The Report for Programming Assignments in Chapter Two

Wenchong Huang

Date: October 9, 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 Initializing

You should include the header `interpolation.h`, then define your function object as a derived class of `Function`. Here is an example where $f(x) = \frac{1}{1+x^2}$.

```
class F : public Function{
public:
    double operator () (const double &x) const{
        return 1.0 / (1.0 + x * x);
    }
} f;
```

After that, you should give your interpolating points with a `std::vector<double>`. Here is an example where $x_k = -5 + 10\frac{k}{n}$ $(k = 0, 1, ..., n)$.

```
std::vector<double> x;
for(int i = 0; i <= n; i++)
    x.push_back(-5.0 + 10.0 * i / n);
```

Finally, you can get the interpolation polynomial by:

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

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

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

2.2 Add an Interpolating Point

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.

2.3 Output

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
```

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

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

3 Results

3.1 Assignment B

Here are the Newton's interpolation results.

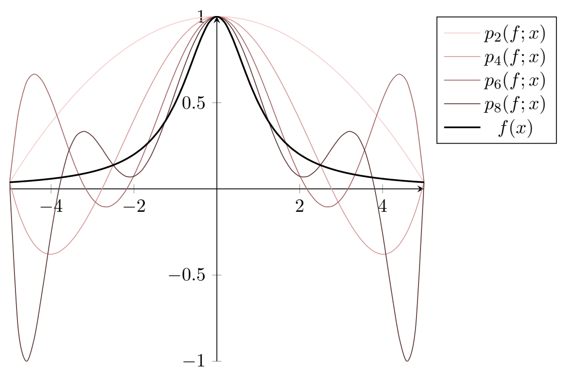
$$p_2(f;x) = 0.0384615 + 0.192308\pi_0(x) - 0.0384615\pi_1(x)$$

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

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

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

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



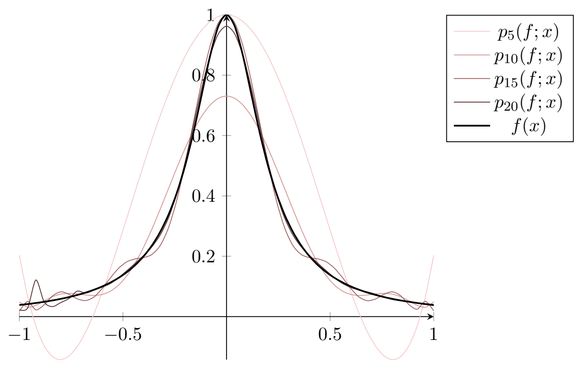
This figure illustrates the Runge phenomenon significantly.

3.2 Assignment C

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

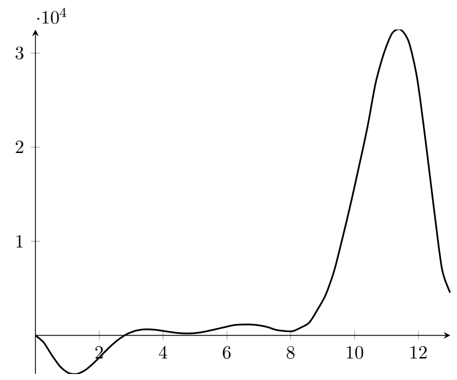
$$x_k = \cos\left(\frac{2k-1}{2n}\pi\right), \qquad k = 1, 2, ..., n$$

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.

4 Assignment D



5 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)!