信计182-41821219-张宇洋

数值分析上机作业

目录

[第一次上机作业 3](#_Toc40975185)

[1.1问题 3](#_Toc40975186)

[1.2算法公式与原理 3](#_Toc40975187)

[1.3程序运行结果 4](#_Toc40975188)

[1.4程序代码 5](#_Toc40975189)

[1.5问题和算法分析 5](#_Toc40975190)

[2.1问题 6](#_Toc40975191)

[2.2算法公式与原理 6](#_Toc40975192)

[2.3程序运行结果 6](#_Toc40975193)

[2.4程序代码 7](#_Toc40975194)

[2.5问题和算法分析 9](#_Toc40975195)

[3.1 问题 9](#_Toc40975196)

[3.2算法公式与原理 10](#_Toc40975197)

[3.3程序运行结果 11](#_Toc40975198)

[3.4程序代码 12](#_Toc40975199)

[3.5问题和算法分析 14](#_Toc40975200)

[第二次上机作业 14](#_Toc40975201)

[1.1问题 14](#_Toc40975202)

[1.2算法公式与原理 15](#_Toc40975203)

[1.3程序运行结果 15](#_Toc40975204)

[1.4程序代码 17](#_Toc40975205)

[1.5问题和算法分析 17](#_Toc40975206)

[2.1问题 17](#_Toc40975207)

[2.2算法公式与原理 18](#_Toc40975208)

[2.3程序运行结果 19](#_Toc40975209)

[2.4程序代码 21](#_Toc40975210)

[2.5问题和算法分析 21](#_Toc40975211)

[第三次上机作业 22](#_Toc40975212)

[1.1问题 22](#_Toc40975213)

[1.2算法公式与原理 22](#_Toc40975214)

[1.3程序运行结果 22](#_Toc40975215)

[1.4程序代码 22](#_Toc40975216)

[1.5问题和算法分析 22](#_Toc40975217)

[2.1问题 22](#_Toc40975218)

[2.2算法公式与原理 23](#_Toc40975219)

[2.3程序运行结果 23](#_Toc40975220)

[2.4程序代码 23](#_Toc40975221)

[2.5问题和算法分析 23](#_Toc40975222)

[3.1问题 23](#_Toc40975223)

[3.2算法公式与原理 23](#_Toc40975224)

[3.3程序运行结果 23](#_Toc40975225)

[3.4程序代码 23](#_Toc40975226)

[3.5问题和算法分析 23](#_Toc40975227)

# 第一次上机作业

## 1.1问题

分别用不动点迭代与Newton法求解方程的正根与负根.



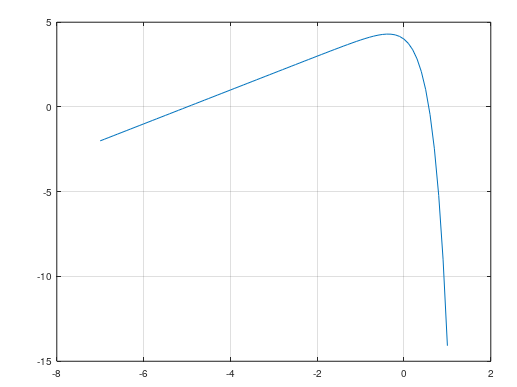
### 1.2算法公式与原理

不动点：



Newton法：



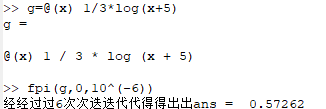


### 1.3程序运行结果

由图中可以得出x=-5,x=0为方程的近似解

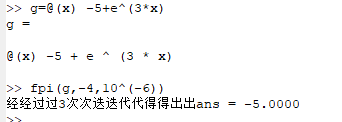
为了让方程收敛，不动点法计算x=0处时，



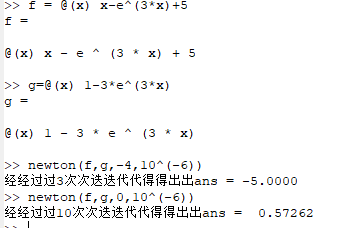


x=-5时，



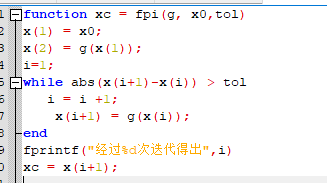


Newton法计算





### 1.4程序代码



### 1.5问题和算法分析

在使用不动点法进行迭代时要注意方程是否满足迭代条件，如果不满足需要对方程进行变形，使得其满足收敛条件并且对初始值不敏感。可以看出，Newton法与不动点法的收敛速度都较快。

## 2.1问题



1) 应用Newton法求函数的零点，

2）用求重根的方法求f(x) 的零点.

3）再用Steffensen’s method加速其收敛.

### 2.2算法公式与原理

Newton法：



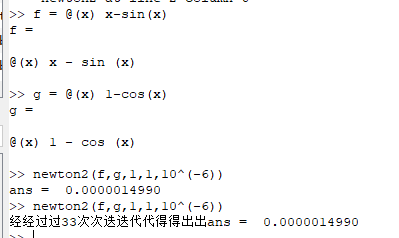


Steffensen方法



### 2.3程序运行结果

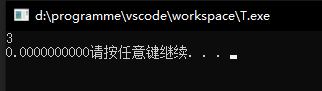
1．一般Newton法



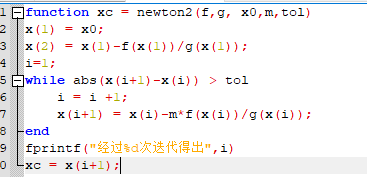
2.Newton的重根方法，计算得出此处为三重跟，所以取m=3

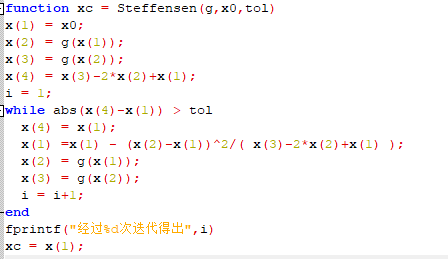


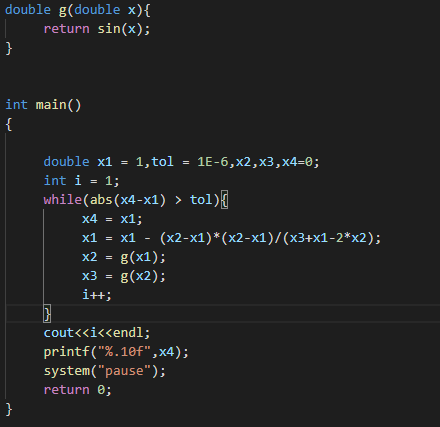




### 2.4程序代码





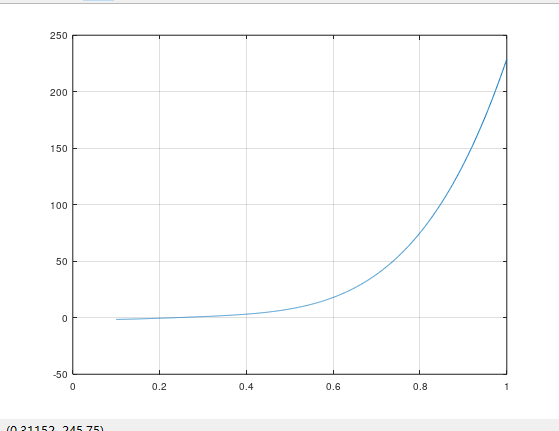
****

### 2.5问题和算法分析

当使用牛顿法进行重根的估计时效果不好，但是可以利用重根的牛顿法或者Steffensen方法进行计算,但是Steffensen方法会少到机器的限制，可能发挥不了本身算法的效果，本次换用cpp才得到想要的结果

## 3.1 问题





### 3.2算法公式与原理

1.Bisection



2.Newton



3.Scant method

当方程导数不容易求时用割线代替导数。



4. Method of False Position

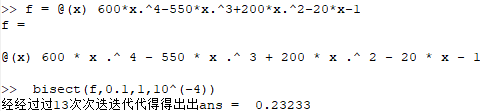


5.Muller’s method

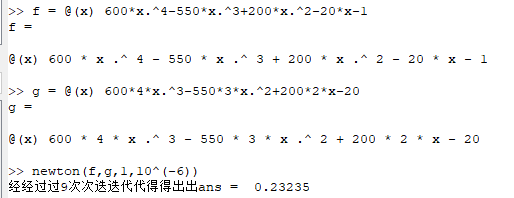
拓展算法

### 3.3程序运行结果

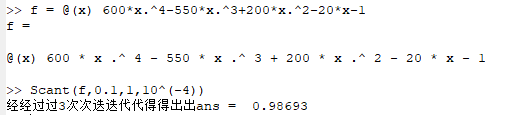
1.Bisection



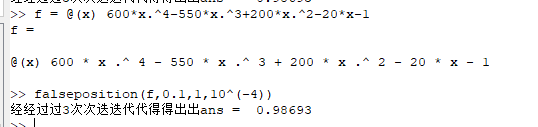
2.Newton



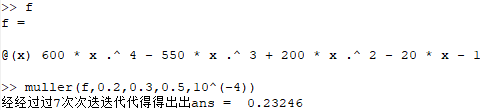
3.Scant method



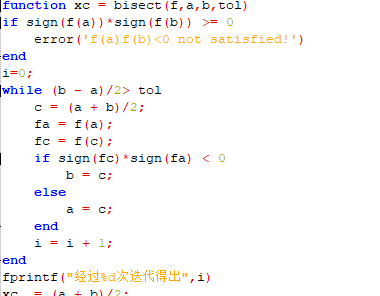
4. falseposition

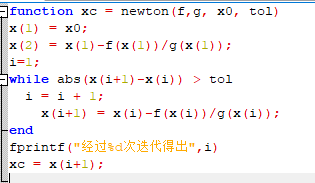


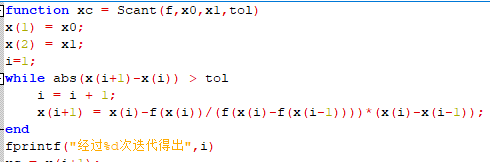
5. Muller’s method

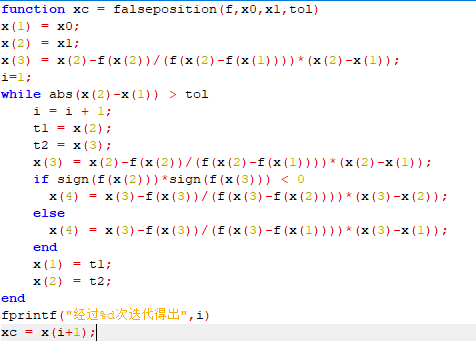


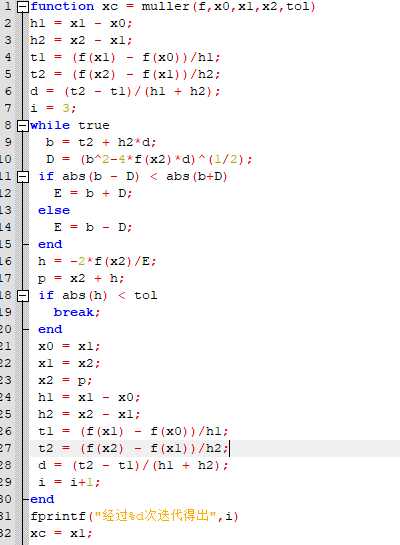
### 3.4程序代码











### 3.5问题和算法分析

# 第二次上机作业

## 1.1问题

**Use Romberg integration to compute approximations to**



**a. Determine *R*1*,*1, *R*2*,*1, *R*3*,*1, *R*4*,*1, and *R*5*,*1, and use these**

**approximations to predict the value of the integral.**

**b. Determine *R*2*,*2, *R*3*,*3, *R*4*,*4, and *R*5*,*5, and modify your**

**prediction.**

**c. Determine *R*6*,*1, *R*6*,*2, *R*6*,*3, *R*6*,*4, *R*6*,*5, and *R*6*,*6, and**

**modify your prediction.**

**d. Determine *R*7*,*7, *R*8*,*8, *R*9*,*9, and *R*10*,*10, and make a final**

**prediction.**

**e. Explain why this integral causes difficulty with Romberg**

**integration and how it can be reformulated to more easily**

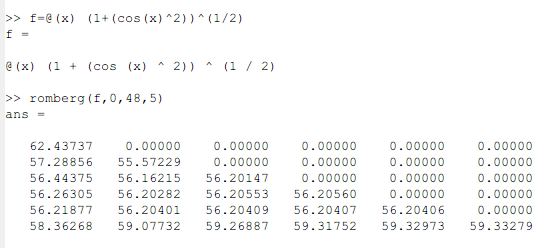
**determine an accurate approximation.**

### 1.2算法公式与原理

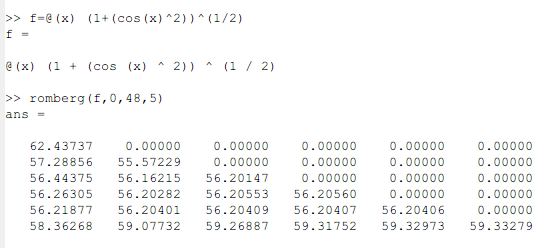


### 1.3程序运行结果

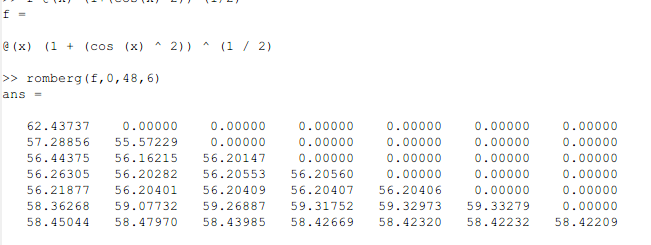
a. 



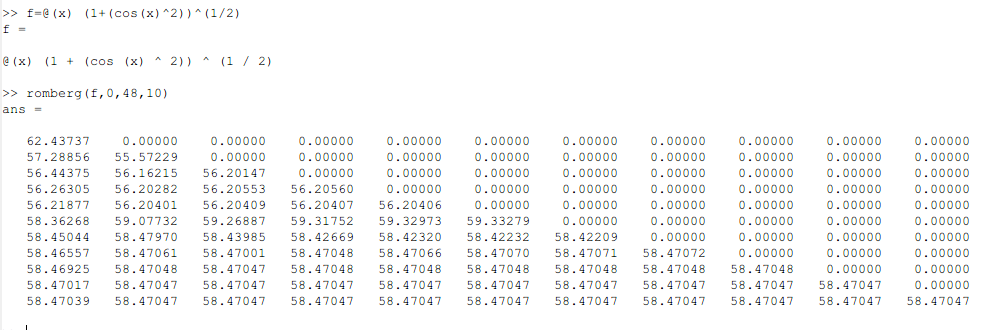
b. 



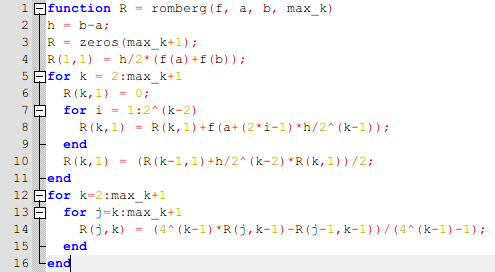
c. 



d. 



### 1.4程序代码



### 1.5问题和算法分析

## 2.1问题

**2.** 书(**8**版**265**页第**11**题）

**Given the initial-value problem**



**with exact solution *y*** = ***e^(***−***t)*** + ***t***

**Approximate *y(*5*)* with *h* = 0*.*2, *h* = 0*.*1, and *h* = 0*.*05.**

**a. Euler’s method; (P257)**

**b. Modified Euler’s method; (P277)**

**c. Runge-Kutta Method order four. (P278)**

**And analysis the error.**

### 2.2算法公式与原理

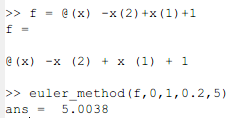


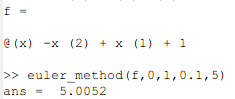


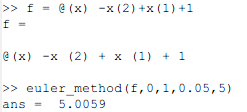


### 2.3程序运行结果

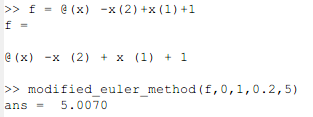
a.

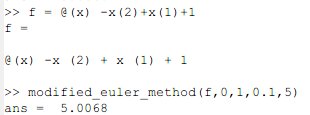


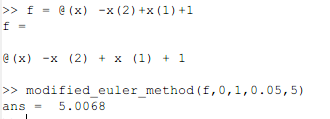




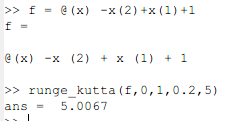
b.

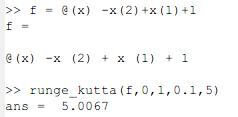


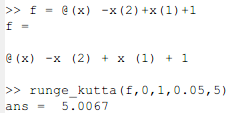




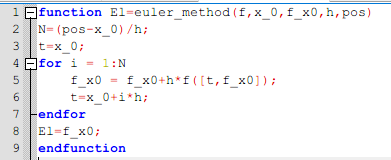
c.

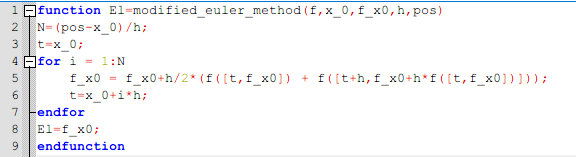


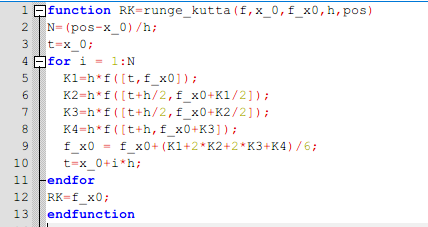




### 2.4程序代码







### 2.5问题和算法分析

# 第三次上机作业

## 1.1问题

**Using**

**1) Gaussian elimination**

**2) Gaussian elimination with partial pivoting.**

**3) Gaussian elimination with scaled partial pivoting.**

**and three-digit chopping arithmetic to solve**

**the following linear systems, and compare the**

**approximations to the actual solution.**

**1*.*19*x*1 + 2*.*11*x*2 − 100*x*3 + *x*4 = 1*.*12*,***

**14*.*2*x*1 − 0*.*122*x*2 + 12*.*2*x*3 − *x*4 = 3*.*44*,***

**100*x*2 − 99*.*9*x*3 + *x*4 = 2*.*15*,***

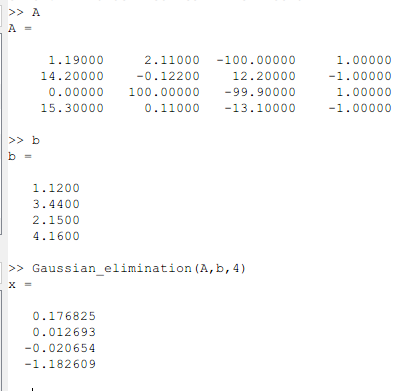
**15*.*3*x*1 + 0*.*110*x*2 − 13*.*1*x*3 − *x*4 = 4*.*16*.***

**Actual solution [0*.*176*,* 0*.*0126*,*−0*.*0206*,*−1*.*18].**

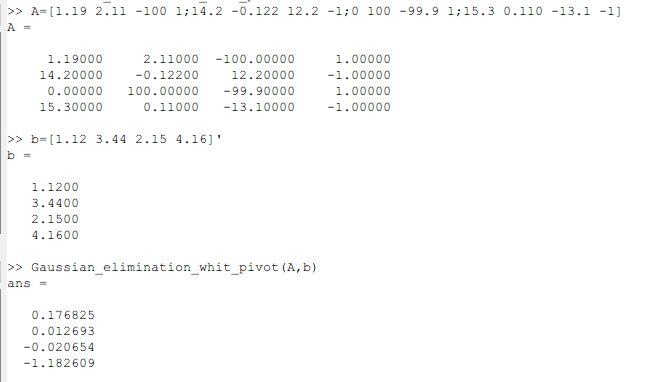
### 1.2算法公式与原理

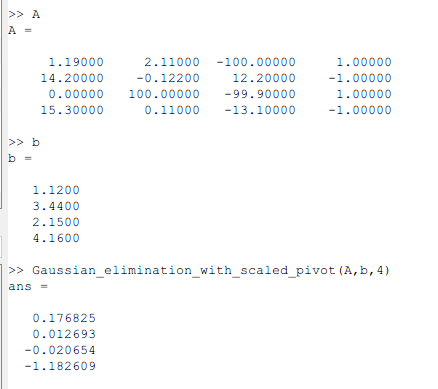
### 1.3程序运行结果

a.



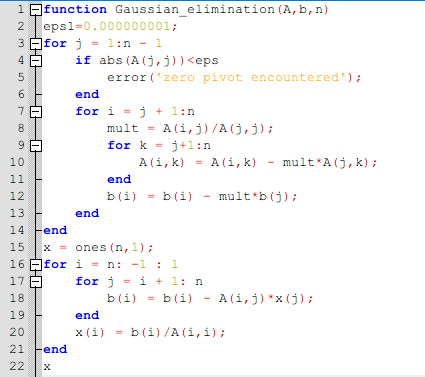
b.



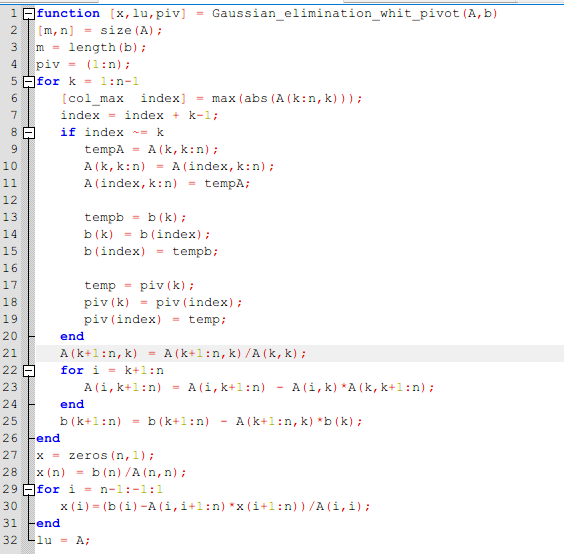


### 1.4程序代码

a.



b.



c.



### 1.5问题和算法分析

## 2.1问题

**Solve *A*x = b using the Crout factorization for tridiagonal systems.**

**Let *A* be the 10**×**10 tridiagonal matrix given by**

**Let b be the ten-dimensional column vector given by**

***aii* = 2*, ai,i+*1 = *ai,i−*1 = −1, for each *i* = 2*, . . . ,* 9,**

**and *a*11 = *a*10,10 = 2*, a*12 = *a*10,9 = −1.**

***b*1 = *b*10 = 1 and *bi* = 0, for each *i* = 2*,* 3*, . . . ,* 9.**

### 2.2算法公式与原理

### 2.3程序运行结果

### 2.4程序代码

### 2.5问题和算法分析

## 3.1问题

分别用**Jacobi, GS, SOR(ω=1.2)**方法求解方程组**Ax=b**，

**x0=[0 0 … 0]’**；**e=10^(-5)**



**and *bi* = 1*.*5*i* − 6, for each *i* = 1*,* 2*, . . . ,* 40.**

### 3.2算法公式与原理

### 3.3程序运行结果

### 3.4程序代码

### 3.5问题和算法分析