数值分析

大作业二

$sin(x)$的求解

学　　号	2017011589
姓　　名	吾尔开西
班　　级	自 76

目录

一、需求分析

本次大作业需要求解 sin(x)，x 的取值范围是任意的，需要保证算法能精确到小数点后 4 位，需要对运算结果进行误差分析，包括方法误差和舍入误差。根据分析得到误差确定迭代次数等。

二、逼近法

1、算法原理

由于$\sin(x+2k\pi)=\sin(x)$, $k=\pm1,\pm2,\ldots$,首先将 x 归到$[0,2\pi]$区间内，$x_0\in[0,2\pi]$, $sin(x)=sin(x_0)$。之后使用泰勒展开来逼近$\sin(x_0)$，多项式项数足够高时即可满足精度:

$$\sin(x_0)=x_0-\frac{{x_0}^3}{3!}+\frac{{x_0}^5}{5!}+\cdots$$

比如用前 k 项来近似$f(x_0)=\sin(x_0)$

$$f_k=x_0-\frac{{x_0}^3}{3!}+\frac{{x_0}^5}{5!}+\cdots+(-1)^{k+1}\frac{{x_0}^{2k-1}}{(2k-1)!}$$

2、误差分析

（1）方法误差

假设用泰勒展开的前 k 项来近似$\sin(x)$

$$f(x_0)=x_0-\frac{{x_0}^3}{3!}+\frac{{x_0}^5}{5!}+\cdots+(-1)^{k+1}\frac{{x_0}^{2k-1}}{(2k-1)!}+\frac{\sin^{2k+1}(\xi)}{(2k+1)!}{x_0}^{2k+1}$$

$$f(x_0)-f_k=\frac{\sin^{2k+1}(\xi)}{(2k+1)!}{x_0}^{2k+1}$$

其中，$\xi\in[0,x]$，方法误差：

$$\Delta_{k方法}=\left|\frac{\sin^{2k+1}(\xi)}{(2k+1)!}{x_0}^{2k+1}\right|\le\left|\frac{{x_0}^{2k+1}}{(2k+1)!}\right|$$

（2）存储误差

假设所有数据都用小数点后 m 位的变量来存储（实际编程中用 double 型变量）。

首先，将 x 归一化过程中可能需要加减π，所以x_0可能会带来误差，$\Delta x_0=\frac{1}{2}\times10^{-m}$，误差：

$$\Delta_{k存储\,1}=\frac{\partial f_k}{\partial x_0}\Delta x_0=\left(\sum_{i=1}^{k}\frac{x_0^{2i-2}}{(2i-2)!}\right)\Delta x_0$$

for 循环运算过程中，f_k的每次计算结果的赋值也可能产生存储误差

$$\Delta_{k存储\,2}=(k-1)\times\frac{1}{2}\times10^{-m}$$

$$\Delta_{k存储}=\frac{k-1}{2}\times10^{-m}+\left(\sum_{i=1}^{k}\frac{x_0^{2i-2}}{(2i-2)!}\right)\times\frac{1}{2}\times10^{-6}$$

（3）总误差：

$$\Delta_k = \Delta_{k存储} + \Delta_{k方法} \le \left|\frac{{x_0}^{2k+1}}{(2k+1)!}\right| + \frac{k-1}{2} \times 10^{-m} + \left(\sum_{i=1}^{k} \frac{x_0^{2i-2}}{(2i-2)!}\right) \times \frac{1}{2} \times 10^{-m}$$

其中，m 为变量的存储位数，double 型变量的 m 一般为 15。

由于当 n 足够大时，$(2n+1)! \gg a^{2n+1}$，**所以Δ_k能达到任意精度**。

x_0越大，总误差越大，所以考虑$x_0 = 2\pi$的情况。

取 k=11，得$\Delta_k = 8.8235 \times 10^{-5}$，符合要求。

三、常微分方程

1、算法原理

（1）得出方程

由于$\sin(x+k\pi) = (-1)^k \sin(x), k = \pm1, \pm2, \ldots$,首先将 x 归到$[-\frac{\pi}{2}, \frac{\pi}{2}]$区间内，$x_1 \in [-\frac{\pi}{2}, \frac{\pi}{2}]$，$sin(x) = (-1)^k sin(x_1)$。再将$x_1$化为$x_0 = \frac{x_1}{2}$（为了使 f 的导数有范围），$x_0 \in [-\frac{\pi}{4}, \frac{\pi}{4}]$，求出$sin(x_0)$后用倍角公式求出$sin(x)$。因为

$$\sin^2(x_0) + \cos^2(x_0) = 1$$

且$sin'(x) = \cos(x)$，设$y(x) = \sin(x)$，所以在$x \in [-\frac{\pi}{4}, \frac{\pi}{4}]$区间内：

$$y'(x) = \sqrt{1 - y^2(x)}$$

（2）欧拉公式

设$y_0 = 0, x_0 = 0, f(x,y) = \sqrt{1-y^2(x)}$，则$y' = f(x,y)$。将 0 到$x_0$的区间划分成 t 份，$h = \frac{x_0}{t}$。使用公式：

$$y_{n+1} = y_n + hf(x_n, y_n)$$

迭代计算 y，直到 n=t，$\sin(x_0) = y_t$。再用倍角公式求出

$$sin(x) = 2(-1)^k sin(x_0)\sqrt{1 - \sin^2(x_0)}$$

（3）开根号

由于运算过程中用到开根号运算，所以需要用方程求根的方法计算。

为了计算$\sqrt{q}$，设$\varphi(x)=x-\frac{x^2}{4}+\frac{q}{4}$，通过$x_{k+1}=\varphi(\mathrm{x_k})$的方式迭代计算$\sqrt{q}$。

q=$1-\sin^2(x_0)\in[0,1]$，所以$\varphi(x)\in[0,1]$，$|\varphi'(\mathrm{x})|=1-\frac{\mathrm{x}}{2}<1$，因此该方法迭代足够多次后可以收敛到$\sqrt{q}$

2、误差分析

$\left|\frac{\partial f}{\partial y}(x,y)\right|=\left|\frac{y}{\sqrt{1-y^2}}\right|\le 1.2688=M;\ y^{(2)}(\xi_n)\le 1=L;\ f(x,y)=\sqrt{1-y^2(x)}\le 1$

（1）方法误差

局部截断误差：

$$y_n=y(x_n)=y(x_{n+1})-hy'(x_{n+1})+\frac{h^2}{2}y^{(2)}(x_{n+1})+\cdots$$

$$y(x_{n+1})-y_{n+1}=-\frac{h^2}{2}y^{(2)}(x_{n+1})$$

设Δ_n是第 n 步累积的方法误差，方法累积误差：

$$\Delta_{n+1}=\left[1+h\frac{\partial f}{\partial y}(x_n,\eta_n)\right]\Delta_n+\frac{h^2}{2}y^{(2)}(\xi_n)$$

因为$\left|\frac{\partial f}{\partial y}(x,y)\right|=\left|\frac{y}{\sqrt{1-y^2}}\right|\le 1.2688=M$，$y^{(2)}(\xi_n)\le 1=L$，所以：

$$\Delta_{n+1}\le[1+hM]\Delta_n+\frac{h^2}{2}L$$

构造等比数列：

$$\Delta_{n+1}+\frac{1}{hM}\frac{Lh^2}{2}\le(1+hM)[\Delta_n+\frac{1}{hM}\frac{Lh^2}{2}]$$

$$\Delta_{n+1}\le[(1+hM)^{n+1}-1]\frac{1}{hM}\frac{Lh^2}{2}$$

$$=\left[(1+hM)^{\frac{x_0}{h}+1}-1\right]\frac{1}{hM}\frac{Lh^2}{2}$$

（2）存储误差

假设所有数据都用小数点后 m 位的变量来存储（实际编程中用 double 型变量）。

x_0的存储误差会给$h=\frac{x_0}{t}$带来误差Δh，同时每一步计算结果的存储也会带来误差。设δ_n是第 n 步存储误差的积累，则：

$$\delta_{n+1} \le \left[1+h\left|\frac{\partial f}{\partial y}(x_n,\eta_n)\right|\right]\delta_n+\frac{1}{2}\times 10^{-m}+|f(x_n,y_n)|\Delta h$$

$$|f(x_n,y_n)| \le 1$$

$$\delta_{n+1} \le [1+hM]\delta_n+\frac{1}{2}\times 10^{-m}+\Delta h$$

构造等比数列：

$$\delta_{n+1} \le (1+hM)\left[\delta_n+\frac{\frac{1}{2}10^{-m}+\Delta h}{hM}\right]$$

$$\le (1+hM)^{n+1}\frac{\frac{1}{2}10^{-m}+\Delta h}{hM}$$

又因为$\Delta h=\frac{\Delta x_0}{t}=\frac{10^{-m}}{2t}$

$$\delta_{n+1} \le (1+hM)^{n+1}\frac{\frac{1}{2}10^{-m}+\frac{1}{2t}10^{-m}}{hM}$$

（3）$\sin(x_0)$总误差：

$$\Delta y_n \le \delta_n+\Delta_n=(1+hM)^n\frac{\frac{1}{2}10^{-m}+\frac{1}{2t}10^{-m}}{hM}+[(1+hM)^n-1]\frac{Lh}{2M}$$

代入$M=1.2688$; $\mathrm{L}=1$; $t=\frac{x_0}{h}\ge 1$; $n=\frac{x_0}{h}\le\frac{\pi}{4h}$; $m=15$

$$\Delta y_n \le (1+1.2688h)^{\frac{\pi}{4h}}\frac{10^{-m}}{1.2688h}+\left[(1+1.2688h)^{\frac{\pi}{4h}}-1\right]\frac{h}{2.5376}$$

当 h 足够小时，$(1+1.2688h)^{\frac{\pi}{4h}}$趋向常数，通过控制 m，**可以达到任意精度**。

设$q=\sin(x_0)$，$|\sin(x)|=2q\sqrt{1-q^2}$，$\sin(x)$总误差：

$$\Delta y_n' \le 2\left(\sqrt{1-q^2} - \frac{q^2}{\sqrt{1-q^2}}\right)\Delta y_n$$

$$= 2\left(2\sqrt{1-q^2} - \frac{1}{\sqrt{1-q^2}}\right)\Delta y_n$$

$$0 \le q^2 \le \frac{1}{2}$$

$$\Delta y_n' \le 2\Delta y_n = 2(1+1.2688h)^{\frac{\pi}{4h}}\frac{10^{-15}}{1.2688h} + \left[(1+1.2688h)^{\frac{\pi}{4h}} - 1\right]\frac{h}{1.2688}$$

当$\boldsymbol{h \le \frac{\pi}{4}\frac{1}{12000}}$，即$\boldsymbol{t \ge 12000}$，$\boldsymbol{\Delta y_n' \le 8.8 \times 10^{-5} < \frac{1}{2} \times 10^{-4}}$

四、cordic 算法

1、算法原理

cordic 算法使用角度逼近的方法迭代求得$\sin(x)$。首先将 x 的值归一化到$[0,\frac{\pi}{2}]$区间（使用公式$\cos\left(x-\frac{\pi}{2}\right) = \sin(x)$），以向量$v_0 = (1,0)$为初始向量，向靠近 x 的方向旋转$\arctan(\left(\frac{1}{2}\right)^0)$的角度大小得到向量$v_1$;向量$v_1$再向靠近 x 的方向旋转$\arctan\left(\left(\frac{1}{2}\right)^1\right)$的角度大小得到向量$v_2$，依次类推。

即在第 i 步迭代时，向量v_i向靠近 x 的方向旋转$\arctan\left(\left(\frac{1}{2}\right)^i\right)$的角度大小得到向量$v_{i+1}$。

问题是向量v_{i+1}的坐标如何从向量v_i的坐标计算得到。由坐标旋转公式：

$$v_{i+1} = \begin{bmatrix} cos\theta_i & -sin\theta_i \\ sin\theta_i & cos\theta_i \end{bmatrix} v_i$$

其中，$\theta_i = \pm\arctan\left(\left(\frac{1}{2}\right)^i\right)$，（正负由 x 的大小决定），又因为：

$$cos\theta_i = \frac{1}{\sqrt{1+\tan^2\theta_i}}$$

$$sin\theta_i = \frac{tan\theta_i}{\sqrt{1+\tan^2\theta_i}}$$

$$v_{i+1} = \frac{1}{\sqrt{1+\tan^2\theta_i}}\begin{bmatrix} 1 & -tan\theta_i \\ tan\theta_i & 1 \end{bmatrix} v_i$$

由于$\theta_i=\pm\arctan\left(\left(\frac{1}{2}\right)^i\right)$，所以$tan\theta_i=\pm\left(\frac{1}{2}\right)^i$。将$\left(\frac{1}{2}\right)^i$与$\arctan\left(\left(\frac{1}{2}\right)^i\right)$都提前算好，迭代足够多次后，向量$v_i$与 x 轴的夹角足够靠近角度 x，可达到任意精度。

2、误差分析

（1）方法误差

当算法迭代到第 n 步时（n 足够大，如大于 5），设向量v_n与 x 轴的夹角为α_n，所求的角度为 x，可认为$|\alpha_n-x|\leq\theta_{n-1}$，因此：

$$|\sin(\alpha_n)-\sin(x)|\leq\max|\sin'(\xi)|\,|\alpha_n-x|\leq|\alpha_n-x|\leq|\theta_{n-1}|$$

$$=\arctan\left(\left(\frac{1}{2}\right)^{n-1}\right)$$

因此方法误差$\Delta_n\leq\arctan\left(\left(\frac{1}{2}\right)^{n-1}\right)$

（2）存储误差

设向量$v_i=\begin{bmatrix}x_i\\y_i\end{bmatrix}$，则

$$\begin{cases}x_{i+1}=K_i[x_i-(tan\theta_i)y_i]\\y_{i+1}=K_i[(tan\theta_i)x_i+y_i]\end{cases}$$

其中，$K_i,tan\theta_i$都是提前算好的系数，舍入误差为$\frac{1}{2}\times10^{-m}$（m 为变量的存储位数）。设第 i 步 x 的舍入误差为δ_{xi}，y 的舍入误差为δ_{yi}。所以：

$$\begin{cases}\delta_{x(i+1)}=K_i\delta_{xi}+K_{\mathrm{i}}tan\theta_i\delta_{yi}+\dfrac{1}{2}\times10^{-m}(|x_i|\ \ +tan\theta_i|y_i|)+\dfrac{1}{2}\times10^{-m}\\\delta_{y(i+1)}=K_itan\theta_i\delta_{xi}+K_{\mathrm{i}}\delta_{yi}+\dfrac{1}{2}\times10^{-m}(tan\theta_i|x_i|\ \ +|y_i|)+\dfrac{1}{2}\times10^{-m}\end{cases}$$

其中，$K_i\leq1$，$K_{\mathrm{i}}tan\theta_i=sin\theta_i\leq1,|x_i|\leq1,tan\theta_i|y_i|\leq1$

因此：

$$\begin{cases}\delta_{x(i+1)}\leq\delta_{xi}+\delta_{yi}+\dfrac{3}{2}\times10^{-m}\\\delta_{y(i+1)}\leq\delta_{xi}+\delta_{yi}+\dfrac{3}{2}\times10^{-m}\end{cases}$$

其中$\delta_{x0}=\delta_{y0}=0$，因此$\delta_{yi}=\delta_{xi}$

$$\delta_{y(i+1)}\leq2\delta_{yi}+\frac{3}{2}\times10^{-m}$$

$$\delta_{y(i+1)}+\frac{3}{2}\times 10^{-m}\le 2(\delta_{yi}+\frac{3}{2}\times 10^{-m})$$

$$\delta_{yn}+\frac{3}{2}\times 10^{-m}\le 2^n\times\frac{3}{2}\times 10^{-m}$$

$$\delta_{yn}\le (2^n-1)\times\frac{3}{2}\times 10^{-m}$$

（3）总误差

总误差=$\delta_{yn}+\Delta_n\le (2^n-1)\times\frac{3}{2}\times 10^{-m}+\arctan\left(\left(\frac{1}{2}\right)^{n-1}\right)$

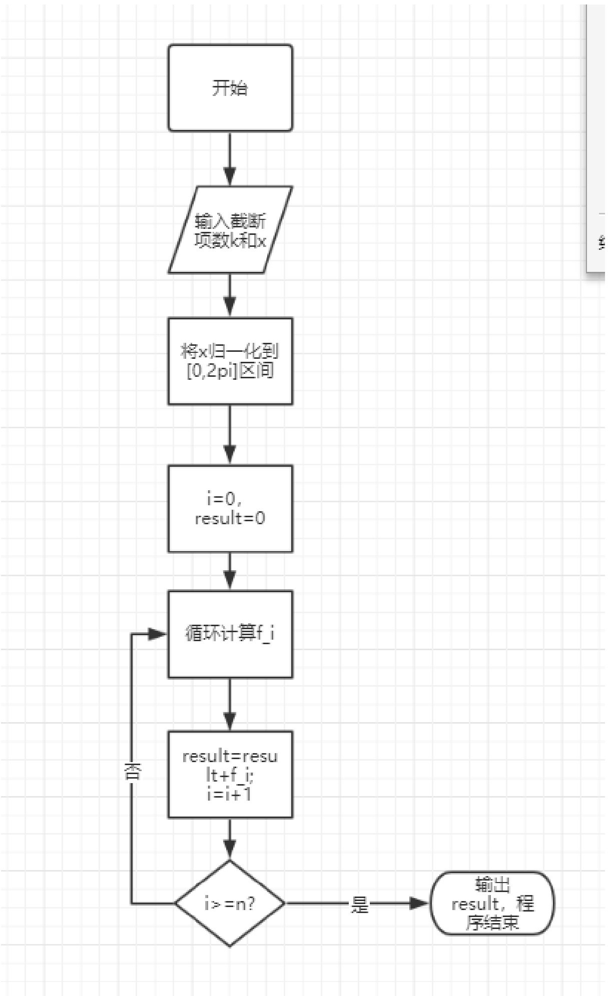
$\arctan\left(\left(\frac{1}{2}\right)^{n-1}\right)$随 n 的增大而减小，通过控制 m，**可以达到任意精度。**

当 n=15 时，m=15（double 型变量的小数点后存储位数）时，**总误差**$\le \mathbf{6.104\times 10^{-5}<\frac{1}{2}\times 10^{-4}}$

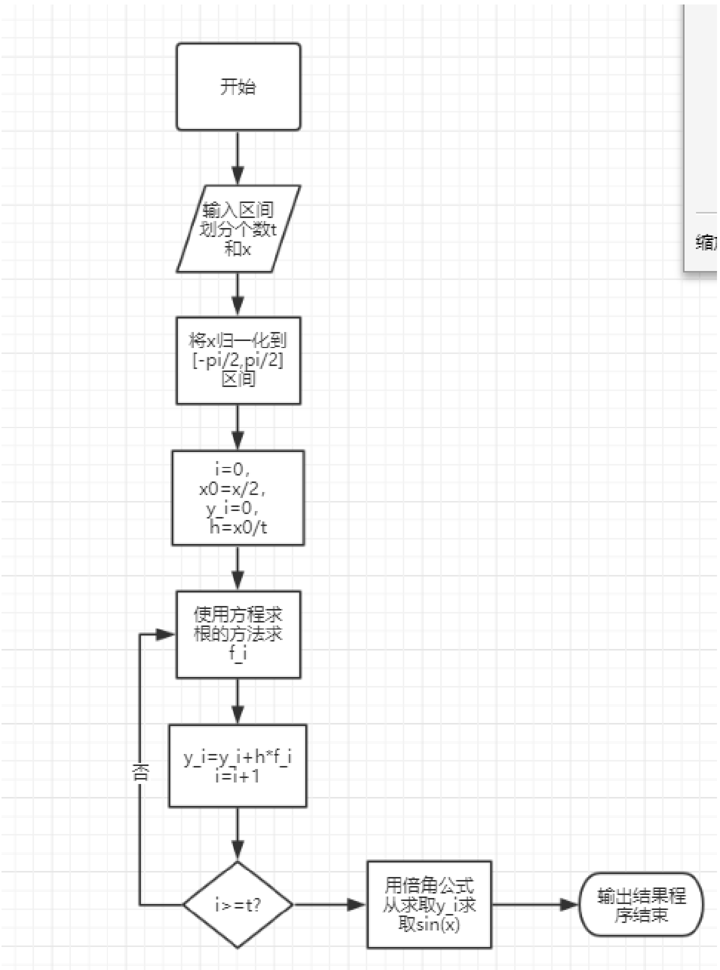
五、程序流程

各个算法的思路已在第四部分叙述，下面是算法的流程图：

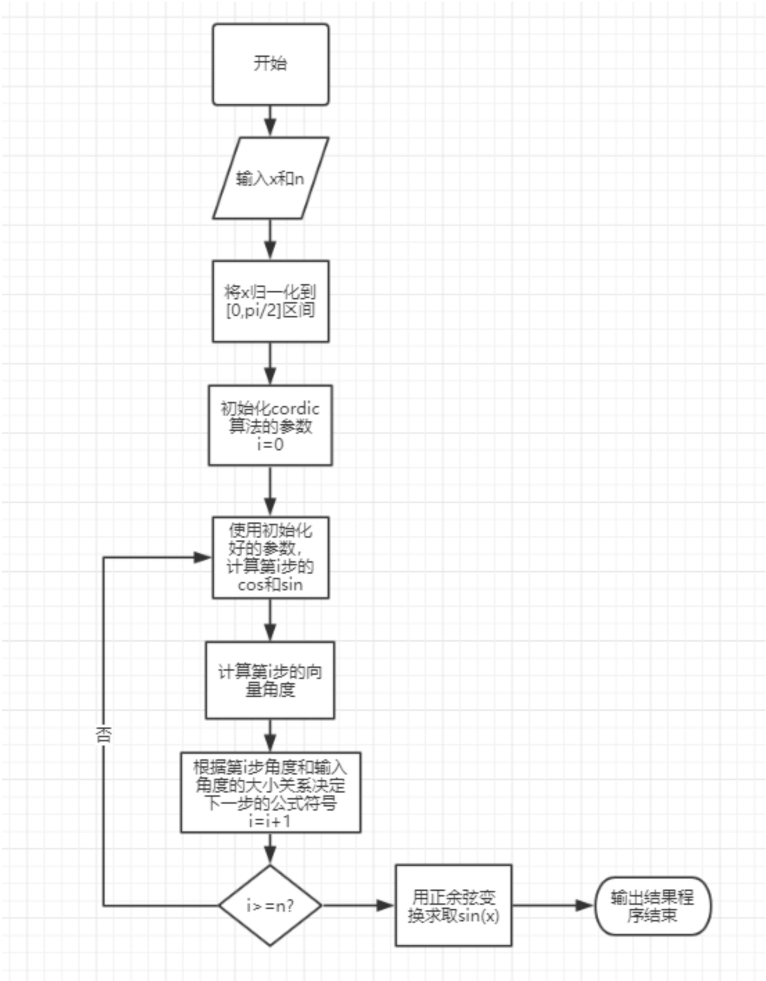
Taylor 展开逼近法的流程图：



常微分方程方法的流程图:



cordic 算法的流程图



六、计算代价与收敛速度

各个算法的方法误差、舍入误差以及总体误差已在第四部分充分分析，下面我们来分析一下各个算法的计算复杂度和收敛速度。

1、Taylor 展开

（1）收敛速度

Taylor 展开方法误差：

$$\Delta_{k方法}=\left|\frac{\sin^{2k+1}(\xi)}{(2k+1)!}{x_0}^{2k+1}\right| \leq \left|\frac{{x_0}^{2k+1}}{(2k+1)!}\right|$$

收敛速度：

$$\frac{\Delta_{n+1}}{\Delta_n} \approx \frac{x_0^2}{(2n+3)(2n+2)}$$

（2）计算代价

Taylor 展开的方法由于在计算第 i 项时需要计算 2i-1 次幂，所以总计算复杂度为$\sum_{i=1}^{n} 2i-1 = O(n^2)$，由第四部分的分析知当$n \geq 11$时该方法误差满足要求,运算次数约为 121。

2、常微分方程

（1）收敛速度

常微分方程方法误差:

$$\Delta_{n+1} \leq \left[(1+hM)^{\frac{x_0}{h}+1} - 1\right]\frac{1}{M}\frac{Lh}{2}$$

其中，$(1+hM)^{\frac{x_0}{h}+1}$趋近于常数 A，则Δ_n收敛速度：

$$\frac{\Delta_{n+1}}{\Delta_n} = \frac{\Delta_{x_0/(n+1)}}{\Delta_{x_0/n}}$$

$$\approx \frac{\frac{x_0}{n+1}}{\frac{x_0}{n}} = \frac{n}{n+1}$$

（2）计算代价

常微分方程的算法由于需要用方程求根的方法来进行开根号操作，所以运算复杂度为$O(nk)$，其中 k 为方程求根算法的迭代次数，在程序中设为 200。前面已经指出当$t \geq 12000$时可以满足误差要求，运算次数>2400000。

3、cordic 算法

（1）收敛速度

cordic 算法方法误差：

$$\Delta_n \leq \arctan\left(\left(\frac{1}{2}\right)^{n-1}\right)$$

收敛速度：

$$\frac{\Delta_{n+1}}{\Delta_n} = \frac{\arctan\left(\left(\frac{1}{2}\right)^{n}\right)}{\arctan\left(\left(\frac{1}{2}\right)^{n-1}\right)}$$

由于 x=0 时，$(\arctan x)' = \frac{1}{1+x^2} = 1$，所以当 n 足够大时，$\frac{\Delta_{n+1}}{\Delta_n} \approx \frac{\left(\frac{1}{2}\right)^n}{\left(\frac{1}{2}\right)^{n-1}} = 0.5$

（2）计算代价

cordic 算法一次只需要循环 n 次，所以运算复杂度为$O(n)$，第四部分中已经说明当$n \geq 15$时满足误差要求，运算次数约为 15。

4、比较

三种算法比较来看，计算代价：常微分方程算法>Taylor 展开逼近>cordic 算法，cordic 算法的运算次数远少于前两种算法。

收敛速度，$\frac{x_0^2}{(2n+3)(2n+2)} < 0.5 < \frac{n}{n+1}$，所以常微分方程收敛速度最快，Taylor 展开最慢，cordic 算法收敛速度居中。

综上所述，cordic 算法计算代价小，收敛速度居中，其他两个算法收敛速度和计算代价互补。

七、结果

我用 c#编写了用户界面，用户可以选择运算方法和迭代次数，下面是用三种方法进行计算的结果。

八、总结

通过完成这次大作业，我提高了运用理论知识解决实际问题的能力，比如改进欧拉法、方程根的数值解等；同时也学到了很多新知识，比如 cordic 算法。

这次大作业的难点在与误差分析，在对实际问题的结果进行误差分析的过程中，我加深了对方法误差和舍入误差的理解，认识到了实际问题中误差分析的必要性。

将一个数值算法投入应用中，除了要考虑它的误差，另一个重要因素是算法的复杂度，cordic 算法的实现虽然需要提前准备一些数据，但其运算速度很快，这也是它被广泛应用的原因。

此外，我为我的程序设计了简单的用户界面，感觉程序的实用性更强了。

总之，通过这次大作业我增长了很多新知识，加深了对理论知识的理解，也加强了我动手解决实际问题的能力。