The Report for Programming Assignments in Chapter Two

Wenchong Huang

Date: October 10, 2022

1 How to Test

Enter the folder `Programming-Chapter2/src` with terminal, `make` here, you will see some executable files whose names are corresponding assignments. Run them directly and you will see the results.

2 Mannual

This package is for Newton's polynomial interpolation.

2.1 Newton's Interpolation

You should include the header `interpolation.h`. Then you should give your interpolating points and interpolating values with two `std::vector<double>`. Here is an example where $x_k = -5 + 10\frac{k}{n}$ $(k = 0, 1, ..., n)$ and $f(x) = \frac{1}{1+x^2}$.

```
class F{
public:
    double operator () (const double &x) const{
        return 1.0 / (1.0 + x * x);
    }
} func;
std::vector<double> x, f;
for(int i = 0; i <= n; i++){
    x.push_back(-5.0 + 10.0 * i / n);
    f.push_back(func(x[i]));
}
```

After that, you can get the interpolation polynomial by:

```
NewtonInterpolation poly(x, f);
```

You can use the following code to get the value of $p_n(f; x)$ at point x.

```
double v = poly(x);
```

One of the advantage of Newton's interpolation is that it can add interpolating points conviniently. In this package, you can add one with the following code.

```
poly.addPoint(x_new);
```

It will increse the order of the polynomial `poly` by one to make it coincides with f at x_{new} in an $O(n)$ time.

2.2 Hermite's Interpolation

Hermite's Interpolation depends not only on function values, but also deriviate values. In this package, we only support the first order deriviate.

You should include the header `interpolation.h`. Give your interpolating points and corresponding function values and deriviate values with three `std::vector<double>`. Here is an example.

```
const int n = 5;
const double xvalues[] = {0, 3, 5, 8, 13};
const double fvalues[] = {0, 225, 383, 623, 993};
const double dfvalues[] = {75, 77, 80, 74, 72};
std::vector<double> x(xvalues, xvalues + n);
std::vector<double> f(fvalues, fvalues + n);
std::vector<double> df(dfvalues, dfvalues + n);
HermiteInterpolation hpoly(x, f, df);
```

You can get the value of the interpolation polynomial at point x with the following code.

```
double v = poly(x);
```

2.3 Output

The output of Newton's interpolation and Hermite's interpolation are the same, since they are both derived classes of `NewtonPolynomial`.

This package supports three output modes: Normal, Latex and Tikz.

The Normal mode is for human to read directly, as following

```
2+0.1*(x+5)-0.03*(x+5)*x
```

The Latex mode is for showing the result as a formula in a LaTeXdocument, as following

```
2+0.1\pi_{0}(x)-0.03\pi_{1}(x)
```

The Tikz mode is for drawing the image of the result in the **Tikz** package, as following

```
2+0.1*(\x+5)-0.03*(\x+5)*\x
```

Use standard IO stream to output.

```
std::cout << poly << std::endl;
```

The default output mode is Normal. To change it, using one of the following codes.

```
NewtonPolynomial::setOutput(NewtonPolynomial::OUTPUT_NORMAL);
NewtonPolynomial::setOutput(NewtonPolynomial::OUTPUT_LATEX);
NewtonPolynomial::setOutput(NewtonPolynomial::OUTPUT_TIKZ);
```

2.4 Standardization and Diriviate Function

We provided a standard polynomial class `Polynomial`. To turn a polynomial of Newton's form to the canonical form, using the following code, where `poly` is an object of `NewtonPolynomial` or its derived classes.

```
Polynomial p = poly.standardize();
```

Get the deriviate function of a standard polynomial with the following code.

```
Polynomial dp = p.diff();
```

The point value and output of standard polynomial are the same as `NewtonPolynomial`. A little difference is that the Normal mode and the Latex mode are merged to one, and looks more beautiful, as following.

```
10+75x+7.16191x^{2}-10.0953x^{3}
```

3 Results

3.1 Assignment B

Here are the Newton's interpolation results.

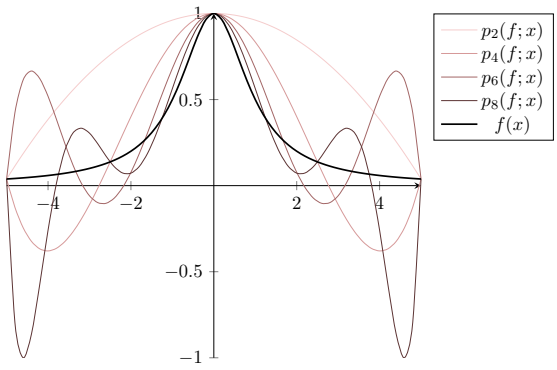
$$p_2(f;x) = 0.0384615 + 0.192308\pi_0(x) - 0.0384615\pi_1(x)$$

$$p_4(f;x) = 0.0384615 + 0.0397878\pi_0(x) + 0.061008\pi_1(x) - 0.0265252\pi_2(x) + 0.00530504\pi_3(x)$$

$$\begin{aligned} p_6(f;x) = {} & 0.0384615 + 0.0264644\pi_0(x) + 0.0248454\pi_1(x) + 0.0149446\pi_2(x) - 0.0131699\pi_3(x) \\ & + 0.00420316\pi_4(x) - 0.000840633\pi_5(x) \end{aligned}$$

$$\begin{aligned} p_8(f;x) = {} & 0.0384615 + 0.0223428\pi_0(x) + 0.013956\pi_1(x) + 0.0117043\pi_2(x) + 0.000674338\pi_3(x) \\ & - 0.00489646\pi_4(x) + 0.00243964\pi_5(x) - 0.000687223\pi_6(x) + 0.000137445\pi_7(x) \end{aligned}$$

The following figure shows the images of the interpolating polynomials and the original function.



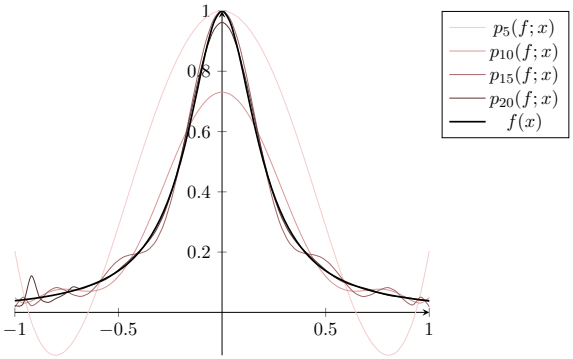
This figure illustrates the Runge phenomenon significantly.

3.2 Assignment C

In this assignment, we choose the interpolating points to be the zeros of Chebyshev polynomials T_n, which are

$$x_k = \cos\left(\frac{2k-1}{2n}\pi\right), \qquad k = 1, 2, ..., n$$

The interpolation results are too long to show. Please run the corresponding program directly to see the results. The following figure shows the images of the interpolating polynomials and the original function.



This figure illustrates that the Chebyshev interpolation is free of the wide oscillations in assignment B.

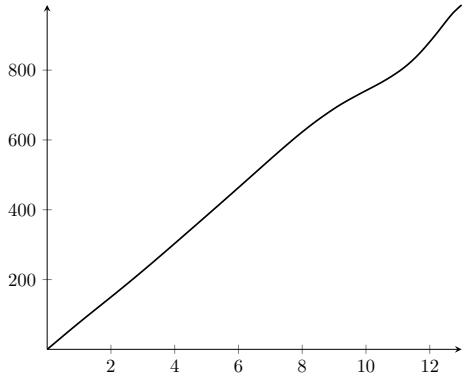
3.3 Assignment D

Let $f(x)$ be the position at time x, then the vilocity at time x is $f'(x)$.

Interpolating with the function values and deriviate values at $0, 3, 5, 8, 13$, we can get an polynomial of order 9, as following.

$$\begin{aligned} f(x) =& 75x + 7.16191x^2 - 10.0953x^3 + 5.50812x^4 - 1.5383x^5 + 0.243041x^6 \\ & - 0.0218757x^7 + 0.00104059x^8 - 2.02236 \times 10^{-5}x^9 \end{aligned}$$

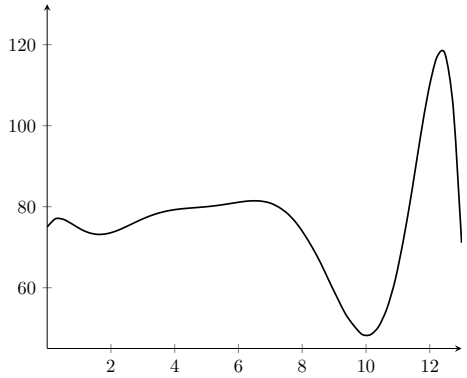
The image of the function sees the following figure.



To get the vilocity image, we can calculate the deriviate of f, as following.

$$\begin{aligned} f'(x) =& 75 + 14.3238x - 30.2859x^2 + 22.0325x^3 - 7.69148x^4 + 1.45825x^5 \\ & - 0.15313x^6 + 0.00832472x^7 - 0.000182013x^8 \end{aligned}$$

The image of vilocity function sees the following figure.



Now we can answer the problems:

(a) The position of time $t = 10s$ is $f(10) = 742.503$ (feet).

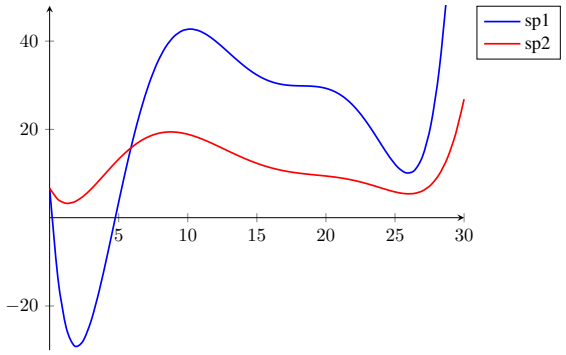
(b) As the vilocity image shows, it could be easily observed that the vilocity around time $t = 12s$ reaches about 120 feet per second. That's an excessive speeding.

3.4 Assignment E

Here are the interpolating results.

$$\begin{aligned} p_{\text{sp1}}(x) =& 6.67 + 1.77167\pi_0(x) + 0.457833\pi_1(x) - 0.124778\pi_2(x) + 0.013566\pi_3(x) \\ & - 0.000978085\pi_4(x) + 4.1477 \times 10^{-5}\pi_5(x) \\ p_{\text{sp2}}(x) =& 6.67 + 1.57167\pi_0(x) - 0.0871667\pi_1(x) - 0.0152729\pi_2(x) + 0.00257908\pi_3(x) \\ & - 0.000204804\pi_4(x) + 8.6768 \times 10^{-6}\pi_5(x) \end{aligned}$$

The images see the following figure.



In fact, that's not an acceptable result. Following the result below, the larvaes in sp1 died soon after birth, then reborned at about the 5th day. Morever, the larvaes will have eternal lives. The average weight after another 15 days is

$$p_{\text{sp1}}(43) = 14640.3$$

$$p_{\text{sp2}}(43) = 2981.48$$

Amazing.

This problem shows that predicting the trend with high-order polynomial interpolation is ridiculous.

4 Summary

The figures in this report are drawn with **Tikz**.

This template is designed by ElegantLaTeX Program.

Many appreciations for your carefully reading. If you found any mistakes, please contact me directly. Have fun with your loving one (boyfriend, girlfriend or coding)!