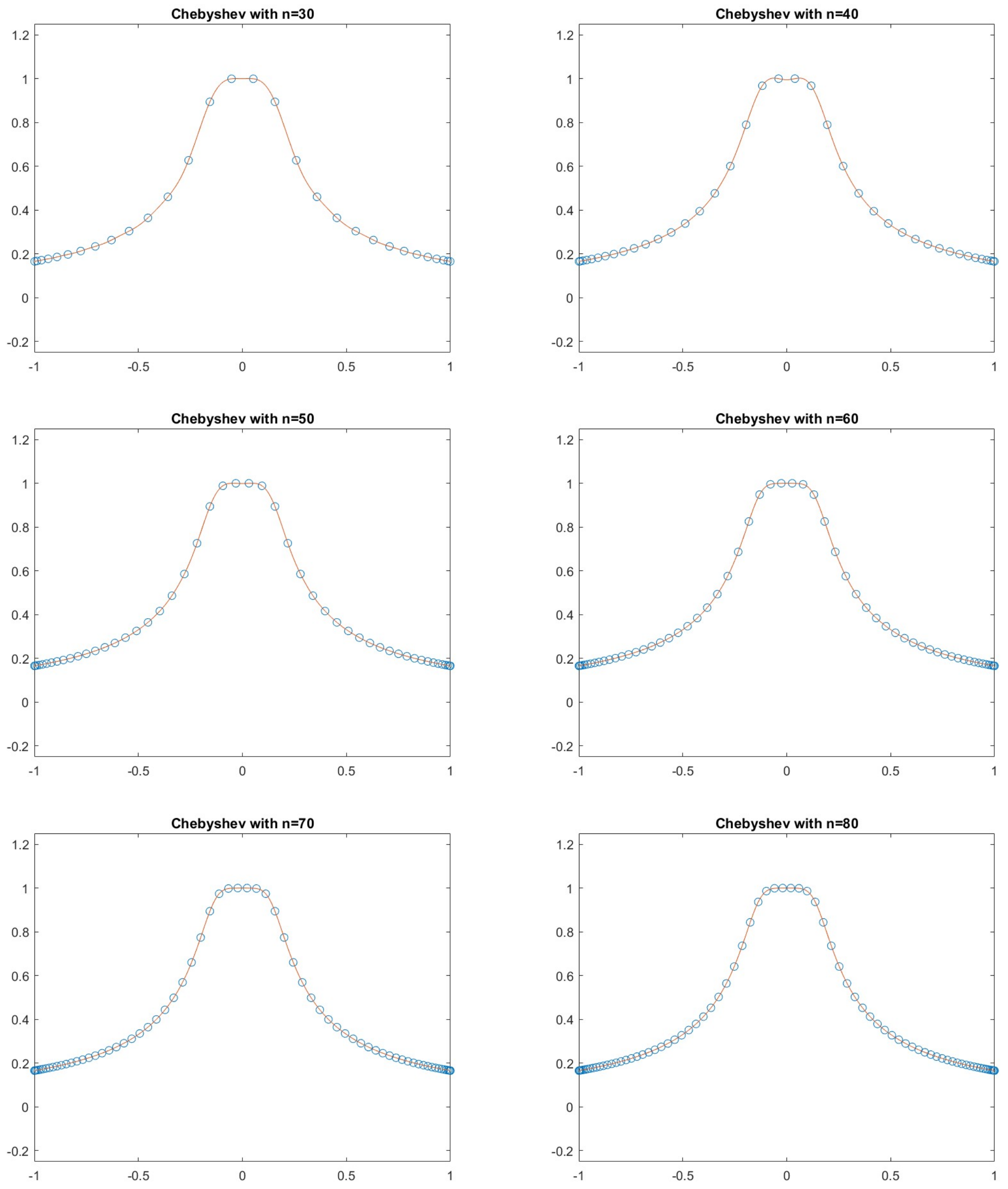

Figure 3. Increasingly fine interpolations on a Chebyshev grid.

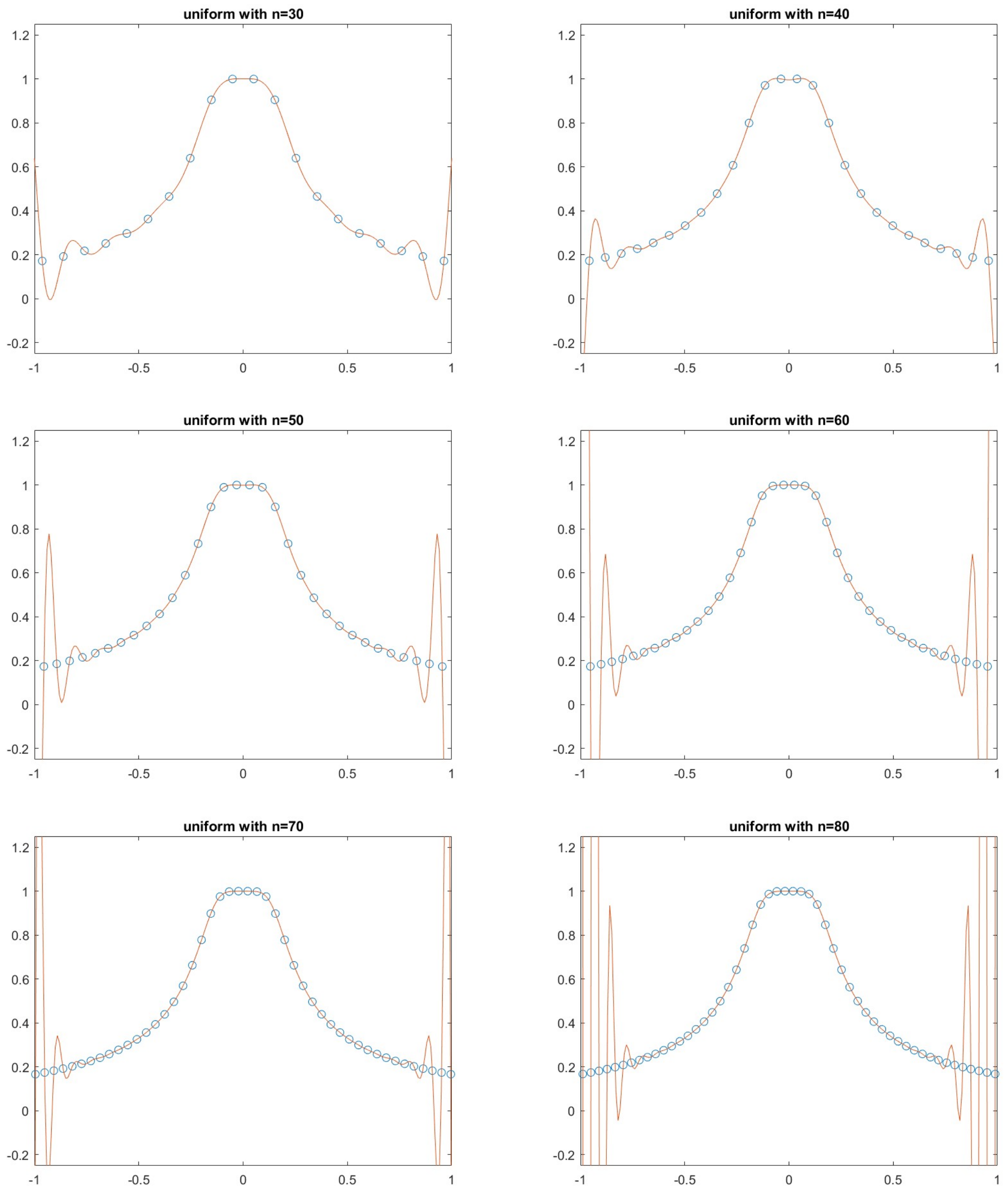

Figure 4. Increasingly fine interpolations on a uniform grid.

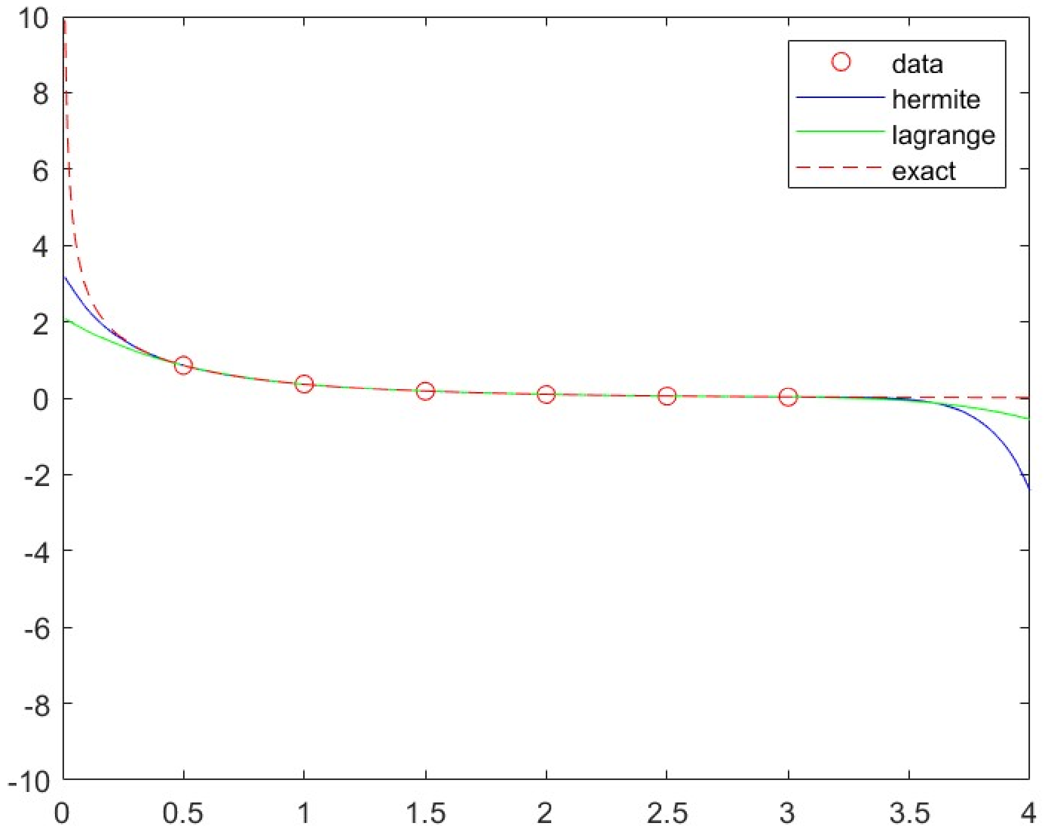

Figure 5. Hermit and Lagrange interpolations of $\exp(-x)/\sqrt{x}$.

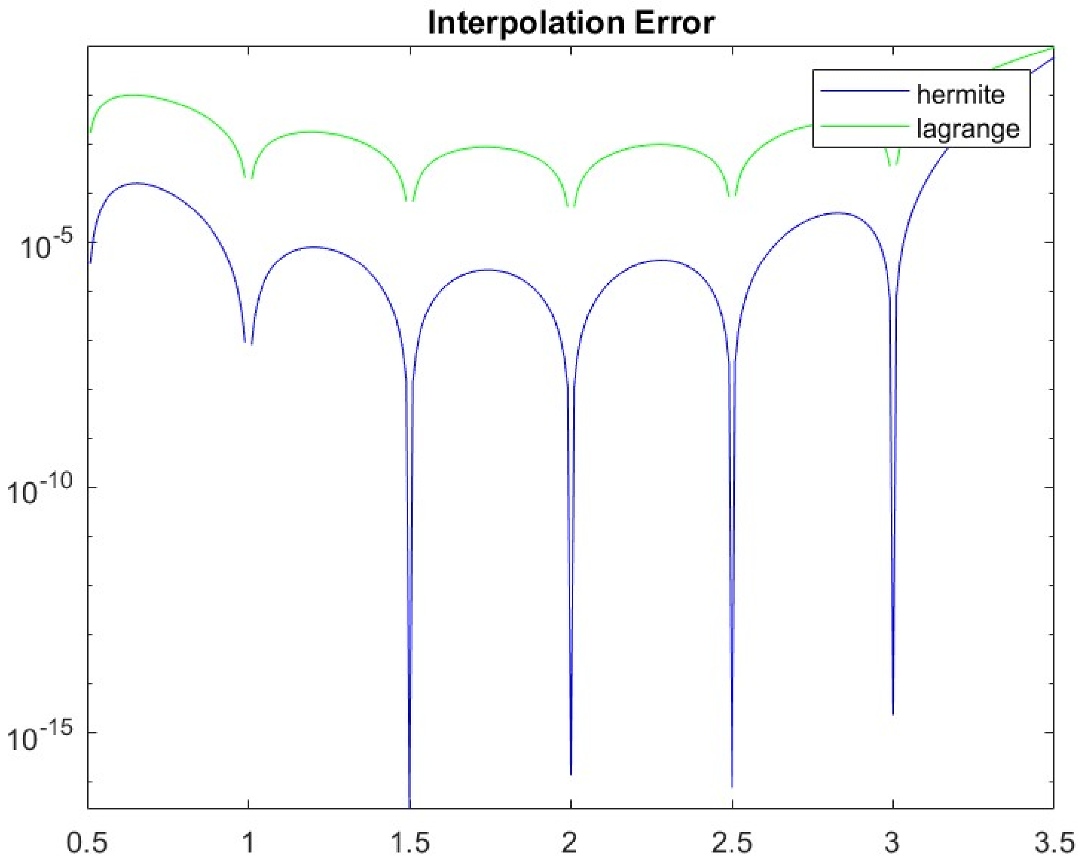

Figure 6. Respective errors of the Hermit and Lagrange interpolations of $\exp(-x)/\sqrt{x}$.

MATLAB Code

Program 1. interp.m.

```
clear all

x=1:7;
y=[ -17 -29   -55   -155   -581   -1537   661];
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  1.2 1.4  1.6  1.8   2  3];
y=[17.0000000  17.8929291  19.2467442  21.1810760  23.8244495  27.3137093  29.0457592];

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`.

```
'Chebyshev␣grid'

```

```
kk=1:30;
x=cos((2.*kk-1).*pi./60);
y=(1+8000.*abs(x).^5).^(-1/5);
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=(1+8000.*abs(x).^5).^(-1/5);
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=(1+8000.*abs(x).^5).^(-1/5);
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=(1+8000.*abs(x).^5).^(-1/5);
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=(1+8000.*abs(x).^5).^(-1/5);
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=(1+8000.*abs(x).^5).^(-1/5);
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=(1+8000.*abs(x).^5).^(-1/5);
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=(1+8000.*abs(x).^5).^(-1/5);
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=(1+8000.*abs(x).^5).^(-1/5);
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=(1+8000.*abs(x).^5).^(-1/5);
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=(1+8000.*abs(x).^5).^(-1/5);
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=(1+8000.*abs(x).^5).^(-1/5);
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])
```

Program 5. hermitedif.m.

```
function d=hermitedif(x,y,yp)

n=length(x);
nn=2*n;

for i=1:n
        xx(2*i-1)=x(i);
        xx(2*i)=x(i);
end

d(1)=y(1);

for j=1:n
        d(2*j)=yp(j);
end

for j=n:-1:2
        d(2*j-1)=(y(j)-y(j-1))/(x(j)-x(j-1));
end

for i=2:nn-1
        for j=nn:-1:i+1
                d(j)=(d(j)-d(j-1))/(xx(j)-xx(j-i));
        end
end
```

Program 6. hermiteint.m.

```
function v=hermiteint(x,y,yp,u)

n=length(x);
nn=2*n;

d=hermitedif(x,y,yp);

for i=1:n
        xx(2*i-1)=x(i);
        xx(2*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
```

Program 7. `ive.m`.

```
x=0.5:0.5:3.0;
y=exp(-x)./sqrt(x);
yp=-(1+1./x/2).*y;
u=0.01:0.01:4.0;
v=hermiteint(x,y,yp,u);
w=exp(-u)./sqrt(u);
plot(x,y,'ro',u,v,'b-',u,w,'r--')
axis([0 4 -10 10])
legend('data','hermite','exact')
pause

z=baryctrint(x,y,u);
figure;
plot(x,y,'ro',u,v,'b-',u,z,'g-',u,w,'r--')
axis([0 4 -10 10])
legend('data','hermite','lagrange','exact')
pause

figure; plot(u,w-v,'b-',u,w-z,'g-')
title('Extrapolation␣Error')
legend('hermite','lagrange')
axis([0 4 0 2])
pause

figure; semilogy(u,abs(w-v),'b-',u,abs(w-z),'g-')
title('Interpolation␣Error')
legend('hermite','lagrange')
axis([0.5 3.5 0 .1])
```