数值分析第二周上机

By 211870125 陈睿硕

2023.2.28

§1 问题

1. 编写并测试子程序，计算 $y = x - \sin x$，使得有效位的丢失最多 1 位。
2. 计算 $y_n = \int_0^1 x^n e^x dx \quad (n \geq 0)$。
3. 考虑由
$$\begin{cases} x_0 = 1 \quad x_1 = \dfrac{1}{3} \\ x_{n+1} = \dfrac{13}{3}x_n - \dfrac{4}{3}x_{n-1} \quad (n \geq 1) \end{cases}$$
归纳定义的实数序列。将初值改为 $x_0 = 1$ 和 $x_1 = 4$ 数值稳定吗？

§2 算法思路

I 对??的解答

根据精度损失定理，我们可知当 $1 - \frac{\sin x}{x} \geq \frac{1}{2}$ 时，精度损失不超过一位。故 $|x| \geq 2$ 时，我们调用 python 中 math 库的 sin 函数直接计算即可。 $|x| \geq 2$ 时，我们对 $x - \sin x$ 进行泰勒展开：

$$x - \sin(x) = \frac{x^3}{3!} - \frac{x^5}{5!} + \frac{x^7}{7!} - \frac{x^9}{9!} + \frac{x^{11}}{11!} - \cdots$$

为了保证使用泰勒级数截断近似后不会丢失超过一位，我们研究 $x - \sin x$ 泰勒展开的拉格朗日余项：

$$R_n(x) = \frac{(-1)^{n+1}\cos(\xi)}{(n+1)!}x^{n+1} \tag{1}$$

我们希望 $|R_n(x)| \leq 2^{-24}$，事实上当 $n = 15$ 时，有 $|R_n(x)| \leq \frac{2^{15}}{15!} < 2^{-24}$。故 $|x| \geq 2$ 时，我们可以按照如下公式计算：

$$x - \sin(x) \approx \frac{x^3}{3!} - \frac{x^5}{5!} + \frac{x^7}{7!} - \frac{x^9}{9!} + \frac{x^{11}}{11!} - \frac{x^{13}}{13!}$$

我们将算法在 python 中实现后，在 $x = 10^t, t = 0, -1, -2, \cdots, -1$ 时直接计算和用泰勒展开计算 $x - \sin x$，得到如下结果：

10^{-7} 量级以上极其相近，这也就说明了泰勒展开的近似较为准确。但直接计算的方法在 10^{-7} 量级以下发生严重偏移，这正是相减相消后精度不够导致的。

$x - \sin x \sim x^3 (x \to 0)$, 这也和图**??**中图像的线性以及直线的斜率相符合。

II 对??的解答

易见 $y_0 = e - 1$，故计算 y_0 时会发生舍入误差，若按照递推公式直接迭代计算 y_n，这个舍入误差和每次迭代时计算 e 的舍入误差因多次与远大于一的数相乘而变得十分巨大，导致算法不稳定。，我们可以得到 y_n 的递推公式：

$$y_{n+1} = e - (n+1)y_n$$

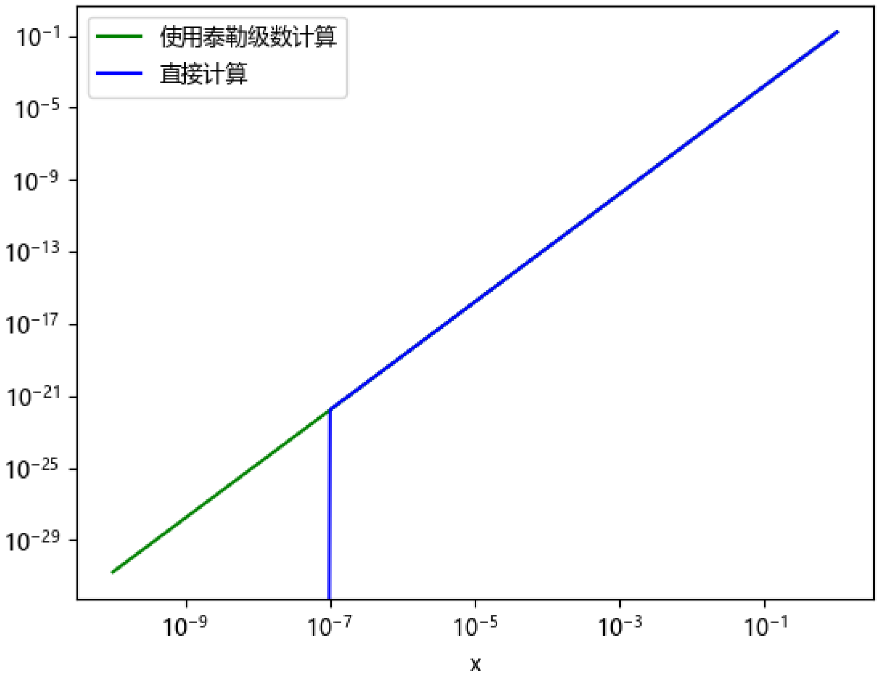

图 1: 直接计算与用泰勒展开计算结果

利用上述递推公式我们可以理论上地算出：

$$\int_0^1 x^n e^x dx = (-1)^n n!(e-1) + (-1)^n n! e \sum_{k=1}^{n} (-1)^k \frac{1}{k!}$$

注意到

$$e^{-x} - 1 = \sum_{k=1}^{\infty} \frac{(-x)^k}{k!} = \sum_{k=1}^{n} \frac{(-x)^k}{k!} + R_n(x)$$

其中，$R_n(x) = \frac{e^{-cx}}{(n+1)!} x^{n+1}, c \in (0,1)$。
故有

$$\left| \int_0^1 x^n e^x dx \right| = \left| (-1)^n n!(e-1) + (-1)^n n! e(e^{-1} - 1 - R_n(1)) \right| = \frac{e^{1-c}}{(n+1)} \le \frac{e}{n+1}$$

故当 n 足够大时（这里我们取界限为 2^{24}），我们可以认为 $y_n = 0$，并将递推公式改写为：

$$y_n = \frac{e - y_{n+1}}{n+1}$$

如此由后向前逆向递推，可以使误差不断乘远小于 1 的数从而阶乘级缩小。
y_n 十分接近 0，计算的近似值可能会在 0 的上下波动，我们对两种方法计算出的 $|y_i|, i = 1, 2, \cdots, 100$ 进行作图:
y_n 在 n 不较小的情况下表现出了指数级增长，这显然不符合 y_n 的单调变化。而逆向递推法求得的 y_n 始终平稳地减小，这是符合实际的。

III 对??的解答

使用递推公式及初值可以求得 $x_n = \frac{1}{3^n}$。使用 python 程序直接计算即可，但要注意的是应计算 $\frac{1}{3^n}$ 而非 $\left(\frac{1}{3}\right)^n$，后者会将舍入误差幂次计算，从而增大误差 (见附录代码中的) 我们使用 python 程序，

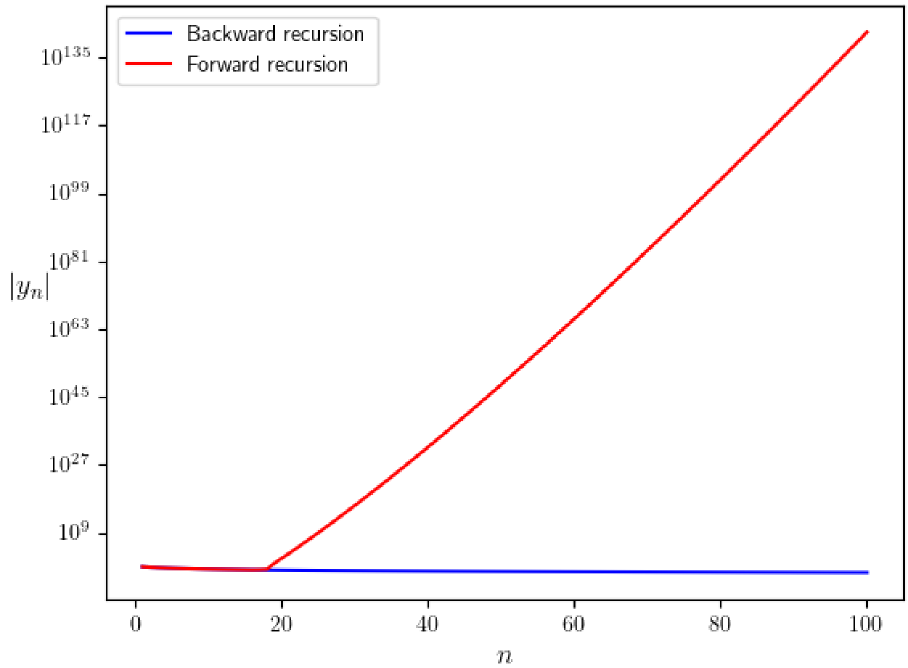

图 2: 两种递推法算出的 y_n

将数字都进行单精度表示后利用递推公式迭代计算 $x_n(n \geq 2)$，并计算其与单精度表示的 $\frac{1}{3^n}$ 之差，得到下图:

，误差呈指数型增长，在 $n = 200$ 是更是达到了惊人的 10^{97} 量级，说明递推地正向计算是不稳定的。我们可以发现，误差主要来自于非 3 倍数的整数除以 3 发生的舍入误差在数列的正向递推中不断累积（$\frac{1}{3^n}$ 计算时也发生了舍入误差，但与数列递推产生的误差相比可以忽略不计），我们记

$$x_n = \frac{1}{3^n}(1 + \varepsilon_n) \tag{2}$$

记计算 $\frac{1}{3}$ 时产生的舍入误差为 ε。易见 $\varepsilon_0 = 0$，$\varepsilon_1 = \varepsilon$。将 (??) 式代入递推公式得（事实上递推公式中计算 $\frac{13}{3}$ 与 $\frac{4}{3}$ 也会发生舍入误差，但当 n 不过小时，此舍入误差的量级远远小于 ε_n，故在下面的推导中忽略不计）:

$$\varepsilon^{n+1} = 13\varepsilon_n - 12\varepsilon_{n-1}(n \geq 1)$$

解得 $\varepsilon_n = \frac{1}{11}(12^n - 1)\varepsilon$，故可见实际误差（即 x_n 与 $\frac{1}{3^n}$ 之间的误差）满足:

$$\widehat{\varepsilon_n} = \frac{1}{11}\left[4^n - \left(\frac{1}{3}\right)^n\right]\varepsilon \approx \frac{4^n}{11}\varepsilon$$

$\widehat{\varepsilon_n}$ 确实如图??关于 n 指数增长。并且结合 Week 1 的上机作业可知，$\varepsilon_{mach} \approx 5.96046 \times 10^{-8}$，$\frac{1}{3}$ 的舍入误差 ε 量级上应比 ε_{mach} 小，于是 $\varepsilon_2 \approx \frac{16}{11}\varepsilon$ 的量级小于 10^{-8}，与图??的起始点位置相符合。

我们令 $x_0 = 1$，$x_1 = 4$，得到误差图如下:

可以看到误差也是呈指数型增长。事实上，若类似上文的定义，应有 $\varepsilon_0 = \varepsilon_1 = 0$，那为何还会发生舍入误差的累积呢？这是因为其实由于计算 $\frac{13}{3}$ 与 $\frac{4}{3}$ 时发生的舍入误差，$\varepsilon_2 \neq 0$，故同前推导，误差依然会呈指数增长。但此处与上例不同的是，折线在 $n = 15$ 处发生了一次“突变”，我认为这是 $\varepsilon_i(i = 2, 3 \cdots, 15)$ 不足以远大于计算 $\frac{13}{3}$ 与 $\frac{4}{3}$ 时发生的舍入误差导致的。

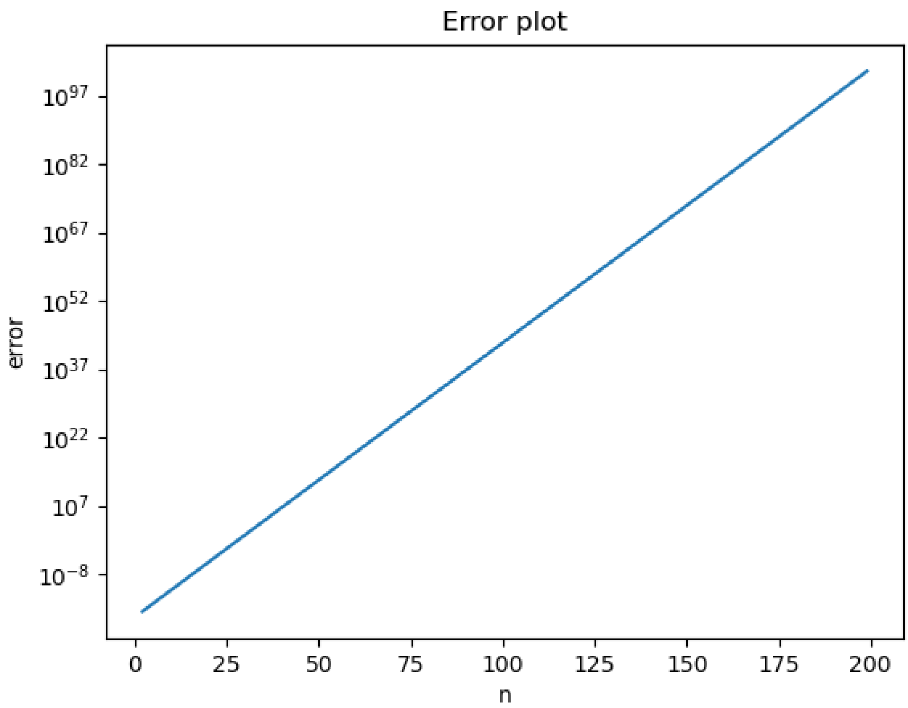

图 3: x_n 与 $\frac{1}{3^n}$ 之间的误差

§3 结果分析

§4 附录：程序代码

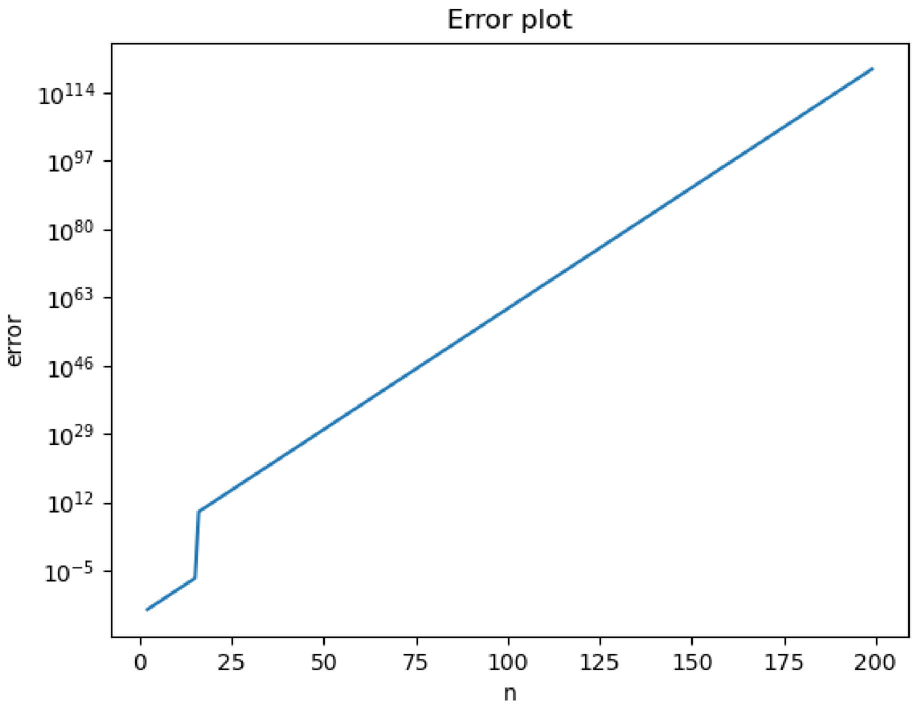

图 4: x_n 与 $\frac{1}{3^n}$ 之间的误差