数值分析实验作业

宋振华

July 4, 2018

学院	泰山学堂
年级	2016级
专业	计算机科学与技术
学号	201605301357
指导老师	刘保东

Contents

1 Summary **3**

2 Chapter 2 **3**
2.1 Problem 1 3
2.2 Problem 2 8
2.3 Problem 3 9
2.4 Problem 4 13

3 Chapter 3 **18**
3.1 Problem 1 18

4 Chapter 4 **21**
4.1 Problem 1 21

5 Chapter 5 **26**
5.1 Problem 1 26
5.2 Problem 2 31

6 Chapter 6 **34**
6.1 Problem 1 34
6.2 Problem 2 35

7 Chapter 7 **37**
7.1 Problem 1 37

8 Chapter 8 **41**
8.1 Problem 1 41

9 Chapter 9 **47**
9.1 Problem 1 47

10 Conclusion **53**

1 Summary

数值计算指有效使用数字计算机求数学问题近似解的方法与过程, 主要研究如何利用计算机更好的解决各种数学问题, 包括连续系统离散化和离散形方程的求解, 并考虑误差、收敛性和稳定性等问题.

从数学类型来分, 数值运算的研究领域包括数值逼近、数值微分和数值积分、数值代数、最优化方法、常微分方程数值解法、积分方程数值解法、偏微分方程数值解法、计算几何、计算概率统计等. 随着计算机的广泛应用和发展, 许多计算领域的问题, 如计算物理、计算力学、计算化学、计算经济学等都可归结为数值计算问题.

本学期实验对常用算法进行总结, 并用MATLAB编程实现.

2 Chapter 2

2.1 Problem 1

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

1. $x_{k+1}=f_1(x_k)=15-2x_k^2$, $k=0,1,2,\cdots$, 取初始值$x_0=2$;
2. $x_{k+1}=f_2(x_k)=\frac{15}{2x_k+1}$, $k=0,1,2,\cdots$, 取初始值$x_0=2$;
3. $x_{k+1}=f_3(x_k)=x_k-\frac{2x_k^2+x_k-15}{4x_k+1}$, $k=0,1,2,\cdots$, 取初始值$x_0=2$.

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

迭代法收敛条件 ：设函数$\phi(x)$在区间$[a,b]$上满足条件:

1. $\forall x\in[a,b]$, 都有$a\le\phi(x)\le b$,
2. $\exists L\in(0,1)$, st $\forall x,y\in[a,b]$, $|\phi(x)-\phi(y)|\le L|x-y|$.

则有如下结论:

1. $x=\phi(x)$ 在$[a,b]$上有唯一的根x^*;
2. $\forall x_0\in[a,b]$, 迭代序列$x_{n+1}=\phi(x)$收敛于x^*;
3. $|x^*-x_n|\le\frac{L}{1-L}|x_n-x_{n-1}|$;
4. $|x^*-x_n|\le\frac{L^n}{1-L}|x_1-x_0|$.

一般来说, 构造迭代函数, 使其在预先指定的较大区间上满足上述定理的条件比较困难, 但若取初值x_0充分接近根x^*, 则收敛性的讨论可在根的附近进行.

局部收敛定理 ：设x^*为方程$x=\phi(x)$的根, 如果函数$\phi(x)$在x^*的某一邻域$O(x^*,\delta^*)$连续可微, 且$\left|\phi'(x)\right|<1$, 则$\exists\delta\in(0,\delta^*]$, st $\forall x_0\in[x^*-\delta,x^*+\delta]$, 迭代序列$x_{n+1}=\phi(x_n)$收敛于$x^*$.

不动点迭代法收敛速度 由于 $\lim\limits_{n\to\infty}\frac{p_{n+1}-p}{p_n-p}=\lim\limits_{n\to\infty}g^{'}(\xi_n)=g^{'}(p)$, 即 $\lim\limits_{n\to\infty}\frac{|p_{n+1}-p|}{|p_n-p|}=\left|g^{'}(p)\right|$, 即不动点迭代法线性收敛.

Code :

```
clear; clc;
F = {'@(x)15-2*x*x'; '@(x)15/(2*x+1)'; '@(x)x - (2*x*x+x-15)/(4*x+1)'};
N = 1000; x0=2;A=zeros(3,N); TOL = 1e-5;
A(1,1) = x0; A(2,1) = x0; A(3,1)=x0; mysize = [1,1,1];
for tp=1:3
  tmp = F(tp); i = 2; f = str2func(tmp{1});
  while i < N
    A(tp,i) = f(A(tp,i-1));
    if abs(f(A(tp,i)) - A(tp,i)) < TOL
      fprintf('Function %d: ',tp);
      fprintf('Answer %f Iteration times %d\n',A(tp,i),i);
      mysize(tp) = i; break;
    end
    i = i+1;
  end
  if i >= N
    fprintf('Function %d: Not convergence.\n',tp);
  end
end

```

Answer Iteration Sequence of $x_0 = 2$

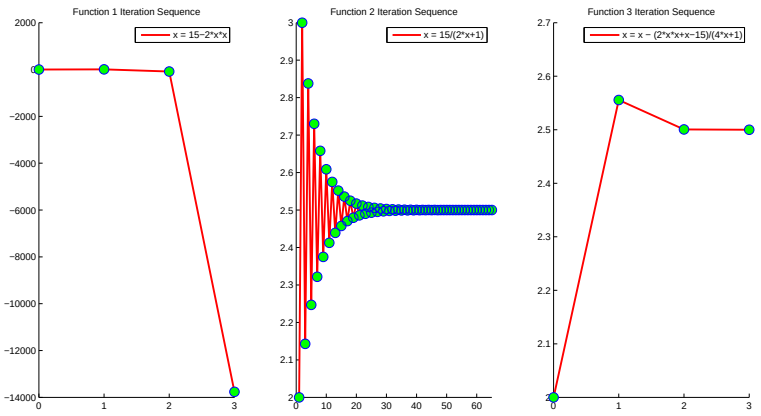

Figure 1: 不动点法迭代序列. 左图为$x_{k+1}=15-x_k^2$迭代序列, 在第3次迭代后, 取值已经过大. 此后迭代序列将很快溢出. 中图为$x_{k+1}=\frac{15}{2x_k+1}$, 右图为$x_{k+1}=x_k-\frac{2x_k^2+x_k-15}{4x_k+1}$. 可以观察出, $x_{k+1}=x_k-\frac{2x_k^2+x_k-15}{4x_k+1}$收敛速率更快

迭代函数	迭代次数	解	迭代停止条件	初始值
$x_{k+1} = 15 - x_k^2$	-	No Solution	$\|f(x) - x\| < TOL$	$x_0 = 2$
$x_{k+1} = \frac{15}{2x_k+1}$	64	2.499995	$\|f(x) - x\| < TOL$	$x_0 = 2$
$x_{k+1} = x_k - \frac{2x_k^2+x_k-15}{4x_k+1}$	3	2.500000	$\|f(x) - x\| < TOL$	$x_0 = 2$

稳定性测试 ：略微改变初值x_0, 观察解是否发生变化. 取$x_0^* = x_0 \pm 0.05, 0.10, 0.20$.

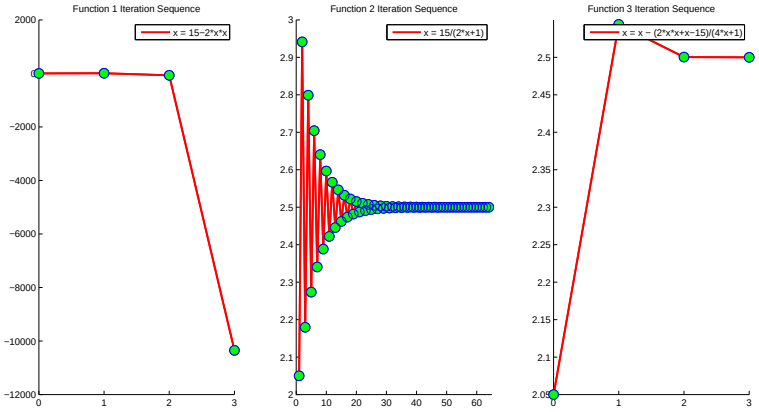

Figure 2: 不动点法迭代序列, $x_0^* = x_0 + 0.05$. 收敛性与初值x_0时相同.

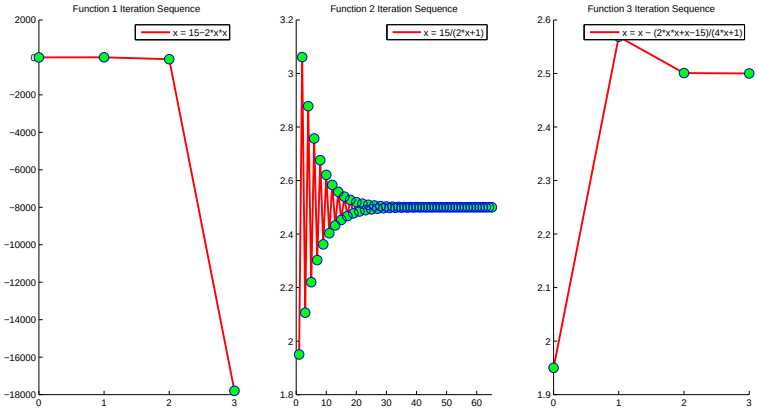

Figure 3: 不动点法迭代序列, $x_0^* = x_0 - 0.05$. 收敛性与初值x_0时相同.

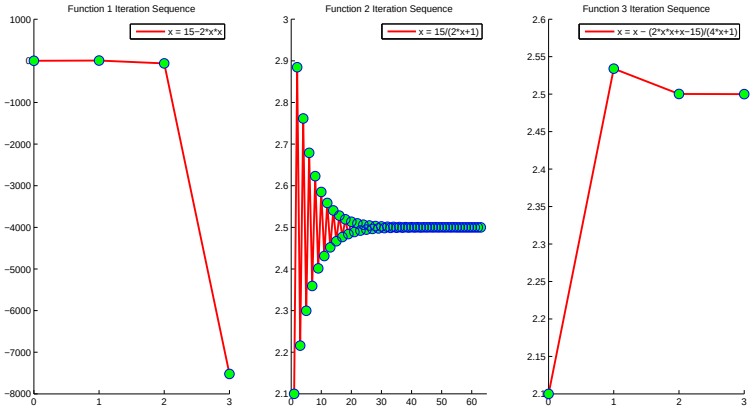

Figure 4: 不动点法迭代序列, $x_0^* = x_0 + 0.10$. 收敛性与初值x_0时相同.

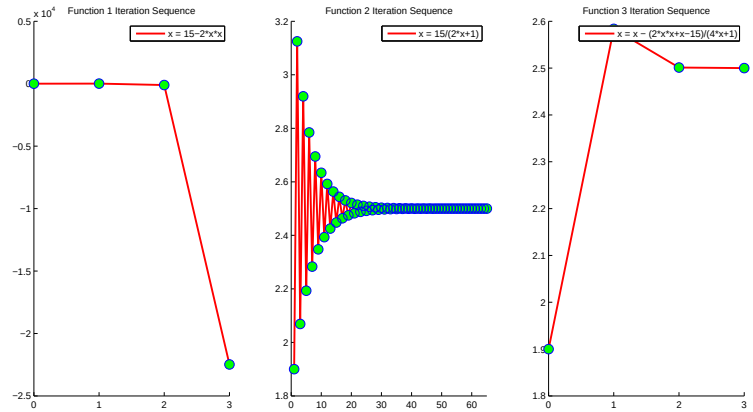

Figure 5: 不动点法迭代序列, $x_0^* = x_0 - 0.10$. 收敛性与初值x_0时相同.

稳定性测试结果:

	$x = f_1(x)$	$x = f_2(x)$		$x = f_3(x)$	
$x_0^* = x_0 + 0.05$	No Solution	63	2.500005	3	2.500000
$x_0^* = x_0 - 0.05$	No Solution	64	2.499995	3	2.500000
$x_0^* = x_0 + 0.10$	No Solution	62	2.499995	3	2.500000
$x_0^* = x_0 - 0.10$	No Solution	65	2.500005	3	2.500000
$x_0^* = x_0 + 0.20$	No Solution	61	2.500005	3	2.500000
$x_0^* = x_0 - 0.20$	No Solution	66	2.499995	3	2.500001

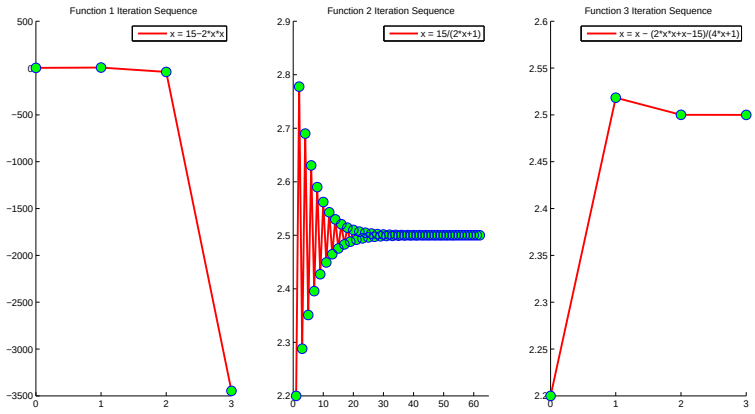

Figure 6: 不动点法迭代序列, $x_0^* = x_0 + 0.20$. 收敛性与初值x_0时相同.

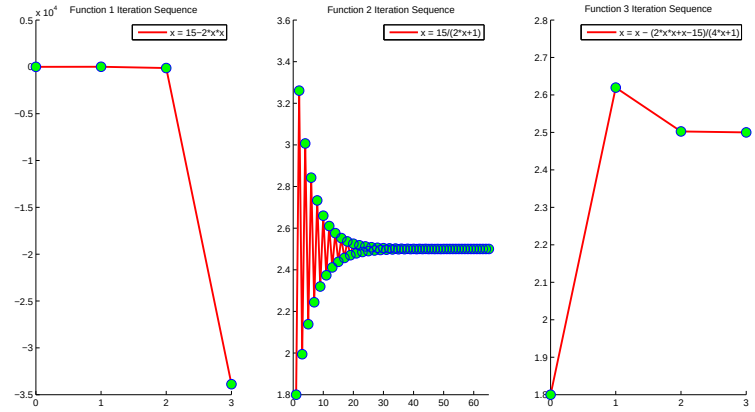

Figure 7: 不动点法迭代序列, $x_0^* = x_0 - 0.20$. 收敛性与初值x_0时相同.

Analysis :

1. 对于迭代$x^k = f_1(x_{k-1})$, 当$x \in [0,4]$时, $-17 \leq f_1(x) \leq 15$. 对于给定的初始点所在邻域, 不满足不动点迭代条件$f_1(x) \in [0,4]$, 即该迭代发散.
2. 对于迭代$x^k = f_2(x_{k-1})$, 当$x \in [1.5,4.5]$时, $f_2(x) \in [1.5,4.5]$, 满足不动点迭代条件, 该迭代收敛.
3. 对于迭代$x^k = f_3(x_{k-1})$, 当$x \in [1.5,4.5]$时, $f_3(x) \in [1.5,4.5]$, 满足不动点迭代条件, 该迭代收敛. 在此区间上, 由于$\max \left|f_3'(x)\right| < \max \left|f_2'(x)\right|$, 因

此使用$x^k = f_3(x_{k-1})$收敛速率更快速.

2.2 Problem 2

题目描述 :
证明方程$2-3x-\sin(x)=0$在$(0,1)$内有且仅有一个实根, 使用二分法求误差不大于0.0005的根, 以及需要的迭代次数.

Proof :
令$f(x)=2-3x-\sin(x)$.
则$f'(x)=-3-\cos(x)<0$.
而$f(0)=2>0, f(1)=1-\sin(1)<0$
因此方程$2-3x-\sin(x)=0$在$(0,1)$内有且仅有一个实根.

```
f = @(x)2-3*x-sin(x);
TOL = 0.0005; N = 100; Left = 0.0; Right = 1.0;
X = Left:(Right-Left)/400:Right;
Y = 2 - 3*X - sin(X);
FL = f(Left); CNT = 0; Mid = 0; A = [];
while CNT < N
  CNT = CNT + 1;
  Mid = (Left + Right)/2; FMid = f(Mid); A(CNT) = FMid;
  if FL*FMid < 0
    Right = Mid;
  else
    if FL*FMid == 0
      break
    else
      FL = FMid; Left = Mid;
    end
  end
  if Right - Left < TOL
    break
  end
end
subplot(1,2,1);
plot(X,Y,Mid,FMid,'o',[Mid,Mid],[min(Y),max(Y)],'r:',[X(1),X(length(X))],[FMid,FMid],':r');
title('Zero Point of Function')
legend('2-3*x-sin(x)','Zero Point')
subplot(1,2,2);
plot(1:CNT,A,'o',1:CNT,A,'-r');
title('Iteration Sequence of f(x)');
legend('f(x) of each iteration')
fprintf('Total time of iteration is %d. The answer is %f.',CNT,Mid)

```

输出 :
Total time of iteration is 11. The answer is 0.505371.

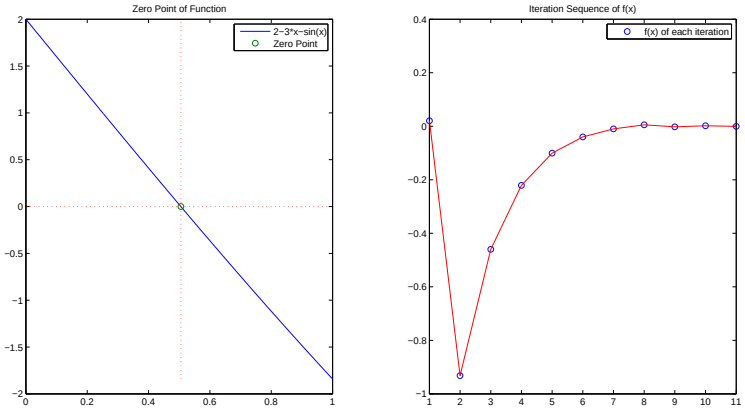

Figure 8: 二分法迭代结果. 左图为计算出的零点, 右图为$f\left(x\right)$迭代序列

Iteration Sequence Graph

二分法误差分析 ：

2.3 Problem 3

题目描述 ：
利用牛顿法求解方程
$\frac{1}{2}+\frac{1}{4}x^2-x\sin\left(x\right)-\frac{1}{2}\cos\left(2x\right)=0$ 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.

牛顿法推导 ：
假设$f\in C^2\left[a,b\right]$, 并且x^*是$f\left(x\right)=0$的一个解.

令$\overline{x}\in\left[a,b\right]$是对$x_*$的一个近似, 使得$f'\left(\overline{x}\right)\neq0$且$|\overline{x}-x^*|$比较小.

考虑$f\left(x\right)$在$\overline{x}$处展开的一阶泰勒多项式$f\left(x\right)=f\left(\overline{x}\right)+\left(x-\overline{x}\right)f'\left(\overline{x}\right)+\frac{\left(x-\overline{x}\right)^2}{2}f''\left(\xi\left(x\right)\right)$,
其中$\xi\left(x\right)$在x和$\overline{x}$之间.

因为$f\left(x^*\right)=0$, 令$x=x^*$, 此时有$0=f\left(x^*\right)=f\left(\overline{x}\right)+\left(x^*-\overline{x}\right)f'\left(\overline{x}\right)+\frac{\left(x^*-\overline{x}\right)^2}{2}f''\left(\xi\left(x\right)\right)$.

忽略余项, 得到$0=f\left(x^*\right)\approx f\left(\overline{x}\right)+\left(x^*-\overline{x}\right)f'\left(\overline{x}\right)$

求得$x^*\approx\overline{x}-\frac{f(\overline{x})}{f'(\overline{x})}$

因此定义迭代序列为: $x_n=x_{n-1}-\frac{f(x_{n-1})}{f'(x_{n-1})},\forall n\geq1.$

Code ：

```
clear;clc;
f = @(x)0.5+0.25*x^2-x*sin(x)-0.5*cos(2*x);
```

```
        x0 = pi*5;
        diff_f = @(x)x/2 + sin(2*x) - sin(x) - x*cos(x);
        TOL = 5e-5; N = 1000; CNT = 1; A = x0;
        while CNT < N
          CNT = CNT + 1;
          x = x0 - f(x0)/diff_f(x0); A(CNT) = x;
          if abs(x - x0) < TOL
            fprintf('Total time of iteration is %d. The answer is %
    f.',CNT,A(CNT))
            return;
          end
          x0 = x;
        end
        disp('No Solution');

```

Analysis :

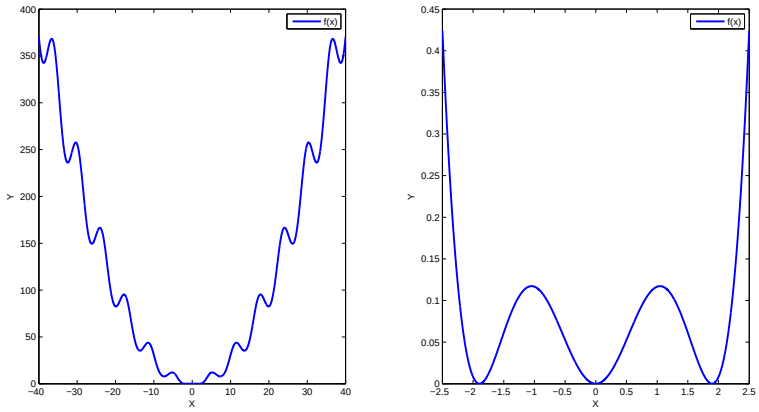

Figure 9: 函数图像. 观察左图可以发现, $\lim_{x\to\infty} f(x) = +\infty$, 函数的零点在$x = 0$附近. 观察右图可以发现, $f(x)$的一个零点位于$x = 0$处, 一个零点位于$(1.5, 2)$, 一个零点位于$(-2, -1.5)$. 根据初始值不同, 牛顿法迭代可能收敛到不同的零点, 也可能不收敛.

Answer

初始值x_0	$\pi/2$	5π	10π
迭代次数	12	16	不收敛
零点	1.895447	1.895452	-
TOL	5e-5	5e-5	5e-5

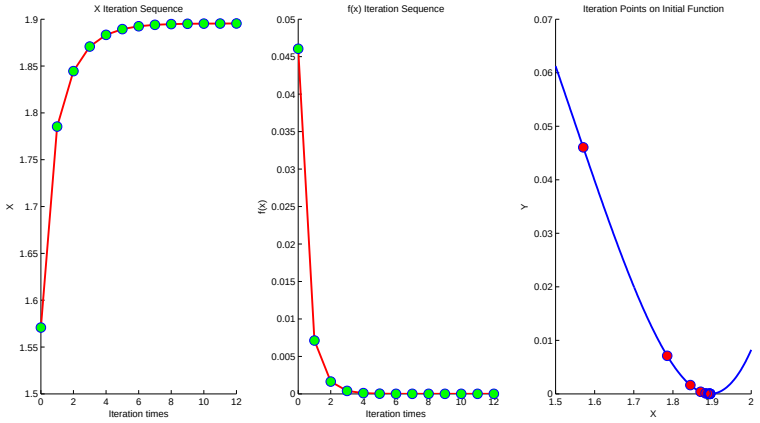

Figure 10: $x_0 = \pi/2$. 左图: x迭代序列. 中图: 迭代序列对应的$f(x)$值. 右图: 迭代点在函数图像上的分布

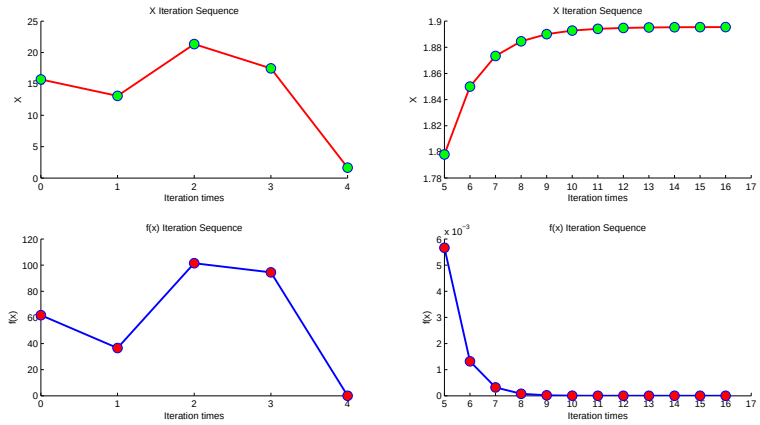

Figure 11: $x_0 = 5\pi$. 左上: x迭代序列$(t = 0, 1, 2, 3, 4)$; 右上: x迭代序列$(t = 5, \cdots, 16)$; 左下: 迭代序列对应的$f(x)$值$(t = 0, 1, 2, 3, 4)$; 右下: 迭代序列对应的$f(x)$值$(t = 5. \cdots, 16)$

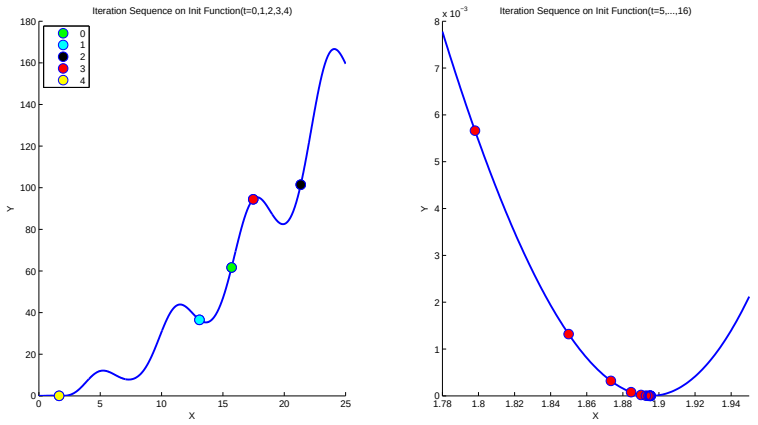

Figure 12: $x_0 = 5\pi$. 左图: 迭代点在函数图像上的分布($t = 0, 1, 2, 3, 4$); 右上:迭代点在函数图像上的分布($t = 0, 1, 2, 3, 4$)

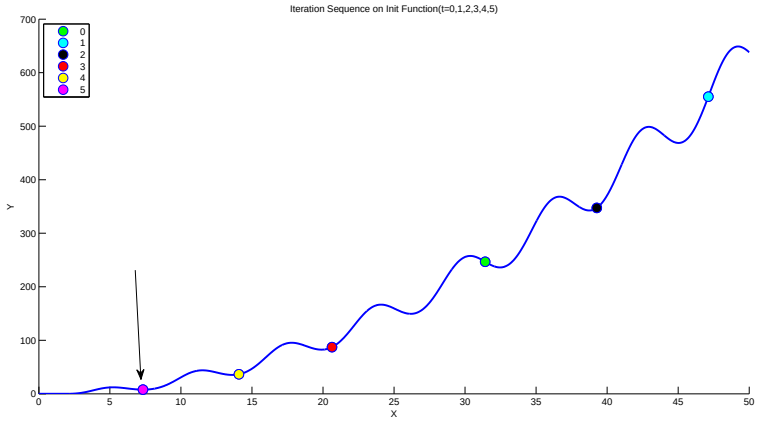

Figure 13: $x_0 = 10\pi$. 迭代点在函数图像上的分布($t = 0, 1, 2, 3, 4, 5$). 注意到$t = 5$时, 迭代点所处位置(箭头处), 函数导数约为0.

根据牛顿法迭代公式$x_n = x_{n-1} - \frac{f(x_{n-1})}{f'(x_{n-1})}$, 当在某一迭代点处, 导数约为0时, 下一次迭代数值将会很大, 这样将会导致不收敛或溢出的现象. 在本题中, $x_0 = 10\pi$时, 第6次迭代结果为$1.9215e + 03$.

本题函数有一定波动性, 使用牛顿法容易出现不收敛的情况. 观察结果可知, 如果给定的初始值比较靠近零点, 那么有利于牛顿法收敛. 当给定初始值偏离零点较远时, 容易出现不收敛的情况.

2.4 Problem 4

题目描述 ：
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、错位法设计相应的计算格式, 并编程求解(精确到4位小数).

分析 ：
当$x\in[0,1]$时, $f'(x)=5-e^x>0$, 而$f(0)=-1<0, f(1)=5-e>0$. 因此$f(x)$在$[0,1]$上有且仅有一个零点.

二分法 :代码

```
f = @(x)5*x-exp(x);
TOL = 5e-5; N = 100; Left = 0.0; Right = 1.0;
FL = f(Left); A = [];
X = Left:(Right - Left)/400:Right; Y = f(X);
CNT = 0; Mid = 0;
while CNT < N
  CNT = CNT + 1;
  Mid = (Left + Right)/2; FMid = f(Mid); A(CNT) = FMid;
  if FL*FMid < 0
    Right = Mid;
  else
    if FL*FMid == 0
      break
    else
      FL = FMid;
      Left = Mid;
    end
  end
  if Right - Left < TOL
    break
  end
end
subplot(1,2,1);
plot(X,Y,Mid,FMid,'o',[Mid,Mid],[min(Y),max(Y)],'r:',[X(1),X(length(X))],[FMid,FMid],':r');
title('Zero Point of Function')
legend('2-3*x-sin(x)','Zero Point')
subplot(1,2,2);
plot(1:CNT,A,'o',1:CNT,A,'-r');
title('Iteration Sequence of f(x)');
legend('f(x) of each iteration')
fprintf('Total time of iteration is %d. The answer is %.4f.',CNT,Mid)

```

Output Total time of iteration is 15. The answer is 0.2592.
Graph:

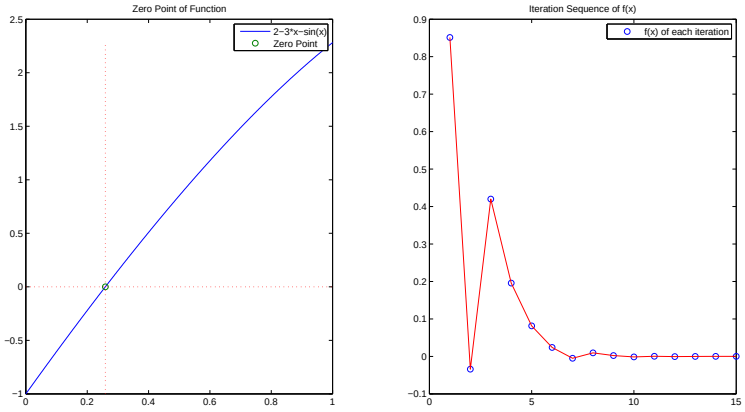

Figure 14: 二分法迭代序列. 左图为函数零点, 右图为迭代$f(x)$序列

牛顿法

代码 ：

```
f = @(x)5*x-exp(x); diff_f = @(x)5-exp(x);
Left = 0; Right = 1;
TOL = 5e-5; N = 100; CNT = 1;
x0 = 0; A = x0;
while CNT < N
  CNT = CNT + 1;
  x = x0 - f(x0)/diff_f(x0);
  A(CNT) = x;
  if abs(x - x0) < TOL
    break;
  end
  x0 = x;
end
fprintf('Total time of iteration is %d. The answer is %
f.',CNT,A(CNT))

```

Output Total time of iteration is 4. The answer is 0.259171. Graph:

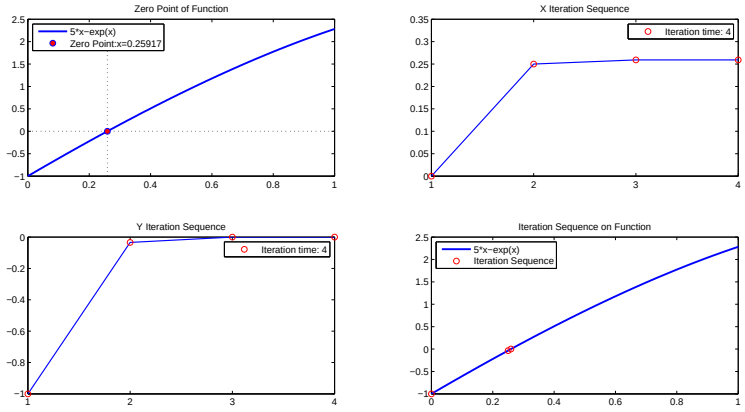

Figure 15: 牛顿法迭代序列. 左上为函数零点, 右上为x迭代序列, 左下为对应$f(x)$序列, 右上为迭代点在函数图像上的分布

割线法

推导 ：
用割线近似代替牛顿法中的切线. 得到公式$x_{k+1} = x_k - f(x_k)\frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})}$.

代码 ：

```
f = @(x)5*x-exp(x)
Left = 0; Right = 1;
TOL = 5e-5; N = 100; CNT = 1;
x0 = Right; x1 = Left; A = x1;
while CNT < N
  CNT = CNT + 1;
  x = x1 - f(x1)*(x1-x0)/(f(x1)-f(x0))
  A(CNT) = x;
  if abs(x - x1) < TOL
    fprintf('Total time of iteration is %d. The answer is %f.',CNT,x1);
    return;
  end
  x0 = x1; x1 = x;
end
disp('No Solution');

```

Output Total time of iteration is 5. The answer is 0.259156.
Graph:

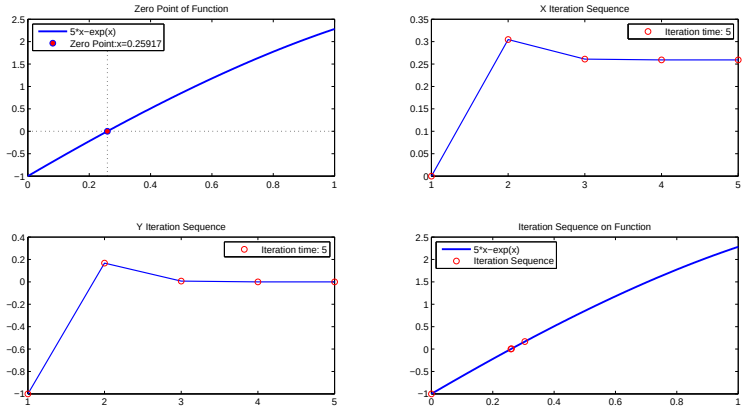

Figure 16: 割线法迭代序列. 左上为函数零点, 右上为x迭代序列, 左下为对应$f(x)$迭代序列, 右下为迭代点在函数图像上的分布

错位法

推导 :要在区间$[p_0, p_1]$上寻找$f(x) = 0$的解, 其中$f(p_0) f(p_1) < 0$.
使用与切线法相同的方式, 找到近似点p_2.
为确定使用哪一条线计算p_3, 计算$f(p_2) f(p_1)$和$f(p_2) f(p_0)$.
如果$f(p_2) f(p_1) < 0$, 说明p_1, p_2中间包含了一个根, 那么取$(p_1, f(p_1)), (p_2, f(p_2))$线段的斜率, 作为切线法中的斜率.

代码 :

```
f = @(x)5*x-exp(x);
Left = 0; Right = 1; TOL = 5e-5; N = 100; CNT = 1;
p0 = Left; p1 = Right; q0 = f(p0); q1 = f(p1); A = p1;
while CNT < N
  CNT = CNT + 1
  p = p1 - q1*(p1-p0)/(q1-q0); A(CNT) = p;
  if abs(p-p1) < TOL
    fprintf('Total time of iteration is %d. The answer is %f.',CNT,p);
    return
  end
  q = f(p)
  if q*q1 < 0
    p0=p; q0=q;
  else
    p1=p; q1=q;
  end
end
disp('No Solution')

```

Output Total time of iteration is 6. The answer is 0.259171. Graph:

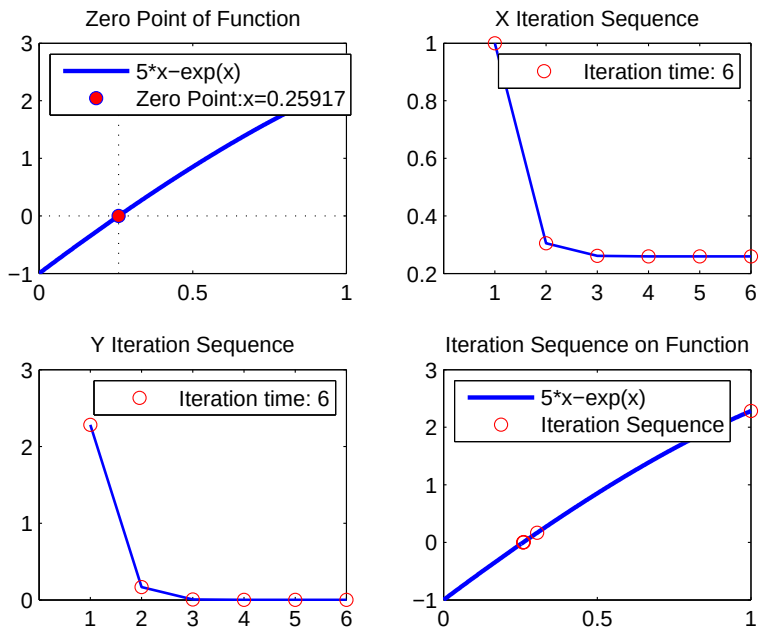

Figure 17: 错位法迭代序列. 左上为函数零点, 右上为x迭代序列, 左下为对应$f(x)$迭代序列, 右下为迭代点在函数图像上的分布

画图代码 :

```
        X = Left:(Right-Left)/400:Right; Y = f(X);
        B = f(A);
        subplot(2,2,1);
        plot([min(X),max(X)],[B(CNT),B(CNT)],':k',[A(CNT),A(CNT)
    ],[min(Y),max(Y)],':k');
        hold on
        tmp1 = plot(X,Y,'LineWidth',2);
        tmp2 = plot( A(CNT),B(CNT), 'o', 'markerfacecolor', [ 1,
    0, 0 ] );
        legend([tmp1,tmp2],'5*x-exp(x)',['Zero Point:x=' num2str(
    A(CNT))],2)
        title('Zero Point of Function');
        subplot(2,2,2);
        hold on;
        tmp3 = plot(1:CNT,A,'-b','LineWidth',1.2);
        tmp4 = plot(1:CNT,A,'or');
        set(gca,'xtick',1:CNT);
        legend(tmp4,['Iteration time: ' num2str(CNT)])
```

```
            title('X Iteration Sequence');
            subplot(2,2,3);
            hold on;
            tmp5 = plot(1:CNT,B,'-b','LineWidth',1.2);
            tmp6 = plot(1:CNT,B,'or');
            set(gca,'xtick',1:CNT);
            legend(tmp6,['Iteration time: ' num2str(CNT)])
            title('Y Iteration Sequence');
            subplot(2,2,4);
            hold on;
            tmp7 = plot(X,Y,'b','LineWidth',2);
            plot(A,B,'or');
            title('Iteration Sequence on Function');
            legend('5*x-exp(x)','Iteration Sequence',2);

```

3 Chapter 3

3.1 Problem 1

题目描述 ：
利用函数$y=\frac{1}{1+x^2}$, $x\in[-5,5]$生成相应的网格点数据, 分别取$n=2,4,6,8,10$, 画出该函数在$[-5,5]$上的n次拉格朗日多项式函数图形, 并与原函数图形比较, 分析插值多项式的次数对插值计算结果的影响.

拉格朗日插值多项式 对于给定的$n+1$个点, n次拉格朗日插值多项式以此通过这$n+1$个点.

拉格朗日多项式形式如下:

$$P(x)=\sum_{k=0}^{n}L_{n,k}(x)f(x)$$

.

其中$L_{n,k}(x)$为基函数, 满足这样的性质:

$$L_{n,k}(x_i)=\begin{cases}0, & i\neq k\\ 1, & i=k\end{cases}$$

取

$$L_{n,k}(x)=\prod_{i=0,i\neq k}^{n}\frac{x-x_i}{x_k-x_i}$$

.

拉格朗日余项 $f(x)=P(x)+R(x)$, 其中$P(x)$为n次多项式,

$$R(x)=\frac{f^{(n+1)}(\xi(x))}{(n+1)!}\prod_{i=0}^{n}(x-x_i)$$

为余项. 用于误差估计.

Code & Output :
拉格朗日插值:

```
function yy=lagrange(x1,y1,xx)
  syms x
  n=length(x1);
  for i=1:n
    t=x1;t(i)=[];L(i)=prod((x-t)./(x1(i)-t));
  end
  u=sum(L.*y1);
  yy=double(subs(u,x,xx));
end

```

对$y=\frac{1}{1+x^2}$处理、画图

```
clear
f = @(x)1/(1+x*x);
left = -5; right = 5;
X=left:(right-left)/200:right;
P = [];
for n=2:2:10
  step = (right - left) / n;
  A = left:step:right; B = [f(left)];
  for l = 1:n
    B(l+1) = f(A(l+1));
  end
  P = [P;lagrange(A,B,X)];
end
Y0=[];
for l = 1:length(X)
  Y0(l) = f(X(l));
end
plot(X,Y0,X,P(1,:),X,P(2,:),X,P(3,:),X,P(4,:),X,P(5,:))
legend('Initial Function','Lagrange function:2th','4th','6th','8th','10th');

```

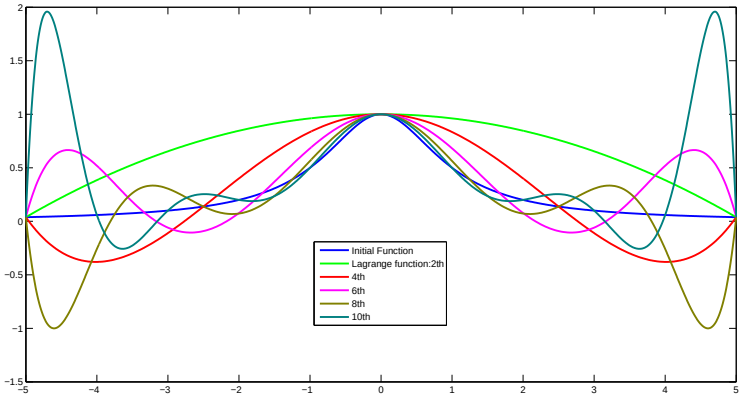

Figure 18: 插值曲线与原始函数比较. 包括了原始函数、$2, 4, 6, 8, 10$次插值多项式

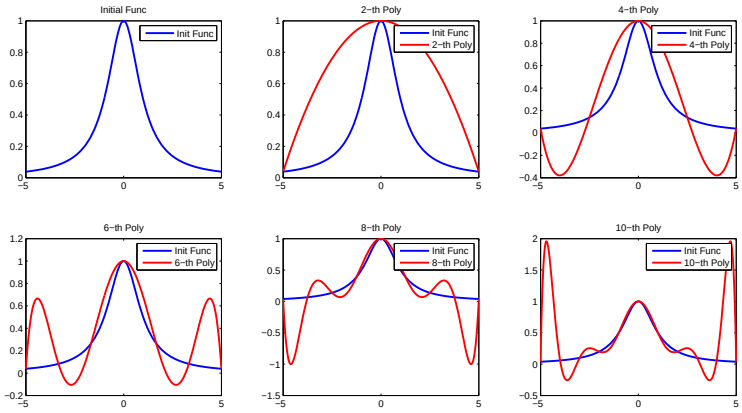

Figure 19: 插值曲线与原始函数比较. 分别为原始函数、$2, 4, 6, 8, 10$次多项式

分析 :

误差分析:

n-th Lagrange Poly	2	4	6	8	10
$\max\left\|f^{(n+1)}(\xi)\right\|$	4.6686	100.4528	4.3910e+03	3.2423e+05	3.6289e+07
$\max\left\|\prod_{i=0}^{n}(x-x_i)\right\|$	48.1124	354.6317	3.4236e+03	3.6722e+04	4.1655e+05
$\|R(x)\|$上界	37.4363	296.8646	2.9827e+03	3.2811e+04	3.7869e+05
$\max\|R(x)\|$	0.6462	0.4383	0.6169	1.0452	1.9156

对于此问题, 用给定的$R(x)$表达式来估算误差上界, 效果不好. 原因在于$\left|f^{(n+1)}(\xi)\right|$在$n$较大时, 在个别点取值较大, 从而影响了$\inf\left|f^{(n+1)}(x)\right|$取值.

在本题实际情况下, 往往取不到上界.

龙格现象:

误差项中含有多项式:$\prod_{i=0}^{n}(x-x_i)$. 对于该多项式, 绘图如下:

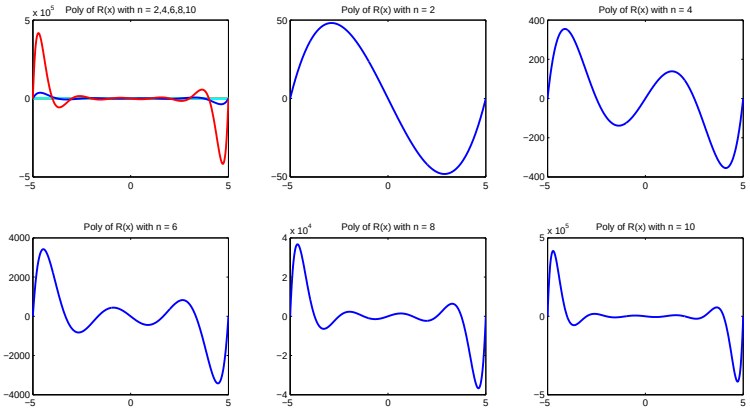

Figure 20: 误差项多项式比较. 左上为$2,4,6,8,10$次多项式总体比较. 之后依次是$2,4,6,8,10$次多项式

从图中可以发现, 当x偏离中心时, 多项式绝对值较大. 此为龙格现象发生的原因.

结论

1. 当拉格朗日插值多项式次数较小的时候, 由于采样点数目较少, 插值多项式不能很好地贴合原始函数.
2. 当多项式次数较大时, 在插值区间的边界部分插值函数会出现很大波动, 明显偏离原始函数(龙格现象); 其次, 从舍入误差看, 高次插值由于计算量大, 可能会产生严重的误差积累, 稳定性得不到保障. 所以拉格朗日插值次数不宜过高.

4 Chapter 4

4.1 Problem 1

题目描述 :

计算积分$I(x)=\int_0^{\frac{\pi}{2}} e^{\sin x}dx$的数值解:

1. 分别取$h=\frac{\pi}{8000},\frac{\pi}{800},\frac{\pi}{80}$, 用复合梯形格式计算其数值积分计算结果, 观察$h$的大小对计算结果的有效数字和绝对误差的影响.
2. 用辛普森公式重复上述计算过程.
3. 将计算结果的绝对误差与理论误差比较, 验证理论结果的正确性.

Question 1

梯形法简介 ：把被积区间划分成若干小区间, 对于每一小区间, 用函数两个端点与x轴围成的梯形面积, 近似代替函数图像与x轴围成的面积. 将所有小梯形面积累加, 即为结果.(或者说, 每一个小区间都用线性拉格朗日插值多项式来代替).

积分公式可以表述为:

$$f(x)\,dx = \frac{h}{2}\left[f(a) + 2\sum_{j}^{n-1} f(x_j) + f(b)\right] - \frac{b-a}{12}h^2 f''(\mu)$$

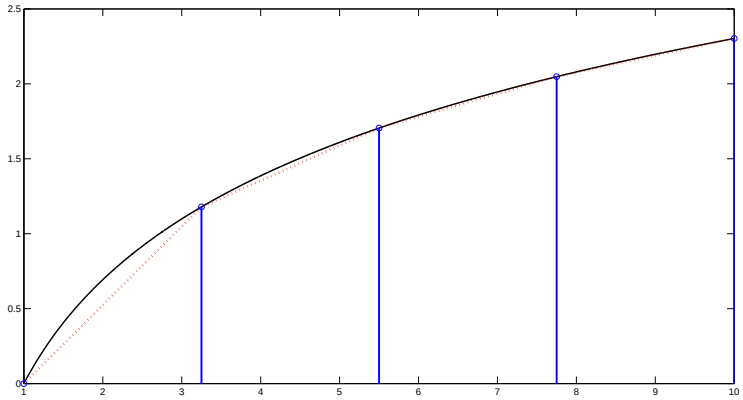

Figure 21: 梯形法图示

Code ：

```
left = 0; right = pi/2; h = pi/8000;
f = @(x)exp(sin(x));
tot = 0; tp = f(left);
for i=left+h:h:right
tr = f(i);
tot = tot + h/2*(tp + tr);
tp = tr;
end
fprintf('ans = %f\n',tot)

```

Answer

1. $h = \frac{\pi}{8000}$, $Ans = 3.104379$.

2. $h=\frac{\pi}{800}$, $Ans=3.104378$.

3. $h=\frac{\pi}{80}$, $Ans=ans=3.104251$.

Analysis 在$\left[0,\frac{\pi}{2}\right]$上, 令$f(x)=e^{\sin x}$, 则$f'(x)=e^{\sin x}\cos x>0$, $f''(x)=e^{\sin x}\cos^2 x-e^{\sin x}\sin x$. 有$\max\left|f''(x)\right|=f''(1)\approx -1.2748$.

因此误差项$\left|\frac{b-a}{12}h^2f''(\mu)\right|=\left|\frac{\pi/2*h^2}{12}f''(\mu)\right|\leq\left|\frac{\pi/2*h^2}{12}f''(1)\right|$

代入不同的h, 可以得到:

h	$\pi/80$	$\pi/800$	$\pi/8000$
$\left\|\frac{\pi/2*h^2}{12}f''(1)\right\|$	2.5734e-04	2.5734e-06	2.5734e-08

Question 2

辛普森公式推导 :

设$x_1=x_0+h, x_2=x_0+2h\,(h>0)$ $\forall x\in[x_0,x_2]$,$\exists\xi(x)\in[x_0,x_2]$, 使得

$$f(x)=f(x_1)+f'(x_1)(x-x_1)+\frac{f''(x_1)}{2}(x-x_1)^2+$$

$$\frac{f'''(x_1)}{6}(x-x_1)^3+\frac{f^{(4)}(\xi(x))}{24}(x-x_1)^4$$

. 积分, 得$\int_{x_0}^{x_2}f(x)\,dx=$

$$\left[f(x_1)(x-x_1)+\frac{f'(x_1)}{2}(x-x_1)^2+\frac{f''(x_1)}{6}(x-x_1)^3+\frac{f'''(x_1)}{24}(x-x_1)^4\right]_{x_0}^{x_2}$$

$$+\frac{1}{24}\int_{x_0}^{x_2}f^{(4)}(\xi(x))(x-x_1)^4\,dx$$

由于$\forall x\in[x_0,x_2],(x-x_1)^4\geq 0$, 因此

$$\frac{1}{24}\int_{x_0}^{x_2}f^{(4)}(\xi(x))(x-x_1)^4\,dx=\frac{f^{(4)}(\xi_1)}{24}\int_{x_0}^{x_2}(x-x_1)^4\,dx=\frac{f^{(4)}(\xi_1)}{120}(x-x_1)^5\,|_{x_0}^{x_2}$$

其中$\xi_1\in(x_0,x_2)$.

利用$h=x_2-x_1=x_1-x_0$, 上述积分表述为

$$\int_{x_0}^{x_2}f(x)\,dx=2hf(x_1)+\frac{h^3}{3}f''(x_1)+\frac{f^{(4)(\xi_1)}}{60}h^5$$

.

对于$f''(x_1)$, 用如下方式处理:

$$f(x_0+h)=f(x_0)+f'(x_0)h+\frac{1}{2}f''(x_0)h^2+\frac{1}{6}f^{(3)}(x_0)h^3+\frac{1}{24}f^{(4)}(\xi_1)h^4$$

.

$$f(x_0+h)=f(x_0)-f'(x_0)h+\frac{1}{2}f''(x_0)h^2-\frac{1}{6}f^{(3)}(x_0)h^3+\frac{1}{24}f^{(4)}(\xi_1)h^4$$

. 两式相加,

$$f(x_0+h)+f(x_0-h)=2f(x_0)+f''(x_0)h^2+\frac{h^4}{24}\left[f^{(4)}(\xi_1)+f^{(4)}(\xi_{-1})\right]$$

取$f^{(4)}(\xi)=\frac{1}{2}\left[f^{(4)}(\xi_1)+f^{(4)}(\xi_{-1})\right]$, 把$f''(x_1)$的值代入, 可以求得

$$\int_{x_0}^{x_2}f(x)dx=\frac{h}{3}\left[f(x_0)+4f(x_1)+f(x_2)\right]-\frac{h^5}{12}\left[\frac{1}{3}f^{(4)}(\xi_2)-\frac{1}{5}f^{(4)}(\xi_1)\right]$$

余项可以写成$\frac{h^5}{90}f^{(4)}(\xi)$的形式.

对于复合辛普森公式,

$$\int_a^b f(x)dx=\sum_{j=1}^{n/2}\int_{x_{2j-2}}^{x_{2j}}f(x)dx=$$

$$\frac{h}{3}\left[f(x_0)+2\sum_{j=1}^{n/2-1}f(x_{2j})+4\sum_{j=1}^{n/2}f(x_{2j-1}+f(x_n))\right]+E(f)$$

其中$E(f)=-\frac{h^5}{90}\sum_{j=1}^{n/2}f^{(4)}(\xi_j)=-\frac{b-a}{180}h^4f^{(4)}(\mu)$

辛普森公式实质: 相邻的3个点用二次多项式进行插值, 误差项可以用上述方式推导.

辛普森公式图示:

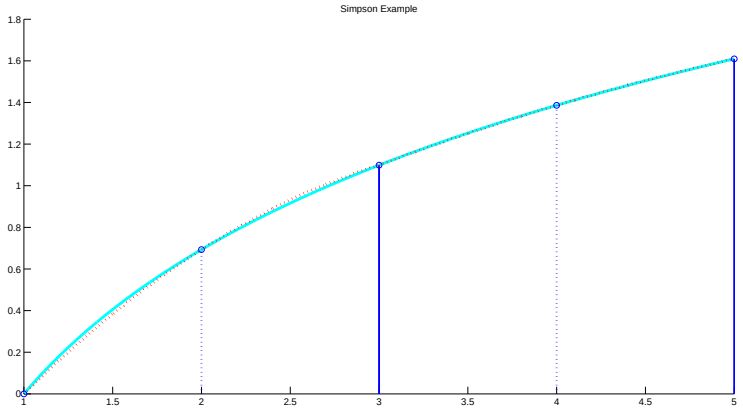

Figure 22: 辛普森公式图示

Code & Answer :

```
clear;
f = @(x)exp(sin(x));
left = 0; right = pi/2;
n = 4000; h = (right - left) / n;
tot = f(left) + f(right);
for j=1:n/2-1
  tot = tot + 2*f(left + 2*j*h);
end
for j=1:n/2
  tot = tot + 4*f(left + (2*j-1)*h);
end
fprintf('ans = %f\n',h*tot/3)

```

1. $h = \frac{\pi}{8000}$, $Ans = 3.104379017856$;
2. $h = \frac{\pi}{800}$, $Ans = 3.104379017856$;
3. $h = \frac{\pi}{80}$, $Ans = 3.104379017836$.

Analysis $|E(f)| = \left|\frac{b-a}{180}h^4 f^{(4)}(\mu)\right|$, 而$\max\left|f^{(4)}(\mu)\right| \approx 10.8731$.

h	$\pi/80$	$\pi/800$	$\pi/8000$
$\max\left\|\frac{b-a}{180}h^4 f^{(4)}(\mu)\right\|$	2.2565e-07	2.2565e-11	2.2565e-15

可以发现, 在相同步长的情况下, 辛普森公式比梯形法更加精确. 由于该积分不存在数值解, 而当h较小时, 复合辛普森公式的误差很小, 可以近似作为积分的精确值. 可以验证, h较小的复合辛普森公式结果, 与其他积分方式结果近似相等.

5 Chapter 5

5.1 Problem 1

题目描述 :
已知常微分方程初值问题:

$$\begin{cases} y' = y - \dfrac{2x}{y} \\ y_0 = 1 \end{cases}$$

1. 求该问题的解析解.
2. 取网格步长$h = 0.1, 0.05, 0.01, 0.001$, 分别用欧拉显式格式、预估校正格式、四阶龙格库塔格式求其数值解; 并与解析解比较, 分析各种格式的累积误差随网格步长大小的关系.

Question 1 :使用MATLAB求解该微分方程, 得到结果$y = \sqrt{2x+1}$.

```
y=dsolve('Dy = y-2*x/y','y(0)=1','x')

```

取$[0, 1]$作为计算处理区间. 可以得到, $y(1) \approx 1.732050807569$.

Question 2

欧拉显式格式 :
公式推导: 根据泰勒展开, 有
$y(t_{j+1}) + hf(t_j, y(t_j)) + \frac{h^2}{2} y''(\xi_j)$
忽略余项, 可以得到欧拉显式格式:

$$\begin{cases} w_0 = \alpha \\ w_{j+1} = w_j + hf(t_j, w_j) \end{cases}$$

误差分析:
假设f在$D = \{(t, y) | a \le t \le b, -\infty < y < \infty\}$连续,
满足Lipschitz条件(Lipschitz常数为L),
且满足$\left|y''(t)\right| \le M, \forall t \in [a, b]$.
则$|y(t_j) - w_j| \le \frac{hM}{2L}\left[e^{L(t_j - a)} - 1\right]$.
在某些情况下, M, L较难确定. 但可以发现, 随着向后递推, 欧拉显式格式的误差逐渐增大.

欧拉法误差分析:
对于$|y(x_j) - w_j| \le \frac{hM}{2L}\left[e^{L(x_j - a)} - 1\right]$, 需要确定$M, L$的值.
由于$M = \max\left|y''(x)\right|, x \in [a, b]$,
而$y(x) = \sqrt{2x+1}, y''(x) = \frac{-1}{(2x+1)^{3/2}}$.

有$M=\max\left|y''(x)\right|=\left|f''(-1)\right|=1$,

$\frac{\partial f}{\partial y}=1+\frac{2x}{y^2}=1+\frac{2x}{1+2x}$, $L=\max\left|\frac{\partial f}{\partial y}\right|=\left|\frac{\partial f}{\partial y}\big|_{x=1}\right|\approx 1.6667$.

因此$|y(1)-w_n|\leq\frac{hM}{2L}\left[e^L-1\right]$.

代码:

```
clear;
left = 0; right = 1;
f = @(x,y)y - 2*x/y; y0 = 1;
n_lst = [10, 20, 100 ,1000];
g = @(x)sqrt(2*x+1);
X = left:(right - left)/50:right; Y = [];
for i = 1:length(X)
  Y(i) = g(X(i));
end
for tp = 1:length(n_lst)
  n = n_lst(tp); h = (right - left) / n;
  A = []; A(1) = y0;
  for i = 2:n+1
    A(i) = A(i-1) + h * f(left + (i-2)*h, A(i-1));
  end
  fprintf('%f\n',A(n+1))
  subplot(2,2,tp);
  plot(X,Y,left:(right-left)/n:right,A);
  title(['Euler Method with n =' num2str(n)])
  legend('Initial Function','Euler Method',2)
end

```

结果: 取右端点的理论值与计算值, 作为衡量累计误差的标准. 随着n增大(h减小), 欧拉显式格式的误差逐渐减小.

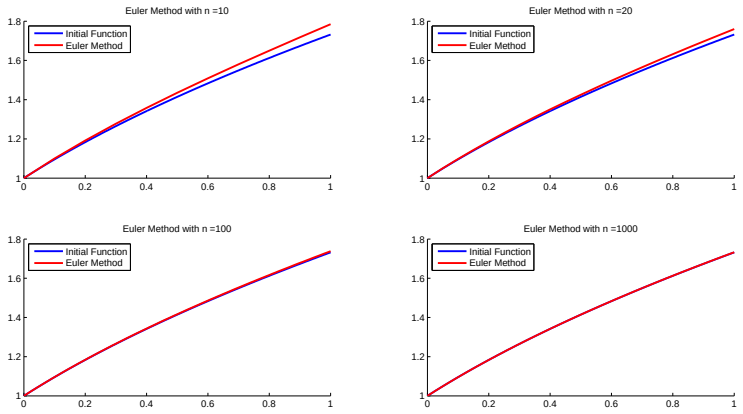

Figure 23: 欧拉法与原始函数比较. 红色为欧拉法结果, 蓝色为原始函数. 左上为$n = 10$, 右上为$n = 20$, 左下为$n = 100$, 右下为$n = 1000$. 可以看到, 随着n增大, 红色曲线与蓝色曲线趋于重合

结果对比:

n	10	20	100	1000
$y(1)$	1.78477083	1.76003786	1.73795017	1.73264816
Err	-0.05272002	-0.02798705	-0.00589936	-0.00059735
$\max \lvert y(1) - w_n \rvert$	0.1288	0.0644	0.0129	0.0013

预估校正格式 Steps

1. 使用4阶Runge-Kutta方法, 估计出y_1, y_2, y_3的值.
2. 使用4步Adams-Bashforth法作为预估:
   $y_4^{(0)} = y_3 + \frac{h}{24}\left[55f(t_3, y_3) - 59f(t_2, y_2) + 37f(t_1, y_1) - 9f(t_0, y_0)\right]$
3. 使用3步Adams-Moulton法作为校正:
   $y_{k+1} = y_i + \frac{h}{24}\left[9f\left(t_{i+1}, y_{i+1}^{(k)}\right) + 19f(t_i, y_i) - 5f(t_{i-1}, y_{i-1}) + f(t_{i-2}, y_{i-2})\right]$

代码:

```
clear;
left = 0; right = 1;
f = @(x,y)y - 2*x/y; y0 = 1;
n_lst = [10, 20, 100 ,1000];
g = @(x)sqrt(2*x+1);
X = left:(right - left)/50:right;Y = [];
for i = 1:length(X)
  Y(i) = g(X(i));
end

for tp=1:length(n_lst)
```

```
            n = n_lst(tp); h = (right-left)/n;
            t = left; w = y0; A = [y0];
            for i=1:3
              K1 = h*f(t,w);
              K2 = h*f(t+h/2,w+K1/2);
              K3 = h*f(t+h/2,w+K2/2);
              K4 = h*f(t+h,w+K3);
              w = w + (K1 + 2*K2 +2*K3 + K4)/6;
              t = t + h; A(i+1) = w;
            end
            w3 = A(4); w2 = A(3); w1 = A(2); w0 = A(1);
            t3 = left + 3*h; t2 = left + 2*h; t1 = left + h; t0 =
    left;
            for i=4:n
              t = t + h;
              w = w3 + h*(55*f(t3,w3) - 59*f(t2,w2) +37*f(t1,w1)-9*
    f(t0,w0))/24;
              w = w3 + h*(9*f(t,w) + 19*f(t3,w3) - 5*f(t2,w2) + f(
    t1,w1))/24;
              t0 = t1; t1 = t2; t2 = t3; t3 = t;
              w0 = w1; w1 = w2; w2 = w3; w3 = w; A(i+1) = w;
            end
            fprintf('%d %.12f\n',n,A(n+1))
            subplot(2,2,tp);
            plot(X,Y,left:(right-left)/n:right,A);
            title(['Predictor-Corrector with n =' num2str(n)])
            legend('Initial Function','Predictor-Corrector',2)
          end

```

结果:

n	10	20	100	1000
$f(1)$	1.732050719875	1.732052801860	1.732050818170	1.732050807570
Err	0.000000087694	-0.000001994291	-0.000000010601	-0.000000000001

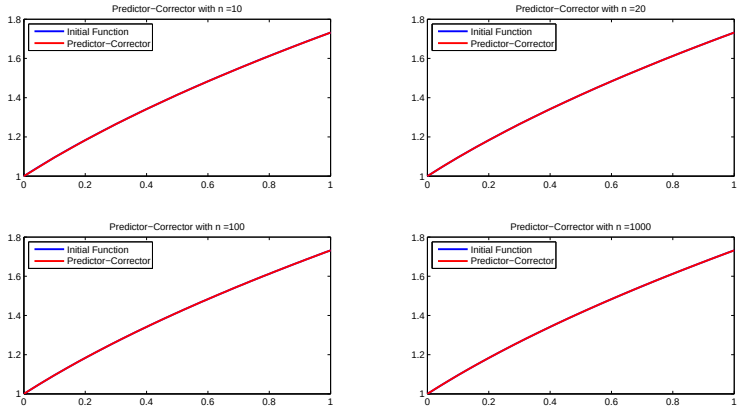

Figure 24: 预估校正法与原始函数比较. 红色为预估校正法结果, 蓝色为原始函数. 左上为$n = 10$, 右上为$n = 20$, 左下为$n = 100$, 右下为$n = 1000$. 可以看出, 相比欧拉法而言, 预估校正法具有更高的精度. 对于$n = 10$的情况, 两个函数曲线已经近似重合

四阶龙格库塔格式 :
代码:

```
clear;
left = 0; right = 1;
f = @(x,y)y - 2*x/y; y0 = 1;
n_lst = [10, 20, 100 ,1000];
g = @(x)sqrt(2*x+1);
X = left:(right - left)/50:right; Y = [];
for i = 1:length(X)
  Y(i) = g(X(i));
end
for tp=1:length(n_lst)
  n = n_lst(tp); h = (right-left)/n;
  t = left; w = y0; A = [y0];
  for i=1:n
    K1 = h*f(t,w);
    K2 = h*f(t+h/2,w+K1/2);
    K3 = h*f(t+h/2,w+K2/2);
    K4 = h*f(t+h,w+K3);
    w = w + (K1 + 2*K2 +2*K3 + K4)/6;
    t = t + h;
    A(i+1) = w;
  end
  fprintf('%f\n',A(n+1))
  subplot(2,2,tp);
  plot(X,Y, left:(right-left)/n:right ,A);
  title(['Runge–Kutta with n =' num2str(n)])
  legend('Initial Function','Runge–Kutta',2)
end

```

结果:

n	10	20	100	1000
$f(1)$	1.732056365166	1.732051148140	1.732050808103	1.732050807569
Err	-0.000005557597	-0.000000340571	-0.000000000534	-0.000000000000

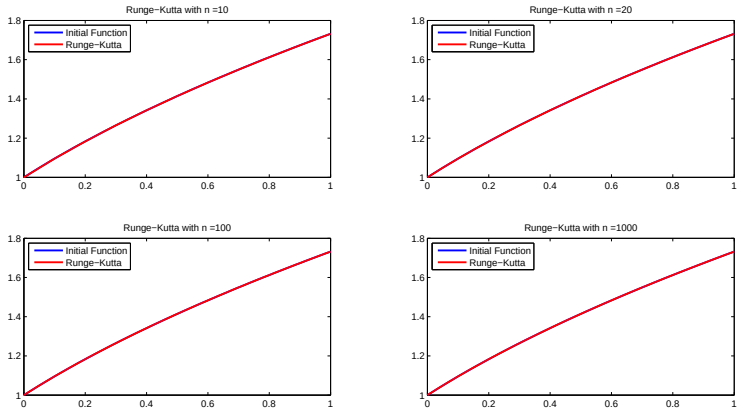

Figure 25: 龙格-库塔与原始函数比较. 红色为龙格-库塔法结果, 蓝色为原始函数. 左上为$n = 10$, 右上为$n = 20$, 左下为$n = 100$, 右下为$n = 1000$. 可以看出, 在$n = 10$的情况, 两个函数曲线已经近似重合

5.2 Problem 2

题目描述 :
求解描述振荡器的经典的van der Pol微分方程

$$\begin{cases} \dfrac{d^2y}{dt^2} - \mu\left(1 - y^2\right)\dfrac{dy}{dt} + y = 0 \\ y\left(0\right) = 1, y'\left(0\right) = 0 \end{cases}$$

试分别取$\mu = 10, 1, 0.1$, 分别用龙格库塔四阶格式计算其数值解, 并作图比较μ的大小对解的影响.

Analysis

常微分方程组解法　给定一个常微分方程组:

$$\begin{cases} \dfrac{dy_1}{dx} = f_1\left(x, y_1, y_2, \cdots, y_n\right) \\ \dfrac{dy_2}{dx} = f_2\left(x, y_1, y_2, \cdots, y_n\right) \\ \quad \vdots \\ \dfrac{dy_n}{dx} = f_n\left(x, y_1, y_2, \cdots, y_n\right) \end{cases}$$

初始状态$y_0^{(0)}, y_2^{(0)}, \cdots, y_n^{(0)}$已知.令

$$\begin{cases} K_i^{(1)} = h * f_i\left(x^{(k)}, y_1^{(k)}, \cdots, y_n^{(k)}\right) \\ K_i^{(2)} = h * f_i\left(x^{(k)} + \dfrac{h}{2}, y_1^{(k)} + \dfrac{1}{2}K_1^{(1)}, \cdots, y_n^{(k)} + \dfrac{1}{2}K_n^{(1)}\right) \\ K_i^{(3)} = h * f_i\left(x^{(k)} + \dfrac{h}{2}, y_1^{(k)} + \dfrac{1}{2}K_1^{(2)}, \cdots, y_n^{(k)} + \dfrac{1}{2}K_n^{(2)}\right) \\ K_i^{(4)} = h * f_i\left(x^{(k)} + h, y_1^{(k)} + K_1^{(3)}, \cdots, y_n^{(k)} + K_n^{(3)}\right) \end{cases}$$

高阶微分方程数值解　给定一个n阶微分方程, 且知道这个方程在某初始点处函数值和1到$n-1$阶导数值.求解方程:

$$y^{(n)} = f\left(x, y, y^{'}, \cdots, y^{(n-1)}\right)$$

令$y = y_1, y^{'} = y_2, y^{''} = y_3, \cdots, y^{(n-1)} = y_n$ 可以转化为下面的形式:

$$\begin{cases} \quad y = y_1 \\ \dfrac{dy_1}{dx} = y_2 \\ \dfrac{dy_2}{dx} = y_3 \\ \quad \vdots \\ \dfrac{dy_n}{dx} = f\left(x, y, y_1, \cdots, y_n\right) \end{cases}$$

于是这个问题转化为前面的n阶

本问题求解　令

$$\begin{cases} y_1 = y \\ y_2 = y_1^{'} \end{cases}$$

将Van der Pol微分方程化成标准形式:

$$\begin{cases} y_1^{'} = y_2 \\ y_2^{'} = \mu\left(1 - y_1^2\right) y_2 - y_1 \end{cases}$$

代码:

```
clear;
left = 0; right = 30; y1_0 = 1; y2_0 = 0;
n = 3000; h = (right-left)/n; miu_lst = [10, 1, 0.1];
f1 = @(t,y1,y2)y2; P = [];
for tp=1:length(miu_lst)
  miu = miu_lst(tp);
  f2 = @(t,y1,y2)miu*(1-y1*y1)*y2 - y1;
  y1 = y1_0; y2 = y2_0; A = [y1];
  for i=1:n
    t = left + i*h;
    K11 = h*f1(t, y1, y2);
    K21 = h*f2(t, y1, y2);

    K12 = h*f1(t + h/2, y1 + K11/2, y2 + K21/2);
    K22 = h*f2(t + h/2, y1 + K11/2, y2 + K21/2);

    K13 = h*f1(t + h/2, y1 + K12/2, y2 + K22/2);
    K23 = h*f2(t + h/2, y1 + K12/2, y2 + K22/2);

    K14 = h*f1(t + h, y1 + K13, y2 + K23);
    K24 = h*f2(t + h, y1 + K13, y2 + K23);

    y1 = y1 + (K11 + 2*K12 + 2*K13 + K14)/6;
    y2 = y2 + (K21 + 2*K22 + 2*K23 + K24)/6;
    A(i+1) = y1;
  end
  subplot(2,2,tp);
  plot(left:(right-left)/n:right,A);
  title(['with mu = ' num2str(miu)]);
  P = [P; A];
end
subplot(2,2,4); X = left:(right-left)/n:right;
plot(X,P(1,:),X,P(2,:),X,P(3,:));
title('All')
legend(['with mu = ' num2str(miu_lst(1))],['with mu = '
num2str(miu_lst(2))],['with mu = ' num2str(miu_lst(3))])

```

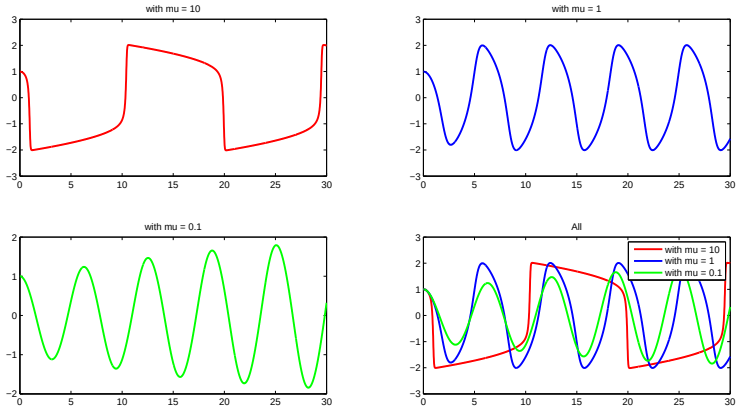

Figure 26: 龙格-库塔求解高阶微分方程(Van der Pol方程). 左上为$\mu = 10$的情况, 右上为$\mu = 1$的情况, 右下为3种情况对比

结论: 当μ减小时, 函数变得更加光滑, 且震荡速率加快.

6 Chapter 6

6.1 Problem 1

题目描述 :
求矩阵

$$\mathbf{A} = \begin{pmatrix} 2 & -1 & 1 \\ 3 & 3 & 9 \\ 3 & 3 & 5 \end{pmatrix}$$

的LU分解.

LU分解形式 :
设$\mathbf{A} = (a_{i,j})_{n\times n}$, 分解为$\mathbf{A} = \mathbf{LU}$的形式, 其中$\mathbf{L}$为对角元为1的下三角矩阵, $\mathbf{U}$为上三角矩阵. 即

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ l_{21} & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & \cdots & 1 \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ 0 & u_{22} & \cdots & u_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & u_{nn} \end{pmatrix}$$

按照矩阵乘法规则, 比较系数, 可得

$$\begin{cases} u_{1j} = a_{1j} & (j = 1, 2, \cdots, n) \\ l_{i1} = \dfrac{a_{i1}}{u_{11}} & (i = 2, 3, \cdots, n) \\ u_{ij} = a_{ij} - \displaystyle\sum_{k=1}^{i-1} l_{ik} u_{kj} & (i = 2, \cdots, n, j = i, \cdots, n) \\ l_{ij} = \left(a_{ij} - \displaystyle\sum_{k=1}^{j-1} l_{ik} u_{kj} \right) & (j = 1, 2, \cdots, n, i = j+1, \cdots, n) \end{cases}$$

代码 :

```
clear;
A = [2,-1,1; 3,3,9; 3,3,5]
n = length(A);
L=eye(n,n); U=zeros(n,n);
for k=1:n
  for j=k:n
    U(k,j)=A(k,j)-sum(L(k,1:k-1).*U(1:k-1,j)');
  end
  for i=k+1:n
    L(i,k)=(A(i,k)-sum(L(i,1:k-1).*U(1:k-1,k)'))/U(k,k);
  end
end
disp(L); disp(U);

```

结果 $\mathbf{L} = \begin{pmatrix} 1.0000 & 0 & 0 \\ 1.5000 & 1.0000 & 0 \\ 1.5000 & 1.0000 & 1.0000 \end{pmatrix}$ $\mathbf{U} = \begin{pmatrix} 2.0000 & -1.0000 & 1.0000 \\ 0 & 4.5000 & 7.5000 \\ 0 & 0 & -4.0000 \end{pmatrix}$

检验: $\mathbf{LU} = \mathbf{A}$.

6.2 Problem 2

题目描述 :
求一下对称矩阵的LL^T, LDL^T分解.

$$\mathbf{A} = \begin{pmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{pmatrix}$$

LDL^T分解 对于正定对称矩阵$\mathbf{A}$, 可以分解为如下形式:

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{12} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1n} & a_{2n} & \cdots & a_{nn} \end{pmatrix} = \mathbf{LDL}^T$$

$$= \begin{pmatrix} 1 & 0 & \cdots & 0 \\ l_{21} & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & \cdots & 1 \end{pmatrix} \begin{pmatrix} d_{11} & 0 & \cdots & 0 \\ 0 & d_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & d_{nn} \end{pmatrix} \begin{pmatrix} 1 & l_{21} & \cdots & l_{n1} \\ 0 & 1 & \cdots & l_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix}$$

$$= \begin{pmatrix} d_{11} & 0 & \cdots & 0 \\ l_{21}d_{11} & d_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ l_{n1}d_{11} & l_{n2}d_{22} & \cdots & d_{nn} \end{pmatrix} \begin{pmatrix} 1 & l_{21} & \cdots & l_{n1} \\ 0 & 1 & \cdots & l_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix}$$

因此,

$$\begin{cases} d_{11} = a_{11} \\ l_{j1} = \dfrac{a_{1j}}{d_{11}} & (j = 2, \cdots, n) \\ d_{ii} = a_{ii} - \displaystyle\sum_{k=1}^{i-1} l_{ik}^2 d_{kk} & (i = 2, 3, \cdots, n) \\ l_{ji} = \dfrac{a_{ij} - \sum_{k=1}^{i} l_{ik} l_{jk} d_{kk}}{d_{ii}} & (j = i+1, \cdots, n) \end{cases}$$

Code :

```
clear;
A = [2,-1,0;-1,2,-1;0,-1,2]; n = length(A);
d = zeros(n,n); L = eye(n,n);
d(1,1) = A(1,1);
for j=2:n,L(j,1) = A(1,j)/d(1,1); end
for i=2:n
  d(i,i) = A(i,i);
  for j=1:i-1
    d(i,i) = d(i,i) - L(i,j)^2 * d(j,j);
  end
  for j=i+1:n
    L(j,i) = A(i,j);
    for k=1:i-1
      L(j,i) = L(j,i) - L(i,k)*L(j,k)*d(k,k);
    end
    L(j,i) = L(j,i)/d(i,i);
  end
end
disp(L);disp(d)

```

Answer

$$\mathbf{L} = \begin{pmatrix} 1.0000 & 0 & 0 \\ -0.5000 & 1.0000 & 0 \\ 0 & -0.6667 & 1.0000 \end{pmatrix} \mathbf{D} = \begin{pmatrix} 2.0000 & 0 & 0 \\ 0 & 1.5000 & 0 \\ 0 & 0 & 1.3333 \end{pmatrix}$$

经检验, $\mathbf{LDL}^T = \mathbf{A}$.

$\mathbf{LL}^T$分解 :
令$\overline{\mathbf{D}} = \left(\sqrt{d_{i,j}}\right)_{n\times n}$, 则$\mathbf{A} = \mathbf{LDL}^T = \mathbf{L}\overline{\mathbf{DD}}\mathbf{L^T} = \left(\mathbf{L}\overline{\mathbf{D}}\right)\left(\mathbf{L}\overline{\mathbf{D}}\right)^T = \overline{\mathbf{LL}}^T$.

Code :

```
clear;
A = [2,-1,0;-1,2,-1;0,-1,2]; n = length(A);
d = zeros(n,n); L = eye(n,n);
d(1,1) = A(1,1);
for j=2:n,L(j,1) = A(1,j)/d(1,1); end
for i=2:n
  d(i,i) = A(i,i);
  for j=1:i-1
    d(i,i) = d(i,i) - L(i,j)^2 * d(j,j);
  end
  for j=i+1:n
    L(j,i) = A(i,j);
    for k=1:i-1
      L(j,i) = L(j,i) - L(i,k)*L(j,k)*d(k,k);
    end
    L(j,i) = L(j,i)/d(i,i);
  end
end
disp(L*sqrt(d))

```

Answer

$$\overline{\mathbf{L}} = \begin{pmatrix} 1.4142 & 0 & 0 \\ -0.7071 & 1.2247 & 0 \\ 0 & -0.8165 & 1.1547 \end{pmatrix}$$

经检验, $\overline{\mathbf{LL}}^T = A$.

7 Chapter 7

7.1 Problem 1

题目描述 :
求解线性方程组

$$\begin{cases} 4x_1 + 3x_2 = 24 \\ 3x_1 + 4x_2 = 30 \\ -x_2 + 4x_3 = -24 \end{cases}$$

分别利用Jacobi, Gauss-Seidel方法计算其数值解(取初始解向量$\mathbf{x} = (1,1,1)^T$), 对给定的收敛误差, 试比较不同范数(L_∞, L_2)对迭代次数的影响.

Jacobi

Jacobi迭代法推导 **假**设方程组

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\ \quad\quad \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_1 \end{cases}$$

的系数矩阵$\mathbf{A}$非奇异, 不妨设$a_{ii} \neq 0\,(i = 1, 2, \cdots, n)$.将方程组变形为:

$$\begin{cases} x_1 = \dfrac{1}{a_{11}}\left(-a_{12}x_2 - a_{13}x_3 - \cdots - a_{1n}x_n + b_1\right) \\ x_2 = \dfrac{1}{a_{22}}\left(-a_{21}x_1 - a_{23}x_3 - \cdots - a_{2n}x_n + b_2\right) \\ \quad\quad \vdots \\ x_n = \dfrac{1}{a_{nn}}\left(-a_{n1}x_1 - a_{n2}x_2 - \cdots - a_{n,n-1}x_{n-1} + b_n\right) \end{cases}$$

建立迭代公式:

$$\begin{cases} x_1^{(k+1)} = \dfrac{1}{a_{11}}\left(-a_{12}x_2^{(k)} - a_{13}x_3^{(k)} - \cdots - a_{1n}x_n^{(k)} + b_1\right) \\ x_2^{(k+1)} = \dfrac{1}{a_{22}}\left(-a_{21}x_1^{(k)} - a_{23}x_3^{(k)} - \cdots - a_{2n}x_n^{(k)} + b_2\right) \\ \quad\quad \vdots \\ x_n^{(k+1)} = \dfrac{1}{a_{nn}}\left(-a_{n1}x_1^{(k)} - a_{n2}x_2^{(k)} - \cdots - a_{n,n-1}x_{n-1}^{(k)} + b_n\right) \end{cases}$$

选定初始向量$\mathbf{x}^{(0)}$后, 反复迭代可以得到向量序列$\left\{\mathbf{x}^{(k)}\right\}$ 迭代公式为:

$$\begin{cases} \mathbf{x}^{(0)} = \left(x_1^{(0)}, x_2^{(0),\cdots,x_n^{(0)}}\right)^T \\ x_i^{(k+1)} = \dfrac{1}{a_{ii}}\left(b_i - \sum\limits_{j=1, j\neq i}^{n} a_{ij}x_j^{(k)}\right) \end{cases}$$

Jacobi迭代法也可以写为向量递推的形式

$$\mathbf{x}^{(k)} = \mathbf{T}\mathbf{x}^{(k)} + \mathbf{c}$$

设$\mathbf{D}$是对角元与$\mathbf{A}$相同的对角阵; $-\mathbf{L}$是严格下三角矩阵, 下三角元素与$\mathbf{A}$相同; $-\mathbf{U}$是严格上三角矩阵, 上三角元素与$\mathbf{A}$相同. 因此$\mathbf{A} = \mathbf{D} - \mathbf{L} - \mathbf{U}$.

对于方程$\mathbf{A}\mathbf{x} = \mathbf{b}$, 或$(\mathbf{D} - \mathbf{L} - \mathbf{U})\,\mathbf{x} = \mathbf{b}$,

可以变形为

$$\mathbf{D}\mathbf{x} = (\mathbf{L} + \mathbf{U})\,\mathbf{x} + \mathbf{b}$$

也就是

$$\mathbf{x} = \mathbf{D}^{-1}(\mathbf{L}+\mathbf{U})\mathbf{x} + \mathbf{D}^{-1}\mathbf{b}$$

写成递推式的形式:

$$\mathbf{x}^{(k)} = \mathbf{D}^{-1}(\mathbf{L}+\mathbf{U})\mathbf{x}^{(k-1)} + \mathbf{D}^{-1}\mathbf{b}$$

问题分析 在本题中, 方程组系数矩阵

$$\mathbf{A} = \begin{pmatrix} 4 & 3 & 0 \\ 3 & 4 & -1 \\ 0 & -1 & 4 \end{pmatrix}$$

由于$|\mathbf{A}| = 24 > 0$, $|\mathbf{D}| > 0$, 且$\rho(\mathbf{T}) \approx 0.7906 < 1$, 因此迭代收敛.

Code :

```
clear;
A = [4,3,0; 3,4,-1; 0,-1,4];
b = [24,30,-24];
x0 = [1,1,1]; x = x0;
TOL = 1e-6; N = 100;
for k=1:N
  for i=1:length(x0)
    x(i) = b(i);
  for j=1:length(x0)
    if j == i,continue;end
      x(i) = x(i) - A(i,j)*x0(j);
    end
    x(i) = x(i)/A(i,i);
  end
  if norm(x-x0,Inf) < TOL
    fprintf('Answer = (%f,%f,%f), Iteration times: %d\n
',x(1),x(2),x(3),k)
    return
  end
  x0 = x;
end
fprintf('No solution.\n')

```

对于2范数, 用如下方法求得:

```
if norm(x-x0,2) < TOL

```

Answer $\mathbf{x} = (3.000000, 4.000000, -5.000000)$, 与求解线性方程组结果一致.

范数	2	∞
迭代次数	69	68

由于$||\mathbf{x}||_\infty \leq ||\mathbf{x}||_2$, 因此使用无穷范数收敛速度略快.

Gauss-Seidel

Gauss-Seidel迭代法推导 ：如果把Jacobi迭代公式改成以下形式

$$\begin{cases} x_1^{(k+1)} = \dfrac{1}{a_{11}}\left(-a_{12}x_2^{(k)} - a_{13}x_3^{(k)} - \cdots - a_{1n}x_n^{(k)} + b_1\right) \\ x_2^{(k+1)} = \dfrac{1}{a_{22}}\left(-a_{21}x_1^{(k+1)} - a_{23}x_3^{(k)} - \cdots - a_{2n}x_n^{(k)} + b_2\right) \\ \vdots \\ x_n^{(k+1)} = \dfrac{1}{a_{nn}}\left(-a_{n1}x_1^{(k+1)} - a_{n2}x_2^{(k+1)} - \cdots - a_{n,n-1}x_{n-1}^{(k+1)} + b_n\right) \end{cases}$$

选取初始向量$\mathbf{x}^{(0)}$, 用迭代公式:

$$\begin{cases} \mathbf{x}^{(0)} = \left(x_1^{(0)}, x_2^{(0),\cdots,x_n^{(0)}}\right)^T \\ x_i^{(k+1)} = \dfrac{1}{a_{ii}}\left(b_i - \sum\limits_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum\limits_{j=i+1}^{n} a_{ij}x_j^{(k)}\right) \end{cases}$$

Gauss-Seidel迭代法也可以写为向量递推的形式:

$$\mathbf{x}^{(k)} = \mathbf{T}_g\mathbf{x}^{(k)} + \mathbf{c}_g$$

或

$$(\mathbf{D} - \mathbf{L})\,\mathbf{x}^{(k)} = \mathbf{U}\mathbf{x}^{(k-1)} + \mathbf{b}$$

即

$$\mathbf{x}^{(k)} = (\mathbf{D} - \mathbf{L})^{-1}\,\mathbf{U}\mathbf{x}^{k-1} + (\mathbf{D} - \mathbf{L})^{-1}\,\mathbf{b}$$

问题分析 ：
在本题中, $\rho(\mathbf{T}_g) \approx 0.6250 < 1$, 因此迭代收敛.

code

```
clear;
A = [4,3,0; 3,4,-1; 0,-1,4]; b = [24,30,-24];
x = [1,1,1]; TOL = 1e-6; N = 100; x0 = x;
for k=1:N
  for i=1:length(x)
    x(i) = b(i);
    for j=1:length(x)
      if j == i, continue; end
      x(i) = x(i) - A(i,j)*x(j);
    end
    x(i) = x(i)/A(i,i);
  end
  if norm(x-x0,Inf) < TOL
    fprintf('Answer = (%f,%f,%f), Iteration times: %d\n
',x(1),x(2),x(3),k)
    return
```

```
            end
            x0 = x;
        end
        fprintf('No solution.\n')

```

Answer :

范数	2	∞
迭代次数	27	27
结果	$(3.000001, 3.999999, -5.000000)$	$(3.000001, 3.999999, -5.000000)$

此时, 无穷范数迭代次数与2范数迭代次数相同.

对比Jacobi迭代法, Gauss-Seidel迭代法收敛速度更快. 但是对于某些问题, 可能Jacobi法收敛速度更快, 甚至可能出现Gauss-Seidel不收敛的情况.

8 Chapter 8

8.1 Problem 1

题目描述 :

[原书第八章习题一第13题(第9版本P509)] In a paper dealing with the efficiency of energy utilization of the larvae of the modest sphinx moth (Pachysphinx modesta), L. Schroeder [Schr1] used the following data to determine a relation between W, the live weight of the larvae in grams, and R, the oxygen consumption of the larvae in milliliters/hour. For biological reasons, it is assumed that a relationship in the form of $R = bW^a$ exists between W and R.

(a) Find the logarithmic linear least squares polynomial by using
$\ln R = \ln b + a \ln W$.

(b) Compute the error associated with the approximation in part (a):

$$\sum_{i=1}^{37} (R_i - bW_i^a)^2$$

(c) Modify the logarithmic least squares equation in part (a) by adding the quadratic term $c(\ln Wi)^2$, and determine the logarithmic quadratic least squares polynomial.

(d) Determine the formula for and compute the error associated with the approximation in part (c).

W	R	W	R	W	R	W	R	W	R
0.017	0.154	0.025	0.23	0.020	0.181	0.020	0.180	0.025	0.234
0.087	0.296	0.111	0.357	0.085	0.260	0.119	0.299	0.233	0.537
0.174	0.363	0.211	0.366	0.171	0.334	0.210	0.428	0.783	1.47
1.11	0.531	0.999	0.771	1.29	0.87	1.32	1.15	1.35	2.48
1.74	2.23	3.02	2.01	3.04	3.59	3.34	2.83	1.69	1.44
4.09	3.58	4.28	3.28	4.29	3.40	5.48	4.15	2.75	1.84
5.45	3.52	4.58	2.96	5.30	3.88			4.83	4.66
5.96	2.40	4.68	5.10					5.53	6.94

Question 1

推导 Cost Function:

$$Q = \sum_{i=1}^{m} (y_i - \beta_0 - \beta_1 x)^2$$

为使Cost Function最小, 对β_0, β_1分别求偏导数,有

$$\begin{cases} \dfrac{\partial Q}{\partial \beta_0} = -2 \sum\limits_{i=1}^{m} (y_i - \beta_0 - \beta_1 x_i) = 0 \\ \dfrac{\partial Q}{\partial \beta_1} = -2 \sum\limits_{i=1}^{m} (y_i - \beta_0 - \beta_1 x_i) x_i = 0 \end{cases}$$

求解方程组, 可得

$$\begin{cases} \beta_0 = \dfrac{\sum\limits_{i=1}^{m} x_i^2 \sum\limits_{i=1}^{m} y_i - \sum\limits_{i=1}^{m} x_i y_i \sum\limits_{i=1}^{m} x_i}{m \left(\sum\limits_{i=1}^{m} x_i^2 \right) - \left(\sum\limits_{i=1}^{m} x_i \right)^2} \\ \beta_1 = \dfrac{m \sum\limits_{i=1}^{m} x_i y_i - \sum\limits_{i=1}^{m} x_i \sum\limits_{i=1}^{m} y_i}{m \left(\sum\limits_{i=1}^{m} x_i^2 \right)^2 - \left(\sum\limits_{i=1}^{m} x_i \right)^2} \end{cases}$$

Code :

```
X = log(W); Y = log(R);
A = mypoly(X,Y);
left = min(X); right = max(X);
f = @(a,b,x)a*x+b;
DrX = [left -1/10*(right-left),right+1/10*(right-left)];
DrY = [f(A(2),A(1),DrX(1)),f(A(2),A(1),DrX(2))];
subplot(1,2,1);
plot(X,Y,'*',DrX,DrY)
```

```
legend('Initial Points','Linear Poly Fit',2);
title('lnR = lnb + alnW');
subplot(1,2,2)
b = exp(A(1)); a = A(2);
left = min(W); right = max(W);
X1 = left:(right-left)/50:right; Y1 = W;
for i=1:length(X1)
  Y1(i) = b*X1(i)^a;
end
plot(W,R,'*',X1,Y1);
legend('Initial Points','Poly Fit',2);
title('R = b*W^a')
E = 0;
for i=1:length(X)
  E = E + (R(i) - b*W(i)^a)^2;
end
disp(E)

```

求线性拟合最小二乘系数:

```
function A = mypoly(x,y)
  sx = 0; sx2 = 0; n = length(x);
  sxy = 0; sy = 0;
  for i=1:n
    sx2 = sx2+x(i)*x(i);
    sy = sy + y(i);
    sx = sx + x(i);
    sxy = sxy + x(i)*y(i);
  end
  A = [0,0];
  A(1) = (sx2 * sy - sxy * sx)/(n*sx2 - sx*sx);
  A(2) = (n*sxy - sx*sy)/(n*sx2 - sx*sx);
end

```

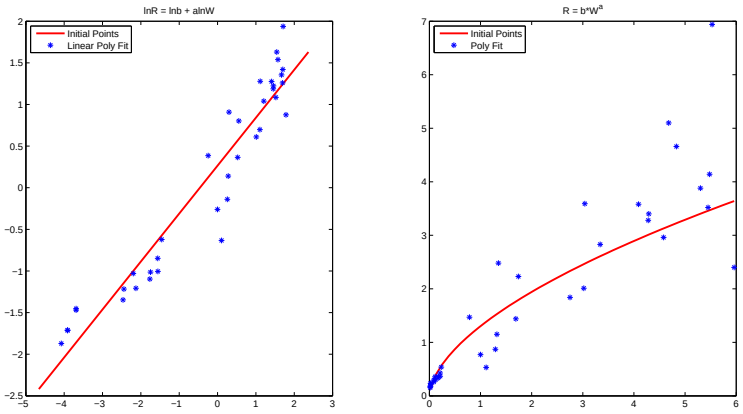

Figure 27: 拟合与原始点比较. 左图为$\ln R = \ln b + a \ln W$, 右图为$R = bW^a$

Answer 对于$\ln R = \ln b + a \ln W$, 取$\ln b = 0.2646, a = 0.5756$

Question 2 :

对于误差公式$E = \sum_{i=1}^{37} (R_i - bW_i^a)^2$, 计算可得$E = 25.2953$

Question 3

一般最小二乘多项式拟合形式 :

对于$P_n(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$ 目标:

$$\min E = \sum_{i=1}^{m} (y_i - P_n(x_i))^2 = \sum_{i=1}^{m} \left(y_i - \sum_{k=1}^{n} a_k x_i^k \right)^2$$

令

$$0 = \frac{\partial E}{\partial a_j} = -2 \sum_{i=1}^{m} y_i x_i^j + 2 \sum_{k=1}^{n} a_k \sum_{i=1}^{m} x_i^{j+k}$$

即

$$\sum_{k=1}^{n} a_k \sum_{i=1}^{m} x_i^{j+k} = \sum_{i=1}^{m} y_i x_i^j$$

令

$$\mathbf{R} = \begin{pmatrix} x_1^n & x_1^{n-1} & \cdots & x_1 & 1 \\ x_2^n & x_2^{n-1} & \cdots & x_2 & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_m^n & x_m^{n-1} & \cdots & x_m & 1 \end{pmatrix}, \mathbf{a} = \begin{pmatrix} a_n \\ a_{n-1} \\ \vdots \\ a_1 \\ a_0 \end{pmatrix}, \mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_{m-1} \\ y_m \end{pmatrix}$$

则$\mathbf{R}^T\mathbf{R}\mathbf{a} = \mathbf{R}^T\mathbf{y}$, 即$\mathbf{a} = \left(\mathbf{R}^T\mathbf{R}\right)^{-1}\mathbf{R}^T\mathbf{y}$

Code :
问题求解:

```
X = log(W); Y = log(R);
A = mypolyn(X,Y,2);
disp(A)
X_MIN = min(X); X_MAX = max(X); stp = X_MAX - X_MIN;
X1 = X_MIN-0.1*stp:1.2*stp/100: X_MAX+0.1*stp; Y1 = X1;
f = @(a,b,c,x)a*x*x+b*x+c;
for i=1:length(X1)
  Y1(i) = f(A(3),A(2),A(1),X1(i));
end
subplot(1,2,1);
plot(X,Y,'*',X1,Y1);
legend('Initial Value','Poly Fit');
title('lnR = a*(lnW)^2 + b*lnW + c');
subplot(1,2,2);
X1 = exp(X1); Y1 = exp(Y1);
plot(W,R,'*',X1,Y1);
legend('Initial Value','Fit Function');
title('R = exp(a*(lnW)^2 + b*lnW + c)');
E = 0;
for i=1:length(W)
  E = E + (R(i) - exp(f(A(3),A(2),A(1),X(i))))^2;
end
disp(E)

```

多项式拟合

```
function A = mypolyn(x,y,n)
  m = length(x); R = ones(m,n+1);
  for i=1:m
    for j=n:-1:1
      R(i,j) = R(i,j+1) * x(i,1);
    end
  end
  A = myLUsolver(R'*R, R'*y);
end

```

LU分解求线性方程组的解:

```
function MX = myLUsolver(A,b)
  n = length(A);
  L=eye(n,n); U=zeros(n,n);
  for k=1:n
    for j=k:n
      U(k,j)=A(k,j)-sum(L(k,1:k-1).*U(1:k-1,j)');
    end
    for i=k+1:n
      L(i,k)=(A(i,k)-sum(L(i,1:k-1).*U(1:k-1,k)'))/U(k,k);
    end
  end
  X=zeros(1,3);Y=zeros(1,3);
```

```
            Y(1)=b(1);
            for i=2:n
               for j=1:i-1
                  b(i)=b(i)-L(i,j)*Y(j);
               end
               Y(i)=b(i);
            end
            X(n)=Y(n)/U(n,n);
            for i=(n-1):-1:1
               for j=n:-1:i+1
                  Y(i)=Y(i)-U(i,j)*X(j);
               end
               X(i)=Y(i)/U(i,i);
            end
            MX = X';
         end

```

Answer 对于二次多项式拟合, 结果为: $\mathbf{a} = (0.0669, 0.7006, 0.0496)^T$.

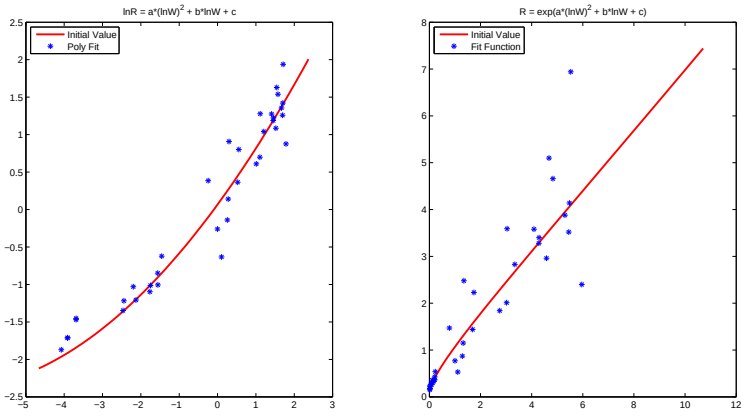

Figure 28: 拟合与原始点比较. 左图为$\ln R = a(\ln W)^2 + b\ln W + c$, 右图为$R = \exp\left(a(\ln W)^2 + b\ln W + c\right)$

Question 4 $E = 20.5971$

9 Chapter 9

9.1 Problem 1

题目描述 ：已知矩阵

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

是一个对称矩阵, 且其特征值为$\lambda_1=6,\lambda_2=3,\lambda_3=1$.分别利用幂法则, 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$x^{(0)}=(1,1,1)^T$

幂法

推导 **假**定$n\times n$的矩阵$\mathbf{A}$有n个特征值$|\lambda_1|>|\lambda_2|\geq\cdots\geq|\lambda_n|$, 对应线性无关的特征向量$\left\{\mathbf{v}^{(1)},\mathbf{v}^{(2)},\cdots,\mathbf{v}^{(n)}\right\}$,

$\forall\mathbf{x}\in\mathbb{R}^n,\exists\beta_1,\beta_2,\cdots,\beta_n,st\ \mathbf{x}=\beta_1\mathbf{v}^{(1)}+\beta_2\mathbf{v}^{(2)}+\cdots+\beta_n\mathbf{v}^{(n)}=\sum_{j=1}^n\beta_j\mathbf{v}^{(j)}$

即

$$\begin{aligned}\mathbf{A}\mathbf{x}=&\sum_{j=1}^n\beta_j\mathbf{A}\mathbf{v}^{(j)}=\sum_{j=1}^n\beta_j\lambda_j\mathbf{v}^{(j)}\\ \mathbf{A}^2\mathbf{x}=&\sum_{j=1}^n\beta_j\mathbf{A}\mathbf{v}^{(j)}=\sum_{j=1}^n\beta_j\lambda_j^2\mathbf{v}^{(j)}\\ \mathbf{A}^2\mathbf{x}=&\sum_{j=1}^n\beta_j\mathbf{A}\mathbf{v}^{(j)}=\sum_{j=1}^n\beta_j\lambda_j^2\mathbf{v}^{(j)}\\ &\vdots\\ \mathbf{A}^k\mathbf{x}=&\sum_{j=1}^n\beta_j\mathbf{A}\mathbf{v}^{(j)}=\sum_{j=1}^n\beta_j\lambda_j^k\mathbf{v}^{(j)}\\ =&\lambda_1^k\left(\beta_1\mathbf{v}^{(1)}+\sum_{j=2}^n\beta_j\left(\frac{\lambda_j}{\lambda_1}\right)^k\mathbf{v}^{(j)}\right)\end{aligned}$$

由于$\forall j\in\{2,\cdots,n\},|\lambda_1|>|\lambda_j|$,因此$\lim\limits_{k\to\infty}\left(\frac{\lambda_j}{\lambda_1}\right)^k=0.$ 即

$$\lim_{k\to\infty}\mathbf{A}^k\mathbf{x}=\lim_{k\to\infty}\lambda_1^k\beta_1\mathbf{v}^{(1)}$$

用$x_i^{(k)}$ 表示$\mathbf{x}^{(k)}$的第i个分量.由于

$$\frac{x_i^{(k+1)}}{x_i^{(k)}}\approx\frac{\lambda_1^{k+1}\beta_1\mathbf{v}_i^{(1)}}{\lambda_1^k\beta_1\mathbf{v}_i^{(k)}}=\lambda_1$$

需要注意, 当$|\lambda_1| > 1$时, $\mathbf{x}^{(k)}$的各分量趋于无穷, 当$|\lambda_1| < 1$时, $\mathbf{x}^{(k)}$的各分量趋于0. 为了克服这一缺点, 需要将迭代向量规范化. 采用如下方式进行迭代:

$$\begin{cases} \mathbf{x}^{(0)} = \mathbf{y}^{(0)} \neq \mathbf{0} \\ \mathbf{x}^{(k)} = \mathbf{A}\mathbf{y}^{(k-1)} \\ \mathbf{y}^{(k)} = \dfrac{\mathbf{x}^{(k)}}{\left|\left|\mathbf{x}^{(k)}\right|\right|_\infty} \end{cases}$$

$$\lim_{k\to\infty} \mathbf{y}^{(k)} = \frac{v^{(1)}}{\left|\left|\mathbf{v}^{(1)}\right|\right|}, \lim_{k\to\infty} ||\mathbf{x}||_\infty = \lambda_1$$

.

易知最终的$\mathbf{x}$为所求特征向量.

Code :

```
clear;
A = [4,-1,1;-1,3,-2;1,-2,3]; n = length(A);
x0 = [1;1;1]; y0 = x0; N = 100; TOL = 1e-7;
mx0 = max(x0);
for i=1:N
  x = A*y0; mx = max(x);
  y0 = x/mx;
  if abs(mx-mx0) < TOL
    fprintf('Answer %f Iteration time %d\n',mx,i)
    disp(x);
    return
  end
  x0 = x; mx0 = mx;
end
fprintf('No Solution\n');

```

Answer Max Eigen Value: 6.000000, Iteration time 27.
Eigen Vector: $\mathbf{x} = (6.0000, -6.0000, 6.0000)^T$

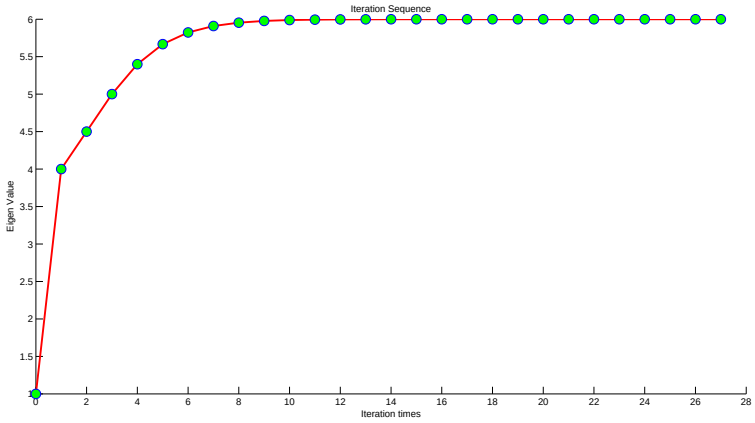

Figure 29: 幂法特征值迭代序列

对称幂法

简介 **假**设$\mathbf{A}$是对称矩阵, $\mathbf{A}$有n个实数特征向量$|\lambda_1| > |\lambda_2| > \cdots > |\lambda_n|$, 对应$n$个标准正交特征向量$\left\{\mathbf{v}^{(1)}, \mathbf{v}^{(2)}, \cdots, \mathbf{v}^{(n)}\right\}$.

$\forall \mathbf{x}_0 \in \mathbb{R}^n, \mathbf{x}_0 = \beta_1 \mathbf{v}^{(1)} + \beta_2 \mathbf{v}^{(2)} + \cdots + \mathbf{v}^{(n)}$.

对于$\mathbf{x}_k = \mathbf{A}^k \mathbf{x}_0, \mathbf{x}_k = \beta_1 \lambda_1^k \mathbf{v}^{(1)} + \beta_2 \lambda_2^k \mathbf{v}^{(2)} + \cdots + \beta_n \lambda_n^k \mathbf{v}^{(n)}$. 因为$n$个特征向量标准正交, 因此

$$\mathbf{x}_k^T \mathbf{x}_k = \sum_{j=1}^{n} \beta_j^2 \lambda_j^{2k} = \beta_1^2 \lambda_1^{2k} \left[1 + \sum_{j=2}^{n} \left(\frac{\beta_j}{\beta_1}\right)^2 \left(\frac{\lambda_j}{\lambda_n}\right)^{2k}\right]$$

且

$$\mathbf{x}_k^T \mathbf{A} \mathbf{x}_k = \sum_{j=1}^{n} \beta_j^2 \lambda_j^{2k+1} = \beta_1^2 \lambda_1^{2k+1} \left[1 + \sum_{j=2}^{n} \left(\frac{\beta_j}{\beta_1}\right)^2 \left(\frac{\lambda_j}{\lambda_1}\right)^{2k+1}\right]$$

于是

$$\lim_{k\to\infty} \frac{\mathbf{x}_k^T \mathbf{A} \mathbf{x}_k}{\mathbf{x}_k^T \mathbf{x}_k} = \lambda_1, \lim_{k\to\infty} \frac{\mathbf{x}_k}{||\mathbf{x}_k|_2|} = \frac{\mathbf{v}^{(1)}}{\left|\left|\mathbf{v}^{(1)}\right|\right|_2}$$

.

code :

```
clear;clc;
A = [4,-1,1;-1,3,-2;1,-2,3]; n = length(A);
x = [1;1;1]; N = 1000; TOL = 1e-7;
x = x/norm(x,2);
for i = 1:N
```

```
            y = A*x; u = x'*y;
            if norm(y,2) < 1e-10
              fprintf('A has eigen value 0. select new vector x
    and restart.\n')
              return
            end
            norm_y = norm(y,2);
            err = norm(x-y/norm_y,2);
            x = y/norm_y;
            if err < TOL
              fprintf('Answer %f Iteration time %d\n',u,i)
              disp(x')
              return
            end
          end
          fprintf('No Solution\n')

```

Answer Max Eigen Value: 6.000000, Iteration time 24
特征向量:

$$\mathbf{x} = (0.5774, -0.5774, 0.5774)$$

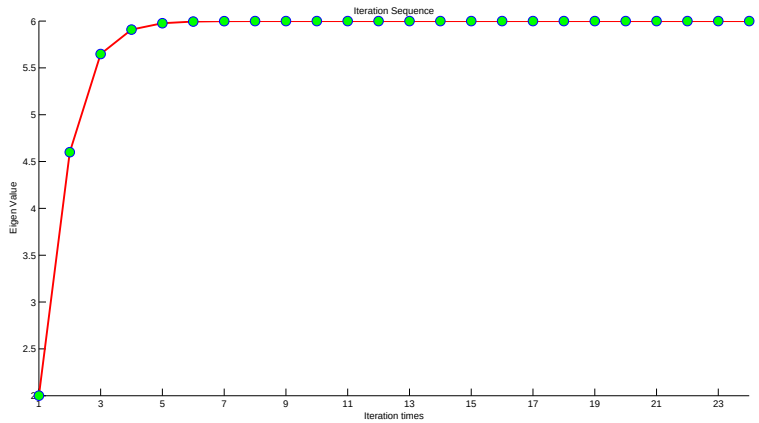

Figure 30: 对称幂法特征值迭代序列

反幂法

推导 设$\mathbf{A}$为$n \times n$阶非奇异矩阵, λ和$\mathbf{v}$为对应的特征向量, 即$\mathbf{Au} = \lambda\mathbf{v}$.
由于$\mathbf{A}^{-1}\mathbf{v} = \frac{1}{\lambda}\mathbf{v}$. 如果$\mathbf{A}$的特征值的顺序为$|\lambda_1| \geq |\lambda_2| \geq \cdots \geq |\lambda_n|$,
则$\mathbf{A}^{-1}$的特征值$\left|\frac{1}{\lambda_n}\right| \geq \left|\frac{1}{\lambda_{n-1}}\right| \geq \cdots \geq \left|\frac{1}{\lambda_1}\right|$.
因此, 若对矩阵$\mathbf{A}^{-1}$用幂法, 即可计算出$\mathbf{A}^{-1}$的最大特征值$\frac{1}{\lambda_n}$, 从而求得$\mathbf{A}$按模最小的特征值λ_n.

因为$\mathbf{A}^{-1}$的计算比较麻烦, 因此在实际运算时, 以求解方程组$\mathbf{A}\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)}$替代.

在本题中, 需要求解最大特征值. 根据圆盘定理, 矩阵$\mathbf{A}$的特征值有上界Ma. 设λ为$\mathbf{A}$的特征值, $k \in \mathbb{R}$. 则$(\mathbf{A} - k\mathbf{I})\,\mathbf{x} = (\lambda - k)\,\mathbf{x}$. 此时令$k = Ma + 2$, 则$(\mathbf{A} - (Ma+2)\,\mathbf{I})$的特征值均小于0. 设用反幂法求出$(\mathbf{A} - (Ma+2)\,\mathbf{I})^{-1}$的绝对值最小特征值为$|\lambda^*|$, 则$\mathbf{A}$的最大特征值为$Ma + 2 - \frac{1}{|\lambda^*|}$.

注: 圆盘定理:设$\mathbf{A}$是一个$n \times n$的矩阵, $\mathbb{R}_i$表示以a_{ii}为圆心, $\sum_{j=1,j\neq i}^{n} |a_{ij}|$的圆. 令

$$\mathbb{R}_i = \left\{ z \in \mathbb{C} \middle| \, |z - a_{ii}| \leq \sum_{j=1,j\neq i}^{n} |a_{ij}| \right\}$$

$\mathbf{A}$的特征值包含在$\bigcup_{i=1}^{n} \mathbb{R}_i$中.

对幂法进行改造, 反幂法计算的主要步骤如下:

1. 对$\mathbf{A}$进行三角形分解$\mathbf{A} = \mathbf{L}\mathbf{U}$
2. 做如下迭代:

$$\begin{cases} Let\ \mathbf{x}^{(0)} = \mathbf{y}^{(0)} \neq 0 \\ Solve: \mathbf{A}\mathbf{x}^{(k)} = \mathbf{y}^{(k-1)} \\ Then: \mathbf{y}^{(k)} = \dfrac{\mathbf{x}^{(k)}}{\left|\left|\mathbf{x}^{(k)}\right|\right|_{\infty}} \end{cases}$$

在反幂法中, 最终得到的$\mathbf{x}$即为所求特征向量.

Code :

```
clear;clc;
A = [4,-1,1;-1,3,-2;1,-2,3]; n = length(A); TOL = 1e-7;
Ma = zeros(1,n);
for i=1:n
  for j=1:n
    if i==j
      Ma(i) = Ma(i) + A(i,i);
    else
      Ma(i) = Ma(i) + abs(A(i,j));
    end
  end
end
change = max(Ma) + 2; A = A - change*eye(n);
L=eye(n,n); U=zeros(n,n);
for k=1:n
  for j=k:n
    U(k,j)=A(k,j)-sum(L(k,1:k-1).*U(1:k-1,j)');
  end
  for i=k+1:n
    L(i,k)=(A(i,k)-sum(L(i,1:k-1).*U(1:k-1,k)'))/U(k,k);
  end
end
x0 = [1;1;1]; MAXN = 100; mx0 = max(x0);y0 = x0/mx0;
```

```
            for CNT = 1:MAXN
              b = y0;
              X=zeros(1,3);Y=zeros(1,3);
              Y(1)=b(1);
              for i=2:n
                for j=1:i-1
                  b(i)=b(i)-L(i,j)*Y(j);
                end
                Y(i)=b(i);
              end
              X(n)=Y(n)/U(n,n);
              for i=(n-1):-1:1
                for j=n:-1:i+1
                  Y(i)=Y(i)-U(i,j)*X(j);
                end
                X(i)=Y(i)/U(i,i);
              end
              x = X';mx = max(x); y0 = x/mx;
              if abs(mx-mx0) < TOL
                fprintf('Answer %f Iteration time %d\n',change-1/mx,
    CNT)
                disp(x)
                return
              end
              x0 = x; mx0 = mx;
            end
            fprintf('No solution')
```

Answer Max Eigen Value: 6.000000, Iteration time 20
Eigen Vector: $\mathbf{x} = (-0.5000, 0.5000, -0.5000)^T$

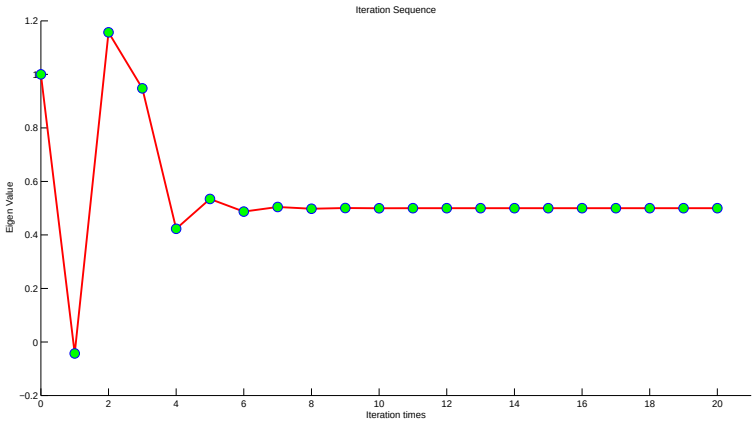

Figure 31: 反幂法特征值迭代序列

10 Conclusion

在本学期数值计算实验中, 我掌握了如下技能:

1. MATLAB的基本使用方式;
2. 二分法、牛顿法;
3. 多项式插值, 包括拉格朗日插值、Hermite插值等;
4. 根据插值多项式进行数值积分;
5. 求给定初始条件的常微分方程组;
6. 求解矩阵特征值;
7. 求解线性方程组;
8. 通过最小二乘进行多项式拟合;
9. 加深对数学分析、高等代数的理解;
10. Latex的使用;
11. 误差分析.

最后, 感谢刘保东老师, 在本学期的数值计算课程中, 让我对数学、计算有了新的认识.