求 π^x

自 64　赵文亮　2016011452

2018 年 12 月 26 日

1　引言

本文实现了通过数值方法求 π^x。本文中的算法流程简述如下：

1. 求 π：使用了外推法和级数法。其中级数法的求解过程中使用了方程求根的方法。
2. 求 $\ln\pi$：使用数值积分法。
3. 求 π^x：对于一般的 x，使用常微分方程法求解；特别地，如果输入 x 为整数，则直接通过第一节的结果计算。

本文后续内容中，第 2 节至第 4 节详细地介绍了全部算法的算法原理、误差分析、算法流程、计算代价及收敛速度；第 5 节介绍了程序的结构、功能和界面；第 6 节中展示了几组求解示例，并加以分析；最后对实验过程总结。

2　求 π

首先我们对 π 进行简单估计。根据格雷戈里-莱布尼茨级数可得：

$$\pi = 4\left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} - \cdots\right) \tag{2.1}$$

从而可以有

$$\begin{aligned} \pi &< 4(1 - \frac{1}{3} + \frac{1}{5}) = \frac{52}{15} < 3.5 \\ \pi &> 4(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7}) = \frac{304}{105} > 2.8 \end{aligned} \tag{2.2}$$

下面我们通过外推法和级数法求 π。

2.1　外推法

2.1.1　算法原理

根据 $\sin x$ 在 $x_0 = 0$ 处的 Taylor 展开式，有

$$\sin(x) = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n+1}}{(2n+1)!} = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{\sin(\xi)x^9}{9!} \tag{2.3}$$

其中 $0 \leq \xi \leq x$。令 $x = \pi/n$，则有

$$n \sin\left(\frac{\pi}{n}\right) = \pi - \frac{\pi^3}{3!n^2} + \frac{\pi^5}{5!n^4} - \frac{\pi^7}{7!n^6} + \frac{\sin(\xi)\pi^9}{9!n^8} \tag{2.4}$$

令

$$f(n) = n \sin\left(\frac{\pi}{n}\right) \tag{2.5}$$

则 $|f(n) - \pi| \sim o(1/n^2)$，下面使用外推法加速。

$$\begin{cases} f(n) = \pi - \frac{\pi^3}{3!n^2} + \frac{\pi^5}{5!n^4} - \frac{\sin(\xi_1)\pi^7}{7!n^6} \\ f(2n) = \pi - \frac{1}{4} \cdot \frac{\pi^3}{3!n^2} + \frac{1}{16} \cdot \frac{\pi^5}{5!n^4} - \frac{1}{64} \cdot \frac{\sin(\xi_2)\pi^7}{7!n^6} \end{cases} \tag{2.6}$$

i 设 $g(n) = \lambda_1 f(n) + \lambda_2 f(2n)$，我们希望消去 $1/n^2$ 项即

$$\begin{cases} \lambda_1 + \lambda_2 = 1 \\ -\lambda_1 - \frac{1}{4}\lambda_2 = 0 \end{cases} \tag{2.7}$$

解得：

$$\begin{cases} \lambda_1 = -\frac{1}{3} \\ \lambda_2 = \frac{4}{3} \end{cases} \tag{2.8}$$

从而

$$\begin{aligned} g(n) &= -\frac{1}{3} f(n) + \frac{4}{3} f(2n) \\ &= \pi - \frac{1}{4} \cdot \frac{\pi^5}{5!n^4} + \left[\frac{1}{3}\sin(\xi_1) - \frac{1}{48}\sin(\xi_2)\right] \cdot \frac{\pi^7}{7!n^6} \\ &\triangleq \pi - \frac{1}{4} \cdot \frac{\pi^5}{5!n^4} + N_1 \cdot \frac{\pi^7}{7!n^6} \end{aligned} \tag{2.9}$$

其中 $N_1 = \frac{1}{3}\sin(\xi_1) - \frac{1}{48}\sin(\xi_2)$，注意到 $\sin(\xi_1) \geq 0, \sin(\xi_2) \geq 0$ 我们有

$$\begin{cases} N_1 \leq \frac{1}{3}\sin(\xi_1) \leq \frac{1}{3} \\ N_1 \geq -\frac{1}{48}\sin(\xi_2) \geq -\frac{1}{48} \end{cases} \tag{2.10}$$

注意此处的 N_1 的具体取值与 n 有关。同理有

$$g(2n) = \pi - \frac{1}{64} \cdot \frac{\pi^5}{5!n^4} + \frac{N_2}{64} \cdot \frac{\pi^7}{7!n^6} \tag{2.11}$$

其中 N_2 与 N_1 的取值范围相同。进一步，令 $h(n) = \mu_1 g(n) + \mu_2 g(2n)$，消去 $1/n^4$ 项，同理可得：

$$\begin{cases} \mu_1 + \mu_2 = 1 \\ -\frac{1}{4}\mu_1 - \frac{1}{64}\mu_2 = 0 \end{cases} \tag{2.12}$$

解得

$$\begin{cases} \mu_1 = -\frac{1}{15} \\ \mu_2 = \frac{16}{15} \end{cases} \tag{2.13}$$

可得

$$\begin{aligned} h(n) &= -\frac{1}{15}h(n) + \frac{16}{15}h(2n) \\ &= \pi + \left(\frac{N_2}{60} - \frac{N_1}{15}\right) \cdot \frac{\pi^7}{7!n^6} \\ &= \pi + M \cdot \frac{\pi^7}{7!n^6} \end{aligned} \tag{2.14}$$

其中 $M = \frac{N_2}{60} - \frac{N_1}{15}$ 的范围是：

$$\begin{cases} M \leq \frac{1}{60} \cdot \frac{1}{3} + \frac{1}{15} \cdot \frac{1}{48} = \frac{1}{144} \\ M \geq -\frac{1}{60} \cdot \frac{1}{48} - \frac{1}{15} \cdot \frac{1}{3} = -\frac{13}{576} \end{cases} \tag{2.15}$$

即有 $|M| \leq \frac{13}{576}$。

$$|h(n) - \pi| \leq M \cdot \frac{\pi^7}{7!n^6} < \frac{13}{576} \cdot \frac{3.5^7}{7!n^6} < \frac{0.03}{n^6} \tag{2.16}$$

将 $h(n)$ 的显示表达式写出，有

$$\begin{aligned} h(n) &= -\frac{1}{15}g(n) + \frac{16}{15}g(2n) \\ &= -\frac{1}{15}(-\frac{1}{3}f(n) + \frac{4}{3}f(2n)) + \frac{16}{15}(-\frac{1}{3}f(2n) + \frac{4}{3}f(4n)) \\ &= \frac{1}{45}f(n) - \frac{4}{9}f(2n) + \frac{64}{45}f(4n) \end{aligned} \tag{2.17}$$

如果我们已知 $f(2^{k-1}n_0)$，则

$$\begin{aligned} f(2^k n_0) = 2^k n_0 \sin(\frac{\pi}{2^k n_0}) &= 2^k n \sqrt{\frac{1 - \sqrt{1 - \sin^2 \frac{\pi}{2^{k-1}n_0}}}{2}} \\ &= 2^k n_0 \sqrt{\frac{1 - \sqrt{1 - \frac{f^2(2^{k-1}n_0)}{2^{2k-2}n_0^2}}}{2}} \end{aligned} \tag{2.18}$$

我们取 $n_0 = 6$，并记 $p_k = f(2^k n_0)$，则 $p_0 = f(6) = 6\sin(\pi/6) = 3$。$p$ 的递推式为

$$p_k = 2^k n_0 \sqrt{\frac{1 - \sqrt{1 - \frac{p_{k-1}^2}{2^{2k-2}n_0^2}}}{2}} \tag{2.19}$$

2.1.2 误差分析

方法误差已经由式 (2.16) 给出：

$$\Delta = |h(n) - \pi| < \frac{0.03}{n^6} \tag{2.20}$$

为了分析舍入误差，我们首先来研究 p_k 的一些性质。设 $x = \pi/n$，若 $n \geq 2$，则有 $x \in [0, \pi/2]$，那么

$$f(n) = n\sin(\frac{\pi}{n}) = \frac{\pi \sin x}{x} \triangleq \pi Sinc(x) \tag{2.21}$$

分析 $Sinc(x)$ 单调性

$$Sinc'(x) = \frac{x\cos x - \sin x}{x^2} \tag{2.22}$$

当 $x \in [0, \pi/2]$ 时，$x \le \tan x$，则 $Sinc'(x) \le 0$，$Sinc(x)$ 单调递减，所以 $f(n)$ 关于 n 单调递增，所以

$$\begin{cases} p_k \ge p_0 = 3 \\ p_k \le \lim_{i\to\infty} p_i = \pi < 3.5 \end{cases}, \forall k \ge 0 \tag{2.23}$$

$p_k \ge p_0 = 3, \forall k \ge 0$。

下面分析舍入误差，假设每次运算精确到小数点后 m_1 位，则

$$\begin{aligned}
\delta p_k &\le \max\left|\frac{\partial p_k}{\partial p_{k-1}}\right| \delta p_{k-1} + \frac{1}{2}\times 10^{-m_1} \\
&= \max\left|2^k n_0 \cdot \frac{1}{2}\left(\frac{1-\sqrt{1-\frac{p_{k-1}^2}{2^{2k-2}n_0^2}}}{2}\right)^{-\frac{1}{2}} \cdot \frac{1}{4}\left(1-\frac{p_{k-1}^2}{2^{2k-2}n_0^2}\right)^{-\frac{1}{2}} \cdot \frac{-2p_{k-1}}{2^{2k-2}n^2}\right| \delta p_{k-1} + \frac{1}{2}\times 10^{-m_1} \\
&= \frac{1}{2^k n_0}\max\left|\left(\frac{2(1+\sqrt{1-\frac{p_{k-1}^2}{2^{2k-2}n_0^2}})}{\frac{p_{k-1}^2}{2^{2k-2}n_0^2}}\right)^{\frac{1}{2}}\left(1-\frac{p_{k-1}^2}{2^{2k-2}n_0^2}\right)^{-\frac{1}{2}} p_{k-1}\right| \delta p_{k-1} + \frac{1}{2}\times 10^{-m_1} \\
&< \frac{1}{2^k n_0}\left|\left(\frac{4\cdot 2^{2k-2}n_0^2}{p_0}\right)^{\frac{1}{2}} \cdot \left(1-\frac{\pi}{2^0\times 6^2}\right)^{-\frac{1}{2}} \pi\right| \delta p_{k-1} + \frac{1}{2}\times 10^{-m_1} \\
&< 2.2\delta p_{k-1} + \frac{1}{2}\times 10^{-m_1}
\end{aligned} \tag{2.24}$$

由于 $p_0 = 3$ 为精确值，我们认为 $\delta p_0 = 0$，进而有

$$\begin{aligned}
\delta p_k + \frac{\frac{1}{2}\times 10^{-m_1}}{1.2} &< 2.2\left(\delta p_{k-1} + \frac{\frac{1}{2}\times 10^{-m_1}}{1.2}\right) < \cdots \\
&< 2.2^k\left(\delta p_0 + \frac{\frac{1}{2}\times 10^{-m_1}}{1.2}\right) \\
&= 2.2^k \cdot \frac{\frac{1}{2}\times 10^{-m_1}}{1.2}
\end{aligned} \tag{2.25}$$

即

$$\delta p_k < \frac{2.2^k - 1}{1.2} \cdot \frac{1}{2}\times 10^{-m_1} < \frac{2.2^k}{1.2} \cdot \frac{1}{2}\times 10^{-m_1} \tag{2.26}$$

我们计算 π 的方法是通过式 (2.17) 。令 $\pi_k = h(2^k n_0)$，我们有

$$\pi_k = \frac{1}{45}p_k - \frac{4}{9}p_{k+1} + \frac{64}{45}p_{k+2} \tag{2.27}$$

由式 (2.16) 可得方法误差为：

$$\Delta\pi_k < \frac{0.03}{2^{6k}n_0^6} = \frac{0.03}{2^{6k}\times 3^6} \tag{2.28}$$

舍入误差来自于 p_k 的计算，每一项的表达式都可以通过式 (2.26) 计算，可得

$$\begin{aligned}
\delta\pi_k &< \frac{1}{45}\delta p_k + \frac{4}{9}\delta p_{k+1} + \frac{64}{45}\delta p_{k+2} \\
&< \left(\frac{1}{45} + \frac{4}{9}\cdot 2.2 + \frac{64}{45}\cdot 2.2^2\right)\frac{2.2^k}{1.2} \cdot \frac{1}{2}\times 10^{-m_1} \\
&< 6.57 \cdot 2.2^k \cdot \frac{1}{2}\times 10^{-m_1}
\end{aligned} \tag{2.29}$$

2.1.3 算法流程

至此，我们已经可以给出一个求解 π 的通用算法。若要求精确到小数点后 d_1 位，则可以按照以下步骤求解：

1. 误差分配，设 $\Delta\pi_k = \frac{1}{4}\times 10^{-d_1}$，$\delta\pi_k = \frac{1}{4}\times 10^{-d_1}$
2. 根据式 (2.28) 求出
$$k = \left\lceil \frac{1}{6}\log_2 \frac{0.12\times 10^{d_1}}{3^6} \right\rceil \tag{2.30}$$
3. 根据式 (2.29) 求出
$$m_1 = \lceil d_1 + \lg 13.14 + k\lg 2.2 \rceil \tag{2.31}$$
4. 设定每次运算精确到 m_1 位小数，设定 $n_0 = 6, p_0 = 3$，利用式 (2.19) 迭代计算得到 p_k, p_{k+1}, p_{k+2}
5. 通过式 (2.27) 求出 π_k

例如，如果要求 $d_1 = 6$，则可以计算得到 $k = 2, m_1 = 8$，按照上述流程可得 $\pi \doteq 3.141593$。

2.1.4 计算代价及收敛速度

从方法误差的表达式中，可以看出 $\Delta = o(1/n^6)$，而 $n = 2^k n_0$，所以该算法得到的 π 的近似值随 k 指数收敛。从式 (2.30) 可知，$k = \mathcal{O}(d_1)$，算法中为了求 π_k 需要求解到 p_{k+2}，故总的循环次数为 $k+2$ 次，算法复杂度为 $\mathcal{O}(k+2) = \mathcal{O}(d_1)$。

2.2 级数法

本节中使用 $\arctan(\sqrt{3}/3) = \pi/6$ 来计算 π，为此，首先计算 $\sqrt{3}$。

2.2.1 计算 $\sqrt{3}$

本节使用方程求根的方法求解 $\sqrt{3}$，并做误差分析。设 $f(x) = x^2 - 3$，使用牛顿法，设
$$\begin{aligned} \phi(x) &= x - \frac{f(x)}{f'(x)} \\ &= x - \frac{x^2-3}{2x} \\ &= \frac{x}{2} + \frac{3}{2x} \end{aligned} \tag{2.32}$$
则迭代公式为
$$x_{n+1} = \phi(x_n) = \frac{x_n}{2} + \frac{3}{2x_n} \tag{2.33}$$
易求得 $\phi'(x) = \frac{1}{2} - \frac{3}{2x^2}, \phi''(x) = \frac{3}{x^3}$。由于 $f(1.731) < 0, f(1.733) > 0$，所以设初始区间为 $[1.731, 1.733]$。在此区间内，有
$$\begin{cases} |\phi'(x)| \le |\frac{1}{2} - \frac{3}{2\times 1.731^2}| < 0.0007 \\ |\phi''(x)| \le |\frac{3}{1.731^2}| < 1.1 \triangleq M \end{cases} \tag{2.34}$$

设 $x^* = \sqrt{3}, x_0 = 1.732$，方法误差 $e_n = x_n - x^*$，则 $|e_0| < 0.001$，那么

$$\begin{aligned}
|e_n| &= |\phi'(x^*)e_{n-1} + \frac{1}{2}\phi''(\xi)e_{n-1}^2| \\
&= \left|\frac{1}{2}\phi''(\xi)e_{n-1}^2\right| \le \frac{M}{2}|e_{n-1}|^2 \\
&\le \frac{M}{2}\left|\frac{M}{2}|e_{n-2}|^2\right|^2 \\
&\le \cdots \\
&\le \frac{2}{M}\left[\frac{M}{2}|e_0|\right]^{2^n} < \frac{2}{1.1}[\frac{1.1}{2}\cdot 0.001]^{2^n} < 2 \times 0.0006^{2^n}
\end{aligned} \tag{2.35}$$

下面分析舍入误差，设每次运算精确到 m_0 位小数，第 n 步舍入误差为 ϵ_n，则

$$\begin{aligned}
\epsilon_n &\le \max|\phi'(x)|\epsilon_{n-1} + \frac{1}{2}\times 10^{-m_0} \\
&\le 0.0007\epsilon_{n-1} + \frac{1}{2}\times 10^{-m_0}
\end{aligned} \tag{2.36}$$

可以认为 $\epsilon_0 = 0$，则

$$\begin{aligned}
\epsilon_n - \frac{\frac{1}{2}\times 10^{-m_0}}{0.9993} &\le 0.0007\left(\epsilon_{n-1} - \frac{\frac{1}{2}\times 10^{-m_0}}{0.9993}\right) \\
&\le \cdots \le 0.0007^n\left(\epsilon_0 - \frac{\frac{1}{2}\times 10^{-m_0}}{0.9993}\right) \\
&= -0.0007^n\frac{\frac{1}{2}\times 10^{-m_0}}{0.9993}
\end{aligned} \tag{2.37}$$

即

$$\epsilon_n \le \frac{1-0.0007^n}{0.9993}\cdot\frac{1}{2}\times 10^{-m_0} \le 0.51\times 10^{-m_0} \tag{2.38}$$

所以如果要求 $\sqrt{3}$ 精确到第 d_0 位小数，则可以按照如下步骤计算：

1. 误差分配，设 $e_n = \epsilon_n = \frac{1}{4}\times 10^{-d_0}$
2. 由式 (2.35) 求出
$$n = \left\lceil \log_2\frac{\lg 0.125 - d_0}{\lg 0.0006}\right\rceil \tag{2.39}$$
3. 由式 (2.38) 求出
$$m_0 = \lceil \lg 2.04 + d_0\rceil = d_0 + 1 \tag{2.40}$$
4. 设定每次运算精确到 m_0 位小数，初始值 $x_0 = 1.732$，按照式 (2.33) 迭代 n 步，求得 x_n 即为满足精度要求的 $\sqrt{3}$ 值。

2.2.2 算法原理

考虑设 $r(x) = \frac{1}{1+x}$ 如下 Taylor 级数：

$$\begin{aligned}
r(x) = \frac{1}{1+x} &= \sum_{n=0}^{\infty}(-1)^n x^n \\
&= \sum_{i=0}^{n}(-1)^i x^i + \frac{r^{(n+1)}(\xi)x^{n+1}}{(n+1)!} \qquad \xi\in[0,x] \\
&= \sum_{i=0}^{n}(-1)^i x^i + (-1)^{n+1}\frac{x^{n+1}}{(1+\xi)^{n+2}}
\end{aligned} \tag{2.41}$$

上式中将 x 用 x^2 代替，可得：

$$\frac{1}{1+x^2} = \sum_{i=0}^{n}(-1)^i x^{2i} + (-1)^{n+1}\frac{x^{2n+2}}{(1+\xi)^{n+2}} \qquad \xi \in [0, x^2] \tag{2.42}$$

两端取 $[0,t]$ 的定积分，可得：

$$\arctan t = \sum_{i=0}^{n}\frac{(-1)^i t^{2i+1}}{2i+1} + (-1)^{n+1}\frac{t^{2n+3}}{(2n+3)(1+\xi)^{n+2}} \qquad \xi \in [0, t^2] \tag{2.43}$$

令 $t = \sqrt{3}/3$，可得：

$$\frac{\pi}{6} = \frac{\sqrt{3}}{3}\sum_{i=0}^{n}\frac{(-1)^i}{3^i(2i+1)} + (-1)^{n+1}\frac{\sqrt{3}}{3^{n+2}(1+\xi)^{n+2}} \qquad \xi \in [0, \frac{1}{3}] \tag{2.44}$$

设

$$\pi_n = 2\sqrt{3}\sum_{i=0}^{n}\frac{(-1)^i}{3^i(2i+1)} \tag{2.45}$$

则由式 (2.44) 可知，当 $n \to \infty$ 时，$\pi_n \to \pi$。

2.2.3 误差分析

首先分析方法误差。由式 2.44 可知：

$$\begin{aligned}\Delta = |\pi_n - \pi| &= \left|(-1)^{n+1}\frac{6\sqrt{3}}{3^{n+2}(1+\xi)^{n+2}}\right| \qquad \xi \in [0, \frac{1}{3}] \\ &< \frac{4}{3^{n+1}}\end{aligned} \tag{2.46}$$

其次分析舍入误差。假设每次计算均精确到 m_1 位小数，则由于求和带来的舍入误差为：

$$\delta_1 = n \cdot \frac{1}{2} \times 10^{-m_1} \tag{2.47}$$

另外，第 2.2.1 节求出的 $\sqrt{3}$ 的数值解也存在误差，该误差也会在此步骤产生舍入误差。若 $\sqrt{3}$ 的近似值精确到第 d_0 位小数，则

$$\begin{aligned}\delta_2 &\le \left|\frac{1}{2} \times 10^{-d_0} \times 2\sum_{i=0}^{n}\frac{(-1)^i}{3^i(2i+1)}\right| \\ &= \frac{\frac{1}{2}\times 10^{-d_0}}{\sqrt{3}} \times \pi_n \\ &\le \frac{\frac{1}{2}\times 10^{-d_0}}{\sqrt{3}} \times (\pi + \frac{4}{3^{n+1}}) \\ &< \frac{\frac{1}{2}\times 10^{-d_0}}{1.7} \times (3.2 + \frac{4}{3^{n+1}}) \\ &< (1 + \frac{1.2}{3^{n+1}}) \times 10^{-d_0} \le 1.4 \times 10^{-d_0}\end{aligned} \tag{2.48}$$

则总误差

$$\begin{aligned}A &= \Delta + \delta_1 + \delta_2 \\ &< \frac{4}{3^{n+1}} + n \cdot \frac{1}{2} \times 10^{-m_1} + 1.4 \times 10^{-d_0}\end{aligned} \tag{2.49}$$

2.2.4 算法流程

在精确到 d_1 位小数的要求下，则使用级数法计算 π 的流程为：

1. 误差分配，令 $\Delta = \delta_1 = \delta_2 = \frac{1}{8} \times 10^{-d_1}$
2. 由式 (2.46) 计算出
$$n = \lceil \log_3(32 \times 10^{d_1}) - 1 \rceil \tag{2.50}$$
3. 由式 (2.47) 计算出
$$m_1 = \lceil d_1 + \lg(4n) \rceil \tag{2.51}$$
4. 由式 (2.48) 计算出
$$d_0 = \lceil d_1 + \lg 1.4 \rceil = d_1 + 1 \tag{2.52}$$
5. 给定精度要求 d_0，利用第 2.2.1 节的算法求出 $\sqrt{3}$ 的近似值（记作 $\tilde{S}_3$）
6. 固定每次计算精确到小数点后 m_1 位，使用式 (2.45) 计算 π 的近似值（其中 $\sqrt{3}$ 用 $\tilde{S}_3$ 近似）。

2.3 计算代价及收敛速度

首先分析求解 $\sqrt{3}$ 的近似值的过程。由于使用了牛顿法，求解 $\sqrt{3}$ 的过程中收敛非常快，误差 $|e_n| = o(0.006^{2^n})$。给定 d_0，则只需迭代 $\mathcal{O}(\log d_0)$ 次即可得到满足要求的 $\sqrt{3}$ 近似值。另一方面，级数求和收敛速度为 $o(1/3^n)$，求和项数 $n = \mathcal{O}(d_1)$。综上，级数法求解复杂度为 $\mathcal{O}(d_1) + \mathcal{O}(\log d_0) = \mathcal{O}(d_1)$，与外推法基本类似。

3 求 $\ln \pi$

3.1 算法原理

本节中将实现 $\ln \pi$ 的求解。注意到
$$\ln x = \int_1^x \frac{1}{t}\,\mathrm{d}t \tag{3.1}$$
所以我们可以通过数值积分的方法来求解 $\ln \pi$，本节中使用复化辛甫生公式来求解。设上一节中求得的 π 的近似值为 $\tilde{\pi}$，不妨设 $\tilde{\pi}$ 精确到六位小数（即 $d_1 \geq 6$），将区间 $[1, \tilde{\pi}]$ 进行 n 等分，得到 n 个子区间为
$$[x_k, x_{k+1}], \quad k = 0, 1, 2, \cdots n-1 \tag{3.2}$$
区间长度 $h = (\tilde{\pi} - 1)/n$。记 $u(x) = \frac{1}{x}$，则
$$\begin{aligned} I_n &= \sum_{k=0}^{n-1} \int_{x_k}^{x_{k+1}} u(x)\,\mathrm{d}x \\ &\doteq \sum_{k=0}^{n-1} \frac{h}{6}\left[u(x_k) + 4u(x_k + \frac{h}{2}) + u(x_{k+1})\right] \\ &= \frac{h}{6}\left[u(1) + 2\sum_{k=1}^{n-1} u(x_k) + 4\sum_{k=0}^{n-1} u(x_k + \frac{h}{2}) + u(\tilde{\pi})\right] \end{aligned} \tag{3.3}$$

3.2 误差分析

由数值积分带来的方法误差为

$$\begin{aligned} R[u] &= \sum_{k=0}^{n-1} R_k[u] \\ &= \sum_{k=0}^{n-1} -\frac{h^5}{2880} u^{(4)}(\eta_k) \qquad \eta_k \in [x_k, x_{k+1}] \\ &= -\frac{h^5}{2880} \cdot n u^{(4)}(\eta) \qquad \eta \in [1, \tilde{\pi}] \\ &= -\frac{(\tilde{\pi}-1)^5}{2880 n^4} u^{(4)}(\eta) \end{aligned} \tag{3.4}$$

其中

$$u^{(4)}(x) = \frac{24}{x^5} \tag{3.5}$$

则方法误差为

$$\begin{aligned} \Delta = |R[u]| &\le \max_{\eta \in [1,\tilde{\pi}]} \left| \frac{(\tilde{\pi}-1)^5}{2880 n^4} u^{(4)}(\eta) \right| \\ &= \max_{\eta \in [1,\tilde{\pi}]} \left| \frac{(\tilde{\pi}-1)^5}{2880 n^4} \frac{24}{\eta^5} \right| \\ &\le \frac{(3.1416-1)^5 \times 24}{2880 n^4} \\ &< \frac{0.38}{n^4} \end{aligned} \tag{3.6}$$

假设每次计算精确到 m_2 位小数，则 $hu(x_k)$ 的舍入误差为

$$\begin{aligned} \delta[hu(x_k)] &\le h \max_{x \in [1,\tilde{\pi}]} \left| \frac{\partial u}{\partial x} \right| \delta x_k + \max_{x_k \in [1,\tilde{\pi}]} |u(x_k)| \delta h \\ &\le h \max_{x \in [1,\tilde{\pi}]} \left(\frac{1}{x^2} + \frac{1}{x} \right) \cdot \frac{1}{2} \times 10^{-m_2} \\ &\le \left(\frac{2.15}{n} + 1 \right) \cdot \frac{1}{2} \times 10^{-m_2} \end{aligned} \tag{3.7}$$

$hu(x_k + \frac{h}{2})$ 的舍入误差为

$$\begin{aligned} \delta[hu(x_k + \frac{h}{2})] &\le \max \left| \frac{\partial hu(x_k + h/2)}{\partial h} \right| \delta h + \max \left| \frac{\partial hu(x_k + h/2)}{\partial x_k} \right| \delta x_k \\ &= \max \left| \frac{x_k}{(x_k + \frac{h}{2})^2} \right| \delta h + \max \left| \frac{h}{(x_k + \frac{h}{2})^2} \right| \delta x_k \\ &\le \left(\frac{3.15}{1^2} + \frac{(3.15-1)/n}{1} \right) \times \frac{1}{2} \times 10^{-m_2} \\ &= \left(3.15 + \frac{2.15}{n} \right) \cdot \frac{1}{2} \times 10^{-m_2} \end{aligned} \tag{3.8}$$

由式 (3.3) 可得，由数值积分带来的舍入误差为；

$$\begin{aligned} \delta_1 &\le \frac{1}{6} \left[2n\delta[hu(x_k)] + 4n\delta[hu(x_k + \frac{h}{2})] \right] \\ &= \frac{1}{6} \left(2n \cdot \left(\frac{2.15}{n} + 1 \right) \cdot \frac{1}{2} \times 10^{-m_2} + 4n \cdot \left(3.15 + \frac{2.15}{n} \right) \cdot \frac{1}{2} \times 10^{-m_2} \right) \\ &< (2.5n + 2.2) \cdot \frac{1}{2} \times 10^{-m_2} \end{aligned} \tag{3.9}$$

另一方面，由于 $\tilde{\pi}$ 不准也会带来舍入误差，设 $\tilde{\pi}$ 精确到 d_1 位小数，则

$$\begin{aligned}\delta_2 &= \left|\int_{\tilde{\pi}}^{\pi} \frac{1}{x}\,\mathrm{d}x\right| = \left|\frac{1}{\xi}\int_{\tilde{\pi}}^{\pi} \mathrm{d}x\right| \qquad \xi \in [\tilde{\pi}, \pi] \\ &\le \frac{1}{\xi}|\tilde{\pi} - \pi| \\ &\le \frac{1}{6} \times 10^{-d_1}\end{aligned} \tag{3.10}$$

其中最后一步利用了 $\xi \ge \tilde{\pi} \ge 3$。则总误差为

$$\begin{aligned}A &= \Delta + \delta_1 + \delta_2 \\ &< \frac{0.38}{n^4} + (2.5n + 2.2) \cdot \frac{1}{2} \times 10^{-m_2} + \frac{1}{6} \times 10^{-d_1}\end{aligned} \tag{3.11}$$

3.3 算法流程

若要求 $\ln\pi$ 的结果精确到 d_2 位小数，则可以按照如下流程计算：

1. 误差分配，设 $\Delta = \delta_1 = \delta_2 = \frac{1}{8} \times 10^{-d_2}$
2. 根据式 (3.6) 求出
$$n = \left\lceil \sqrt[4]{3.04 \times 10^{d_2}} \right\rceil \tag{3.12}$$
3. 根据式 (3.9) 求出
$$m_2 = \lceil d_2 + \lg(10n + 8.8) \rceil \tag{3.13}$$
4. 根据式 (3.10) 求出
$$d_1 = \max\left\{\left\lceil d_2 + \lg\frac{4}{3}\right\rceil, 6\right\} = \max\{d_2 + 1, 6\}; \tag{3.14}$$
5. 由第 2 节的算法计算精确到 d_1 位小数的 $\tilde{\pi}$
6. 将 $[1, \tilde{\pi}]$ 区间分为 n 份，利用式 (3.3) 的复化辛甫生公式求出 $\ln\pi$ 的近似值。

例如，令 $d_2 = 6$，则可以求出 $n = 42, m_2 = 9, d_1 = 7$，计算得到 $\ln\pi$ 的近似值为 $\tilde{\ln}\pi = 1.144730$。

3.4 计算代价及收敛速度

本算法中误差量级为 $o(1/n^4)$，即随 n^4 收敛。子区间个数 $n = \mathcal{O}(10^{d_2/4})$，即随精确位数增加而指数增加。计算中循环次数为 n 次，所以计算复杂度为 $\mathcal{O}(n) = \mathcal{O}(10^{d_2/4})$。

4 求 π^x

设 $a = x\ln\pi$，则 $\pi^x = e^{x\ln\pi} = e^a$。假设上一节中求得的 $\ln\pi$ 的近似值为 $\tilde{\ln}\pi$，且精确到六位小数（$d_2 \ge 6$）。令 $\tilde{a} = x\tilde{\ln}\pi$，对于一般的 x，本节将通过常微分方程的数值解来求出 $e^{\tilde{a}}$。另一方面，如果输入的 x 为整数，我们可以直接利用第 2 节的结果来计算（见第 4.4 节）。

4.1 算法原理

考虑下面的常微分方程：

$$\begin{cases} y' = y \triangleq f(x,y) \\ y(0) = 1 \end{cases} \tag{4.1}$$

不难求出，该常微分方程的解为 $y = e^x$。将区间 $[0, \tilde{a}]$ 进行 N 等分，区间长度为 $h = \tilde{a}/N$。则 $h \geq 1.144x/N$。首先对 $e^{\tilde{a}}$ 进行简单的估计：

$$\begin{aligned} e^{\tilde{a}} &= e^{x \tilde{\ln \pi}} \leq e^{x \ln \pi + x \cdot \frac{1}{2} \times 10^{-6}} \\ &\leq \pi^x \left(3^{\frac{1}{2} \times 10^{-6}}\right)^x \\ &< (3.15 \times 1.01)^x \\ &< 3.2^{\lceil x \rceil} \triangleq E_{\tilde{a}} \end{aligned} \tag{4.2}$$

上式中对 x 向上取整是为了便于指数计算，后文中使用 $E_{\tilde{a}}$ 来表示 $e^{\tilde{a}}$ 的一个上界估计值。同理可得 $e^h \leq 3.2^{\lceil x/N \rceil} \triangleq E_h$。使用四阶龙格——库塔公式求解：

$$\begin{cases} y_{n+1} = y_n + \dfrac{h}{6}(K_1 + 2K_2 + 2K_3 + K_4) \\ K_1 = f(x_n, y_n) = y_n \\ K_2 = f(x_n + \dfrac{h}{2}, y_n + \dfrac{h}{2}K_1) = y_n + \dfrac{h}{2}K_1 \\ K_3 = f(x_n + \dfrac{h}{2}, y_n + \dfrac{h}{2}K_2) = y_n + \dfrac{h}{2}K_2 \\ K_4 = f(x_{n+1}, y_n + hK_3) = y_n + hK_3 \end{cases} \tag{4.3}$$

其中 $x_n = na/N$。

4.2 误差分析

将 $y(x_{n+1})$ 在 x_n 处 Taylor 展开，可得

$$\begin{aligned} y(x_{n+1}) &= y(x_n) + hy'(x_n) + \frac{h^2}{2!}y^{(2)}(x_n) + \frac{h^3}{3!}y^{(3)}(x_n) + \frac{h^4}{4!}y^{(4)}(x_n) + \frac{h^5}{5!}y^{(5)}(\xi_n) \\ &= y(x_n)(1 + h + \frac{h^2}{2!} + \frac{h^3}{3!} + \frac{h^4}{4!}) + y(\xi_n)\frac{h^5}{5!} \qquad \xi_n \in [x_n, x_{n+1}] \end{aligned} \tag{4.4}$$

由式 (4.3) 可得：

$$\begin{cases} K_2 = y_n + \dfrac{h}{2}y_n = \left(1 + \dfrac{h}{2}\right) y_n \\ K_3 = y_n + \dfrac{h}{2}(y_n + \dfrac{h}{2}y_n) = \left(1 + \dfrac{h}{2} + \dfrac{h^2}{4}\right) y_n \\ K_4 = y_n + h\left(1 + \dfrac{h}{2} + \dfrac{h^2}{4}\right) y_n = \left(1 + h + \dfrac{h^2}{2} + \dfrac{h^3}{4}\right) y_n \end{cases} \tag{4.5}$$

以及

$$\begin{aligned} y_{n+1} &= y_n + \frac{h}{6}\left(y_n + 2\left(1 + \frac{h}{2}\right) y_n + 2\left(1 + \frac{h}{2} + \frac{h^2}{4}\right) y_n + \left(1 + h + \frac{h^2}{2} + \frac{h^3}{4}\right) y_n\right) \\ &= y_n \left(1 + h + \frac{h^2}{2} + \frac{h^3}{6} + \frac{h^4}{24}\right) \end{aligned} \tag{4.6}$$

则有

$$\begin{aligned}|y(x_{n+1})-y_n| &= |\left(y(x_n)-y_n\right)\left(1+h+\frac{h^2}{2!}+\frac{h^3}{3!}+\frac{h^4}{4!}\right)+y(\xi_n)\frac{h^5}{5!}| \\ &\leq |y(x_n)-y_n|e^h+\left|y(\xi_n)\frac{h^5}{5!}\right|\end{aligned} \tag{4.7}$$

设累积到第 n 步的方法误差 $\Delta_n=|y(x_{n+1})-y_n|$，则有

$$\begin{aligned}\Delta_{n+1} &\leq \Delta_n e^h+\frac{h^5}{5!}e^{\xi_n} \\ &\leq \Delta_n e^h+\frac{h^5}{5!}e^{x_{n+1}} \\ &= \Delta_n e^h+\frac{h^5}{5!}e^{(n+1)h}\end{aligned} \tag{4.8}$$

利用两边除以 $e^{(n+1)h}$，可得

$$\frac{\Delta_{n+1}}{e^{(n+1)h}} \leq \frac{\Delta_n}{e^{nh}}+\frac{h^5}{5!} \tag{4.9}$$

可以认为 $\Delta_0=0$，则有

$$\frac{\Delta_N}{e^{Nh}} \leq N\frac{h^5}{5!} \tag{4.10}$$

则最终的方法误差为：

$$\begin{aligned}\Delta=\Delta_N \leq Ne^{Nh}\frac{h^5}{5!} &= \frac{e^{\tilde{a}}\tilde{a}^5}{120N^4}=\frac{e^{x\tilde{\ln\pi}}x^5\tilde{\ln\pi}^5}{120N^4} \\ &\leq \frac{E_{\tilde{a}}x^5(\ln\pi+\frac{1}{2}\times 10^{-6})^5}{120N^4} \\ &< \frac{1}{120N^4}\left(E_{\tilde{a}}x^5\cdot 1.1448^5\right) \\ &< \frac{1}{120N^4}\left(E_{\tilde{a}}x^5\cdot 2\right) \\ &< \frac{E_{\tilde{a}}x^5}{60N^4}\end{aligned} \tag{4.11}$$

假设每次计算精确到 m_3 位小数，则由式 (4.6) 可得，迭代到第 $n+1$ 步中常微分方程数值解中产生的舍入误差为：

$$\begin{aligned}\delta y_{n+1} &= \max\left|1+h+\frac{h^2}{2}+\frac{h^3}{6}+\frac{h^4}{24}\right|\delta y_n+y_n\max\left|1+h+\frac{h^2}{2}+\frac{h^3}{6}\right|\delta h+\frac{1}{2}\times 10^{-m_3} \\ &< e^h\delta y_n+(y_Ne^h+1)\cdot\frac{1}{2}\times 10^{-m_3}\end{aligned} \tag{4.12}$$

令

$$q=(y_Ne^h+1)\cdot\frac{1}{2}\times 10^{-m_3} \tag{4.13}$$

$$\delta y_{n+1}+\frac{q}{e^h-1}<e^h\left(\delta y_n+\frac{q}{e^h-1}\right) \tag{4.14}$$

可以认为 $\delta y_0=0$，则

$$\delta y_N+\frac{q}{e^h-1}<e^{Nh}\left(\delta y_0+\frac{q}{e^h-1}\right)=e^{Nh}\frac{q}{e^h-1} \tag{4.15}$$

即

$$\begin{aligned}\delta y_N &< \frac{q}{e^h-1}\left(e^{Nh}-1\right) \\ &= (y_N e^h+1)\cdot\frac{1}{2}\times 10^{-m_3}\cdot\frac{e^{\tilde{a}}-1}{e^h-1} \\ &< \left((e^{\tilde{a}}+\Delta)E_h+1\right)\cdot\frac{e^{\tilde{a}}-1}{e^h-1}\cdot\frac{1}{2}\times 10^{-m_3} \\ &< ((E_{\tilde{a}}+\Delta)\times E_h+1)\cdot\frac{E_{\tilde{a}}-1}{1.144x/N}\cdot\frac{1}{2}\times 10^{-m_3}\end{aligned} \tag{4.16}$$

记 $\delta_1=\delta y_N$，表示由常微分方程数值解带来的舍入误差。另一方面，由于 $\tilde{\ln\pi}$ 不准带来的舍入误差为：

$$\begin{aligned}\delta_2 &= |e^a-e^{\tilde{a}}| = e^{\xi_a}|a-\tilde{a}|, \qquad \xi_a\in[a,\tilde{a}] \\ &\le e^{\tilde{a}}|a-\tilde{a}| \\ &< xE_{\tilde{a}}\cdot\frac{1}{2}\times 10^{-d2}\end{aligned} \tag{4.17}$$

总误差为

$$\begin{aligned}A &= \Delta+\delta_1+\delta_2 \\ &= \frac{E_{\tilde{a}}x^5}{60N^4}+((E_{\tilde{a}}+\Delta)E_h+1)\cdot\frac{E_{\tilde{a}}-1}{1.144x/N}\cdot\frac{1}{2}\times 10^{-m_3}+xE_{\tilde{a}}\cdot\frac{1}{2}\times 10^{-d2}\end{aligned} \tag{4.18}$$

4.3 算法流程

给定 x，如果要求 π^x 精确到 d_3 位小数，则按照以下流程计算：

1. 误差分配，设 $\Delta=\delta_1=\delta_2=\frac{1}{8}\times 10^{-d_3}$
2. 计算 $e^{\tilde{a}}$ 的上界估计值 $E_{\tilde{a}}=3.2^{\lceil x\rceil}$
3. 由式 (4.11) 求得
$$N=\left\lceil\sqrt[4]{10^{d_3}\times\frac{2E_{\tilde{a}}x^5}{15}}\right\rceil \tag{4.19}$$
4. 由式 (4.16) 求得
$$m_3=\left\lceil d_3+\lg\frac{4(E_{\tilde{a}}-1)((E_{\tilde{a}}+\Delta)E_h+1)}{1.144x/N}\right\rceil \tag{4.20}$$
5. 由式 (4.17) 求得
$$d_2=\max\{\lceil d_3+\lg 4xE_{\tilde{a}}\rceil, 6\} \tag{4.21}$$
6. 给定 d_2 的精度要求，根据第 3 节的算法求出 $\tilde{\ln\pi}$
7. 给定 $y_0=1$，$\tilde{a}=x\tilde{\ln\pi}$，由式 (4.6) 逐步迭代求解出 y_N，即为 π^x 的近似值

4.3.1 计算代价及收敛速度

由方法误差表达式可知，该算法以 $1/N^4$ 速度收敛。给定 d_3，则循环次数为 $N=\mathcal{O}(10^{d_3/4})$。所以计算代价也为 $\mathcal{O}(10^{d_3/4})$ 量级。

4.4 x 为整数时的快速算法

4.4.1 算法原理和误差分析

若 x 为整数，则可以直接利用第 2 节求得的近似值 $\tilde{\pi}$ 来求解，下面简要分析误差。首先显然可知，该算法不存在方法误差。连乘过程可以表示为：

$$\begin{cases} w_0 = 1 \\ w_k = \tilde{\pi} w_{k-1}, \qquad k = 1, 2, \ldots, x \end{cases} \tag{4.22}$$

则 $w_k = \tilde{\pi} w_{k-1}$。若每次计算精确到 m_3 位小数，则每次乘积的舍入误差为：

$$\begin{aligned} \delta w_k &\le \tilde{\pi} \delta w_{k-1} + \frac{1}{2} \times 10^{-m_3} \\ &= 3.15 \delta w_{k-1} + \frac{1}{2} \times 10^{-m_3} \end{aligned} \tag{4.23}$$

则

$$\begin{aligned} \delta w_k + \frac{\frac{1}{2} \times 10^{-m_3}}{2.15} &\le 3.15 \left(\delta w_{k-1} + \frac{\frac{1}{2} \times 10^{-m_3}}{2.15} \right) \\ &\le \cdots \\ &\le 3.15^k \left(\delta w_0 + \frac{\frac{1}{2} \times 10^{-m_3}}{2.15} \right) \end{aligned} \tag{4.24}$$

认为 $\delta w_0 = 0$，则由乘积带来的总舍入误差 δ_1 为：

$$\delta_1 = \delta w_x = \frac{3.15^x - 1}{2.15} \cdot \frac{1}{2} \times 10^{-m_3} \le \frac{3.15^x}{2.15} \cdot \frac{1}{2} \times 10^{-m_3} \tag{4.25}$$

$\tilde{\pi}$ 不准也会产生舍入误差，若 $\tilde{\pi}$ 精确到 d_1 位小数，则

$$\begin{aligned} \delta_2 &= |\pi^x - \tilde{\pi}^x| \le \max |x \pi^{x-1}| |\pi - \tilde{\pi}| \\ &\le x \times 3.15^{x-1} \times \frac{1}{2} \times 10^{-d_1} \end{aligned} \tag{4.26}$$

所以总误差为：

$$A = \delta_1 + \delta_2 = \frac{3.15^x - 1}{2.15} \cdot \frac{1}{2} \times 10^{-m_3} + x \times 3.15^{x-1} \times \frac{1}{2} \times 10^{-d_1} \tag{4.27}$$

4.4.2 算法流程

若要求精确到 d_3 位小数，则按照如下流程计算：

1. 误差分配，设 $\delta_1 = \delta_2 = \frac{1}{4} \times 10^{-d_3}$
2. 由式 (4.25) 计算出：
$$m_3 = \lceil d_3 + x \lg 3.15 \rceil \tag{4.28}$$
3. 由式 (4.26) 计算出：
$$d_1 = \left\lceil d_3 + 2x \cdot 3.15^{x-1} \right\rceil \tag{4.29}$$
4. 给定 d_1 的要求，根据第 2 节的算法求解出 $\tilde{\pi}$
5. 设定每步精确到 m_3 位小数，按照式 (4.22) 求出 π^x 的近似值。

4.4.3 计算代价及收敛速度

本算法由于没有方法误差，无需讨论收敛速度。计算代价即为 $\mathcal{O}(x)$。

5 程序编写

基于前文所述的求解算法，我使用 C++ 编写了求解程序，并包括用户界面。本节将介绍程序结构和功能等内容。

5.1 程序结构与框图

本程序主要分为四个求解类：Pi 用来求解 π，LnX 用来求解 $\ln\pi$，EXP 用来求解 π^x，Sqrt3 用来求解 $\sqrt{3}$（在级数法求解 π 的过程中用到）。每个求解类中都有 init 和 calc 两个方法。init 方法输入给定误差要求，通过误差分析的结果初始化内部参数（例如迭代次数，运算精度等等），calc 函数按照算法的流程进行计算。程序主要的流程图如图 1 所示，其中每一步的初始化和求解的具体过程在前文相应章节中已经有完整的表述，故此处不再赘述。另一方面，由于 x 为整数时的特殊处理比较简单，且关键步骤在图 1 中也有体现（求 π 的初始化以及计算），故不再通过流程图表述。

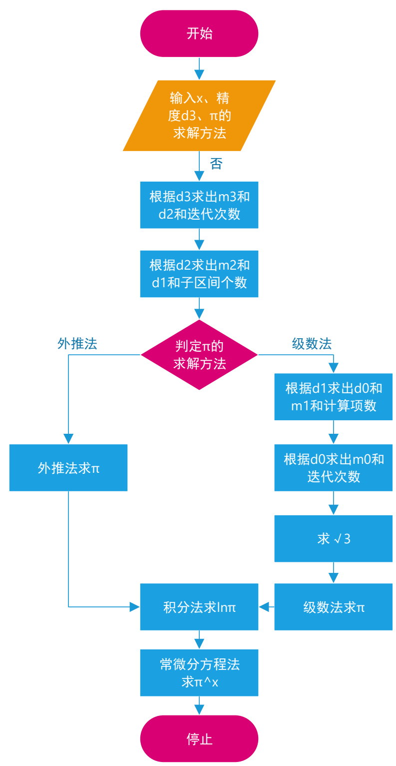
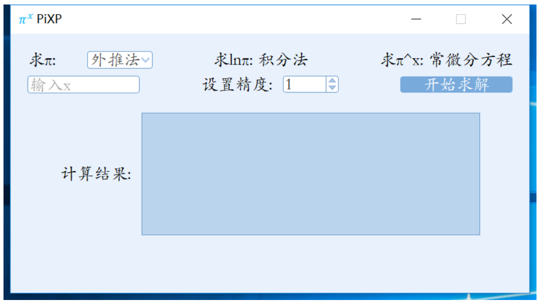


图 1: 程序主体流程图

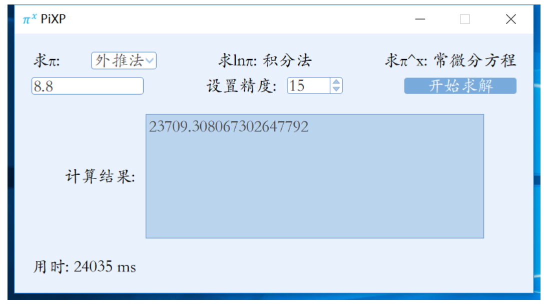
此外值得说明的是，为了实现任意精度的求解，我使用了高精度计算库 mpfr。该运算库使用字符串保存数字，并可以设定精度值。

5.2 程序界面与功能

主程序如图 2 所示。用户可以选择求 π 的两种方法：外推法和级数法。在第二行左侧的输入框中，用户可以输入 $[1,10]$ 范围内的 x（使用正则表达式限制）；用户通过上下箭头来设置精度。若 x 为整数，则精度范围为 $[1,100]$，否则为 $[1,15]$。用户在输入框中点击回车或按下开始求解按钮均可以启动求解过程。为了防止求解过程阻塞 UI 线程，我将求解的过程放在一个新的线程进行。



(a) 程序初始界面

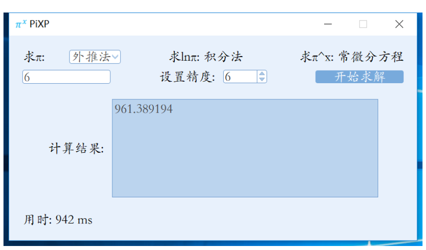


(b) 运算结束界面

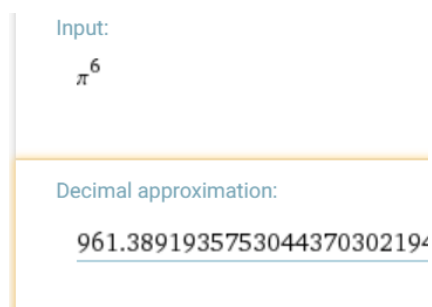
图 2: 程序截图

6 实验结果及分析

由于编写了良好的用户界面，我可以方便地测试不同精度要求的实验结果。我使用 Wolfram Alpha 上的计算结果进行验证。例如，给定精度为 6，$x = 6$，实验结果如图 3a 所示，可见求得的结果为 961.389194。使用 Wolfram Alpha 计算的结果如图 3a 所示，四舍五入到六位小数之后结果相同。



(a) π^6 计算结果

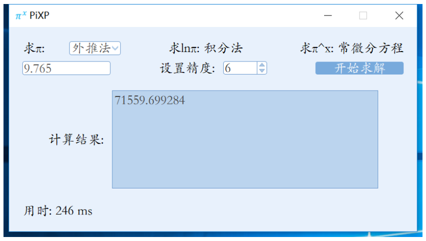


(b) Wolfram Alpha π^6 计算结果

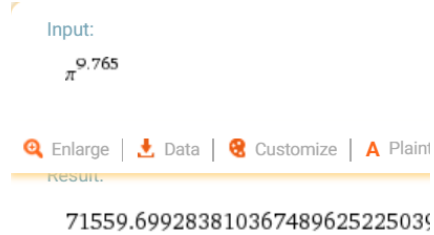
图 3: π^6 计算结果验证

再测试一组小数输入的情况。令 $x = 9.765$，$d = 6$，求得结果为 71559.699284，经 Wolfram Alpha 验证得知正确。

最后，我们再来尝试以下最大精度的情况。为了直观起见，令 $x = 1$，$d = 100$，即可求解 π 的小数点后 100 位。结果如图 5 所示。通知可以通过运行时间来对比两种求 π 算法的效率，可见这两种算法的耗时相当，这也与之前运算代价与收敛性分析的部分相吻合。

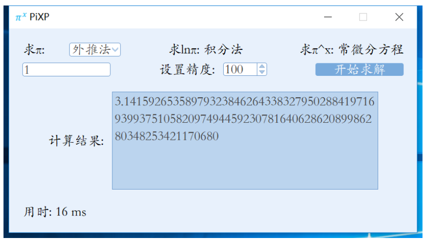


(a) $\pi^{9.765}$ 计算结果

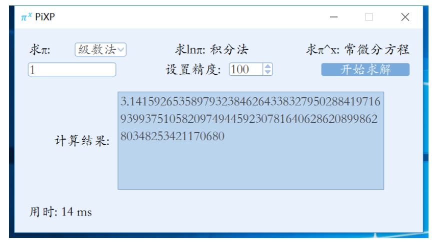


(b) Wolfram Alpha $\pi^{9.765}$ 计算结果

图 4: $\pi^{9.765}$ 计算结果验证



(a) 外推法求解 π100 位小数



(b) 级数法求解 π100 位小数

图 5: 求解 π 的 100 位小数

7 实验总结

通过本次实验，我温习了课上讲过的多种数值分析算法：外推法加速、Taylor 级数、牛顿法求根、数值积分、求解常微分方程等算法。在误差分析的过程中，我更加熟悉了方法误差、舍入误差的分析方法。

本次大作业中，我的完成方法是首先设计算法，其次进行充分的误差分析，再通过误差分析的结果来完成代码。我认为这样的顺序能很好地指导代码中的一些参数的设置，果然取得了较好的效果。

此外，由于我一开始就想做无限精度的求解问题，我查找了许多资料，最终选择了 mpir 和 mpfr 库来实现高精度运算。这些库在网上难以找到，只能从源码编译。我在这个过程中锻炼了配置库的能力。

总体来说，我在本次作业中的收获是多方面的。相信我在本次实验中锻炼的能力会对我未来的学习和科研有所帮助。