数值积分算法实验

姓名：王旋宇
学号：2012049010022
班级：数理基科班
老师：赖生建

2015-10-27

1 实验原理

1.1 组合梯形公式和辛普森公式

定理1.（组合梯形公式）设等距节点$[x_k, x_{k+1}], k = 0, 1, \cdots, M$将区间划分为宽度为$h = (b-a)/M$的$M$个子区间$x_k = a + kh$。$M$个子区间的组合梯形公式有两种等价表示方式

$$T(f,h) = \frac{h}{2}\sum_{k=1}^{M}(f(x_{k-1}) + f(x_k)) \tag{1}$$

$$T(f,h) = \frac{h}{2}(f(a) + f(b)) + h\sum_{k-1}^{M-1} f(x_k) \tag{2}$$

它们是区间$[a,b]$上$f(x)$的积分的逼近，记为

$$\int_a^b f(x)dx \approx T(f,h) \tag{3}$$

定理2.（组合辛普森公式）设$x_k = a + kh, k = 0, 1, \cdots, 2M$将区间$[a,b]$划分为$2M$个宽度为$h = (b-a)/(2M)$的等距子区间$[x_k, x_{k+1}]$。它们的组合辛普森公式有两种等价表示方法

$$S(f,h) = \frac{h}{3}\sum_{k=1}^{M}(f(x_{2k-1}) + 4f(x_{2k-1}) + f(x_{2k})) \tag{4}$$

$$S(f,h) = \frac{h}{3}(f(a) + f(b)) + \frac{2h}{3}\sum_{k=1}^{M-1} f(x_{2k}) + \frac{4h}{3}\sum_{k=1}^{M} f(x_{2k-1}) \tag{5}$$

它们是区间$[a,b]$上$f(x)$的积分的逼近，记为

$$\int_a^b f(x)dx \approx S(f,h) \tag{6}$$

推论3．（梯形公式的误差分析）设区间$[a,b]$划分为宽度为$h=(b-a)/M$的M个子区间$[x_k,x_{k+1}$，组合梯形公式

$$T(f,h)=\frac{h}{2}(f(a)+f(b))+h\sum_{k-1}^{M-1}f(x_k) \tag{7}$$

是对积分

$$\int_a^b f(x)dx = T(f,h)+E_T(f,h) \tag{8}$$

的逼近。如果$f\in C^2[a,b]$，则存在值$c, a<c<b$，使得误差项$E_T(f,h)$具有形式

$$E_T(f,h)=\frac{-(b-a)f^{(2)}(c)h^2}{12}=\boldsymbol{O}(h^2) \tag{9}$$

推论4．（辛普森公式的误差分析）设区间$[a,b]$划分为宽度为$h=(b-a)/2M$的$2M$个子区间$[x_k,x_{k+1}$，组合梯形公式

$$S(f,h)=\frac{h}{3}(f(a)+f(b))+\frac{2h}{3}\sum_{k=1}^{M-1}f(x_{2k})+\frac{4h}{3}\sum_{k=1}^{M}f(x_{2k-1}) \tag{10}$$

是对积分

$$\int_a^b f(x)dx = S(f,h)+E_S(f,h) \tag{11}$$

的逼近。如果$f\in C^4[a,b]$，则存在值$c, a<c<b$，使得误差项$E_S(f,h)$具有形式

$$E_T(f,h)=\frac{-(b-a)f^{(4)}(c)h^4}{180}=\boldsymbol{O}(h^4) \tag{12}$$

1.2 龙贝格积分

定义1．定义$[a,b]$上$f(x)$的面积公式序列$\{R(J,K):J\geq K\}_{j=0}^{\infty}$如下：

$$R(J,0) = T(J), J\geq K, \text{为连续梯形公式} \tag{13}$$

$$R(J,1) = S(J), J\geq K, \text{为连续辛普森公式} \tag{14}$$

$$R(J,2) = B(J), J\geq K, \text{为连续布尔公式} \tag{15}$$

定理5 . （龙贝格积分的理查森改进）改进的一般形式为

$$R(J,K)=\frac{4^K R(J,K-1)-R(J-1,K-1)}{4^K-1},\quad J\geq K \tag{16}$$

1 .3 自适应积分

定义2 . 自适应积分：组合积分公式要求使用等距节点。这并没有考虑到曲线的某些部分变化剧烈，需要比其他部分多加考虑的情况，因此引入一种能够在函数值变化大的区间采用较小步长的方法是很有用的。该技术成为自适应积分，它的基础是辛普森公式。

原理及误差分析 区间$[a_k,b_k]$先划分为两个相等的子区间$[a_{k1},b_{k1}],[a_{k2},b_{k2}]$并在两段上递归的应用辛普森公式。这只需要增加两个$f(x)$来求值计算，其结果为

$$\begin{aligned}S(a_{k1},b_{k1})+S(a_{k2},b_{k2})=&\frac{h}{6}(f(a_{k1})+4f(c_{k1})+f(b_{k1}))\\&+\frac{h}{6}(f(a_{k2})+4f(c_{k2})+f(b_{k2}))\end{aligned} \tag{17}$$

其中$a_{k1}=a_k,b_{k1}=a_{k2}=c_k,b_{k2}=b_k$，$c_{k1}$是$[a_{k1},b_{k1}]$ 的中点，c_{k2}是$[a_{k2},b_{k2}]$的中点。步长为$\frac{h}{2}$。

进一步，如果有$f\in C^4[a,b]$，则存在值$d_2\in[a_k,b_k]$，使得

$$\int_{a_k}^{b_k}f(x)dx=S(a_{k1},b_{k1})+S(a_{k2},b_{k2})-\frac{h^5}{16}\frac{f^{(4)}(d_2)}{90} \tag{18}$$

因为对于辛普森公式，在区间$[a_k,b_k]$上有

$$\int_{a_k}^{b_k}f(x)dx=S(a_k,b_k)-h^5\frac{f^{(4)}(d_1)}{90}\quad d_1\in[a_k,b_k] \tag{19}$$

设$f^{(4)}(d_1)\approx f^{(4)}(d_2)$则由(18)与(19)右端比较得到关系

$$-h^5\frac{f^{(4)}(d_2)}{90}\approx\frac{16}{15}(S(a_{k1},b_{k1})+S(a_{k2},b_{k2})-S(a_k,b_k)) \tag{20}$$

将(20)带入到(19)得到误差估计

$$\begin{aligned}&\left|\int_{a_k}^{b_k}f(x)dx-S(a_{k1},b_{k1})-S(a_{k2},b_{k2})\right|\\&\qquad\approx\frac{1}{15}|S(a_{k1},b_{k1})+S(a_{k2},b_{k2})-S(a_k,b_k)|\end{aligned} \tag{21}$$

由于假设$f^{(4)}(d_1)\approx f^{(4)}(d_2)$，因此在使用该方法时用$\frac{1}{10}$替换(21)右端的$\frac{1}{15}$

精度测试 设对区间$[a_k, b_k]$指定容差$\epsilon_k > 0$。如果

$$\frac{1}{10}|S(a_{k1}, b_{k1}) + S(a_{k2}, b_{k2}) - S(a_k, b_k)| < \epsilon_k \tag{22}$$

则误差小于给定的容差ϵ_k

步骤 通过辛普森公式实现自适应积分。从$\{[a_0, b_0], \epsilon_0\}$开始，其中$\epsilon_0$是区间$[a_0, b_0]$上的数值积分的容差。该区间细分为两个子区间，如果通过了精度测试，则在区间$[a_0, b_0]$上使用公式(19)，过程结束；否则，在两个子区间上采用容差$\epsilon_1 = \epsilon_0/2$，对它们进行进一步的测试。就这样细分、测试下去。不过需要保证所有子区间的容差加起来和ϵ_0相等。

2 实验目的

- 掌握组合梯形公式以及组合辛普森公式，会估计误差并得到达到合适精度的步长，会编程实现。
- 掌握改进龙贝格积分，会估计精度，并编程实现。
- 掌握自适应积分原理，步骤。会编程实现。

3 实验内容

- P290 1.

  (a) 对习题中的每个积分，计算M和步长h，使得组合梯形公式计算得到的精确到小数点后9位的结果。

  (b) 对习题中的每个积分，计算M和步长h，使得组合辛普森公式计算得到的精确到小数点后9位的结果。

$$(a)\int_{-1}^{1}(1+x^2)^{-1}dx \quad (b)\int_{0}^{1}(2+sin(2\sqrt{x}))dx \quad (c)\int_{0.25}^{4}dx/\sqrt{x}$$
$$(d)\int_{0}^{4}x^2e^{-x}dx \quad (e)\int_{0}^{2}2xcos(x)dx \quad (f)\int_{0}^{\pi}sin(2x)e^{-x}dx$$

- P301 2. 利用龙贝格积分求下面两个顶积分，精确到小数点后10位。两个顶积分的精确值都是π，解释两个龙贝格序列中明显的积分速度差别。

$$(a)\int_0^2 \sqrt{(}4x-x^2)dx \qquad (b)\int_0^1 \frac{4}{1+x^2}dx$$

- P307 1. 用自适应积分求一下的积分的近似值，起始容差$\epsilon_0 = 0.00001$

$$(a)\int_0^3 \frac{sin(2x)}{1+x^5}dx \quad (b)\int_0^3 sin(4x)e^{-2x}dx \quad (c)\int_0 .04^1 \frac{1}{\sqrt{x}}dx$$
$$(d)\int_0^3 \frac{1}{x^2+\frac{1}{10}}dx \quad (e)\int_{frac12\pi}^2 sin(\frac{1}{x})dx \quad (f)\int_0^2 \sqrt{4x}-x^2dx$$

4 实验分析

- P290 1. 在编程实现时，组合梯形公式和组合辛普森公式都选择第二种等价形式，即式(2)和(5) 。这样在每个点上函数值只用计算一次即可。

  在使用梯形公式计算积分的时候，由(9)可以计算合适的M值，然后每题的M值分析如下

  (a) 函数的二阶导数为$\frac{8x^2-2(x^2+1)}{(x^2+1)^3}$，在区间上的绝对值最大值点为$f(0)=2$，所以误差项$|-\frac{4}{3}\frac{1}{M^2}| < 1\times 10^{-9}$，解得$M \geq 36515$。

  (b) 函数的二阶导数为$-\frac{cos(2\sqrt{x})}{2x^{\frac{3}{2}}} - \frac{sin(2\sqrt{x})}{x}$，在区间上的绝对值最大值点为$f(1)=2+\sin(2)$，所以误差项$|-\frac{2+sin2}{12M^2}| < 1\times 10^{-9}$，解得$M \geq 15571$。

  (c) 函数的二阶导数为$\frac{3}{4x^{\frac{5}{2}}}$，在区间上的绝对值最大值点为$f(0.25)=24$，所以误差项$|-\frac{105.46875}{M^2}| < 1\times 10^{-9}$，解得$M \geq 324760$。

  (d) 函数的二阶导数为$\frac{x^2-4x+2}{e^x}$，在区间上的绝对值最大值点为$f(0)=2$，所以误差项$|-\frac{32}{3M^2}| < 1\times 10^{-9}$，解得$M \geq 103280$。

  (e) 函数的二阶导数为$-2x\cos(x)-4\sin(x)$，在区间上的绝对值最大值约为4.7，所以误差项$|-\frac{9.4}{3M^2}| < 1\times 10^{-9}$，解得$M \geq 55976$。

  (f) 函数的二阶导数为$\frac{-3sin(2x)-4cos(2x)}{e^x}$，在区间上的绝对值最大值点为$f(0)=-4$，所以误差项$|-\frac{\pi^3}{3M^2}| < 1\times 10^{-9}$，解得$M \geq 101664$。

在使用辛普森公式计算积分的时候，由(12)可以计算合适的M值，然后每题的M值分析如下

(a) 函数的四阶导数为$\frac{8x^2-2(x^2+1)}{(x^2+1)^3}$，在区间上的绝对值最大值点为$f(0)=24$，所以误差项$|-\frac{16}{15M^4}|<1\times10^{-9}$，解得$M\geq181$。

(b) 函数的四阶导数为$-\frac{15cos(2\sqrt{x})}{8x^{\frac{7}{2}}}-\frac{15sin(2\sqrt{x})}{4x^3}-\frac{3cos(2\sqrt{x})}{x^{\frac{5}{2}}}-\frac{sin(2\sqrt{x})}{x^2}$，在区间上的绝对值最大值点为?????，所以误差项$|?????|<1\times10^{-9}$，解得$M\geq$????。

(c) 函数的四阶导数为$\frac{105}{16x^{\frac{9}{2}}}$，在区间上的绝对值最大值点为$f(0.25)=3360$，所以误差项$|-\frac{3.75^5*336}{18M^4}|<1\times10^{-9}$，解得$M\geq1929$。

(d) 函数的四阶导数为$\frac{x^2-8x+12}{e^x}$，在区间上的绝对值最大值点为$f(0)=12$，所以误差项$|-\frac{4^5}{15M^4}|<1\times10^{-9}$，解得$M\geq512$。

(e) 函数的四阶导数为$2x\cos(x)+8\sin(x)$，在区间上的绝对值最大值点约为8.4，所以误差项$|-\frac{8.4*24}{90M^2}|<1\times10^{-9}$，解得$M\geq218$。

(f) 函数的四阶导数为$\frac{24cos(2x)-7sin(2x)}{e^x}$，在区间上的绝对值最大值点为$f(0)=24$，所以误差项$|-\frac{2\pi^5}{15M^4}|<1\times10^{-9}$，解得$M\geq450$。

- P301 2. 由龙贝格公式(16)很容易看出这个计算公式是使用递归计算的。所以在编程时候采用递归方式实现。递归到连续梯形公式时候再返回，也可以选择向下到连续布尔公式时候就返回。同时建立一个矩阵来保存已经计算出来的$R(J,K)$。

- P307 1. 根据自适应积分原理编程即可。

5 实验结果

5 .1 P290 1.

	使用组合梯形公式计算结果	使用组合辛普森公式计算结果	精确值
(a)	1.5707963265	1.5707963268	1.570796326794897
(b)	2.8707953358	2.8707904148	2.870795549959983
(c)	3.0000000000	3.0000000000	3.0
(d)	1.5237933889	1.5237933889	1.523793388892911
(e)	0.8048960335	0.8048960342	0.804896034208442
(f)	0.3827144325	0.3827144327	0.382714432694491

Table 1: 组合梯形公式和组合辛普森公式计算结果
(计算结果保留到小数点后10位)

5 .2 P301 2.

对第一个函数的龙贝格积分表如题所示

结果是题目(a)中计算到$R(22, 22) = 3.1415926535$，题目(b)计算到$R(4, 4) = 3.1415926653$

5 .3 P307 1.

各个题目的定积分近似值如下表所示

J \ K	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
0	2.0000000000																						
1	2.7320508076	2.9760677434																					
2	2.9957090681	3.0835951549	3.0907636490																				
3	3.0898191444	3.1211891698	3.1236954374	3.1242181642																			
4	3.1232530378	3.1343976690	3.1352782356	3.1354620895	3.1355061834																		
5	3.1351024229	3.1390522179	3.1393625212	3.1394273511	3.1394429011	3.1394467493																	
6	3.1392969128	3.1406950761	3.1408046000	3.1408274901	3.1408329808	3.1408343397	3.1408346785																
7	3.1407807924	3.1412754189	3.1413141085	3.1413221959	3.1413241359	3.1413246160	3.1413247358	3.1413247657															
8	3.1413055830	3.1414805131	3.1414941861	3.1414974445	3.1414977301	3.1414978998	3.1414979422	3.1414979527	3.1414979554														
9	3.1414911527	3.1415530093	3.1415578424	3.1415588528	3.1415590952	3.1415591552	3.1415591701	3.1415591730	3.1415591748	3.1415591750													
10	3.1415567665	3.1415786378	3.1415803464	3.1415807036	3.1415807893	3.1415808105	3.1415808158	3.1415808171	3.1415808174	3.1415808175	3.1415808175												
11	3.1415799654	3.1415876984	3.1415883024	3.1415884287	3.1415884590	3.1415884665	3.1415884684	3.1415884688	3.1415884689	3.1415884690	3.1415884690	3.1415884690											
12	3.1415881676	3.1415900017	3.1415911152	3.1415911599	3.1415911706	3.1415911732	3.1415911739	3.1415911741	3.1415911741	3.1415911741	3.1415911741	3.1415911741	3.1415911741										
13	3.1415910675	3.1415920342	3.1415921097	3.1415921255	3.1415921293	3.1415921302	3.1415921304	3.1415921305	3.1415921305	3.1415921305	3.1415921305	3.1415921305	3.1415921305	3.1415921305									
14	3.1415920928	3.1415923446	3.1415924613	3.1415924669	3.1415924682	3.1415924685	3.1415924686	3.1415924686	3.1415924687	3.1415924687	3.1415924687	3.1415924687	3.1415924687	3.1415924687	3.1415924687								
15	3.1415924553	3.1415925762	3.1415925856	3.1415925876	3.1415925881	3.1415925882	3.1415925882	3.1415925882	3.1415925882	3.1415925882	3.1415925882	3.1415925882	3.1415925882	3.1415925882	3.1415925882	3.1415925882							
16	3.1415925835	3.1415926262	3.1415926296	3.1415926303	3.1415926304	3.1415926305	3.1415926305	3.1415926305	3.1415926305	3.1415926305	3.1415926305	3.1415926305	3.1415926305	3.1415926305	3.1415926305	3.1415926305	3.1415926305						
17	3.1415926288	3.1415926439	3.1415926451	3.1415926453	3.1415926454	3.1415926454	3.1415926454	3.1415926454	3.1415926454	3.1415926454	3.1415926454	3.1415926454	3.1415926454	3.1415926454	3.1415926454	3.1415926454	3.1415926454	3.1415926454					
18	3.1415926448	3.1415926502	3.1415926506	3.1415926507	3.1415926507	3.1415926507	3.1415926507	3.1415926507	3.1415926507	3.1415926507	3.1415926507	3.1415926507	3.1415926507	3.1415926507	3.1415926507	3.1415926507	3.1415926507	3.1415926507	3.1415926507				
19	3.1415926505	3.1415926524	3.1415926525	3.1415926526	3.1415926526	3.1415926526	3.1415926526	3.1415926526	3.1415926526	3.1415926526	3.1415926526	3.1415926526	3.1415926526	3.1415926526	3.1415926526	3.1415926526	3.1415926526	3.1415926526	3.1415926526	3.1415926526			
20	3.1415926525	3.1415926532	3.1415926532	3.1415926532	3.1415926532	3.1415926532	3.1415926532	3.1415926532	3.1415926532	3.1415926532	3.1415926532	3.1415926532	3.1415926532	3.1415926532	3.1415926532	3.1415926532	3.1415926532	3.1415926532	3.1415926532	3.1415926532	3.1415926532		
21	3.1415926532	3.1415926534	3.1415926535	3.1415926535	3.1415926535	3.1415926535	3.1415926535	3.1415926535	3.1415926535	3.1415926535	3.1415926535	3.1415926535	3.1415926535	3.1415926535	3.1415926535	3.1415926535	3.1415926535	3.1415926535	3.1415926535	3.1415926535	3.1415926535	3.1415926535	
22	3.1415926535	3.1415926535	3.1415926535	3.1415926535	3.1415926535	3.1415926535	3.1415926535	3.1415926535	3.1415926535	3.1415926535	3.1415926535	3.1415926535	3.1415926535	3.1415926535	3.1415926535	3.1415926535	3.1415926535	3.1415926535	3.1415926535	3.1415926535	3.1415926535	3.1415926535	3.1415926535

Table 2: 题目(a)的龙贝格积分表

J \ K	0	1	2	3	4	5	6
0	3.0000000000						
1	3.1000000000	3.1333333333					
2	3.1311764706	3.1415686275	3.1421176471				
3	3.1389884945	3.1415925025	3.1415940941	3.1415857838			
4	3.1409416120	3.1415926512	3.1415926611	3.1415926384	3.1415926653		
5	3.1414298932	3.1415926536	3.1415926537	3.1415926536	3.1415926536	3.1415926536	
6	3.1415519635	3.1415926536	3.1415926536	3.1415926536	3.1415926536	3.1415926536	3.1415926536

Table 3: 题目(b)的龙贝格积分表

a_k	b_k	$S(a_{k1}, b_{k1}) + S(a_{k2}, b_{k2})$	误差估计	该区间内的容差
0.00000	0.18750	0.03474388056	0.00000002726	0.000000625
0.18750	0.37500	0.09913596173	0.00000024608	0.000000625
0.37500	0.56250	0.14636735471	0.00000000428	0.000000625
0.56250	0.65625	0.08093275622	0.00000003482	0.000000313
0.65625	0.75000	0.07867739972	0.00000005217	0.000000313
0.75000	0.93750	0.12944248437	0.00000050515	0.000000625
0.93750	1.03125	0.04494625631	0.00000003513	0.000000313
1.03125	1.12500	0.03190181868	0.00000003580	0.000000313
1.12500	1.31250	0.03377087425	0.00000044964	0.000000625
1.31250	1.50000	0.00988154078	0.00000005976	0.000000625
1.50000	1.68750	-0.00048854749	0.00000010543	0.000000625
1.68750	1.87500	-0.00392947611	0.00000006309	0.000000625
1.87500	2.25000	-0.00795629170	0.00000069702	0.000001250
2.25000	3.00000	-0.00566584592	0.00000215176	0.000002500

Table 4: $\int_0^3 \frac{sin(2x)}{1+x^5} dx$近似值及其验证

(a)中的近似值为0.67176016612, Matlab中计算结果为0.67175786463.

a_k	b_k	$S(a_{k1},b_{k1})+S(a_{k2},b_{k2})$	误差估计	该区间内的容差
0.00000	0.09375	0.01535140911	0.00000007647	0.000000313
0.09375	0.18750	0.03722398916	0.00000004720	0.000000313
0.18750	0.37500	0.09362378357	0.00000028600	0.000000625
0.37500	0.46875	0.03989835930	0.00000001574	0.000000313
0.46875	0.56250	0.02943000420	0.00000002348	0.000000313
0.56250	0.65625	0.01798395026	0.00000002540	0.000000313
0.65625	0.75000	0.00751935641	0.00000002306	0.000000313
0.75000	0.93750	-0.00709839883	0.00000048465	0.000000625
0.93750	1.12500	-0.01918599294	0.00000011632	0.000000625
1.12500	1.31250	-0.01594353155	0.00000011194	0.000000625
1.31250	1.50000	-0.00697260601	0.00000016753	0.000000625
1.50000	1.68750	0.00051839611	0.00000011562	0.000000625
1.68750	1.87500	0.00381500992	0.00000003715	0.000000625
1.87500	2.25000	0.00540235184	0.00000093581	0.000001250
2.25000	2.62500	-0.00060622730	0.00000060373	0.000001250
2.62500	3.00000	-0.00124594045	0.00000016846	0.000001250

Table 5: $\int_0^3 sin(4x)e^{-2x}dx$近似值及其验证

(b)中的近似值为0.19971391279,Matlab中计算结果为0.19971466216

a_k	b_k	$S(a_{k1},b_{k1})+S(a_{k2},b_{k2})$	误差估计	该区间内的容差
0.04000	0.05500	0.06904168200	0.00000015006	0.000000156
0.05500	0.07000	0.06010871599	0.00000004316	0.000000156
0.07000	0.08500	0.05394493839	0.00000001631	0.000000156
0.08500	0.10000	0.04936034752	0.00000000734	0.000000156
0.10000	0.13000	0.08865478486	0.00000008908	0.000000313
0.13000	0.16000	0.07888976622	0.00000003118	0.000000313
0.16000	0.22000	0.13808336401	0.00000030012	0.000000625
0.22000	0.28000	0.12021743199	0.00000008633	0.000000625
0.28000	0.40000	0.20661104471	0.00000070473	0.000001250
0.40000	0.52000	0.17730956971	0.00000017816	0.000001250
0.52000	0.76000	0.30134001823	0.00000131482	0.000002500
0.76000	1.00000	0.25644063722	0.00000030826	0.000002500

Table 6: $\int_0 .04^1 \frac{1}{\sqrt{x}}dx$近似值及其验证

(c)中的近似值为1.60000230085,Matlab计算结果为1.6

a_k	b_k	$S(a_{k1},b_{k1})+S(a_{k2},b_{k2})$	误差估计	该区间内的容差
0.00000	0.03125	0.31148868141	0.00000002237	0.000000156
0.03125	0.06250	0.30555889310	0.00000001631	0.000000156
0.06250	0.12500	0.57334409611	0.00000008097	0.000000313
0.12500	0.18750	0.50206738886	0.00000028158	0.000000313
0.18750	0.25000	0.42299080736	0.00000027735	0.000000313
0.25000	0.31250	0.34944769491	0.00000015332	0.000000313
0.31250	0.37500	0.28697931377	0.00000005834	0.000000313
0.37500	0.50000	0.43207357330	0.00000000615	0.000000625
0.50000	0.62500	0.30210806987	0.00000036628	0.000000625
0.62500	0.75000	0.21942754073	0.00000025364	0.000000625
0.75000	0.87500	0.16513907291	0.00000014411	0.000000625
0.87500	1.00000	0.12813500930	0.00000007970	0.000000625
1.00000	1.25000	0.18497468734	0.00000109982	0.000001250
1.25000	1.50000	0.12651500511	0.00000039460	0.000001250
1.50000	1.75000	0.09171750183	0.00000016041	0.000001250
1.75000	2.00000	0.06943333346	0.00000007246	0.000001250

Table 7: $\int_0^3 \frac{1}{x^2+\frac{1}{10}}dx$近似值及其验证

(d)中的近似值为4.47140066937,Matlab计算结果为4.47139939436

a_k	b_k	$S(a_{k1},b_{k1})+S(a_{k2},b_{k2})$	误差估计	该区间内的容差
0.15915	0.17354	-0.00376406532	0.00000006228	0.000000078
0.17354	0.18792	-0.00969342261	0.00000002420	0.000000078
0.18792	0.21668	-0.02736991953	0.00000002172	0.000000156
0.21668	0.24544	-0.02639794167	0.00000013914	0.000000156
0.24544	0.27421	-0.01862932304	0.00000009205	0.000000156
0.27421	0.30297	-0.00917652665	0.00000004515	0.000000156
0.30297	0.33173	-0.00033337701	0.00000001924	0.000000156
0.33173	0.38926	0.02019346357	0.00000012582	0.000000313
0.38926	0.44679	0.03884992190	0.00000002557	0.000000313
0.44679	0.50431	0.04936593393	0.00000003567	0.000000313
0.50431	0.56184	0.05474397759	0.00000002510	0.000000313
0.56184	0.61937	0.05702173670	0.00000001556	0.000000313
0.61937	0.73442	0.11428409203	0.00000023325	0.000000625
0.73442	0.84947	0.10955807958	0.00000008500	0.000000625
0.84947	1.07958	0.19812804175	0.00000066617	0.000001250
1.07958	2.00000	0.56728137640	0.00000277045	0.000005000

Table 8: $\int_{frac12\pi}^{2} sin(\frac{1}{x})dx$近似值及其验证

(e)中的近似值为1.11406204761，Matlab计算结果为1.11407449426

a_k	b_k	$S(a_{k1},b_{k1})+S(a_{k2},b_{k2})$	误差估计	该区间内的容差
0.00000	0.00000	0.00000000122	0.00000000000	0.000000000
0.00000	0.00000	0.00000000227	0.00000000000	0.000000000
0.00000	0.00000	0.00000000642	0.00000000000	0.000000000
0.00000	0.00001	0.00000001816	0.00000000000	0.000000000
0.00001	0.00002	0.00000005137	0.00000000000	0.000000000
0.00002	0.00003	0.00000014531	0.00000000000	0.000000000
0.00003	0.00006	0.00000041100	0.00000000000	0.000000000
0.00006	0.00012	0.00000116248	0.00000000001	0.000000000
0.00012	0.00024	0.00000328798	0.00000000002	0.000000001
0.00024	0.00049	0.00000929978	0.00000000006	0.000000001
0.00049	0.00098	0.00002630357	0.00000000017	0.000000002
0.00098	0.00195	0.00007439634	0.00000000049	0.000000005
0.00195	0.00391	0.00021041337	0.00000000138	0.000000010
0.00391	0.00781	0.00059504899	0.00000000389	0.000000020
0.00781	0.01563	0.00168233345	0.00000001100	0.000000039
0.01563	0.03125	0.00475260356	0.00000003111	0.000000078
0.03125	0.06250	0.01339636085	0.00000008800	0.000000156
0.06250	0.12500	0.03752237469	0.00000024890	0.000000313
0.12500	0.18750	0.04778135836	0.00000003964	0.000000313
0.18750	0.25000	0.05540241520	0.00000001205	0.000000313
0.25000	0.37500	0.12714968016	0.00000011213	0.000000625
0.37500	0.50000	0.14112973785	0.00000003409	0.000000625
0.50000	0.75000	0.29566232523	0.00000031715	0.000001250
0.75000	1.00000	0.27459952995	0.00000009641	0.000001250
1.00000	1.50000	0.32448910799	0.00000089703	0.000002500
1.50000	2.00000	-0.21992043054	0.00000027268	0.000002500

Table 9: $\int_0^2 \sqrt{4x-x^2}dx$近似值及其验证

(f)中的近似值为1.10456794501，Matlab计算结果为1.10456949966

6 实验结果分析

6 .1 P290 1.

这几题中(b)题有一个小问题，根据分析在$x \to 0$的时候二阶导数和四阶导数值趋于负无穷，在使用误差分析(12) 和(9)的时候无法进行。只能根据尝试得到数据。

除此之外，可以由实验分析看出使用辛普森公式远远比梯形公式所需要的步数要少。通过计算可以发现辛普森公式中M的平方和梯形公式中M是大约在一个数量级的。所以也验证了辛普森的衰减速度是梯形公式的平方倍。

6 .2 P301 2.

(a)与(b)的计算时间有明显差别。在计算(a)的时候，大约需要1.25 1.28秒左右，在计算(b)的时候基本在0.001秒之内就可以运行完毕。在第一个版本中，没有使用矩阵保存已经计算过的$R(J,K)$，此时程序(a)的运行时间将近7000s，而那时(b)的计算时间依然在1s内可以完成。可以看出(a)的计算时间大约是(b)的几千倍。

寻找一下原因，发现(a)中计算到R(22,22)时才有足够精确的结果，(b)中只要计算到R(6,6)就有足够精确的结果了。观察两个函数的龙贝格积分表，可以看出在第一列中(a)的结果远远没有(b)中的精确，也就是说，(a)中低阶逼近精度不高，所以需要更多的数据来改进精度。于是(a)中需要计算大量的龙贝格积分。至于为什么(a)中低阶的龙贝格积分逼近精度不高，可以通过比较(a)与(b)中两个函数图像得出结论。如图

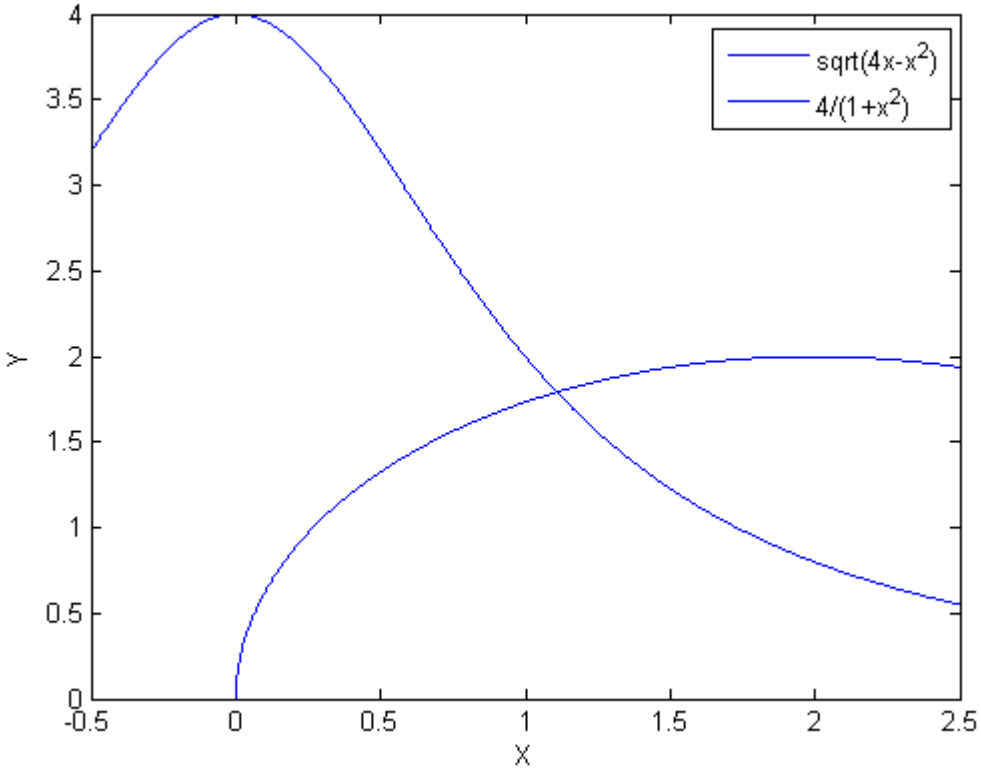

Fig 1: $\sqrt{4x-x^2}$与$\frac{4}{1+x^2}$函数图像

可以看出来(a)中函数在0附近斜率非常大，由于梯形公式是连接端点函数值计算梯形面积，在函数图形变化很大而步长又较大的时候会产生较大的误差，而(b)中的函数斜率就小的多，图像看起来更加平缓，所以(b)的误差较小。这就是两个函数积分时间差别的根本原因。

6 .3 P307 1.

可以看出使用自适应积分之后每个函数的积分步长都随着函数本身而不停改变。最后得出来的结果经过验证都是满足起始容差的。

处于好奇，我将龙贝格积分中的题目(a)使用自适应积分程序计算，在达到同样精度的情况下所需时间大约只有前者的$\frac{1}{100}$，并且计算量粗略估计也并不是很多。而将(b)题使用自适应积分计算时候所需时间相差不大。这

初步说明自适应积分对于函数图像是否平缓反应不大。对于不同的函数都能有一个比较稳定的计算时间，不像梯形公式对于某些函数误差很大，某些函数计算误差很小的情况。我想这也许就是反应了“智能”的一个特点，就是适应性强，对于不同的情况都有较好的适应，虽然不一定是最好的适应。

7 实验结论

- 梯形公式精度没有辛普森公式高，收敛的也慢。辛普森公式收敛速度大约是梯形公式的平方倍数量级。
- 梯形公式在函数图形比较平缓的时候效果较好，在函数图像变化较快的时候精度很差。说明梯形公式只适合计算某一类函数。
- 在使用龙贝格积分的时候，或者类似的递归计算的时候，保存底层的数据是一项重要的措施，甚至能提高几千倍的计算速度。
- 自适应积分对于所计算的函数的特征依赖较小，对于大多数函数都能在稳定的时间内完成计算，这一点在需要面对多样化的函数的时候尤其重要。

8 实验代码展示

8 .1 P290 1.

main.c

```
#include <stdio.h>
#include <stdlib.h>
#include "operation.h"

int main()
{
    freopen("tf5.txt","r",stdin);
    //freopen("tf5_out.txt","w",stdout);
```

```
    double left_end, right_end;
    double M;
//Initialize the parameters
    initialize_parameters(&left_end,&right_end,&M);
//To intergrate in the interval
    intergrate(left_end,right_end,M);
    return 0;
}
```

operation.h

```
#ifndef OPERATION_H_INCLUDED
#define OPERATION_H_INCLUDED

/*----------------Includes---------------*/
//the head files that were included
#include <math.h>
/*----------------Define---------------*/
#define ebase 2.718281828459

double function(double x);

/*********************************************************
 Function Name: initialize_parameters
 Function Description: Initialize some parameters for integration.
Inputs: the left end and right end of the interval integrated.
 Outputs: No output
*********************************************************/
void initialize_parameters(double *left_end,double *right_end,double *M)
{
    printf("the left end and right end of interval:");
    scanf("%lf%lf",left_end,right_end);
    printf("The interval would be subdivided into M subintervals:");
```

```
    scanf("%lf",M);
}

/*********************************************************
Function Name:integrate
Description:integrate the function over interval assigned.
        One way is Composite Trapezoidal, another is Composite
        Simpson's.
Input:'Xnode',nodes of x, interval's ends and number of subinterval.
    and 'mode' to choose which way will be used to integrate.
Output:none.
*********************************************************/
void intergrate(double left_end, double right_end, double M)
{
    char mode;
    printf("the mode you'll use (T or S):\n");
    scanf("%c",&mode);
    printf("mode = %c\n",mode);

    if(mode == 'T'){
        printf("T mode\n");
        composite_trapezoidal(left_end,right_end,M);
    }
    if(mode == 'S'){
        printf("S mode\n");
        composite_simpson(left_end,right_end,M);
    }
}

/*********************************************************
Function Name:composite_trapezoidal
Description:use composite trapezoidal rule to integrate.
Input:'Xnode',nodes of x, interval's ends and number of subinterval.
```

```
Output:None
************************************************************/
void composite_trapezoidal(double left_end, double right_end, double M)
{
    int i;
    double h;
    double integration_2, integration_1;

    integration_1 = 0;
    integration_2 = 0;
    h = (right_end - left_end) / M;

    //calculate the second part
    integration_1 += h * (function(left_end) + function(right_end)) / 2;
    //calculate the first part of result
    for(i = 1; i < M; i++){
        integration_2 += function(left_end + i * h);
    }
    integration_2 *= h;

    printf("the parameters is:\nleft end = %lf\nright end = %lf\nM = %lf\n"
                                                ,left_end,right_end,M);
    printf("the result is: %.10lf\n",integration_1 + integration_2);
}
/************************************************************
Function Name:composite_trapezoidal
Description:use composite trapezoidal rule to integrate.
Input:'Xnode',nodes of x, interval's ends and number of subinterval.
Output:None
************************************************************/
void composite_simpson(double left_end, double right_end, double M)
{
    int i;
```

```
    double h;
    double integration_1,integration_2,integration_3;

    integration_1 = 0;
    integration_2 = 0;
    integration_3 = 0;
    h = (right_end - left_end) / M / 2;

    //calculate the first part of result
    integration_1 += h * (function(left_end) + function(right_end)) / 3;
    //calculate the second part of result
    for(i = 1; i < M; i++){
        integration_2 += function(left_end + (2 * i) * h);
    }
    integration_2 *= (2 * h) / 3;
    //calculate the third part
    for(i = 1; i <= M; i++){
        integration_3 += function(left_end + (2 * i - 1) * h);
    }
    integration_3 *= (4 * h) / 3;

    printf("the parameters is:\nleft end = %lf\nright end = %lf\nM = %lf\n"
                          ,left_end,right_end,M);
    printf("the result is: %.10lf\n",integration_1 + integration_2 + integration_3);
}

/*************************************************************
Function Name:function
Description:return a mathematical function's value by the
                    given number.
Input:a number x
Output:the value of f(x)
*************************************************************/
```

```
double function(double x)
{
    //return 1/(1+pow(x,2));//-1~1,M>=18258
    //return (2 + sin(2 * sqrt(x)));//0~1,M>=unknown
    //return 1/sqrt(x);//0.25~4,M>=324760
    //return pow(x,2) * pow(ebase,-x);//0~4,M>=103280
    return 2 * x * cos(x);//0~2,M>=36264
    //return sin(2 * x) * pow(ebase,-x);//0~pi,M>=101664
}
#endif // OPERATION_H_INCLUDED
```

8 .2 P301 2.

main.c

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "operation.h"
#define PI 3.141592653589793
int main()
{
    double left_end, right_end;//endpoints of intervals
    double J,K;//R(J,K)
    double error,tol,result;//tol->tolerance, result->calculation's result
    double **ma;//matrix
    int i,j;//for counting

    //Initialize the matrix "ma"
    ma = malloc(25*sizeof(double*));printf("OK\n");
    for(i = 0; i<25;i++){
        for(j = 0; j < 25; j++){
            *(ma+i) = malloc(25*sizeof(double));
```

```
            *(*(ma+i)+j) = 0;
        }
    }
//Initialize parameters
    initialize_parameters(&left_end, &right_end, &J, &K);
    tol = 1e-10;
    error = 0;
//calculate R(J,K)
    do{
        result = romberg(ma,J,K,left_end,right_end);
        error = fabs( *(*(ma+(int)J)+(int)K) - *(*(ma+(int)J-1)+(int)K-1));
        J++;
        K++;
    }while(error > tol);
    //freopen("matrix1.txt","w",stdout);
    printf("R(%.1lf,%.1lf) result = %.11lf\n",J,K,result);
    printf("error=%.12lf\n",result,error);
    return 0;
}
```

operation.h

```
#ifndef OPERATION_H_INCLUDED
#define OPERATION_H_INCLUDED

/*---------------Includes--------------*/
//the head files that were included
#include <math.h>

/*----------------define-----------------*/
#define PI 3.141592653589793

double function(double x);
```

```
double trapozidal(double (*f)(),double J, double left_end, double right_end);

/*********************************************************
 Function Name: initialize_parameters
 Function Description: Initialize some parameters for integration.
Inputs: the left end and right end of the interval integrated.
 Outputs: No output
*************************************************************/
void initialize_parameters(double *left_end,double *right_end,double *J,double *K)
{
    printf("the left end and right end of interval:\n");
    scanf("%lf%lf",left_end,right_end);
    printf("The J & K:\n");
    scanf("%lf%lf",J,K);
}

/*************************************************************
Function Name: romberg
Description:using romberg rule to integrate.
Input:matrix ma, J & K, endpoints of interval.
Output:The value R(J,K)
*************************************************************/
double romberg(double **ma, double J, double K, double left_end, double right_end)
{
    //printf("calculating R(%.0lf,%.0lf)\n",J, K);
    int i,j;
    double (*f)();
    f = function;
    //R(J,0) need to return values
    if(K == 0){
        if(*(*(ma + (int)J)+(int)K) == 0){
                *(*(ma + (int)J)+(int)K) =
                                    trapozidal(f, J, left_end, right_end);
```

```
        }
        return *(*(ma+(int)J)+(int)K);
    }
    else{
            //to save value R(J,K-1) in matrix
            if(*(*(ma+(int)J)+(int)(K-1)) == 0){
                *(*(ma+(int)J)+(int)(K-1)) =
                            romberg(ma,J,K-1,left_end,right_end);
            }
            //to save value R(J-1,K-1) in matrix
            if(*(*(ma+(int)J-1)+(int)(K-1)) == 0){
                *(*(ma+(int)(J-1))+(int)(K-1)) =
                                    romberg(ma,J-1, K-1,left_end,right_end);
            }
            //save value R(J,K) into matrix
            *(*(ma+(int)J)+(int)K) = (pow(4,K) * (*(*(ma+(int)J)+(int)(K-1)))-
                                    *(*(ma+(int)(J-1))+(int)K-1)) / (pow(4,K) - 1);
            //return value to R(J,K)
            return *(*(ma+(int)J)+(int)K);
    }
}
/*************************************************************
Function Name: trapozidal
Description:using trapozidal rule to integrate
Input:J, endpoints of interval.and f means which function to be integrated.
Output:The value integration
*************************************************************/
double trapozidal(double (*f)(),double J, double left_end, double right_end)
{
    int i;
    double M;
    double h;
    double sec_part = 0;
```

```
    //Initialize the step length
    h = (right_end - left_end) / pow(2,J);
//using recursive trapozidal rule
    if(J == 0){
        return h * (f(left_end) + f(right_end)) / 2;
    }
// calculate two parts to add for result
    else{
        M = pow(2, J - 1);
        sec_part = 0;
        for(i = 1; i <= M; i++){
            sec_part += f(left_end + (2 * i -1) * h);
        }
        sec_part *= h;
        return trapozidal(f,J - 1,left_end,right_end) / 2 + sec_part;
    }
}
/*************************************************************
Function Name: f
Description:The function be integrated.
Input:x for calculating f(x)
Output:The value f(x)
*************************************************************/
double function(double x)
{
    //return (pow(x,2) + x + 1) * cos(x);//test
    return sqrt(4 * x - pow(x,2));
    //return 4/(1 + pow(x,2));
}
#endif // OPERATION_H_INCLUDED
```

8 .3 P307 1.

main.c

```
  #include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "operation.h"



int main()
{
    //freopen("out.txt","w",stdout);
    double tolerance, error, result;
    double left_end, right_end, mid;
    //initialize the variables
    tolerance = 0.00001;
    initialize_parameters(&left_end,&right_end);

    result = adapting(left_end, right_end, tolerance);
    printf("result=%.11lf\n",result);
    return 0;
}
```

operation.h

```
#ifndef OPERATION_H_INCLUDED
#define OPERATION_H_INCLUDED
/*----------------Includes---------------*/
//the head files that were included
#include <math.h>
/*-----------------Define-----------------*/
#define ebase 2.718281828459045

/*--------------------------------------functions----------------------*/
```

```
double simpson(double left_end, double right_end);
double f(double x);

/*********************************************************
 Function Name: initialize_parameters
 Function Description: Initialize some parameters for integration.
Inputs: the left end and right end of the interval integrated.
 Outputs: No output
*********************************************************/
void initialize_parameters(double *left_end,double *right_end)
{
    printf("the left end and right end of interval:\n");
    scanf("%lf%lf",left_end,right_end);
}
/*************************************************************
Function Name: f
Description:The function be integrated.
Input:x for calculating f(x)
Output:The value f(x)
*************************************************************/
double f(double x)
{
    //return sin(2*x)/(1 + pow(x,5));//
    //return sin(4*x)*pow(ebase,-2*x);
    //return 1/sqrt(x);//
    //return 1/(pow(x,2)+0.1);//
    //return sin(1/x);//
    //return sqrt(4*x) - pow(x,2);//

    return sqrt(4*x - pow(x,2));
}

/*************************************************************
```

```
Function Name: adapting
Description:using adapting quadrature rule to integrate.
Input:endpoints of interval, and tolerance
Output:the result of integration
************************************************************/
double adapting(double left_end, double right_end, double tolerance)
{
    double error;
    double mid;
    double result=0;
//calculate the midpoint
    mid = (left_end + right_end) / 2;
//calculate the error
    error = fabs(simpson(left_end,mid) + simpson(mid,right_end)
         - simpson(left_end, right_end));
    error /= 10;
//if error > tolerance,cut interval into two part and calculate again.
    if(error < tolerance){
        printf("%.5lf & %.5lf & %.11lf & %.11lf & %.9lf\\\\\\hline\n"
            ,left_end,right_end,simpson(left_end,mid) +
             simpson(mid,right_end),error,tolerance);
        result += simpson(left_end,mid) + simpson(mid,right_end);
    }
    else{
        result += adapting(left_end, mid, tolerance/2);
        result += adapting(mid, right_end, tolerance/2);
    }
    return result;
}
/************************************************************
Function Name: simpson
Description:using simpson rule to integrate.
Input:endpoints of interval
```

```
Output:the result of integration
************************************************************/
double simpson(double left_end, double right_end)
{
    double h;
    double mid;

    h = (right_end - left_end) / 2;
    mid = (left_end + right_end) / 2;

    return h * (f(left_end) + 4 * f(mid) + f(right_end)) / 3;
}
#endif // OPERATION_H_INCLUDED
```