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

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

**Solution:**

(a) Consider the function of \(x_n\) defined by

\[ f(x_n) = \det V[x_0, ..., x_{n-1}, x_n] = \begin{vmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n-1} & x_{n-1}^2 & \cdots & x_{n-1}^n \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{vmatrix}. \]

Expanding the Vandermonde matrix along the last row from right to left, we see clearly that \(f(x_n)\) is a polynomial of degree \(n\) in \(x_n\). Furthermore,

\[ f(x_n) = x_n^n + \ldots \]

so that the leading coefficient is \(\det V[x_0, ..., x_{n-1}]\). Note also when \(x_n = x_i\), the \((i+1)\)th row and the \((n+1)\)th row are identical, and hence \(f(x_i) = 0\). By the Fundamental Theorem of Algebra, \(f(x_n)\) has exactly \(n\) roots \(x_0, ..., x_{n-1}\). Therefore

\[ f(x_n) = c_n \prod_{i=0}^{n-1} (x_n - x_i), \]

where \(c_n\) is the leading coefficient \(\det V[x_0, ..., x_{n-1}]\). Therefore,

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

(b) For \(n = 0\), it is trivial.

For \(n = 1\),

\[ \det V[x_0, x_1] = \begin{vmatrix} 1 & x_0 \\ 1 & x_1 \end{vmatrix} = x_1 - x_0. \]
Now assume $\det V[x_0, \ldots, x_{n-1}] = \prod_{0 \leq i < j \leq n-1} (x_j - x_i)$. Apply the result from (a):

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

$$= \prod_{0 \leq i < j \leq n-1} (x_j - x_i) \prod_{i=0}^{n-1} (x_n - x_i) = \prod_{0 \leq i < j \leq n} (x_j - x_i),$$

since the only factors with $j = n$ in the last product have $i = 0, 1, \ldots, n - 1$. 
Problem 2 Consider the following seven points:

\[
(x_1, y_1) = (1, 4), (x_2, y_2) = (2, 9), (x_3, y_3) = (3, -10), (x_4, y_4) = (4, 1),
(x_5, y_5) = (5, 264), (x_6, y_6) = (6, -871), (x_7, y_7) = (7, -326)
\]

For this data, find the 6th-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 seven points along with the interpolating polynomial over the interval [0.9, 5.1].

a) Monomial basis: We use the following Matlab code to find the desired interpolation:

```matlab
x=1:7;
y=[4 9 -10 1 264 -871 -326];
u=0.9:0.01:5.1;

% 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))),'xˆ',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)
```

We get the following polynomial as a result:

\[
12x^6 - 269x^5 + 2362x^4 - 10326x^3 + 23492x^2 - 26036x + 10769
\]

b) Barycentric Lagrange Form: We use the following similar Matlab code to get the barycentric form

```matlab
w=baryctrwt(x);
w=abs(round(1./w));
wgtnm=['w=['
for k=1:n-1
    if w(k)>0
        wgttm=[wgttm,'1/','num2str(ww(k)),','']
    else
        wgttm=[wgttm,'-1/','num2str(ww(k)),','']
    end
end
wgtnm=[wgttm,'num2str(abs(c(n-1)))','x']
wgtnm=[wgtnm,'num2str(abs(c(n)))'];
disp(wgtnm)
```

We get the following polynomial as a result:

\[
12x^6 - 269x^5 + 2362x^4 - 10326x^3 + 23492x^2 - 26036x + 10769
\]
The resulting barycentric weights are given by:

\[ w = \left[ \frac{1}{720}, -\frac{1}{120}, \frac{1}{48}, -\frac{1}{36}, \frac{1}{48}, -\frac{1}{120}, \frac{1}{720} \right] \]

c) Finally, for the Newton form we use

\[ d = [4, 5, -12, 9, 7, -17, 12] \]

Below is the graph of the interpolating polynomial over the data points. It is produced through Matlab through the following code:

Next, to compute wall clock times, we use the following Matlab script:
We find that, in this example, computation of (a) the monomial basis coefficients took clock time 1.6238e-05, (b) barycentric weights took 1.9426e-06, and (c) Newton divided differences took 1.3426e-06. These times are machine and run dependent, but the time ratios are more objective. We thus find that calculating the coefficients of the monomial basis requires about 12 times more time than calculating the Newton divided differences, and calculating the barycentric weights requires about 1.5 more time. Thus, (a) is the slowest computation, and (b),(c) are in a similar range.
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

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

Solution:

By the stated expression, the \( n \)-point divided differences are given by

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

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

Hence,

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

Note that

\[
\frac{f(x_n)}{\prod_{j=1}^{n-1} (x_n - x_j)} = (x_n - x_0) \frac{f(x_n)}{\prod_{j=0}^{n-1} (x_n - x_j)}
\]

and

\[
\frac{f(x_0)}{\prod_{j=1}^{n-1} (x_0 - x_j)} = -(x_n - x_0) \frac{f(x_0)}{\prod_{j=1}^{n-1} (x_0 - x_j)}.
\]

and

\[
\prod_{j=1}^{n} (x_i - x_j) - \prod_{j=0, j \neq i}^{n} (x_i - x_j) = \prod_{j=0, j \neq i}^{n} ((x_i - x_0) - (x_i - x_n))
\]

\[
= (x_n - x_0) \prod_{j=0, j \neq i}^{n} (x_i - x_j).
\]

Thus,

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

\[
= \sum_{i=0}^{n} \frac{f(x_i)}{\prod_{j=0, j \neq i}^{n} (x_i - x_j)} = \sum_{i=0}^{n} \Psi'_n(x_i).
\]
Problem 4:

(a) Let $x_0, ..., x_n$ be distinct real points, and consider the following interpolation problem. Choose a function

$$q_n(x) = \sum_{j=0}^{n} c_j e^{jx}$$

such that $q_n(x_i) = y_i$, $i = 0, 1, ..., n$. with the $\{y_i\}$ given data. Show there is a unique choice of $c_0, ..., c_n$.

Hint: The problem can be reduced to that of ordinary polynomial interpolation.

(b) The interpolation functions by exponential polynomials in part (a) can have quite different properties than polynomial interpolants. We consider here their use for interpolating and extrapolating data which are known a priori to be positive and increasing. Suppose that an explosive chemical reaction occurs, with the density $n(t)$ of the main chemical product measured at six times $t$ between 1 and 2:

$$(t_1, n_1) = (1, 108.657), \quad (t_2, n_2) = (1.2, 263.914), \quad (t_3, n_3) = (1.4, 662.291),$$

$$(t_4, n_4) = (1.6, 1701.177), \quad (t_5, n_5) = (1.8, 4441.883), \quad (t_6, n_6) = (2.0, 11731.820).$$

Use the idea of your proof in part (a) to determine the interpolating function of the form $q_5(t) = \sum_{j=0}^{5} c_j e^{j t}$ for $t_i = 1 + (0.2)i$ and data $n_i$ with $i = 0, ..., 5$.

(c) Compare this with the interpolating polynomial $p_5(t)$ for the same data, over the larger interval $0 < t < 3$. Comment on the relative accuracy of $q_5(t)$ and $p_5(t)$ both for interpolation within the fitting interval $1 < t < 2$ and for extrapolation to the larger interval $0 < t < 3$. In particular, which interpolant provides a reasonable estimate of the density of the chemical product at time $t = 0$?

Solution:

(a) We can reformulate the interpolating problem by setting $\xi_i = e^{x_i}$, $i = 0,1,\ldots,n$. It is then equivalent to find the interpolating polynomial

$$\Pi_n(\xi) = \sum_{j=0}^{n} c_j \xi^j$$

such that

$$\Pi_n(\xi_i) = y_i, \quad i = 0, 1, \ldots, n.$$ 

To see this, notice that

$$q_n(x) = \Pi_n(e^x).$$

Then

$$q_n(x_i) = \Pi_n(e^{x_i}) = \Pi_n(\xi_i) = y_i.$$ 

Since we know that a (unique) polynomial interpolating function of appropriate degree always exists, we know a solution exists. We now verify that the solution is unique. Suppose we find a distinct interpolating function

$$\tilde{q}_n(x) = \sum_{j=0}^{n} \tilde{c}_j e^{jx}$$

with $\tilde{c}_j \neq c_j$ for at least one $j$. Then if we define

$$\tilde{\Pi}_n(\xi) = \tilde{q}_n(\ln \xi),$$
we see that

\[ \hat{\Pi}_n(\xi_i) = y_i, \]

which contradicts the fact that \( \Pi_n \) is the unique polynomial interpolant.

(b): We can use the following Matlab script to interpolate the data with exponential polynomials:

```matlab
1 m=5;
2 x=1+(0:m)/m;
3 xi=exp(x);
4 y=[108.657 263.914 662.291 1701.177 4441.883 11731.820];
5 u=0:0.02:3;
6 ui=exp(u);
7 vi=vanderint(xi,y,ui);
8 A=vander(xi);
9 disp(' ')
10 disp('coefficients of q_5:')
11 c=A\y.
```

The coefficients for \( q_5(x) \) obtained are \( c_0 = 0.4900, c_1 = 0.1673, c_2 = 0.9857, c_3 = 0.6801, c_4 = 0.5772, c_5 = 0.4014 \).

(c) Now we find the interpolating polynomial and produce plots of the interpolants and the data as follows:

```matlab
1 v=newtonint(x,y,u);
2 plot(u,vi,'-b',u,v,'-g',x,y,'ro')
3 xlabel('x','FontSize',15)
4 ylabel('y','FontSize',15)
5 title('Comparison of Interpolants','FontSize',18)
6 legend('exponential','polynomial','Location','NorthWest')
7 axis([0 3 -1e5 2e6])
8 figure;
9 semilogy(u,vi,'-b',u,v,'-g',x,y,'ro')
10 xlabel('x','FontSize',15)
11 ylabel('y','FontSize',15)
12 title('Comparison of Interpolants','FontSize',18)
13 legend('exponential','polynomial','Location','NorthWest')
14 axis([0 3 0.1 2e6])
```

We obtain the following plot over the \( 1 < t < 2 \):
Where the exponential interpolation is given by the blue curve, and polynomial interpolation given by the green curve. Within $1 < t < 2$, the curves are relatively similar, except perhaps between the first two data points. The polynomial seems to 'wobble' a bit more than the exponential, and we can see that as soon as $t < 1$, the polynomial takes a sharp turn downward which does not seem to agree well with the distribution of the data points. For the larger interval $0 < t < 3$, we obtain this graph:

Here we can see how the exponential and regular polynomial interpolations separate from another. For larger $t$, the exponential grows faster than the polynomial as we would expect. As $t$ gets close to 0, we can see that the polynomial actually becomes negative, whereas the exponential always stays positive. The polynomial interpolation gives us an impossible
“estimate” of the density at zero, since density cannot be negative. Thus, the exponential interpolation is clearly the better model in this case! Note that, in general, an exponential polynomial with positive coefficients is everywhere positive, but an ordinary polynomial can become negative at some points even if all of its coefficients are positive.
Problem 5. Another example of the Runge phenomenon is provided by the function

\[ f(x) = \begin{cases} 
\exp \left( 1 - \frac{1}{|x|^2/3} \right) & x \neq 0 \\
0 & x = 0
\end{cases} \]

which is continuous on the interval \([-1, 1]\).

(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? Plot the errors in the interpolating polynomial to test your conjecture.

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

Hint: Use the barycentric Lagrange method in this problem. The parameters of the interpolating polynomial are more stable than for competing methods.

Solution: We can create the Chebyshev grid and plot the corresponding interpolating polynomial and errors with the following code:

```matlab
1 f=@(x) exp(1-1./abs(x).^2/3))
2 u=-1:0.01:1;
3 uu=u;
4 w=f(u);
5 'Chebyshev grid'
6 n=12;
7 for ii=1:n
8    kk=1:10*ii;
9    x=cos((2.*kk-1).*pi./20/ii);
10   y=f(x);
11   v=baryctrint(x,y,u);
12 end
13 figure
14 subplot(2,1,1)
15 plot(x,y,'o',u,v,'-b',u,w,'-r')
16 axis([-1 1 0 1])
17 title(['Interpolant on Chebyshev grid, n=',num2str(10*ii)])
18 subplot(2,1,2)
19 plot(u,v-w,'-b')
20 title(['Error on Chebyshev grid, n=',num2str(10*ii)])
21 pause
22 end
```

For \(n = 10, 20, \ldots, 110, 120\), we obtain the following plots for interpolation and errors with the Chebyshev grid:
We can see that the error becomes smaller and smaller as $n$ increases (notice the scaling on the y-axis for the errors). It would be reasonable to conjecture that the polynomials converge uniformly to on this interval as we let $n$ go to infinity since we see the errors becoming smaller and smaller.
To compare this to using a uniform grid, we use the following code:

```matlab
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);
    figure
    subplot(2,1,1)
    plot(xx,yy,'o',uu,vv,'-b',u,w,'-r')
    axis([-1 1 0 1])
    title(['Interpolant on regular grid, n=',num2str(10*ii)])
    subplot(2,1,2)
    plot(uu,vv-w)
    title(['Error on regular grid, n=',num2str(10*ii)])
    pause
end
```

We get the following results:

![Graphs showing interpolant and error for different n values](image)

We can see that the interpolating polynomial behave wildly as \( n \) increases. It is equal to our function at the sampled points, but has huge swings in between points. We can see that the error grows rapidly as well, up to a magnitude of \( 10^{28} \) in the \( n = 120 \) case. We would conjecture here that the interpolation does not converge everywhere as \( n \) increases, and here we even observe that the interpolation for \( |x| \gtrsim 0.5 \) oscillates wildly between larger positive and negative values as we increase \( n \). This is the Runge phenomenon!