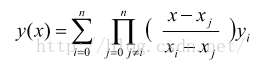
# 插值法 汇总

## Lagrange插值



和范德蒙行列式有关系，暂时未处理（线性代数不会）

## 牛顿插值法优缺点

牛顿插值法的优点是计算较简单，尤其是增加节点时，计算只增加一项，这是拉格朗日插值无法比的。   
缺点是仍没有改变拉格朗日的插值曲线在节点处有尖点，不光滑，插值多项式在节点处不可导等缺点。

### 牛顿法应用范围

牛顿法主要有两个应用方向：（名称不同严格讲）

**Newton法**1、目标函数最优化求解。例：已知 f(x)的表达形式，g(x)=\min\left\|{f(x)}\right\|，求ming(x)，及g(x)取最小值时的 x ？ 由于||f(x)||通常为误差的二范数，此时这个模型也称为最小二乘模型，即\min\{{f^2}(x)\}。

结论：当最优化问题的理论最小值为0时，Newton法求解就可变为Guass-Newton法求解。

**Guass-Newton（高斯牛顿）法**。2、方程的求解（根）。例：求方程的解：g(x) = 0，求 x ？这两个应用方面都主要是针对g(x)为非线性函数的情况。

2中，如果g(x)为线性情况下的求解通常使用最小二乘法求解。

详见知乎网址：<https://www.zhihu.com/question/22320408>

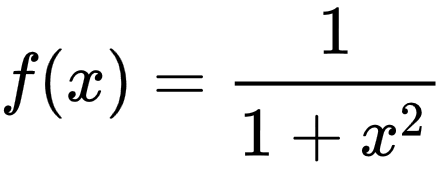
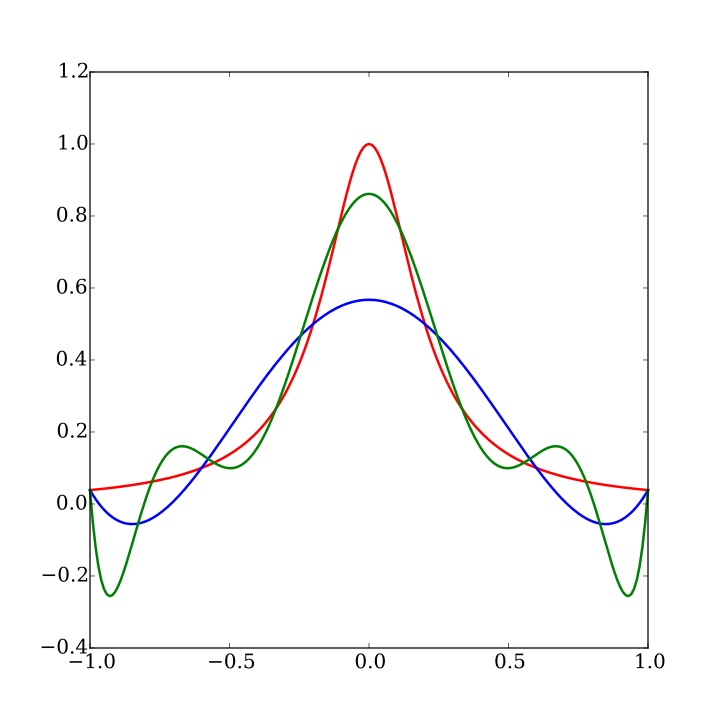
  牛顿法在进行编程实现的时候有可能会失败，具体原因及解决方法见《最优化方法》-张薇 东北大学出版社 第155页。

## Lagrange插值代码c++实现

1. **using** **namespace** std;
3. //预先定义插值节点的个数为1000个，根据控制台输入的个数num从而确定插值节点的个数
4. **const** **int** N=1000;
6. //arrX[N],arrY[N]分别存放的是插值节点(Xi,Yi)中的Xi,Yi,参数n为插值节点的个数,而参数x为待求解的插值节点的X值
7. //函数返回值为求解出来的插值节点X对应的Y值
8. //注意整个过程涉及的变量，除了循环变量为int外，其他均为float
9. **float** Lagrange(**float** arrX[],**float** arrY[],**int** n,**float** x)
10. {
11. **float** yResult=0.0;
13. //LValue[N]存放的是每次求解的插值基函数的通项
14. **float** LValue[N];
16. //循环变量k,m
17. **int** k,m;
18. //插值基函数中的上下累乘temp1,temp2
19. **float** temp1,temp2;
21. **for**(k=0;k<n;k++)
22. {
23. temp1=1.0;
24. temp2=1.0;
25. **for**(m=0;m<n;m++)
26. {
27. **if**(m==k)
28. {
29. **continue**;
30. }
31. temp1 \*= (x-arrX[m]);
32. temp2 \*= (arrX[k]-arrX[m]);
33. }
34. LValue[k]=temp1/temp2;
35. }
36. **for**(**int** i=0;i<n;i++)
37. {
38. yResult += arrY[i]\*LValue[i];
39. }
41. **return** yResult;
42. }
43. **int** main()
44. {
45. **float** arrX[N],arrY[N];
46. **int** num;
47. cout<<"输入插值节点的个数(小于"<<N<<"个): ";
48. cin>>num;
50. cout<<"\n--接下来输入这些插值节点(先输入X 再输入对应的Y)--\n";
51. **for**(**int** i=0;i<num;i++)
52. {
53. cout<<"第"<<i+1<<"个节点的X值: ";
54. cin>>arrX[i];
55. cout<<"第"<<i+1<<"个节点的Y值: ";
56. cin>>arrY[i];
57. }

60. **float** X;
61. cout<<"\n--请输入待求解的插值节点的X值--\n";
62. cin>>X;
64. **float** Res = Lagrange(arrX,arrY,num,X);
65. cout<<"\n--插值结果为: "<<Res<<endl;
66. **return** 0;
67. }
68. <span style="font-size:14px;">（5）下面是进行测试的结果，进行插值的三个节点为(11,0.190809),(12,0.107912),(13,0.224951) ,待测试的是(11.5,Y) 其中的函数          为y=sin(x) </span><span style="font-size:14px;">         测试结果为0.199368或者0.199369</span>

## 余项的作用

但是当我们的余项不再是无穷小的时候，我们就会看到**龙格现象(Runge)，**余项往往被我们用来分析误差，比如对函数 进行等距采样后的拟合，当拟合的点越多的时候，它的误差越大，实际上是因为它的余项并不是收敛于0的。

## Hermite插值

如果能在增加方程的同时不增加点的数量，实际上就是增加方程次数减少余项大小，本质依然是多项式求解。可以增加的方程有很多，比如增加导数方程让曲线看起来更平滑，这样就产生了**Hermite插值**算法。

埃尔米特插值是另一类插值问题，这类插值在给定的节点处，不但要求插值多项式的函数值与被插函数的函数值相同。同时还要求在节点处，插值多项式的一阶直至指定阶的导数值，也与被插函数的相应阶导数值相等，这样的插值称为埃尔米特插值，或称为Hermite插值。 Hermite插值在不同的节点，提出的差值条件个数可以不同，若在某节点xi，要求插值函数多项式的函数值，一阶导数值，直至m1-1阶导数值均与被插函数的函数值相同及相应的导数值相等。我们称xi为mi重插值点节,因此，Hermite插值应给出两组数，一组为插值点{xi}ni=0节点，另一组为相应的重数标号{mi}ni=0。

matlab中的数组是从1开始的

## 分段线性插值法

## 样条插值法

## Neville插值法

## 变尺度法

# 无约束问题

## 一维搜索方法

### 试探法

### 插值法（抛物线插值法，三次插值法）

### 微积分中的求根法（切线法，二分法）

## 二次插值法

## 无约束极值问题的解法

### 解析法

#### 梯度法（最速下降法）

#### Newton法

#### 变尺度法

### 直接法

#### Powell方法（基本搜索、加速搜索、调整搜索方向）