

使用多种数值方法求 e

数值分析与算法 课程项目

自54班 叶沁媛 2015011469

2017 年 12 月 27 日

摘要 分别使用Taylor展开、常微分方程求解、方程 $\int_1^x \frac{1}{t} dt - 1 = 0$ 求解这三种不同方法来求取自然对数 e 。其中常微分方程求解法又分别使用了梯形公式与改进欧拉法两种不同数值方法。对每种方法进行了方法误差和舍入误差的估计与分析，同时对计算代价、收敛速度进行了分析。对每种数值方法编写程序进行验证，程序运行结果见附录部分。

1 对 e^x 进行Taylor展开

1.1 算法描述

$f(x) = e^x$ 在 $x = 0$ 处Taylor展开，得到：

$$\begin{aligned} e^x &= 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots + \frac{x^n}{n!} + \dots \\ &= 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots + \frac{x^n}{n!} + e^\xi \frac{x^{n+1}}{(n+1)!}, \quad \xi \in (0, x) \end{aligned} \quad (1)$$

本题中，求取 $f(1) = e$ ，即 $x = 1$ ，

$$e = 1 + 1 + \frac{1}{2!} + \frac{1}{3!} + \frac{1}{4!} + \dots + \frac{1}{n!} + \dots \quad (2)$$

令 $y_0 = 1$ ，使用如下的递推公式进行计算：

$$y_{n+1} = y_n + \frac{1}{(n+1)!} \quad (3)$$

1.2 误差分析

方法误差 方法误差为Taylor展开的余项。由于求解时 e 的具体值未知，根据 $e < 3$ 进行放大。

$$\begin{aligned} \Delta_n &= \frac{e^\xi}{(n+1)!}, \quad \xi \in (0, 1) \\ &< \frac{e}{(n+1)!} \\ &< \frac{3}{(n+1)!} \end{aligned} \quad (4)$$

舍入误差 回顾式(3)中的迭代过程。在编写程序时， $n!$ 使用long long类型保存，无误差，同时假设double类型之间的加法不会产生误差，因此只用考虑 n 次除法所产生的舍入误差。故

$$\delta_n \leq \frac{n}{2} \times 10^{-m} \quad (5)$$

总误差

$$r_n \leq \Delta_n + \delta_n \leq \frac{3}{(n+1)!} + \frac{n}{2} \times 10^{-m} \quad (6)$$

题目要求精度达到至少小数点后6位，即 $r_n \leq \frac{1}{2} \times 10^{-6}$ 。使用C++中的double类型，可以近似认为 $m = 14$ ，此时取 $n = 10$ ， $r_n \leq 5 \times 10^{-14} + \frac{3}{11!} \approx \frac{3}{11!} < \frac{1}{2} \times 10^{-6}$ ，符合要求。

1.3 计算代价

Code Snippet 1: Taylor展开求e，核心代码

```
1  for (int i=1; i<=n; i++){
2      sum+=xpow/fac;
3      xpow*=x;
4      fac*=i+1;
5  }
```

其中xpow表示x的i次方，fac表示i的阶乘，在计算过程中使用了上一次循环的结果。每次循环中，加法1次，乘法2次，除法1次，总体运算量在 $O(n)$ 量级。本题中，取 $n = 10$ 。

2 求解常微分方程 $y' = y$

2.1 常微分方程的设计

构造常微分方程

$$\begin{cases} y' = y \\ y(0) = 1 \end{cases} \quad (7)$$

该常微分方程的解为

$$y = e^x \quad (8)$$

$x = 1$ 时， $y(x) = e$ ，使用常微分方程数值解法求解 $y(1)$ 即可得到 e 。

2.2 梯形公式

梯形公式是数值积分中常用的公式，一般写成如下的隐式形式：

$$y_{n+1} = y_n + \frac{h}{2}[f(x_n, y_n) + f(x_{n+1}, y_{n+1})] \quad (9)$$

本题中，由于 $f(x_{n+1}, y_{n+1}) = y_{n+1}$ ，可以将梯形公式整理为显式形式：

$$y_{n+1} = \frac{2+h}{2-h} y_n \quad (10)$$

可以利用此式进行递推求解。

2.2.1 误差分析

方法累计误差

$$y(x_{n+1}) = y_n + hy'(x_n) + \frac{h^2}{2!} y^{(2)}(x_n) + \frac{h^3}{3!} y^{(3)}(x_n) + \dots \quad (11)$$

$$\begin{aligned} y_{n+1} &= y_n + \frac{h}{2} [y'(x_n) + y'(x_{n+1})] \\ &= y_n + \frac{h}{2} [y'(x_n) + y'(x_n) + hy^{(2)}(x_n) + \frac{h^2}{2} y^{(3)}(x_n) + \dots] \end{aligned} \quad (12)$$

$$\therefore y(x_{n+1}) - y_{n+1} = \frac{h^3}{12} y^{(3)}(x_n) = o(h^3) \quad (13)$$

因此，

$$\Delta_n \leq \frac{2+h}{2-h} \Delta_{n-1} + \frac{Th^3}{12} \quad (14)$$

其中， $y^{(3)}(x) \leq T$ 。令 $a = \frac{2+h}{2-h}$ 、 $b = \frac{Th^3}{12}$ ，从而

$$\Delta_{n+1} + \frac{b}{a-1} \leq a(\Delta_n + \frac{b}{a-1}) \leq \dots \leq a^{n+1}(\Delta_0 + \frac{b}{a-1}) \quad (15)$$

$$\Delta_n \leq \frac{a^n - 1}{a - 1} b = \frac{(\frac{2+h}{2-h})^n - 1}{\frac{2+h}{2-h} - 1} \times \frac{Th^3}{12} \quad (16)$$

其中， $\lim_{h \rightarrow 0} (\frac{2+h}{2-h})^n = e$ 。由于 $y^{(3)}(x) = e^x$ ，取 $T = y^{(3)}(1) = e$ 。整理得到：

$$\Delta_n \leq \frac{(e-1)eh^2}{12} \quad (17)$$

由于求解时 e 的具体值未知，根据 $e \leq 3$ 再度进行放大，得到

$$\Delta_n \leq \frac{h^2}{2} \quad (18)$$

舍入累计误差

$$\delta_{n+1} \leq \frac{2+h}{2-h} \delta_n + \frac{1}{2} \times 10^{-m} \quad (19)$$

与方法累计误差的分析方法相类似，将不等式分解为等比数列，再根据 $e \leq 3$ 进行放大，得到

$$\begin{aligned} \delta_n &\leq \frac{(\frac{2+h}{2-h})^n - 1}{\frac{2+h}{2-h} - 1} \times \frac{1}{2} \times 10^{-m} \\ &\leq \frac{10^{-m}}{h} \end{aligned} \quad (20)$$

总误差

$$r_n \leq \Delta_n + \delta_n \leq \frac{h^2}{2} + \frac{10^{-m}}{h} \quad (21)$$

题目要求精度达到至少小数点后6位。使用C++中的double类型，可以近似认为 $m = 14$ ，此时取 $h = 10^{-4}$ ， $r_n \leq \frac{1}{2} \times 10^{-8} + 10^{-10} \leq \frac{1}{2} \times 10^{-6}$ ，符合要求。

2.2.2 计算代价

Code Snippet 2: 梯形公式，核心代码

```
1 double co = (2+h)/(2-h);
2 for (int i=0; i<n; i++){
3     y=co*y;
4 }
```

每次循环中使用一次乘法，计算代价为 $O(n)$ 量级。本题中，由于取 $h = 10^{-4}$ ，故 $n = 10000$ 。

2.3 改进欧拉法

采用改进的欧拉法求解常微分方程，方法如下：

预测 $\bar{y}_{n+1} = y_n + hf(x_n, y_n)$

校正 $y_{n+1} = y_n + \frac{h}{2}[f(x_n, y_n) + f(x_{n+1}, \bar{y}_{n+1})]$

在本题中， $f(x_n, y_n) = y_n$ ，上式可以简化为

预测 $\bar{y}_{n+1} = y_n + hy_n$

校正 $y_{n+1} = y_n + \frac{h}{2}[y_n + \bar{y}_{n+1}]$

2.3.1 误差分析

预测部分采用普通欧拉公式：

$$y(x_{n+1}) = y(x_n) + hy'(x_n) + \frac{h^2}{2!}y^{(2)}(\xi_n), \xi \in (x_n, x_{n+1}) \quad (22)$$

$$y_{n+1} = y_{x_n} + hy'(x_n) \quad (23)$$

$$\therefore y(x_{n+1}) - y_{n+1} = \frac{h^2}{2!}y^{(2)}(\xi_n) = o(h^2) \quad (24)$$

校正部分采用梯形公式，推导方法与式(11)~(13)相同。

方法累计误差

$$\begin{cases} \bar{\Delta}_{n+1} \leq (1+h)\Delta_n + \frac{L}{2}h^2 \\ \Delta_{n+1} \leq \Delta_n + \frac{h}{2}(\Delta_n + \bar{\Delta}_{n+1}) + \frac{Th^3}{12} \end{cases} \quad (25)$$

整理得到：

$$\Delta_{n+1} \leq (\frac{h^2}{2} + h + 1)\Delta_n + (\frac{L}{4} + \frac{T}{12})h^3 \quad (26)$$

其中 $y^{(2)}(x) \leq M$ 、 $y^{(3)}(x) \leq T$ 。令 $a = \frac{h^2}{2} + h + 1$ 、 $b = (\frac{L}{4} + \frac{T}{12})h^3$ ，则式(26)可改写为：

$$\Delta_{n+1} + \frac{b}{a-1} \leq a(\Delta_n + \frac{b}{a-1}) \leq \dots \leq a^{n+1}(\Delta_0 + \frac{b}{a-1}) \quad (27)$$

$$\Delta_n \leq \frac{a^n - 1}{a - 1} b = \frac{(1 + h + \frac{h^2}{2})^n - 1}{h + \frac{h^2}{2}} (\frac{L}{4} + \frac{T}{12}) h^3 \quad (28)$$

其中， $\lim_{h \rightarrow 0} (1 + h + \frac{h^2}{2})^n = e$ 。由于 $y^{(2)}(x) = e^x$ ，取 $L = y^{(2)}(1) = e$ ，同理取 $T = y^{(3)}(1) = e$ 。最终整理得到：

$$\Delta_n \leq \frac{1}{3} e(e-1) h^2 \quad (29)$$

由于求解时 e 的具体值未知，根据 $e \leq 3$ 再度进行放大，得到

$$\Delta_n \leq 2h^2 \quad (30)$$

舍入累计误差

$$\begin{cases} \bar{\delta}_{n+1} \leq (1+h)\delta_n + \frac{1}{2} \times 10^{-m} \\ \delta_{n+1} \leq \delta_n + \frac{h}{2}\delta_n + \frac{h}{2}\bar{\delta}_{n+1} + \frac{1}{2} \times 10^{-m} \end{cases} \quad (31)$$

整理得到：

$$\delta_{n+1} \leq (1 + h + \frac{h^2}{2})\delta_n + (\frac{h}{2} + 1) \times \frac{1}{2} \times 10^{-m} \quad (32)$$

与方法累计误差的分析方法相类似，将不等式分解为等比数列，再根据 $e \leq 3$ 进行放大，得到：

$$\begin{aligned} \delta_{n+1} &\leq \frac{(1 + h + \frac{h^2}{2})^n - 1}{h + \frac{h^2}{2}} (\frac{h}{2} + 1) \times \frac{1}{2} \times 10^{-m} \\ &\leq \frac{10^{-m}}{h} \end{aligned} \quad (33)$$

总误差

$$r_n \leq \Delta_n + \delta_n \leq 2h^2 + \frac{10^{-m}}{h} \quad (34)$$

题目要求精度达到至少小数点后6位。取 $m = 14$ 、 $h = 10^{-4}$ ， $r_n \leq 2 \times 10^{-8} + 10^{-10} \leq \frac{1}{2} \times 10^{-6}$ ，符合要求。

2.3.2 计算代价

Code Snippet 3: 改进欧拉法，核心代码

```
1  for (int i=0; i<n; i++){
2      y_est=y+h*f(x,y);
3      y=y+(h/2)*(f(x,y)+f(x+h,y_est));
4      x=x+h;
5  }
```

所设计的常微分方程为 $y' = y$ ，故本题中的 $f(x, y)$ 函数无需额外的运算，直接返回 y 。在每次循环的过程中，共使用4次加法，2次乘法，1次除法。在循环 n 次的情况下，总体的运算量在 $O(n)$ 量级。本题中，由于取 $h = 10^{-4}$ ，故 $n = 10000$ 。

3 求解方程 $\int_1^x \frac{1}{t} dt - 1 = 0$

3.1 算法描述

复化梯形公式求积分 $f(x) = \frac{1}{x}$ 。将区间 $[1, x]$ 分为 N 个小区间, $h_N = \frac{x-1}{N}$, 在这些小区间上做梯形公式积分。

$$\begin{aligned} I &= \sum_{k=0}^{N-1} \frac{h_N}{2} [f(x_k) + f(x_{k+1})] \\ &= \sum_{k=0}^{N-1} \frac{x-1}{2N} \left[\frac{1}{1+kh_N} + \frac{1}{1+(k+1)h_N} \right] \end{aligned} \quad (35)$$

牛顿迭代法求方程 $f(x) = \int_1^x \frac{1}{t} dt - 1$ 、 $f'(x) = \frac{1}{x}$, 使用下式进行迭代:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, \quad x_{n+1} = x_n \left(2 - \int_1^x \frac{1}{t} dt \right) \quad (36)$$

初值选择 选取初值区间 $[2, 3]$ 。在此区间上, 满足:

- $f(2)f(3) < 0$
- $f^{(2)}(x) = -\frac{1}{x^2}$ 在 $[2, 3]$ 不变号
- $\forall x \in [2, 3] \quad f'(x) = \frac{1}{x} \neq 0$
- $\frac{|f(2)|}{3-2} \leq f'(2)$

因此, $\forall x \in [2, 3]$, 牛顿迭代法收敛。取 $x_0 = 2$ 。

3.2 误差分析

3.2.1 复化梯形公式

假设将区间 $[1, x]$ 分为 N 个小区间, 则 $h_N = \frac{x-1}{N}$ 。在第 k 个小区间上, 使用梯形公式积分。

方法误差

$$R_k[f] = -\frac{h^3}{12} f^{(2)}(\eta_k), \quad \eta_k \in [x_k, x_{k+1}] \quad (37)$$

$$R[f] = \sum_{k=0}^{n-1} R_k(f) \in [\min f^{(2)}(\eta), \max f^{(2)}(\eta)] \quad (38)$$

$$\because f^{(2)}(x) = -\frac{2}{x^3} \in C[2, 3] \quad (39)$$

$$\therefore \exists \eta \in [2, 3] \quad s.t. \sum_{k=0}^{N-1} f^{(2)}(\eta_k) = n f^{(2)}(\eta) \quad (40)$$

$$R[f] = -\frac{(b-a)h_N^2}{12} f^{(2)}(\eta) \quad (41)$$

$$\because |f^{(2)}(\eta)| \leq |f^{(2)}(2)| = \frac{1}{4} \quad \therefore |R[f]| \leq \frac{h_N^2}{48} \quad (42)$$

舍入误差 在 N 个小区间上分别做梯形公式积分，整体可以看做 $2N$ 项相加而成。可以粗略地认为这 $2N$ 项在计算时各产生了 $\frac{1}{2} \times 10^{-m}$ 的舍入误差，故舍入误差 $\delta \leq N \times 10^{-m}$ 。

3.2.2 牛顿迭代法

分为方法误差和舍入误差两部分进行分析，

累计方法误差 回顾式(36)，假设 x_n 无误差，复化梯形积分产生了误差：

$$\Delta_{n+1}^{(1)} \leq \frac{h_N^2}{48} \times x_n < \frac{h_N^2}{16} \quad (43)$$

假设复化梯形积分无误差， x_n 有误差 Δ_n ：

$$\Delta_{n+1}^{(2)} = |\phi'(\xi)| \Delta_n = |1 - \ln \xi| \Delta_n < (1 - \ln 2) \Delta_n < 0.4 \Delta_n \quad (44)$$

综合以上二式，总体方法误差

$$\Delta_{n+1} \leq \Delta_n^{(1)} + \Delta_n^{(2)} \leq \frac{h_N^2}{16} + 0.4 \Delta_n \quad \therefore \Delta_n \leq \frac{1 - 0.4^n}{1 - 0.4} \times \frac{h_N^2}{16} \quad (45)$$

累计舍入误差 假设 x_n 无误差，复化梯形积分产生了舍入误差 $N \times 10^{-m}$ ：

$$\delta_{n+1}^{(1)} \leq x_n \times N \times 10^{-m} \leq 3N \times 10^{-m} \quad (46)$$

假设复化梯形积分无误差， x_n 有舍入误差 δ_n ：

$$\delta_{n+1}^{(2)} = |\phi'(\xi)| \delta_n = |1 - \ln \xi| \delta_n < (1 - \ln 2) \delta_n < 0.4 \delta_n \quad (47)$$

综合以上二式，总体舍入误差

$$\begin{aligned} \delta_{n+1} &\leq \delta_{n+1}^{(1)} + \delta_{n+1}^{(2)} + \frac{1}{2} \times 10^{-m} \\ &\leq 0.4 \delta_n + (3N + \frac{1}{2}) \times 10^{-m} \end{aligned} \quad \therefore \delta_n \leq \frac{1 - 0.4^n}{1 - 0.4} \times (3N + \frac{1}{2}) \times 10^{-m} \quad (48)$$

总误差

$$r_n \leq \Delta_n + \delta_n \leq \frac{1 - 0.4^n}{1 - 0.4} \left[\frac{h_N^2}{16} + (3N + \frac{1}{2}) \times 10^{-m} \right] \quad (49)$$

根据下节（3.3节）的分析，如果不存在误差，则牛顿迭代法迭代4次就可以到符合精度要求的结果。取 $n = 4, h_N = 10^{-4} (N = 10^4)$ 。于是

$$r_4 \leq \frac{1 - 0.4^4}{1 - 0.4} \left[\frac{10^{-8}}{16} + (3 \times 10^4 + \frac{1}{2}) \times 10^{-14} \right] < 1.51 \times 10^{-9} \quad (50)$$

配合 $n = 4$ 时得到的

$$e_4 \leq 4 \times (0.2)^{2^4} \leq 2.63 \times 10^{-11} \quad (51)$$

e_4 与 r_4 之和小于 $\frac{1}{2} \times 10^{-6}$ ，符合精确到小数点后6位的条件。

3.3 收敛速度

对于牛顿迭代法

$$e_{n+1} = \frac{1}{2} \phi^{(2)}(\xi_n) e_n^2 \leq \frac{M}{2} e_n^2 \quad (52)$$

$$e_n \leq \frac{2}{M} \left[\frac{M}{2} e_0 \right]^{2^n}, \quad M = \max |\phi^{(2)}(x)| \quad (53)$$

本题中, 取初始值 $x_0 = 2$, 故 $e_0 = e - 2 < 0.8$ 。同时, $\phi^{(2)}(x) = \frac{1}{x}$, 故取 $M = \frac{1}{2}$ 。为使 $e_n < \frac{1}{2} \times 10^{-6}$, 求得 $n \geq 4$ 。

在编写程序时, 使用误差分析的事后估计, 即当 $\frac{1}{1-L} |x_{k+1} - x_k| < \frac{1}{2} \times 10^{-6}$ 时, 认为 x_k 为满足精度要求的解。取 $L = 0.4 > \phi'(2)$ 。

3.4 计算代价

分析复化梯形积分的计算代价, 将区间 $[a, b]$ 分割成为 N 个小区间。

Code Snippet 4: 复化梯形积分, 核心代码

```
1 for (int i=0; i<n; i++){
2     double x=a+i*h;
3     sum+=(h/2)*(f(x)+f(x+h));
4 }
```

每次调用 $f(x) = \frac{1}{x}$ 时, 使用了一次除法。每次循环中, 计算 x 使用了 1 次加法, 1 次乘法; 每次循环中, 计算 sum 的增量使用了 2 次加法, 1 次乘法, 3 次除法。故总计使用 $3N$ 次加法, $2N$ 次乘法, $3N$ 次除法。总计算量在 $O(N)$ 量级。根据误差分析的估计, 本题中取 $N = 10000$ 。

Code Snippet 5: 牛顿迭代法, 核心代码

```
1 while (true){
2     x_before = x;
3     x = x - f(x) / f_derivative(x);
4     if (5.0/3.0*abs(x-x_before)<eps) break;
5 }
```

运行程序时, 发现牛顿迭代法经过 4 次迭代后即满足精度要求, 迭代至第 5 次是为了进行误差的事后估计。每次调用 $f(x)$ (梯形积分) 需要 $O(N)$ 量级的运算, 与此相比, 调用 $f_{\text{derivative}}(x)$ (即 $\frac{1}{x}$) 的相应运算次数可忽略不计。一般地, 若迭代次数为 k , 则需要 $O(kN)$ 的运算量。

附录：程序运行结果

运行所提交的na_project_2.exe文件，按程序提示输入数字1/2/3/4并回车后即可运行相应的数值方法。按其他任意键退出程序。

```
C:\WINDOWS\system32\cmd.exe
Please select:
1. Taylor series
2. Integral using trapezoid method
3. Integral using improved Euler method
4. Solving equation using Newton's method
Input other keys to quit.
1
Using Taylor Series Method ...
iter #0: 1.000000
iter #1: 2.00000000
iter #2: 2.50000000
iter #3: 2.66666667
iter #4: 2.70833333
iter #5: 2.71666667
iter #6: 2.71805556
iter #7: 2.71825397
iter #8: 2.71827877
iter #9: 2.71828153
iter #10: 2.71828180
Answer: 2.71828180

Please select:
1. Taylor series
2. Integral using trapezoid method
3. Integral using improved Euler method
```

图 1: Taylor展开

```
C:\WINDOWS\system32\cmd.exe
iter #9985: 2.7142074645
iter #9986: 2.7144788988
iter #9987: 2.7147503603
iter #9988: 2.7150218489
iter #9989: 2.7152933647
iter #9990: 2.7155649076
iter #9991: 2.7158364777
iter #9992: 2.7161080749
iter #9993: 2.7163796993
iter #9994: 2.7166513508
iter #9995: 2.7169230295
iter #9996: 2.7171947354
iter #9997: 2.7174664685
iter #9998: 2.7177382287
iter #9999: 2.7180100161
iter #10000: 2.7182818307
Answer: 2.7182818307

Please select:
1. Taylor series
2. Integral using trapezoid method
3. Integral using improved Euler method
4. Solving equation using Newton's method
Input other keys to quit.
```

图 2: 梯形公式求解常微分方程

```
C:\WINDOWS\system32\cmd.exe
iter #9985: 2.7142074577
iter #9986: 2.7144788921
iter #9987: 2.7147503535
iter #9988: 2.7150218421
iter #9989: 2.7152933579
iter #9990: 2.7155649008
iter #9991: 2.7158364789
iter #9992: 2.7161080681
iter #9993: 2.7163796925
iter #9994: 2.7166513440
iter #9995: 2.7169230227
iter #9996: 2.7171947286
iter #9997: 2.7174664617
iter #9998: 2.7177382219
iter #9999: 2.7180100093
iter #10000: 2.7182818239
Answer: 2.7182818239

Please select:
1. Taylor series
2. Integral using trapezoid method
3. Integral using improved Euler method
4. Solving equation using Newton's method
Input other keys to quit.
```

图 3: 改进欧拉法求解常微分方程

```
C:\WINDOWS\system32\cmd.exe
Please select:
1. Taylor series
2. Integral using trapezoid method
3. Integral using improved Euler method
4. Solving equation using Newton's method
Input other keys to quit.
4
Using Newton Method ...
iter #0: 2.0000000000
iter #1: 2.61370564
iter #2: 2.71624392
iter #3: 2.71828106
iter #4: 2.71828182
iter #5: 2.71828182
Answer: 2.71828182

Please select:
1. Taylor series
2. Integral using trapezoid method
3. Integral using improved Euler method
4. Solving equation using Newton's method
Input other keys to quit.
```

图 4: 求解方程 $\int_1^x \frac{1}{t} dt - 1 = 0$