成绩

中国农业大学

课程论文

(2021-2022 学年春季学期)

论文题目：从零开始学 FORTRAN

课程名称：计算方法

任课教师：徐春晖

班　　级：工力 201

学　　号：2020310020119

姓　　名：张家瑞

从零开始
学 ***FORTRAN***

Learn FORTRAN
from scratch

作者・张家瑞
Author・Jia-Rui Zhang

中国农业大学理学院应用力学系工力 201
E-mail: jrzhang@cau.edu.cn

前言

数值分析也称计算数学，是数学科学的一个分支，它研究用计算机求解各种数学问题的数值计算方法及其理论与软件实现。它以数字计算机求解数学问题的理论和方法为研究对象，为计算数学的主体部分。随着计算机的发展，数值分析在处理一些实际问题中有着重要的作用，而 FORTRAN 正是解决这些问题时候的有力软件之一，接下来，让我们走进算法，走进 FORTRAN 吧！

最初，我对 FORTRAN 的使用意义有很大的疑惑，不就是一个计算机编程语言吗，并且比如 MATLAB 等数学软件既容易又方便，干嘛还要去学习 FORTRAN。但是现在我想明白了，MATLAB 固然强大，可是终究也只是一个工具，我们要做到知其然更要知其所以然，要看到软件背后的算法，知道设计的巧妙之处，人们为减少时间及存储成本，或减小误差而做出的一步步努力，这也正是人类智慧的结晶。我们这门课程的学习，也正是走进算法，亲自体验编程的乐趣。或许一个 FFT 指令在 MATLAB 中只需要敲几个字母就可以了，但是其中的变换技巧，还需细细品味。并且据我所知，徐老师的论文中的数据，都是使用 FORTRAN 来处理的，在精度方面有着很大的保障。

请注意，本文的主要目的不是为了讲解如何使用 FORTRAN，而是重在编程的过程，尤其是在面对问题时候的思考与解决方案。同理，在换一种编程语言后，我们依旧能够迅速上手，并实现我们所需要的算法。正如题目，从零我们便可学习，换种计算机语言岂不亦是如此？

本文的内容配套李清扬等主编的《数值分析》一书，建议配合使用。另，在本文中以 Compaq Visual Fortran | Vision 6.6 作为演示。

本文先从 FORTRAN 的简单使用出发，后由浅及深，介绍一些程序，在文中更多的是阐述个人对算法的思考。另，所有程序均为本人设计、编写。

当然，文中错误在所难免，恳请读者批评指正！

张家瑞

2022 年 4 月

于清华东路 17 号

目录

前 言 **2**

第一章 FORTRAN 的引入 **5**

1.1 新建文件 5

1.2 常用语法 6

第二章 隙中窥月——个人想法 **8**

2.1 BMI 的计算 8

2.2 二元一次方程的求解 9

2.3 水仙花数的判断 9

2.4 根值的计算 10

2.5 两种排序方法 12

2.6 平闰年的判断 13

2.7 导函数的三次探索 14

2.8 简单的积分计算 17

2.9 基数猜想的简单求证 19

2.10 体育成绩的计算 21

2.11 一些遗憾与小结 21

第三章 庭中望月——教材初步 **23**

3.1 秦九韶算法 23

3.2 拉格朗日插值 24

线性插值 24

抛物线插值 24

3.3 牛顿插值 25

牛顿插值多项式 25

差分形式的牛顿插值多项式 26

3.4 最小二乘拟合 27

3.5 数值积分 28

牛顿-柯特斯公式 28

复合求积公式 28

龙贝格求积公式 29

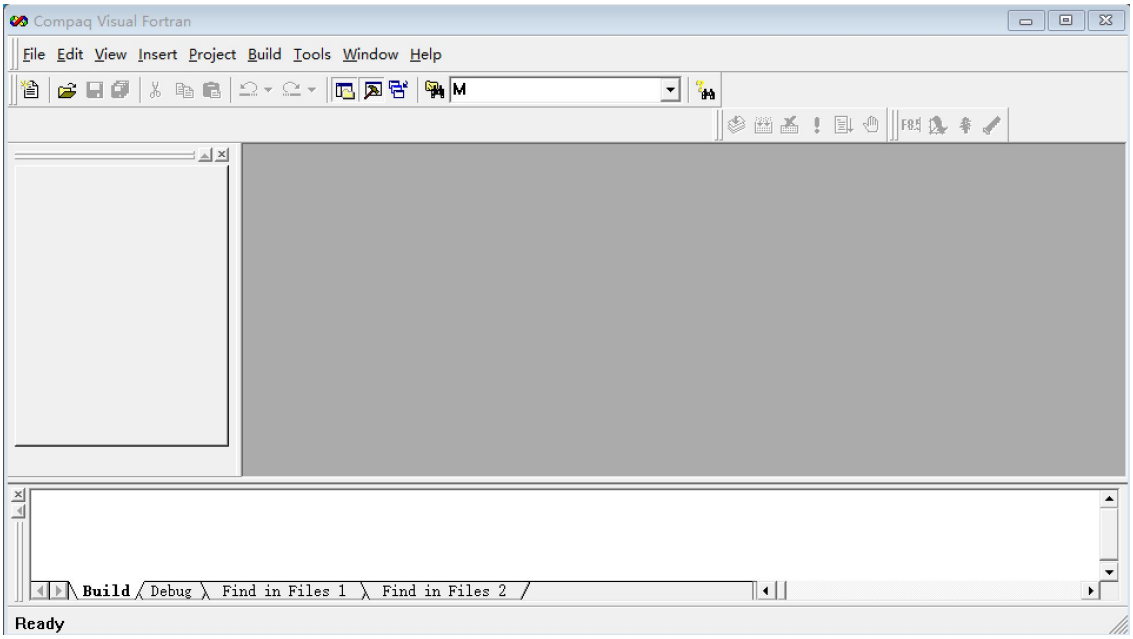
高斯求积公式.....31
3.6 非线性方程求解..... 32
二分法.....32
牛顿迭代法.....33
第四章 台上玩月——线性方程组.....34
4.1 高斯消去法..... 34
高斯消去法.....34
列主元消去法.....35
4.2 矩阵三角分解法..... 38
直接三角分解法.....38
追赶法.....40
4.3 迭代法..... 42
雅可比迭代法.....42
高斯-塞德尔迭代法.....43
总结与心得.....44

第一章 FORTRAN 的引入

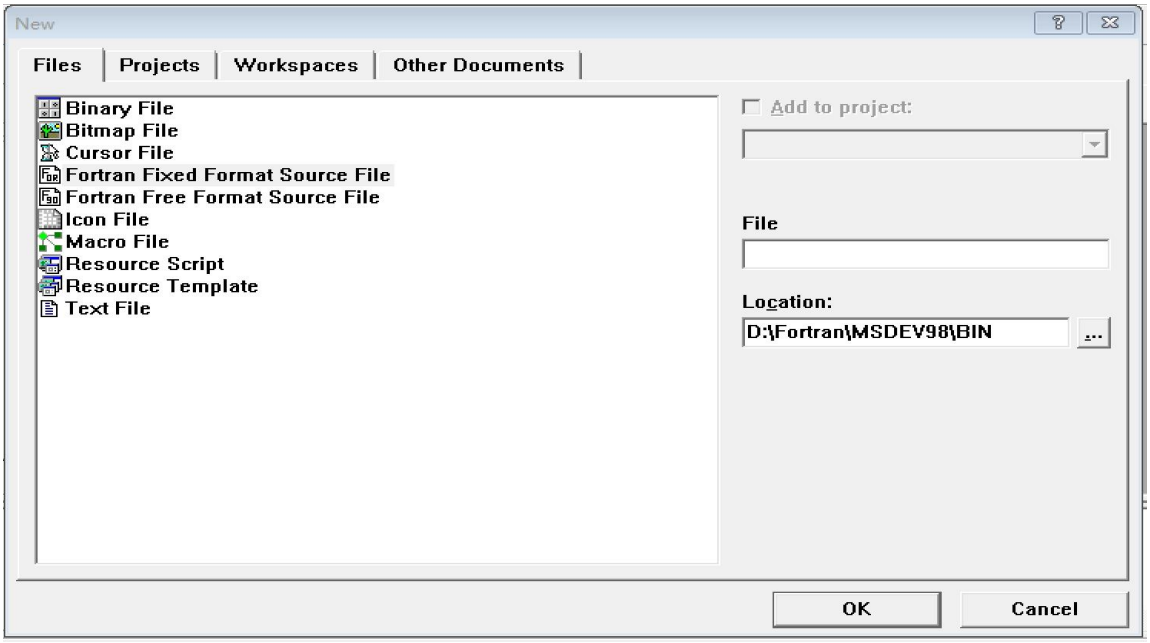
在这部分，将介绍关于软件的简单使用以及一些 FORTRAN 常用的语法规则。作为基础部分的学习。

1.1 新建文件

在打开软件后，我们会看到如下页面：



此时，我们需要点击左上角的“File”，点击第一个“New”之后，会出现如下页面：



本人使用的是“Files”中的“Fortran Fixed Format Source File”，当然其它的也可以。在右边的“Location”部分可以更改新建文件存储位置。

在新建文件后，便是我们的编程页面了，首先我们使用“Tab”键来使程序从页面中的绿标位置开始输入。

我们先从最简单的开始，来熟悉它的语法环境。

在程序中，使用字母的大小写均可，但是为了程序的整齐美观，因此我们在这里统一使用大写。如图所示，就是我们最初的起点：“Hello, World!”

```
PROGRAM EX1
WRITE(*,*) 'Hello, World!'
END
```

程序的开头与结尾分别使用的为“PROGRAM EX1”与“END”，而中间就是我们程序的主体部分，其中“WRITE”为输出函数、“READ”为输入函数。在需要输出某变量时，我们只需把变量名放在“WRITE(*,*)”后面即可，不需要再加引号，而文本内容需要如图所示，加上引号。

最后，我们需要把这些程序编译出来，如图所示，我们依次点击第 1、2、4 个便可以实现程序的运行，得到最终所需要的结果，即蹦出一个黑框。有时候需要输入数据，也正是在此时输入的。在此别的不再过多赘述，更多的内容将在后续编程中展示。

1.2 常用语法

在了解了基础的程序运行后，我们来了解一些语法。

判断语言：“IF”函数，常见用法为

```
IF() THEN
ELSE IF() THEN
ELSE THEN
END IF
```

在括号中写入所需的判断条件、“THEN”后写出所需执行的操作即可。

循环语言：“DO”函数，常见用法为

```
DO I=1,N
END DO
```

```
DO
IF() EXIT
END DO
```

两种方法均可，第一种为已知循环次数的循环，第二种为一直截断条件的循环。在第一种，两行直接输入命令；第二种中，“DO”后写入命令，括号中输入截断条件。

数组：要注意 FORTRAN 中数组为先列后行。使用“DIMENSION Z(4,3)”即为定义了一个 4 列 3 行的二维数组，名字为“Z”。

比大小指令：如果同时使用多种判断条件，需要以“.AND.”、“.OR.”等连接。注意这些命令中前后都有“.”。在 FORTRAN 中判断两数相等时，一般不使用“.EQ.”，而是两数只差“.LE.(10E-7)”即可，此时便可认为两数约等。

.GE.	Greater or Equal	大于等于
.GT.	Greater Than	大于
.LE.	Less or Equal	小于等于
.LT.	Less Than	小于
.EQ.	Equal	等于
.NQ.	Not Equal	不等于

子程序的应用：我们有时候为了使程序运用方面、整洁美观，此时运用子程序不失为一种良策，这样主程序的任务仅为：定义、输入、调用子程序、输出。这样既美观又便捷。那么我们仅需要使用“CALL”加上子程序的函数名和所用变量便可调用子程序，使用“SUBROUTINE”加上子程序的函数名和所用变量来定义子程序。具体的使用会展现在后文中，在此不做赘述。

除此之外，还有一些常用的内置函数：

ABS	求绝对值	ABS(A)
EXP	指数运算	EXP(A)
SIN	正弦计算	SIN(X)
LOG	自然对数计算	LOG(X)
INT	取整函数	INT(A)
MOD	求余函数	MOD(A1,A2)
MAX	求最大值	MAX(A1,A2,…)

在定义中，以“I~N”为首命名的变量默认定义为整型的，其余情况均为浮点数。我们需要注意，主程序、子程序及每个函数后面要有一个“END”对应。

第二章 隙中窥月——个人想法

在此部分中，我们将介绍一些简单的程序（主要是本人想出的一些问题），作为算法的初步入门介绍。

2.1 BMI 的计算

作为大学生，BMI 指数是我们尤为关心的数据之一，因此设计了一种来计算 BMI 值并输出健康状态的程序。

```
PROGRAM EX2 !计算BMI的程序
WRITE(*,*) 'Please input your height (m) and weight (kg)'
READ(*,*) H,W
BMI=W/H/H

IF(BMI.LT.18.5)THEN
WRITE(*,*) 'abnormal-ectomorph'
ELSE IF((BMI.GE.18.5).AND.(BMI.LE.25))THEN
WRITE(*,*) 'normal'
ELSE IF(BMI.GT.25)THEN
WRITE(*,*) 'abnormal-endomorph'
END IF

WRITE(*,*) BMI
END
```

如图所示，这个算法的核心是选择程序：首先通过公式计算出 BMI，后运用选择程序进行判断，最终输出结果。以我本人的数据为例，图示变为输出结果。

```
 Please input your height (m) and weight (kg)
1.71 55
 normal
   18.80921
Press any key to continue
```

家瑞点评：这个程序我们可以在运算速度上进行改进。当处理的数据较多时，我们可以通过基于往年数据，统计出受众群体中 BMI 数值处于哪种状态的人偏多，进而按从多到少的顺序排列判断区间。除此之外，我们可以将判断部分另设置为一个子程序，由于该程序较简单，在此就不展示了。

2.2 二元一次方程的求解

二元一次方程组的求解时。输入数据“A、B、C”分别为方程的三个系数，首先对输入方程的系数进行一个判断，在确定为二元一次方程之后进行求解。本程序还考虑了复根的存在，输出了两个解的实部与虚部。其中“B*B”也可以写作“B**2”，类似的 N 次方可以写作“B**N”。

```
PROGRAM EX3 !二元一次方程求解
READ(*,*) A,B,C

IF(A.EQ.0)THEN
WRITE(*,*) 'This is not a unary quadratic equation'

ELSE
D2=B*B-4*A*C
IF(D2.LT.0) THEN
D=SQRT(-D2)
XRE=(-B)/(2*A)
X1IM=(D)/(2*A)
X2IM=(-D)/(2*A)
WRITE(*,*) 'Re(X1)=',XRE,'Im(X1)=',X1IM
WRITE(*,*) 'Re(X2)=',XRE,'Im(X2)=',X2IM
ELSE
D=SQRT(D2)
X1=(-B+D)/(2*A)
X2=(-B-D)/(2*A)
WRITE(*,*) X1,X2

END IF
END IF
END
```

我们以 x^2+2x+3 为例，输出结果：

```
1 2 3
 Re(X1)=  -1.000000     Im(X1)=   1.414214
 Re(X2)=  -1.000000     Im(X2)=  -1.414214
Press any key to continue
```

家瑞点评：此程序中“X2IM”计算与“X1IM”计算出现了重复，可以直接将输出位置改为“’Im(X2)=-’,X1IM”，进而省略了“X2IM”的计算。除此之外，该程序与周天奕学长的相比，在判断及计算部分要更为简洁，加快了运算速度。

2.3 水仙花数的判断

水仙花数也被称为超完全数字不变数，是指一个三位数，它的每个位上的数字的三次幂

之和等于它本身。此程序的意义在于计算出多位数的各位数值。在以下程序中是使用整型变量的特点进行计算的，通过运算省去了其小数部分。定义的“I1、I2、I3”即为整型数（前面有所提及）。

```
PROGRAM EX4 !判断是否为水仙花数
WRITE (*,*) 'Please input a three-digit number'
READ(*,*) K
I1=K/100
I2=(K-I1*100)/10
I3=K-I1*100-I2*10
K1=I1**3+I2**3+I3**3
IF(K1.EQ.K) THEN
WRITE (*,*) 'This is a narcissistic number'
ELSE
WRITE (*,*) 'This is not a narcissistic number'
END IF
END
```

我们以已知水仙花数 153 为例，运行结果如下：

```
 Please input a three-digit number
153
 This is a narcissistic number
Press any key to continue
```

家瑞点评：在这个程序编写之时，本人未注意到整型变量的设置，使用“I”开头作为变量名是歪打正着了，在进一步了解之后才知此原因。因此，如果不考虑变量类型的情况下，我们可以借助于“MOD”函数，比如“K1=(K-MOD(K,100))/100”，这样可以求出 K 的百位上的数字。

2.4 根值的计算

尽管根值的计算我们可以通过内部函数解决，但它是数值迭代的一个简单的应用，而数值迭代又牵扯到误差的截断等，因此我们有必要在这里进行展示。并且这个程序运用了子程序，使得程序看起来更加简洁。

```
PROGRAM EX3 !计算根值
READ(*,*) A

IF(A.EQ.0)THEN
WRITE(*,*) '0','Simulation'
WRITE(*,*) '0','Truth'
```

```
ELSE

IF(A.LT.0) THEN
B=-A
C=B
CALL GEN(B,C,I)
WRITE(*,*) C,'i','Simulation'
WRITE(*,*) SQRT(B),'i','Truth'
WRITE(*,*) I,'Iteration'

ELSE
B=A
C=B
CALL GEN(B,C,I)
WRITE(*,*) C,'Simulation'
WRITE(*,*) SQRT(B),'Truth'
WRITE(*,*) I,'Iteration'

END IF
END IF
END


SUBROUTINE GEN(B,C,I)
I=1
DO
D=(C+B/C)/2
C=D
IF ((C-SQRT(B)).LE.0.00001) EXIT
I=I+1
END DO
END
```

在这个程序中，“GEN”程序是分别用迭代法与内部函数来计算根值的，其中“D”的运算原理便是迭代公式。而迭代的阶段条件便是用迭代结果与内置函数结果相对比，当误差或相对误差达到设定的精度时，便可截断。该程序不仅输出了迭代解与内置函数解，还输出了总共的迭代次数。我们以 18 为例，以下为所输出的结果：

```
18
   4.242642     Simulation
   4.242640     Truth
          5 Iteration
Press any key to continue
```

家瑞点评：在这个程序设计之初，更多考虑的时迭代的本质，以及截断的误差分析。此外，该程序还可以与前面的二元一次方程求根相结合。

2.5 两种排序方法

排序问题对于数值计算有着重要的应用，能够更快的排好顺序对于我们有着很重要的意义。在这里，我们介绍两种排序方法：起泡排序法与选择排序法。

对于起泡排序法，依次比较相邻的两个数，将小数放在前面，大数放在后面。即在第一趟：首先比较第 1 个和第 2 个数，将小数放前，大数放后。然后比较第 2 个数和第 3 个数，将小数放前，大数放后，如此继续，直至比较最后两个数，将小数放前，大数放后。至此第一趟结束，将最大的数放到了最后。在第二趟：将小数放前，大数放后，一直比较到倒数第二个数（倒数第一的位置上已经是最大的），第二趟结束，在倒数第二的位置上得到一个新的最大数（其实在整个数列中是第二大的数）。如此下去，重复以上过程，直至最终完成排序。

```
PROGRAM EX5 !数组排序-起泡排序法
DIMENSION A(5)
WRITE(*,*) 'Please input A'
READ(*,*) A
N=5
CALL SORT(A,N)
WRITE (*,*) A
END


SUBROUTINE SORT(A,N)
DIMENSION A(N)
DO J=1,N-1
DO I=1,N-J
IF (A(I).GT.A(I+1)) THEN
B=A(I)
A(I)=A(I+1)
A(I+1)=B
END IF
END DO
END DO
END
```

在定义的地方可以设置要排列的数的个数，即设置数组中包含个数的大小。我们以五个数为例，展现出排序结果：

```
 Please input A
3 4 2 1 5
   1.000000        2.000000        3.000000        4.000000        5.000000
Press any key to continue
```

对于选择排序法，它的工作原理是每一次从待排序的数据元素中选出最小的一个元素，与

序列的第一个元素交换位置，然后，再从剩余未排序元素中继续寻找最小元素，再与其中的未排序序列中的第一个元素交换位置。以此类推，直到全部待排序的数据元素排完。

```
PROGRAM EX5 !数组排序-选择排序法
DIMENSION A(5)
WRITE(*,*) 'Please input A'
READ(*,*) A
N=5
CALL SORT(A,N)
WRITE (*,*) A
END


SUBROUTINE SORT(A,N)
DIMENSION A(N)
DO J=1,N-1
DO I=J,N
K=J
MIN=A(J)
IF (A(I).LT.MIN) THEN
K=I
END IF
B=A(J)
A(J)=A(K)
A(K)=B
END DO
END DO
END
```

此处在计算时需要修改的地方与前者相同，我们以五个数为例，展现出排序结果：

```
 Please input A
2 3 6 9 2
   2.000000       2.000000       3.000000       6.000000       9.000000
Press any key to continue
```

家瑞点评：这两种排序方法均不为最快速的排序方法，但它们具有稳定性好等特点，这两种方法运用了嵌套循环函数，对于一些算法的实现也有着很大的借鉴意义。对于 n 个数据元素的排序，起泡排序法需要比较(n*(n-1))/2 次，交换(n*(n-1))/2 次；选择排序法比较(n*(n-1))/2 次，交换(n-1)次。当然还有诸如插入排序、希尔排序、快速排序等方法，在此不做展示。

2.6 平闰年的判断

平润年的判断在我们日常生活中有着重要的作用，闰年每四年一次，能被 400 整除或能被 4 整除而不能被 100 整除的即为闰年，闰年不仅比平年多了一天，并且每届的夏季奥运会也将在闰年举办（2021 年东京奥运会除外），因此这个简单的程序可以帮助我们判断平年、闰年。

```
PROGRAM EX1
WRITE(*,*) 'Please input the year'
READ(*,*) I

IF ((MOD(I,400)).EQ.0) THEN
WRITE(*,*) 'It is a leap year'
ELSE IF((MOD(I,4)).EQ.0.AND.(MOD(I,100)).NE.0) THEN
WRITE(*,*) 'It is a leap year'
ELSE
WRITE(*,*) 'It is a common year'
END IF

END
```

我们通过简单的判断函数实现了分类，如程序中所示，当同时需要两个判断条件时，我们需要使用“.AND.”来连接，并且使用了“MOD”函数作为判断的条件。我们以今年为例，运行结果如下：

```
 Please input the year
2022
 It is a common year
Press any key to continue
```

家瑞点评：该程序重在函数的运用，并通过简单的选择程序进行判断，实现起来也较为简单，但可以在选择结构上进行算法的优化。

2.7 导函数的三次探索

导函数对于处理数学问题有着很重要的意义，在此设想了构建求导函数的程序，由浅及深分成了三个程序，在此将逐一介绍本人对程序的思考。

```
PROGRAM EX1 !多项式求导
DIMENSION Z(9999) !此处设置需要的间隔N-1
DIMENSION G(9999) !此处设置需要的间隔N-1
WRITE(*,*) 'A*x^3+B*x^2+C*x+D in (E,F) for N'
READ(*,*) A,B,C,D,E,F,N
CALL QIUDAO(A,B,C,D,E,F,N,Z)
CALL TRUE(A,B,C,D,E,F,N,G)

WRITE(*,*) '模拟          ','真实          ','误差'
DO I=1,N-1
WRITE(*,*) Z(I),G(I),(ABS(G(I)-Z(I))/G(I))
END DO
END
```

```
SUBROUTINE QIUDAO(A,B,C,D,E,F,N,Z)
DIMENSION Z(9999) !此处设置需要的间隔N-1
H=(F-E)/N
X=E
Y=E+H
DO I=1,N-1
Z(I)=(A*Y*Y*Y+B*Y*Y+C*Y+D-(A*X*X*X+B*X*X+C*X+D))/H
X=X+H
Y=Y+H
END DO
END

SUBROUTINE TRUE(A,B,C,D,E,F,N,G)
DIMENSION G(9999) !此处设置需要的间隔N-1
H=(F-E)/N
X=E
DO I=1,N-1
G(I)=3*A*X*X+2*B*X+C
X=X+H
END DO
END
```

首先是从最简单的多项式求导入手，本人设计了一个可用于求导三次多项式的程序，通过调整间隔来调整精度，并将数值求解与真实结果进行了对比，求出了他们的相对误差，整体思路很简单。最后可以把数据复制到作图软件中，便可画出导函数曲线。在此截取了一种情况下的部分结果作为示例，如下：

```
 A*x^3+B*x^2+C*x+D in (E,F) for N
1 1 1 1 0 10 40
 模拟          真实          误差
   1.312500        1.000000        0.3125000
   2.187500        1.687500        0.2962963
   3.437500        2.750000        0.2500000
   5.062500        4.187500        0.2089552
   7.062500        6.000000        0.1770833
   9.437500        8.187500        0.1526718
   12.18750        10.75000        0.1337209
   15.31250        13.68750        0.1187215
   18.81250        17.00000        0.1066176
   22.68750        20.68750        9.6676737E-02
   26.93750        24.75000        8.8383839E-02
   31.56250        29.18750        8.1370451E-02
   36.56250        34.00000        7.5367644E-02
   41.93750        39.18750        7.0175439E-02
   47.68750        44.75000        6.5642461E-02
   53.81250        50.68750        6.1652280E-02
   60.31250        57.00000        5.8114037E-02
   67.18750        63.68750        5.4955840E-02
   74.43750        70.75000        5.2120142E-02
   82.06250        78.18750        4.9560353E-02
   90.06250        86.00000        4.7238372E-02
   98.43750        94.18750        4.5122761E-02
```

之后，为了增加程序的普适性，使程序可以适用于更多的函数，在此基础上我进行了改进，具体体现为：为了使其适用于更多的函数，我们需要在程序内部直接设置函数，而设置函数

```
PROGRAM EX1 !任意函数求导
DIMENSION Z(99999)
WRITE(*,*) 'the function in (E,F) for N'
READ(*,*) E,F,N
CALL DIFF(E,F,N,Z)
DO I=1,N-1
WRITE(*,*) Z(I)
END DO
END

SUBROUTINE FUNC(X,F)
F = X*X*X*X + EXP(X) + LOG(X*X) !在此处设置待求导函数F(X)
END

SUBROUTINE DIFF(E,F,N,Z)
DIMENSION Z(99999)
H=(F-E)/N
X=E
Y=E+H
DO I=1,N-1
CALL FUNC(X,XX)
CALL FUNC(Y,YY) !此处还可以继续优化
Z(I)= (YY-XX)/H
X=X+H
Y=Y+H
END DO
END
```

的时候，为了尽可能减少需要设置的次数，因此我们构造了一个子程序，用于专门计算函数值，只用在其中的子程序做修改便可以了，这样也是子程序中又嵌套了一个子程序。我们任取了一个函数后作为实验，程序运行结果如下：

```
 the function in (E,F) for N
1 10 50
   10.03277
   13.37780
   17.89145
   23.71483
   31.00222
   39.91802
   50.63591
   63.33891
   78.22047
   95.48581
   115.3539
   138.0600
   163.8587
   193.0273
   225.8709
   262.7268
```

尽管这个程序可以实现结果，但如程序中注释显示，在计算时出现了重复计算的现象，因此本人对其子程序进行了改进：

```
SUBROUTINE DIFF(E,F,N,Z)
DIMENSION Z(99999)
H=(F-E)/N
X=E
Y=E+H
CALL FUNC(X,XX)
CALL FUNC(Y,YY)
Z(1)= (YY-XX)/H
Y=Y+H
DO I=2,N-1
XX=YY
CALL FUNC(Y,YY)
Z(I)= (YY-XX)/H
Y=Y+H
END DO
END
```

在这种形式下，通过一个简单的修正，运用导数的定义，在计算下一个导数值时，继续沿用前面数据，便使得在计算函数值的部分，所用时间减少了将近一半。

家瑞点评：这两次改进，虽然很简单，但是它们体现出了为了增加程序的普适性及速度而做出的努力，在面对想要实现的程序时，我们需要如何做来达到目的的实现。当然，在这个基础上还有很大的改进空间，还有很多的灵活运用的地方，比如《数值分析》一书中的秦九韶算法计算多项式的导数，以及数值微分部分，这些内容将会在后面的地方提及。

2.8 简单的积分计算

在讨论完微分问题后，我们也要提及积分。在编写这个积分程序时，仅仅是使用个人的想法进行编写的，与教材中的算法肯定是没办法比的，在此仅供参考。

```
PROGRAM EX3 !计算积分
WRITE(*,*) 'A*x^3+B*x^2+C*x+D in (E,F) for N'
READ(*,*) A,B,C,D,E,F,N
Y=0
Z=0
W=0
CALL LEFT(A,B,C,D,E,F,N,Y)
CALL RIGHT(A,B,C,D,E,F,N,Z)
CALL MIDDLE(A,B,C,D,E,F,N,G)
CALL SIMPSON(A,B,C,D,E,F,N,W)
WRITE(*,*) Y,'LEFT'
WRITE(*,*) Z,'RIGHT'
WRITE(*,*) G,'MIDDLE'
WRITE(*,*) W,'SIMPSON'
END
```

```
SUBROUTINE LEFT(A,B,C,D,E,F,N,Y)
DX=(F-E)/N
DO I=1,N
X=E+DX*(I-1)
XI=A*X*X*X+B*X*X+C*X+D
Y=Y+XI*DX
END DO
END

SUBROUTINE RIGHT(A,B,C,D,E,F,N,Z)
DX=(F-E)/N
DO I=1,N
X=E+DX*I
XI=A*X*X*X+B*X*X+C*X+D
Z=Z+XI*DX
END DO
END
SUBROUTINE MIDDLE(A,B,C,D,E,F,N,G)
DX=(F-E)/N
EX=(F-E)/N/2
DO I=1,N
X=E+DX*I-EX
XI=A*X*X*X+B*X*X+C*X+D
G=G+XI*DX
END DO
END

SUBROUTINE SIMPSON(A,B,C,D,E,F,N,W)
DX=(F-E)/6
X=E
X1=A*X*X*X+B*X*X+C*X+D
X=F
X2=A*X*X*X+B*X*X+C*X+D
X=(E+F)/2
X3=A*X*X*X+B*X*X+C*X+D
W=(X1+X2+4*X3)*DX
END
```

在这个程序中，我们使用是针对于三次方程进行的，将待积分区间分为很多份后，分别使用小区间上的左函数值、右函数值、中间函数值进行积分，并使用了辛普森公式作为积分结果对比。由此可见，选取不同的积分方法，之间的误差相差也较大，但可以通过设置 N 来实现精度的最大化。

```
 A*x^3+B*x^2+C*x+D in (E,F) for N
1 1 1 1 0 10 100
   2838.100      LEFT
   2949.100      RIGHT
   2893.200      MIDDLE
   2893.333      SIMPSON
Press any key to continue
```

家瑞点评：在这个程序中，分别使用了四种积分方法作为对比，并体现了算法求解积分时候最简单的思想，即微分。但是针对多项式，我们可以选取较少的点来实现较高的精度，有牛顿-柯特斯公式等，也有许多其它的方法来求解积分问题，具体的将会在后续的章节中展示。

2.9 基数猜想的简单求证

基数猜想是由本人与侯同学在高中时候一起发现的（略显幼稚），关于它的证明一直还存在理论上的问题，具体内容可见文章《基数定义的引入及其性质的研究》（仅申请著作权，尚未发表，文章可咨询本人，可能其本身的证明毫无意义，或许早已被他人证明而我们不知道，如果读者有想法，欢迎与本人进行交流）。在此使用循环程序进行简单的证明。

基数猜想的基础表示为：形如 100a+10b+c 的三位数（也可以为任意位数的，在此仅为举例），则 a+b+c 为其初基数，若 a+b+c 能继续表示为 10d+e 的形式，则对其继续对 a+b+c 取基数，称为二基数，以此类推。直至取出的 n 基数大于 0 小于 10，称其为终基数，简称基数，表示为 G。而原本数字的加减乘除的运算与其对应的基数的运算保持一致性，例如：对于 42 与 31，42 的基数为 6，31 的基数为 4，而 42×31=1302,1302 的基数为 6，而 6×4=24，24 的基数也为 6。

我们以 9 以内的数字的乘法计算做验证，设计程序如下：

```
PROGRAM EX1 !基数猜想的验证

DO I=1,9
DO J=I,9
IZZ=I
JZZ=J
IA=I*J
CALL JISHU(IA,IAZ)
CALL JISHU(I,IZ)
CALL JISHU(J,JZ)
IAZZ=IZ*JZ
CALL JISHU(IAZZ,IAZZZ)
IF(IAZZZ.EQ.IAZ) WRITE(*,*) IZZ,JZZ,'are success'
IF(IAZZZ.NE.IAZ) EXIT
END DO
END DO

END
```

该程序分为主程序与子程序，此处展示的为主程序，仅仅是进行一个循环运行，将需要检验的数引入到子程序中，计算出它们的基数并作比较。但由于显示等不明原因，因此在这里只进行了 9 以内的验证。

以下为计算基数的子程序：

```
SUBROUTINE JISHU(III,JJJ)
DIMENSION B(50)
DO
JJ=1
KK=1
JJJ=0

DO
IIJ=III/10
B(JJ)=III-IIJ*10
III=IIJ
JJ=JJ+1
IF(B(JJ-1).EQ.0) EXIT
END DO

DO KK=1,JJ-1
JJJ=JJJ+B(KK)
END DO
III=JJJ
IF(JJJ.LT.10) EXIT
END DO
END
```

在该程序中，对于多位数的截断部分存在问题，比如 200 的基数，其计算结果会显示为 0，因此这个地方仍需要改进，目前的想法为把数字从前到后进行加和，但事实上，此处的程序缺陷对于基数猜想的验证并无太大影响。

以下为程序运行的部分结果：

```
           6           6 are success
           6           7 are success
           6           8 are success
           6           9 are success
           7           7 are success
           7           8 are success
           7           9 are success
           8           8 are success
           8           9 are success
           9           9 are success
Press any key to continue
```

家瑞点评：此程序验证了基数猜想，主要问题体现在结果的显示以及对于以 0 结尾的数字的基数运算上面，该程序的重点在于子程序中计算基数的部分，所运用的截断手法等需要仔细选择。总之，该程序仍有需要修正的地方，如果读者有好的建议也请联系本人。但是对于基本的验证已经是足够了。

2.10 体育成绩的计算

在接近这一章的最后部分，我将介绍一个很为实用的程序。我们大学生对于我们的体测成绩很为关心，当体测成绩不及格时，将会面临一系列的“灾难”。这里将介绍男生的体测成绩计算器，输出最终的体测成绩。由于程序过长，在此处我们将展示主程序部分：

```
PROGRAM EX1
WRITE(*,*) 'Please input your: 50米、坐位体前屈、身高、体重、'
WRITE(*,*) '肺活量、立定跳远、引体向上、1000米'
READ(*,*) AFIVE,AZUO,AHIGHT,AWEIGHT,ALUNG,ALI,AYIN,AONE
SOCCER=0
AFIVE=0
SFIVE=0
CALL FIVE(AFIVE,SFIVE)
CALL ZUO(AZUO,SZUO)
CALL BMI(AHIGHT,AWEIGHT,SBMI)
CALL LUNG(ALUNG,SLUNG)
CALL LI(ALI,SLI)
CALL YIN(AYIN,SYIN)
CALL ONE(AONE,SONE)

SOCCER=SFIVE*0.2+SZUO*0.1+SBMI*0.15
SOCCER=SOCCER+SLUNG*0.15+SLI*0.1+SYIN*0.1+SONE*0.2
WRITE(*,*) SOCCER

END
```

在这里，以本人大一下学期体测成绩为例进行展示：

```
 Please input your: 50米、坐位体前屈、身高、体重、
 肺活量、立定跳远、引体向上、1000米
7.5 13.8 169.9 51.9 4146 193 6 3.42
   70.00000
Press any key to continue
```

此程序可方便我们大学生进行体测成绩计算，对于我们的学习生活有着重要的帮助。

家瑞点评：此程序的难点在于繁琐，需要大量的输入数据，现实意义较强，但是仍有许多可以改进的地方。例如，我们可以对成绩是否及格进行一个判断，然后对比这几项体测的成绩，当提升哪一项时可以最高效率的提升体测总成绩，进而为我们的体育训练提供具有指导性的建议。

2.11 一些遗憾与小结

在此也借助这个机会对前面部分做一个小结吧。在学习这门课之初，我的想法是利用

FORTRAN 来实现孤子数值解的迭代求解过程，这样可以将课外的内容与课内相结合，但是这个想法很容易，做起来终究还是很难得，因为要利用到 FFT 等算法，以及改进平方算子法等内容，在 FORTRAN 中较难实现，并且处理数据量较大。但是这个过程也并非毫无意义，使我也去思考其本质的意义，例如我课外部分，最初只知道 FFT 三个字母便可以在 MTLAB 中实现快速傅里叶变换，但其背后的过程一无所知，而我们要想编出 FORTRAN 程序，我们就需要去考虑其背后的机制，因此对于我对算法的领悟等方面有着很好的意义。但尽管难以实现，我还是会在以后的时间里，尽力尝试。也如前文中有的地方所说，有些程序是不完美的，还需要继续改进。

除此之外，一些简单的程序就没有在前文中列举，重点是要找一些具有算法意义的问题进行求解吧，因此在后面两章，我对课本上的主要程序进行了编写。

第三章 庭中望月——中级程序

3.1 秦九韶算法

与前文中的 2.7 节相对应，在这里我们从最简单的秦九韶算法开始。该算法在计算多项式时更为快捷，避免了重复的运算，并且运用循环计算，很容易的计算出了多项式函数在一点处导数的精确值。在此我们以四阶多项式为例，当更高阶多项式时，我们只需要在程序中进行简单的修改即可。

```
PROGRAM EX1 !四阶多项式求解
DIMENSION A(5)
DIMENSION B(5)
DIMENSION C(5)
WRITE(*,*) '请从高到低阶依次输入四阶多项式系数及X取值'
DO I=1,5
READ(*,*) A(I)
END DO
READ(*,*) X
B(1)=A(1)
DO I=2,5
B(I)=B(I-1)*X+A(I)
END DO
C(1)=B(1)
DO I=2,4
C(I)=C(I-1)*X+B(I)
END DO
WRITE(*,*) B(5),C(4)
END
```

输出结果为在 X 处的多项式函数值及导数值，我们取书本中 P14 例 11 进行验证，输出结果如下：

```
 请从高到低阶依次输入四阶多项式系数及X取值
2
0
-3
3
-4
-2
   10.00000      -49.00000
Press any key to continue
```

家瑞点评：作为计算问题的简化，此程序对于我们很有借鉴意义，当处理数据较多时，我们便可以使用这种方法，这也是算法中的一个重要的原则。但是图中程序并不完美，真正秦九韶算法只需用到 N+2 个存储单元，在循环部分用两个变量即可，因为我们此处想要求出导数，所以便引入了数组。

3.2 拉格朗日插值

线性插值

对于有限已知点，我们可以通过插值方法来近似模拟推测出其它无穷多个点。插值在数值方法中有着很大的作用，在有限元中也应用十分广泛，据本人所知，例如一个房子，我们要想存储它的数据，不可能把每一个点都包含其中，这样所需要的空间是非常大的，我们可以存储有限点，然后利用插值等方法计算出其它的点，因此我们有必要介绍插值。首先是最简单的线性插值。

```
PROGRAM EX1
WRITE(*,*) '请依次输入(x1,y1),(x2,y2)以及要求点x'
READ(*,*) X1,Y1,X2,Y2,X
AL1=(X-X2)/(X1-X2)
AL2=(X-X1)/(X2-X1)
Y=Y1*AL1+Y2*AL2
WRITE(*,*) Y
END
```

我们以课本 P48 第 2 题为例，经过运算后，得到输出结果如下：

```
 请依次输入(x1,y1),(x2,y2)以及要求点x
0.5 -0.693147 0.6 -0.510826 0.54
 -0.6202186
Press any key to continue
```

抛物线插值

接着，我们考虑抛物线插值，即利用三点进行插值，程序如下：

```
PROGRAM EX1
WRITE(*,*) '请依次输入(x1,y1),(x2,y2),(x3,y3)以及要求点x'
READ(*,*) X1,Y1,X2,Y2,X3,Y3,X
AL1=(X-X2)*(X-X3)/(X1-X2)/(X1-X3)
AL2=(X-X1)*(X-X3)/(X2-X1)/(X2-X3)
AL3=(X-X1)*(X-X2)/(X3-X1)/(X3-X2)
Y=Y1*AL1+Y2*AL2+Y3*AL3
WRITE(*,*) Y
END
```

我们还以课本 P48 第 2 题为例，进行抛物线插值，结果如下：

```
 请依次输入(x1,y1),(x2,y2),(x3,y3)以及要求点x
0.4 -0.916291 0.5 -0.693147 0.6 -0.510826 0.54
 -0.6153198
Press any key to continue
```

家瑞点评：此程序为插值的基本程序，除此之外还有拉格朗日插值多项式，可进行更高阶的插值，但是高次插值的结果可能会不理想，在两边出现发散的现象，因此我们可以考虑分段低次插值，即分段多次使用该插值程序，在此便不做展示。同时，虽然这里未提及插值余项，但是我们不能忽视，它在数值计算中有着重要的作用，具体请参考课本。

3.3 牛顿插值

牛顿插值多项式

当我们需要增加或减少插值节点时，运用拉格朗日插值就会很不方便，因此我们引入了均差的概念，使用牛顿插值，但其本质仍为多项式插值。在此展示的为五点四次插值。

```
PROGRAM EX1 !牛顿五点四次插值
DIMENSION A(5,5)
DIMENSION B(10)
Y=0

WRITE(*,*) '请依次输入(x1,y1),(x2,y2),(x3,y3),(x4,y4),(x5,y5)'
DO J=1,10
READ(*,*) B(J)
END DO
WRITE(*,*) '请依次输入待求点x'
READ(*,*) X

DO J=1,5
A(1,J)=B(2*J)
END DO
DO I=2,5
DO J=I,5
A(I,J)=(A(I-1,J)-A(I-1,J-1))/(B(2*J-1)-B(2*J-3-2*(I-2)))
END DO
END DO

Y=A(1,1)+A(2,2)*(X-B(1))+A(3,3)*(X-B(1))*(X-B(3))
Y=Y+A(4,4)*(X-B(1))*(X-B(3))*(X-B(5))
Y=Y+A(5,5)*(X-B(1))*(X-B(3))*(X-B(5))*(X-B(7))
DO I=1,5
WRITE(*,*) A(1:I,I)
END DO
WRITE(*,*) Y
END
```

我们以课本中 P32 例 4 为例子，输出结果如下：

```
 请依次输入(x1,y1),(x2,y2),(x3,y3),(x4,y4),(x5,y5)
0.4
0.41075
0.55
0.57815
0.65
0.69675
0.80
0.88811
0.9
1.02652
 请依次输入待求点x
0.596
  0.4107500
  0.5781500        1.116000
  0.6967500        1.186001       0.2800031
  0.8881100        1.275733       0.3589296       0.1973163
   1.026520        1.384101       0.4334712       0.2129759       3.1319205E-02
  0.6319175
Press any key to continue
```

在此为显得与前面结果不同，在输出结果方面做了改进，因此使用了二维数组。我们在此使用的为五点四次插值，其中“A(I,J)”的计算为本程序的一大难点。

差分形式的牛顿插值多项式

对于牛顿插值多项式，还有另外一种形式，即差分形式，如下：

```
PROGRAM EX1 !牛顿五点四次插值
DIMENSION A(5,5)
DIMENSION B(10)
Y=0

WRITE(*,*) '请依次输入(x1,y1),(x2,y2),(x3,y3),(x4,y4),(x5,y5)'
DO J=1,10
READ(*,*) B(J)
END DO

WRITE(*,*) '请依次输入待求点x,区间长度h'
READ(*,*) X,H
T=(X-B(1))/H
DO J=1,5
A(1,J)=B(2*J)
END DO

DO I=2,5
DO J=I,5
A(I,J)=(A(I-1,J)-A(I-1,J-1))
END DO
END DO

Y=A(1,1)+A(2,2)*T+A(3,3)*T*(T-1)/2+A(4,4)*T*(T-1)*(T-2)/6
Y=Y+A(5,5)*T*(T-1)*(T-2)*(T-3)/24

DO I=1,5
WRITE(*,*) A(1:I,I)
END DO
```

我们以书本 P34 例 5 为例，结果展示如下：

```
 请依次输入(x1,y1),(x2,y2),(x3,y3),(x4,y4),(x5,y5)
0
1
0.1
0.995
0.2
0.98007
0.3
0.95534
0.4
0.92106
 请依次输入待求点x,区间长度h
0.048
0.1
   1.000000
  0.9950000      -4.9999952E-03
  0.9800700      -1.4930010E-02 -9.9300146E-03
  0.9553400      -2.4729967E-02 -9.7999573E-03  1.3005733E-04
  0.9210600      -3.4280002E-02 -9.5500350E-03  2.4992228E-04  1.1986494E-04
  0.9988427
Press any key to continue
```

个人理解相当于是把所需要做除数的地方由计算差值处改为了计算最终一点结果处，对于整体并无太大影响。并且区间与前者相比，是要为均匀分布的，因此需要在前者上做出微小改动。

家瑞点评：在这两个程序中，合理的运用了二维数组，既可以方便运算，而且有利于结果的展示，该程序在均差表的计算存在一点难度，要合理的利用循环变量“I、J”，正如第一个程序中“A(I,J)”的计算，将 I 与间隔联系起来，分母部分需要运用好变量，才能够简洁准确地输出结果。此外，埃尔米特插值等与这两节的插值较为近似，仅仅在于有导函数时，需要求解出其它系数，在此便不做展示。

3.4 最小二乘拟合

在逼近运算中，应用最为广泛的为最小二乘法，它适用于线性拟合（在变量之间不为线性关系的时候，需要将变量经过对数、指数等变换化为线性的关系），使得误差平方和最小。通过给出的几组数据以及所占的权重等，求解出系数，最终拟合曲线格式为“Y=A0+A1*X”，程序设计如下：

```
PROGRAM EX1 !最小二乘拟合
DIMENSION A(10)
DIMENSION B(5)
PHI00=0
PHI01=0
PHI11=0
PHI0F=0
PHI1F=0
WRITE(*,*) '请依次输入(x1,y1),(x2,y2),(x3,y3),(x4,y4),(x5,y5)'
DO J=1,10
READ(*,*) A(J)
END DO
WRITE(*,*) '请依次输入五点的权重'
DO J=1,5
READ(*,*) B(J)
END DO
DO I=1,5
PHI00=PHI00+B(I)
PHI01=PHI01+B(I)*A(2*I-1)
PHI11=PHI11+B(I)*A(2*I-1)*A(2*I-1)
PHI0F=PHI0F+B(I)*A(2*I)
PHI1F=PHI1F+B(I)*A(2*I-1)*A(2*I)
END DO
A1=(PHI0F*PHI01-PHI1F*PHI00)/(PHI01*PHI01-PHI00*PHI11)
A0=(PHI0F*PHI11-PHI1F*PHI01)/(PHI00*PHI11-PHI01*PHI01)
WRITE(*,*) 'Y=',A0,'+',A1,'X'
END
```

我们以书本上 P75 例 9 为例子，程序运行结果如下：

```
 请依次输入(x1,y1),(x2,y2),(x3,y3),(x4,y4),(x5,y5)
1
4
2
4.5
3
6
4
8
5
8.5
 请依次输入五点的权重
2
1
3
1
1
 Y=    2.564815      +    1.203704      X
Press any key to continue
```

当针对于不同点的最小二乘拟合时，我们只需要在程序中进行微小的改动即可。

家瑞点评：对于最小二乘拟合，其重要思想在于公式的推导，即如何将变量改造为一个二元线性方程组的求解问题的。其原理为使法方程 Ga=d 有唯一解，要求矩阵 G 的行列式为 0，根据这个特性，我们求出了拟合系数。最小二乘法在曲线拟合部分有着很重要的意义，可以帮助我们进行预测等，此外，它的求解思想很值得我们去注意。

3.5 数值积分

牛顿-柯特斯公式

牛顿-柯特斯公式对于 n=3 的情况正是前文中提到的辛普森求积公式，见 2.8 节。在 n=4 的情况下，为柯特斯公式。具体的柯特斯公式表可见教材的 P104，在此不做罗列。值得我们去注意的是，当 n 过高时，计算结果会出现不稳定的现象，即龙格现象。因此想要通过增加 n 而追求过高的代数精度是不可靠的，因此我们需要的是分段，通过分段之后进行的积分，可以很大程度上提高我们的精度。

复合求积公式

复合求积公式分为两种，复合梯形公式与复合辛普森求积公式，两种公式的难度都不高，仅仅通过循环便可求出，程序如下：

```
PROGRAM EX1 !计算积分
WRITE(*,*) 'the function in (E,F) for N'
READ(*,*) E,F,N
CALL FUTI(E,F,N,Y)
CALL FUXIN(E,F,N,Z)
WRITE(*,*) Y,'复合梯形公式'
WRITE(*,*) Z,'符合辛普森公式'
END
```

```
SUBROUTINE FUTI(E,F,N,G)
DX=(F-E)/N
G=0
CALL FUN(E,G1)
CALL FUN(F,G2)
G=G1+G2
DO I=1,N-1
X=E+DX*I
CALL FUN(X,XI)
G=G+2*XI
END DO
G=G*DX/2
END
```

```
SUBROUTINE FUXIN(E,F,N,W)
DX=(F-E)/N
W1=0
W2=0
W3=0
CALL FUN(E,W1)
CALL FUN(F,W2)
W=W1+W2
DO I=1,N-1
X=E+DX*(I-0.5)
Y=E+DX*I
CALL FUN(X,XI)
CALL FUN(Y,YI)
W=W+4*XI+2*YI
END DO
CALL FUN(F-DX/2,W3)
W=W+4*W3
W=W*DX/6
END

SUBROUTINE FUN(X,Y)
Y=SIN(X)/X
END
```

依次为两种求积的子程序。我们以课本上 P108 例 3 为例，运行结果如下：

```
 the function in (E,F) for N
1E-10 1 8
  0.9456909      复合梯形公式
  0.9460831      符合辛普森公式
Press any key to continue
```

家瑞点评：这两种复合求积都较为简单，其主要意义在于我们求积分可以将整体化为微元，利用复合的方式提高精度，这也是一个很重要的数学思想。根据运行结果我们可以看到，当 N=8 时，两者便可以取到较高的精度，因此分段积分的运用也较为广泛。

龙贝格求积公式

龙贝格求积是一种通过较小次数的运算便可以达到很高精度的求积公式，其利用外推原理，进而简化了计算所需要的数据量，其程序编写难度较大，程序展示如下：

```
PROGRAM EX1 !使用Romberg算法进行任意函数积分
DIMENSION Z(50,50)
REAL H,AB,XX,YY,ABC,ABCD,K,E,F
I=1
J=1
WRITE(*,*) 'the function in (E,F)'
READ(*,*) E,F
CALL DIFF(E,F,Z,I)
DO J=1,I
WRITE(*,*) Z(1:J,J)
END DO
write(*,*) I
END

SUBROUTINE FUNC(X,F)
F = 1/X !在此处设置待积分函数F(X)
END

SUBROUTINE DIFF(E,F,Z,I)
REAL H,AB,XX,YY,K,X,E,F
REAL*8 ABC,ABCD
DIMENSION Z(50,50)
N=1
I=1
H=(F-E)/N
CALL FUNC(E,XX)
CALL FUNC(F,YY)
Z(I,I)=0.5*H*(XX+YY) !求解出Z(1,1)
DO
I=I+1
N=N*2
H=(F-E)/N
AB=0
DO K=1,N/2
X=E+H+2*(K-1)*H
CALL FUNC(X,XX)
AB=AB+XX
END DO
Z(1,I)=0.5*Z(1,I-1) + H*AB !求解出Z(1,I)
DO K=2,I
ABC = 4**(K)/(4**(K)-1)
ABCD = 1/(4**(K)-1)
Z(K,I) = ABC*Z(K-1,I)-ABCD*Z(K-1,I-1)
END DO !求解出Z(I,J)
IF (ABS(Z(I,I)-Z(I-1,I-1)).LE.0.000001) EXIT !截断条件
END DO
END
```

该程序分为三部分，第一部分为主程序，仅负责输入、调用子程序、输出等功能，第一个子程序“FUNC”为调用函数程序，负责输入 x，输出 f(x)，即在此处可以设置被积函数。第二个子程序则为该程序的核心，通过逐步计算出 Z(I,J)，输出结果，并且设置了截断条件，当

外推结果之间的误差很小时，进行截断。我们以书本 P137 第 14 题为例，输出结果如下：

```
 the function in (E,F)
0 1
  0.5000000
  0.4267767      0.4218951
  0.4070181      0.4057009      0.4054438
  0.4018125      0.4014654      0.4013982      0.4013823
  0.4004634      0.4003735      0.4003561      0.4003520      0.4003510
  0.4001177      0.4000947      0.4000902      0.4000892      0.4000889
  0.4000889
  0.4000298      0.4000239      0.4000228      0.4000225      0.4000224
  0.4000224      0.4000224
  0.4000075      0.4000060      0.4000057      0.4000056      0.4000056
  0.4000056      0.4000056      0.4000056
  0.4000019      0.4000015      0.4000014      0.4000014      0.4000014
  0.4000014      0.4000014      0.4000014      0.4000014
  0.4000005      0.4000004      0.4000003      0.4000003      0.4000003
  0.4000003      0.4000003      0.4000003      0.4000003      0.4000003
  0.4000001      0.4000000      0.4000000      0.4000000      0.4000000
  0.4000000      0.4000000      0.4000000      0.4000000      0.4000000
  0.4000000
          11
Press any key to continue
```

并且我们可见，当进行到第 11 次时，便达到了所需精度，输出了最终结果。在此处显示不太美观的原因之一是因为输出框的长度所限。

家瑞点评：龙贝格求积公式作为一种重要的求积公式，其外推加速的思想值得我们去仔细思考，这种加速方法也等于是将建立在原有的基础上，通过调和占比，复合求出了新的结果，对于加快积分运算有着很重要的意义。

高斯求积公式

与牛顿-柯特斯求积公式相似，高斯-勒让德求积公式也是在积分区间选取内选取几个点进行积分，以下为设计程序：

```
      PROGRAM EX1 !计算积分
      WRITE(*,*) 'the function in (E,F)'
      READ(*,*) E,F
      CALL GAO(E,F,Y)
      WRITE(*,*) Y,'三点高斯-勒让德公式'
      END

      SUBROUTINE GAO(E,F,G)
      DX=(F-E)/2
      EX=(F+E)/2
      G1=0.7745967*DX+EX
      G2=EX
      G3=-0.7745967*DX+EX
      CALL FUN(G1,G11)
      CALL FUN(G2,G22)
      CALL FUN(G3,G33)
      G=(0.5555556*G11+0.8888889*G22+0.5555556*G33)*DX
      END

      SUBROUTINE FUN(X,Y)
      Y=SIN(X)/X
      END
```

但是与之不同的是，牛顿-柯特斯公式中要使用的是积分区间端点的值，而高斯-勒让德公式并不需要；而高斯-勒让德公式需要将积分区间改为[-1,1]。因此，对于一些问题，该公式会更加方便。在该程序中，我们使用了区间变化，尤为重要的是，不要忘记在进行自变量替换的同时，dx 到 dt 也需要乘以区间长度的比值。

为了作为对比，我们取了与前文中复合求积公式相同的例子，程序运行如下：

```
 the function in (E,F)
0 1
  0.9460832      三点高斯-勒让德公式
Press any key to continue
```

我们可以看到，不需要像前者一样从 1E-10 作为积分区间的开始，而是可以直接从 0 开始，并且在只取三个点的情况下，也可以取得较高的精度。

家瑞点评：对于此程序，我们在编写的时候要理解其原理，并且要注意区间的变换，在程序中使用了“(F-E)/2”与“(F+E)/2”来实现。整体难度不大，要理解其与牛顿-柯特斯求积公式的异同。

3.6 非线性方程求解

二分法

非线性问题在现实生活中经常出现，并且在数值计算领域有着很重要的地位，而非线性问题的求解也具有一定的难度，在此我们只考虑单根的求解。首先是二分法，即预先通过判断

```
PROGRAM EX1 !二分法
WRITE(*,*) '请输入大致解的区间(E,F)'
READ(*,*) E,F
CALL FUN(E,Y2)
CALL FUN(F,Y3)
IF(Y2*Y3.GE.0) THEN
WRITE(*,*) '请输入正确的区间'
ELSE
CALL ERFEN(E,F,X)
WRITE(*,*) X
END IF
END

SUBROUTINE ERFEN(E,F,X)
CALL FUN(E,Y1)
DO
X=(E+F)/2
CALL FUN(X,Y)
IF(Y*Y1.LT.0) THEN
F=X
ELSE
E=X
END IF
IF(ABS(F-E).LE.1E-5) EXIT
END DO
X=(E+F)/2
END

SUBROUTINE FUN(X,Y)
Y=X*X*X-X-1
END
```

知道根在哪个区间后，计算区间中点对应的函数值的正负，并与区间两端的函数值的正负作比较，进而缩小区间，并不断进行二分，达到所需计算精度。程序设计如图，在此程序中，可以通过调节最后一个子程序来更改被积分函数，在第一个子程序中，可设置误差截断条件。

我们以课本 P214 例 2 为例子，运行结果如下：

```
 请输入大致解的区间(E,F)
1 1.5
   1.324718
Press any key to continue
```

牛顿迭代法

牛顿迭代法的实质是一种线性化方法，其根本思想是将非线性方程转化为线性方程求解的问题，即利用导数与原函数进行求解，该程序很简单，程序展示如下：

```
PROGRAM EX1 !牛顿迭代法
WRITE(*,*) '请输入解的大致值X'
READ(*,*) X
CALL NIU(X)
WRITE(*,*) X
END

SUBROUTINE NIU(X)
DO
CALL FUN(X,Y)
CALL FUND(X,Y1)
X=X-Y/Y1
IF(Y/Y1.LE.1E-5) EXIT
END DO
END

SUBROUTINE FUN(X,Y) !原函数
Y=X*EXP(X)-1
END

SUBROUTINE FUND(X,Y) !导函数
Y=EXP(X)+X*EXP(X)
END
```

我们以课本 P223 例 7 为例，运行结果如下：

```
 请输入解的大致值X
0.5
  0.5710204
Press any key to continue
```

家瑞点评：这两个程序难度均不大，这两种程序均为非线性方程的求解方法，除此之外，还有简化牛顿法、牛顿下山法、弦截法、抛物线法等，运用迭代的思想，相当于化非线性问题为线性问题，这些方法都简单易操作，在此便不再赘述。而对与非线性方程的求解肯定远远不止于此，我们还有更多要学习的，而数值求解作为非线性方程求解的一大“法宝”，本人亦深有体会，因此我们对于非线性求解问题不可大意。

第四章 台上玩月——线性方程组

这部分，让我们走进矩阵，来体会矩阵的美妙吧！

对于线性方程组的求解问题，分为两大种方法：直接方法与迭代方法。本章中前两节为直接法，最后一节为迭代法。两种方法最适用的对象不同：直接法适合低阶稠密矩阵方程组和某些大型稀疏矩阵方程组；迭代法适合大型稀疏矩阵方程组。

4.1 高斯消去法

高斯消去法

高斯消去法是线性方程组求解的最基本方法，其原理是通过行的初等变换，利用逐次消去未知数的方法把原线性方程组化为与其等价的三角形线性方程组，即分为两大部分：三角化和求解。其程序如下：

```
PROGRAM EX1
DIMENSION Z(4,3) !在此处进行修正数量，注意其中是先列后行
DIMENSION X(3)
WRITE(*,*) 'Please input the equation, column then row'
DO I=1,4 !输入线性方程组部分
DO J=1,3
READ(*,*) Z(I,J)
END DO
END DO
CALL XIAO(Z)
CALL QIU(Z,X)
DO I=1,3
WRITE(*,*) X(I)
END DO
END

SUBROUTINE XIAO(Z)
DIMENSION Z(4,3)
DIMENSION X(3)
DO I=1,2
DO J=I+1,3
M=Z(I,J)/Z(I,I)
DO K=1,4
Z(K,J)=Z(K,J)-Z(K,I)*M
END DO
END DO
END DO
DO I=1,3
WRITE(*,*) Z(1,I),Z(2,I),Z(3,I),Z(4,I)
END DO
END
```

```
SUBROUTINE QIU(Z,X)
DIMENSION Z(4,3)
DIMENSION X(3)
X(3)=Z(4,3)/Z(3,3)
DO I=2,1,-1 !此处尤为重要，当从大到小时，要修改间隔为-1
W=0
DO J=3,I+1,-1
W=W+X(J)*Z(J,I)
END DO
X(I)=(Z(4,I)-W)/Z(I,I)
END DO
END
```

（注：程序中变量 M 需全部改为 Q）

我们以课本 P143 例 2 为例子，输出结果如下：

```
 Please input the equation, column then row
1
0
2
1
4
-2
1
-1
1
6
5
1
   1.000000        1.000000        1.000000        6.000000
  0.0000000E+00    4.000000       -1.000000        5.000000
  0.0000000E+00   0.0000000E+00   -2.000000       -6.000000
   1.000000
   2.000000
   3.000000
Press any key to continue
```

家瑞点评：对于这个程序，在编写时略有困难，在于 FORTRAN 中的二位数组为先列后行，这点与我们之前的不同，并且在第二个子程序中的循环，当我是“I=2,1”时，输出结果始终不正确，后经过仔细排查，通过输出各种变量后，才发现是 W 不正确，但对于其为什么不正确，可谓是想了很久，才明白是因为“I=2,1,-1”才是正确的格式，因为在 FORTRAN 中的循环，我们直接用从 2 到 1 的话，默认间距为 1，那么这个循环是不会执行的，因此我们要去定义其间距为-1，这个地方也算是一个需要特别注意的小细节吧。

列主元消去法

当系数矩阵的主对角线含有 0 的时候，上述程序就会出现问题，并且即使非零的情况下，当其很小时，也会因为其它元素数量级的严重增长和舍入误差的扩散，使得最终的结果不可靠。因此我们有必要考虑在消去对应列时，将其最大值所在的行提到首位中，进而减小误差，并且我们可以通过对上个程序进行一些简单的改造即可。程序展示如下：

```
PROGRAM EX1
DIMENSION Z(4,3) !在此处进行修正数量，注意其中是先列后行
DIMENSION X(3)
WRITE(*,*) 'Please input the equation, column then row'
DO I=1,4 !输入线性方程组部分
DO J=1,3
READ(*,*) Z(I,J)
END DO
END DO
CALL XIAO(Z)
CALL QIU(Z,X)
DO I=1,3
WRITE(*,*) X(I)
END DO
END
```

```
SUBROUTINE PAI(Z,I)
DIMENSION Z(4,3)
C=0
MAX=ABS(Z(I,I))
K=I
DO N=I+1,3
IF(ABS(Z(I,N)).GT.MAX) THEN
MAX=ABS(Z(I,N))
K=N
END IF
END DO
DO N=1,4
C=Z(N,K)
Z(N,K)=Z(N,I)
Z(N,I)=C
END DO
DO K=1,3
WRITE(*,*) Z(1,K),Z(2,K),Z(3,K),Z(4,K)
END DO
WRITE(*,*) '----------'
END
```

```
SUBROUTINE XIAO(Z)
DIMENSION Z(4,3)
DIMENSION X(3)
L=1
DO I=1,2
CALL PAI(Z,L)
L=L+1
DO J=I+1,3
Q=Z(I,J)/Z(I,I)
DO K=1,4
Z(K,J)=Z(K,J)-Z(K,I)*Q
END DO
END DO
END DO
DO I=1,3
WRITE(*,*) Z(1,I),Z(2,I),Z(3,I),Z(4,I)
END DO
WRITE(*,*) '----------'
END
```

```
      SUBROUTINE QIU(Z,X)
      DIMENSION Z(4,3)
      DIMENSION X(3)
      X(3)=Z(4,3)/Z(3,3)
      DO I=2,1,-1 !此处尤为重要，当从大到小时，要修改间隔为-1
      W=0
      DO J=3,I+1,-1
      W=W+X(J)*Z(J,I)
      END DO
      X(I)=(Z(4,I)-W)/Z(I,I)
      END DO
      END
```

对于这个程序确实有点长了，但是理解起来的话不会那么苦难，主要也是在于细节之处。这个程序与前面相比，多了一个找到列主元然后进行换序的步骤。然后让我们以课本 P148 例 4 为例子，看看效果吧。我们可以看到，经过几次交换位置、消去后，取得的最终结果较为准确，稳定性较高。

```
 Please input the equation, column then row
0.001
-1
-2
2
3.712
1.072
3
4.623
5.643
1
2
3
  -2.000000        1.072000        5.643000        3.000000
  -1.000000        3.712000        4.623000        2.000000
  1.0000000E-03    2.000000        3.000000        1.000000
 ----------
  -2.000000        1.072000        5.643000        3.000000
  0.0000000E+00    3.176000        1.801500       0.5000000
  0.0000000E+00    2.000536        3.002821        1.001500
 ----------
  -2.000000        1.072000        5.643000        3.000000
  0.0000000E+00    3.176000        1.801500       0.5000000
  0.0000000E+00   2.1807054E-08    1.868072       0.6865542
 ----------
 -0.4903964
 -5.1035203E-02
  0.3675203
Press any key to continue
```

家瑞点评：原本想在这个程序中加入一个判断算法，当需要交换的时候才会进行交换，但是发现运行不同，最后只能作罢，但是对于结果并无影响，只会影响运算速度。此外，我们可以通过调节循环系数“I、J、K”等进行不同元的线性方程组求解，这个程序中只是以三元的为例，但是本着对于任意元都适用的目的，所以程序中所有地方均采用了循环方法，这样就可以通过简单改动而实现其普适性。此外，通过这两个编程，我收获最大的是要充分利用循环系数，它是一大法宝。

4.2 矩阵三角分解法

直接三角分解法

对于线性方程组求解，LU 分解（三角分解）是一种较高斯消元法更为简洁的方法，通过将系数矩阵分为两部分，再引入中间变量，便可以实现快速的计算，而其中的直接方法如下：

```
PROGRAM EX1
DIMENSION A(3,3) !在此处进行修正数量，注意其中是先列后行
DIMENSION AL(3,3)
DIMENSION U(3,3)
DIMENSION B(3)
DIMENSION X(3)
DIMENSION Y(3)
WRITE(*,*) 'Please input the equation, column then row'
DO I=1,3 !输入线性方程组部分A
DO J=1,3
READ(*,*) A(I,J)
END DO
END DO
DO I=1,3 !输入b
READ(*,*) B(I)
END DO
CALL FEN(A,AL,U)
CALL QIUY(AL,Y,B)
CALL QIUX(U,X,Y)
DO I=1,3
WRITE(*,*) AL(1,I),AL(2,I),AL(3,I)
END DO
WRITE(*,*) '----------'
DO I=1,3
WRITE(*,*) U(1,I),U(2,I),U(3,I)
END DO
WRITE(*,*) '----------'
DO I=1,3
WRITE(*,*)  Y(I)
END DO
WRITE(*,*) '----------'
DO I=1,3
WRITE(*,*) X(I)
END DO
END
```

```
SUBROUTINE FEN(A,AL,U) !LU分解
DIMENSION A(3,3)
DIMENSION AL(3,3)
DIMENSION U(3,3)
U(1,1)=A(1,1)
DO I=1,3 !让L矩阵对角线为1
AL(I,I)=1
END DO
DO I=2,3
U(I,1)=A(I,1)
AL(1,I)=A(1,I)/U(1,1)
END DO
```

```
DO I=2,3
U(I,1)=A(I,1)
AL(1,I)=A(1,I)/U(1,1)
END DO
DO I=2,3 !计算U与L
DO J=I,3
ALU=0
DO K=1,I-1
ALU=ALU+AL(K,I)*U(J,K)
END DO
U(J,I)=A(J,I)-ALU
END DO
DO J=I+1,3
ALU=0
DO K=1,I-1
ALU=ALU+AL(K,J)*U(I,K)
END DO
AL(I,J)=(A(I,J)-ALU)/U(I,I)
END DO
END DO
END
```

```
SUBROUTINE QIUY(AL,Y,B)
DIMENSION AL(3,3)
DIMENSION Y(3)
DIMENSION B(3)
Y(1)=B(1)
DO I=2,3
ALY=0
DO J=1,I-1
ALY=ALY+AL(J,I)*Y(J)
END DO
Y(I)=B(I)-ALY
END DO
END

SUBROUTINE QIUX(U,X,Y)
DIMENSION U(3,3)
DIMENSION X(3)
DIMENSION Y(3)
X(3)=Y(3)/U(3,3)
DO I=2,1,-1
UX=0
DO J=I+1,3
UX=UX+U(J,I)*X(J)
END DO
X(I)=(Y(I)-UX)/U(I,I)
END DO
END
```

该程序较长，但是如果按照书本上的过程书写，则不会太过于麻烦。

家瑞点评：此程序分为三大部分，主程序负责输入与输出，而子程序为进行 LU 分解和求解，该过程虽然繁琐，但是个人推荐按照课本定义进行，则会明了很多，在手算时候，可以利用两个矩阵相乘来计算 L 与 U 矩阵的系数，这样会简单很多，但是既然交给计算机工作，那么按公式走就 OK 了。平方根法与此类似，只是进行部分变换，在此不做阐述。

以课本 P153 例 5 为例，运行结果如下：

```
 Please input the equation, column then row
1
2
3
2
5
1
3
2
5
14
18
20
   1.000000       0.0000000E+00  0.0000000E+00
   2.000000        1.000000       0.0000000E+00
   3.000000       -5.000000        1.000000
 ----------
   1.000000        2.000000        3.000000
  0.0000000E+00    1.000000       -4.000000
  0.0000000E+00   0.0000000E+00   -24.00000
 ----------
   14.00000
  -10.00000
  -72.00000
 ----------
   1.000000
   2.000000
   3.000000
Press any key to continue
```

追赶法

在一些实际生活应用中，三对角系数矩阵更为常见，作为三角分解的一个特殊情况，我们有必要在此讨论，程序设计如下：

```
      PROGRAM EX1
      DIMENSION a(4)
      DIMENSION b(5)
      DIMENSION c(4)
      DIMENSION beta(4)
      DIMENSION f(5)
      DIMENSION x(5)
      DIMENSION y(5)
      WRITE(*,*) '输入线性方程组a部分'
      DO I=1,4
      READ(*,*) a(I)
      END DO
      WRITE(*,*) '输入线性方程组b部分'
      DO I=1,5
      READ(*,*) b(I)
      END DO
      WRITE(*,*) '输入线性方程组c部分'
      DO I=1,4
      READ(*,*) c(I)
      END DO
      WRITE(*,*) '输入线性方程组f部分'
      DO I=1,5
      READ(*,*) f(I)
      END DO
```

```
      CALL FEN(a,b,c,beta)
      CALL QIUY(a,b,f,y,beta)
      CALL QIUX(x,y,beta)

      DO I=1,5
      WRITE(*,*) y(I)
      END DO
      WRITE(*,*) '----------'
      DO I=1,5
      WRITE(*,*) x(I)
      END DO
      END
      SUBROUTINE FEN(a,b,c,beta) !分解
      DIMENSION a(4)
      DIMENSION b(5)
      DIMENSION c(4)
      DIMENSION beta(4)
      beta(1)=c(1)/b(1)
      DO I=2,4
      beta(I)=c(I)/(b(I)-a(I-1)*beta(I-1))
      END DO
      END

      SUBROUTINE QIUY(a,b,f,y,beta)
      DIMENSION a(4)
      DIMENSION b(5)
      DIMENSION beta(4)
      DIMENSION f(5)
      DIMENSION y(5)
      y(1)=f(1)/b(1)
      DO I=2,5
      y(I)=(f(I)-a(I-1)*y(I-1))/(b(I)-a(I-1)*beta(I-1))
      END DO
      END

      SUBROUTINE QIUX(x,y,beta)
      DIMENSION beta(4)
      DIMENSION x(5)
      DIMENSION y(5)
      x(5)=y(5)
      DO I=4,1,-1
      x(I)=y(I)-beta(I)*x(I+1)
      END DO
```

家瑞点评：追赶法其实是高斯消去法的特殊化，针对于三对角系数矩阵，其中所需要算的项相较高斯消去要少很多，计算起来也很方便（当然这是原本方程特性的功劳）。

以课本 P177 第 9 题为例，输出结果如下：

```
 输入线性方程组a部分
-1
-1
-1
-1
 输入线性方程组b部分
2
2
2
2
2
 输入线性方程组c部分
-1
-1
-1
-1
 输入线性方程组f部分
1
0
0
0
0
  0.5000000
  0.3333333
  0.2500000
  0.2000000
  0.1666667
----------
  0.8333334
  0.6666667
  0.5000000
  0.3333333
  0.1666667
Press any key to continue
```

4.3 迭代法

雅可比迭代法

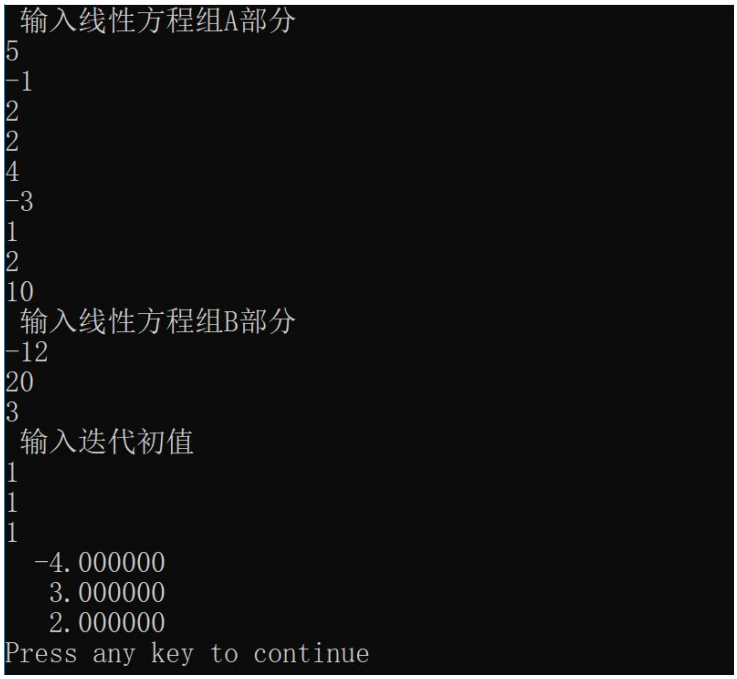
迭代法在求解线性微分方程中也有着很重要的作用，在将方程经过变换后，得到迭代公式，通过设置初值，得到满足一定精度的最终结果。下面，是最基础的雅可比迭代法：

```
PROGRAM EX1
DIMENSION A(3,3)
DIMENSION B(3)
DIMENSION X(3)
WRITE(*,*) '输入线性方程组A部分'
DO I=1,3
DO J=1,3
READ(*,*) A(I,J)
END DO
END DO
WRITE(*,*) '输入线性方程组B部分'
DO I=1,3
READ(*,*) B(I)
END DO
WRITE(*,*) '输入迭代初值'
DO I=1,3
READ(*,*) X(I)
END DO
CALL DIE(A,B,X)
DO I=1,3
WRITE(*,*) X(I)
END DO
END
```

```
SUBROUTINE DIE(A,B,X)
DIMENSION A(3,3)
DIMENSION B(3)
DIMENSION X(3)
DIMENSION Y(3)
DO
DO I=1,3
Y(I)=X(I)
END DO
DO I=1,3
AX=-A(I,I)*X(I)
DO J=1,3
AX=AX+A(J,I)*X(J)
END DO
X(I)=(B(I)-AX)/A(I,I)
END DO
ABC=SQRT((Y(3)-X(3))**2+(Y(2)-X(2))**2+(Y(1)-X(1))**2)
IF(ABC.LT.1E-10) EXIT
END DO
END
```

家瑞点评：雅可比迭代也需要在满足一定条件下才成立，在我看来，正是条件使得其迭代的过程收敛，即需要主对角线上的元素比其相邻的大。这样可以保证它的收敛性。这个程序也较为简单。

我们以课本 P209 第 1 题为例，程序运行结果如下：



高斯-赛德尔迭代

高斯-赛德尔迭代是雅可比迭代的“加速版”，运用初步迭代出的结果去替代原来的结果，相对于雅可比迭代，其本质仍为相通的，在此不再列出其程序。我用一个词来形容该迭代：“实时更新”。

针对这两种迭代，我们了解到，初值的选择尤为重要，因此初值问题亦不可忽略。与此同样重要的还有截断误差，我们应选取合适的方式来判断并截断。

这部分到此就结束啦，谢谢！

总结与心得

经过六周的学习，计算方法这门课结课了；但是，对于数值分析的学习，我们永远在路上！

这篇文章也终于写完了，写的四不像，像是说明书，也像是学习总结文，也像是代码范例，也像是个人独白。

先说些套话，经过这段学习，了解了书本上的算法知识，学会了 FORTRAN，训练了自己的逻辑思考能力，增强了编程能力，那别的呢？

这短短的时间里，其实最重要的是对数值的热爱吧。就我本人而言，对数值接触也较早一些，在去年便开始了我的“科研”，那时便了解到偏微分方程的解是有解析解与数值解的，解析解在大多数情况下其实很难求，甚至求不出来，而数值，正式最好的替代。将曲线离散化，密密麻麻的点，亦可串起函数的旋律，数值真的太方便了，但其背后也有着我们需要去不断探索的地方：改进算法、初值选取、误差分析等等。我也曾质疑过数值的准确性，也不止依次怀疑过，数值是否真的可靠，但经过实验、观察，我也慢慢接受了数值，相信了数值，这也是我与数值求解的故事。

除此之外，数值方法与力学也关联甚多，有限插值、谱方法、有限元，三大数值法宝，总能解决许多问题，总之，学习永无止境。

在以后的时间吧，我尽力把前面文中提到的遗憾补上，画上一个圆满的开头（不是句号！）。

最后，谢谢徐老师的教导！

张家瑞