$sin(x)$ 的数值计算与误差分析

陈昭熹 2017011552

2019 年 11 月 25 日

目录

1 引言

本文实现了通过数值方法求 $sin(x)$，在一开头给出本文所实现的全部算法流程简述：

逼近法 使用级数逼近 $sin(x)$

常微分方程法 通过三角函数关系 $\sin^2 x+\cos^2 x=1$，利用改进欧拉法求解关于 $sin(x)$ 的微分方程。

方程求根法 利用函数逼近 $\arcsin y$，再利用牛顿法求解 $x=\arcsin y$ 得到 y，即 $sin(x)$ 的值。

后面章节中，逐节介绍各个算法的原理、误差分析、算法流程以及计算代价和收敛速度。第 5 节介绍算法利用 C++ 的程序实现以及具体流程（以流程图形式给出），第 6 节展示一些实验结果并做总结。

2 逼近法

本节介绍利用逼近法计算 $sin(x)$ 的任意精度算法。

2.1 算法原理

众所周知，根据 $sinx$ 在 $x_0=0$ 的邻域 $(-h+x_0,h+x_0)$ 内的 Taylor 展开式，有如下式子成立:

$$\sin x=\sum_{n=0}^{\infty}\frac{(-1)^n x^{2n+1}}{(2n+1)!}=x-\frac{x^3}{3!}+\frac{x^5}{5!}-\frac{x^7}{7!}\cdots+\frac{x^{2n+1}}{(2n+1)!}+R_k(x) \tag{1}$$

注意这里写了余项形式 (为了表示方便，将展开阶次表示为 k，有 k=2n+1)，因此可以精确取等。若存在正实数 M_k 使得区间 $(-h,h)$ 上的任意 x 均有 $|f^{(k+1)}(x)|\leq M_k$，则上式中余项估计为:

$$|R_k(x)|\leq M_k\frac{h^{k+1}}{(k+1)!} \tag{2}$$

这样的一个上界估计对 x_0 的邻域内的任意 x 均成立，是一个一致估计。利用式 (1)，只需要控制展开的阶数，就可以实现任意精度的 $sinx$ 数值计算。

需要注意的是，这里有一个前提，即上述展开是在原点处的邻域内进行的，因此需要**通过 $sinx$ 的周期性，尽可能将自变量 x 变换到原点附近**，避免不满足邻域条件导致误差过大。

2.2 误差分析

2.2.1 方法误差

上一小节已给出逼近法的余项形式，这里进行误差分析。由 $sinx$ 无穷阶光滑特性及周期性可以得到

$$M_k\leq 1$$

因此有方法误差

$$|R_k(x)| \leq \frac{h^{k+1}}{(k+1)!} \tag{3}$$

事实上，这里可以通过微分中值定理得到一个精确地误差表达形式，即用 $\sin^{(k+1)}(\xi)$ 来代替 M_k，但是这样无法进行量化分析，因此进行一定的放缩，给出上界。式子中的 h 即实现时候将所有 x 值利用周期性映射到原点附近的区间 $(-h, h)$，一般为 $(-\pi/2, \pi/2)$。依据这个上界，可以通过控制多项式展开的阶数 k 来控制方法误差，在理论上达到任意精度。

值得注意的是对于偶数位精度要求，本方法具备天然高一阶的精度 (即 k=2n+1)。

2.2.2 舍入误差

假设每个阶次的计算均精确到 d 位小数，则存储带来的舍入误差为:

$$\delta_0 = \frac{1}{2} \times 10^{-d}$$

则求和带来的舍入误差为:

$$\delta = (n+1) \cdot \delta_0 = (n+1) \times \frac{1}{2} \times 10^{-d} \tag{4}$$

2.2.3 总误差

因此总误差

$$A = |R_k(x)| + \delta \leq \frac{h^{k+1}}{(k+1)!} + \frac{n+1}{2} \times 10^{-d} \tag{5}$$

2.3 算法流程

2.4 计算代价与收敛速度

计算代价方面，由于使用递归算法，因此计算阶乘和幂级数的代价均为 $O(k)$，而求和的代价也为 $O(k)$，因此总体来说还是一个线性的计算速度 $O(3k)$。由于余项收敛是泰勒展开成立的条件，因此无需担心收敛速度慢的问题，因为阶乘的增长速度远比幂级数快得多，严格计算则需要归约 $\frac{h^{k+1}}{(k+1)!}$，这个数学问题显然比本问题复杂得多。值得注意的是，当 $k+1>h$ 时，$|R_k(x)|$ 开始以 $O(a^k), 0<a<1$ 的速度收敛, 因此整体的收敛速度仍是常数 $O(h)$。这也从另一个侧面说明，在计算过程中将自变量 x 变换到较小的原点邻域内的巨大作用，若使用原始的 x 进行计算，会使得 h 值过大，导致在相同精度要求下，收敛速度急剧变慢，计算代价升高。

3 常微分方程法

本节介绍利用常微分方程法计算 $sinx$ 的任意精度算法。

3.1 算法原理

利用 $\frac{dsin(x)}{dx} = cos(x)$，可以得到下面的常微分方程组：

$$\frac{d}{dx}\begin{pmatrix} cosx \\ sinx \end{pmatrix} = \begin{pmatrix} -sinx \\ cosx \end{pmatrix} \tag{6}$$

尽管需要计算 $sinx$ 的数值，但还是存在可以加以利用的先验知识——$sin0 = 0$，利用这一点的值作为初始条件，采用改进欧拉法即可求解这一微分方程组在给定点的解。

改进欧拉法分为两个步骤：预测和矫正，若选择的区间间隔为 h，则可以表达为下式：

$$\begin{cases} \bar{y}_{n+1} = y_n + hf(x_n, y_n) \\ y_{n+1} = y_n + \frac{h}{2}[f(x_n, y_n) + f(x_{n+1}, \bar{y}_{n+1})] \end{cases} \tag{7}$$

对于本问题，有初始条件：

$$x_0 = 0, cos(x_0) = 1, sin(x_0) = 0 \tag{8}$$

虽然改进欧拉法只给出了求解一个方程的步骤，注意到本问题的微分方程组有良好的相关性，可以“串联”在一起，就可以较好的将改进欧拉法推广到上面。为表达方便将 $cos(x)$ 简写为 c, 将 $sin(x)$ 简写为 s，下标表示与 x 的含义相同，则具体做法如下：

$$\begin{aligned} \bar{c}_{n+1} &= c_n + h \times (-s_n) \\ \bar{s}_{n+1} &= s_n + h \times c_n \\ c_{n+1} &= c_n + \frac{h}{2}[(-s_n) + (-\bar{s}_{n+1})] \\ s_{n+1} &= s_n + \frac{h}{2}(c_n + \bar{c}_{n+1}) \end{aligned} \tag{9}$$

3.2 误差分析

3.2.1 方法误差

3.2.2 舍入误差

3.2.3 总误差

3.3 算法流程

3.4 计算代价与收敛速度

4 方程求根法

4.1 算法原理

4.2 误差分析

4.3 算法流程

4.4 计算代价与收敛速度

5 算法流程与实现

6 结果与总结