Polynomial Interpolation

Ling Siu Hong 3200300602

1. How to test

Copy all the file and open makefile.

2. Interpolation Method

a) Newton Interpolation

Using the following code to calculate divided difference.

```
vector<double>getNewton(const vector<double>& x, vector<double>f)
{
    int n = x.size();
    vector<double> c(n), temp(n);
    c[0] = f[0];
    for (int i = 1; i < n; i++)
    {
        for (int j = 0; j < n - i; j++) temp[j] = (f[j + 1] - f[j]) / (x[j + i] - x[j]);
        f = temp;
        c[i] = f[0];
    }
    return c;
}
```

Using the following code to get polynomial.

```
void NewtonPolynomial(vector<double>& c, const vector<double>& x)
{
    int n = x.size();
    cout << c[0];
    for (int i = 1; i < n; i++)
    {
        cout << showpos << c[i];
        for (int j = 0; j < i; j++) cout << "(x" << showpos << -x[j] << ")";
    }
    cout << '\n';
}
```

Using the following code to get the value of function with x.

```
double getvalue(const vector<double>& c, const vector<double>& x, double xval)
{
    int n = c.size();
    double fx = c[0], poly = 1.0;
    for (int i = 1; i < n; i++)
    {
        poly *= (xval - x[i - 1]);
        fx += c[i] * poly;
    }
    return fx;
}
```

b) Hermite Interpolation

```cpp
void HermitePoly(vector<double> x, vector<double> y, vector<double> _x)
{
    vector<vector<double>> f;
    f.resize(x.size() + 1);
    for (int i = 0; i <= x.size(); i++)
    {
        f[i].resize(x.size() + 1);
    }
    for (int i = 0; i < x.size(); i++)
    {
        f[i][0] = y[i];
    }
    for (int i = 1; i < x.size(); i = i + 2)
    {
        f[i][1] = _x[i];
    }
    for (int i = 2; i < x.size(); i = i + 2)
    {
        f[i][1] = (f[i][0] - f[i - 1][0]) / (x[i] - x[i - 2]);
    }
    for (int i = 2; i < x.size(); i++)
    {
        for (int j = i; j < x.size(); j++)
        {
            f[j][i] = (f[j][i - 1] - f[j - 1][i - 1]) / (x[j] - x[j - i]);
        }
    }
    for (int i = 0; i < f.size(); i++)
    {
        cout << showpos << f[i][i];
        for (int j = 0; j < i; j++) cout << "(x" << showpos << -x[j] << ")";
    }
    cout << '\n';
```

The output will directly be a polynomial.

3. Result

- Question B

The Newton's Interpolation results.

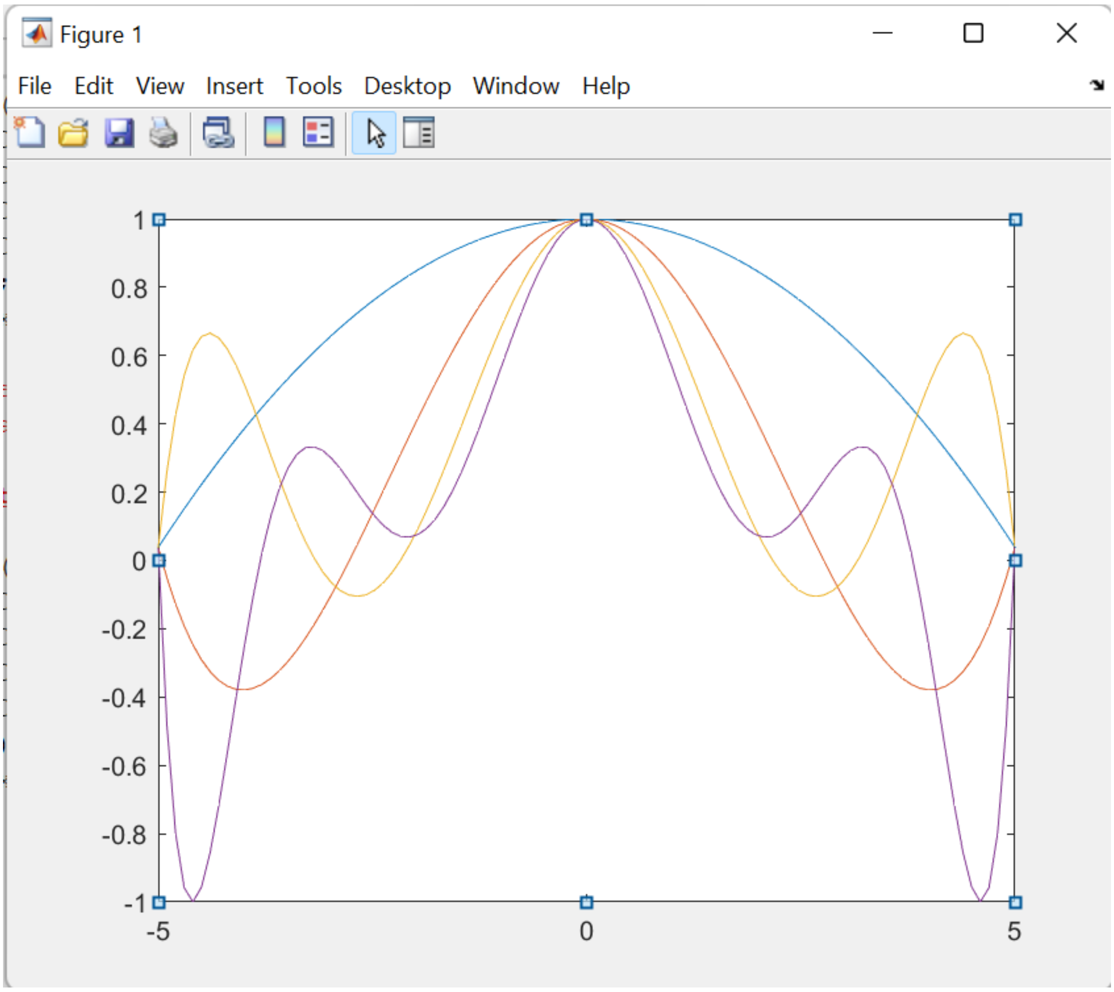
```
p2(f;x ) = 0. 0384615+0. 192308(x+5)-0. 0384615(x+5)(x-0)

p4(f;x ) = 0. 0384615+0. 0397878(x+5)+0. 061008(x+5)(x+2. 5)-0. 0265252(x+5)(x+2. 5)(x-0
)+0. 00530504(x+5)(x+2. 5)(x-0)(x-2. 5)

p6(f;x ) = 0. 0384615+0. 0264644(x+5)+0. 0248454(x+5)(x+3. 33333)+0. 0149446(x+5)(x+3. 3
3333)(x+1. 66667)-0. 0131699(x+5)(x+3. 33333)(x+1. 66667)(x-0)+0. 00420316(x+5)(x+3. 333
33)(x+1. 66667)(x-0)(x-1. 66667)-0. 000840633(x+5)(x+3. 33333)(x+1. 66667)(x-0)(x-1. 666
67)(x-3. 33333)

p8(f;x ) = 0. 0384615+0. 0223428(x+5)+0. 013956(x+5)(x+3. 75)+0. 0117043(x+5)(x+3. 75)(x
+2. 5)+0. 000674338(x+5)(x+3. 75)(x+2. 5)(x+1. 25)-0. 00489646(x+5)(x+3. 75)(x+2. 5)(x+1. 2
5)(x-0)+0. 00243964(x+5)(x+3. 75)(x+2. 5)(x+1. 25)(x-0)(x-1. 25)-0. 000687223(x+5)(x+3. 7
5)(x+2. 5)(x+1. 25)(x-0)(x-1. 25)(x-2. 5)+0. 000137445(x+5)(x+3. 75)(x+2. 5)(x+1. 25)(x-0)
(x-1. 25)(x-2. 5)(x-3. 75)
```

Runge phenomenon



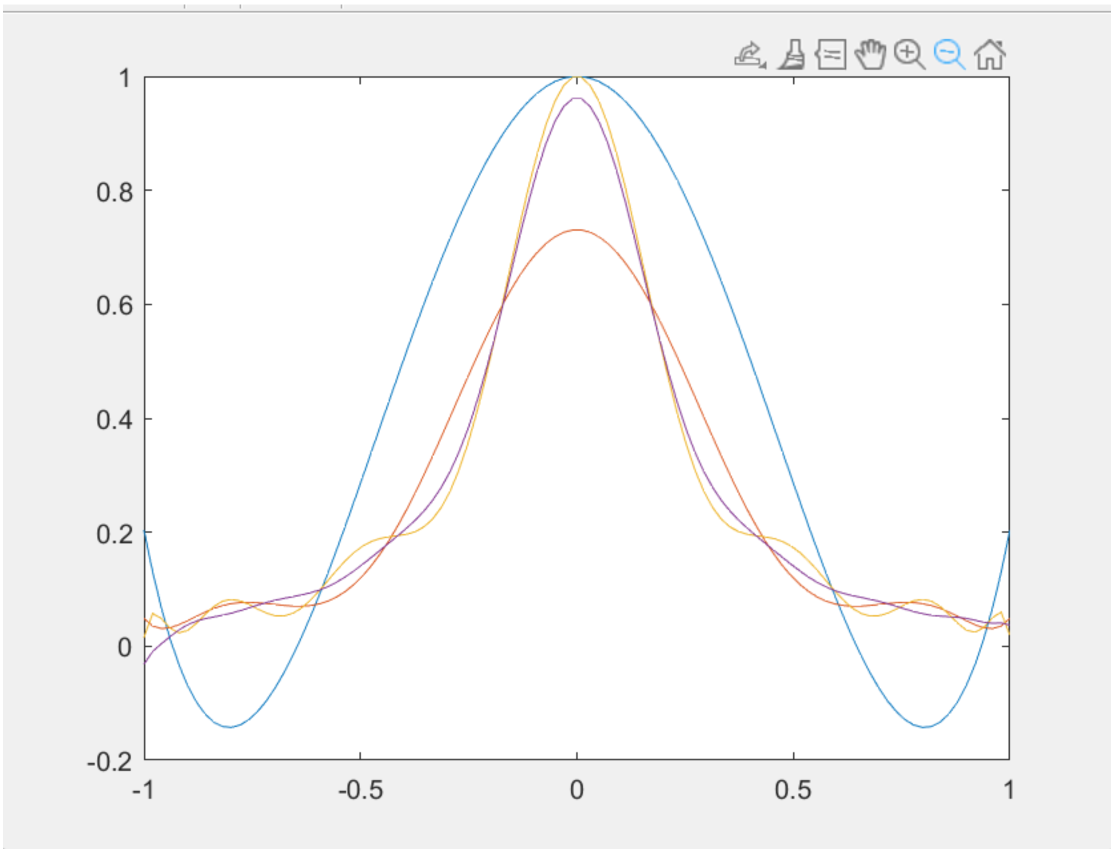
- Question C

Just show for the polynomial with n = 5 and n =10.

```
p5(f;x ) = 0.0423501-0.169057(x-0.951057)+1.42548(x-0.951057)(x-0.587785)+2.61208(x-0.951057)(x-0.587785)(x-6.12323e
-17)+2.7465(x-0.951057)(x-0.587785)(x-6.12323e-17)(x+0.587785)

p10(f;x ) = 0.0393884-0.0887389(x-0.987688)+0.189679(x-0.987688)(x-0.891007)-0.534305(x-0.987688)(x-0.891007)(x-0.70
7107)+2.11681(x-0.987688)(x-0.891007)(x-0.707107)(x-0.45399)+8.28743(x-0.987688)(x-0.891007)(x-0.707107)(x-0.45399)(
x-0.156434)+11.9543(x-0.987688)(x-0.891007)(x-0.707107)(x-0.45399)(x-0.156434)(x+0.156434)+10.3568(x-0.987688)(x-0.8
91007)(x-0.707107)(x-0.45399)(x-0.156434)(x+0.156434)(x+0.45399)+5.51277(x-0.987688)(x-0.891007)(x-0.707107)(x-0.453
99)(x-0.156434)(x+0.156434)(x+0.45399)(x+0.707107)+8.9925e-16(x-0.987688)(x-0.891007)(x-0.707107)(x-0.45399)(x-0.156
434)(x+0.156434)(x+0.45399)(x+0.707107)(x+0.891007)
```

Chebyshev interpolation is free of the wide oscillations .



- Question D

Let s(x) be the position at time x , the speed at time x is f'(x).

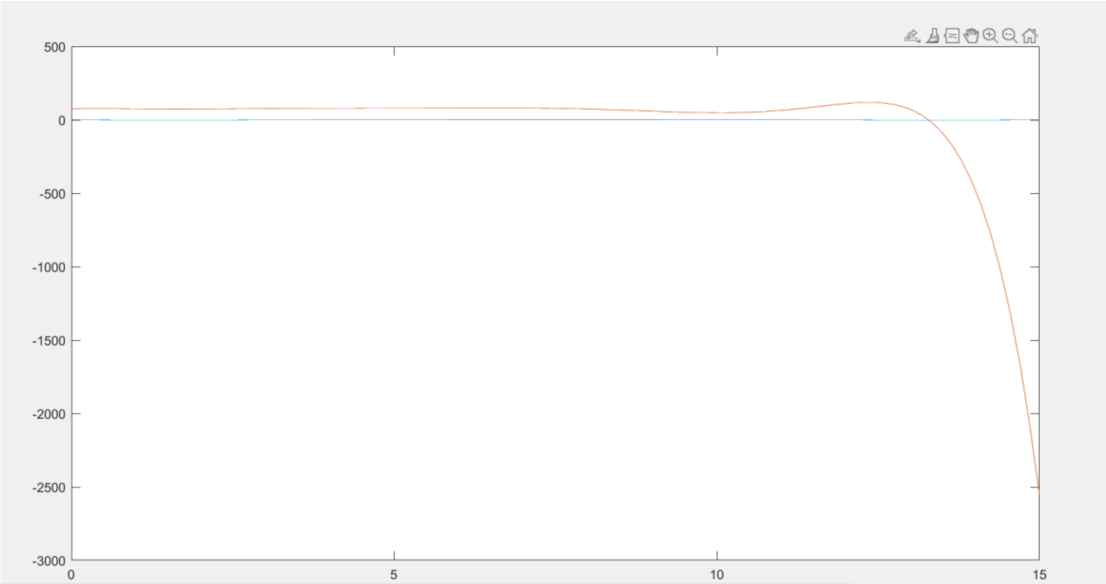
The following polynomial is f(x) with Newton's Interpolation

```
 Hermite Polynomial  :  +0+75(x-0)+0(x-0)(x-0)+0.222222(x-0)(x-0)(x-3)-0.0311111(x-0)(x-0)(x-3)(x-3
)-0.00644444(x-0)(x-0)(x-3)(x-3)(x-5)+0.00226389(x-0)(x-0)(x-3)(x-3)(x-5)(x-5)-0.000913194(x-0)(x
-0)(x-3)(x-3)(x-5)(x-5)(x-8)+0.000130527(x-0)(x-0)(x-3)(x-3)(x-5)(x-5)(x-8)(x-8)-2.02236e-05(x-0)
(x-0)(x-3)(x-3)(x-5)(x-5)(x-8)(x-8)(x-13)+0(x-0)(x-0)(x-3)(x-3)(x-5)(x-5)(x-8)(x-8)(x-13)(x-13)
```

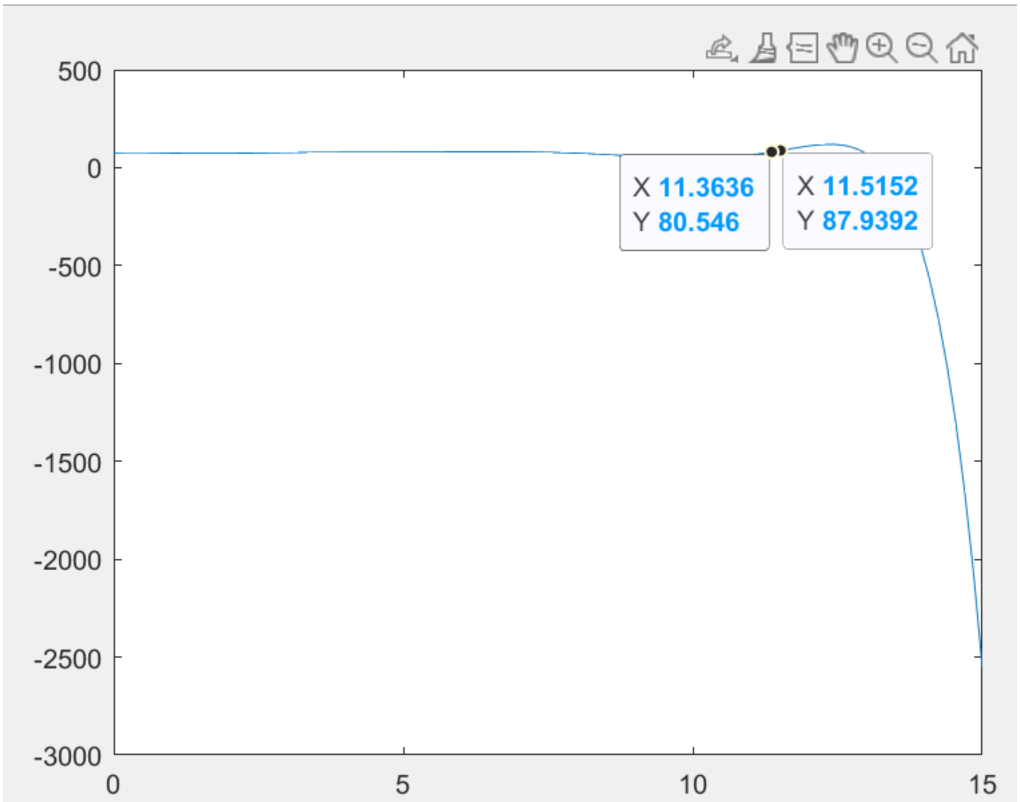
a)

When x = 10s , the position should be f(10) = 742.503(feet),

the speed should be f'(10)=48.3537(feet per second)



b)



When the time x = 11.5, the speed of car exceeds 81(feet per second)

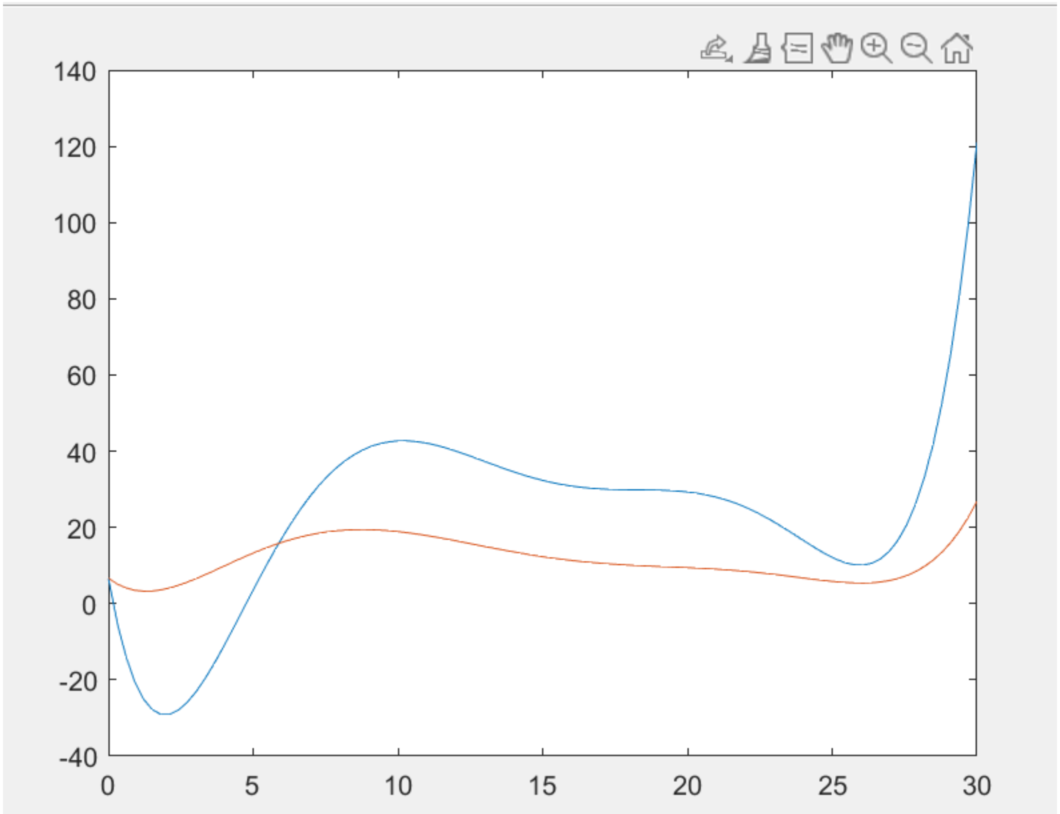
- Question E

The Newton's Interpolation results.

```
The polynomial of Sp1 is f1(x) = 6.67+1.77167.*(x-0)+0.457833.*(x-0).*(x-6)-0.124778.*(x-0).*(x-6).*
(x-10)+0.013566.*(x-0).*(x-6).*(x-10).*(x-13)-0.000978085.*(x-0).*(x-6).*(x-10).*(x-13).*(x-17)+4.14
77e-05.*(x-0).*(x-6).*(x-10).*(x-13).*(x-17).*(x-20)

The polynomial of Sp2 is f2(x) = +6.67+1.57167.*(x-0)-0.0871667.*(x-0).*(x-6)-0.0152729.*(x-0).*(x-6
).*(x-10)+0.00257908.*(x-0).*(x-6).*(x-10).*(x-13)-0.000204804.*(x-0).*(x-6).*(x-10).*(x-13).*(x-17)
+8.6768e-06.*(x-0).*(x-6).*(x-10).*(x-13).*(x-17).*(x-20)
```

The blue line and red line represent f1 and f2 respectively.



b)

```
When x = 43 , f1(43) = 14640.3
When x = 43 , f2(43) = 2981.48
```

Both samples of larvae will still alive.