数值分析上机报告

吴秉哲

2015 年 3 月 15 日

这学期每次的上机报告都按如下格式编写：

每份报告分为几个章节，每个章节分别介绍上机作业中的一题，其中每个题目又分为以下几个部分：

1. 问题提出及相关背景知识
2. 问题的理论分析及解决方案
3. 程序的部分设计思路与细节
4. 计算成果与分析
5. 本次上机的反思

每个题目中要求回答的问题均蕴含在问题的理论分析与计算成果分析两部分

在程序方面，都是由 C++ 编写完成（其中用到了自己编写的库：矩阵（Matrix)，向量 (Vector)，以及数值代数的相关算法 (linear))，在 linux 下由 g++ 编译。数据方面一般使用 matlab 可视化处理，少部分使用了 python。

1 第一章第二题

1.1 问题提出及相关背景知识

提出如下形式的积分：

$$I_n = \int_0^1 \frac{x^n}{x+5} dx \quad n = 0, 1, 2 \cdots$$

题目要求计算 $I_1, I_2, \cdots I_{20}$ 的值

1.2 问题的理论分析及解决方案

记 $f_n(x) = \dfrac{x^n}{x+5}$，则当 $x \in [0, 1)$ 时，有:

$$\lim_{n \to \infty} f_n(x) = 0$$

且上述的收敛是一致的，则进一步有：

$$\lim_{n\to\infty} I_n = 0$$

由上式可以知道，如果算法设计合理，那么当 n 足够大的时候，程序计算出来的 $|I_n - I_{n-1}|$ 足够小。下面在理论上推导程序需要的递推关系，首先推导题目中要求的向前递推公式，易知:

$$\begin{aligned}\frac{x^n}{x+5} &= \frac{x}{x+5}x^{n-1} \\ &= \frac{x+5-5}{x+5}x^{n-1} \\ &= x^{n-1} - 5\frac{x^{n-1}}{x+5}\end{aligned}$$

由上面推导可以得到：

$$\begin{aligned}I_0 &= \int_0^1 \frac{1}{x+5} = ln1.2 \\ I_n &= -5I_{n-1} + \int_0^1 x^{n-1} \\ &= -5I_{n-1} + \frac{1}{n}\end{aligned} \tag{1}$$

(1) 式即为题目要求推导的递推关系式, 在上式中，形式上是由前一项计算后一项的值，于是，将其称为向前递推公式。

但从 (1) 式来看，当 n 增大时，I_{n-1} 与 $\frac{1}{n}$ 会逐渐靠近，这样用 (1) 计算积分时，会出现两个大小接近的数相减，会损失精度，会有大的误差。另一方面，设计算的初始误差为 E_0(这是由于计算机进行实数运算肯定会有一定的误差)，将第 n 步的误差记为 E_n 即有 $E_n = I_n - \overline{I_n}$, 而 $\overline{I_n}$ 表示第 n 步计算解，则由上面的递推公式得$E_n = (-1)^n(5!)E_0$. 由此可以知道，即使初始误差很小，但误差任然被不断的放大，导致后面的计算结果严重失真，这是一个不稳定的算法。后面的计算结果也印证了这一点。于是，我们可以预知用这个递推式计算时，计算结果会很不合理。到这里，我们可以根据递推式的特点，设计另外一种算法如下：首先对上面推导得出的递推式变形，得到向后递推公式：

$$I_{n-1} = -\frac{1}{5}I_n + \frac{1}{5n}$$

上面的公式是教材题目要求的递推公式的变形，它的主要特点是由 I_n 计算 I_{n-1}, 初始值 I_{20} 我们可以通过梯形法得一个近似值，实验表明，虽然初始误差相比第一种方法较大，但是随着计算的进行，舍入误差被不断的缩小。可以证明这是一个稳定的数值算法。下面我们编写程序来验证我们的两种方法。

1.3 程序设计的思路与细节

本题的编写很简单，只要按照 (1) 式，直接编写程序即可。本题编写了一个由前项 I_{n-1} 计算 I_n 的子函数。

1.4 计算成果与分析

为了使程序的结果有对比，我还使用了 Matlab 中自带的数值积分的计算函数计算题目所给积分的值。由于篇幅，只列出了题目要求的第一种方法的计算结果。下面列出在 double 与 float(C 语言) 环境下，程序的计算结果：

n	精确值	float 运算结果	差值	double 运算结果	差值
0	0.182322	0.182322	0.000000	0.182322	0.000000
1	0.088392	0.088392	0.000000	0.088392	0.000000
2	0.058039	0.058039	0.000000	0.058039	0.000000
3	0.043139	0.043138	0.000001	0.043139	0.000000
4	0.034306	0.034310	0.000004	0.034306	0.000000
5	0.028468	0.028448	0.000021	0.028468	0.000000
6	0.024325	0.024428	0.000103	0.024325	0.000000
7	0.021233	0.020719	0.000514	0.021233	0.000000
8	0.018837	0.021407	0.00257	0.018837	0.000000
9	0.016926	0.004076	0.01285	0.016926	0.000000
10	0.015368	0.079618	0.06425	0.015368	0.000000
11	0.014071	-0.307181	0.321252	0.014071	0.000000
12	0.012977	1.619237	1.606261	0.012977	0.000000
13	0.012040	-8.019263	8.031303	0.01204	0.000000
14	0.011229	40.167744	40.156514	0.011229	0.000000
15	0.010521	-200.772049	200.782570	0.010522	0.000001
16	0.009896	1003.922729	1003.912833	0.009891	0.000005
17	0.009342	-5019.554688	5019.564029	0.009368	0.000026
18	0.008846	25097.828125	25097.819279	0.008717	0.000130
19	0.008400	-125489.085938	125489.094338	0.009049	0.000648
20	0.007998	627445.500000	627445.492002	0.004757	0.003241

从上述结果明显可以看出，递推公式 (1) 的计算误差很明显，尤其是在 float 精度下，从 n=8 时，I_n 与真实值开始有明显差距。并且随着 n 的增大，误差也在变大。这与我们的理论分析相符合。因此我们可以回答题目的第二问：这个递推公式在电脑上的计算结果并不合理，原因见上面的理论分析。而第二种方法的计算结果，和我们理论分析一样，随着 n 的增大，计算的值与精确值

也越来越接近。

1.5 本次上机反思

从这个题目的例子，我们知道很多在数学理论上很合理的结果，在数值上并不可靠，这是由于在电脑上，实数的运算并不遵循数学的规则，任何运算都会有误差，而有些不当操作会使误差变大，在设计算法时，思维不要局限在通常数学推导的范围，比如这个题，我们一般都会首先考虑向前递推，而不是更合理的向后递推，要对我们解决的实际问题进行充分的分析，然后设计数值稳定的算法，这样才能很好的解决问题。

2 第二章第二题

2.1 问题提出及相关背景知识

1. 问题提出

   本体要求使用不同的插值方法（等距节点的 newton，非等距节点的 Lagrange 插值，等距节点的分段线性，等距节点的 Hermite，三次自然样条插值）对 Runge 函数 $R(x) = \dfrac{1}{1+x^2}$ 在区间 $[-5, 5]$ 进行插值逼近。最后得出不同的计算结果，从而对比各种方法的优劣。

2. 问题的背景知识

   本题中的 Runge 函数，在使用等距节点上的高次插值时，会出现 Runge 现象：插值多项式在区间端点附件会出现震荡，严重偏离真实值。产生这种现象的原因从复函数角度来看，是由于 R(z) 在 $z = \pm i$ 处有奇性。

   由此可见本题的目的就是让我们在实验中验证这种现象的存在，以及提出不同的插值方法来克服这种现象。

2.2 问题的理论分析及解决方案

首先，由于上述 Runge 现象的存在，我们通常使用的等距高次插值就失效了，其次出了上述的 Runge 现象外，等距高次插值方法的数值稳定性很差 (详见教材 P28) 这诱导我们从两个方向去改良方法，其一是将等距变为非等距，其二是将方法将高次变为分段低次插值。基于这两种思路，我们就可以从题目中提到的方法去实验了。

2.3 程序设计的思路与细节

本次作业编写了一个关于多项式插值的一个库 interpolation。将本次一些细节列在下面：

1. 编写 Newton 插值，需要写一个计算高阶差商的子函数，在这次上机中我采用等式 $f[x_0, x_1, \cdots, x_m] = \sum_{i=0}^{m} \frac{f(x_i)}{\prod_{j=0, j\neq i}^{m} (x_i - x_j)}$ 进行计算。也可以使用递归的方法编写。

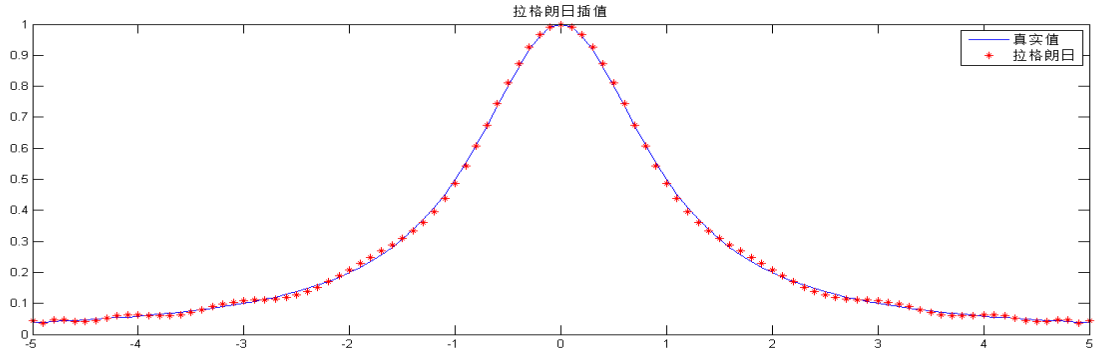
2. 在使用分段低次插值的算法时，首先需要确定 x 所在的区间 $[x_i, x_{i+1}]$, 我选择二分法进行搜索

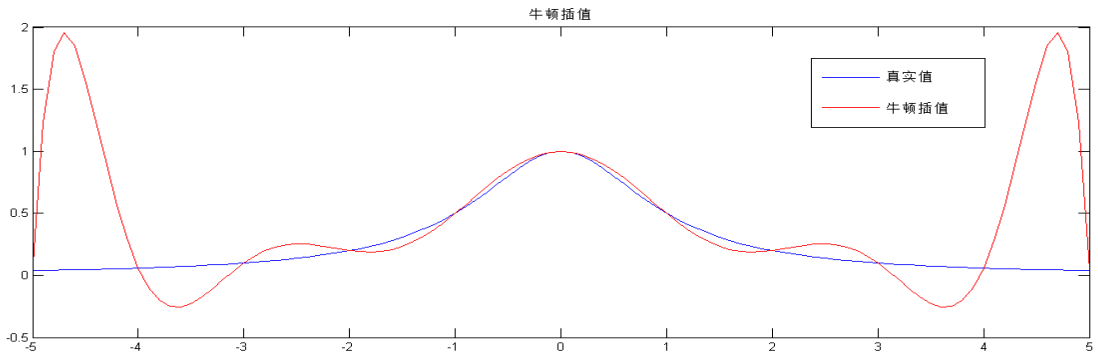
3. 注意到相应的分段插值，只需使用二分法确定 x 所在区间 $[x_i, x_{i+1}]$ 后，在相应区间使用两点插值即可，这个特点可以减少我们的工作量

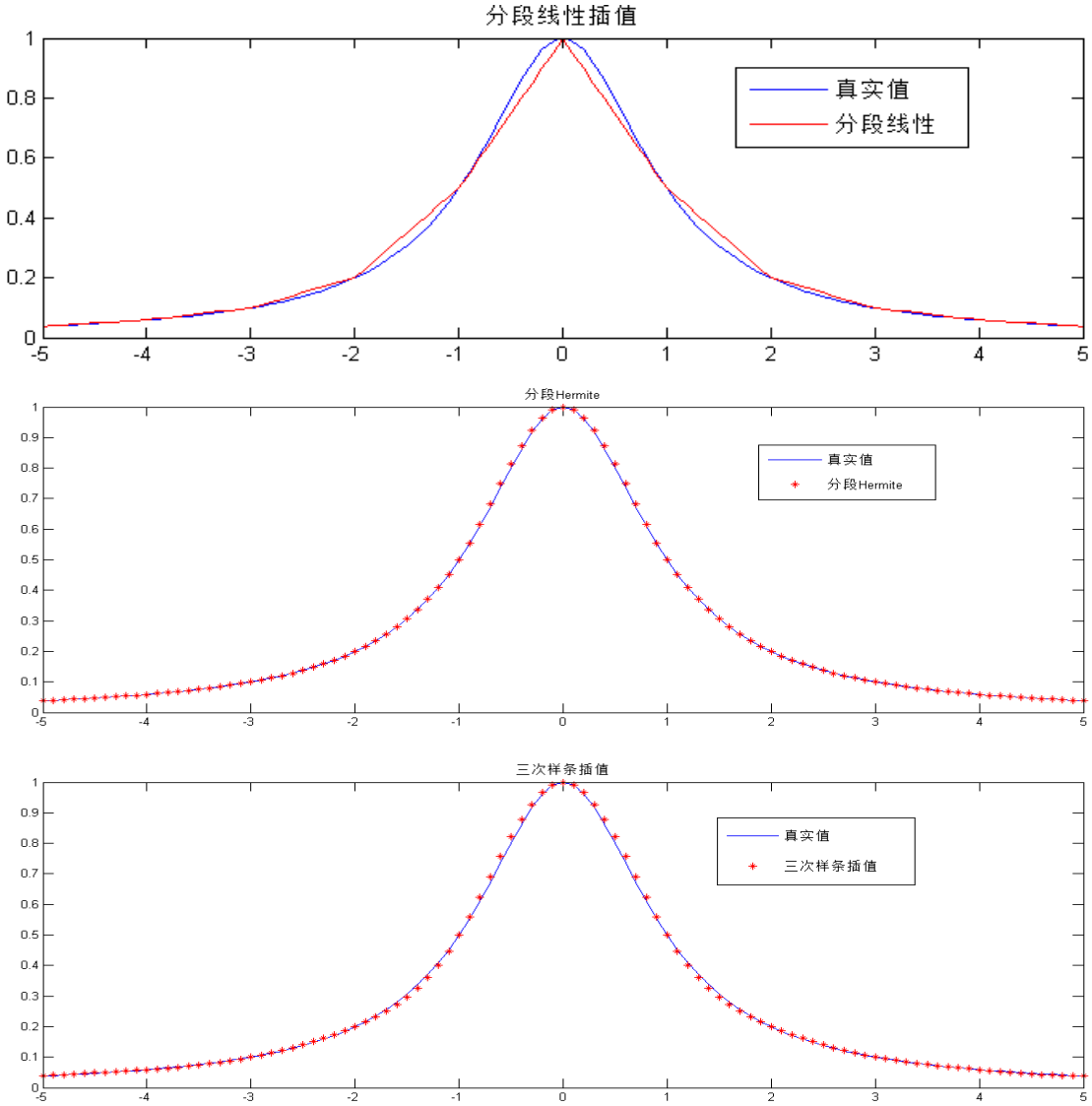
4. 在实现三次样条插值时，需要解一个线性方程组，只需注意到系数矩阵为三对角矩阵，且严格对家占优。于是，可以通过追赶法和 SOR 迭代法求解。在本次上机中，我选取了追赶法。

2.4 计算成果与分析

在上机中，首先在 c++ 中将输出定向到一个文件，然后将文件的数据在 matlab 中绘图，下面分别列出各种方法与精确值的对比图象：







由上面的图像以及程序运行的结果可以得到这样一些特点：

1. 由牛顿插值所得到的图像，可以看出的确存在理论分析上的 Runge 现象
2. 相对于等距的牛顿插值，非等距的拉格朗日插值并没有出现 Runge 现象，已经可以得到与真实函数曲线相近的图像了，但仔细观察数据，还是发现有一定的误差
3. 分段线性插值所得到的结果不是特别好，有明显的误差，以及分段线性所得的曲线并不光滑，不能满足实际应用的一些要求
4. 综合各种因素（数据，效果，算法实现的难易）来看，分段 Hermite 与三次自然样条插值所得到的结果比较好，其中三次自然样条插值所得的曲线

更加光滑

5. 从数据分析，在双精度环境下，分段 Hermite 插值与三次样条插值的精度基本达到了 10^{-3}(在区间分为 10 等分下)，进一步细分区间可以得到更好的拟合曲线

6. 综合上面的几幅图象，可以观察出当插值曲线的曲线变陡时，导数的绝对值值变大时，拟合的误差明显变大，这也与理论上分析出的截断误差表达式相符合。

2.5 本次上机反思

这次上机总结了上机中遇到的问题

1. 在本次上机发现，选择合适的插值点，使用非等距的高次插值也会得到不错的效果，所以在以后实际应用中，我们需要提前在理论上分析，然后得到合理的插值点，会起到事半功倍的效果

2. 在编写三次样条插值时，需要编写程序求解三对角方程，在这次上机中，多次由于边界条件处理不当，出现了程序的逻辑错误，所以以后在编写复杂的程序时，需要事先想清楚，然后再手动推算，这样会大大地提高程序的准确性。

3. 这次的上机程序的流程比较清楚，先写好伪码，再上机实习，难度不大，出错的几率也较小

4. 由于自己是为了编写数值算法的一些程序，自己学习了 C++，对 OOP 编程的思想理解得不是很熟悉，在总结时，发现可以将插值的算法写一个类，这样可以大大的提高代码的重复可用性

5. 在本次上机编写程序之前，在理论上进行了详细的分析，程序运行结果与理论上分析的大部分特点符合，也进一步印证了程序的正确性。所以在编写这种数值的算法之前一定要进行数学理论分析，这样既可以保证程序的逻辑正确，也可以提高自己对算法的认识。

6. 具体到本次的上机作业，综合各种因素，我认为三次样条插值方法是性价比最高的方法，在应用中应该是最有效的方法之一。

7. 本次上机还有一些收获：虽然在这个题目中，通过使用非等距插值有效的消除了振荡现象，但是实验表明，对于一般的非等距高次插值，振荡还是不能避免，为了验证这一点，我将插值点用 C++ 的随机数生成函数生成，最后得到的结果还是存在严重的振荡现象。这要求我们在实际应用中，尽量使用分段低次插值。

综上，就是本次上机的所有内容