山东大学数值计算课程实验报告

班级: 17级计科一班
学号: 201700162026
姓名: 王捷

目录

1 实验目的

独立完成数值计算，加深对知识点的理解，提升数值计算相关知识的运用能力。

本次实验一共八个实验:误差理论，非线性方程组求解，线性方程组的求解、插值多项式、数值微分与数值积分、常微分方程数值解、曲线拟合与函数逼近、特征值与特征方程这八章的内容。有些实验比如插值多项式、数值微分与数值积分、常微分方程求解、特征值与特征方程，在理解书本内容的基础之上,提升利用数值计算知识自身解决问题的能力。

2 实验环境及实验要求

2.1 实验环境

ASUS华硕电脑 8G内存 Matlab2019a

2.2 实验要求

2.2.1 实验一：第一章误差理论

1. 理解误差的来源、类型(模型误差？截断误差？舍入误差？浮点运算舍入误差？)
2. 误差的度量方法: 相对误差、绝对误差
3. 理解代打误差的收敛性？误差的收敛阶（定义域表达），以及阶的估计与表达。
4. 误差的传播途径、积累以及局部误差、总体误差等。

2.2.2 实验二：第二章非线性方程求解

1. 理解方程的根，不动点，迭代，收敛性和收敛速度，误差及其控制
2. 算法及其收敛速率：不动点迭代，二分法，牛顿法，割线法，试位法。
3. 理解算法的优劣性，收敛速率，初始值的选择。 [Richard.L.Burden(2005)]

2.2.3 实验三：第六、七线性方程组求解

1. 了解基本概念:响亮与矩阵范数，特殊矩阵（对称正定，对角占优）

2. 算法及其收敛速率:直接求解算法——LU分解、对称矩阵的LL^T, LDL^T分解；迭代算法 Jacobi、Gauss-Seidel、SOR

3. 注意难点：算法的优劣性，收敛速率

2.2.4 实验四：第三章插值多项式

1. 理解插值问题的基本概念、插值多项式的唯一条件
2. 理解不同边界条件的样条函数计算公式推导，以及熟悉样条插值的原理与过程。
3. 理解并运用Largrange插值基函数、n次Largrange插值多项式、Herimite插值及其误差估计、样条插值

2.2.5 实验五：第四章数值微分与数值积分

1. 熟悉掌握利用Taylor展开式构造一阶导数、二阶导数的差商近似计算格式以及其误差估计:向前差分、向后差分、中心差分。

2. 熟练掌握利用拉格朗日插值多项式构造数值差分格式的方法以及误差估计。

3. 理解掌握：数值积分定义，利用拉格朗日多项式逼近构造数值积分计算格式的基本过程、常见计算格式（梯形、抛物线型即Simpson格式、中矩）及其误差估计、代数精确度。基于误差控制的逐次半积分方法 [John.H.Mathews(2019)]。

2.2.6 实验六：第五章常微分方程数值解

1. 掌握求取微分方程数值解的基本步骤与过程，解唯一存在的条件。
2. 掌握把微分方程离散成数值计算格式的三种常见方法:泰勒法、差商法、积分法。
3. 掌握常见的欧拉法、梯形格式、预估校正格式，龙格库塔四阶格式。

2.2.7 实验七：第八章曲线拟合与函数逼近

掌握里理解曲线的拟合与函数的逼近,掌握最小二乘法。

2.2.8 实验八：第九章特征值与特征方程

1. 掌握向量的范数、矩阵的范数、向量序列收敛，收敛矩阵。
2. 谱半径，圆盘定理
3. 幂法、反幂法、对称幂法的算法、异同。
4. Housecholder变换，Givens旋转变化的相关理论以及矩阵的QR分解。

3 实验一：第一章误差理论

3.1 子实验要求

1. 了解误差来源以及误差类型：截断误差、舍入误差等。
2. 了解误差的度量方法：相对误差、绝对误差等。
3. 理解迭代序列的收敛性，误差的收敛阶以及阶的估计表达。
4. 理解误差的传播途径、误差的积累、局部误差、总体误差等。

3.1.1 作业题

第一题

1. 1.3.9(P26) 习题2，5，8

分析讨论题

1. 求方程$x^2+(\alpha+\beta)x+10^9=0$的根，其中$\alpha=-10^9,\beta=-1$,讨论如何设计计算格式才能有效地减少误差，提高计算精度.

2. 以计算x^{31}为例，讨论如何设计计算格式才能有效减少计算次数.

3.1.2 上机要求

1.3.10(P28) 算法与程序:1,2

3.2 实现思路

3.2.1 作业题

***1.3.9* 完成下列计算**$\int_0^{\frac{1}{4}} e^{x^2}dx \approx \int_0^{\frac{1}{4}} \left(1+x^2+\frac{x^2}{2!}+\frac{x^6}{3!}\right)dx = \hat{p}$**指出在这种情况下会出现哪种情况的误差，并将计算结果与真实值**$p=0.2553074606$**进行比较**

解:

该情况的误差源于截断误差，计算过程如下:

$$\int_0^{\frac{1}{4}} \left(1+x^2+\frac{x^2}{2!}+\frac{x^6}{3!}\right)dx$$

$$=\frac{1}{4}+\frac{1}{4^3\times 3}+\frac{1}{4^5 5(2!)}+\frac{1}{4^7 7(3!)}$$

$$=\frac{1}{4}+\frac{1}{192}+\frac{1}{10240}+\frac{1}{688128}=0.2553074428$$

$$p-\hat{p}=0.0000000178$$

$$\frac{(p-\hat{p})}{p}=0.0000000699$$

习题*2* 有时利用三角或者代数恒等式，重新排列函数中的项，可以避免精度损失。求下列函数的等价公式，以免精度损失

(a) $ln(x+1)-ln(x)$,其中x较大

(b) $\sqrt{x^2+1}-x$, 其中x 较大 .

(c) $cos^2(x)-sin^2(x)$,其中 $x\approx\frac{1}{4}$

(d)$\sqrt{\frac{1+cos(x)}{2}}$, 其中$x\approx\pi$.

解:

(a) $ln(x+1)-ln(x)=ln(\frac{x+1}{x})=ln(1+\frac{1}{x})$

(b) $\sqrt{x^2+1}-x=1/(\sqrt{x^2+1}+x)$

(c) $cos^2(x)-sin^2(x)=(cosx+sinx)(cosx-sinx)$

(d) $\sqrt{(\frac{1+cos(x)}{2})}=\sqrt{0.5+\frac{cos(x)}{2}}$

习题5 讨论下列计算过程中的误差传播

*(a)*三个数的和*:*

$$p+q+r=(\hat{p}+\varepsilon_p)+(\hat{q}+\varepsilon_p)+(\hat{r}+\varepsilon_r)$$

*(b)*两个数的商*:*

$$\frac{p}{q}=\frac{\hat{p}+\varepsilon_p}{\hat{q}+\varepsilon_p}$$

*(c)*三个数的积

$$pqr=(\hat{p}+\varepsilon_p)+(\hat{q}+\varepsilon_p)+(\hat{r}+\varepsilon_r)$$

解*:*

(a) 误差传播为$\varepsilon_p+\varepsilon_q+\varepsilon_r$.

(b) 有:

$$\frac{p}{q}=\frac{\hat{p}+\varepsilon_p}{\hat{q}+\varepsilon_q}=\frac{\hat{p}}{\hat{q}}+\frac{\varepsilon_p+\frac{\hat{p}}{\hat{q}}\varepsilon_q}{\hat{q}+\varepsilon_q}$$

所以，如果$1<|\hat{p}|<|\hat{p}|$,则原始误差可能会放大。

(c)

$$pqr=(\hat{p}+\varepsilon_p)+(\hat{q}+\varepsilon_q)(\hat{r}+\varepsilon_r)$$

$$=\hat{p}\hat{q}\hat{r}+\hat{p}\hat{r}\varepsilon q+\hat{q}\hat{r}\varepsilon_p+\hat{p}\hat{q}\varepsilon_r+\hat{r}\varepsilon_p\varepsilon_q+\hat{q}\varepsilon_p\varepsilon_r+\hat{p}\varepsilon_q\varepsilon_r$$

$$=\hat{p}\hat{q}\hat{r}+\hat{p}\hat{r}\varepsilon q+\hat{q}\hat{r}\varepsilon_p+\hat{p}\hat{q}\varepsilon_r+\hat{r}\varepsilon_p\varepsilon_q+\hat{q}\varepsilon_p\varepsilon_r+\hat{p}\varepsilon_q\varepsilon_r$$

$$=\hat{p}\hat{q}\hat{r}+(\hat{p}\hat{r}\varepsilon_q+\hat{q}\hat{r}\varepsilon_p+\hat{p}\hat{q}\varepsilon_r)+(\hat{r}\varepsilon_q\varepsilon_p+\hat{q}\varepsilon_p\varepsilon_r+\hat{p}\varepsilon q\varepsilon r)+\varepsilon_p\varepsilon_q\varepsilon_r$$

其中$\hat{q},\hat{q},\hat{r}$的绝对值又可能把原来的误差$\varepsilon_p\varepsilon_q$和$\varepsilon_r$

习题8 设有泰勒展开式$\cos(h)=1-\frac{h^2}{2!}+\frac{h^4}{4!}+O(h^6)$和$\sin(h)=h-\frac{h^3}{3!}+\frac{h^5}{5!}+O(h^7)$判定他们的和与积的近似阶。

解*:*

$$\begin{aligned}&\cos(h)+\sin(h)\\&=1+h-\tfrac{h^2}{2!}-\tfrac{h^3}{3!}+\tfrac{h^4}{4!}+\tfrac{h^5}{5!}+O(h^6)+O(h^7)\\&=1+\mathrm{h}-\tfrac{h^2}{2}-\tfrac{h^3}{6}+\tfrac{h^4}{24}+O(h^5)\end{aligned}$$

其中$\frac{h^5}{5!}+O(h^6)+O(h^7)=O(h^5)$

$$\begin{aligned}&\cos(h)\times\sin(h)\\&=\left(1-\tfrac{h^2}{2!}+\tfrac{h^4}{4!}+O(h^6)\right)\left(h-\tfrac{h^3}{3!}+\tfrac{h^5}{5!}+O(h^7)\right)\\&=\left(h-\tfrac{h^3}{3!}+\tfrac{h^5}{5!}+O(h^7)\right)-\tfrac{h^2}{2!}\left(h-\tfrac{h^3}{3!}+\tfrac{h^5}{5!}+O(h^7)\right)\\&+\tfrac{h^4}{4!}\left(h-\tfrac{h^3}{3!}+\tfrac{h^5}{5!}+O(h^7)\right)+O(h^6)\,h-\tfrac{h^3}{3!}+\tfrac{h^5}{5!}+O(h^7)\Big)\\&=h-\tfrac{2h^3}{3}+\tfrac{2h^5}{15}-\tfrac{h^7}{90}+\tfrac{h^9}{2880}\end{aligned}$$

分析讨论题

1. 求方程$x^2+(\alpha+\beta)x+10^9=0$的根,其中$\alpha=-10^9,\beta=-1$,讨论如何设计计算格式才能有效的减少误差，提高计算精度

解:

由于$x^2-(10^9+1)\,x+10^9=(x-10^9)\,(x-1)$

$\therefore x_1=10^9,x_2=1$

如果利用求根公式对于$\sqrt{b^2-4ac}$,若$b^2>>4ac$其结果为b，但其中必然会产生误差。对此，我们可以利用根与系数的关系，$x1x2=\frac{c}{a}$则有

$$x_1=\frac{-b-sqn(b)\sqrt{b^2-4ac}}{2a}$$
$$x_2=\frac{c}{ax}$$

其中$sqn(b)$为b的符号函数，当$b>>0$时，$sqn(b)=1$,当$b<0$时，$sqn(b)=-1$，这样就能避免上面办法产生的误差。

2. 以计算x^{31}为例，讨论如何设计计算格式才能减少计算次数

解:

我们先进行五次$x_{K+1}=(x_K)^2$运算,其中当K=0时x=x。随后我们可以获得x^{32}.然后我们再执行x^{32}/x便可以获得x^{31}的解，期间一共进行了6次运算。

3.2.2 上机题

1. 构造算法和matlab程序计算下列二次方程的根:

$$(a).\ x^2-1000.001x+1=0$$
$$(b).\ x^2-1000.0001x+1=0$$
$$(c).\ x^2-10000.00001x+1=0$$
$$(d).\ x^2-100000.00000x1+1=0$$

对于二次方程的求解，当$|b|=\approx\sqrt{b^2-4ac}$时，会由于值过小而导致误差偏大，我们可以将二次方程的求根公式进行转化(去除分子的根号)获得:

$$\mathrm{x}_1=\frac{-b+\sqrt{b^2-4ac}}{2a}\quad x_2=\frac{-b-\sqrt{b^2-4ac}}{2a}\qquad(a)$$

$$x_1=\frac{-2c}{b+\sqrt{b^2-4ac}}\quad x_2=\frac{-2c}{b-\sqrt{b^2-4ac}}\qquad(b)$$

当b¿0时，用公式(a)计算x1,用式(b)计算x2

当b¡0时，用公式(b)计算x2,用式(b)计算x1.

2. 引入一个小的初始误差，对下列三个差分方程计算前十个数值近似值，构造序列表及图进行输出

(a). $r_0=0.994,r_n=\frac{1}{2}r_{n-1},n=1,2,\cdots$

(b). $p_0=1,p_1=0.497,p_n=\frac{3}{2}p_{n-1}-\frac{1}{2}p_{n-2},n=2,3,\cdots$

(c). $q_0=1,q_1=0.497,q_n=\frac{5}{2}q_{n-1}-p_{n-2},n=2,3,\cdots$

3.3 实验结果与分析

上机实验任务一：

四个二次方程的解如下所示：

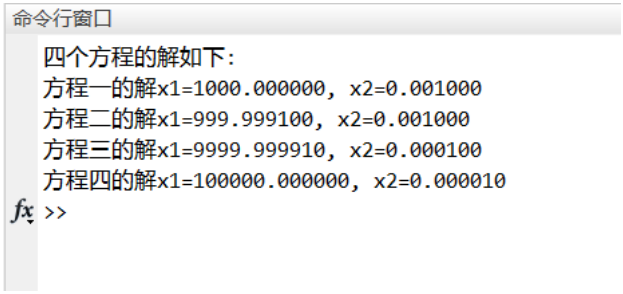

图 1: 四个二次方程解

上机实验任务二： 差分递推结果如图所示：

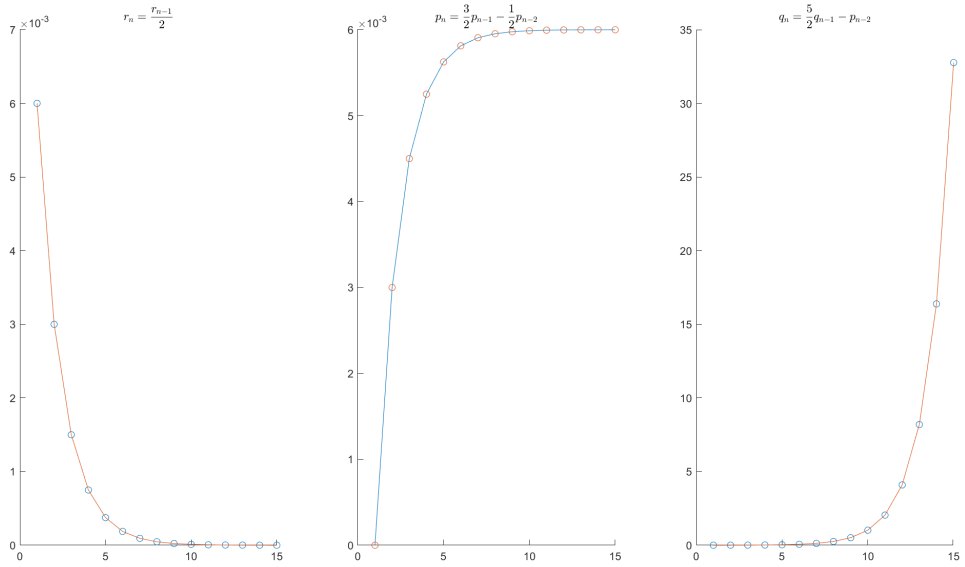

图 2: 三个差分递推结果

4 实验二：第二章非线性方程求解

4.1 子实验要求

上机要求

1. 求方程$2x^2+x-15=0$的正根($x^*=2.5$)近似值，分别利用如下三种格式编程计算:

1. x_{k+1}=15-x_k^2, k=0,1,2,..., 取初始值$x_0=2$.
2. $x_{k+1}=\frac{15}{2x_k+1}$, k=0,1,2,..., 取初始值$x_0=2$
3. $x_{k+1}=x_k-\frac{2x_k^2+x_k-15}{4x_k+1}$, x=0,1,2,..., 取初始值$x_0=2$,

依次计算$x_1,x_2,...,x_k,...$,并作图观察解的稳定性与收敛性，并分析其原因.

2. 证明方程$2-3x-sin(x)=0$在(0,1)内有且只有一个实根，使用二分法求误差不大于0.0005的根，及其需要的迭代次数。

3. 利用牛顿法求解方程

$$\frac{1}{2}+\frac{1}{4}x^2-xsin(x)-\frac{1}{2}cos(2x)=0$$

分别取$x_0=\frac{\pi}{2},5\pi,10\pi$,使得精度不超过$10^{-5}$，比较初值对计算结果的影响

4. 已知

$$f(x)=5x-e^x$$

在(0,1)之间有一个实根，试分别利用二分法、牛顿法、割线法、试位法设计相应的计算格式，并编程求解(精确到4位小数）

4.2 实现思路

针对**第一题**，利用题目所给的信息，利用matlab编程，将三种分解方法进行迭代，并将结果用图像展示出来，判断解的稳定性与收敛性。针对稳定性，我们在初始条件$x_0=2$取点判断收敛情况。针对收敛性我们求收敛时的迭代次数。

针对**第二题**,首先证明方程在(0,1)只有一个解。

证:

令$f(x)=2-3x-sin(x)$,故有$f'(x)=-3-cos(x)$。

又有$x\in(0,1)$, 所以有$f'(x)<0$恒成立, $f(x)$在(0,1)内单调递减。

又有$f(0)=2,f(1)=-2$, 故$f(x)$在(0,1)只有一个解.

得证。

随后设计二分法，从$xl=0,xr=1$开始二分搜索，将每次所得的之(中点)与正确结果做差并取绝对值，判断是否小于0.0005,如果满足，这输出迭代的次数。

针对**第三题**,我们设计牛顿法求不同初值所得的计算结果，并各个结果进行判断是否有影响。

$$f(x)=f(x_k)+f'(x_k)(x-x_k)+\frac{1}{2}f''(x_k)(x-x_k)^2$$

对上式求导并令其为0，则为

$$f'(x_k) + f''(x_k)(x - x_k) = 0$$

即得到

$$x = x_k - \frac{f'(x_k)}{f''(x_k)}$$

针对**第四题**,设计二分法、牛顿法、割线法、错位法来求根，并在上下两个结果误差绝对值小于0.0001时停止。

对于割线法，由迭代方程

$$x_{n+1} = x_n - \frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})} f(x_n)$$

我们可以编写函数来逼近零点。

对于试位法，由公式

$$\frac{f(x_l)}{x_r - x_l} = \frac{f(x_u)}{x_r - x_u}$$

推出迭代方程

$$x_r = x_u - \frac{f(x_u)(x_l - x_u)}{f(x_l) - f(x_u)} = \frac{x_u f(x_l) - x_l f(x_u)}{f(x_l) - f(x_u)}$$

利用迭代方程编程求解。

4.3 实验结果与分析

第一题:

对于$x_{k+1} = 15 - x_k^2$格式，变成计算发现函数不收敛，不具备稳定性与收敛性。

对于$x_{k+1} = \frac{15}{2x_k+1}$与$x_{k+1} = x_k - \frac{2x_k^2+x_k-15}{4x_k+1}$ 格式。编程绘制图像[1]如图所示:

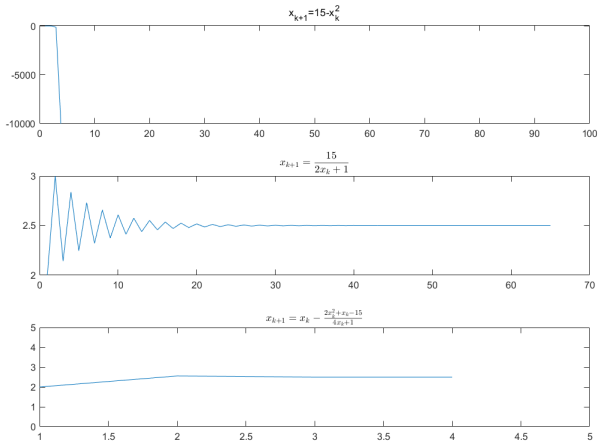

图 3: 三种不动点格式收敛结果

[1]The code is appended in the code part.

分析图像可知，第一种方式不具收敛性，而第二与第三种方法可以收敛于2.5。由计算可以获得，第二种方式需迭代64次可将误差控制在0.00001，而第三种方式只需要3次迭代即可。

随后我们给第二、三种方式的初值进行修改(分别为-5,0,1,5),查看结果。

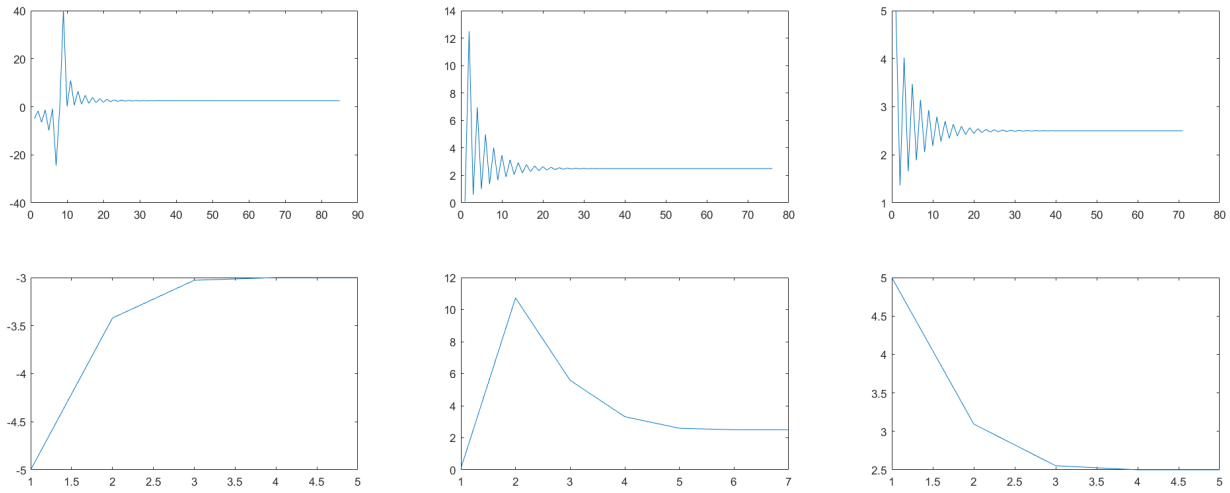

图 4: 修改初值收敛结果

由最终的结果可见，初始值的变化对方法二的影响较大，对方法上的影响小，故第三种方法的稳定性更好。

第二题：编程求解，并将迭代的中间结果用图像的展示出来，如图所示。

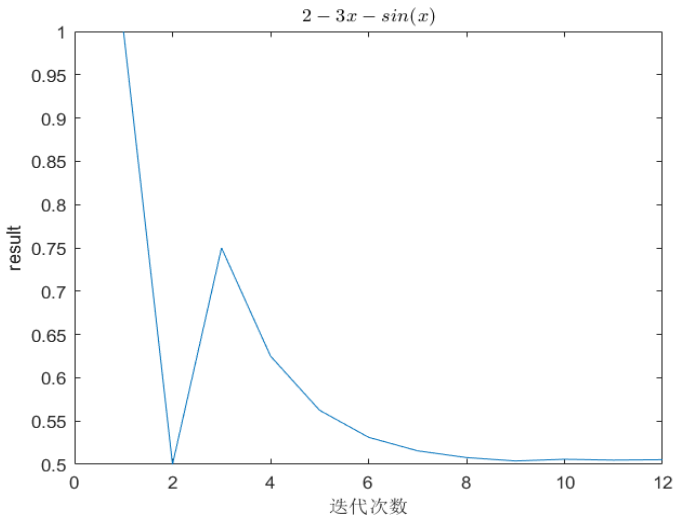

图 5: $2-3x-sin(x)$收敛情况

最终求得迭代的次数为11次。

第三题:利用牛顿法公式:

$$x1 = x0 - \frac{f(x_n)}{f'(x_n)}$$

编写代码，并分别设初值为$\frac{\pi}{4}, \frac{pi}{2}, 5\pi, 10\pi$,不断迭代，查看收敛情况。

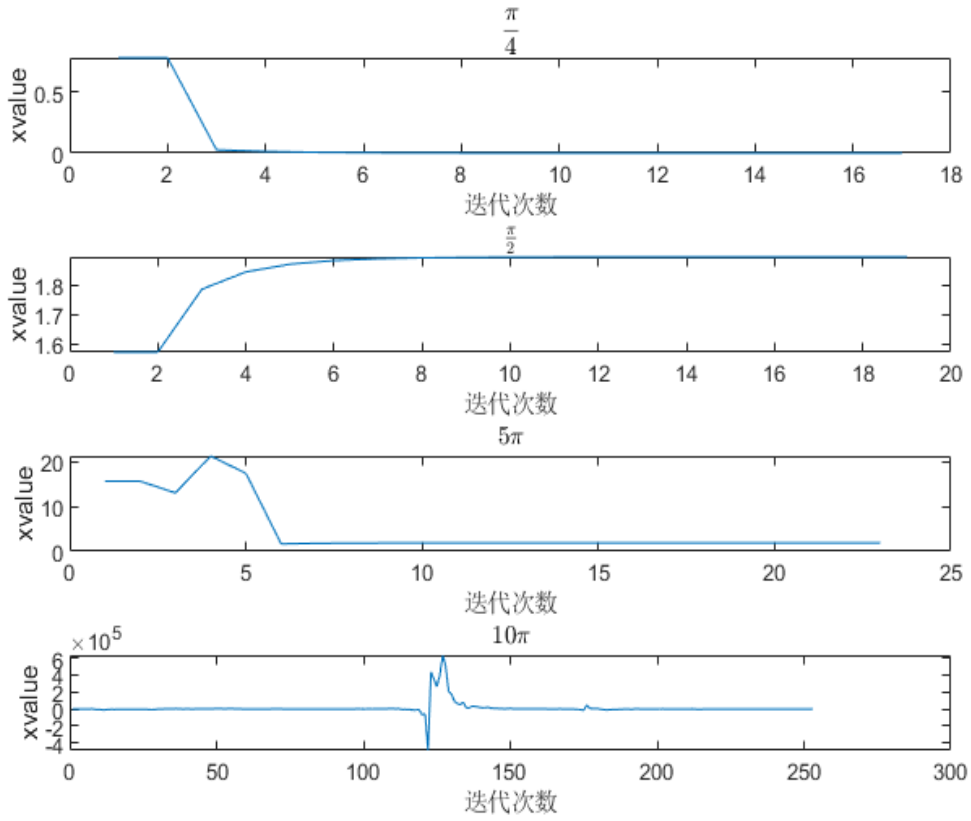

图 6: 初值对牛顿法影响

发现初值对牛顿法的收敛结果有影响。对于初值$\frac{\pi}{4}$，结果收敛于0，对于$\frac{\pi}{2}, 5\pi$结果收敛于1.8955,对于10π收敛于-1.8955。

我们先分析函数$\frac{1}{2} + \frac{x^2}{4} - xsin(x) - \frac{cos(2x)}{2}$，利用matlab绘制图像可见

发现，结果都为函数的零点。故牛顿法的初始值会影响最终的解。

第四题

实验结果如图所示,相对于牛顿法与割线法，二分法的迭代次数相对较多。最终结果收敛于x = 0.2592（保留四位小数） [Barnes(1965)]。

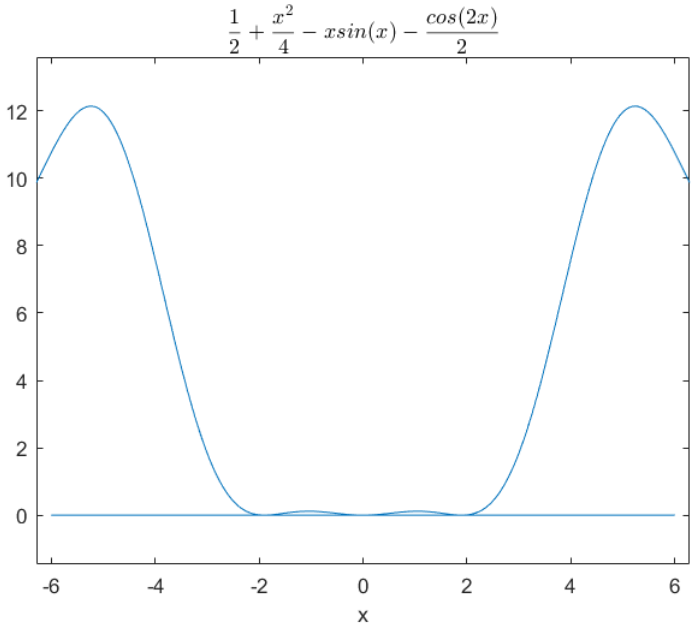

图 7: $\frac{1}{2}+\frac{x^2}{4}-xsin(x)-\frac{cos(2x)}{2}$图像

5 实验三：第六、七线性方程组求解

5.1 子实验要求

1. 求解线性方程组

$$\begin{cases} 4x-y+z=7 \\ 4x-8y+z=-21 \\ -2x+t+5z=15 \end{cases}$$

(1) 使用LU分解求解此方程组。

(2) 别试用Jacobi,Gauss-Seidel方法求解此方程组。

2. 3.6.5算法与程序(P118):3,4

3. 拓展题：

(1)分别写出$f(x+h), f(x-h)$在x点的二阶泰勒展开式

(2) 根据1的结论，给出$f^{'}(x), f^{''}(x)$的数值计算格式，给出误差估计。

5.2 实现思路

第一题

对于方程组，首先化作$Ax=b$形式。

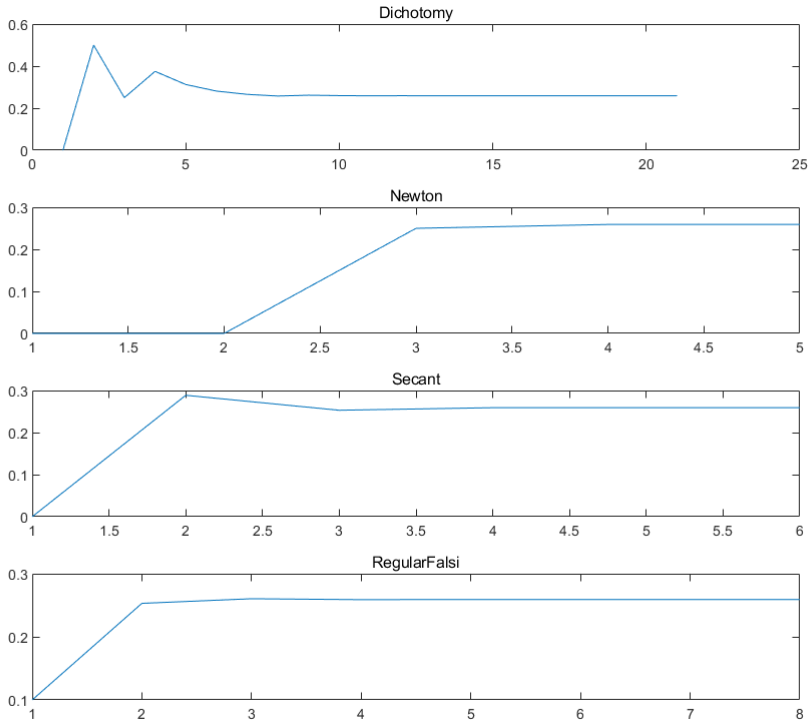

图 8: 实验结果

可得

$$\begin{pmatrix} 4 & -1 & 1 \\ 4 & -8 & 1 \\ -2 & 1 & 5 \end{pmatrix} \begin{pmatrix} x1 \\ x2 \\ x3 \end{pmatrix} = \begin{pmatrix} 7 \\ -21 \\ 15 \end{pmatrix}$$

故有

$$A = \begin{pmatrix} 4 & -1 & 1 \\ 4 & -8 & 1 \\ -2 & 1 & 5 \end{pmatrix}, b = \begin{pmatrix} 7 \\ -21 \\ 15 \end{pmatrix}$$

随后我们对A进行LU分解，采用Doolittle算法，编程求解。

第二题

利用Jacobi,Gauss-Seidel方法求解该方程组。对于Jacobi迭代法，该方法反复迭代，可以讲结果逼近正确值。

5.3 实验结果与分析

实验一

LU分解结果

Jacobi 迭代法迭代结果:

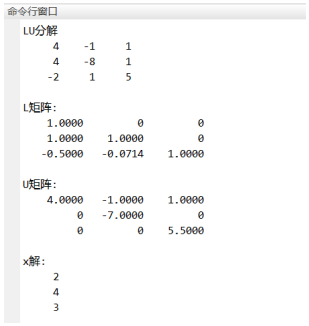

图 9: LU分解结果

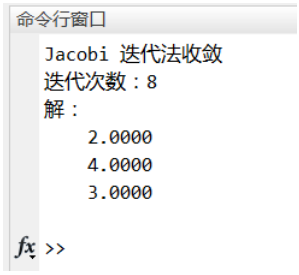

图 10: Jacobi 迭代法迭代结果

实验二

Jacobi、Gauss-Seidel迭代法解收敛情况:

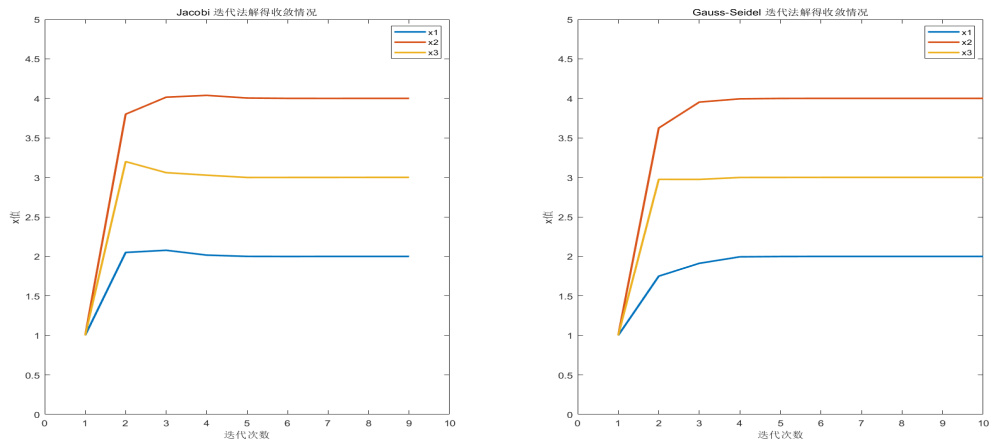

图 11: Jacobi与Gauss-Seidel收敛情况

6 实验四：第三章插值多项式

(1) 以$y = sin(x)$为例，在[0,π]区间内生成11个，21个数据点，涉及算法或程序，用上述4个边界条件，分别计算其样条插值，并作图比较，分析其差异性。

6.1 子实验要求

1. 基于不同边界条件的样条函数计算公式推导:

1.自然边界

2.固定边界

3.周期边界

4.强制第一个子区间和第二个子区间样条多项式的三阶导数相等，倒数第二个子区间和最后一个子区间的三次样条函数的三阶导数相等。

2.以$y = sin(x)$为例，在$[0, \pi]$区间内生成11个、21个数据点，设计算法或程序，用上述4个边界条件，分别计算其样条插值，并做图比较分析，分析器差异性

6.2 实现思路

边界条件推导:

自然边界: 其特殊点在于

$$S^{''}(x_0) = S^{''}(x_n)$$

第一个点与最后一个点的插值函数二阶导数相等且都为0.

固定边界：其特征条件固定边界为

$$S^{'}(x_0) = f^{'}(x_0)$$

和

$$S^{'}(x_n) = f^{'}(x_n)$$

周期边界:

$$S'(x_1 + 0) = S'(x_n - 0), \quad S''(x + 0) = S''(x_n - 0)$$

一、自然边界情况的证明

定义三次多项式

$$S_j(x) = a_j + b_j(x - x_j) + c_j(x - x_j)^2 + d_j(x - x_j)^3, \quad j = 0, 1, \cdots, n-1$$

由于

$$S_j(x_j) = a_j = f(x_j)$$

以$h = x_{j+1} - x_j, j = 0, 1, 2, 3...$且定义$a_n = f(x_n)$我们可以获取：

$$a_{j+1} = a_j + b_j h_j + c_j h_j^2 + d_j h_j^3$$

一次类推定义$b_n = S'(x_n)$获得

$$b_{j+1} = b_j + 2c_j h_j + 3d_j h_j^2, \quad j = 0, 1, \cdots, n-1$$

又有关系$c_n = \frac{S''(x_n)}{2}$，从而获得$c_{j+1} = c_j + 3d_j h_j$,可以解得:

$$a_{j+1} = a_j + b_j h_j + \frac{h_j^2}{3}(2c_j + c_{j+1})$$

和

$$b_{j+1} = b_j + h_j(c_j + c_{j+1})$$

随后可以解出b_j为：

$$b_j = \frac{1}{h_j}(a_{j+1} - a_j) - \frac{h_j}{3}(2c_j + c_{j+1})$$

然后下标减1得到b_{j-1},即

$$b_{j-1} = \frac{1}{h_{j-1}}(a_j - a_{j-1}) - \frac{h_{j-1}}{3}(2c_{j-1} + c_j)$$

将该公式带入上（7）式中，得到自然样条插值条件下得线性方程组:

$$h_{j-1}c_{j-1} + 2(h_{j-1} + h_j)c_j + h_j c_{j+1} = \frac{3}{h_j}(a_{j+1} - a_j) - \frac{3}{h_{j-1}}(a_j - a_{j-1})$$

二、固定边界情况与自然边界类似，引用书本证明

证明：因为$f'(a) = S'(a) = S'(x_0) = b_0$,所以取j=0可得

$$f(a) = \frac{1}{h_0}(a_1 - a_0) - \frac{h_0}{3}(2c_0 + c_1)$$

从而

$$2h_0c_0 + h_0c_1 = \frac{3}{h_0}(a_1 - a_0) - 3f'(a)$$

类似地，有

$$f'(b) = b_n = b_{n-1} + h_{n-1}(c_{n-1} + c_n)$$

取j=n-1可得

$$\begin{aligned} f'(b) &= \frac{a_n - a_{n-1}}{h_{n-1}} - \frac{h_{n-1}}{3}(2c_{n-1} + c_n) + h_{n-1}(c_{n-1} + c_n) \\ &= \frac{a_n - a_{n-1}}{h_{n-1}} + \frac{h_{n-1}}{3}(c_{n-1} + 2c_n) \end{aligned}$$

和

$$h_{n-1}c_{n-1} + 2h_{n-1}c_n = 3f'(b) - \frac{3}{h_{n-1}}(a_n - a_{n-1})$$

以及方程

$$2h_0c_0 + h_0c_1 = \frac{3}{h_0}(a_1 - a_0) - 3f'(a)$$

和

$$h_{n-1}c_{n-1} + 2h_{n-1}c_n = 3f'(b) - \frac{3}{h_{n-1}}(a_n - a_{n-1})$$

决定了线性方程组AX=b。

三、周期边界情况可有下列方程组推导出

$$\begin{cases} s_3(x_0+0) = s_3(x_n-0) \\ s_3'(x_0+0) = s_3'(x_n-0) \\ s_3''(x_0+0) = s_3''(x_n-0) \end{cases}$$

四、非扭结样条边界推导

$$S_0'''(x_0) = S_1'''(x_1)$$
$$S_{n-2}'''(x_{n-2}) = S_{n-1}'''(x_{n-1})$$

由于

$$S_i'''(x) = 6d_i$$

并且，

$$d_i = \frac{m_{i+1} - m_i}{6h_i}$$
$$d_0 = d_1 \quad d_{n-2} = d_{n-1}$$

即

$$h_1(m_1 - m_0) = h_0(m_2 - m_1)$$
$$h_{n-1}(m_{n-1} - m_{n-2}) = h_{n-2}(m_n - m_{n-1})$$

新的矩阵可以写为

$$\begin{bmatrix} -h_1 & h_0+h_1 & -h_0 & \cdots & \cdots & 0 \\ h_0 & 2(h_0+h_1) & h_1 & 0 & & \vdots \\ 0 & h_1 & 2(h_1+h_2) & h_2 & 0 & \vdots \\ \vdots & 0 & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & h_{n-2} & 2(h_{n-2}+h_{n-1}) & h_{n-1} \\ 0 & \cdots & \cdots & -h_{n-1} & h_{n-2}+h_{n-1} & \cdots h_n \end{bmatrix}$$

五、上机函数设计

有了上题的四种边界条件样条插值的分析，我们可以依照这种方式设计对应的四种插值函数。由题意给出的函数为$f = sin(x)$，我们在区间$[0, \pi]$随机生成11个点、21个点，作为初始值。然后带入对应的边界条件插值函数方程组，绘制图像，分析比较各种情况的差异性。

6.3 实验结果与分析

初始点图像的分布

四种情况的插值函数图像

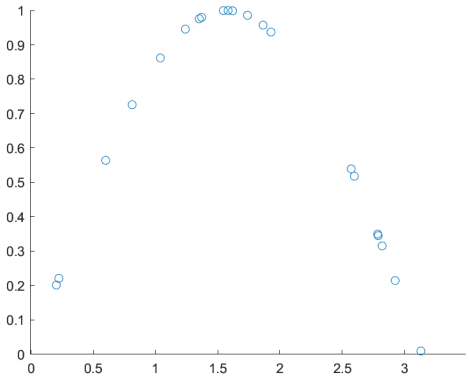

图 12: 初始点的分布

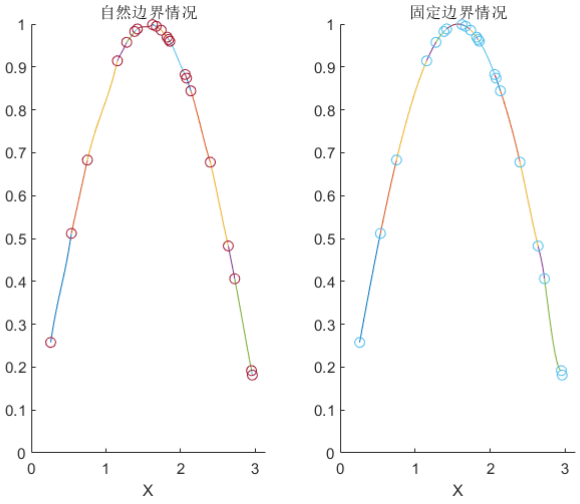

图 13: 自然边界与固定边界

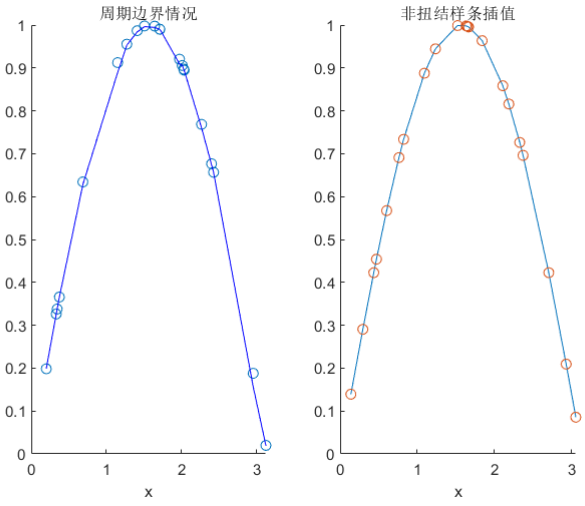

图 14: 周期边界与非扭结边界

7 实验五：第四章数值微分与数值积分

7.1 子实验要求

7.1.1 作业题

1.推导复合(Compsite)梯形公式及其误差估计；推导基于误差控制的逐次半积分梯形及其误差估计。

2. let $h=(b-1)/3, x_0=a, x_1=a+h, x_2=b$，Find the Degree oF precision of the quadrature formula.

$$\int_a^b f(x)dx = \frac{9}{4}hf(x_1)+\frac{3}{4}hf(x_2)$$

7.1.2 上机要求

1.自行编制复合梯形公式、Simpson公式的计算程序

2.取h=0.01，分别利用复合梯形、Simpson公式计算定积分

$$I(f)=\frac{1}{\sqrt{2\pi}}\int_0^1 exp^{-\frac{x^2}{2}}dx$$

试与精确解比较，说明两种格式的优劣

3.若取计算精度为10^{-4},则h=?,n=?

7.2 实现思路

7.2.1 作业结果

1.推导、复合(Compsite)梯形公式及其误差估计；推导基于误差控制的逐次半积分梯形及其误差估计

我们假设**复合梯形**公式将分为M个区间。对于每一个字区间一个子区间$[x_{k-1},x_k]$，我们求和，相当于函数的求积分:

$$\int_a^b f(x)dx=\sum_{k-1}^{M}\int_{x_{k-1}}^{x_k} f(x)dx \approx \sum_{k=1}^{M}\frac{h}{2}(f(x_k-1)+f(x_k))$$

由于$\frac{h}{2}$为常数，我们可以获得复合梯形积分的公式

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

或者

$$T(f,h)=\frac{h}{2}f_0+2f_1+2f_2+...+2f_{M-1}+f_M$$

或者

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

对于由Lagrange多项式推导复合梯形公式的误差误估计有:

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

2.let $h=(b-1)/3, x_0=a, x_1=a+h, x_2=b$***，Find the Degree oF precision of the quadrature formula.***

$$\int_a^b f(x)dx=\frac{9}{4}hf(x_1)+\frac{3}{4}hf(x_2)$$

解:

$$\int_a^b f(x)dx=\frac{9}{4}hf(a+h)+\frac{3}{4}hf(a+3h)$$

$$\int_a^{a+3h} dx=[x]_a^{a+3h}=3h$$

$$\frac{9}{4}h+\frac{3}{4}h=3h$$

$$\int_a^{a+3h} xdx=3ah+\frac{9}{2}h^2$$

$$\frac{9}{4}h(a+h)+\frac{3}{4}h(a+3h)=3ah+\frac{9}{2}h^2$$

$$\int_a^{a+3h} x^2dx=3a^2h+9ah^2+9h^3$$

$$\frac{9}{4}h(a+h)^2+\frac{3}{4}h(a+3h)^2=3a^2h+9ah^2+9h^3$$

$$\int_a^{a+3h} x^3dx=3a^3h+\frac{27}{2}a^2h^2+27ah^3+\frac{81}{4}h^4$$

$$\frac{9}{4}h(a+h)^3+\frac{3}{4}h(a+3h)^3=3a^3h+\frac{27}{2}a^2h^2+27ah^3+\frac{45}{2}h^4$$

综上所述，degree为2

7.2.2 上机实验

1.自行编制复合梯形公式、Simpson公式的计算程序

我们利用复合梯形公司以及Simpson公式可以编写相关的matlab代码。给定函数，以及相关区间。我们将固定M个相同间隔的点以及对应的函数值作为初始，结果返回积分结果。

2.取h=0.01，分别利用复合梯形、Simpson公式计算定积分

$$I(f)=\frac{1}{\sqrt{2\pi}}\int_0^1 exp^{-\frac{x^2}{2}}dx$$

试与精确解比较，说明两种格式的优劣有了第一题的复合梯形公式以及Simpson公式，我们可以确定区间为[0,1]，步长h=0.01，获得相应的点以及函数值，利用复合梯形公式以及Simpson公式，求解最后的结果。最后我们利用matlab内置的函数求解准确的积分值，与我们的获得结果进行比较。

3.若取计算精度为10^{-4}***,则h=?,n=?***

7.3 实验结果与分析

第一题

1. 复合梯形运行结果：

```
>> help traprl
  复合梯形求积公式
  function s=traprl(f,a,b,M)
  Input - f 是被积函数
        - a 和 b是积分的上下界限
        - M 为子区间的个数

>> s2 = traprl(f,1,2,10)

s2 =

   0.693771403175428

>>
```

图 15: 复合梯形运行结果

2.Simpson运行结果：

```
>> help simprl
  Simpson求积公式
  function s=simprl(f,a,b,M)
  Input - f 是被积函数
        - a 和 b是积分的上下界限
        - M 为子区间的个数

>> f = @(x)1/x;
>> s1 = simprl(f,1,2,10)

s1 =

   0.693147374665116
```

图 16: Simpson运行结果

第二题

我们分别输出int(matlab内置积分函数，复合梯度积分，Simpson公式积分，输出保留12位的小数。

```
命令行窗口
  标准解积分结果:0.341344746069

  复合梯型积分结果:0.341342729639
  复合梯度积分方法的误差绝对值:0.000002016429

  Simpson公式积分结果:0.341344746070
  Simpson公式方法的误差绝对值:0.000000000002

  Simpson公式方法的误差较小。
fx >>
```

图 17: 三种积分结果

结果分析可得Simpson公式法所求的积分值误差较小。

第三题

我们发现子区间数为15时，复合梯度办法求积能够满足精度小于10^{-4};

```
标准解积分结果:0.341344746069

子区间数为:15

复合梯型积分结果:0.341255114001
复合梯度积分方法的误差绝对值:0.000089632068

Simpson公式积分结果:0.341344749389
Simpson公式方法的误差绝对值:0.000000003321

Simpson公式方法的误差较小。
```

图 18: 实验结果

我们发现子区间数为2时，Simpson公式办法求积能够满足精度小于10^{-4};

```
命令行窗口
  标准解积分结果:0.341344746069

  子区间数为:2

  复合梯型积分结果:0.336260914612
  复合梯度积分方法的误差绝对值:0.005083831456

  Simpson公式积分结果:0.341355487857
  Simpson公式方法的误差绝对值:0.000010741788

  Simpson公式方法的误差较小。
fx >>
```

图 19: 实验结果

8 实验六：第五章常微分方程数值解

8.1 子实验要求

1. 求$y^{'}=1+y^2, y(0)=0$的数值解（分别用欧拉显格式、梯形预估修正格式、4阶龙格库塔格式，并与解析解比较这三种格式的收敛性）

2. 用龙格库塔4阶方法求解描述振荡器经典的van der Pol微分方程

$$\begin{cases} \frac{\mathrm{d}^2y}{\mathrm{d}t^2}-\mu\left(1-y^2\right)\frac{\mathrm{d}y}{\mathrm{d}t}+y=0 \\ y(0)=1, y'(0)=0 \end{cases}$$

分别取μ=0.01，0.1,1，作图比较计算结果。

8.2 实现思路

第一题：

为了比较误差，我们需要一组标准解，我们利用matlab内置**ODE45**函数进行求解，之后与三种方法的结果进行做差比较获得误差结果。

首先，对于欧拉显格式，我们可以根据方程

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

编写相关matlab代码，为了与ODE45结果相减获得误差，我们进行56次迭代。

对于**梯形修正预估格式**.由积分中值定理我们有

$$\exists\xi\in[a,b]\int_a^b f(x)dx=(b-a)f(\xi)$$

但由于ξ确定，故难以准确算出f(ξ)的值。故在此基础上，用两端点$f(a)f(b)$的算术平均值来估算f(ξ),即

$$\int_a^b f(x)dx\approx\frac{b-a}{2}[f(a)+f(b)]$$

此为我们所用梯形公式。我们对于我们每一个子区间的积分都用梯形公式，就形成了复合梯形公式：

$$\begin{aligned}\int_a^b f(x)dx&\approx\frac{b-a}{2N}\sum_{n=1}^{N}\left(f\left(x_n\right)+f\left(x_{n+1}\right)\right)\\&=\frac{b-a}{2N}\left[f\left(x_1\right)+2f\left(x_2\right)+\ldots+2f\left(x_N\right)+f\left(x_{N+1}\right)\right]\end{aligned}$$

梯形公式示意图:

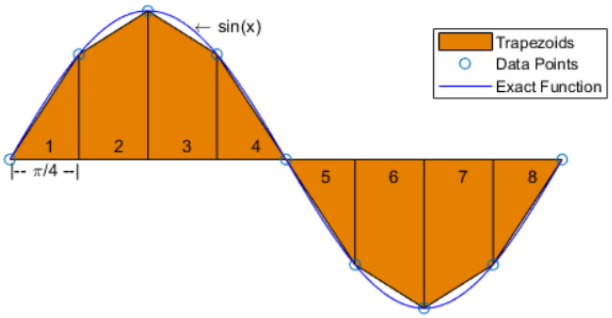

图 20: 复合梯形积分示意图

最后对于四阶的**龙格库塔方法** [薛定宇(2018)]，我们根据龙格库塔通式：

$$= \begin{cases} y_{i+1} = y_i + c_1K_1 + c_2K_2 + \cdots + c_pK_p \\ K_1 = hf\left(x_i, y_i\right) \\ K_2 = hf\left(x_i + a_2h, y_i + b_{21}K_1\right) \\ \cdots\cdots\cdots\cdots \\ K_p = hf\left(x_i + a_ph, y_i + b_{p1}K_1 + \cdots + b_{p,p-1}K_{p-1}\right) \end{cases}$$

我们将阶数p确定为4，则公式为：

$$\begin{cases} y_{n+1} = y_n + \frac{h}{6}\left(K_1 + 2K_2 + 2K \quad {}_3K\right)_4 \\ K_1 = f\left(x_n, y_n\right) \\ K_2 = f\left(x_n + \frac{h}{2}, y_n + \frac{h}{2}K_1\right) \\ K_3 = f\left(x_n + \frac{h}{2}, y_n + \frac{h}{2}K_2\right) \\ K_4 = f\left(x_n + h, y_n + hK_3\right) \end{cases}$$

根据该公式我们可以编写相关的matlab代码来求解题目的微分方程.

第二题

对于第二题利用四阶龙格库塔解决van der Pol振荡器微分方程:

$$\begin{cases} \frac{\mathrm{d}^2y}{\mathrm{d}t^2} - \mu\left(1 - y^2\right)\frac{\mathrm{d}y}{\mathrm{d}t} + y = 0 \\ y(0) = 1, y'(0) = 0 \end{cases}$$

我们可以根据所给方程的第一条，写出该方程的matlab函数。由给定初值$y(0) = 1, y^{'}(0) = 0$定义初始条件。我们设定求解的区间为$[0, 30]$，子区间为100，代入我们第一题编写Runge-Kutta方程中，获得最后的解。该解一共由三个列向量构成$Y = [t, y1, y2]$,其中有t为步长，$y_1 = y, y_2 = \dot{y}$.

最后有我们获得步长以及一阶导数与函数值，绘制图像。我们再将μ分别设为$0.01, 0.1, 1$，然后绘制结果的图像。

8.3 实验结果与分析

第一题四种方法的积分

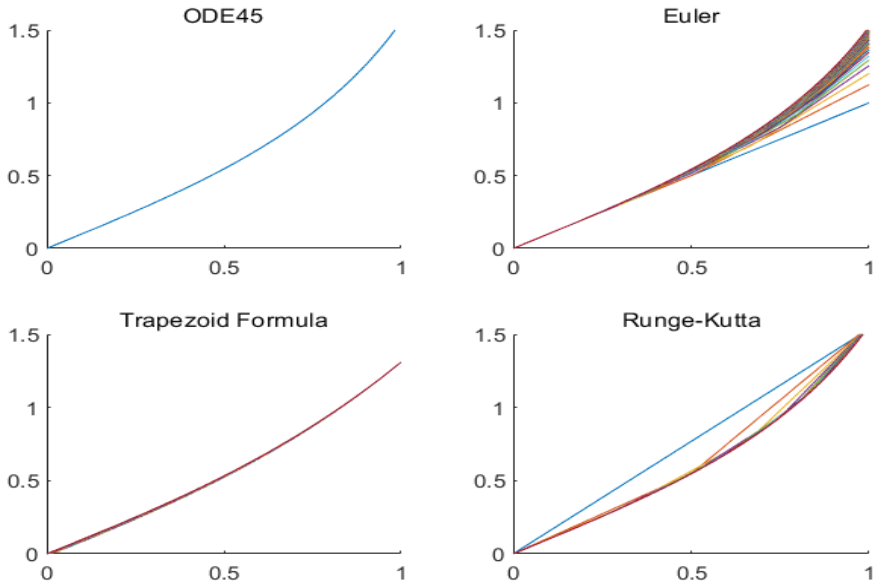

图 21: 四种方法的积分

第一题三种积分方法的误差

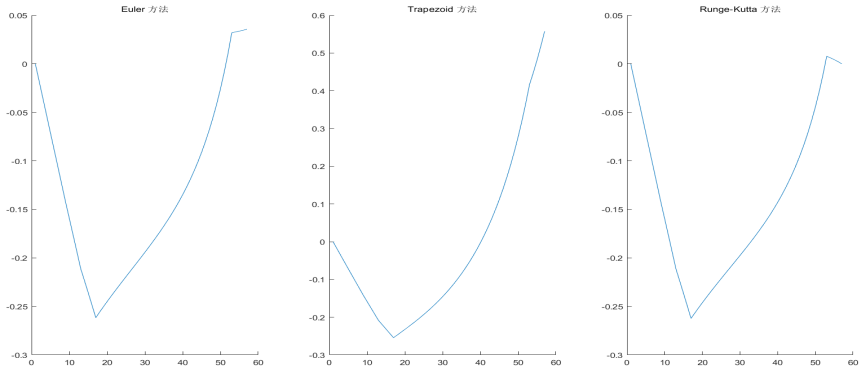

图 22: 三种积分方法的误差

第二题

van der pol 微分方程图像

$\mu = 1$ 的图像

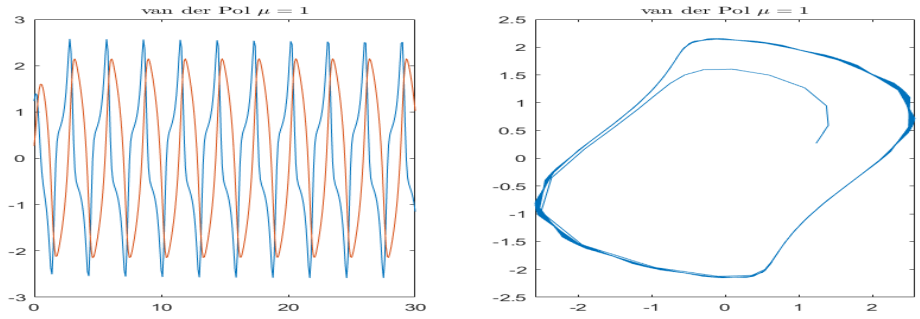

图 23: 三种积分方法的误差

$\mu = 0.1$ 的图像

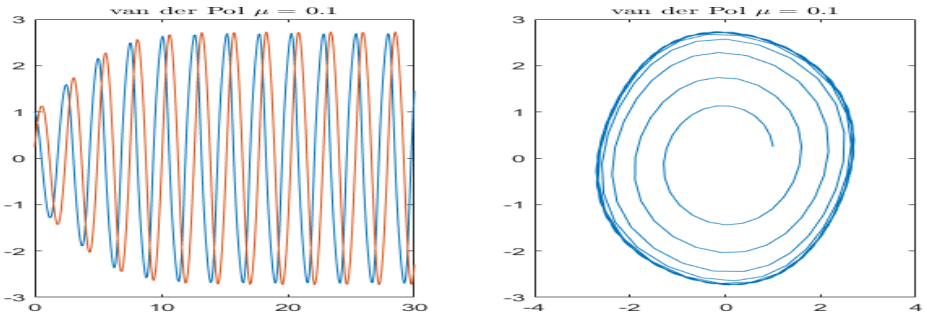

图 24: 三种积分方法的误差

$\mu = 0.01$ 的图像 .

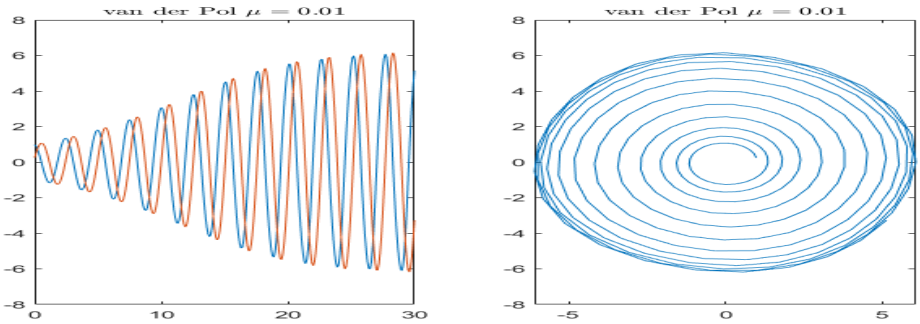

图 25: 三种积分方法的误差

9 实验七：第八章曲线拟合与函数逼近

9.1 子实验要求

已知观测数据

x	-2	-1	0	1	2
f(x)	0	1	2	1	0

求一个二次多项式拟合这组数据，试写出其最小二乘拟合模型，并给出正则方程组及其解。

9.2 实现思路

我们设$(x_k, y_k)_{k=1}^N$为N个初始点。对于最小二乘二次多项式方程的系数表示为:

$$y = f(x) = Ax^2 + Bx + C$$

其正则线性方程组为:

$$\begin{cases} \left(\sum_{k=1}^N x_k^4\right) A + \left(\sum_{k=1}^N x_k^3\right) B + \left(\sum_{k=1}^N x_k^2\right) C = \sum_{k=1}^N y_k x_k^2 \\ \left(\sum_{k=1}^N x_k^3\right) A + \left(\sum_{k=1}^N x_k^2\right) B + \left(\sum_{k=1}^N x_k\right) C = \sum_{k=1}^N y_k x_k \\ \left(\sum_{k=1}^N x_k^2\right) A + \left(\sum_{k=1}^N x_k\right) B + NC = \sum_{k=1}^N y_k \end{cases}$$

为求解正则方程组，我们需要计算各个观测数据对对应的 $x_k, y_k, x_k^2, y_k^2, x_k^3, y_k^3, x_k^4, x_k y_k, x_k^2 y_k$ 计算结果如图表所示:

x_k	y_k	x_k^2	x_k^3	x_k^4	$x_k y_k$	$x_k^2 y_k^*$
-2	0	4	-8	16	0	0
-1	1	1	-1	1	-1	1
0	2	0	0	0	0	0
1	1	1	1	1	1	1
2	0	4	8	16	0	0
0	4	10	0	34	0	2

利用获取的数据，我们带入正则方程组，求解A、B、C系数。

$$\begin{cases} \mathrm{A} = -0.4286 \\ \mathrm{B} = 0 \\ \mathrm{C} = 1.6571 \end{cases}$$

故求得最小二乘拟合方程为:$y = -0.4268x^2 + 1.6571$

9.3 实验结果与分析

图像拟合结果

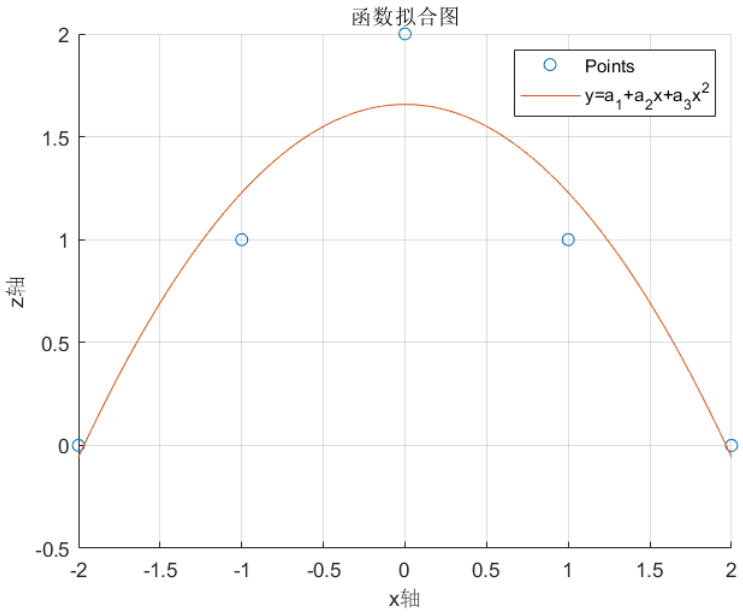

图 26: 图像拟合情况

正则方程左系数与右结果:

```
正则方程左系数:
     5     0    10
     0    10     0
    10     0    34

正则方程右项结果:
     4
     0
     2
```

图 27: 正则方程左系数与右结果

10 实验八：第九章特征值与特征方程

10.1 子实验要求

上机实验：

第一题:

已知矩阵

$$A=\begin{bmatrix} 4 & -1 & 1 \\ -1 & 3 & -2 \\ 1 & -2 & 3 \end{bmatrix}$$

是一个对称矩阵，且其特征值为$\lambda_1=6,\lambda_2=3,\lambda_3=1$。分别利用幂法、对称幂法、反幂法求其最大特征值和特征向量。

注意: 可取初始向量$x^0=(1 \quad 1 \quad 1)^T$.

第二题:

验证实验：写出PPT中关于Householder变换实例中的H1,H2,H3,并验证实验结果。

10.2 实现思路

第一题

对于**第一题**，我们可以根据**幂法**的定义与原理。一个向量乘以一个可对角化矩阵，该向量会向主特征值所对应的特征向量偏移。故我们如果将一个向量与该矩阵多次迭代乘法，我们最后获得的向量将会再特征向量方向上。之后我们便可以求解主特征值与对应的一个特征向量。我们可以根据一下的公式进行matlab代码的编写:

$$\vec{v}_0=\sum_{i=1}^{n}a_i\vec{x}_i\neq\overrightarrow{0},\quad(\alpha_1\neq0),\vec{v}_k=A\vec{v}_{k-1},(k=1,2,\cdots)$$

$$\text{(a)}\ \lim_{k\to\infty}\frac{\vec{v}_k}{\lambda_1^k}=\alpha_1\vec{x}_1$$

$$\text{(b)}\ \lim_{k\to\infty}\frac{(\vec{v}_{k+1})_i}{(\vec{v}_k)_i}=\lambda_1$$

对称幂法是针对于对称矩阵，在幂法的基础上可以更快速判断是否有0的特征值，我们可以根据书本$p502\sim p503$的伪码进行matlab代码的编写。

反幂法与幂法相反，适用于求模最小的特征值机器特征向量。这是因为对于A矩阵的逆A^{-1}其对应特征值为$\frac{1}{\lambda_n}$,所以多次乘A^{-1}迭代后，向量向$\frac{1}{\lambda_n}$最大的特征值的特征向量方向逼近，此时对应的特征值λ却是最小的。利用这种方法，我们可以求解出最小的特征是。其中求解逆向量以及方程组的求解，我们可以利用LU法。理论支持：

$$\text{(a)}\ \lim_{k\to\infty}u_k=\frac{x_n}{\max(x_n)}$$

$$\text{(b)}\ \lim_{k\to\infty}\mu_k=\frac{1}{\lambda_n}$$

其中收敛速度由比值$r=\left|\frac{\lambda_n}{\lambda_{n-1}}\right|$所确定。

第二题：

对于豪斯霍尔德(Householder)矩阵的计算有下列公式作为理论支撑:

首先豪斯霍尔德这一变换在数值线性代数上的意义。这一变换将一个向量变换为由一个超平面反射的镜像，是一种线性变换。

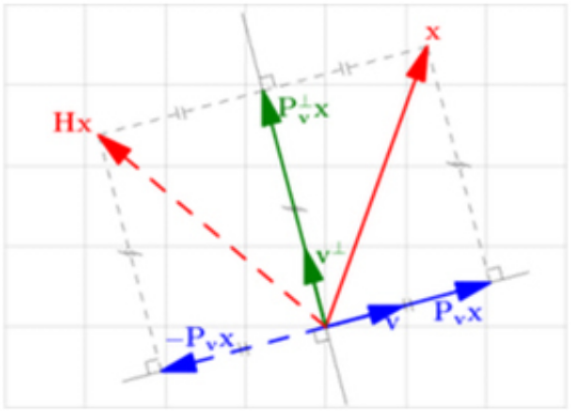

图 28: 豪斯霍尔德镜像变化

如果v为单位向量I为单位矩阵，则上述线性变化是豪斯霍尔德矩阵

$$\mathbf{H} = \mathbf{I} - \frac{2}{\langle \mathbf{v}, \mathbf{v} \rangle} \mathbf{v}\mathbf{v}^H$$

其中Housholder向量v需要满足

$$\mathbf{v} = \mathbf{x} + \operatorname{sgn}(\mathbf{x}_1) \|\mathbf{x}\|_2 \mathbf{e}_1$$

其可用于QR分解，利用该操作我们可以对$m * n$的矩阵A进行QR分解。矩阵Q可以被用于对一个向量以一种特定的方式进行反射变换，使得它除了一个维度以外的其他所有分量都化为0。我们在一次迭代的基础上，再次获得Householder矩阵再次迭代，这样就将原矩阵变成上三角矩阵$R = Q_t \cdots Q_2 Q_1 A$。根据这种思想，我们可以编写matlab代码进行分析求解。

10.3 实验结果与分析

第一题：

幂法求主特征值与特征向量

```
幂法求特征值:
特征值互异，是对角化矩阵
第1.000000次迭代
主特征值=4.00000000
主特征向量v=
     4
     0
     2
```

图 29: 幂法主特征值与特征向量初值

34次迭代最终结果

```
第34.000000次迭代
主特征值=6.00000000
主特征向量v=
   5.999999999301508
  -5.999999998253770
   5.999999998253770

****************************
主特征值为:
lambda1 =

   5.999999999301508

特征向量为:

beta1 =

   5.999999999301508
  -5.999999998253770
   5.999999998253770
```

图 30: 幂法34次迭代结果

对称幂法求主特征值与特征向量.

```
对称幂法求特征值：
特征值互异，是对角化矩阵
第1.000000次迭代
最大特征值为：2.000000e+00
特征向量为：
x0 =

    0.894427190999916
                    0
    0.447213595499958

第2.000000次迭代
最大特征值为：4.600000e+00
特征向量为：
x0 =

    0.814821714382667
   -0.362142984170074
    0.452678730212593
```

图 31: 对称幂法主特征值与特征向量初值

31次迭代最终结果

```
第31.000000次迭代
最大特征值为：6
特征向量为：
x0 =

    0.577350269727325
   -0.577350268920776
    0.577350268920776

*****************************
主特征值为：
lambda2 =

      6

特征向量为：

beta2 =

    0.577350269727325
   -0.577350268920776
    0.577350268920776
```

图 32: 对称幂法31次迭代结果

反幂法求最小主特征值与特征向量.

```
反幂法求特征值:
特征值互异，是对角化矩阵
第1.000000次迭代
主特征值=0.99656357
主特征向量v=
   0.367132867132867
   5.087412587412583
   4.912587412587408
```

图 33: 反幂法主特征值与特征向量初值

10次迭代最终结果

```
第10.000000次迭代
主特征值=1.00000000
主特征向量v=
   0.000000000128526
   4.999999999357719
   4.999999999229228

*****************************
主特征值为:
lambda3 =

   1.000000000025691

特征向量为:

beta3 =

   0.000000000128526
   4.999999999357719
   4.999999999229228
```

图 34: 反幂法10次迭代结果

第二题：三个Householder矩阵结果

```
第1次Houserholder矩阵为:
Hi =

    0.4472    0.4472    0.4472    0.4472    0.4472
    0.4472    0.6382   -0.3618   -0.3618   -0.3618
    0.4472   -0.3618    0.6382   -0.3618   -0.3618
    0.4472   -0.3618   -0.3618    0.6382   -0.3618
    0.4472   -0.3618   -0.3618   -0.3618    0.6382

第2次Houserholder矩阵为:
Hi =

   -0.8279   -0.5117   -0.1954    0.1208
   -0.5117    0.8568   -0.0547    0.0338
   -0.1954   -0.0547    0.9791    0.0129
    0.1208    0.0338    0.0129    0.9920

第3次Houserholder矩阵为:
Hi =

   -0.1483    0.1475    0.9779
    0.1475    0.9810   -0.1256
    0.9779   -0.1256    0.1673
```

图 35: 三次Householder矩阵

11 结论与心得

通过本次实验，我们进一步加深对数值计算知识的了解，通过作业与实验，我们呢能够亲自动手，将书本上的知识进行实现、应用。

本次实验，我们一共做了八个实验:误差理论，非线性方程组求解，线性方程组的求解、插值多项式、数值微分与数值积分、常微分方程数值解、曲线拟合与函数逼近、特征值与特征方程这八章的内容。有些实验比如插值多项式、数值微分与数值积分、常微分方程求解、特征值与特征方程，在理解书本内容的基础之上，我们同时也得去网上搜索相关的资料来补充自己的理解。比如四阶龙格库塔来求数值微分与数值积分，有一定的难度，我们不能只在这个知识点上进行思考，我们同时也得明白这个算法的目的以及由来，然后利用书本的伪代码来解决具体的问题。

这次数值计算实验我采用了latex语言进行实验报告的编写，用matlab取实现算法，并用matlab进行绘图分析。当我们真的用代码去实现我们的内容的时候，我们会遇到书本上想不到的问题。不仅仅是知识点的吸收好坏，也是对个人编程能力的考验。在矩阵的运算上面，常常会遇到矩阵维度不一致而中途停止。我常常要花费大量的时间去寻找问题所在。最后还是得养成先写出算法的思路，确定算法中矩阵运算的维度是一致，然后再去着手编码，这样才能减少算法层面上的问题，然后可以把问题聚焦在编程的层面。这个是非常重要的一点，可以帮我们提高自己的debug能力.

最后，通过本次数值计算实验，我独立自主实践完成了实验的要求，用matlab实现书本上知识，解决实验给出的问题，加深了我们对于数值计算的理论与应用的理解。

参考文献

[Barnes(1965)] J. G. P. Barnes. An Algorithm for Solving Non-Linear Equations Based on the Secant Method. *The Computer Journal*, 8(1):66–72, 04 1965. ISSN 0010-4620. doi: 10.1093/comjnl/8.1.66. URL `https://doi.org/10.1093/comjnl/8.1.66`.

[John.H.Mathews(2019)] Kurtis D Fink John.H.Mathews. *数值分析(MATLAB版)*. 电子工业出版社, 三河市华成印务有限公司, 2019. ISBN 978-7-121-33920-2.

[Richard.L.Burden(2005)] 冯烟丽 Richard.L.Burden, J.Douglas.Faires. *数值分析(第七版)*. 高等教育出版社, 北京市西城区德外大街, 2005. ISBN 7-04-017146-5.

[薛定宇(2018)] 陈阳泉薛定宇. *高等应用数学问题的MATLAB求解(第三版)*, pages 242–245. 清华大学出版社, 清华大学学研大厦A座, 2018. ISBN 978-7-302-33263-3. URL `http://www.tup.com.cn`.

A 源代码

A.1 problem1

```
%% 第一题：problem1
    clc,clear;
    format long;
    tol=0.1;
    a=1;
    b=[-1000.001,-1000.0001,-10000.00001,-100000.00001];
    c=1;
    for i=1:4
    [m1,m2]=solve_quadratic_equation(a,b(i),c,tol);
    M = [m1;m2];
    A(:,i)=M;
    end
    fprintf四个方程的解如下(":\n");
    fprintf方程一的解("x1=%f, x2=%f\n",A(1,1),A(2,1));
    fprintf方程二的解("x1=%f, x2=%f\n",A(1,2),A(2,2));
    fprintf方程三的解("x1=%f, x2=%f\n",A(1,3),A(2,3));
    fprintf方程四的解("x1=%f, x2=%f\n",A(1,4),A(2,4));


function [x1,x2]=solve_quadratic_equation(a,b,c,err)
    % 用于求解二次方程
    % [x1,x2]=solve_quadratic_equation(a,b,c,err)
    % 函数需要输入a,b,c 以及允许的和derta|b的差|
    deta=(b^2-4*a*c)^(1/2);
        if abs(b)-deta<err          %如果差太小，则使用解的另一种形式，
           防止巨量误差
            x1=(deta-b)/2/a;
            x2=(-deta-b)/2/a;
        elseif(b>=0)                %否则，b时，如下得到>0，x1x2
            x1=-2*c/(b+deta);
            x2=-2*c/(b-deta);
        else                        %b时，如下得到<0，x1x2
            x2=-2*c/(b+deta);
            x1=-2*c/(b-deta);
        end
    end

% 实验一第二题
clc,clear;
```

```
r(1)=0.994;  %设定初始值
p(1)=1; p(2)=0.497;
q(1)=1;q(2)=0.497;
x(1)=1;
for i=2:16      %递推
    if i==2
        r(i)=r(i-1)/2;
        x(i)=x(i-1)/2;
    else
        x(i)=x(i-1)/2;
        r(i)=r(i-1)/2;
        p(i)=1.5*p(i-1)-p(i-2)/2;
        q(i)=2.5*q(i-1)-q(i-2);
    end
end

for i=1:15      %计算每一项误差
    dr(i)=x(i)-r(i);
    dp(i)=x(i)-p(i);
    dq(i)=x(i)-q(i);
end

m=1:15;     %作图

subplot(1,3,1);
hold on;
scatter(m,dr)
plot(m,dr);
title('$$r_{n}=\frac{r_{n-1}}{2}$$','Interpreter','latex');
hold off;

subplot(1,3,2);
hold on;
plot(m,dp)
scatter(m,dp);
title('$$p_{n}=\frac{3}{2}p_{n-1}-\frac{1}{2}p_{n-2}$$','
    Interpreter','latex');
hold off;

subplot(1,3,3);
hold on;
scatter(m,dq);
plot(m,dq)
```

```
title('$$q_{n}=\frac{5}{2}q_{n-1}-p_{n-2}$$','Interpreter','latex');
hold off;
```

A.2 problem2

```
%% problem2 code
    % 三种不同的方式迭代求解，并观察解的稳定性与收敛性
    % 定义三种迭代方法
    clc, clear;
    syms x;
    x0=2;xf=2;xl=2;
    count1=0;
    count2=0;
    count3=0;

    X1 = [2];X2=[2];X3=[2];
    % count = 0;
    judge = inf;
    X = [];
    f1 = @(x)(15-x^2);
    f2 = @(x)(15/(2*x+1));
    f3 = @(x)(x-(2*x^2+x-15)/(4*x+1));
    F = {f1,f2,f3};

% 第一种迭代方法
    xf = f1(x0);
    judge = abs(xf-x0);
    while(judge>0.00001)
    if(count1>2)
    break
    end
    count1 = count1+1;
    X1 = [X1,xf];
    xl = xf;
    xf = f1(xf);
    judge = abs(xf-xl);
    end
    subplot(3,1,1);
    plot(X1);
    title('x_{k+1}=15-x_{k}^{2}')
    axis([0,100,-10000,100]);
```

```
37
38      x0=2;
39      xf = f2(x0);
40      judge = abs(xf-x0);
41      while(judge>0.00001)
42      if(count2>100)
43      break
44      end
45      count2 = count2+1;
46      X2 = [X2,xf];
47      xl = xf;
48      xf = f2(xf);
49      judge = abs(xf-xl);
50      end
51
52      subplot(3,1,2);
53      plot(X2);
54      h=title('$$x_{k+1}=\frac{15}{2x_{k}+1}$$');
55      set(h,'Interpreter','latex');
56
57      x0=2;
58      xf = f3(x0);
59      judge = abs(xf-x0);
60      while(judge>0.00001)
61      if(count3>100)
62      break
63      end
64      count3 = count3+1;
65      X3 = [X3,xf];
66      xl = xf;
67      xf = f3(xf);
68      judge = abs(xf-xl);
69      end
70
71      subplot(3,1,3);
72      plot(X3);
73      axis([10,5,0,5]);
74      h = title('$x_{k+1}=x_{k}-\frac{2x_{k}^{2}+x_{k}-15}{4x_{k
            }+1}$');
75      set(h,'Interpreter','latex');
76
77      fprintf("迭代次数fun1:%s\迭代次数nfun2:%d\迭代次
            数nfun3:%d\n","NAN",count2,...
```

```
78      count3);
79
80      %% 稳定性分析
81      %% problem1 code
82      % 三种不同的方式迭代求解，并观察解的稳定性与收敛性
83      % 定义三种迭代方法
84      clc, clear;
85      syms x;
86      x = [-5,0.1,5];
87      xf=0;
88      f2 = @(x)(15/(2*x+1));
89      f3 = @(x)(x-(2*x^2+x-15)/(4*x+1));
90
91  % 第二种方式
92      judge = inf;
93
94      for i=1:3
95      count=0;
96      x0 = x(i);
97      xl = f2(x0);
98      xr = f2(x0);
99      X = [x0];
100     judge = abs(xr-x0);
101     while(judge>0.00001)
102     if(count>100)
103     break
104     end
105     count = count+1;
106     X = [X,xr];
107     xl = xr;
108     xr = f2(xr);
109     judge = abs(xr-xl);
110     end
111     subplot(2,3,i);
112     plot(X);
113     end
114
115 % 第三种方式
116     judge = inf;
117     for i=1:3
118     count=0;
119     x0 = x(i);
120     xl = f3(x0);
```

```
121     xr = f3(x0);
122     X = [x0];
123     judge = abs(xr-x0);
124     while(judge>0.00001)
125     if(count>100)
126     break
127     end
128     count = count+1;
129     X = [X,xr];
130     xl = xr;
131     xr = f3(xr);
132     judge = abs(xr-xl);
133     end
134     subplot(2,3,i+3);
135     plot(X);
136     end
137
138 %% 实验一的第二题
139     clc,clear;
140     syms x;
141     f = @(x)(2-3*x-sin(x));
142     xl=0;
143     xr=1;
144     judge = inf;
145     xbefore = 0;
146     result = [1];
147     count = 0;
148     while (judge>0.0005)
149     count = count+1;
150     xresult =  (xl+xr)/2;
151     result = [result,xresult];
152     judge = abs(xresult-xbefore);
153     xbefore = xresult;
154     if(f(xresult)>0)
155     xl = xresult;
156     else
157     xr = xresult;
158     end
159     end
160
161     plot(result);
162     h = title('$$2-3x-sin(x)$$');
163     set(h,'Interpreter','latex');
```

```
164     xlabel('迭代次数');
165     ylabel('result');
166     fprintf函数迭代的次数为(":%d\n",count);
167
168
169     % 最终发现pi/2,5*结果收敛于pi1.8955
170     % 而10*收敛于pi-1.8955
171     clc,clear;
172     syms x;
173     tor = 0.000001;
174     f = 1/2 + x^2/4 - x*sin(x) - cos(2*x)/2;
175     x0 = [pi/4,pi/2,5*pi,10*pi];
176     str = ['pi/4','pi/2','5pi','10pi'];
177     latex_str = {'$$\frac{\pi}{4}$$','$\frac{\pi}{2}$','$5\pi$',
            '$10\pi$'};
178     disp("Newton 迭代法: ")
179     for i =1:size(x0,2)
180     Values = [];
181     [nitr, value, Values] = Newton(x0(i),f,tor);
182     display(str(i)迭代结果：+"");
183     display迭代次数(": "+num2str(nitr));
184     display零点("值x: "+num2str(value));
185     fprintf("\n");
186     subplot(4,1,i);
187     plot(Values);
188     h = title(latex_str{i});
189     set(h,"Interpreter","latex");
190     xlabel迭代次数("");
191     ylabel("xvalue");
192     end
193
194 %% 牛顿法求解
195     function [niter, xvalue,Values] = Newton(x0, f, tor)
196     % 该函数利用法对一元方程进行求解Newton
197     % function [nitr, value] = Newton(x0, f, tor)
198     % niter: 迭代次数为最终解    xvalue
199     tmp = x0;
200     xvalue = x0;
201     Values = [xvalue];
202     niter = 0;
203     dif = inf;
204     % f = matlabFunction(f);
205     df = diff(f);
```

```
206     df = matlabFunction(df);
207     f = matlabFunction(f);
208     while(dif > tor)
209     if(niter > 10000)
210     break;
211     end
212     niter = niter+1;
213     xvalue = tmp;
214     Values = [Values,xvalue];
215     tmp = xvalue - f(xvalue)/df(xvalue);
216     dif = abs(tmp-xvalue);
217     end
218     end
219
220 % 二分法求解
221     clc,clear
222     X = [];
223     syms x;
224     tor = 0.000001;
225     xr = 1;
226     xl = 0;
227     f = 5*x - exp(x);
228     [nitr, value, X] = Dichotomy(xl,xr,f,tor);
229     subplot(3,1,1);
230     plot(X);
231     title('Dichotomy')
232     disp("求解零点：Dichotomy");
233     display迭代次数(": "+num2str(nitr));
234     display零点("值x: "+num2str(value));
235
236 % 牛顿法求解
237     clc,clear;
238     syms x;
239     X = [];
240     tor = 0.00001;
241     f = 5*x - exp(x);
242     [nitr, value, X] = Newton(0,f,tor);
243     subplot(3,1,2);
244     plot(X);
245     title('Newton')
246     disp("Newton 求解零点");
247     display迭代次数(": "+num2str(nitr));
248     display零点("值x: "+num2str(value));
```

```
249
250 % 割线法
251     clc, clear;
252     syms x;
253     xr = 1;
254     xl = 0.1;
255     tor = 0.00001;
256     X = [];
257     f = 5*x - exp(x);
258     [niter, xvalue, X] = Secant(xl,xr, f, tor);
259     subplot(3,1,3);
260     plot(X);
261     title('Secant')
262     disp("Secant 求解零点");
263     display迭代次数(": "+num2str(niter));
264     display零点("值x: "+num2str(xvalue));
```

A.3 problem3

```
1     %% 实验三第一题
2     % 对方程组进行分解LU
3     clc,clear;
4     syms L U x;
5     % 为分解矩阵，AL,分别为分解后的矩阵U
6     A = [4,-1,1;4,-8,1;-2,1,5];
7     b = [7;-21;15];
8     [L,U,x]=LUresolve(A,b);
9     fprintf("分解LU\n");disp(A);
10    fprintf("矩阵L:\n");disp(L);
11    fprintf("矩阵U:\n");disp(U);
12    fprintf("解x:\n");disp(x);
13
14    %% 实验三的第二题
15    % 利用，JacobiGauss–方法求解方程组Seidel
16    % 迭代法求解Jacobi
17    clc,clear;
18    syms L U x1 X;
19    % 为分解矩阵A
20    A = [4,-1,1;4,-8,1;-2,1,5];
21    b = [7;-21;15];
22    k = 10; % 迭代次数
23    [x1] = Jacobi(A,b,k,0.0001);
```

```
24
25 %% 分解法求解LU
26     function [L,U,x] = LUresolve(A,b)
27     % 该函数进行分解LU
28     % function [L,U,x] = LUresolve(A,b)
29     n=length(A);
30     A(2:n,1)=A(2:n,1)/A(1,1);
31     for t=2:n-1  %进行分解LU
32     A(t,t:n)=A(t,t:n)-A(t,1:t-1)*A(1:t-1,t:n);
33     A(t+1:n,t)=(A(t+1:n,t)-A(t+1:n,1:t-1)*A(1:t-1,t))/A(t,t);
34     end
35     A(n,n)=A(n,n)-A(n,1:n-1)*A(1:n-1,n);
36     L=tril(A,-1)+eye(n);
37     U=triu(A); %矩阵分解出和ALU
38     for t=2:n    %解Lx=b
39     b(t)=b(t)-L(t,1:t-1)*b(1:t-1);
40     end
41     b(n)=b(n)/U(n,n);    %解Ux=b
42     for t=1:n-1
43     k=n-t;b(k)=(b(k)-U(k,k+1:n)*b(k+1:n))/U(k,k);
44     end
45     x=b;  %方程Ax=的解即为b
46     end
47
48 %% Jacobi 迭代求解法
49     function [x] = Jacobi(A,b,k,tol)
50     % function [x] = Jacobi(A,b,k)
51     % 矩阵,, 以及迭代次数Abk
52     judge = 0;
53     col = size(A,1);
54     x0 = ones(col,1);
55     X = [x0];
56     L = -tril(A,-1);U=triu(A,1);D=A+L+U;
57     T = D\(L+U);
58     c = D\b;
59     while(judge <= k)
60     x = T*x0+c;
61     if(sum(abs(x-x0))<tol)
62     fprintf('Jacobi 迭代法收敛\n');
63     fprintf迭代次数: ('%d\n',judge);
64     fprintf解: ('\n')
65     disp(x);
66     n = size(X,2);
```

```
67      i = 1:n;
68      plot(i,X,'LineWidth',2);
69      title("Jacobi 迭代法解得收敛情况");
70      xlabel迭代次数("");
71      ylabel("值x");
72      legend('x1','x2','x3');
73      return;
74      end
75      judge = judge+1;
76      x0 = x;
77      X = [X,x];
78      end
79      fprintf("Jacobi 迭代法不收敛\n");
80      n = size(X,2);
81      i = 1:n;
82      plot(i,X);
83      end
84
85  %% 方法迭代法求解Gauss_Siedel
86      function X = Gauss_Siedel( A ,B )
87      % function x = gauss_siedel( A ,B )
88      % check if the entered matrix is a square matrix
89      [na , ma ] = size (A);
90      if na ~= ma
91      disp('ERROR: Matrix A must be a square matrix')
92      return
93      end
94
95      % check if B is a column matrix
96      [nb , mb ] = size (B);
97      if nb ~= na || mb~=1
98      disp( 'ERROR: Matrix B must be a column matrix')
99      return
100     end
101
102     D = diag(diag(A));
103     L = tril(A)- D;
104     U = triu(A)- D;
105     [ma,na] = size(A);
106     X0 = ones(ma,1);
107
108
109     tol = 0.000001*ones(na,1);
```

```
110     k= 1;
111
112     X( : , 1 ) = X0;
113     err= 1000000000*rand(na,1);% initial error assumption for
            looping
114     while sum(abs(err) >= tol) ~= zeros(na,1)
115     X ( : ,k+ 1 ) = -inv(D+L)*(U)*X( : ,k) + inv(D+L)*B;% Gauss-
            Seidel formula
116     err = X( :,k+1) - X( :, k);% finding error
117     k = k + 1;
118     end
119
120     fprintf ('Gaussi-的迭代次数为Siedel%g\n', k);
121     n = size(X,2);
122     i = 1:n;
123     plot(i,X,'LineWidth',2);
124     title("Gaussi-Siedel 迭代法解得收敛情况");
125     xlabel迭代次数("");
126     ylabel("值x");
127     legend('x1','x2','x3');
```

A.4 problem4

```
1  %% 第三章样条插值
2  % 参考网站：https://www.cnblogs.com/lingr7/p/9780429.html
3  % 在[0,pi生成随机的点]
4  clc,clear;
5  x = pi*rand(11,1);
6  x = x';
7  f = @(x)sin(x);
8  y = f(x);
9
10 %开始构造函数
11 n=size(x,2);
12 for k=2:n %计算h(i)
13     h(k-1)=x(k)-x(k-1);
14 end
15 for k=1:(n-2) %计算μ和λ
16     mu(k)=h(k)/(h(k)+h(k+1));
17     lambda(k)=1-mu(k);
18 end
19 for k=1:(n-2)
```

```
20      g(k)=3*(lambda(k)*(y(k+1)-y(k))/h(k)+mu(k)*(y(k+2)-y(k+1))/h
            (k+1)); %计
            算g(1)到g(n-2)
21  end
22  fprintf('边界条件类型选择:\n1已知.f(a)和f(b)的二阶导数\n2已
        知.f(a)和f(b)的一阶导数\n3.y=f(x)是以T=b-为周期的周期函数a\n');
23  in=input('请输入对应序号：');
24
25  if in==1
26      M(1)=input('请输入f(a)的二阶导数值：');
27      M(n)=input('请输入f(b)的二阶导数值：');
28      A=zeros(n,n);       %构造追赶法所需的和Ab
29          for k=2:(n-1)
30          A(k,k)=2;
31          A(k,k+1)=mu(k-1);
32          A(k,k-1)=lambda(k-1);
33      end
34      A(1,1)=2;
35      A(1,2)=1;
36      A(end,end)=2;
37      A(end,end-1)=1;
38      b=zeros(n,1);
39      for k=2:(n-1)
40      b(k,1)=g(k-1);
41  end
42
43  b(1,1)=3*((y(2)-y(1))/h(1)-2*h(1)*M(1));
44  b(n,1)=3*((y(n)-y(n-1))/h(n-1)+2*h(n-1)*M(n));
45  b=b';
46  m=zhuigan(A,b);         %利用追赶法求解成功，这里的参数形式应为行向量而非列
        向量b
47
48  elseif in==2
49      y0=input('请输入f(a)的一阶导数值：');
50      yn=input('请输入f(b)的一阶导数值：');
51      A=zeros(n-2,n-2);       %构造追赶法所需的和Ab
52
53      for k=2:(n-3)
54          A(k,k)=2;
55          A(k,k+1)=mu(k);
56          A(k,k-1)=lambda(k);
57      end
58      A(1,1)=2;
59      A(1,2)=mu(1);
```

```
60        A(end,end)=2;
61        A(end,end-1)=lambda(n-2);
62        b=zeros(n-2,1);
63        for k=2:(n-3)
64        b(k,1)=g(k);
65    end
66    b(1,1)=g(1)-lambda(1)*y0;
67    b(end,1)=g(n-2)-mu(n-2)*yn;
68    b=b';
69    m=zhuigan(A,b);%利用追赶法求解
70    %这里解出m(1)至m(n-2)为能代入带一阶导数的分段三次埃米尔特插值多项式，要
          对,进行调整m
71    for k=(n-2):-1:1
72    m(k+1)=m(k);
73    end
74    m(1)=y0;
75    m(n)=yn;
76    elseif in==3
77    A=zeros(n,n);       %构造追赶法所需的和Ab
78    for k=2:(n-1)
79    A(k,k)=2;
80    A(k,k+1)=mu(k-1);
81    A(k,k-1)=lambda(k-1);
82    end
83    A(1,1)=2;
84    A(1,2)=mu(1);
85    A(1,end)=lambda(1);
86    A(end,end)=2;
87    A(end,end-1)=lambda(n-1);
88    A(end,1)=mu(n-1);
89    b=zeros(n-1,1);
90    for k=1:(n-1)
91    b(k,1)=d(k+1);
92    end
93    N=LU_fenjieqiuxianxingfangcheng(A,b);       %利用分解求解线性方程组LU
94    for k=1:(n-1)
95    M(k+1)=N(k,1);
96    end
97    M(1)=M(n);
98    else
99    fprintf('您输入的序号不正确');
100   end
101
```

```
102 function y=cubicspline3(f,x0,x)
103 %第三类周期边界条件下三次样条插值;
104 %x = linspace(-1,1,40);
105 %所求点; x
106 %所求点函数值; y
107 %x0 已知插值点;
108 %y0 已知插值点函数值;
109 %左端点一次导数值; f_0
110 %右端点一次导数值; f_n
111 % x0 = linspace(0,2*pi,20);
112 y0=feval(f,x0);
113 n = length(x0);
114 z = length(y0);
115 h = zeros(n-1,1);
116 k=zeros(n-2,1);
117 l=zeros(n-2,1);
118 S=2*eye(n);
119 for i=1:n-1
120 h(i)= x0(i+1)-x0(i);
121 end
122
123 for i=1:n-2
124 k(i)= h(i+1)/(h(i+1)+h(i));
125 l(i)= 1-k(i);
126 end
127
128 %对于第一种边界条件:
129 k = [0;k];
130 l = [l;0];
131
132 %构建系数矩阵: A
133 for i = 1:n-1
134 S(i,i+1) = k(i);
135 S(i+1,i) = l(i);
136 end
137 S(2,n)=S(2,1);
138 S(2,1)=0;
139 S(n,2)=h(1)/(h(1)+h(n-1));
140 S(n,n-1)=h(n-1)/(h(1)+h(n-1));
141
142 %建立均差表:
143 F=zeros(n-1,2);
144 for i = 1:n-1
```

```
F(i,1) = (y0(i+1)-y0(i))/(x0(i+1)-x0(i));
end
D = zeros(n-2,1);
for i = 1:n-2
F(i,2) = (F(i+1,1)-F(i,1))/(x0(i+2)-x0(i));
D(i,1) = 6 * F(i,2);
end

%构建函数：D
d0 =0;
dn = 6*(F(1,1)+F(n-1,1))/(h(1)+h(n-1));
D = [d0;D;dn];
m= S\D;
m(1)=m(n);
%寻找所在位置，并求出对应插值：x
for i = 1:length(x)
for j = 1:n-1
if (x(i)<=x0(j+1))&&(x(i)>=x0(j))
y(i) =( m(j)*(x0(j)-x(i))^3)/(6*h(j))+...
(m(j+1)*(x(i)-x0(j))^3)/(6*h(j))+...
(m(j)*(x(i)-x0(j))^2)/2+...
(y0(j)+(m(j)*h(j)^2)/6)*(x0(j)-x(i))/h(j)+...
(y0(j+1)-(m(j+1)*h(j)^2)/6)*(x(i)-x0(j))/h(j)+...
y0(j);
break;
else
continue;
end
end
end
plot(x,y,'b')%插值函数图像
% t=sin(x);
% hold on
% plot(x,t,'r') 原函数图像%
```

A.5 problem5

```
%% 实验五第四章数值微分与数值积分
clc,clear;
format long;
syms x t;
x1=0;
```

```
6       xr=1;
7       M = 2;
8       f = @(x) exp(-x^2/2)/sqrt(2*pi);
9       %标准解
10      s0 = int(f,x,0,1);
11      s0 = double(s0);
12      fprintf标准解积分结果(":%.12f\n\n",s0);
13
14      fprintf子区间数为(":"+num2str(M)+"\n\n");
15      % 复合梯度积分
16      s1 = traprl(f,xl,xr,M);
17      fprintf复合梯型积分结果(":%.12f\n",s1);
18      deta0 = abs(s0-s1);
19      fprintf复合梯度积分方法的误差绝对值(":%.12f\n\n",deta0);
20
21      % 积分Simpson
22      s2 = simprl(f,xl,xr,M);
23      fprintf("公式积分结果Simpson:%.12f\n",s2);
24      deta1 = abs(s0-s2);
25      fprintf("公式方法的误差绝对值Simpson:%.12f\n\n",deta1);
26
27      if (deta0<deta1)
28      fprintf复合梯度积分方法的误差较小。("\n")
29      else
30      fprintf("公式方法的误差较小。Simpson\n")
31      end
32
33      %% Simpson 公式求积分
34      function s=simprl(f,a,b,M)
35      % 求积公式Simpson
36      % function s=simprl(f,a,b,M)
37      % Input - f 是被积函数
38      %       - a 和是积分的上下界限 b
39      %       - M 为子区间的个数
40      h=(b-a)/(2*M);
41      s1=0;s2=0;
42      for k=1:M
43      x=a+h*(2*k-1);
44      s1=s1+feval(f,x);
45      end
46      for k=1:(M-1)
47      x=a+h*2*k;
48      s2=s2+feval(f,x);
```

```
    end
    s=h*(feval(f,a)+feval(f,b)+4*s1+2*s2)/3;
    end

    %% 复合梯形求积分方法
    function s=traprl(f,a,b,M)
    % 复合梯形求积公式
    % function s=traprl(f,a,b,M)
    % Input - f 是被积函数
    %       - a 和是积分的上下界限 b
    %       - M 为子区间的个数
    h = (b-a)/M;
    s=0;
    for k=1:(M-1)
    x=a+h*k;
    s = s+feval(f,x);
    end
    s = h*(feval(f,a)+feval(f,b))/2+h*s;
    end
```

A.6 problem6

```
%% 实验六第一题
clc,clear;
% 初始值
syms y;
f = @(t,y)1+y^2;
left = 0; % 定义边界
right = 1;
y0=0;

figure;
subplot(2,2,1);
hold on;
% 方法ode45
[type,values] = ode45(f,[0,1],0);
plot(type,values);
title('ODE45');
axis([0,1,0,1.5]);
hold off;

subplot(2,2,2);
```

```
hold on;
% 欧拉显格式
for M = 1:56
E = Euler1(f,left,right,0,M);
plot(E(:,1),E(:,2));
end
err = values-E(:,2);
title('Euler')
axis([0,1,0,1.5]);
hold off;

% 梯形预估修正格式
subplot(2,2,3);
hold on;

h=(right-left)/M;
for j=1:56
s = 0;
t = 0;
for i=1:j
t = [t,i*h];
s = [s,trape(f,left,i*h,j)];
end
plot(t,s);
end
err = [err,(values-t')];
title('Trapezoid Formula');
axis([0,1,0,1.5]);
hold off;

% 龙格库塔
subplot(2,2,4);
hold on;
for M=1:56
h = (right-left)/M;
E = runge_kutta1(f,0,h,left,right);
plot(E(:,1),E(:,2));
end
err = [err,(values-E(:,2))];
title("Runge-Kutta");
axis([0,1,0,1.5]);
hold off;

```

```
figure;
hold on;
subplot(1,3,1);
hold on;
plot(err(:,1));
title('Euler 方法');
hold off;
subplot(1,3,2);
hold on;
title('Trapezoid 方法');
plot(err(:,2));
hold off;
subplot(1,3,3);
hold on;
plot(err(:,3));
title('Runge-Kutta 方法');
hold off;
hold off;

function E=Euler1(f,a,b,ya,M)
%function E=euler(f,a,b,ya,M)
% Input  - f 是函数句柄
%        - a 和是左右的结束节点 b
%        - ya 是初始条件y(a)
%        - 是步数M
% Output -E=[T' Y'] 是横坐标向量T
%        - 是纵坐标向量Y
h=(b-a)/M;
% T=zeros(1,M+1);
Y=zeros(1,M+1);
T=a:h:b;
Y(1)=ya;
for j=1:M
Y(j+1)=Y(j)+h*feval(f,0,Y(j));
end
E = [T',Y'];
end

function t = trape(f,a,b,n)
% 复化梯形求积公式
%Input  -fun 被积函数 -[a,b] 积分区间 -n 等分n
%Output -t 积分值
x = linspace(a,b,n);
```

```
107 h = (b - a)/n;
108 s = 0;
109 for i = 2:n-1
110 s = s + 2*feval(f,0,x(i));
111 end
112 t = h/2*(feval(f,0,a) + feval(f,0,b) + s);
113
114 function E=runge_kutta1(f,y0,h,a,b)%参数表顺序依次是微分方程组的函数
    名称，初始值向量，步长，时间起点，时间终点（参数形式参考了函数）ode45
115 n=floor((b-a)/h);%求步数
116 x(1)=a;%时间起点
117 y(:,1)=y0;%赋初值，可以是向量，但是要注意维数
118 for t=1:n
119 x(t+1)=x(t)+h;
120 k1=f(x(t),y(:,t));
121 k2=f(x(t)+h/2,y(:,t)+h*k1/2);
122 k3=f(x(t)+h/2,y(:,t)+h*k2/2);
123 k4=f(x(t)+h,y(:,t)+h*k3);
124 y(:,t+1)=y(:,t)+h*(k1+2*k2+2*k3+k4)/6;
125 %按照龙格库塔方法进行数值求解
126 end
127 E = [x' y'];
128 end
129
130 %% pro2 第二题求解van der pol
131 clc,clear;
132 ts=[0 30];
133 x0=[1;0];
134 M=300;
135 mu = [0.01 0.1 1];
136 for i=1:3
137 f = @(t,x)vanderpol(mu(i),t,x);
138 E = rk_4(f,ts(1),ts(2),x0,M);
139 E = E';
140 t = E(:,1);
141 x1 = E(:,2);x2 = E(:,3);
142
143 figure;
144 hold on;
145 subplot(1,2,1);plot(t,E(:,2:3));
146 str = ['van␣der␣Pol␣','$\mu=',num2str(mu(i)),'$'];
147 title(str,'Interpreter','latex');
148 subplot(1,2,2);plot(x1,x2);
```

```
str = ['van der Pol ','$\mu=',num2str(mu(i)),'$'];
title(str,'Interpreter','latex');
hold off;
end
```

A.7 problem7

```
%% 实验七曲线拟合与函数逼近
clc,clear;
x=-2:1:2;
y=[0,1,2,1,0];
scatter(x,y);

xlabel '轴x'
ylabel '轴z'
title 散点图''
hold on;
grid on;

m=4;n=2;
A=zeros(n+1);
for j=1:n+1
    for i=1:n+1
        for k=1:m+1
            A(j,i)=A(j,i)+x(k)^(j+i-2);
        end
    end
end

B=[0 0 0];
for j=1:n+1
    for i=1:m+1
        B(j)=B(j)+y(i)*x(i)^(j-1);
    end
end

B=B';
a=inv(A)*B;
x=[-2.0:0.0001:2.0];
z=a(1)+a(2)*x+a(3)*x.^2;
plot(x,z)
```

```
35 legend('Points','y=a_{1}+a_{2}x+a_{3}x^{2}','Interpretation','
       latex')
36 title函数拟合图('')
37 fprintf正则方程左系数(":\n");
38 disp(A);
39 fprintf正则方程右项结果(":\n")
40 disp(B);
```

A.8 problem8

```
1  %% 实验九特征值与特征向量
2  % 第一题：
3  clc,clear;
4  times = 100;
5  tol = 1e-9;
6  A = [4,-1,1;-1,3,-2;1,-2,3];
7  v0 = [1,1,1]';
8
9  % 幂法求特征值与特征向量
10 fprintf幂法求特征值(":\n")
11 [lambda1,beta1] = power_method(A, times,v0,tol);
12 fprintf("*****************************\n");
13 fprintf主特征值为(":");
14 display(lambda1);
15 fprintf特征向量为(":\n");
16 display(beta1);
17
18 % 对称幂法
19 fprintf对称幂法求特征值(":\n")
20 [lambda2,beta2] = symmetry_method(A,times,v0,tol);
21 fprintf("*****************************\n");
22 fprintf主特征值为(":");
23 display(lambda2);
24 fprintf特征向量为(":\n");
25 display(beta2);
26
27 fprintf反幂法求特征值(":\n")
28 % 反幂法求特征值与特征向量
29 lambda_start = 0.8;
30 [lambda3,beta3] = inverse_power_method(A,times,v0,lambda_start,
       tol);
31 fprintf("*****************************\n");
```

```
32 fprintf主特征值为(":");
33 display(lambda3);
34 fprintf特征向量为(":\n");
35 display(beta3);
36
37 function [lambda,beta]=power_method(A,times,x0,epsilon)
38     %这个是一个检验能不能用幂法的函数A
39     %[lambda,beta]=power_method(A,times,x0,epsilon)
40     %是最后得出的特征值lambda
41     %是最后得到的特征值对应的特征向量beta
42     %是待求矩阵A
43     %是最大迭代次数times
44     %是初始向量x0
45     %是最大误差epsilon
46
47     format long;
48     b=eig(A);          %求的特征值所组成的向量A
49     c=size(b);
50     d=c(1)*c(2);   % 矩阵元素数量，即特征值的总个数
51     e=length(unique(b));   % 有几个代表值
52     if(d==e)
53         fprintf('特征值互异，是对角化矩阵\n');   %QUESTION1:fprintf
54         [lambda,beta]=power_method_cal(A,times,x0,epsilon);   %再
               去调用计算函数，代码在下面
55     else
56         error('特征值有重复，不是对角化矩阵\n');
57     end
58 end
59
60
61 function [alpha,x]=power_method_cal(A,times,x0,epsilon)
62     %[alpha,x]=power_method_cal(A,times,x0,epsilon)
63     format long;
64     k=1;
65     u=0;
66     [m,n]=size(x0);
67     y=zeros(m,n);
68     y=x0;
69     while k <= times
70         x=A*y;
71         alpha=max(x);     %用max()求一个矩阵中最大的元素
72         y=x./alpha;
73         fprintf第("%次迭代f\n",k);
```

```
74          fprintf('主特征值=%.8f\n',alpha)
75          fprintf('主特征向量v=\n');
76          disp(x);
77          if abs(alpha-u)<epsilon
78              break
79          else
80              k=k+1;
81              u=alpha;
82          end
83      end
84  end
85
86  function [lambda,beta]=symmetry_method(A,times,x0,epsilon)
87      %这个是一个检验能不能用幂法的函数A
88      %lambda,beta]=symmetry_method(A,times,x0,epsilon)
89      %是最后得出的特征值lambda
90      %是最后得到的特征值对应的特征向量beta
91      %是待求矩阵A
92      %是最大迭代次数times
93      %是初始向量x0
94      %是最大误差epsilon
95
96      if(sum(sum(~(A==A'))))
97      error这不是对称矩阵请检查("");
98      end
99
100     format long;
101     b=eig(A);          %求的特征值所组成的向量A
102     c=size(b);
103     d=c(1)*c(2);   % 矩阵元素数量，即特征值的总个数
104     e=length(unique(b));   % 有几个代表值
105     if(d==e)
106         fprintf('特征值互异，是对角化矩阵\n');   %QUESTION1:fprintf
107         [lambda,beta]=symmetry_method_cal(A,times,x0,epsilon);
                %再去调用计算函数，代码在下
                面
108     else
109     error('特征值有重复，不是对角化矩阵\n');
110 end
111
112
113 function [lambda,beta]=symmetry_method_cal(A,times,x0,epsilon)
114     %[lambda,beta]=symmetry_method_cal(A,x0,times,epsilon)
```

```
115     format long;
116     k=1;
117     x0 = x0/norm(x0);
118     flag = 0;
119     while(k<=times)
120         fprintf第("%次迭代f\n",k);
121         y=A*x0;
122         mu=x0'*y;
123         if(norm(y)==0)
124             fprintf特征向量(": v");
125             display(x0);
126             fprintf("有特征值为，选择新的初始向量A0x");
127         end
128         err = norm(x0 - y/(norm(y)));
129         x0 = y/norm(y);
130         fprintf最大特征值为(":%d\n",mu);
131         fprintf特征向量为(":");
132         display(x0);
133     if(err<epsilon)
134         flag=1;
135         lambda=mu;
136         beta=x0;
137         break;
138         else
139         k=k+1;
140     end
141     end
142     if(~flag)
143         fprintf超出最大迭代次数("");
144     end
145 end
146
147
148 function [lambda,beta]=inverse_power_method(A,times,x0,
        lambda_star,epsilon)
149     %[lambda,beta]=inverse_power_method(A,x0,lambda_star,times,
            epsilon)
150     %是最后得出的特征值lambda
151     %是最后得到的特征值对应的特征向量beta
152     %是待求矩阵A
153     %是初始向量x0
154     %是特征值的近似值lambda_starlambda
155     %是最大迭代次数times
```

```
156     %是最大误差epsilon
157
158     format long;
159     b=eig(A);
160     c=size(b);
161     d=c(1)*c(2);  % 矩阵元素数量
162     e=length(unique(b));  % 有几个代表值
163     if(d==e)
164         fprintf('特征值互异，是对角化矩阵\n');
165         [lambda,beta]=inverse_power_method_cal(A,x0,lambda_star,
                times,epsilon);
166         else
167         error('特征值有重复，不是对角化矩阵\n');
168     end
169 end
170
171
172 function [lambda,x]=inverse_power_method_cal(A,x0,lambda_star,
    times,epsilon)
173     %[lambda,beta]=inverse_power_method_cal(A,x0,lambda_star,
            times,epsilon)
174     format long;
175     k=1;
176     u=1;
177
178     [m,n]=size(x0);
179     y=zeros(m,n);
180     x=x0;
181
182     I=eye(m);
183     H=A-lambda_star.*I;
184     [L,U]=lu(H);       %分解函数（自带）LUmatlab
185     while k <= times
186         beta_b=max(x);
187         y=x./beta_b;
188         z=L\y;
189         x=U\z;
190         beta_b=max(x);
191         fprintf第("%次迭代f\n",k);
192         lambda=lambda_star+1/beta_b;
193         fprintf('主特征值=%.8f\n',lambda)
194         fprintf('主特征向量v=\n');
195         disp(x);
```

```
196             if abs(1/beta_b-1/u)<epsilon
197                 break
198             else
199                 k=k+1;
200                 u=beta_b;
201             end
202         end
203     end
204
205
206     %% 第二题：Houserholder
207         clc,clear;
208         A = [1 -1 1;1 -0.5 0.25;1 0 0;1 0.5 0.25;1 1 1];
209         A_hang = size(A,1);
210         A_lie = size(A,2);
211         H = cell( A_lie, 1 );
212         Hs = eye(A_hang, A_hang);
213         R2 = A;
214         Q2 = eye(A_hang,A_hang);
215         for i = 1:A_lie
216             [Hi, vi, betai] = Householder(R2(i:end,i));
217             fprintf第(""+num2str(i)次+"矩阵为Houserholder:");
218             display(Hi);
219             %H{i} = blkdiag(eye(i-1), Hi);
220             R2(i:end,i:end) = Hi*R2(i:end,i:end);
221             Q2 = Q2*blkdiag(eye(i-1), Hi);
222         end
223         Q2(find(abs(Q2) < 1e-10)) = 0;
224         R2(find(abs(R2) < 1e-10)) = 0;
225         [Q1, R1] = qr(A); %内部算法（非常非常的快）matlab
226         Q1*R1 - A;
227         Q2*R2 - A;
228         erroQ = Q1 + Q2;
229         erroR = R1 + R2;
230
231         function [ H, v, beta ] = householder( x )
232         % x : inout param. x is a vector which size is n*1
233         % v and beta : is param which construct H matrix
234         % H is hoseholder Matrix. H = I - beta*v*v'
235
236         %derive parameter v and beta
237         v = zeros(size(x));
238         beta = zeros(size(x));
```

```
239
240
241 %Houserholder Algorithm begin
242     x_len = length(x);
243     x_max = max(abs(x));
244     x = x./x_max;
245     zgama = x(2:end)'*x(2:end);
246     v(1) = 1;
247     v(2:end) = x(2:end);
248     if zgama == 0
249         beta = 0;
250     else
251         alpha = sqrt( x(1)^2 + zgama);
252         if   x(1) <=0
253             v(1) = x(1) - alpha;
254         else
255             v(1) = -zgama./( x(1) + alpha);
256         end
257         beta = 2*v(1)^2./( zgama + v(1)^2 );
258         v = v./v(1);
259     end
260     %beta = 2./(v'*v);
261     H = eye(x_len,x_len) - beta*v*v';
262 end
```