Supplemental 1: Consider these three points: $\{(1,1),(2.5,8),(4,5)\}$. Find the polynomial $P(x)$ of degree 2 which passes through these points. Do this three different ways, by using

(a) the Vandermonde matrix method,

Solution:
By the Vandermonde matrix method we can solve for the coefficients of $P(x)$ by solving the following system,

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 2.5 & 2.5^2 \\ 1 & 4 & 4^2 \end{pmatrix}\begin{pmatrix} c_0 \\ c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 8 \\ 5 \end{pmatrix}$$

Solving with MATLAB we get,
Console:

```
>> x = [1  2.5  4]

x =

    1.0000    2.5000    4.0000

y = [1  8  5]

y =

     1     8     5

V = [x'.^0, x', x'.^2]

V =

    1.0000    1.0000    1.0000
    1.0000    2.5000    6.2500
    1.0000    4.0000   16.0000

c = V\y'

c =

   -9.2222      %c_0
   12.4444      %c_1
   -2.2222      %c_2
```

This gives us that our polynomial is,

$$P(x) = -2.22x^2 + 12.44x - 9.22$$

(b) The Newton form and its triangular matrix method

Solution:
The Newton Form of the interpolation polynomial is,

$$P(x) = c_0 + c_1(x - x_0) + c_2(x - x_0)(x - x_1)$$

Applying $P(x_i) = y_i$ and solving for c_i is the same as solving the lower triangular system

$$\begin{pmatrix} 1 & 0 & 0 \\ 1 & 2.5-1 & 0 \\ 1 & 4-1 & (4-1)(4-2.5) \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 8 \\ 5 \end{pmatrix}$$

Solving with MATLAB we get,
Console:

```
>> x = [1  2.5  4]

x =

    1.0000    2.5000    4.0000

>> y = [ 1  8  5]

y =

     1     8     5

>> N = NewtonMatrix(x)

N =

    1.0000         0         0
    1.0000    1.5000         0
    1.0000    3.0000    4.5000

>> c = N\y'

c =

    1.0000      %c_0
    4.6667      %c_1
   -2.2222      %c_2
```

Substituting our coefficients in the Newton Form,

$$P(x) = 1.0 + 4.66(x-1) - 2.222(x-1)(x-2.5).$$

(c) the Lagrange form.

Solution:
We can easily write the basis functions $\phi(x)$ for the lagrange form polynomial $P(x)$,

$$\phi_0(x) = \frac{(x-2.5)(x-4)}{(1-2.5)(1-4)}$$

$$\phi_1(x) = \frac{(x-1)(x-4)}{(2.5-1)(2.5-4)}$$

$$\phi_2(x) = \frac{(x-1)(x-2.5)}{4-1)(4-2.5)}$$

Following the Lagrange Form for the interpolation polynomial,

$$P(x) = \sum_{k=0}^{n} y_k\phi_k(x),$$

$$P(x) = \frac{(x-2.5)(x-4)}{(1-2.5)(1-4)} + 8\frac{(x-1)(x-4)}{(2.5-1)(2.5-4)} + 5\frac{(x-1)(x-2.5)}{4-1)(4-2.5)},$$

Supplemental 2: Consider the x coordinates $x_0 = 0$, $x_1 = \pi/3$, $x_2 = 2\pi/3$ and $x_3 = \pi$.

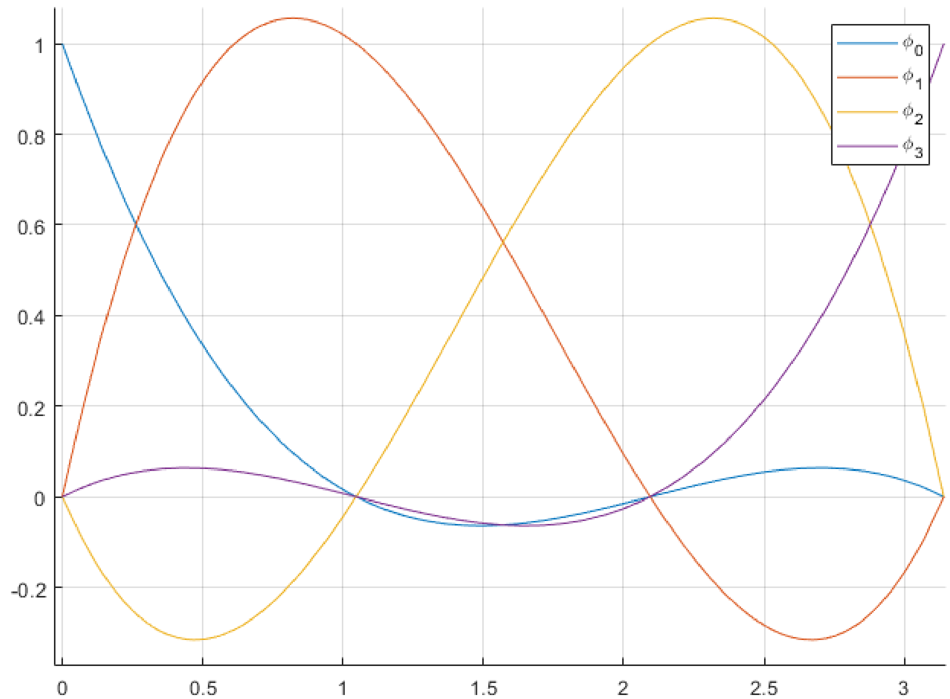
a) Plot the four Lagrange basis functions ϕ_k $k = 0, \ldots, 4$ on a single graph with domain $[0, \pi]$.

Solution:
Console:

```
>> basis_0

 basis_0 =

   function_handle with value:
```

```
    @(x)(x-z(2))/(z(1)-z(2))*
    (x-z(3))/(z(1)-z(3))*
    (x-z(4))/(z(1)-z(4))

>> basis_1

basis_1 =

  function_handle with value:

    @(x)(x-z(3))/(z(2)-z(3))*
    (x-z(4))/(z(2)-z(4))*
    (x-z(1))/(z(2)-z(1))

>> basis_2

basis_2 =

  function_handle with value:

    @(x)(x-z(4))/(z(3)-z(4))*
    (x-z(1))/(z(3)-z(1))*
    (x-z(2))/(z(3)-z(2))

>> basis_3

basis_3 =

  function_handle with value:

    @(x)(x-z(1))/(z(4)-z(1))*
    (x-z(2))/(z(4)-z(2))*
    (x-z(3))/(z(4)-z(3))

>> hold off
>> hold on
fplot(basis_0 ,[0 , pi])
fplot(basis_1 ,[0 , pi])
fplot(basis_2 ,[0 , pi])
fplot(basis_3 ,[0 , pi])
grid on
```

Figure 1: Plot of the Lagrange Basis Functions



b) Plot the four Newton interpolation basis functions ψ_k, each on its own individual graph.

Solution:
Console:

```
>> z

z =

        0    1.0472    2.0944    3.1416

>> N = NewtonMatrix(z)

N =

   1.0000         0         0         0
   1.0000    1.0472         0         0
   1.0000    2.0944    2.1932         0
   1.0000    3.1416    6.5797    6.8903
```

```
>> x = linspace(0 , pi, 300);
>> plot(x,polyval(flipud(N(1,:)'),x))
plot(x,polyval(flipud(N(2,:)'),x))
plot(x,polyval(flipud(N(3,:)'),x))
plot(x,polyval(flipud(N(4,:)'),x))
>>
```

Figure 2: Plot of ψ_0

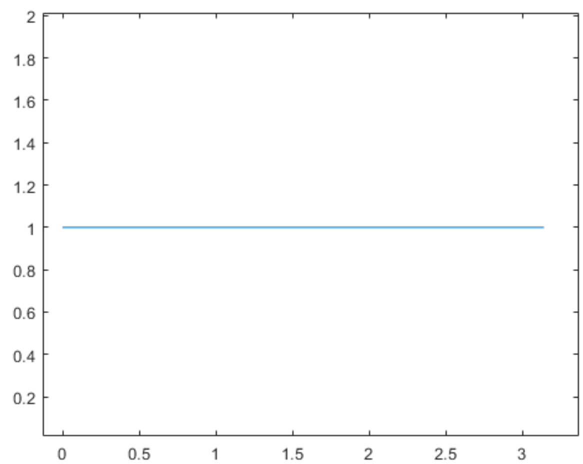


Figure 3: Plot of ψ_1

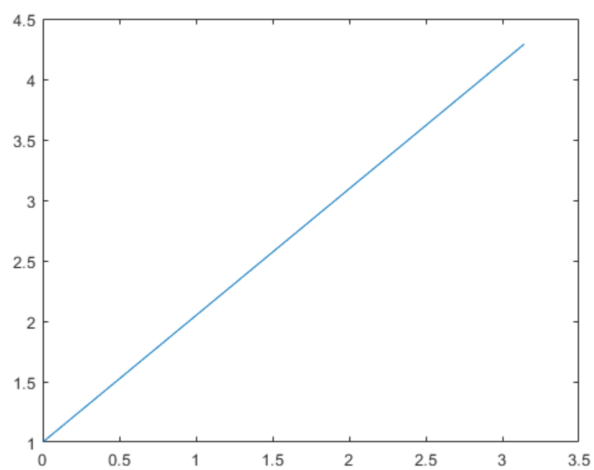


Figure 4: Plot of ψ_2

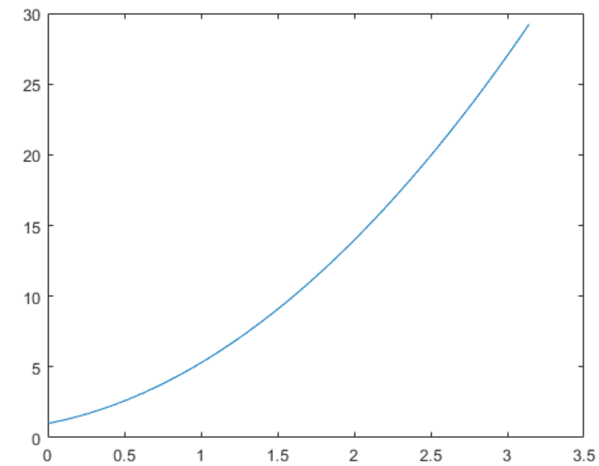
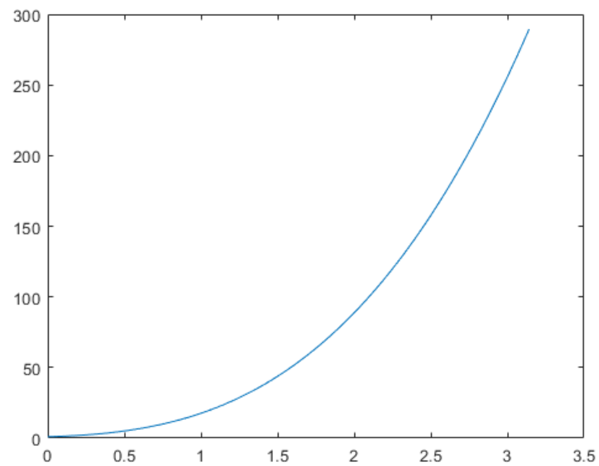


Figure 5: Plot of ψ_3



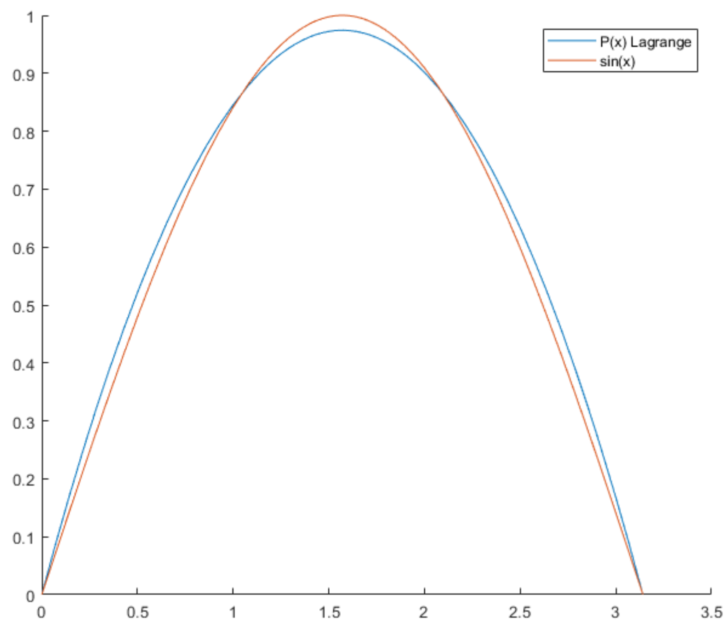
c) Plot the graph of $\sin(x)$ along with its Lagrange interpolant p_{Lag}.

Solution:
Console:

```
>> lagrange

lagrange =

   function_handle with value:
```

```
    @(x)y(1).*basis_0(x)+y(2).*basis_1(x)
    +y(3).*basis_2(x)+y(4).*basis_3(x)

>> sin = @(x) sin(x)

sin =

  function_handle with value:

    @(x)sin(x)

>> hold on
fplot(lagrange, [0,pi], 'displayname', 'P(x) Lagrange')
fplot(sin, [0,pi], 'displayname', 'sin(x)')
lgd = legend
```

Figure 6: Plot of p_{Lag} and $sin(x)$

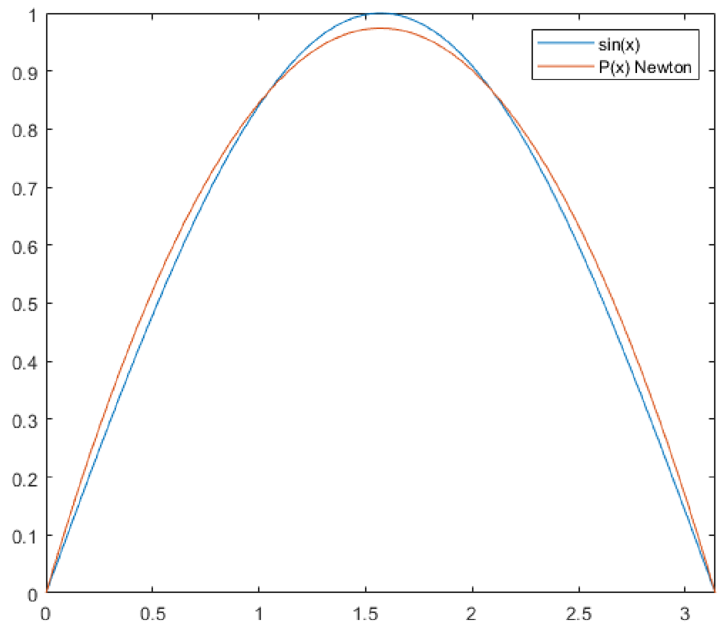


d) Plot the graph of $\sin(x)$ along with its Newton interpolant p_{Newt}.

Solution:
Console:

```
>> N = NewtonMatrix(z)

N =

    1.0000         0         0         0
    1.0000    1.0472         0         0
    1.0000    2.0944    2.1932         0
    1.0000    3.1416    6.5797    6.8903

>> c = N\y'

c =

         0
    0.8270
   -0.3949
    0.0000

>> newton = @(x) c(1) + c(2)*(x - z(1)) +
c(3)*(x - z(1))*(x - z(2)) +
c(4)*(x - z(1))*(x - z(2))*(x - z(3))

newton =

  function_handle with value:

    @(x)c(1)+c(2)*(x-z(1))+
    c(3)*(x-z(1))*(x-z(2))+
    c(4)*(x-z(1))*(x-z(2))*(x-z(3))
```

Figure 7: Plot of p_{Newt} and $sin(x)$



e) What is the relative error of $p_{\text{Lag}}(\pi/4)$?

Solution:
The relative error of our lagrange interpolant at a given point x is calculated by,

$$e_{relative} = |\frac{sin(x) - p_{\text{Lag}}(x)}{sin(x)}|,$$

Console:

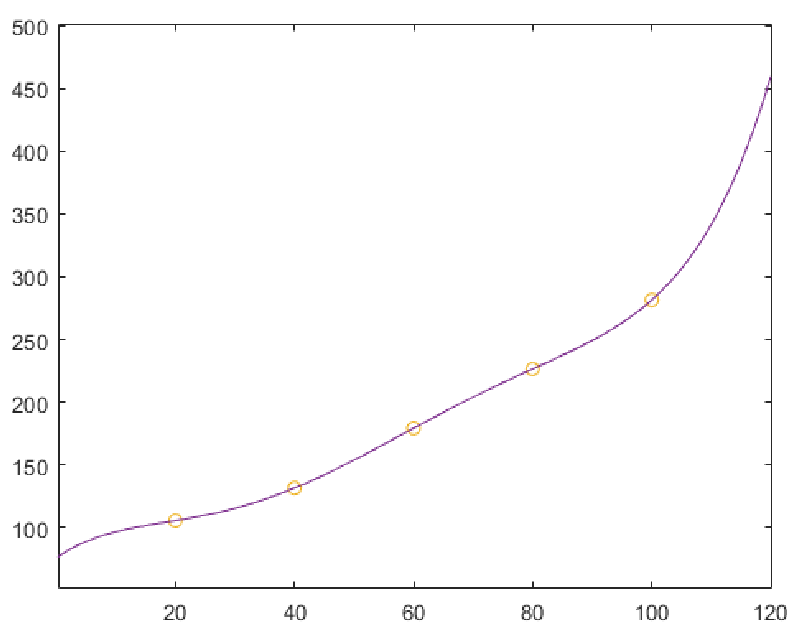
```
>> abs((sin(pi/4) - lagrange(pi/4))/(sin(pi/4)))

ans =

    0.0334
```

Exercise 8.1: 1. Plot population versus years after 1900 using MATLAB by entering the years after 1900 into a vector ...

Solution:
Console:

```
>> x

x =

     0    20    40    60    80   100

>> y

y =

   76.0000  105.7000  131.7000  179.3000  226.5000  281.4000

>> c = polyfit(x,y,5)

c =

    0.0000   -0.0001    0.0048   -0.1711    3.3644   76.0000

>> xx = linspace(0,120,120);
>> hold on
>> plot(x,y,'o')
>> xx = linspace(0,120,120);
```

Figure 8: Census Extrapolation with Vandermonde Interpolation (red)



2. Write down the Lagrange form of the second degree polynomial that interpolates the population in the years 1900, 1920, and 1940.

Solution:
From the definition we know that the Lagrange form of the second degree polynomial is,

$$p_{\text{Lag}} = 76\frac{(x-20)(x-40)}{(0-20)(0-40)} + 105.7\frac{(x-0)(x-40)}{(20-0)(20-40)} + 131.7\frac{(x-0)(x-20)}{(40-0)(40-20)},$$

3. Determine the coefficients of the Newton form of the interpolants of degrees 0, 1, and 2, that interpolate the first one, two, and three data points, respectively. Verify that the second degree polynomial that you construct here is identical to part 2

Solution:
Using MATLAB to find the coefficients for the newton interpolants of degree 0,1,2 for the first three data points of the census.
Console:

```
>> N = NewtonMatrix(x(1:1))

N =

     1

>> c = N\y(1:1)'

c =

    76

>> N = NewtonMatrix(x(1:2))

N =

     1     0
     1    20

>> c = N\y(1:2)'

c =
```

```
    76.0000
     1.4850

>> N = NewtonMatrix(x(1:3))

N =

     1     0     0
     1    20     0
     1    40   800

>> c = N\y(1:3)'

c =

    76.0000
     1.4850
    -0.0046
```

Exercise 8.2: Called Muller's method, fits a quadratic through the three points, $(x_{k2}, (x_{k2}))$, $(x_{k1}, (x_{k1}))$, and $(x_k, f(x_k))$, and takes the root of this quadratic that is closest to x_k as the next approximation x_{k+1}. Write down a formula for this quadratic. Suppose $f(x) = x^3 - 2$, $x_0 = 0$, $x_1 = 1$, and $x_2 = 2$. Find x_3

Solution:
Using a Vandermonde interpolation we can find the formula for the Muller quadratic,
Console:

```
>> f = @(x) x.^3 - 2

f =

  function_handle with value:

    @(x)x.^3-2

>> x = [0 1 2]

x =

     0     1     2

>> y = f(x)
```

```
y =

    -2    -1     6

>> polyfit(x,y,2)

ans =

     3    -2    -2
```

We get a polynomial,

$$P(x) = 2x^2 - 2x - 2.$$

Using the secant method we can rootfind, and we get that the next value is $x_3 = 1.61802$,
Console:

```
>> f  = @(x) 2.*x.^2 - 2.*x - 2

f =

  function_handle with value:

    @(x)2.*x.^2-2.*x-2

>> [root, history] = hw4secant(f, 1,3,.0001,.0001,40)
Inside of f tolerance

root =

     1.618025751072961e+00
```