实验名称

拉格朗日插值及牛顿插值算法的实现

完成人：车保良、杜翔宇

实验目的

> a. 验证拉格朗日插值算法对于不同函数的插值效果；
> b. 验证随着插值结点的增多插值曲线的变化情况。
> c. 验证插商的基本性质；
> d. 比较拉格朗日插值与牛顿插值的插值结果。

实验内容

一、 验证拉格朗日插值算法对于不同函数的插值效果

为了验证对不同函数的插值效果，我们选择了多项式函数、指数函数、三角函数、对数函数、分式函数五种不同类型的函数进行验证。

为了方便起见，我们统一将插值节点数设置成11个。

并且根据拉格朗日余项公式计算误差

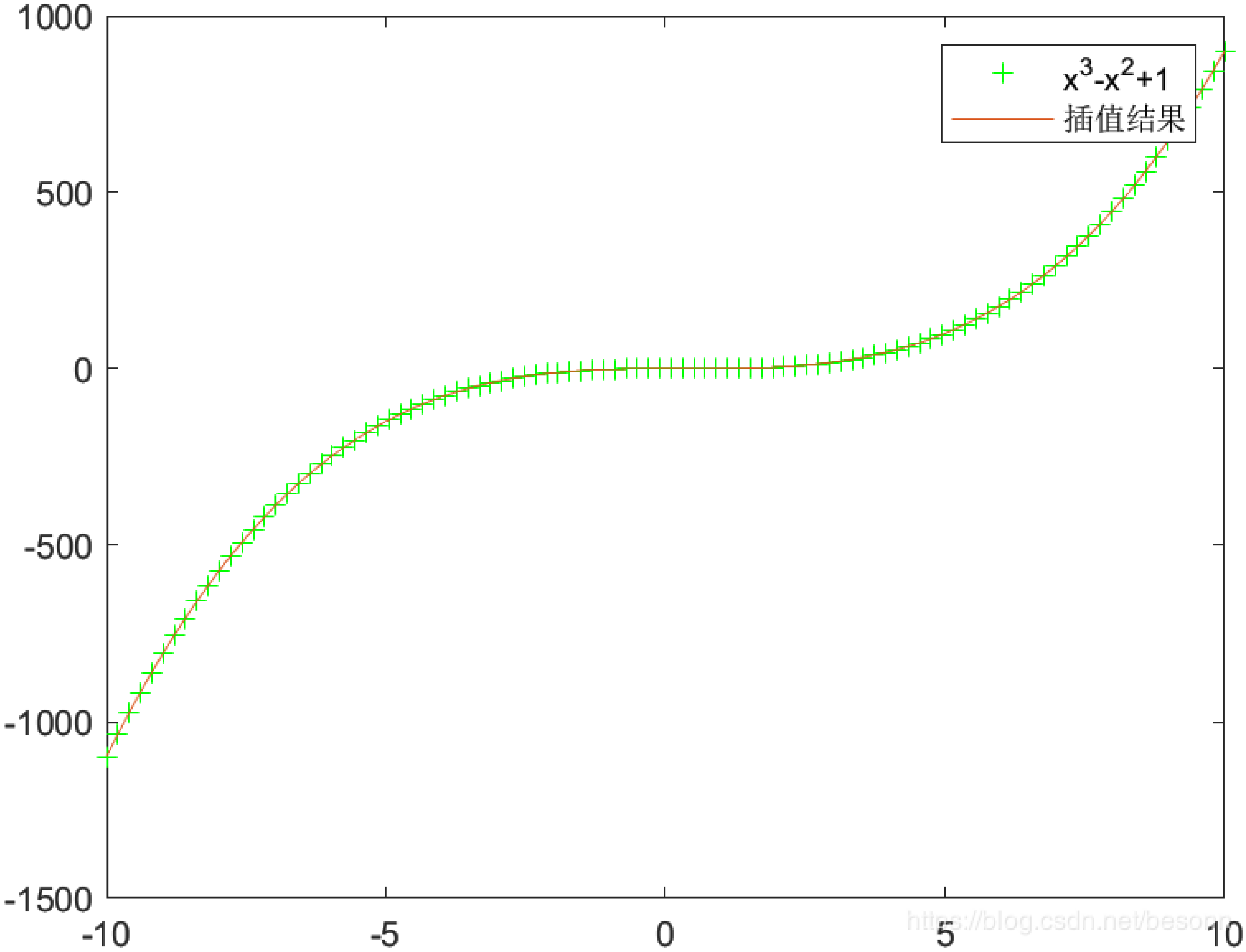
$$f(x)-p_n(x)=\frac{f^{(n+1)}(\varepsilon)}{(n+1)!}\prod_{i=0}^{n}(x-x_k)$$

多项式函数

我们选择的多项式函数为，函数的范围是$[-10,10]$

$$f(x)=x^3-x^2+1$$

下面是我们的结果



误差分析

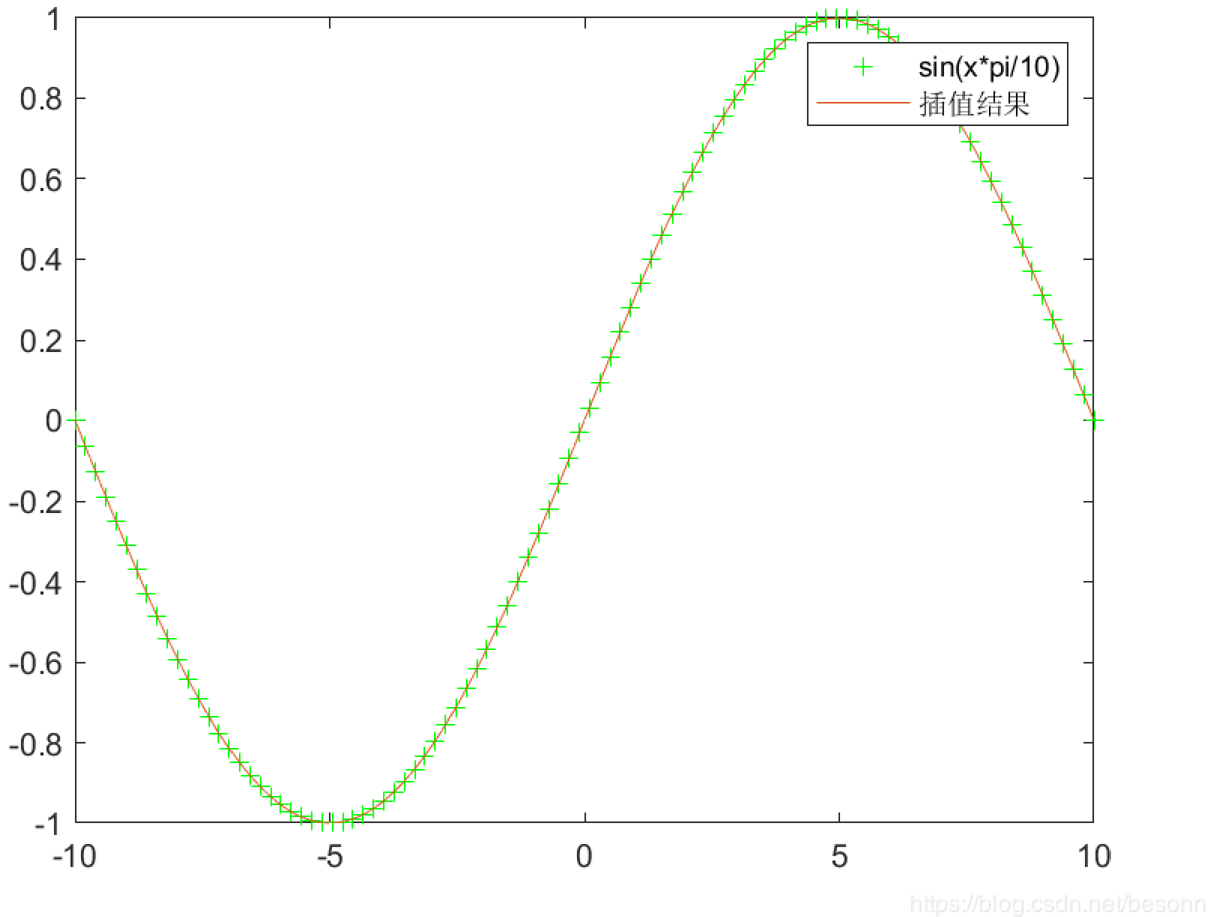
由于多项式函数的$n+1$阶导数为0，所以误差$f(x)-p_n(x)=0$

三角函数

我们选择的三角函数为

$$f(x)=sin(\frac{\pi}{10}x)$$

下面是我们的结果



误差分析

经过计算，误差方程为

$$f(x) - p_n(x) = \frac{\frac{pi}{10}^{(10+1)} cos(\frac{pi}{10}x)}{(10+1)!} \prod_{i=0}^{n}(x - x_k)$$

计算得到的误差限为

$$1.6130 \times 10^{-6}$$

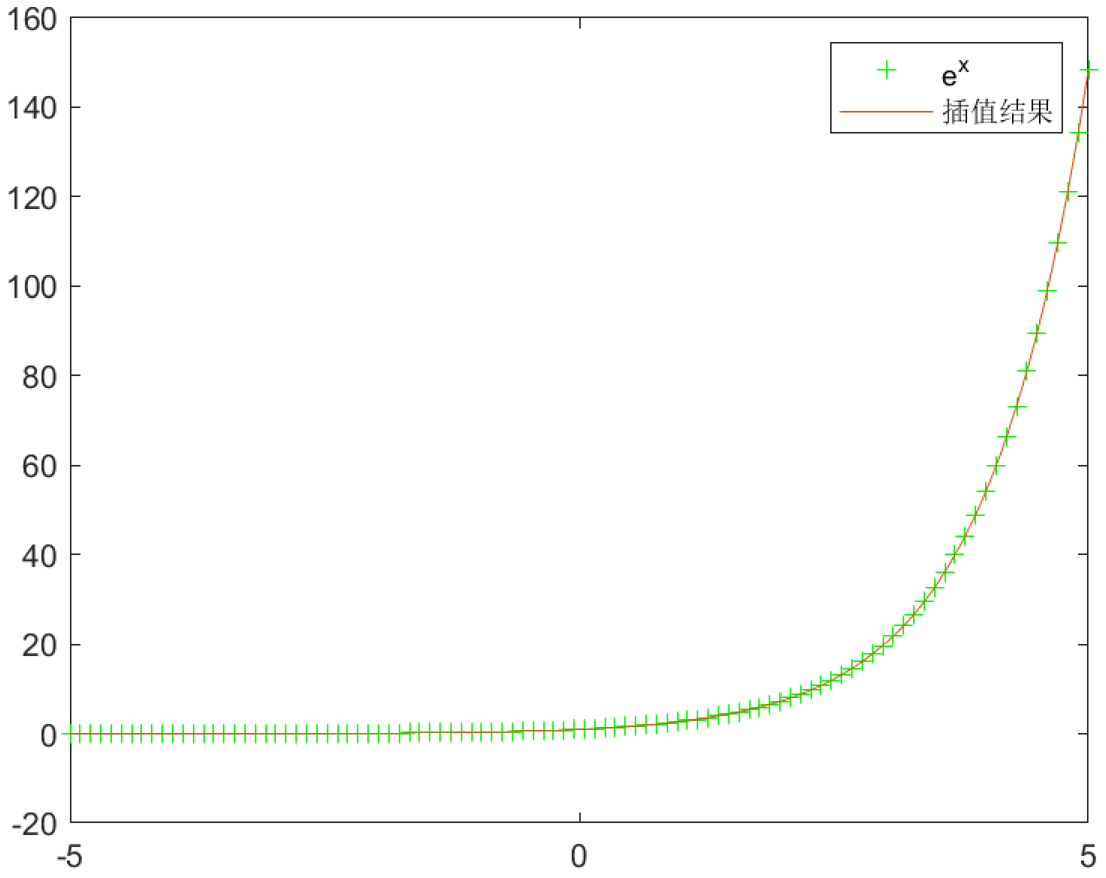
指数函数

我们选择的指数函数为

$$f(x) = e^x$$

由于指数函数的上升非常快，为了画图方便，我们选择$[-5, 5]$的表示区间

下面是我们的结果



误差分析

我们得到的误差方程为

$$f(x)-p_n(x)=\frac{e^x}{(10+1)!}\prod_{i=0}^{n}(x-x_k)$$

计算得到误差限为

$$0.0104$$

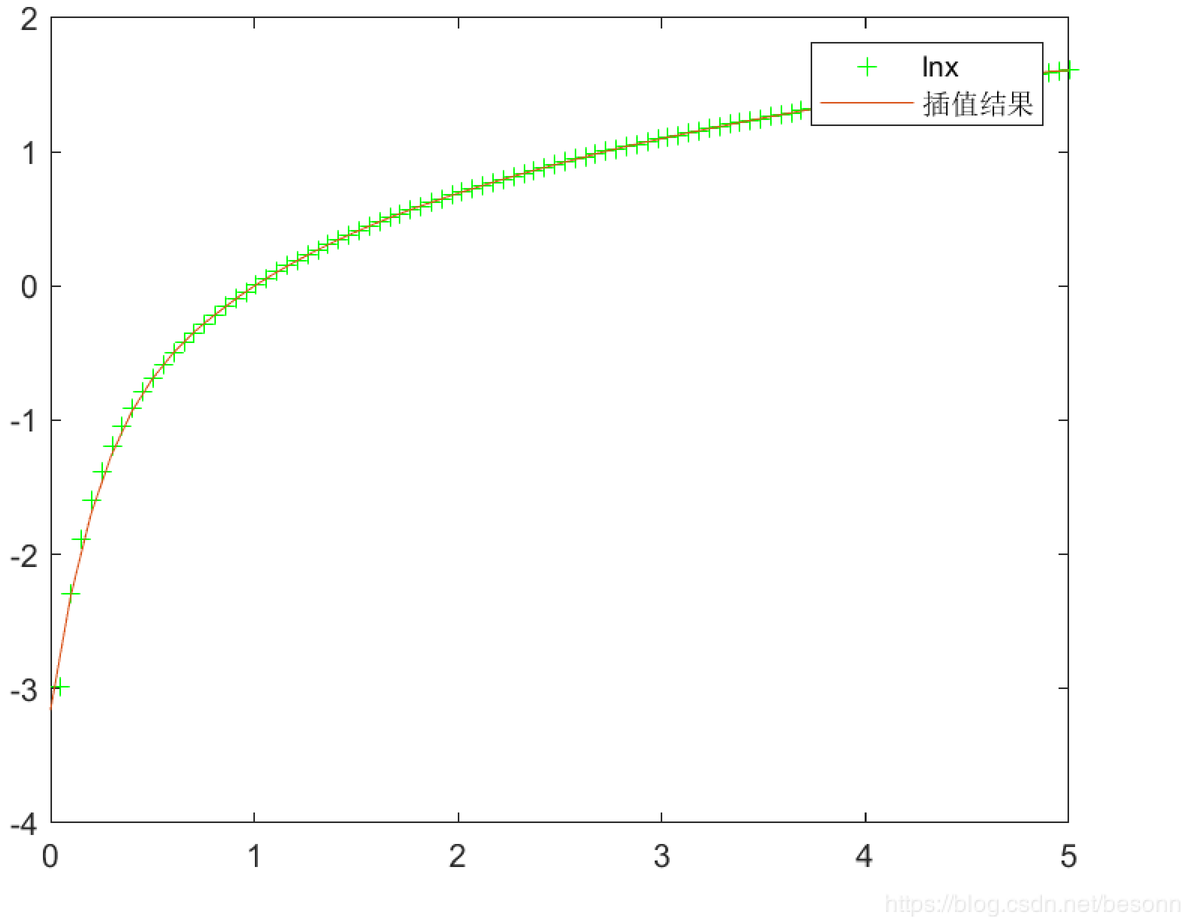
对数函数

我们选择的对数函数为

$$f(x)=lnx$$

区间为$(0,5]$

下面是我们的结果



误差分析

我们得到的误差方程为

$$f(x)-p_n(x)=\frac{-1}{x^{11}(10+1)!}\prod_{i=0}^{n}(x-x_k)$$

计算得到误差限为

$$9.7656\times 10^{-4}$$

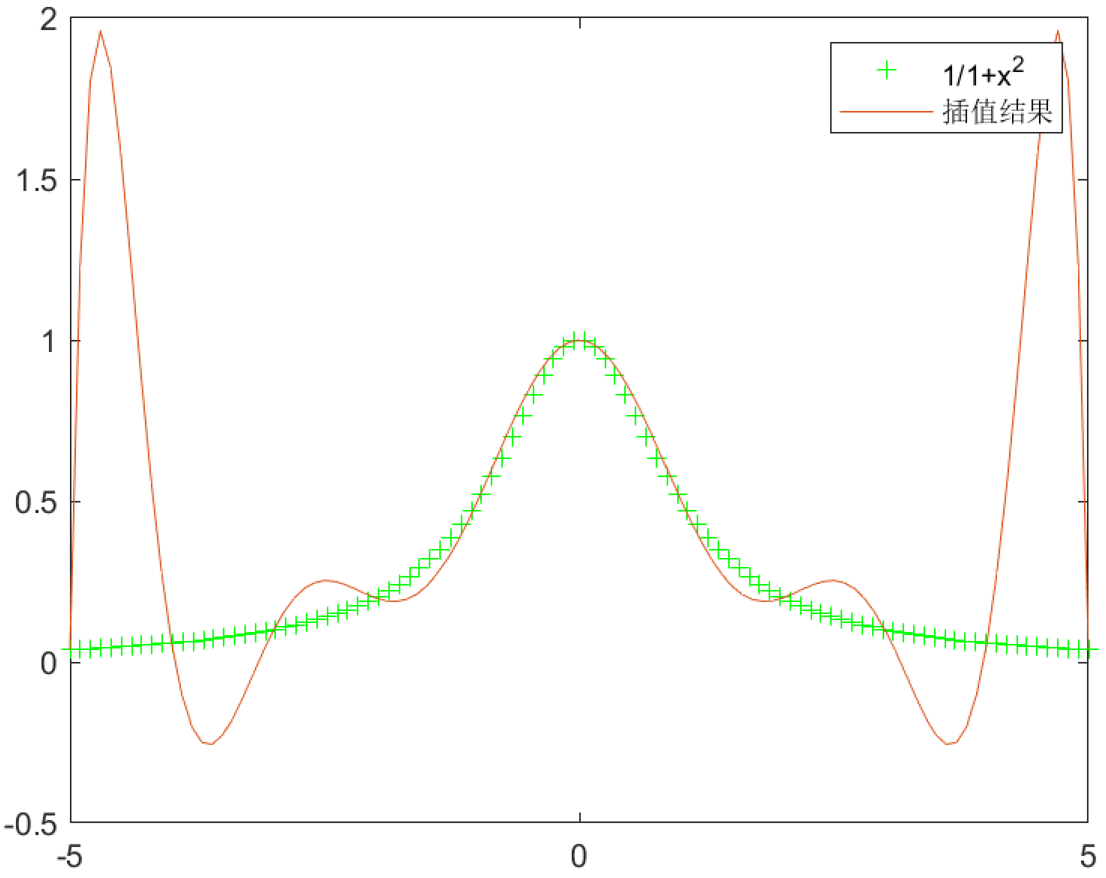
分式函数

我们选择的分式函数为

$$f(x)=\frac{1}{1+x^2}$$

区间为$[-5,5]$

我们得到的结果是



可以看到，在较为靠近0的区间里，函数的拟合效果比较好，两侧的效果并不是很好，发生了龙格现象。

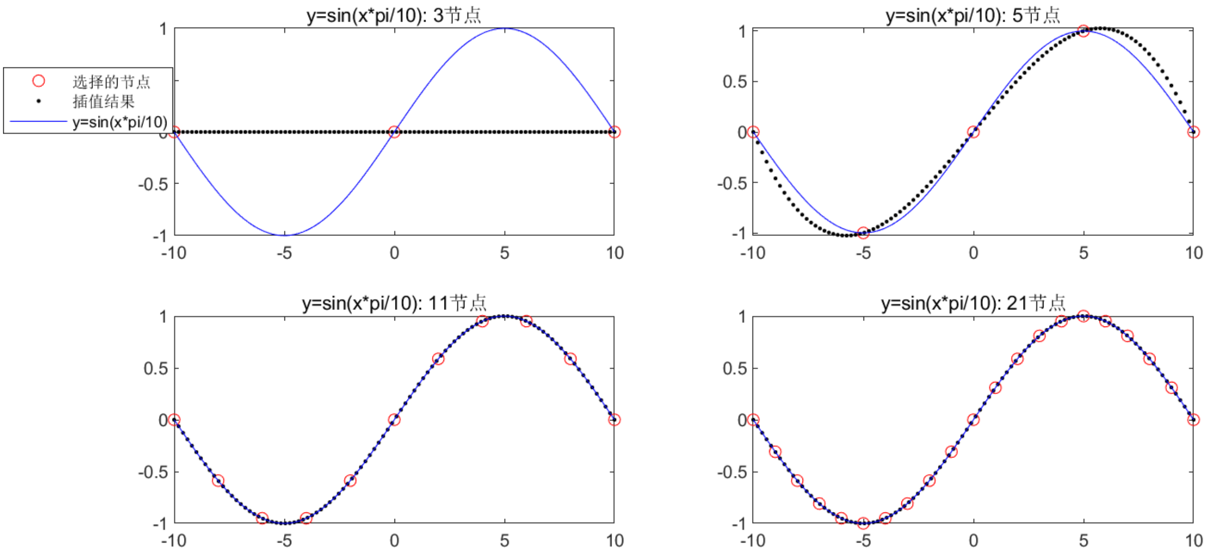
根据误差公式，我们计算得到的误差限为

$$0.0104$$

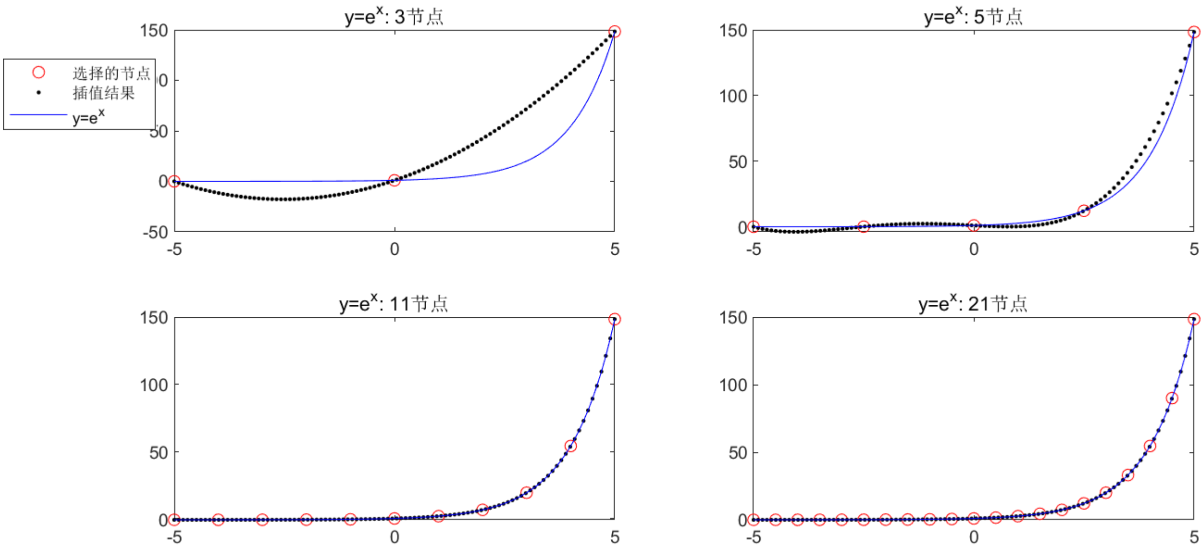
二、 验证随着插值结点的增多插值曲线的变化情况

为了进行实验，我们选择了3、5、11、21节点来验证随着插值结点的增多插值曲线的变化情况。

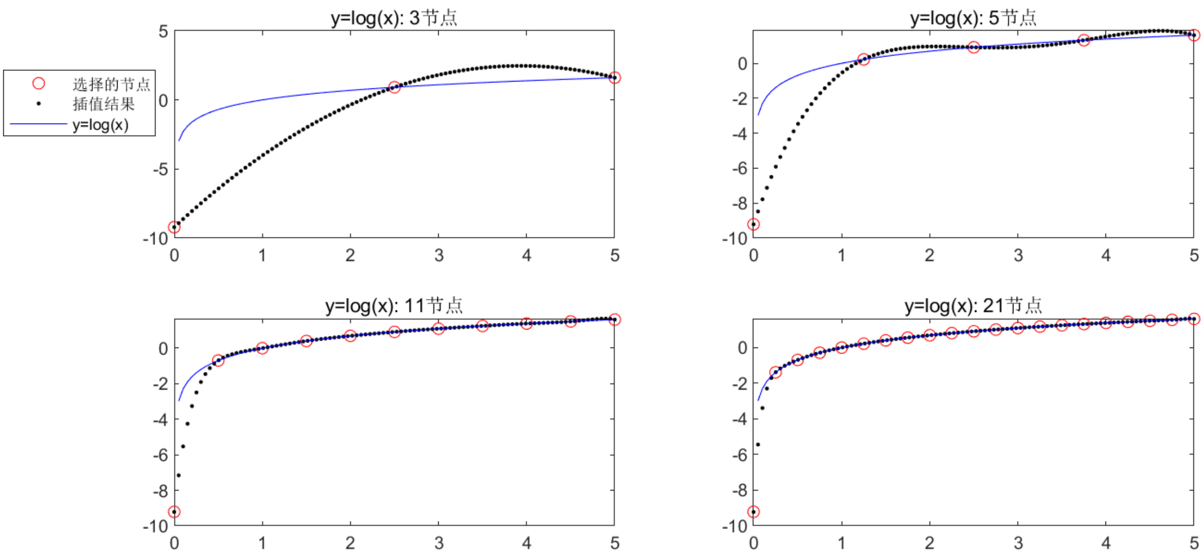
三角函数



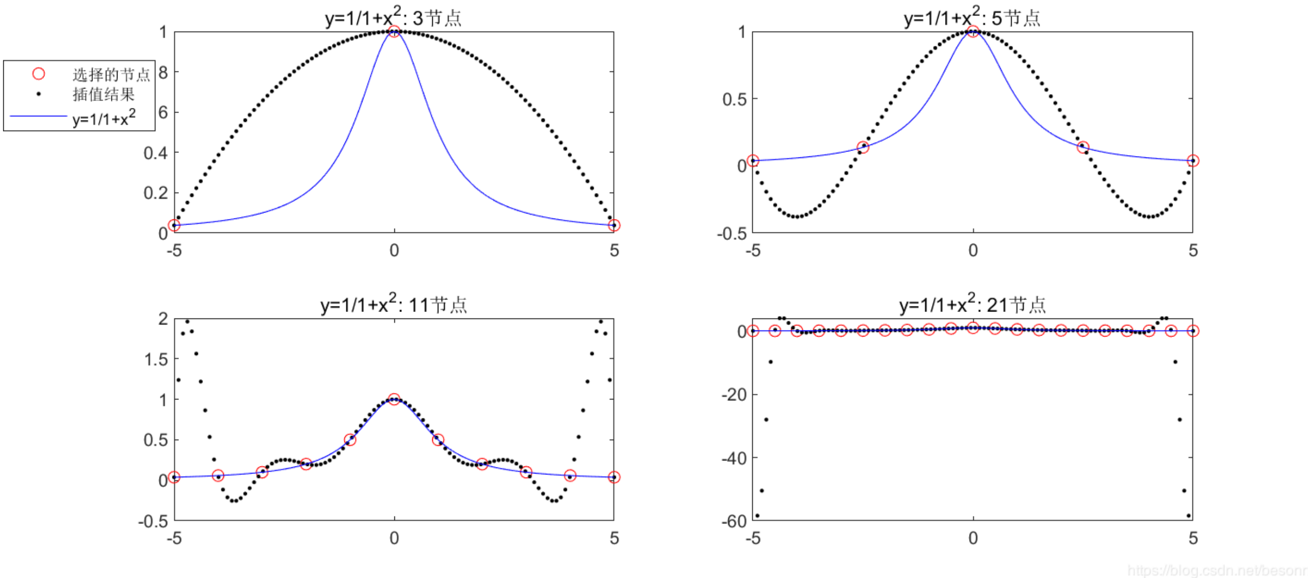
指数函数



对数函数



分式函数



由以上结果可以发现，随着插值节点数增多，曲线的拟合效果更好。为了更加精确的表示，我们将以上函数的误差限做成以下的表格。

	多项式函数	指数函数	对数函数	三角函数	分式函数
3个	64.1493	8.0187	1.0023	64.1493	8.0187
5个	94.5353	2.9542	0.0923	94.5353	2.9542
11个	21.3272	0.0104	5.0837×10^-6	21.3272	0.0104
21个	0.0045	2.1384×10^-9	1.0192×10^-15	0.0045	2.1384×10^-9

总体随着插值个数增多，误差变小。

三、 验证插商的基本性质

为了验证插商的基本性质，我们选择函数形式比较好的$f(x)=e^x$函数来进行实验。

性质一

k阶插商可以表示成k+1个函数值的线性组合

$$f[x_0, x_1, \cdots, x_k] = \sum_{j=0}^{k} \frac{f(x_j)}{(x_j - x_0)\cdots(x_j - x_{j-1})(x_j - x_{j+1})\cdots(x_j - x_n)}$$

首先求出该函数的1~5阶差商

```
开始计算...
X[0]=-5.000000   Y[0]=0.006700
X[1]=-2.500000   Y[1]=0.081000
X[2]=0.000000   Y[2]=1.000000
X[3]=2.500000   Y[3]=12.182500
X[4]=5.000000   Y[4]=148.413200
        求得0阶差商:0.006700
插值多项式第0个分量:0.006700
        求得1阶差商:0.029720
插值多项式第1个分量:0.297200
        求得2阶差商:0.067576
插值多项式第2个分量:5.068200
        求得3阶差商:0.100467
插值多项式第3个分量:37.675200
        求得4阶差商:0.112390
```

-5	0.0067	0.0301	0.0674	0.1005	0.1124
-2.5000	0.0821	0.3672	0.8212	1.2244	0
0	1	4.4730	10.0039	0	0
2.5000	12.1825	54.4923	0	0	0
5	148.4132	0	0	0	0

第5阶插商为0.11239

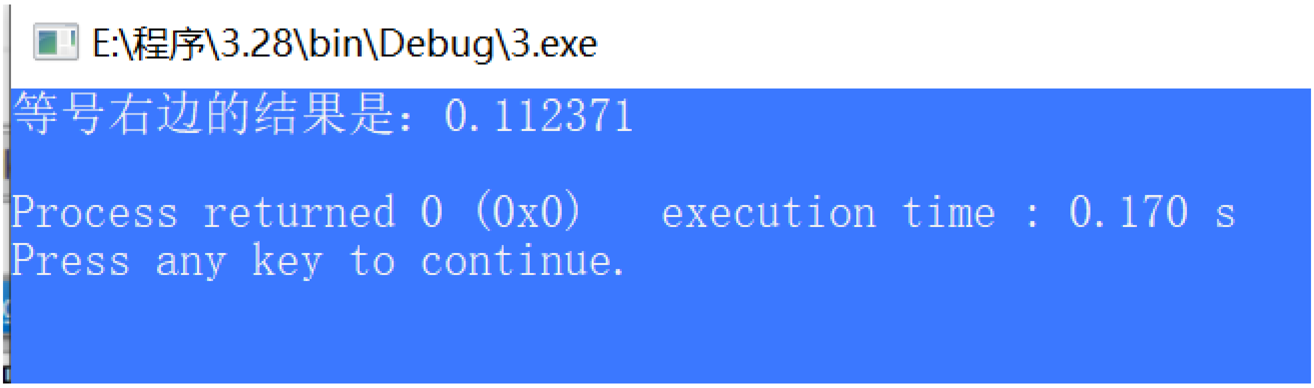
求插商主要代码如下

```
for(i=0;i<=k;i++)  //求差商 函数值的线性组合
        {
                temp_a=1;
                x=X[i];
                y=Y[i];
                for(j=0;j<=k;j++)
                {
                        if(i!=j)
                        {
                                temp_a *=(X[i]-X[j]);
                        }
                }
                temp_b=y/temp_a;
                temp_c +=temp_b;
                if(i>0)
                {
                        temp_d*=(newx-X[i-1]);  /*temp_d表示累和 x-x[i]*/
```

```
            }
        }
```

等式右边主要计算代码如下

```
float sum=0;
float cnt=1.0;
for(int i=1;i<=5;i++)
{
    cnt=1;
    for(int j=1;j<=5;j++)
    {
        if(i!=j)
            cnt=cnt*(X[i]-X[j]);
    }
    sum=sum+Y[i]/cnt;
}
```



验证结果是$sum = 0.112371$

结果成立。

性质二

验证对称性。

$$f\left[X_{k-1}, X_1, \cdots, X_0, X_k\right] = \frac{f\left[X_{k-1}, X_1, \cdots, X_{k-2}, X_k\right] - f\left[X_{k-1}, X_1, \cdots, X_{k-2}, X_0\right]}{X_k - X_0}$$

根据性质一中的函数，我们可以得知

$$f[x_3, x_2, x_1, x_0, x_4] = 0.1124$$

下面求等式右边的结果

$$f[x_3, x_1, x_2, x_4] = 1.2244$$

$$f[x_3,x_1,x_2,x_0]=0.1005$$

整个分式结果

$$\frac{1.2244-0.1005}{5-(-5)}=0.11239\approx 0.1124$$

性质成立

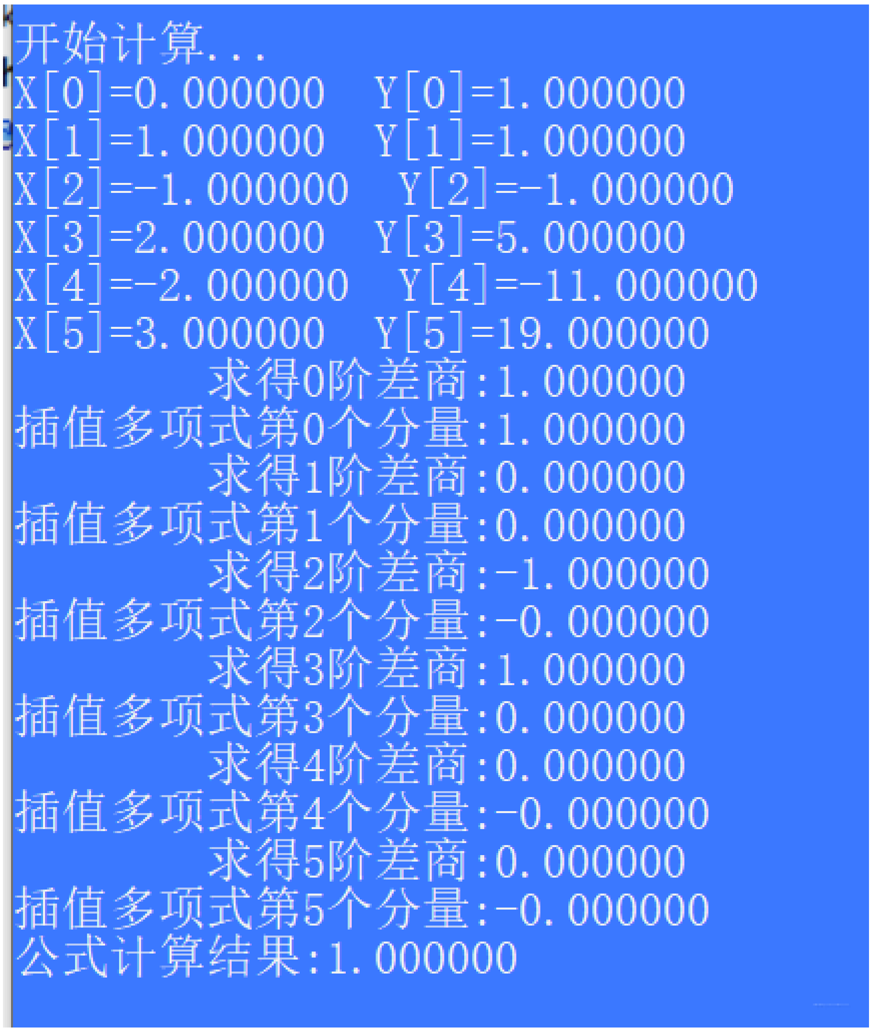
性质三

由于我们的程序无法直接算出阶数，因此我们决定验证$k>n$和$k=n$的情况

我们取多项式

$$f(x)=x^3-x^2+1$$

计算k>3时的差商



显然，可以证明，当k=n时差商是一个常数，不随x值发生变化；当k>n时，差商为0。

性质四

$$f[x_0, x_1, \cdots, x_n] = \frac{f^n(\xi)}{n!}, \xi \in [a, b]$$

为了验证以上等式，我们可以将$n!$转移到等号左边，判断和右边的关系。

由上面我们可以知道

$$f[x_0, x_1, x_2, x_2, x_4] \times n! = 2.6976$$

然而

$$f^n(e^\varepsilon) = e^\varepsilon \quad \varepsilon \in [-5, 5]$$

又因为，e^x单调递增，所以

$$e^\varepsilon \in [0.0067, 148.41]$$

结论显然成立

四、 比较拉格朗日插值与牛顿插值的插值结果

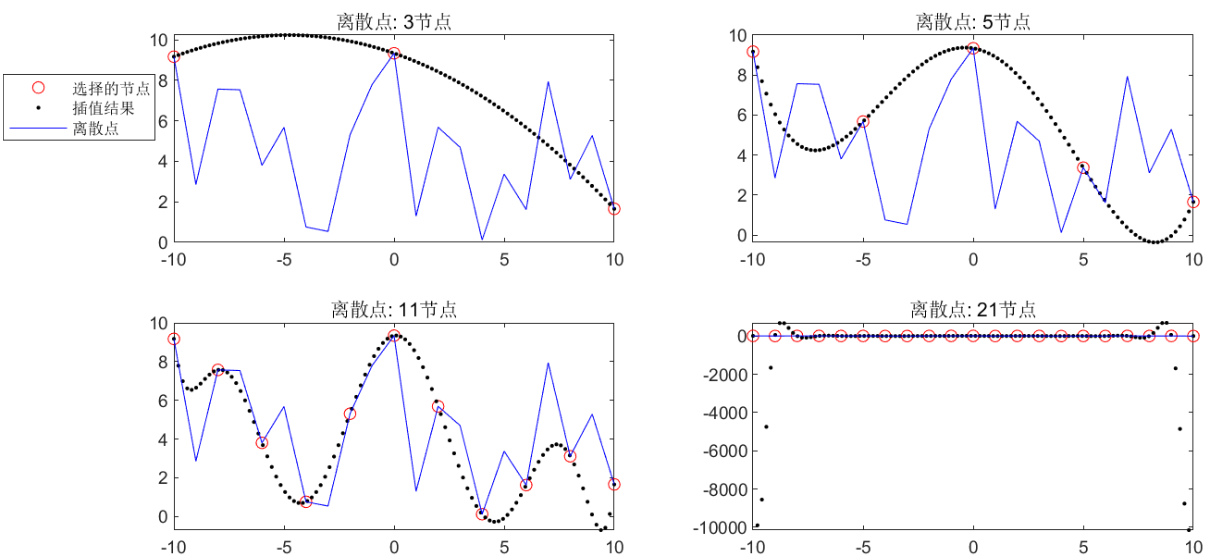
通常的，拉格朗日插值法和牛顿插值法的结果一致。

$$P_n(x_k) = N_n(x_k) = f_n(x_k)$$

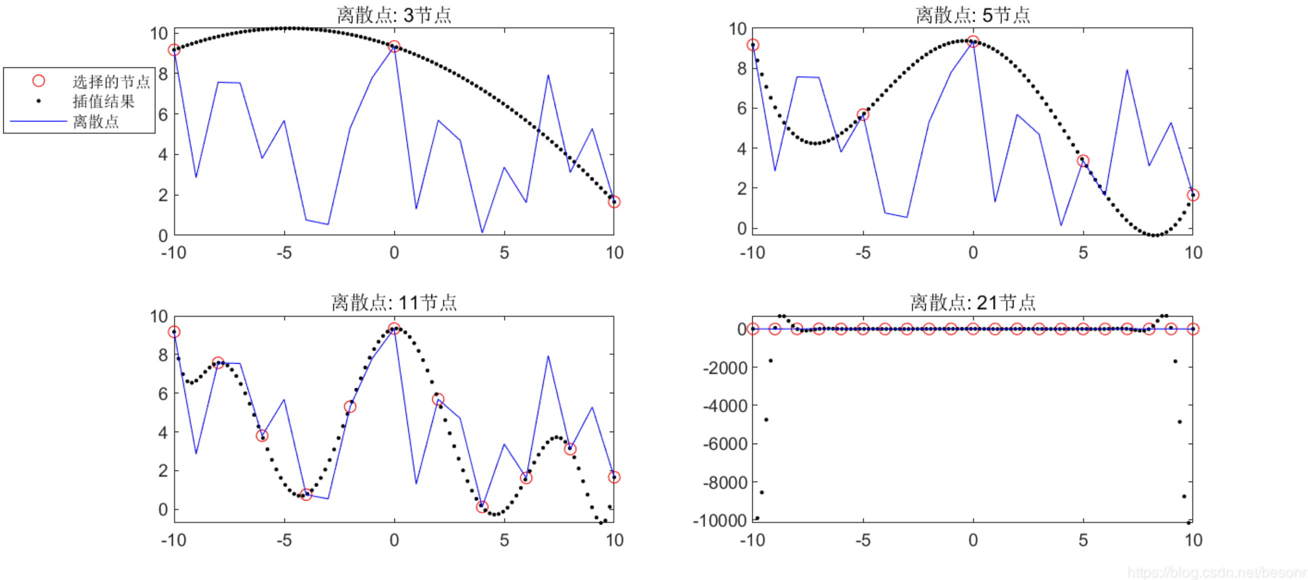
根据余项公式，$P_n(x) \equiv N_n(x)$

为了使得结果更加直观，我们选择了随机数生成了散点图。

拉格朗日插值



牛顿插值



结论：拉格朗日插值和牛顿插值计算结果基本相同，但是毫无疑问牛顿插值法更加适合离散的或者不可导的函数。

心得体会

1. 实际体会了插值方法的作用，加深了对牛顿插值和拉格朗日插值的理解。
2. 一般情况下插值的结果在取点数目比较多的情况下会更加精确，但也有例外。
3. 取点数大于多项式次数时，插值方法总能精确表达出该多项式。
4. 有些函数不适合使用插值方法，有些函数的某些区间不适合使用插值方法。
5. 对于同一个函数，区间越小，插值方法的精确度越高。
6. 用插值方法去推导已知点所处区间之外的点对应的值往往不准确。

附录

拉格朗日插值主要代码

```
#include <stdio.h>
#include <stdlib.h>
struct point{
        double x;
        double y;
};

double lglr(double x0,point* points,int n)
{
        double ret=0;
```

```
        double fenzi;
        double fenmu;
        for(int i=0;i<n;i++)
        {
                fenzi=1;
                fenmu=1;
                for(int j=0;j<n;j++)
                {
                        if(i!=j)
                        {
                                fenzi*=(x0-points[j].x);
                                fenmu*=(points[i].x-points[j].x);
                        }
                }
                ret+=(fenzi*points[i].y/fenmu);
        }
        return ret;
}

int main()
{
        int n;
        printf("input the number of points\n");
        scanf("%d",&n);
        point* points=(point*)malloc(sizeof(point)*n);
        for(int i=0;i<n;i++)
        {
                scanf("%lf",&points[i].x);
                scanf("%lf",&points[i].y);
        }
        printf("input x\n");
        double x0;
        while(1)
        {
                scanf("%lf",&x0);
                printf("%lf\n",lglr(x0,points,n));
        }
}
```

牛顿插值主要代码

```
#include <stdio.h>
#include <stdlib.h>
struct point{
```

```
	double x;
	double y;
};

point* points;

double chafen(int start,int end)
{
	if(start==end)
	{
		return points[start].y;
	}
	return (chafen(start,end-1)-chafen(start+1,end))/(points[start].x-points
}

double nd(double x0,int n)
{
	double ret=points[0].y;
	for(int i=1;i<n;i++)
	{
		double r=1;
		for(int j=0;j<i;j++)
		{
			r*=(x0-points[j].x);
		}
		ret+=r*chafen(0,i);
	}
	return ret;
}

int main()
{
	int n;
	printf("input the number of points\n");
	scanf("%d",&n);
	points=(point*)malloc(sizeof(point)*n);
	for(int i=0;i<n;i++)
	{
		scanf("%lf",&points[i].x);
		scanf("%lf",&points[i].y);
	}
	printf("input x\n");
	double x0;
	while(1)
	{
		scanf("%lf",&x0);
		printf("%lf\n",nd(x0,n));
```

```
        }
}
```