实 验 报 告

课程名称：	数学实验
学生姓名：	陈艺荣
学生学号：	**201530301043**
学生专业：	电子科学与技术(卓越班)
开课学期：	**2016-2016 学年第 2 学期**

数学学院

2017 年 2 月

目　录

实验一　Matlab 基础知识

地　　点：	4 号楼　　4104 房；	**实验台号：**	66
实验日期与时间：	2017 年 3 月 8 日	**评　　分：**	
预习检查纪录：		**实验教师：**	刘小兰
电子文档存放位置：			
电子文档文件名：	卓越班-66-陈艺荣实验一		

批改意见：

1 实验目的

- 掌握Matlab中的常用函数与变量、表达式的定义方法。
- 熟悉Matlab M文件的编写和运行方式。
- 掌握Matlab语言中的程序结构，熟悉画图命令的使用。

2 问题 1

2.1 问题描述

利用 subplot 命令分别在不同的坐标系下画出如下的曲线，为每幅图形加上标题，可自己发挥，对图形进行美化等。

$f(x)=x^3+2x^2-3x+4$

四叶玫瑰线 $r=\cos 2\theta$

参数方程 $x=(1+\sin t-2\cos 4t)\cos t, y=(1+\sin t-2\cos 4t)\sin t$

$y=\sin x+\sin 2x$

2.2 实验原理

1、MATLAB 中，subplot 是在一个图像窗口中显示多个图形的命令，其基本调用

格式为：subplot(m, n, k) 或 subplot(m n k)，其中 m, n, k 取值为 1～9。该函数表示将绘图窗口划分为 m×n 个子窗口（子图），即 m 和 n 代表在一个图像窗口中显示 m 行 n 列个图像，并在第 k 个子窗口中绘图。它的几种调用格式如表 1 所示

表 1 subplot 函数的几种调用格式

编号	subplot 函数的调用格式
1	subplot(m,n,p)
2	subplot(m,n,p,'replace')
3	subplot(m,n,p,'replace')
4	subplot('Position',positionVector)
5	subplot(___,Name,Value)
6	h = subplot(___)
7	subplot(h)

2、MATLAB 中，基本绘图函数 plot 函数的基本调用格式为：

plot(y)

plot(x,y)

plot(x1,y1,x2,y2,……)

以上三种格式中的 x,y 都可以是表达式。plot 是绘制二维曲线的基本函数，但在使用此函数之前，须先定义曲线上每一点的 x 以及 y 坐标。

3、MATLAB 中，polar 函数可用于描绘极坐标图像。

最简单而常用的命令格式：polar(THETA, RHO)

其中，THETA 是用弧度制表示的角度，RHO 是对应的半径。

4、MATLAB 中，ezplot 函数可用于描绘参数方程表示的曲线，其命令格式如下：ezplot(funx,funy,[tmin,tmax])：在默认区间 tmin < t < tmax 绘制参数定义的平面曲线 funx(t)和 funy(t).

2.3 算法与编程

2.3.1 编程描述

对于问题 1，我通过使用函数 subplot 将图像窗口分为2×2 个子窗口；

通过使用 plot 函数绘制曲线 $f(x)=x^3+2x^2-3x+4$；

通过使用 polar 函数绘制极坐标下的四叶玫瑰线 $r=\cos 2\theta$；

通过使用 ezplot 函数绘制参数方程

$x=(1+\sin t-2\cos 4t)\cos t, y=(1+\sin t-2\cos 4t)\sin t$ 表示的曲线；

通过使用 plot 函数绘制曲线 $y=\sin x+\sin 2x$；

通过使用 xlabel、ylabel、title 命令分别设定每一幅子图的横坐标、纵坐标、图标题。通过使用 grid on 等命令进一步美化了图像。

2.3.2 实现代码

对于问题 1 的具体的 MATLAB 函数实现代码（运行环境：matlabR2008a、matlabR2014a 均可以运行）如下

```
%   subplot 函数的使用
hold on;
%使用 hold on  可以保持上一幅图  当前图可以画在同一个轴上，而不覆盖。
subplot(2,2,1);                    %在第 1 个位置画图
x1=(-500:0.01:500);
y1=x1.^3+2*x1.^2-3*x1+4;
plot(x1,y1,'g');                   %'g'表示设定曲线颜色为绿色
xlabel('x');                       %插入横坐标标签
ylabel('y');                       %插入纵坐标标签
title('用 plot 函数画 y=x^3+2x^2-3x+4');        %插入图像的标题
grid on;                           %插入网格线
subplot(2,2,2);                    %在第 2 个位置画图
t=0:0.001:2*pi;
r=cos(2.*t);
polar(t,r,'r')              %用 polar 函数画极坐标图像，'r'表示设定曲线颜色为红色
title('用 polar 函数画四叶玫瑰线 r=cos2t');
subplot(2,2,3);                    %在第 3 个位置画图
ezplot('(1+sin(t)-2.*cos(4.*t)).*cos(t)','(1+sin(t)-2.*cos(4.*t)).*sin(t)');
```

```
%隐函数 f(x,y)=0,ezplot 函数的调用格式为 ezplot(f, [xminxmax] , [yminymax]);
title('用 ezplot 函数画参数方程曲线');
grid on;                          %插入网格线
subplot(2,2,4);                   %在第 4 个位置画图
x4=(-2*pi:0.01:2*pi);
y4=sin(x4)+sin(2.*x4);
plot(x4,y4,'c');
xlabel('x');
ylabel('y');
title('用 plot 函数画 y=sin(x)+sin(2x)');
grid on;                          %插入网格线
```

2.4 实验结果

在 matlab 中运行 2.3 算法与编程中所给出的代码，得到图 1 所示的结果

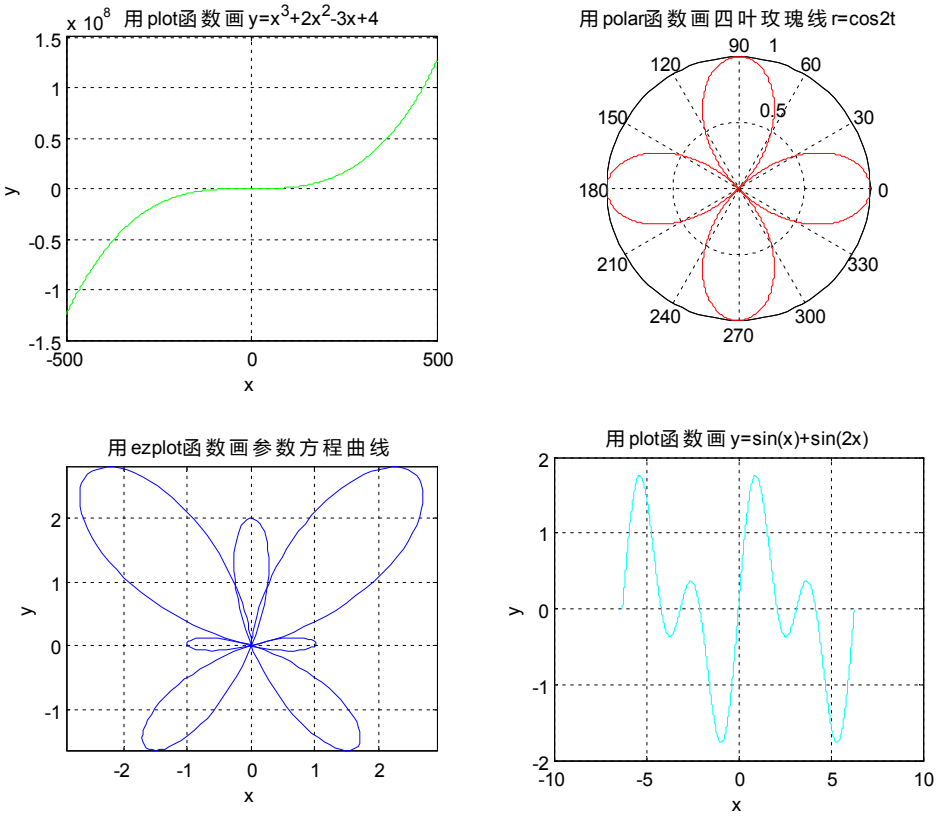


图 1 代码运行后得到的四个曲线图

3 问题 2

3.1 问题描述

编写M文件a_sqrt.m，用迭代法求 $x=\sqrt{a}$ 的值。求平方根的迭代公式为：

$$x_{n+1}=\frac{1}{2}\left(x_n+\frac{a}{x_n}\right)$$

迭代的终止条件为前后两次求出的 x 的差的绝对值小于 10^{-5}。

3.2 实验原理

3.2.1 迭代法求根的原理

设 r 是 $f(x)=0$ 的根，选取 x_0 作为 r 的初始近似值，过点 $(x_0,f(x_0))$ 做曲线 $y=f(x)$ 的切线 L，L 的方程为 $y=f(x_0)+f'(x_0)(x-x_0)$，求出 L 与 x 轴交点的横坐标 $x_1=x_0-\frac{f(x_0)}{f'(x_0)}$，称 x_1 为 r 的一次近似值。过点 $(x_1,f(x_1))$ 做曲线 $y=f(x)$ 的切线，并求该切线与 x 轴交点的横坐标 $x_2=x_1-\frac{f(x_1)}{f'(x_1)}$，称 x_2 为 r 的二次近似值。重复以上过程，得 r 的近似值序列，其中，$x_{n+1}=x_n-\frac{f(x_n)}{f'(x_n)}$ 称为 r 的 $n+1$ 次近似值，上式称为**牛顿迭代公式**。

3.2.2 Matlab 中的循环体

在本次实验中，我用到 for 循环语句和 while 循环语句，同时，利用到循环的嵌套，原理如下：

1、for 语句

for 语句的格式为：

```
for  循环变量=表达式 1:表达式 2:表达式 3
        循环体语句
    end
```

其中表达式 1 的值为循环变量的初值，表达式 2 的值为步长，表达式 3 的值为循

环变量的终值。步长为 1 时，表达式 2 可以省略。

for 语句更一般的格式为：

for 循环变量=矩阵表达式

循环体语句

end

执行过程是依次将矩阵的各列元素赋给循环变量，然后执行循环体语句，直至各列元素处理完毕。

2、while 语句

while 语句的一般格式为：

while (条件)

循环体语句

end

其执行过程为：若条件成立，则执行循环体语句，执行后再判断条件是否成立，如果不成立则跳出循环。

3、循环的嵌套

如果一个循环结构的循环体又包括一个循环结构，就称为循环的嵌套，或称为多重循环结构。

3.3 算法与编程

3.3.1 编程描述

1、用数组 A 装所有测试数据，通过 for 循环，逐个将 A 中的数据进行迭代法求根；

2、先根据输入的 x_1 和迭代公式 $x_{n+1}=\frac{1}{2}\left(x_n+\frac{a}{x_n}\right)$，求出 x_2，然后根据迭代法求根的原理，通过对运算结果 x_{n+1} 的重复代入，以逐渐逼近正确值，因此可以利用 matlab 的循环语句 while—end 语句实现重复代入；

3、限定误差不能超过 10^{-5}，因此可将 $|x_{n+1}-x_n|<10^{-5}$ 作为循环终止的条件，当 $|x_{n+1}-x_n|<10^{-5}$ 时停止循环，并将此时的 x_{n+1} 作为最终解，用数组 X 存放所求得的解；

4、输出数组 A 和数组 X，获得实验结果。

3.3.2 实现代码

对于问题 2 的具体的 MATLAB 函数实现代码（运行环境：matlabR2008a、matlabR2014a 均可以运行）如下

```
%a_sqrt.m
%用迭代法求根
A=[1,3,16,27,64,88,100,200,1000,666];    %装测试数据的数组 A，可修改此数组，从而对其他数据进行求根
for j=1:length(A)          %对测试数组 A 中的每一个数据进行迭代法求根,length(A)获得行矩阵 A 的元素个数
x(1)=A(j);                 %对 x(1)赋值
x(2)=0.5.*(x(1)+x(1)/x(1));    %求 x(2)的值
i=1;
while (abs(x(i+1)-x(i))>10^(-5))   %迭代的终止条件为前后两次求出的 x 的差的绝对值小于 10^(-5)
        x(i+2)=0.5*(x(i+1)+x(1)/x(i+1));    %迭代公式
        i=i+1;
end
X(j)=x(i+1);           %数组 X 用于存放计算结果
end
A       %输出矩阵 A
X       %输出矩阵 X,对应于矩阵 A 中的元素的求根结果
```

3.4 实验结果

在 matlab 中运行 2.3 算法与编程中所给出的代码，得到表 2 所示的结果

表 2 用迭代法求 $x=\sqrt{a}$ 的值实验结果记录表

a	1	3	16	27	64	88	100	200	1000	666
$x=\sqrt{a}$	1.000	1.732	4.000	5.196	8.000	9.381	10.00	14.14	31.62	25.81

（上表数据只保留 4 位有效数字）

代码的原始运行结果见图 2 所示

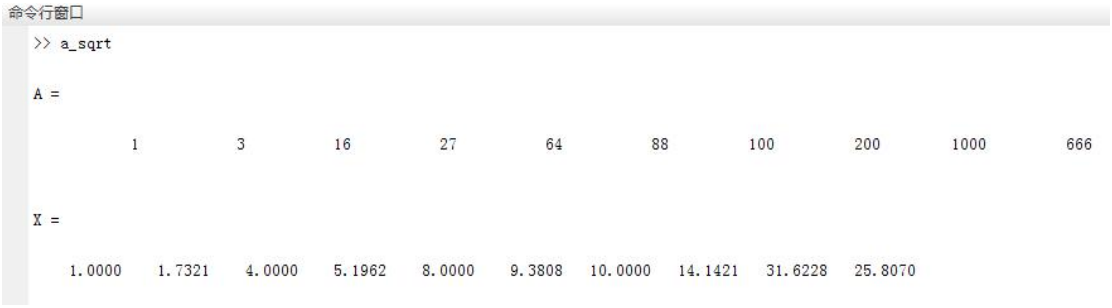


图 2 用迭代法求 $x=\sqrt{a}$ 的值实验结果截图

4 实验总结和实验感悟

4.1 实验总结

1、在本次数学实验中，我学会了使用 plot、subplot、ezplot、polar 等基本绘图函数，并且懂得使用 grid on、hold on、title、xlabel、ylabel 等辅助绘图的命令；

2、通过对问题 2 的求解，我掌握了牛顿迭代公式，并学会使用迭代法逼近方程的根，同时，在 matlab 代码中，我学会了使用 for 循环语句、while 循环语句以及循环体的嵌套。

4.2 实验感悟

这并不是我第一次接触 matlab，在 2016 年 09 月的中国大学生数学建模竞赛和 2017 年 01 月的美国大学生数学建模竞赛中，我均有接触 matlab 的使用。所以，整体上，我对 matlab 还是基本了解。但是，这次实验，还是使得我利用 matlab 绘图的能力得以加强，并且学会了牛顿迭代法求根。

同时，本次数学实验加强了我对代码规范性的认识，以及加深了我对注释的理解——如果将注释做好，那么，我们是完全可以建立自己的函数库。这样就不用花时间去记那些命令，而只需知道有什么命令，然后通过自己的函数库，实现大量代码的重用。当然，前提是自己已经提前对代码做好注释。

实验二　数据拟合

地　　点：	4 号楼　　4104 房；	**实验台号：**	66
实验日期与时间：	2017 年 4 月 5 日	**评　　分：**	
预习检查纪录：		**实验教师：**	刘小兰
电子文档存放位置：			
电子文档文件名：	卓越班-66-陈艺荣实验二		

批改意见：

1 实验目的

- 掌握MATLAB软件中进行数据显示的方式；
- 掌握MATLAB软件中进行数据拟合的方式；
- 认识Fibonacci数列，体验发现其通项公式的过程；
- 探讨利用现代计算机技术寻找数据内在规律的方法，学会用MATLAB软件实现处理和分析数据从而进行数据预测。

2 问题 1

2.1 问题描述

1. 讨论调和级数$\sum_{n=1}^{\infty}\frac{1}{n}$的变化规律，

（1）画出部分和数列$\{S_n\}$变化的折线图，观察变化规律；

（2）引入数列$H_n = S_{2n} - S_n$，作图观察其变化，猜测是否有极限；

（3）引入数列$G_n = S_{2n}$，作图观察其变化，寻找恰当的函数拟合；

（4）讨论调和级数的部分和数列的变化规律。

问题（1）要求我们用 MATLAB 绘制数列$\{S_n\}$变化的折线图，问题（2）要求我

们绘制数列 $H_n = S_{2n} - S_n$ 变化的折线图，借助图像猜测该数列是否有极限，问题（3）要求我们对数列 $G_n = S_{2n}$ 进行函数拟合。

2.2 实验原理

1、MATLAB 作图

利用 plot 函数可以实现对级数的图像绘制，其基本调用格式如下：

plot(y)

plot(x,y)

plot(x1,y1,x2,y2,……)

2、级数的表示

利用迭代的思想可以实现级数的表示，而无需求通项。问题一中的（1）、（2）、（3）问对应的级数可以用下列迭代公式表示。

$$S_n = S_{n-1} + \frac{1}{n-1} \tag{1}$$

$$H_n = S_{2n} - S_n \tag{2}$$

$$G_n = S_{2n} \tag{3}$$

3、数据拟合

在科学实验或社会活动中,人们常常需要观测很多数据的规律, 通过实验或者观测得到量 x 与 y 的一组数据对 (x_i, y_i) (i=1,2, …，N),其中 x_i 是彼此不同的。人们希望用一类与数据本质规律相适应的解析表达式， $y = f(x,c)$ 来反映量 x 与 y 之间的依赖关系，即在一定意义下“最佳”地逼近或拟合已知数据。 $f(x,c)$ 常称作拟合模型，当 c 在 f 中线性出现时，称为线性模型，否者称为非线性模型

寻求合理的近拟表达式，以反映数据变化的规律，这种方法就是数据拟合方法。数据拟合需要解决两个问题：第一，选择什么类型的函数 $\varphi(t)$ 作为拟合函数（数学模型）；第二，对于选定的拟合函数，如何确定拟合函数中的参数。

确定拟合系数的原则是最小二乘法，即所有误差的平方和取得最小值。在 MATLAB 中以多项式为目标函数作数据拟合的函数是 polyfit，它的基本使用格式为：

函数：polyfit(x,y,n)，其中参数 n 为指定多项式的阶。

2.3 算法与编程

2.3.1 编程描述

通过用 for 循环语句和迭代公式来进行循环操作获得表示级数的数组，用 plot 进行画图，通过看图猜测函数的类型，判断是对数函数，取指数后，利用 polyfit 进行拟合，判断猜测成立，从而获得级数的函数拟合。对应代码分别存放在 3 个 M 文件中：tiaohejishu2_1.m、tiaohejishu2_2.m、tiaohejishu2_3.m。

2.3.2 实现代码

1、问题一的第（1）问实现代码如下：

```
%tiaohejishu2_1.m        调和级数变化，画数列 Sn 的折线图
%            n       n 的数值
%            s       Sn 的数值大小

%    Chen Yirong  修改于 2017-04-05

clear                                %清理此前的命令行窗口
clc                                  %清理此前的工作区数据
hold on;                             %画图脚本文件模板语句，多曲线在同一坐标轴显示
grid on;                             %画图脚本文件模板语句，显示网格
%--------函数主体--------
n=1:50; %自变量范围
s(1)=1;
for i=2:50
s(i)=s(i-1)+1/i;
end
plot(n,s,'.-');         %画图
%--------函数主体--------
title('调和级数部分和 Sn 的变化规律');        %画图脚本文件模板语句，标题
xlabel('n');                         %画图脚本文件模板语句，横坐标
```

```
ylabel('Sn');                    %画图脚本文件模板语句，纵坐标
```

2、问题一的第（2）问实现代码如下：

```
%tiaohejishu2_2.m        调和级数变化,Hn 的极限猜测
%              n       n 的数值
%              s       Sn 的数值大小
%              h       Hn 的数值大小

%   Chen Yirong  修改于 2017-03-29

clear                            %清理此前的命令行窗口
clc                              %清理此前的工作区数据
hold on;                         %画图脚本文件模板语句，多曲线在同一坐标轴
显示
grid on;                         %画图脚本文件模板语句，显示网格
%--------函数主体--------
n=1:100;       %自变量范围
a=1:50;
s(1)=1;
h(1)=1/2;
for i=2:100
s(i)=s(i-1)+1/i;
end
for j=2:50
      h(j)=s(2*j)-s(j);
end
plot(a,h,'r.-');            %画图
%--------函数主体--------
title('级数 Hn 的变化规律');        %画图脚本文件模板语句，标题
xlabel('n');                    %画图脚本文件模板语句，横坐标
ylabel('Hn');                    %画图脚本文件模板语句，纵坐标
```

3、问题一的第（3）问实现代码如下：

```
%tiaohejishu2_3.m        调和级数变化
%              n       n 的数值
```

```
%           s      Sn 的数值大小
%           g      Gn 的数值大小
%           p1     输出的矩阵 p1 的两个数分别为拟合曲线 y=alnt+b 的参数 a、b
的数值！
%           R      计算实际数据和拟合数据的误差平方和，结果会在命令行窗口
显示

%   Chen Yirong  修改于 2017-03-29

clear                                %清理此前的命令行窗口
clc                                  %清理此前的工作区数据
hold on;                             %画图脚本文件模板语句，多曲线在同一坐标轴
显示
grid on;                             %画图脚本文件模板语句，显示网格
%--------函数主体--------
n=1:20000;       %自变量范围
a=1:10000;
s(1)=1;
g(1)=1/2;
for i=2:20000
s(i)=s(i-1)+1/i;
end
for j=2:10000
      g(j)=s(2*j);       %提取数据
end
%------对数型数据拟合--------%
p1=polyfit(log(a),g,1)       %对数型曲线拟合
plot(a,g,'r*',a,polyval(p1,log(a)),'b');   %画图
legend('原始数据点','对数型拟合曲线');
R=dot(g-polyval(p1,log(a)),g-polyval(p1,log(a)))     %求误差平方和
%--------函数主体-----------%
title('Gn 的曲线拟合');        %画图脚本文件模板语句，标题
xlabel('n');                    %画图脚本文件模板语句，横坐标
ylabel('Gn');                   %画图脚本文件模板语句，纵坐标
```

2.4 实验结果

1、在 matlab 中运行 tiaohejishu2_1.m，得到图 1 所示的折线图

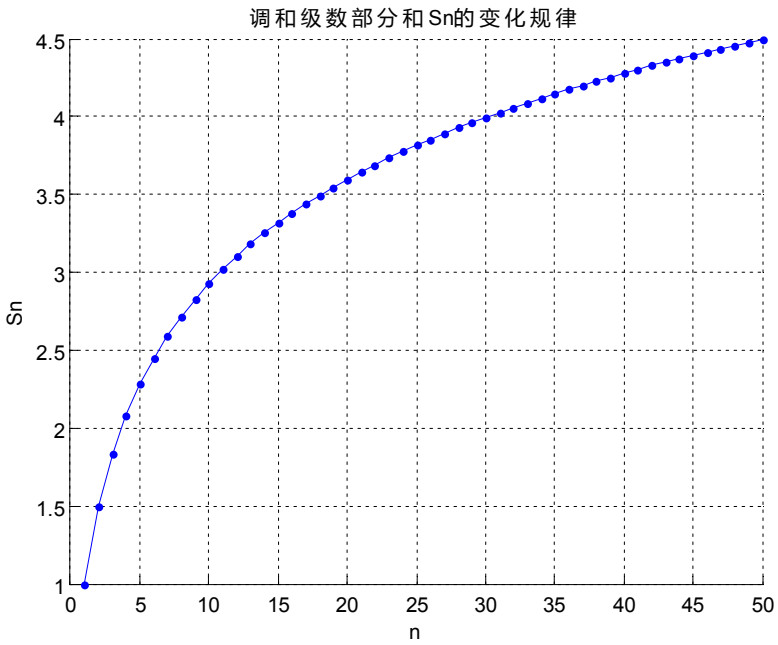


图 1 调和级数部分和 Sn 的变化规律（n=50）

修改 M 文件中的 n 的取值为 10000000，再次运行该 M 文件，得到图 2 所示的折线图

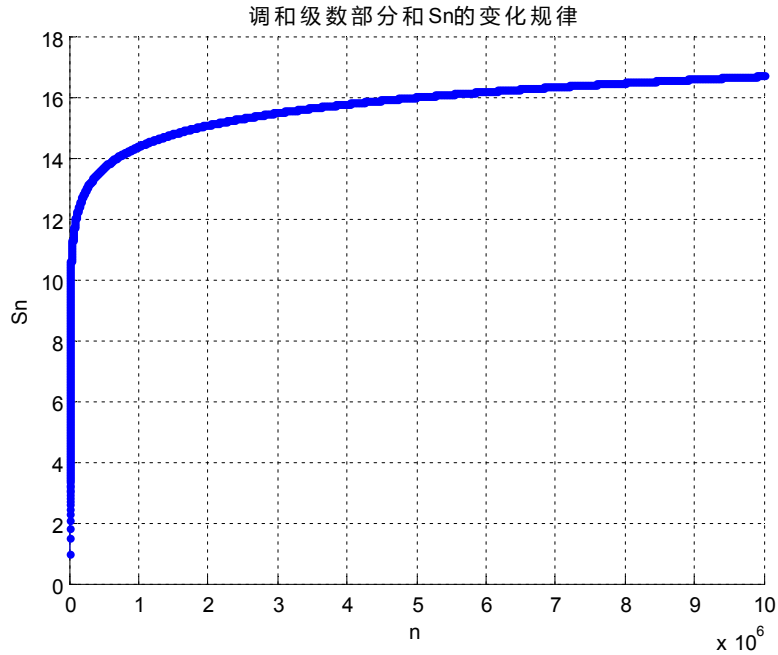


图 2 调和级数部分和 Sn 的变化规律（n=10000000）

结论：由图 1、图 2 可以知道，随着 n 值的增大，Sn 增长速度逐渐变缓，但改

变 n 值发现 Sn 并没有收敛于某一个固定值。

2、在 matlab 中运行 tiaohejishu2_2.m，得到图 3 所示的折线图

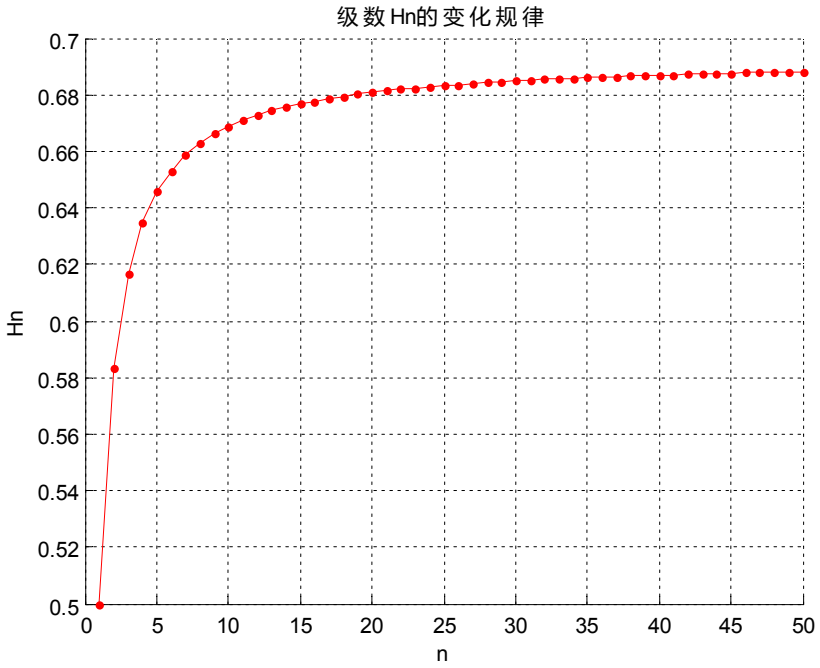


图 3 级数 Hn 的变化规律(n=50)

修改 Hn 的 n 值为 5000，再次运行该 M 文件，得到图 4 所示的折线图

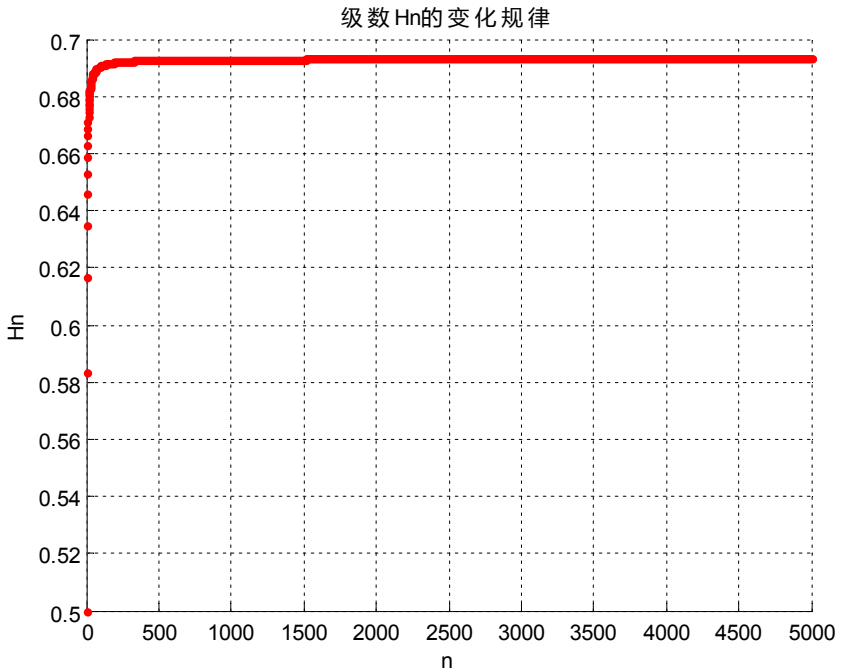


图 4 级数 Hn 的变化规律(n=5000)

结论：由图 3、图 4 可知，随着 n 值的增大级数 $H_n = S_{2n} - S_n$ 的的增长速度变缓，并且最终趋向一个在 0.6 和 0.7 之间的数值。

3、在 matlab 中运行 tiaohejishu2_3.m，得到图 5 所示的折线图

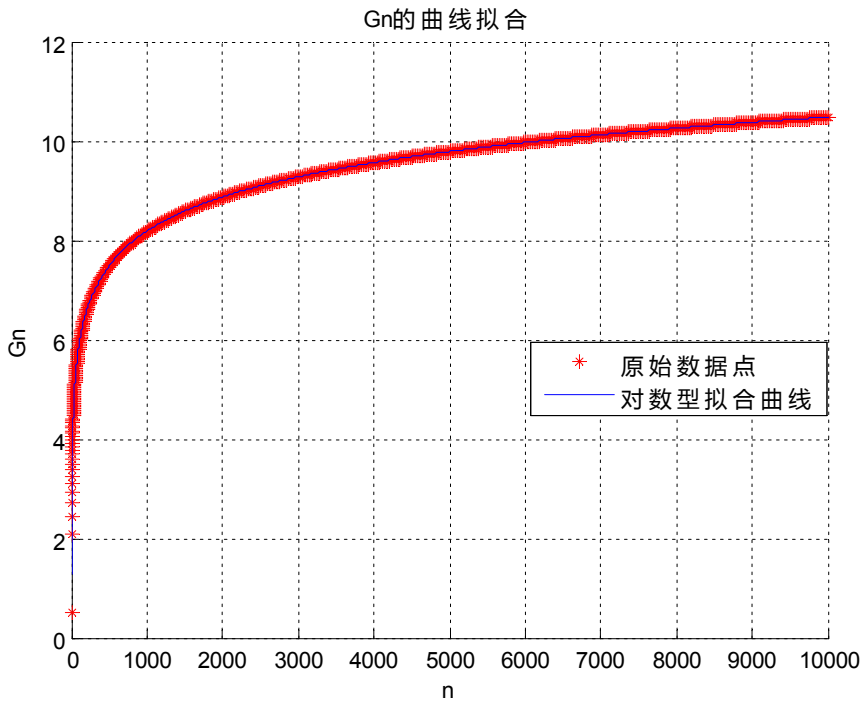


图 5 Gn 的曲线拟合

在命令行窗口输出图 6 所示的结果

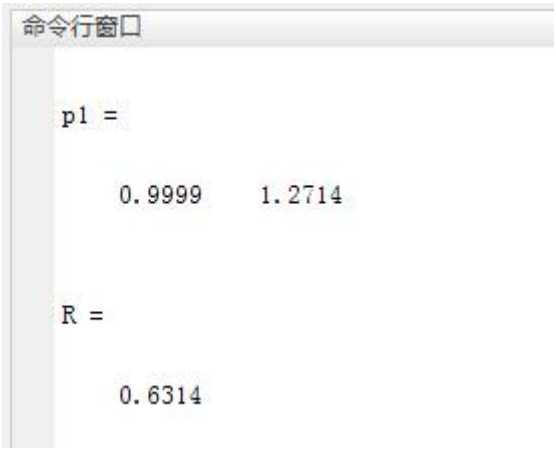


图 6 Gn 的数据拟合系数和协方差

结论：从图 5 直观看，对数型函数拟合效果十分好，实际数据与拟合数据的误差平方和 R=0.6314，表明拟合效果好。

级数 $G_n = S_{2n}$ 的拟合函数：

$$y = 0.9999 \ln n + 1.2714 \tag{4}$$

3 问题 2

3.1 问题描述

人口问题是我国最大社会问题之一，估计人口数量和发展趋势是我们制定一系列相关政策的基础。从人口统计年鉴，可查我国从 1990 年至 2010 年人口数据资料如下，试根据表中数据，分析人口增长的规律，并以此预测 2011 年和 2012 年的人口数量，然后与实际人口数量做对比，评价模型的优劣，并对我国人口政策提出建议。

表 1 不同年份我国的人口数量（万）

年份	1990	1991	1992	1993	1994	1995	1996	1997
数量	114333	115823	117171	118517	119850	121121	122389	123626
年份	1998	1999	2000	2001	2002	2003	2004	2005
数量	124761	125786	126743	127627	128453	129227	129988	130756
年份	2006	2007	2008	2009	2010			
数量	131448	132129	132802	133450	134091			

本问题要求我们利用已知的人口数据建立相应的人口增长模型，并利用该模型计算 2011 年和 2012 年我国的人口数量。

3.2 实验原理

3.2.1 数据的 n 次多项式拟合

已知函数在多个离散点处的函数值，假设拟合函数是 n 次多项式，则需要用所给数据来确定下面的函数

$$y = a_0 + a_1 x + a_2 x^2 + \cdots\cdots + a_n x^n$$

这里要做一个假设，即多项式的阶数 n 应小于所给数据的数目 m。，可得数据的 n 次多项式拟合中拟合函数的系数应满足的正规方程组如下

$$\begin{bmatrix} m & \sum_{k=1}^{m} x_k & \cdots & \sum_{k=1}^{m} x_k^{\,n} \\ \sum_{k=1}^{m} x_k & \sum_{k=1}^{m} x_k^{\,2} & \cdots & \sum_{k=1}^{m} x_k^{\,n+1} \\ & \cdots\cdots & \cdots\cdots & \\ \sum_{k=1}^{m} x_k^{\,n} & \sum_{k=1}^{m} x_k^{\,n+1} & \cdots & \sum_{k=1}^{m} x_k^{\,2n} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} \sum_{k=1}^{m} y_k \\ \sum_{k=1}^{m} x_k y_k \\ \vdots \\ \sum_{k=1}^{m} x_k^{\,n} y_k \end{bmatrix} \tag{5}$$

从这一方程组可以看出，线性拟合方法和二次拟合方法是多项式拟合的特殊情况。从算法上看，数据最小二乘拟合的多项式方法是解一个超定方程组

$$\begin{cases} a_0 + a_1 x_1 + a_2 x_1^{\,2} + \cdots + a_n x_1^{\,n} = y_1 \\ a_0 + a_1 x_2 + a_2 x_2^{\,2} + \cdots + a_n x_2^{\,n} = y_2 \quad (m > n) \\ \cdots\cdots\cdots\cdots \\ a_0 + a_1 x_m + a_2 x_m^{\,2} + \cdots + a_n x_m^{\,n} = y_m \end{cases} \tag{6}$$

的最小二乘解。而多项式拟合所引出的正规方程组恰好是用超定方程组的系数矩阵的转置矩阵去左乘超定方程组左、右两端所得。正规方程组的系数矩阵是一个病态矩阵，这类方程组被称为病态方程组。当系数矩阵或者是右端向量有微小的误差时，可能引起方程组准确解有很大的误差。为了避免求解这样的线性方程组，在做多项式拟合时可以将多项式中的各次幂函数做正交化变换，使得所推出的正规方程的系数矩阵是对角矩阵。

在 MATLAB 中以多项式为目标函数作数据拟合的函数是 polyfit，它的基本使用格式为：

函数：polyfit(x,y,n)，其中参数 n 为指定多项式的阶。

3.3 算法与编程

3.3.1 编程描述

用 polyfit 函数进行 2 阶多项式拟合、4 阶多项式拟合、5 阶多项式拟合，用 R=dot(y-polyval(p,t),y-polyval(p,t)) 计算拟合残差，再用 polyval 函数预测 2011 和 2012 年的人口。

其中，人口数据存放在一个命名为 renkoushuju.txt 的文件当中，设定 1990

年为第 0 年，如图 7 所示。在 matlab 中使用 fopen 函数和 textsan 函数获取相关数据。

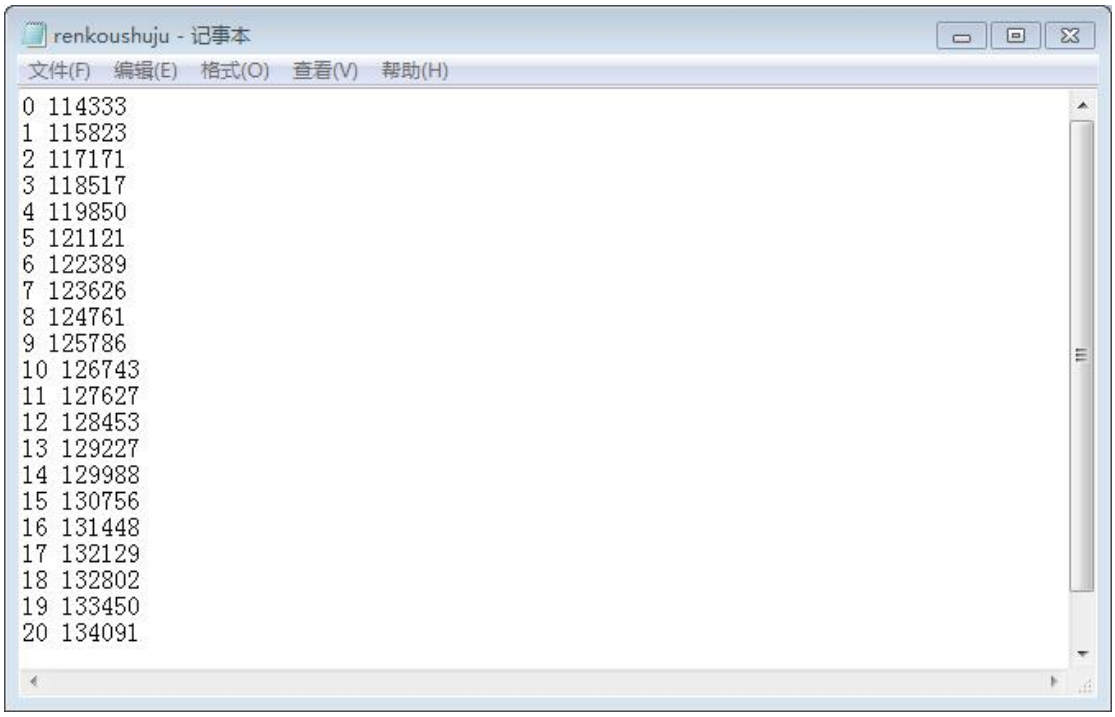


图 7 人口数据存放文件

3.3.2 实现代码

对于问题 2 的实现代码如下：

```
%renkouyuce.m        我国人口增长规律观测
%                C   年份和人口数据矩阵
%             year   年份
%              num    人口
%                a   2011 年预测人口数
%                b   2012 年预测人口数

%Chen Yirong  修改于 2017 年 04 月 05 日

clc;clear all                %清除所有
%提取 1990 年到 2010 年的人口数据，数据存储在 renkoushuju.txt 文件中
renkou=fopen('renkoushuju.txt');   %打开数据总文件
A=textscan(renkou,'%f %f');          %把每一列的数据读入到读入到单元数组 A
中
```

```
C=[A{1} A{2}];                          %从单元数组 R 中提取每列数据赋值给矩阵 C
n=max(size(C));                         %确定读入数据的年份数目
year=C(:,1);num=C(:,2);                 %赋值

%进行 2 阶、4 阶、5 阶拟合
p2= polyfit(year,num,2)                 %2 阶拟合
p4= polyfit(year,num,4)                 %4 阶拟合
p5= polyfit(year,num,5)                 %5 阶拟合
R5= dot(num-polyval(p5,year),num-polyval(p5,year)) %计算拟合残差
R4= dot(num-polyval(p4,year),num-polyval(p4,year)) %计算拟合残差
R2= dot(num-polyval(p2,year),num-polyval(p2,year)) %计算拟合残差

%绘制原始数据和拟合曲线图
hold on;
xlabel('year','color','b');         %设置横坐标名
ylabel('num','color','b');          %设置纵坐标名
title('1990-2010 人口增长曲线','color','m');     %设置标题
grid on          %网格线
plot(year,num,'r*',year,polyval(p2,year),year,polyval(p4,year),year,polyval(p5,year))
legend('我国人口数量','2 阶拟合', '4 阶拟合','5 阶拟合')

%人口预测
a=polyval(p5,21)       %预测 2011 年人口数量
b=polyval(p5,22)       %预测 2012 年人口数量
```

3.4 实验结果

在 matlab 中运行 renkouyuce.m 文件，得到 2、4、5 阶多项式拟合的结果如下：

2 阶多项式拟合函数：$y=10^5\times(-0.0003x^2+0.0148x+1.1437)$ (7)

4 阶多项式拟合函数：$y=10^5\times(0.0142x+1.1434)$ (8)

5 阶多项式拟合函数：$y=10^5\times(-0.0001x^3+0.0003x^3+0.0135x+1.1438)$ (9)

（**单位**：万人，下同）

得到原始数据点和拟合曲线图像如图 8 所示

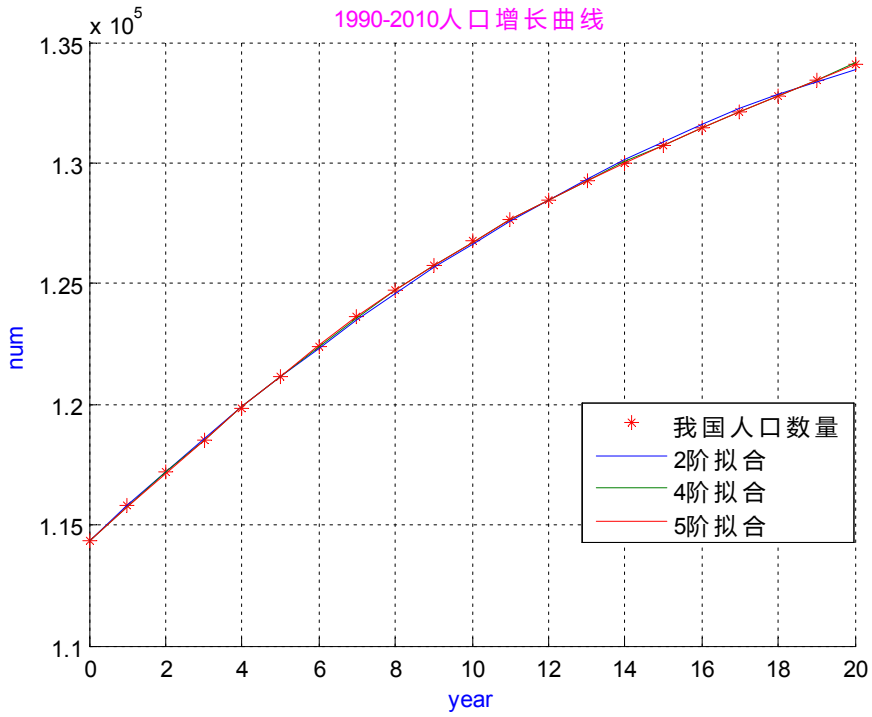


图 8 1990-2010 年我国人口增长曲线及其拟合曲线

在命令行窗口得到 2 阶多项式拟合、4 阶多项式拟合和 5 阶多项式拟合的误差平方和 R2、R4、R5 的数值如下图。

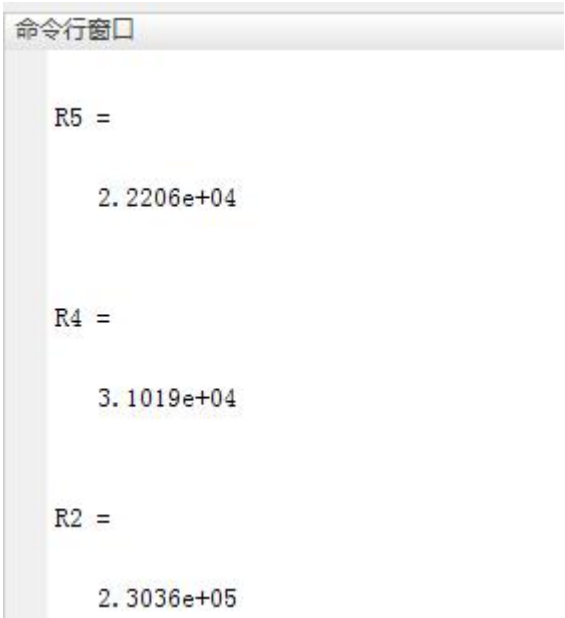


图 9 2 阶、4 阶、5 阶多项式拟合的误差平方和

由图 9 可知，5 阶多项式拟合的误差平方和效果最好，因此最终采用 5 阶多项式拟合函数来预测 2011 年及 2012 年我国人口数量，结果如图 10 所示

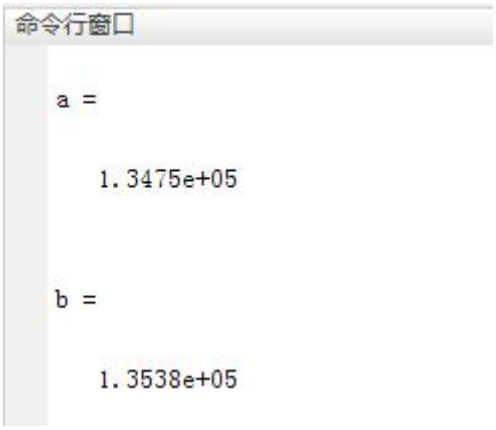


图 10 2011 年及 2012 年我国人口数量预测值

也就是预测到我国 2011 年人口数量为 134750 万人，2012 年人口数量为 135380 万人。而根据国家统计局的统计数据，我国 2011 年和 2012 年实际人口分别为：134735 万人和 135404 万人，见图 11 所示。

(a) (b)

图 11 2011 年及 2012 年我国人口数量实际值

绝对误差分别为：15 万人（2011 年）以及 24 万人（2012 年），这说明预测值比较精确。说明该模型比较好。同时，这也提醒我们，以 2010 年为临界点，我国人口增长依然比较快，在 2011 年到 2015 年这段时间，依然要坚持计划生育这个基本国策不动摇。

4 实验总结和实验感悟

4.1 实验总结

1、在本次数学实验中，我学会了利用 matlab 进行数据拟合并借助数据拟合进行数据预测；

2、通过问题一，我认识到调和级数$\{S_n\}$是没有极限的，尽管它的增长越来

越慢；

3、利用多项式拟合，应该对不同阶数的拟合效果进行比较，选择最佳的拟合函数，从而帮助进行数据预测；

4、在本次实验中，我对 M 文件建立采用模板化的思想，也就是像下面这样的模板，从而大大加快编程速度，并且有利于后续代码修改和重复利用。

```
%muban.m            调和级数变化，画数列 Sn 的折线图
%           n       n 的数值
%           s       Sn 的数值大小

%   Chen Yirong  修改于 2017-04-05

clear                                   %清理此前的命令行窗口
clc                                     %清理此前的工作区数据
hold on;                                %画图脚本文件模板语句，多曲线在同一坐标轴显示
grid on;                                %画图脚本文件模板语句，显示网格
%--------函数主体--------

%--------函数主体--------
title('调和级数部分和 Sn 的变化规律');        %画图脚本文件模板语句，标题
xlabel('n');                            %画图脚本文件模板语句，横坐标
ylabel('Sn');                           %画图脚本文件模板语句，纵坐标
```

4.2 实验感悟

我在做这次实验存在几个问题：

第一是在开始进行实验的时候，我没有考虑改变Sn、Hn的n值上限来绘制多个曲线，借助这个来确定它们是否存在极限。实践表明这是一个不错的方法。

第二是遇到行列式条件出现错误，刚开始的时候，不懂得将1990到2010转变成0到20。

总体来说本次数学实验我做的不错，在学习中我深深体会到实践的重要性。把基本的知识看过之后，就需要找一个实际的程序来动手编一下，在编程的过程中学习，程序需要什么知识再去查阅补充相关的知识。同时，我深刻地认识到“读懂别人的代码，能借鉴别人的代码”是十分重要的。

实验三　微分方程

地　　点：	4 号楼　　4104 房；	**实验台号：**	66
实验日期与时间：	2017 年 04 月 15 日	**评　　分：**	
预习检查纪录：		**实验教师：**	刘小兰
电子文档存放位置：			
电子文档文件名：	卓越班-66-陈艺荣实验三		

批改意见：

1 实验目的

- 了解求微分方程解析解的方法
- 了解求微分方程数值解的方法
- 学习单自由度阻尼系统，观察阻尼系数对系统的影响
- 模拟弹簧振动，学习实时动画编程的原理
- 了解 dsolve，ode45 指令的使用方法
- 了解 Simulink 仿真的设计思想

2 问题 1

2.1 问题描述

用 dsolve 函数求解下列微分方程

（2）$\begin{cases} y''(x) = y'(x) + 2y(x) \\ y(0) = 1,\ y'(0) = 0 \end{cases}$

2.2 实验原理

1、微分方程的解析解

解析解指在一定条件下能够以数学表达式直接表达出来的的解。

2、dsolve 命令

用法：dsolve(‘equation’,’condition’,’v’)

说明：

（1） equation 是方程式，condition 是条件，v 是自变量（缺省为 t）

（2）若不带条件，则解中带积分常数

（3）如果没有显式解，则系统尝试给出隐式解

（4）如果无隐式解，则返回空符号。

格式：

（1）y’表示为 Dy，y’’表示为 D2y，依次类推

（2）有多个方程或多个条件时，写多个相应的参数即可。

2.3 算法与编程

2.3.1 编程描述

在 dsolve(‘equation’,’condition’,’v’)函数中，分别编辑输入微分方程、约束条件、自变量，然后运行 M 文件。

2.3.2 实现代码

```
%shiyan3_1.m          用 dsolve 函数求解微分方程
%题目说明             用 dsolve 函数求解微分方程
%          y          输出的函数因变量
%          x          输出的函数自变量

%   Chen Yirong  修改于 2017-04-15
%   代码编辑 matlab 版本：MATLAB R2014a

clear,clc;     %刷新工作区
```

```
%--------函数主体--------%

y=dsolve('D2y=Dy+2*y','y(0)=1,Dy(0)=0','x')
%D2y=Dy+2*y 为方程式，y(0)=1,Dy(0)=0 为约束条件
%--------函数主体--------%
```

2.4 实验结果

在 matlab 中运行 shiyan3_1，得到图 1 所示的结果

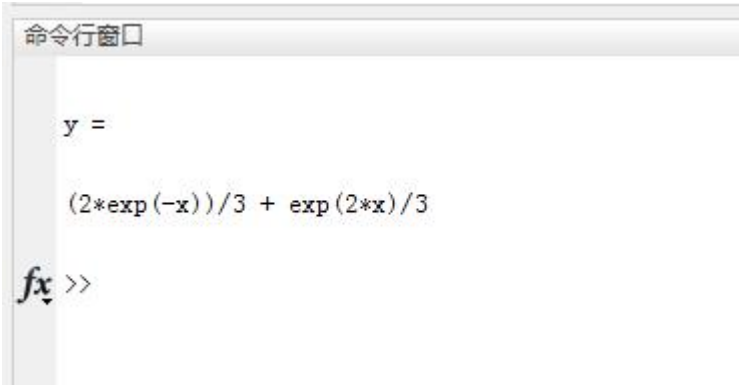


图 1 问题 1 的求解结果

因此，微分方程

$$\begin{cases} y''(x) = y'(x) + 2y(x) \\ y(0) = 1,\ y'(0) = 0 \end{cases}$$

的特解为：

$$y = \frac{2e^{-x} + e^{2x}}{3}$$

3 问题 2

3.1 问题描述

我缉私雷达发现，距离 d 处有一走私船正以匀速 a 沿直线行驶，缉私舰立即以最大速度（匀速 v）追赶。若用雷达进行跟踪，保持船的瞬时速度方向始终指向走私船，则缉私舰的运动轨迹是怎么的？是否能够追上走私船？如果能追上，需要多长时间？

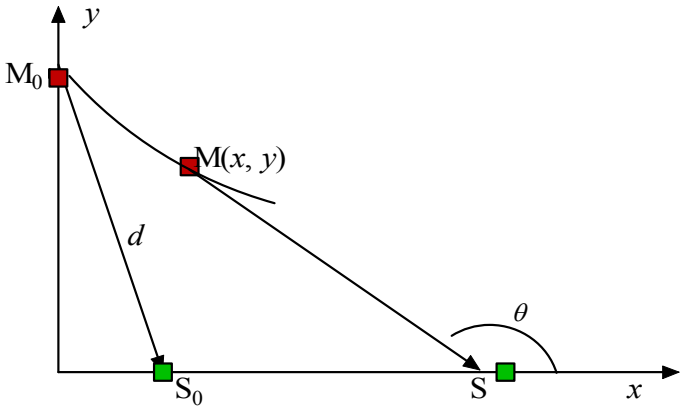


图 2 缉私船与走私船的运动模型

3.2 实验原理

在生产和科研中所处理的微分方程往往很复杂且大多得不出一般解。而在实际上对初值问题，一般是要求得到解在若干个点上满足规定精确度的近似值，或者得到一个满足精确度要求的便于计算的表达式。

一阶微分方程的数值解 ode45 命令

用法：[t,Y]=ode45(odefun,tspan,y0)

说明：

（1） odefun 是待求解一阶微分方程或方程组的句柄，对应一个 M 文件

（2）tspan 求解区间，y0 为初值

（3）返回值 t 为自变量的数据列

（4）返回值 Y 一般是矩阵，每列对应一个待解变量的数据列

（5）对方程组，待解变量，导数，初始值等，全部用数组表示

3.3 算法与编程

3.3.1 编程描述

对缉私船追上走私船的问题，为了便于分析，我们用数值法模拟，然后用MATLAB 求解。

1、基本假设

假设 1　走私船正以匀速v_1沿 x 轴正方向直线行驶

假设 2　缉私船的速度方向始终指向走私船并立即以最大速度（匀速v_0）追赶

假设 3　不考虑风浪的影响，走私船和缉私船在这个过程中没有发生故障

2、符号说明

表 1 问题 2 符号说明

符号	符号说明	单位
M(x,y)	缉私船的位置函数	km
S_0	走私船的初始位置，与坐标原点的距离	km
S(x,y)	走私船的位置函数	km
d	初始时，缉私船与走私船的距离	km
θ	缉私船运动方向与走私船运动方向的夹角	rad
t	缉私船追上走私船用到的时间	h
v_0	缉私船的最大运动速度值	km/h
v_1	走私船的运动速度值	km/h

3、模型建立

设缉私船航行的曲线方程为$y=f(x)$，在时刻 t 时缉私船位于$M(x,y)$，走私船位于 S，S 点坐标为$(v_1t+S_0,0)$，直线 PQ 与缉私船的路线相切，缉私船的方向始终指向走私船。因此，可以建立以下微分方程关系：

$$\frac{dy}{dx}=\frac{0-y}{S_0+v_1t-x} \tag{1}$$

对于(1)式，两边同时对 y 求导，得到(2)式：

$$y\frac{d^2x}{dy^2}=-a\frac{dt}{dy} \tag{2}$$

因为任意时刻，缉私船的速度满足图 3 所示关系：

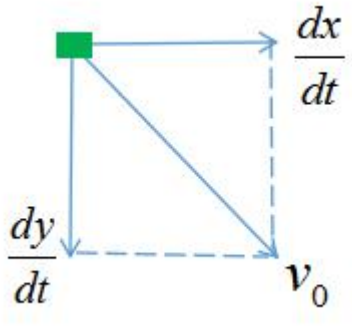


图 3 速度分解示意图

因此，有下列关系：

$$v_0^2=(\frac{dx}{dt})^2+(\frac{dy}{dt})^2 \tag{3}$$

结合方程(2)、(3)，可以得到以下模型：

$$y\frac{d^2x}{dy^2}=\frac{v_1}{v_0}\sqrt{1+(\frac{dx}{dy})^2} \tag{4}$$

进一步地，该追赶微分模型的初始条件为：

$$\begin{cases} x(y_0)=0 \\ \left.\dfrac{dx}{dy}\right|_{y=y_0}=-\dfrac{S_0}{y_0} \end{cases} \tag{5}$$

其中，$y_0=\sqrt{d^2-S_0^2}$ 为缉私船的初始纵坐标。

最终，可以获得完整的追赶微分方程模型如下：

$$\begin{cases} x_1'=x_2 \\ x_2'=\dfrac{k}{y}\sqrt{1+x_2^2} \end{cases} \tag{6}$$

上式中，k 是一个参数，定义如下：

$$k=\frac{v_1}{v_0} \tag{7}$$

在本模型中，通过预先计算参数 k 可以估测追捕的路线以及确定能否追捕的问题。

在求解时，设定 $y_0=50km$ 。

3.3.2 实现代码

对于问题 2，建立 23 个 M 文件求解，其中 shiyan2_2.m 为脚本命令文件，fun1 为求数值解函数，fun2 为求解多组数据并绘图的函数。

1、shiyan2_2.m 代码如下：

```
%shiyan2_2.m        数学实验三第 2 题 MATLAB 求解
%题目说明           题目要求利用 matlab 求解微分方程的数值解的方法求解追捕问题
%本 M 文件调用了 fun2 求解数值解并输出结果

%   Chen Yirong  修改于 2017-04-15
%   代码编辑 matlab 版本：MATLAB R2014a

clear,clc;     %刷新工作区
%--------函数主体--------%

vc=[0.2,0.4,0.7,0.9];     %不同的 k 值输入
fun2(vc);          %不同的 k 值输出

%--------函数主体--------%
```

2、fun2.m 代码如下：

```
function fun2(vc)
%fun2.m       求解数值解并绘图
%vc           走私船与缉私船的速度比值
```

```
%本函数调用了 fun1 函数求数值解

%    Chen Yirong  修改于 2017-04-15
%    代码编辑 matlab 版本：MATLAB R2014a

global k;       %定义一个全局变量,k 为走私船与缉私船的速度比值
hold on          %锁住图形窗口
tspan=50:-0.1:0.1;       %求解范围
color='rgyb';                    %曲线颜色设定
for i=1:length(vc);            %求解不同 k 值对应的数值解
            k=vc(i);
            [y,x]= ode45('fun1',tspan,[0,0]); %求数值解
             plot(x(:,1),y,color(i));       %画图并配色
end
% axis([-0.1,5,0,1.1]);     %画坐标系
y1=0:0;
x1=0:100;
plot(x1,y1,'b*');
legend('k=0.2  缉私船运动轨迹','k=0.4  缉私船运动轨迹','k=0.7  缉私船运动轨迹
','k=0.9  缉私船运动轨迹','走私船运动轨迹')   %为不同的曲线标记说明
xlabel('x');
ylabel('y');
hold off
end
```

3、fun1.m 代码如下：

```
function dx=fun1(y,x)
%fun1.m
%x          坐标系的横坐标
%y          坐标系的纵坐标
```

```
%    Chen Yirong  修改于 2017-04-15
%    代码编辑 matlab 版本：MATLAB R2014a

global k;                  %定义一个全局变量,k 为走私船与缉私船的速度比值
dx=zeros(2,1);          %  存储导数的 2x1 矩阵
dx(1)=x(2);                  %第一个方程
dx(2)=k*sqrt(1+x(2)*x(2))/y; %第二个方程
end
```

3.4 实验结果

在 matlab 命令窗口中运行 shiyan3_2，得到图 4 所示结果

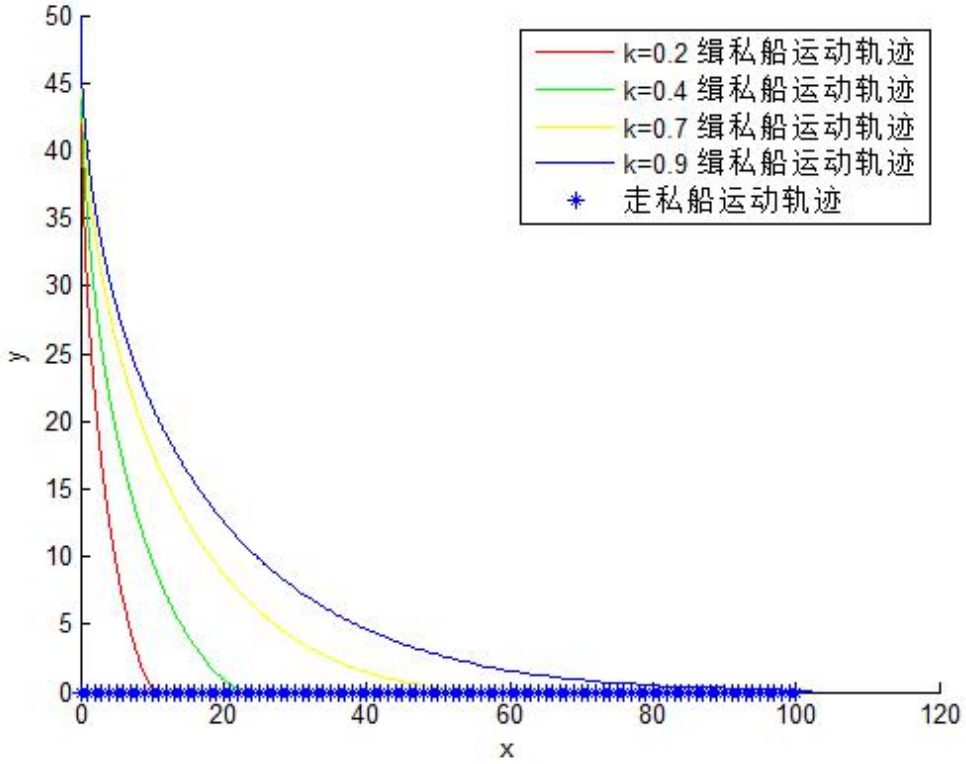


图 4 不同速度比值下的缉私船与走私船的追赶情况

分析如下：

1、当$k=\frac{a}{v}\neq 1$时，方程（6）的解为

$$x(y)=x_0+\frac{1}{2}\left(\frac{Cy^{k+1}}{k+1}+\frac{y^{-k+1}}{C(k-1)}\right)-\frac{1}{2}\left(\frac{Cy_0{}^{k+1}}{k+1}+\frac{y_0{}^{-k+1}}{C(k-1)}\right) \tag{8}$$

2、当$k=\frac{a}{v}$=1时，方程的解为

$$x(y)=x_0+\frac{C}{4}(y^2-y_0{}^2)-\frac{1}{2C}\ln\frac{y}{y_0} \tag{9}$$

其中，$C=\frac{x_0+\sqrt{x_0{}^2+y_0{}^2}}{y_0{}^{1+k}}$

分析可知，

1、当$k\geq 1$时，即缉私船的速度小于走私船的速度时，$\lim_{y\to 0}x=\infty$，因此缉私船不能追上走私船。

2、当 k<1 时，即即缉私船的速度大于走私船的速度时，令 y=0, 解得走私船行驶距离为

$$x=\frac{x_0a^2-av\sqrt{x_0{}^2+y_0{}^2}}{a^2-v^2}=a\frac{x_0a-vd}{a^2-v^2} \tag{10}$$

追上所需时间为

$$T=\frac{x_0a-vd}{a^2-v^2} \tag{11}$$

4 实验总结和实验感悟

4.1 实验总结

1、通过本次实验，我掌握了 dsolve 命令的使用。

2、通过本次实验，我掌握了 ode54 命令的使用。

4.2 实验感悟

这次实验难度比以往的要大，主要是出现了解决应用问题的难点，需要自己理解题意后，找到该图具有的内部条件关系，再列出微分方程，最后利用函数解决。理论上和数学建模十分相似，因此整个求解过程，我采用了自己参加数学建模国赛和美赛时候的经验进行求解的，在这个过程中，建模能力进一步得到加强。

实验四　迭代与分形

地　　点：	4 号楼　　4104 房；	**实验台号：**	66
实验日期与时间：	2017 年 05 月 03 日	**评　　分：**	
预习检查纪录：		**实验教师：**	刘小兰
电子文档存放位置：			
电子文档文件名：	卓越班-66-陈艺荣实验四		

批改意见：

1 实验目的

- 了解分形几何的基本特性；
- 了解通过迭代方式产生分形图的方法；
- 欣赏美妙的分形艺术；
- 掌握使用matlab进行编程常用的程序结构；
- 掌握matlab软件中plot, fill等函数的基本用法。

2 问题 1

2.1 问题描述

1、对一个等边三角形，每条边按照 Koch 曲线的方式进行迭代，产生的分形图称为 Koch 雪花。编制程序绘制出它的图形，并计算 Koch 雪花的面积。

2.2 实验原理

1、Koch 曲线

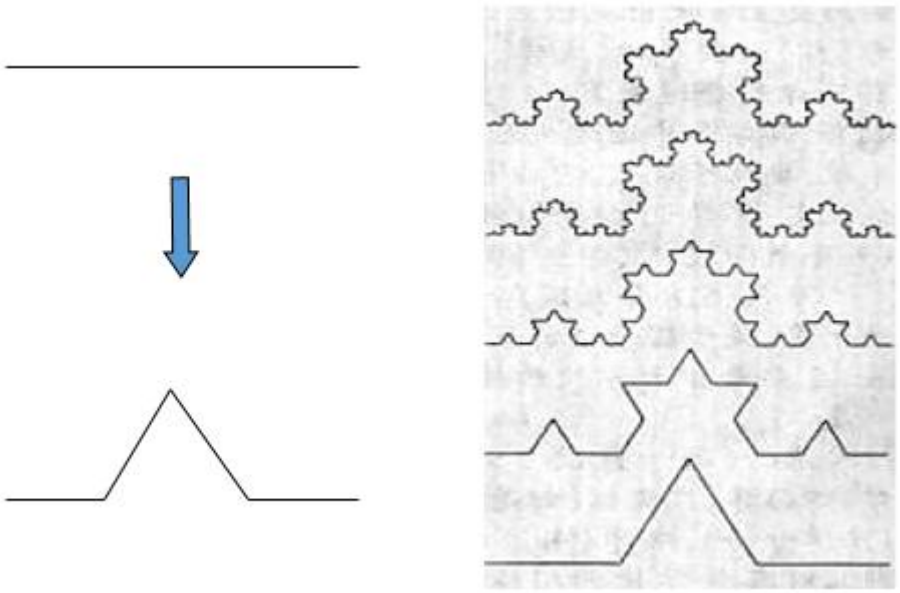
将一根线段三等份，并将中间段用以该段为边的等边三角形的另外两边替代，如图 1 所示。给定线段 AB，科赫曲线可以由以下步骤生成：

（1）将线段分成三等份（AC,CD,DB）；

（2）以 CD 为底，向外（内外随意）画一个等边三角形 DMC；

（3）将线段 CD 移去；

（4）分别对 AC,CM,MD,DB 重复 1~3。



(a)Koch 曲线一次迭代的示意图　　(b)Koch 曲线经过若干次迭代的示意图

图 1 Koch 曲线的形成过程

2、Koch 雪花迭代过程

对一个等边三角形，每条边按照 Koch 曲线的方式进行迭代，产生的分形图称为 Koch 雪花，如图 2 所示。步骤如下：

（1）任意画一个等边三角形，并把每一边三等分；

（2）取三等分后的一边中间一段为边向外作等边三角形，并把这“中间一段”擦掉；

（3）重复上述两步，画出更小的等边三角形。

（4）一直重复，直到无穷，所画出的曲线叫做 Koch 雪花。

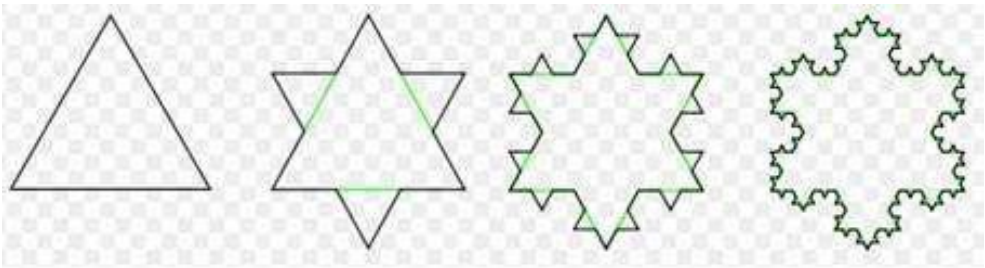


图 2 Koch 雪花产生过程示意图

3、Koch 雪花面积计算公式

经过 k 次迭代的 Koch 雪花图面积计算公式如下：

$$S(k)=\sum_{i=0}^{k}\frac{\sqrt{3}}{4}((\frac{1}{3})^{i}N)^{2}\cdot 3^{i} \tag{1}$$

式中，N 为第 0 次迭代时的正三角形的边长。

2.3 算法与编程

2.3.1 编程描述

根据迭代原理，在等边三角形的三条边上分别按照科赫曲线的迭代规律进行转化，即可形成 Koch 雪花。Koch 曲线的迭代规律为将一条线段三等份，并将中间段用以该段为边的等边三角形的另外两边替代。每次迭代在原图形的每条边上重新生成一个边长为原边长三分之一的小等边三角形。据此可以计算出 Koch 雪花的面积。

仿照 Koch 曲线代码对三角形的每条边进行 Koch 曲线化，函数的输入参数有三角形的迭代次数 k 和三角形的边长 N，输出 Koch 雪花图形以及返回 Koch 雪花所围面积 S。

2.3.2 实现代码

1、Shiyan4.m，该 M 文件是脚本文件，通过调用 plotkoch 实现绘制雪花图并且计算面积。

```
%shiyan4.m        数学实验 4Koch 雪花图测试脚本文件
%文件说明：本文件是测试用 M 文件
%              S        Koch 雪花图的面积
%              其中 S(i)表示经过 i-1 次迭代得到的雪花图的面积

%   Chen Yirong  修改于 2017-04-26
%   代码编辑 matlab 版本：MATLAB R2014a

clear                                %清理此前的命令行窗口
clc                                  %清理此前的工作区数据
%--------函数主体--------
Num=0:7;            %迭代次数矩阵
S=zeros(1,8);   %存储迭代次数不同的雪花图的面积，初始赋值为 0
S(1)=plotkoch(0,10);   %0 次迭代
```

```
S(2)=plotkoch(1,10);   %1 次迭代
S(3)=plotkoch(2,10);   %2 次迭代
S(4)=plotkoch(3,10);   %3 次迭代
S(5)=plotkoch(4,10);   %4 次迭代
S(6)=plotkoch(5,10);   %5 次迭代
S(7)=plotkoch(6,10);   %6 次迭代
S(8)=plotkoch(7,10);   %7 次迭代
figure('Name','实验 4 不同 k 值的 Koch 雪花图面积','Position',[680,80,600,600]);   %创建 1 个新显示图形的窗口
stem(Num,S);     %绘制迭代次数和 Koch 雪花图的面积关系散点图
title('实验 4 不同 k 值下的 Koch 雪花图面积');   %绘制标题
xlabel('k');     %绘制横坐标
ylabel('S');     %绘制纵坐标
%--------函数主体--------
```

2、plotkoch.m 文件是函数文件，该函数有两个参数，为 k 和 N，它们分别代表迭代次数和初始正三角形的边长。该函数返回的是 Koch 雪花图的计算面积。

```
function S=plotkoch(k,N)     %k 为迭代次数,N 为正三角形边长
%function S=plotkoch(k,N)
%plotkoch.m       显示迭代 k 次后的 Koch 雪花图，并且返回计算面积，数学实验 4 第 1 题实现函数
%本函数说明：实验证明,调用本函数，迭代次数 k 最好小于 10，否则运行时间会比较久，甚至可能使程序崩溃
%          k          迭代次数
%          N          迭代的初始三角形的边长
%          S          计算得到的 Koch 雪花图面积

%   Chen Yirong  修改于 2017-04-26
%   代码编辑 matlab 版本：MATLAB R2014a

%初始正三角形的三个顶点
p01=[0,0];
p02=[N/2,sqrt(3)*N/2];
p03=[N,0];
S=0;           % S 为面积，开始设为 0
%迭代
for dnum=0:2           %依次对 3 条边进行 Koch 曲线运算
    if dnum==0;
        p=[p01;p02];
    else if dnum==1;
            p=[p02;p03];
        else if dnum==2;
                p=[p03;p01];
```

```
                end
            end
        end
    n=1;    %存放线段的数量，初始值为 1
    A=[cos(pi/3),-sin(pi/3);sin(pi/3),cos(pi/3)]; %变换矩阵 用于计算新的结点
        for s=1:k
            j=0;      % j 为行数
            for i=1:n
                q1=p(i,:);    %目前线段的起点坐标
                q2=p(i+1,:); %目前线段的终点坐标
                d=(q2-q1)/3;
                j=j+1;r(j,:)=q1;          %原起点存入 r
                j=j+1;r(j,:)=q1+d;        %新 1 点存入 r
                j=j+1;r(j,:)=q1+d+d*A'; %新 2 点存入 r
                j=j+1;r(j,:)=q1+2*d;      %新 3 点存入 r
            end
            n=4*n;        %全部线段迭代一次后，线段数量乘 4
            clear p      %清空 p ，注意：最后一个终点 q2 不在 r 中
            p=[r;q2];   %一条边的全部结点
            clear r
        end
        if dnum==0;    %把第一条边的全部结点放在 a
            a=p;
        else if dnum==1; %把第二条边的全部结点放在 b
                b=p;
            else if dnum==2;   %把第三条边的全部结点放在 c
                    c=p;
                end
            end
        end
end
%绘图
all=[a;b;c];                    %三条边全部结点放入 all
figure('Name','实验 4 第 1 题 Koch 雪花图','Position',[60,80,600,600]);   %创建第 1
个显示图形的窗口
plot(all(:,1),all(:,2),'b')    %连接各个结点
fill(all(:,1),all(:,2),'b')%填充所围区域
title(k); %在图上标题栏显示迭代次数 k
axis equal

%计算
for i=0:k
    S = S + (3^(0.5-i))*0.25*(N^2);   %计算 Koch 雪花的面积
end
```

2.4 实验结果

在 matlab 中运行 shiyan4.m，整个屏幕显示如图 3 所示。

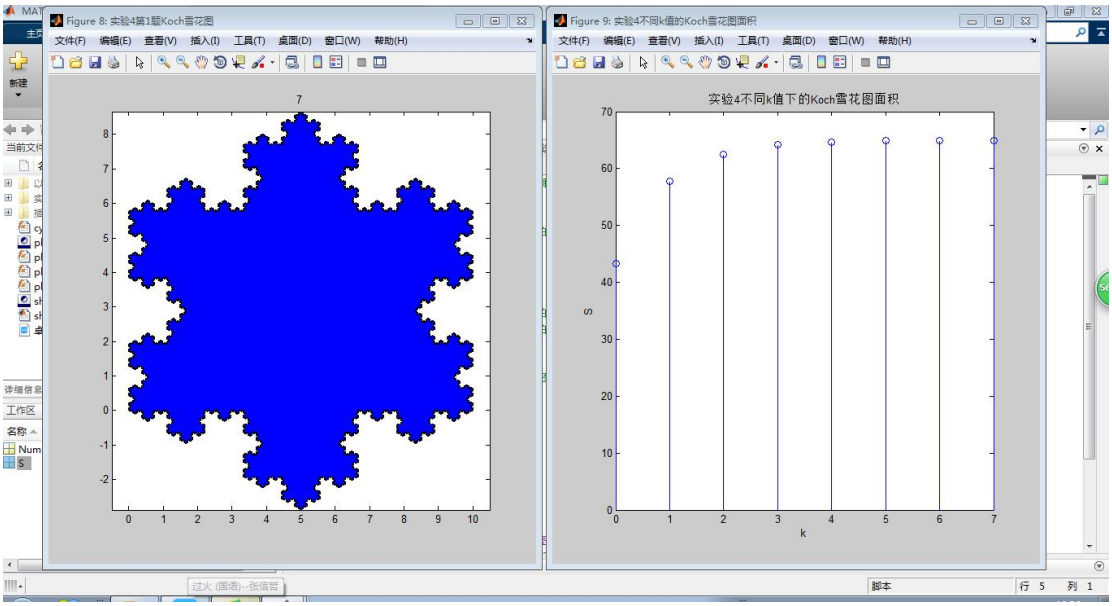


图 3 shiyan4.m 代码运行效果图

k=0 时的 Koch 雪花图：

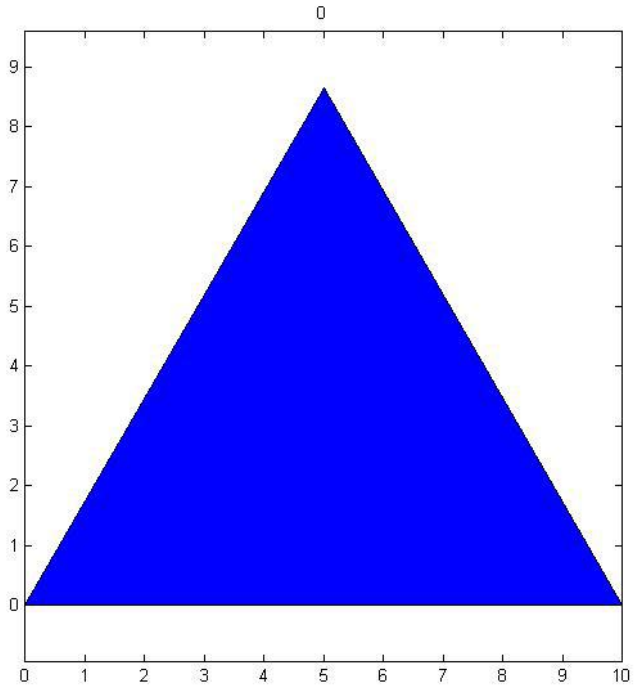


图 4 k=0 时候的 Koch 雪花图

k=1 时的 Koch 雪花图：

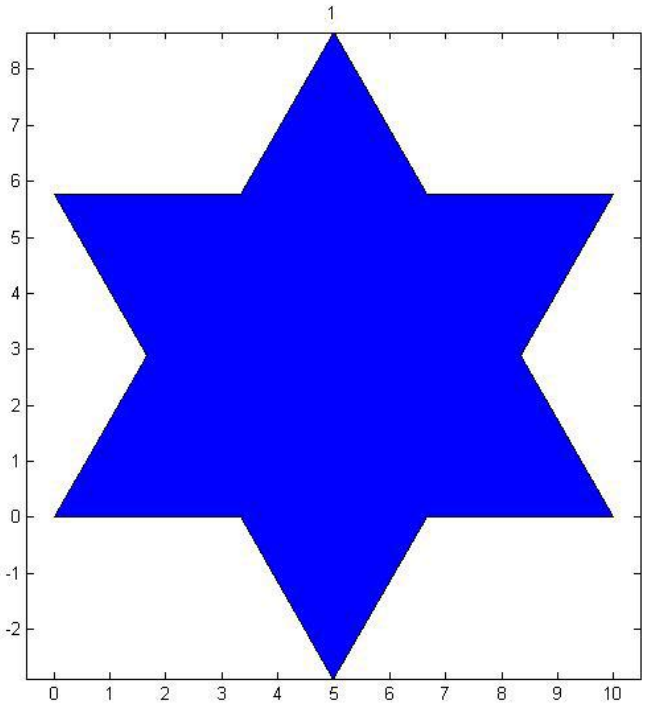


图 5 k=1 时候的 Koch 雪花图

k=3 时的 Koch 雪花图：

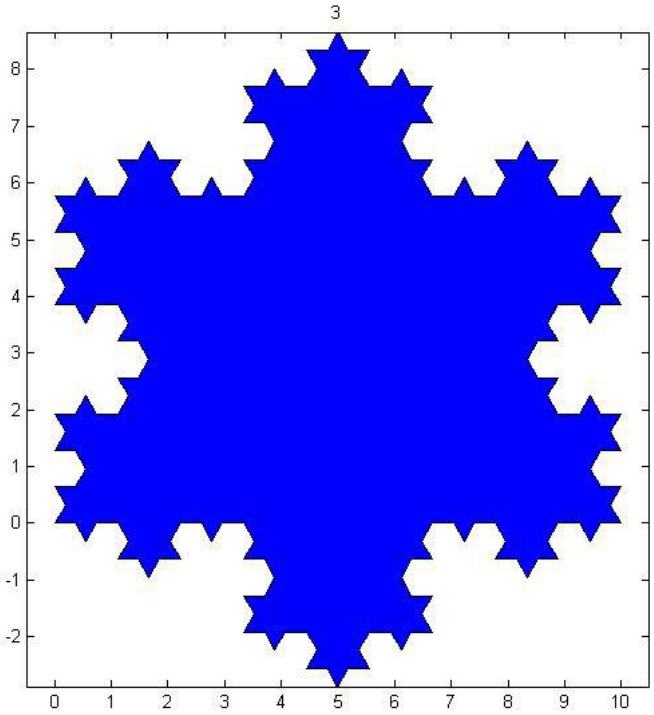


图 6 k=3 时候的 Koch 雪花图

k=5 时候的 Koch 雪花图：

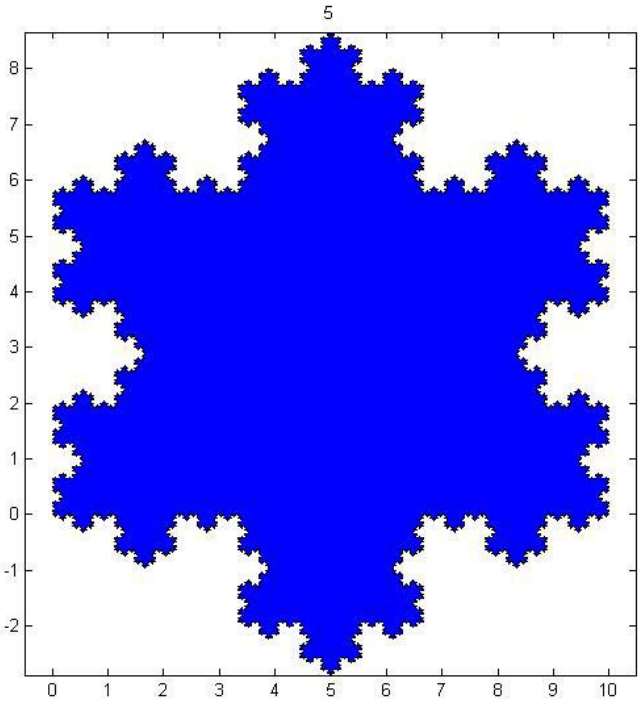


图 7 k=5 时候的 Koch 雪花图

k=7 时候的 Koch 雪花图：

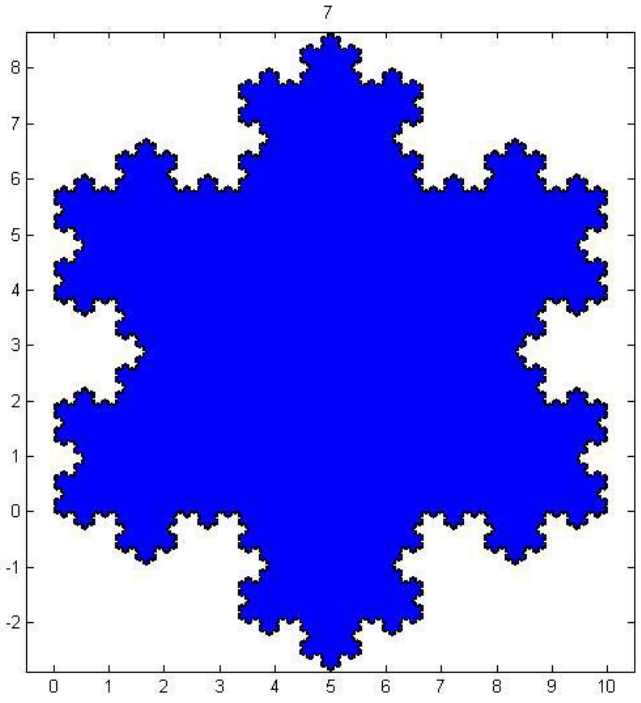


图 8 k=7 时候的 Koch 雪花图

不同的 k 值下的 Koch 雪花图的面积 S 如表 1 所示

表 1 不同的 k 值下的 Koch 雪花图的面积

k	0	1	2	3	4	5	6	7
S	43.301	57.735	62.546	64.150	64.685	64.863	64.922	64.942

（初始正三角形边长为 N=10）

绘制 S 与 k 之间的散点图如下：

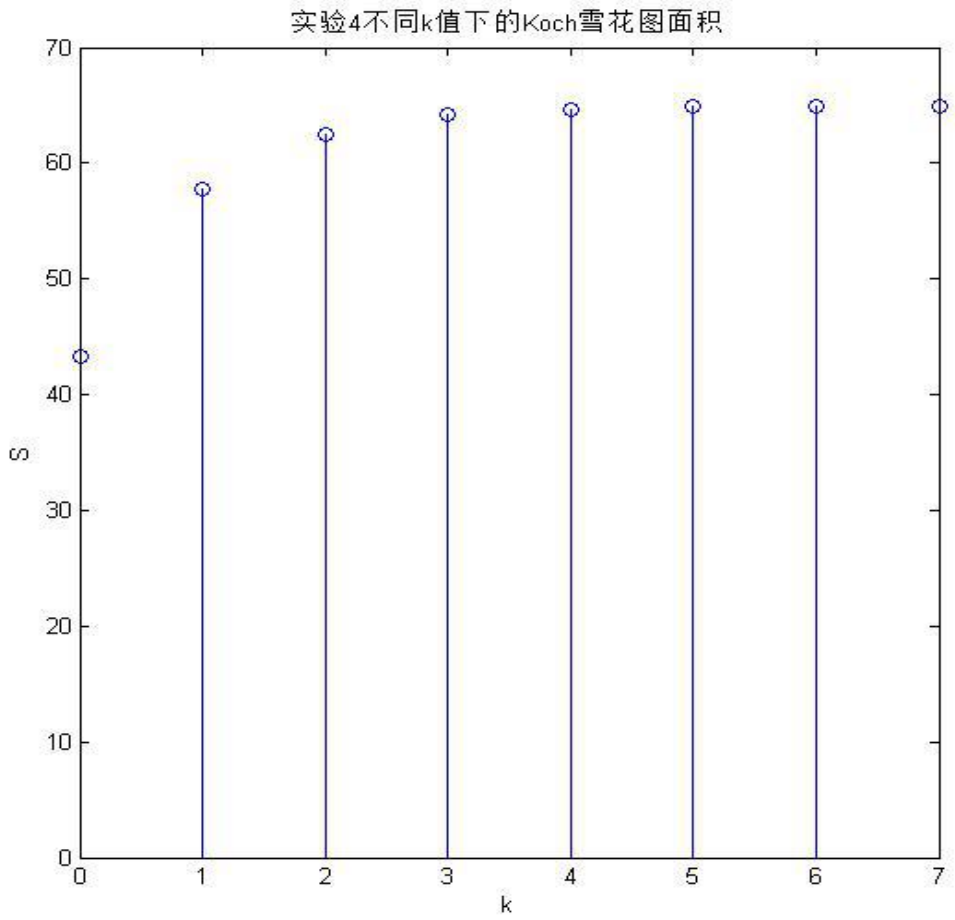


图 9 不同的 k 值下的 Koch 雪花图的面积 S

迭代运算比较复杂，当 k 值比较小的时候，程序运行比较流畅。当输入一个比较大的 k 值（>10）时，程序运行时间比较长，甚至可能死机。由(1)式计算极限可以知道，当 k 取无限大的时候，Koch 雪花的曲线总面积趋于无穷大，但在实验当中测试的 k 值比较小，而且随着 k 值增大，S 增大的速度越来越慢，所以观测效果不明显，如图 9 所示。从这个实验也可以看出，迭代虽然简化了很多问题的数学表示，但实际实现过程当中的计算机要处理数据量十分之大，迭代次

数过多的时候，对计算机的处理能力要求很高，因此有必要研究更加好的算法去减少运算量，特别是重复量，本实验实现代码最大的一个坏处就是每次迭代都是从第 0 次开始，而没有利用好前一次迭代的结果，计算机大量地做重复的工作，因此运算效率大大下降。当然，一个有趣的现象是，迭代次数为 5 所得的图像（见图 7）和迭代次数为 7 所得的图像（见图 8）在肉眼观测过程中是几乎没有区别的。所以实际使用当中，也要考虑迭代次数是否需要很大。

3 问题 2

3.1 问题描述

2、（选做）自己构造生成元（要有创意），按照图形迭代的方式产生分形图，用计算机编制程序绘制出它的图形。

3.2 实验原理

利用 Koch 雪花的绘制经验，我决定利用正方形绘制一个风火轮，因为在绘制 Koch 雪花的时候，基础图形为正三角形，所以考虑美观，在这里我使用正方形作为基础图形；同时在绘制 Koch 雪花的时候，每一次迭代，核心部分是图形的每一条边的两个三等分点之间是向外凸出一个小正三角形。在这里，我逆向思考，对正方形的每一条边，不是向外凸出的，而是向内凹的，类似地也是对每一条边，割掉两个三等分点之间的线段，第 1 个三等分点作和该边夹角为 30 度向内的射线，第 2 个三等分点作和该边夹角为 60 度向内的射线，它们的公共点分别和第 1 个三等分点和第 2 个三等分点连线，这就是一次迭代过程，如图 10 所示。这里我将它称为“风火轮变换”。

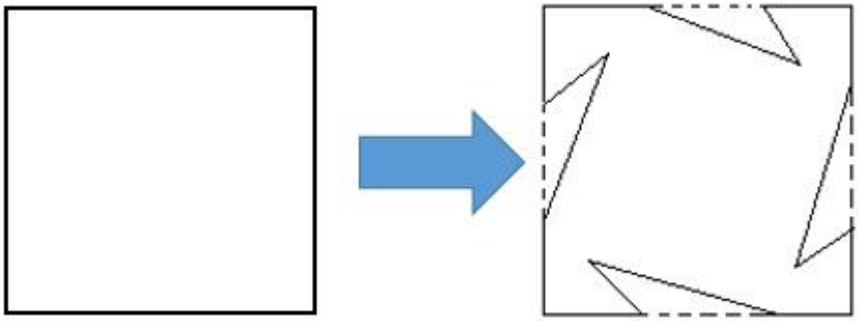


图 10 “风火轮变换”示意图

3.3 算法与编程

3.3.1 编程描述

根据迭代原理，在正方形的四条边上分别按照“风火轮变换”的迭代规律进行转化，即可形成风火轮。

（1）将每一条边分成三等份；

（2）对于每一条边，割掉两个三等分点之间的线段，第 1 个三等分点作和该边夹角为 30 度向内的射线，第 2 个三等分点作和该边夹角为 60 度向内的射线，它们的公共点分别和第 1 个三等分点和第 2 个三等分点连线；

（3）重复操作（1）和操作（2）。

3.3.2 实现代码

对于问题 2 的实现代码，如下：

```
function plotfire(k,N)
%function plotfire(k,N)
%plotfire.m       显示由正方形经过迭代 k 次后得到的风火轮，数学实验 4 第 2 题实现函数
%本函数说明：实验证明,调用本函数，迭代次数 k 最好小于 10
%如果电脑运行内存比较小，那么，当迭代次数大于 10 的时候，会出现内存不足的问题
%命令行报错：内存不足。请键入 HELP MEMORY 查看选项。
%          k          迭代次数
%          N          迭代的初始正方形的边长

%   Chen Yirong  修改于 2017-04-26
%   代码编辑 matlab 版本：MATLAB R2014a

sq=[0,N;N,0;0,-N;-N,0;0,N];
%sq 数组存储的为初始正方形四个顶点的坐标,
%其中第五个点与第一个点重合，确保绘图时首尾相连,以便于绘图
%第一列为 x 坐标,第二列为 y 坐标
n=5; %n 为结点数
A=[cos(-pi/3),-sin(-pi/3);sin(-pi/3),cos(-pi/3)]; %矩阵 A 为旋转矩阵,顺时针旋转 60 度
    for i=1:k       %迭代 k 次
        d=diff(sq)/3; %d 存储的为上一次边长的三分之长度
        m=4*n-3; %每次迭代，风火轮的数组 sq 的结点数变为 4*n-3
        q=sq(1:n-1,:); %数组元素复制，除最后一个结点
        sq(5:4:m,:)=sq(2:n,:);
```

```
        %将上一次迭代得到的风火轮的节点以 4 为间隔放进该轮迭代的风火轮结点数组当中
        sq(2:4:m,:)=q+d;
        %将上一次迭代得到的风火轮的节点，移动到靠近结点的三等分点，并以 4 为间隔放进该轮迭代的风火轮结点数组当中
        sq(3:4:m,:)=q+2*d+d*A';
        %将上一次迭代得到的风火轮的节点，移动到远离结点的三等分点，并以 4 为间隔放进该轮迭代的风火轮结点数组当中，并以原点为中心顺时针旋转 60 度
        sq(4:4:m,:)=q+2*d;
        %将上一次迭代得到的风火轮的节点，移动到远离结点的三等分点，并以 4 为间隔放进该轮迭代的风火轮结点数组当中
        n=m;    %当前的数组 sq 的结点数变为 4*n-3
    end
plot(sq(:,1),sq(:,2),'r');   %绘制风火轮
axis([-N,N,-N,N]);    %绘图范围为-N<=x<=N;-N<=y<=N
title('迭代与分形：由正方形产生风火轮');
xlabel(k);

end
```

3.4 实验结果

在 matlab 的命令行窗口输入：plotfire(0,10)，得到图 11 所示图形。

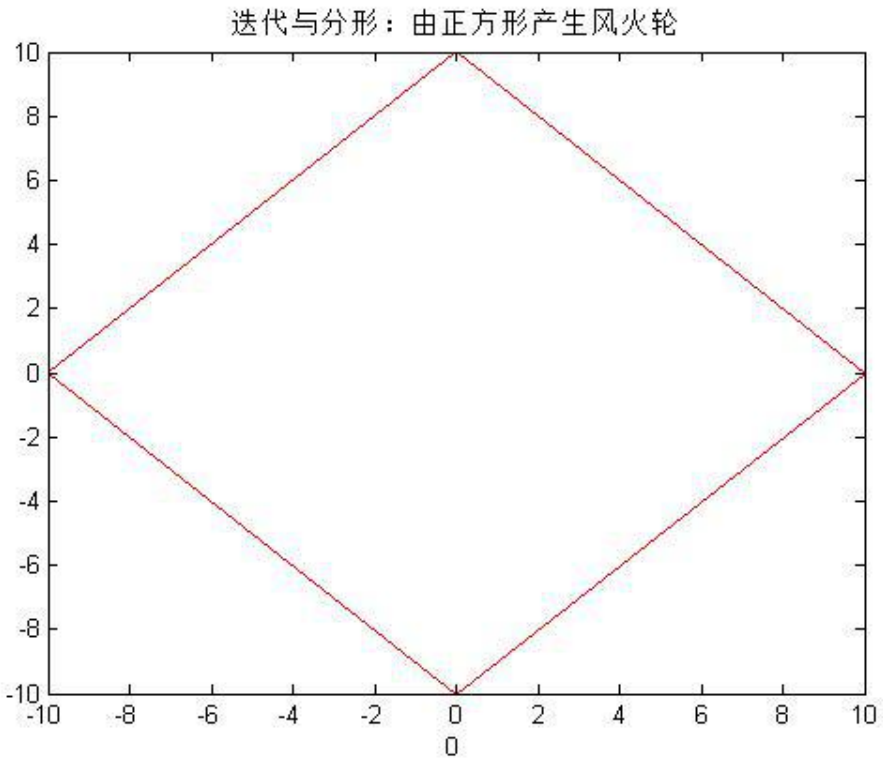


图 11 经过 0 次迭代的风火轮

在 matlab 的命令行窗口输入：plotfire(1,10)，得到图 12 所示图形。

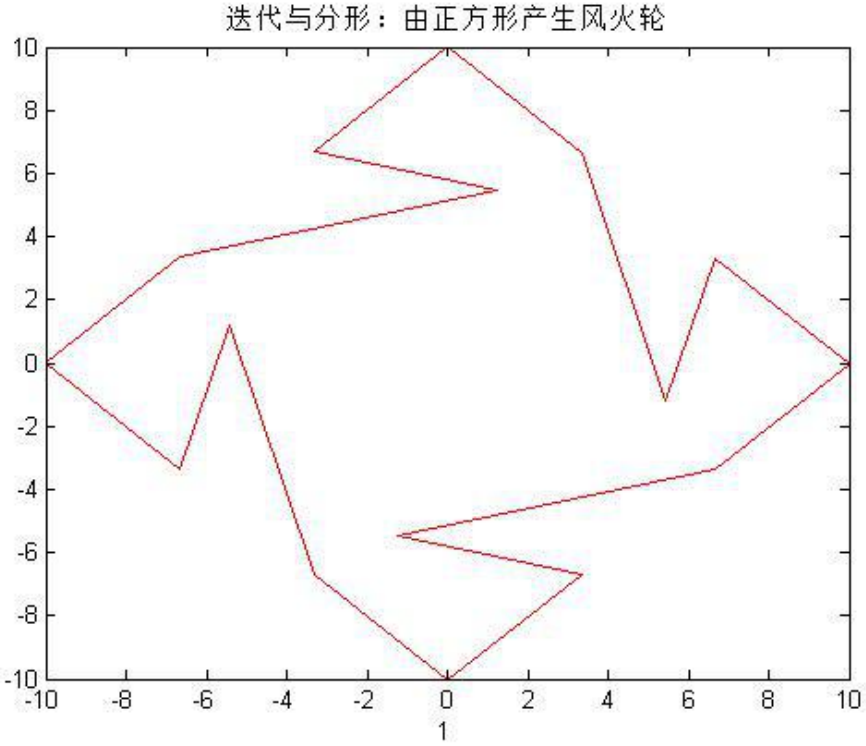


图 12 经过 1 次迭代的风火轮

在 matlab 的命令行窗口输入：plotfire(5,10)，得到图 13 所示图形。

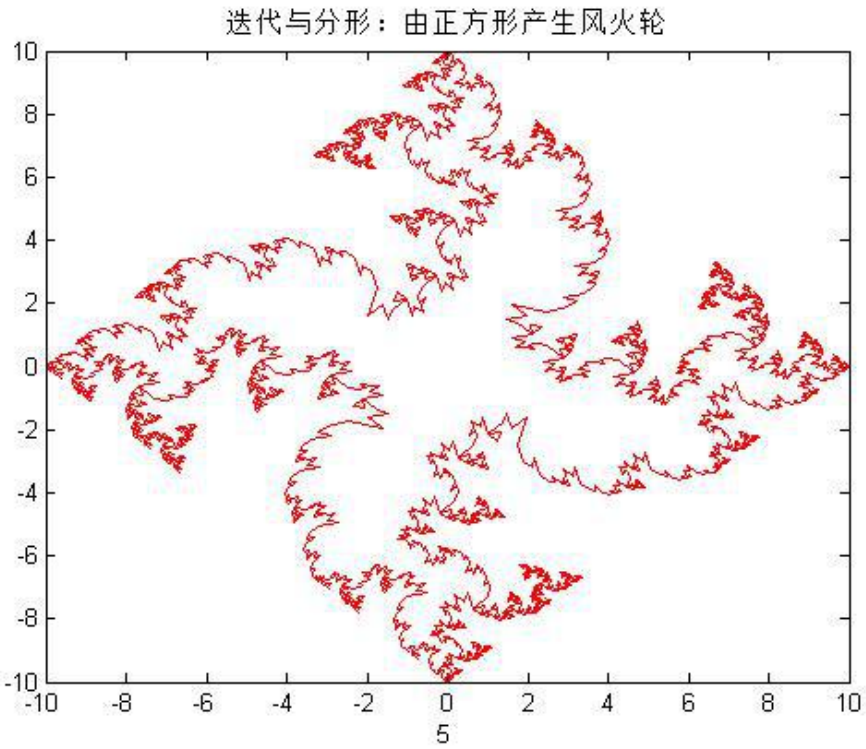


图 13 经过 5 次迭代的风火轮

在 matlab 的命令行窗口输入：plotfire(10,10)，得到所示图形。

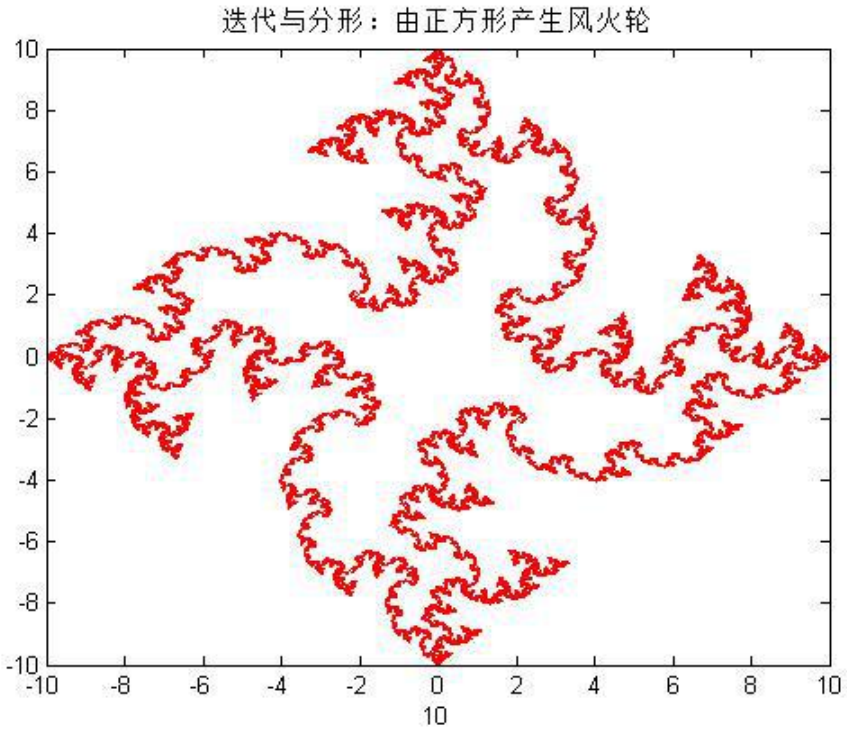


图 14 经过 10 次迭代的风火轮

在 matlab 的命令行窗口输入：plotfire(11,10)，命令行窗口输出如图 15 所示提示。

图 15 内存不足而报错

结论：比较问题 1 和问题 2 的求解，可以发现对封闭图形的分形几何有两个基本思路：其一，对基本图形以一定的法则向外拓展，经过迭代后得到分形图；其二，对基本图形以一定法则向内收缩，经过迭代后得到分形图。但因为它们都是通过迭代得到的，因此对计算机的运算能力要求较高。问题 1 和问题 2 的求解过程，都要求迭代次数小于等于 10 次，否则会出现运行时间过长、内存不足、程序崩溃等错误，如图 15 所示。

4 实验总结和实验感悟

4.1 实验总结

1、在本次数学实验中，我体会到分形几何学的奥妙；

2、通过对问题 2 的求解，我掌握了利用迭代这一基本思想获得分形几何图。

4.2 实验感悟

这次实验让我感触最深的一点就是运算量很重要，对于一个程序，虽然使用迭代可以轻易地实现我们想要的功能，但随着迭代次数的增加，程序的运算量是以指数级别增长的，往往在分形几何中，迭代次数大于 10 的时候，普通电脑就会出现崩溃、内存不足等问题，即使电脑没有崩溃，但其中的运行时间也会让我们崩溃。所以，有时候，迭代固然使程序的编写简单快速，但它是以牺牲大量运行时间为代价的。

同时，在实际中需要产生分形时，我认为迭代次数都是可以控制在 10 之内的，因为过大的迭代次数产生的图形在肉眼看来并没有太大变化，却使得程序运行时间高速增长。

最后，感谢老师的指导，也十分感谢身边的同学在我遇到不懂的地方可以帮忙提供思路，在我的程序遇到我无法解决的错误时可以给出详细的指点。

实验五　人脸识别

地　　点：	4 号楼　　4104 房；	**实验台号：**	66，79
实验日期与时间：	2017 年 05 月 08 日	**评　　分：**	
预习检查纪录：		**实验教师：**	刘小兰
电子文档存放位置：			
电子文档文件名：	卓越班-66，79-陈艺荣何晨晖实验五		

批改意见：

1 实验目的

- 了解数字图像的基本概念，了解人脸识别的基本含义；
- 掌握常见的人脸识别算法的基本原理；
- 了解 Matlab 中基本的文件和图像处理命令；
- 掌握文献检索和各大期刊数据库的使用；
- 掌握数学建模的基本步骤，利用数学知识将问题抽象为数学模型；
- 加强与队友之间的合作，提高团队合作精神。

2 问题描述

2.1 问题描述

利用各种人脸识别算法：基于回归模型的人脸识别算法 LRC，最远子空间分类算法 FS、最近最远子空间分类算法 NFS，或自己掌握或提出的其他人脸识别算法，选择其中的一种或几种算法，对给定的人脸数据库进行识别测试，得出识别正确率和（或）运行时间。在实验过程中，可以察看原始的人脸图片，哪些人脸识别错误？该算法有哪些优缺点？改进方向是什么？

给定的数据库为：Yale_32x32，Yale_64x64，ORL_32 x32，ORL_64 x64，YaleB_32x32。例如 Yale_32x32.mat，包含两个变量，一个是 fea：165*1024，表

示该数据集含有 165 个人脸，每个人脸是 1024 维（32*32 的人脸数据，已经被拉成了 1014 维的向量），一个是 gnd：165*1，代表这 165 个人脸的类别，分别用 1，2，…，15 表示。

2.2 问题背景

人脸识别（Face recognition）是应用计算机技术对人脸图像进行特征分析，抽取出有利于分类识别的判别信息，并以此自动识别出人的身份的一种技术。作为一种生物特征识别技术，不同于虹膜识别、指纹识别及近来出现的静脉识别，人脸识别有着隐蔽性和非侵犯性的特点，在刑侦破案、证件验证、公共安全等领域有十分重要的研究意义，且目前已经在现实中被广泛应用[1]。

一般而言，人脸识别的研究历史可以分为三个阶段。在第一阶段（1950s-1980s），人脸识别被当作一个一般性的模式识别问题，主流技术基于人脸的几何结构特征。在第二阶段（1990s）人脸识别迅速发展，出现了很多经典的方法，例如 Eigen Face, Fisher Face 和弹性图匹配，此时主流的技术路线为人脸表观建模。在第三阶段（1990s 末期到现在），人脸识别的研究不断深入，研究者开始关注面向真实条件的人脸识别问题，主要包括以下四个方面的研究：1）提出不同的人脸空间模型，包括以线性判别分析为代表的线性建模方法，以 Kernel 方法为代表的非线性建模方法和基于 3D 信息的 3D 人脸识别方法。2）深入分析和研究影响人脸识别的因素，包括光照不变人脸识别、姿态不变人脸识别和表情不变人脸识别等。3）利用新的特征表示，包括局部描述子（Gabor Face, LBP Face 等）和深度学习方法。4）利用新的数据源，例如基于视频的人脸识别和基于素描、近红外图像的人脸识别[2]。

人脸在视觉上的特点是：第一，不同个体之间的区别不大，所有的人脸的结构都相似，甚至人脸器官的结构外形都很相似。这样的特点对于利用人脸区分人类个体是不利的；第二，人脸的外形很不稳定，人可以通过脸部的变化产生很多表情，而在不同观察角度，人脸的视觉图像也相差很大。另外，面部识别还受光照条件（例如白天和夜晚，室内和室外等）、人脸的很多遮盖物（例如口罩、墨镜、头发、胡须等）、年龄等多方面因素的影响，因此区分个体变得异常困难[3]。

3 文献调研

3.1 国内外研究现状

人脸识别研究，起源于l9世纪末法国人Sir Franis Gahon的工作。到20世纪90年代，开始作为一个独立学科快速发展起来。人脸识别研究的发展大致分成三个阶段：第一阶段以Allen和Parke为代表，主要研究人脸识别所需要的面部特征。研究者用计算机实现了较高质量的人脸灰度图模型。这阶段的工作特点是识别过程全部依赖于操作人员。第二阶段是人机交互式识别阶段，其中用几何特征参数来表示人脸正面图像是以Harmon和Lesk为代表，将人脸面部特征用多维特征矢量表示出来，并设计了基于这一特征表示法的识别系统。而以Kaya和Kobayashi为代表，则采用了统计识别的方法、用欧氏距离来表示人脸特征。这两类方法都摆脱不了人的干预。第三阶段是真正的机器自动识别阶段，近十余年来，随着高速度高性能计算机的发展，人脸模式识别方法有了较大的突破，提出了多种机器全自动识别系统，人脸识别技术进人了实用化阶段。如Eyematic公司研发的人脸识别系统。我国清华大学的“十五”攻关项目《人脸识别系统》也通过了由公安部主持的专家鉴定。

目前人脸识别的难点主要存在于以下几个方面：

（1）光照变化是影响人脸识别性能的最关键因素，对该问题的解决程度关系着人脸识别实用化进程的成败，在人脸图像预处理或者归一化阶段，尽可能地补偿乃至消除其对识别性能的影响。

（2）成像角度及成像距离等因素的影响，即人脸的姿态的变化，会垂直于图像平面的两个方向的深度旋转，会造成面部信息的部分缺失。

（3）不同年龄的人脸有着较大的差别。身份证是以前照的，在逃犯的照片也是以前的，因此在公安部门的实际应用中，年龄问题是一个最突出的问题。

（4）采集图像的设备较多，主要有扫描仪、数码相机、摄像机等。由于成像的机理不同，形成了同类人脸图像的识别率较高，而不同类间人脸图像识别率较低的情况。随着人脸识别技术的发展，这一问题也将逐步得到解决。

（5）人脸图像的数据量巨大。目前由于计算量的考虑，人脸定位和识别算法研究大多使用尺寸很小的灰度图像。一张64*64像素的256级灰度图像就有4096个数据，每个数据有256种可能的取值。定位和识别算法一般都很复杂，在人脸

库较大的情况下，计算量十分大，很多情况下速度令人难以忍受。而灰度数据事实上是丧失了色彩、运动等有用信息的。如果使用全部的有用信息，计算量就更大了[4]。

3.2 常用人脸识别算法

人脸识别的方法主要有：基于几何特征的人脸识别方法、基于模板匹配的人脸识别方法、基于小波特征的人脸识别方法和基于模型的人脸识别方法。其中基于模型的人脸识别方法包括特征脸法(Eigenface)、神经网络法(NN)、隐马尔可夫模型方法(HMM)等方法[5]。本次实验我们采用的是基于特征脸的人脸识别方法[6]。

3.2.1 基于回归模型的人脸识别方法

LRC 算法是 Naseem 等人 2010 年提出的一种用于人脸识别的线性回归分类器(Linear Regression Classification，LRC)，它本质上是一个最近子空间分类器(Nearest Space, NS)[7]。基于稀疏表示的方法在人脸识别方面体现出优异的性能，然而随后的研究显示，稀疏性对于人脸识别的识别效果并不起主要作用。有研究认为是协同表示(Collaborative Representation,CR)的机制而非稀疏性约束在该类算法中起主体作用，并将稀疏表示分类器(Sparse Representation based Classification,SRC)视为协同表示分类器(Collaborative Representation based Classification,CRC)的一个特例[8]。

LRC 算法认为：

（1）同一个人的人脸图像位于一个线性子空间中，这个线性子空间可由训练样本张成；

（2）任何一个人的人脸图像可以由这个人的训练图像线性表示；

（3）待检测人脸图像应当与训练集中同一类图像张成的子空间距离最小。

3.2.2 基于神经网络的人脸识别方法

基于神经网络的人脸识别方法就是利用神经网络的学习能力和分类能力对人脸进行特征提取与识别，见图 1 所示。目前常用的人工神经网络方法是 BP 神经网络、自组织神经网络、径向基函数神经网络。径向基函数神经网络与 BP 网

络一样都是多层前向网络，它以径向基函数作为基准，以高斯函数作为隐含层的激励函数。这种网络的学习速率快、函数逼近、模式识别等能力均优于 BP 神经网络，并广泛应用于模式识别、图像处理等方面。但是这种网络比 BP 网络所用的神经元数目要多得多，使它的应用受到了一定的限制[9]。

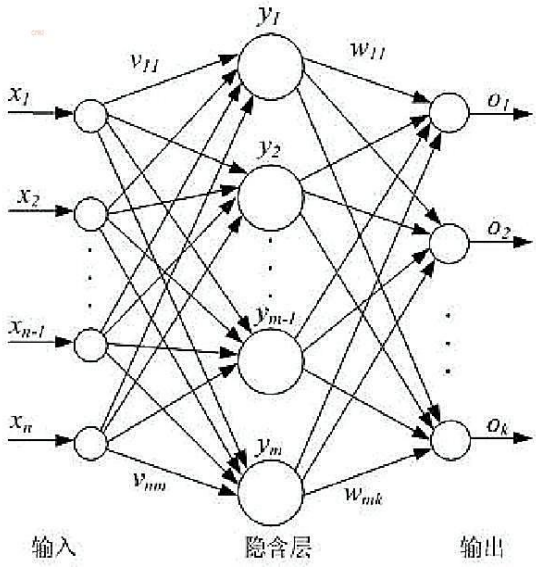


图 1 神经网络示意图

3.2.3 基于特征脸的人脸识别方法

特征脸方法是从主成分分析(PCA)导出的一种人脸识别和描述技术。它将包含人脸的图像区域看作一随机向量，采用 K-L 变换得到正交 K-L 基，对应其中较大特征值的基具有与人脸相似的形状，因此又被称为特征脸。利用这些基的线性组合可以描述、表达和逼近人脸图像，所以可进行人脸识别与合成。识别过程就是将人脸图像映射到由特征脸组成的子空间上，并比较其在特征脸空间中的位置，然后利用对图像的这种投影间的某种度量来确定图像间的相似度，最常见的就是选择各种距离函数来进行度量分类实现人脸识别。

主成分分析原理：

对于一个样本资料，观测 p 个变量 $x_1, x_2, \cdots x_p$， n 个样品的数据资料阵为：

$$X = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \vdots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{np} \end{pmatrix} = \left(x_1, x_2, \cdots x_p\right)$$

其中：$x_j = \begin{pmatrix} x_{1j} \\ x_{2j} \\ \vdots \\ x_{nj} \end{pmatrix}, \quad j = 1,2,\cdots p$

主成分分析就是将 p 个观测变量综合成为 p 个新的变量（综合变量），即

$$\begin{cases} F_1 = a_{11}x_1 + a_{12}x_2 + \cdots + a_{1p}x_p \\ F_2 = a_{21}x_1 + a_{22}x_2 + \cdots + a_{2p}x_p \\ \cdots \\ F_p = a_{p1}x_1 + a_{p2}x_2 + \cdots + a_{pp}x_p \end{cases}$$

简写为：

$$F_j = \alpha_{j1}x_1 + \alpha_{j2}x_2 + \cdots + \alpha_{jp}x_p$$

$$j = 1,2,\cdots,p$$

要求模型满足以下条件：

①F_i, F_j 互不相关（$i \neq j$， $i, j = 1,2,\cdots,p$）

②F_1 的方差大于 F_2 的方差大于 F_3 的方差，依次类推

③${a_{k1}}^2 + {a_{k2}}^2 + \cdots + {a_{kp}}^2 = 1 \quad k = 1,2,\cdots p.$

于是，称 F_1 为第一主成分，F_2 为第二主成分，依此类推，有第 p 个主成分。主成分又叫主分量。这里 a_{ij} 我们称为主成分系数。

上述模型可用矩阵表示为：

$F = AX$，其中

$$F = \begin{pmatrix} F_1 \\ F_2 \\ \vdots \\ F_p \end{pmatrix} \qquad X = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_p \end{pmatrix}$$

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1p} \\ a_{21} & a_{22} & \cdots & a_{2p} \\ \vdots & \vdots & \vdots & \vdots \\ a_{p1} & a_{p2} & \cdots & a_{pp} \end{pmatrix} = \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_p \end{pmatrix}$$

A 称为主成分系数矩阵。

本实验将采用基于主成分分析的方法实现人脸识别。

3.3 算法与编程

3.3.1 编程描述

完整的 PCA 人脸识别的应用包括几个步骤：

① 人脸图像预处理；

② 读入人脸库；

③ 训练形成特征子空间；

④ 把训练图像和测试图像投影到特征子空间上；

⑤ 选择一定的距离函数进行识别。

因为我们在本实验侧重于人脸识别这一过程，所以，对于人脸图像预处理这一部分不用进行编程实现，而是从老师给定的数据库：Yale_32x32，Yale_64x64，ORL_32 x32，ORl_64 x64，YaleB_32x32 中选择数据库进行测试。

所以本实验的算法实现流程如下图所示：

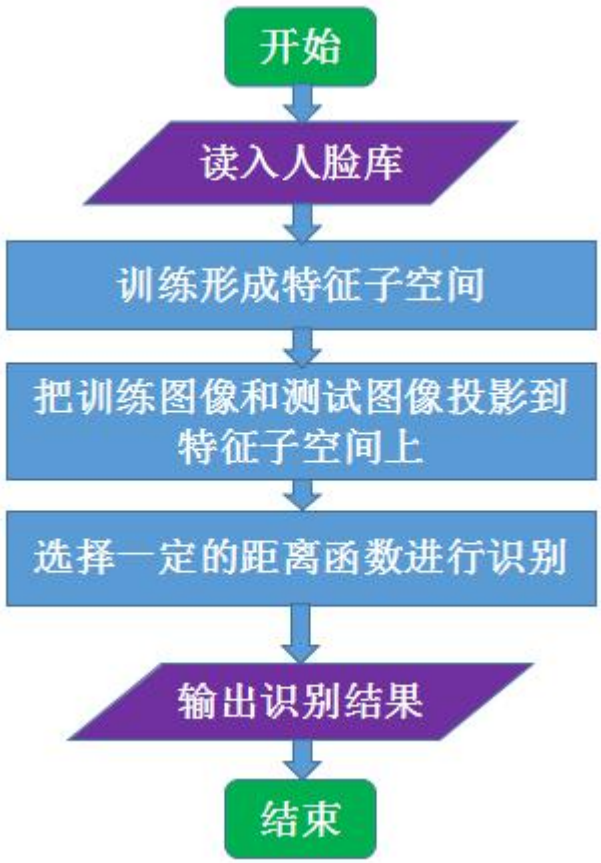


图 2 本实验算法流程图

训练过程如下图所示：

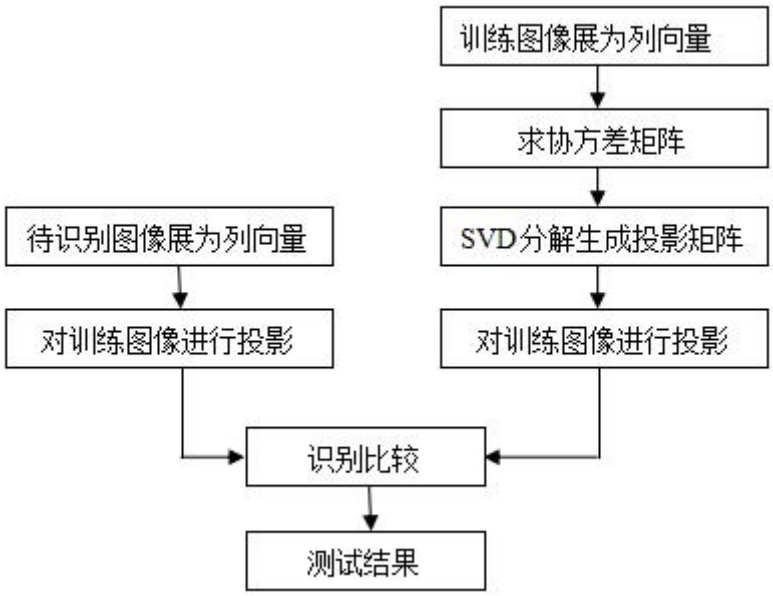


图 3 本实验人脸识别训练过程

实验编写函数 M 文件：EigenfaceCode.m、Recognition.m 分别实现训练和识别两个过程。

实验编写脚本测试 M 文件：renlianshibie.m 实现人脸识别以及结果分析。

其中 Yale_32x32.mat，Yale_64x64.mat，ORL_32 x32.mat，ORL_64 x64.mat 四个数据文件需要和代码文件 EigenfaceCode.m、Recognition.m、renlianshibie.m 存放在同一个位置，如图 4 所示。

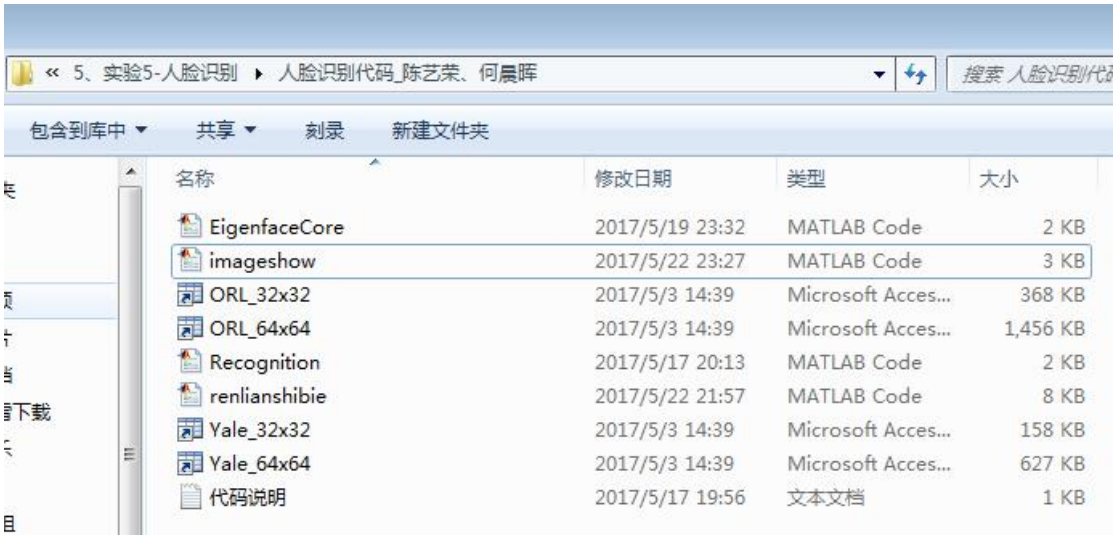


图 4 人脸识别代码存储文件夹

3.3.2 实现代码

renlianshibie.m、EigenfaceCode.m、Recognition.m、imageshow.m 三个 M 文件代码如下：

```
%renlianshibie.m        人脸识别测试脚本文件
%文件说明：本文件是人脸识别测试用 M 文件
%本次测试使用 Yale_32x32.mat 当中的数据
%其中，在测试前应确保 Yale_32x32.mat 已经和本文件在同一路径下
%本例使用数据库 Yale_32x32
%测试的数据集是第 11,22,33,44,55,66,77,88,99,110,121,132,143,154,165 张图片
%                    CRC       识别正确率
%                    t         程序运行时间

%   Chen Yirong and He Chenhui  修改于 2017-05-08
%   代码编辑 matlab 版本：MATLAB R2014a

%准备工作
tic
clear all   %清空工作空间
clc           %清空命令行窗口
close all   %关闭其他 figure 窗口
Judge=zeros(1,15);   %存储判断值，元素取值只有 1 和 0 两种可能，判断正确则为 1，错误则为 0
TestArray=[11,22,33,44,55,66,77,88,99,110,121,132,143,154,165];   %测试图片编号索引

load('Yale_32x32.mat')   %读取数据库 Yale_32x32
%load('Yale_32x32.mat')
% Yale 数据库包含 15 位志愿者共 165 张图像，每位志愿者 11 张图像，图像采集于不同的光照、表情和姿态，图像的大小为 32x32。
%一个是 fea：165*1024，表示该数据集含有 165 个人脸，每个人脸是 1024 维（32*32 的人脸数据，已经被拉成了 1014 维的向量）
%一个是 gnd：165*1，代表这 165 个人脸的类别，分别用 1，2，…，15 表示
%这里选定 fea 中每个人的第一张照片作为训练数据集，全体 fea 集作为测试数据集

%创建训练集
%除了被 11 整除的那些编号的图片，其他图片都用来做训练集，共 150 张训练
```

```
图片
TrainDatabase=zeros(1024,150);     %创建一个 1024*150，元素全为 0 的矩阵，用来存放训练集
tnum=1;       %创建训练集时用到的临时变量
for xlk=1:164
   if(gnd(xlk)==gnd(xlk+1))   %除了被 11 整除的那些编号的图片，其他图片都用来做训练集，共 150 张训练图片
         TEMarrayline=fea(xlk,:);     %提取图片
         TEMarrayrow=reshape(TEMarrayline,1024,1);   %行向量转为列向量
         TrainDatabase(:,tnum)=TEMarrayrow(:,1);       %创建训练数据集
         tnum=tnum+1;
   end
end
T=TrainDatabase;     %T 是一个二维矩阵，若有 P 张 M*N 的图像，则 T 为 M*N 行 P 列的矩阵
temresult=zeros(1,15); %存储测试识别结果：人物编号
temoutputname=zeros(1,15); %存储测试识别结果：图片编号
%人脸识别准备工作
for testnum=1:15
TestNumber=TestArray(testnum);   %读取测试图片编号
RealNumber=gnd(TestNumber);     %确定所读取图片是第几个人，真实值
OrignImage=fea(TestNumber,:);   %读出待测试图片
originim=reshape(OrignImage,32,32);   %行向量转变为 32*32 的矩阵
im=originim;   %待测试图片矩阵
[m,A,Eigenfaces] = EigenfaceCore(T);       %用 PCA 原理决定人脸图像的最优特征，得到一个二维矩阵，包含训练图像向量，返回三个输出
OutputName = Recognition(im ,m ,A , Eigenfaces);     %输出人物索引
Outputresult=floor(OutputName/10)+1;     %根据索引计算人物编号
temresult(testnum)=Outputresult;   %该数组用于存储每一个测试样例的识别结果：人物编号
temoutputname(testnum)=OutputName;   %该数组用于存储每一个测试样例的识别结果：图片编号
ShibieNumber=Outputresult;
      if(ShibieNumber==RealNumber)   %比较获取判断矩阵用于计算识别准确率
            Judge(testnum)=1;   %人脸识别正确，赋值 1
            else Judge(testnum)=0;     %人脸识别错误，赋值 0
      end
end
%计算测试准确率
CorrectNumber=0;   %记录正确测试人数
for j=1:15
      if( Judge(j)==1 )
            CorrectNumber=CorrectNumber+1;
      end
```

```
end
CRC=CorrectNumber/15        %计算识别正确率，存储在 CRC 中
%绘制测试结果
figure('Name','实验 5 显示测试情况','Position',[80,80,500,300]);   %创建 1 个新显示图形的窗口
stem(TestArray,Judge);   %绘制测试结果
title('基于数据集 Yale32x32 的人脸识别结果');   %绘制标题
xlabel('被测试图片编号');    %绘制横坐标
ylabel('识别结果（0 代表错误，1 代表正确）');    %绘制纵坐标
%输出识别错误的样本和其被识别到的图片
figure('Name','实验 5 测试错误结果显示','Position',[680,30,600,550]);   %创建 1 个新显示图形的窗口
ploti=1;   %图片索引
    for tempnum=1:15
        if( Judge(tempnum)==0 )
            subplot(15-CorrectNumber,2,ploti)   %画图
            temp=fea(TestArray(tempnum),:);    %获取识别错误的样本图片
            temppic=reshape(temp,32,32);   %行向量转变为 32*32 的矩阵
            imshow(temppic,[]);    %显示图片
            title('被测试图片');   %绘制标题
            ploti=ploti+1;
            subplot(15-CorrectNumber,2,ploti)   %画图
            temp=fea(temoutputname(tempnum),:);    %获取识别错误的样本图片
            temppic=reshape(temp,32,32);   %行向量转变为 32*32 的矩阵
            imshow(temppic,[]);    %显示图片
            title('测试图片被识别为的错误结果');   %绘制标题
            ploti=ploti+1;
        end
    end
    t=toc   %输出程序运行时间 t
%程序结束

function [m, A, Eigenfaces] = EigenfaceCore(T)
% 用 PCA 原理决定人脸图像的最优特征，得到一个二维矩阵，包含训练图像向量，返回三个输出
% 参数:T 包含训练集中所有的图像信息集合，
% 返回值：m：(M*Nx1)训练均值;
%          Eigenfaces：(M*Nx(P-1))训练集协方差矩阵的特征向量；
%          A：(M*NxP) 每一张图像与均值图像的方差矩阵
```

```
%    Chen Yirong and He Chenhui  修改于 2017-05-08
%    代码编辑 matlab 版本：MATLAB R2014a

m = mean(T,2); %  平均图像/列平均（每一副图像的对应象素求平均）
m=(1/P)*sum(Tj's) (j=1 : P)
Train_Number = size(T,2);%列数，训练图片张数
%计算机每一张图片到均值图像的方差
A = [];
   for i = 1 : Train_Number%对每一列
         temp = double(T(:,i))-m; %每一张图与均值的差异
          A=[A temp]; %方差矩阵
   end
%降维
L = A'*A; % L 是协方差矩阵 C=A*A'的转置.
[V,D] = eig(L); %对角线上的元素是 L|C 的特征值.V:以特征向量为列的满秩矩阵，
D：特征值对角矩阵。即 L*V = V*D.
L_eig_vec = []; %特征值向量
for i = 1 : size(V,2)    %对每个特征向量
%         if( D(i,i)>0 )    %特征值大于 0 时
            L_eig_vec = [L_eig_vec V(:,i)];     %集中对应的特征向量
%         end
end

Eigenfaces = A*L_eig_vec; %  计算协方差矩阵 C 的特征向量，得到降维了的特
征,A 为每一张图像与均值图像的方差构成的矩阵，

function OutputName = Recognition(TestImage, m, A, Eigenfaces)
%本函数用于进行人脸识别并且输出识别结果
%使用本函数之前需要使用 function [m, A, Eigenfaces] = EigenfaceCore(T)获取图
像特征数据
%             m：(M*Nx1)训练均值;
%             Eigenfaces：(M*Nx(P-1))训练集协方差矩阵的特征向量；
%             A：(M*NxP)  每一张图像与均值图像的方差矩阵


%    Chen Yirong and He Chenhui  修改于 2017-05-08
%    代码编辑 matlab 版本：MATLAB R2014a

%读入测试图像，测试前的准备工作
ProjectedImages=[];%映射图像
```

```
Train_Number=size(Eigenfaces,2);%列，降维后，
   for i =1:Train_Number       %对于每一个训练特征
           temp = Eigenfaces'*A(:,i);
           ProjectedImages=[ProjectedImages temp];   %得到 L_eig_vec;
   end
InputImage = TestImage;%读入测试图片
temp = InputImage;   %用临时矩阵装载图片
[irow , icol] = size(temp);%测试图片大小
InImage = reshape(temp, irow*icol ,1);%转置后转为一维
Difference = double(InImage)-m; % L_eig_vec'
%进行测试，特征比较
ProjectedTestImage = Eigenfaces'*Difference; %  测试图像的特征向量
Euc_dist = [];
for i = 1 : Train_Number     %对每列
      q = ProjectedImages(:,i);%取出训练图像
      temp=(norm(ProjectedTestImage-q))^2;%欧氏距离
      Euc_dist = [Euc_dist temp];
end
[Euc_dist_min , Recognized_index] = min(Euc_dist);%得到差值最小的图像的索引号
OutputName = Recognized_index;%返回索引

%imageshow.m        人脸识别测试脚本文件
%文件说明：本文件是人脸识别图片显示 M 文件
%本次测试使用 Yale_32x32.mat 当中的数据
%其中，在测试前应确保 Yale_32x32.mat 已经和本文件在同一路径下
%本例使用数据库 Yale_32x32
%测试的数据集是第 11,22,33,44,55,66,77,88,99,110,121,132,143,154,165 张图片
%运行本 M 文件之前需要确保 renlianshibie.m 文件已经运行

%    Chen Yirong and He Chenhui  修改于 2017-05-08
%    代码编辑 matlab 版本：MATLAB R2014a

%绘制所有图片
%创建第 1 个绘图窗
figure('Name','所有图片显示','Position',[50,30,600,600]);   %创建 1 个新显示图形的窗口
imagetempall=zeros(32*15,32*11);     %存储所有图片矩阵
ak=1;     %图片索引
```

```
    for h1=1:15
        for h2=1:11
            testim1=reshape(fea(ak,:),32,32);   %行向量转变为 32*32 的矩阵
            imagetempall(32*(h1-1)+1:32*h1,32*(h2-1)+1:32*h2)=testim1; %将
每一张图片放在新矩阵 imagetempall 中
            ak=ak+1;   %索引指向下一张图片
        end
    end
imshow(imagetempall,[]); %画图显示图片

%绘制训练集图片
%创建第 2 个绘图窗
figure('Name','训练集图片显示','Position',[680,30,600,600]);   %创建 1 个新显示图
形的窗口
TH=TrainDatabase';   %矩阵转置
imagetempall=zeros(32*15,32*10);   %存储训练集图片矩阵
ak=1; %图片索引
    for h1=1:15
        for h2=1:10
            testim1=reshape(TH(ak,:),32,32);   %行向量转变为 32*32 的矩阵
            imagetempall(32*(h1-1)+1:32*h1,32*(h2-1)+1:32*h2)=testim1; %将
每一张图片放在新矩阵 imagetempall 中
            ak=ak+1;   %索引指向下一张图片
        end
    end
imshow(imagetempall,[]);   %画图显示图片

%绘制训练集图片
%创建第 3 个绘图窗
figure('Name','测试集图片显示','Position',[380,30,600,600]);   %创建 1 个新显示图
形的窗口
TestArray=[11,22,33,44,55,66,77,88,99,110,121,132,143,154,165];   %测试图片编
号索引
%找出测试集图片
temppic=zeros(15,1024);
for ti=1:15
    ap=TestArray(ti);
    temppic(ti,:)=fea(ap,:);
end
imagetemp=zeros(32*5,32*3); %存储测试集图片矩阵
ak=1;
for h=1:5
    for l=1:3
    testim=reshape(temppic(ak,:),32,32);   %行向量转变为 32*32 的矩阵
```

```
        imagetemp(32*(h-1)+1:32*h,32*(l-1)+1:32*l)=testim;%将每一张图片放在新
矩阵 imagetemp 中
        ak=ak+1;%索引指向下一张图片
        end
end
    imshow(imagetemp,[]);      %显示图片
%       end
```

3.4 实验结果

本实验使用的数据库以及训练集、测试集图片如图 5、图 6、图 7 所示。

图 5 Yale_32x32 数据集中所有图片

图 6 Yale_32x32 数据集中被用作训练集的图片图片

图 7 Yale_32x32 数据集中被用作测试集的图片图片

识别结果图，将识别结果通过图形更直接地展示出来，0 为识别错误，1 为识别正确，横轴为测试图片的编号。

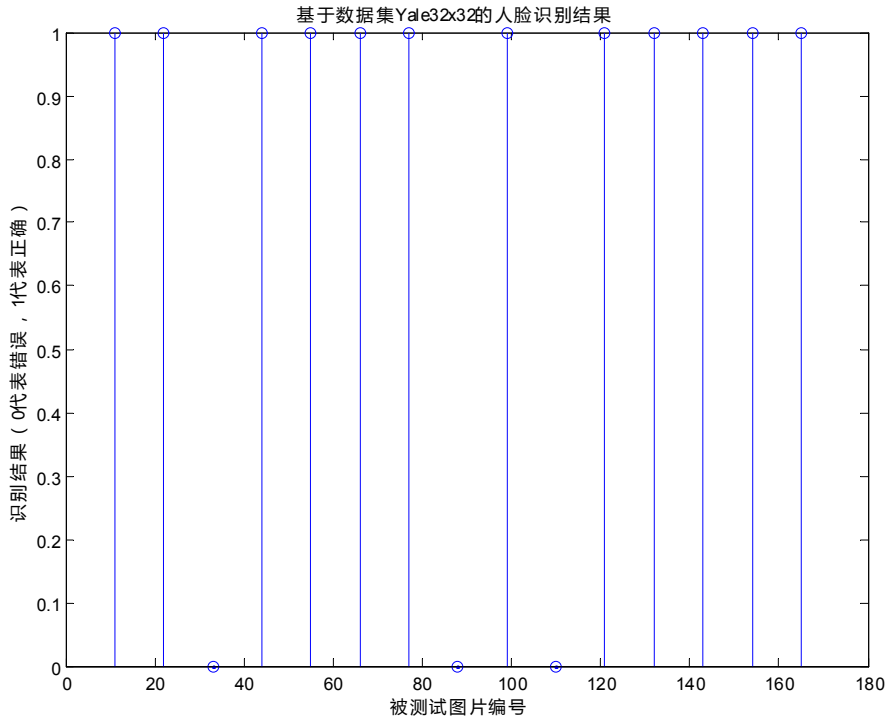


图 8 识别结果图

识别错误结果显示：

图 9 识别错误结果显示图

图 9 表明本次实验有 3 张图片被错误识别。

识别成功率计算结果：

CRC =

0.8000

也就是本算法用在 Yale_32x32.mat 数据集上时，人脸识别成功率达到 80%。

运行时间计算：

t =

1.4238

也就是本算法用在 Yale_32x32.mat 数据集上时，人脸识别完成以及结果分析的总时间为 1.4238 秒。

实验结果分析

1．实验在给定数据集中测试，一个训练集，该集包含 15 个不同人物，每人有 10 张不同表情和姿态下的图片，总共 150 幅。另一个是测试集，该库包含 15 个不同人物，每人有 1 张图片，共 15 幅。用测试样本进行测试，识别率为 80%。说明 PCA 算法可以有效的进行降维，有利于特征提取，而随着训练样本的增加，识别率会有所提升；

2．预处理对识别的效果起着至关重要的作用；

3．PCA 起到了简单的特征脸降维的作用，要有更好的效果，还必须寻找更好的特征表达，使得可以尽量消除光照、表情、遮掩和姿势的影响；

4．相比于其他算法来说，PCA 算法需要计算协方差矩阵，当维数过大时甚至无法计算，导致程序执行时间过长，属于非快速算法 。

4 实验总结和实验感悟

4.1 实验总结

1、在本次数学实验中，我了解了人脸识别技术的应用及其难点，发展与现状，研究内容与主要方法，及常用的人脸识别标准数据库。

2、通过对求解，我掌握了人脸识别的流程、KL 变换、特征脸方法（PCA），使用特征脸方法设计出人脸识别程序并计算出识别成功率。

3、在整个实验的设计以及代码的调试中，我们学会了模块化的思想，使得

整个实验条理更清楚，同时，在编写代码的过程中，我和队友要考虑到自己所编写的代码还要给队友看，因此我们都尽量规范自己的代码并且加入了大量的注释帮助理解，这使得我们的沟通更加方便，合作效率高。

4、通过本次实验，我和队友学会了文献检索方法，包括使用华工图书馆提供的各种数据库以及使用谷歌学术搜索对人脸识别进行文献检索，这为我们的实验提供了很大的帮助。

4.2 实验感悟

1、整个实验过程中，通过多种识别方法的比较与测试，我发现每种人脸识别方法都各有优缺点，如何充分利用现有的各种人脸识别方法，发挥某一类方法的优点，克服某一类方法的缺点，将它们进行有效的综合和组合，也将是以后一个探索的方向。

2、同时，此次实验成功也得益于与队友的团结协作，也让我认识到了团队合作的重要性，还有也要感谢老师的大力指导与帮助，希望在以后的实验中能够收获更多知识和方法。

实验六　增量人脸识别

地　　点：	4 号楼　　4104 房；	**实验台号：**	66，79
实验日期与时间：	2017 年 06 月 05 日	**评　　分：**	
预习检查纪录：		**实验教师：**	刘小兰
电子文档存放位置：			
电子文档文件名：	卓越班-66，79-陈艺荣何晨晖实验六		

批改意见：

摘要：人脸识别是当今十分热门的话题，关于人脸识别的算法层出不穷。但是，大多数算法随着样本数目的增加，或者样本数目的更新，运行时间成倍地增加。要将算法设计为适用于实际应用，则要考虑样本数目很大和样本需要经常更新。针对人脸识别算法随着训练样本数目增大的问题，我们小组提出一种基于主成分分析（PCA）的增量人脸识别算法。传统的增量主成分分析（IPCA）包括 CCIPCA 算法和 I2DCA 算法。本实验报告在 CCIPCA 的基础上作出进一步改进，提出了自己的 IPCA 模型，并通过 matlab 编程对 Yale_32x32，Yale_64x64，ORL_32 x32，ORL_64 x64 这四个数据集进行了测试，对它们的识别精度和运行时间进行了比较。最终得到良好的工程代码文件和比较适合推广使用的函数接口。

关键词：人脸识别、主成分分析、特征匹配、增量算法

1 实验目的

- 了解数字图像的基本概念，了解人脸识别的基本含义；
- 掌握常见的人脸识别算法的基本原理；
- 了解数字图像的基本概念，了解人脸识别的基本含义；
- 掌握基于回归模型的增量人脸识别算法的基本原理；
- 了解 Matlab 中基本的文件和图像处理命令；
- 掌握文献检索和各大期刊数据库的使用；
- 掌握数学建模的基本步骤，利用数学知识将问题抽象为数学模型；
- 加强与队友之间的合作，提高团队合作精神。

2 问题描述

2.1 问题描述

利用各种增量人脸识别算法：基于回归模型的增量人脸识别算法，最远子空间增量分类算法、最近最远子空间增量分类算法或其他快速算法，选择其中的一种或几种算法，对给定的人脸数据库进行识别测试，得出识别正确率和（或）运行时间。并与第 5 节不采用增量学习的算法进行比较，分析实验结果。在实验过程中，可以察看原始的人脸图片，哪些人脸识别错误？该算法有哪些优缺点？改进方向是什么？如果有新的样本加入训练集合中，如何处理？

当训练集的样本数较多时，如何处理？

(1) 传统的处理方法是，将新增加的训练样本和原来的训练样本放在一起，重新训练模型，将会造成时间和存储空间的巨大开销，严重影响计算的效率。

(2) 这会使得训练数据库的样本不断增多

给定的数据库为：Yale_32x32，Yale_64x64，ORL_32 x32，ORL_64 x64，YaleB_32x32。例如 Yale_32x32.mat，包含两个变量，一个是 fea：165*1024，表示该数据集含有 165 个人脸，每个人脸是 1024 维（32*32 的人脸数据，已经被拉成了 1014 维的向量），一个是 gnd：165*1，代表这 165 个人脸的类别，分别用 1，2，…，15 表示。

2.2 问题背景

人脸识别（Face recognition）是应用计算机技术对人脸图像进行特征分析，抽取出有利于分类识别的判别信息，并以此自动识别出人的身份的一种技术。作为一种生物特征识别技术，不同于虹膜识别、指纹识别及近来出现的静脉识别，人脸识别有着隐蔽性和非侵犯性的特点，在刑侦破案、证件验证、公共安全等领域有十分重要的研究意义，且目前已经在现实中被广泛应用[10]。

一般而言，人脸识别的研究历史可以分为三个阶段。在第一阶段（1950s-1980s），人脸识别被当作一个一般性的模式识别问题，主流技术基于人脸的几何结构特征。在第二阶段（1990s）人脸识别迅速发展，出现了很多经典的方法，例如 Eigen Face, Fisher Face 和弹性图匹配，此时主流的技术路线为人脸表观建模。在第三阶段（1990s 末期到现在），人脸识别的研究不断深入，研究者开始关注面向真实条件的人脸识别问题，主要包括以下四个方面的研究：1）提出不同的人脸空间模型，包括以线性判别分析为代表的线性建模方法，以 Kernel 方法为代表的非线性建模方法和基于 3D 信息的 3D 人脸识别方法。2）深入分析和研究影响人脸识别的因素，包括光照不变人脸识别、姿态不变人脸识别和表情不变人脸识别等。3)利用新的特征表示，包括局部描述子（Gabor Face, LBP Face 等）和深度学习方法。4）利用新的数据源，例如基于视频的人脸识别和基于素描、近红外图像的人脸识别[11]。

人脸在视觉上的特点是：第一，不同个体之间的区别不大，所有的人脸的结构都相似，甚至人脸器官的结构外形都很相似。这样的特点对于利用人脸区分人类个体是不利的；第二，人脸的外形很不稳定，人可以通过脸部的变化产生很多表情，而在不同观察角度，人脸的视觉图像也相差很大。另外，面部识别还受光照条件（例如白天和夜晚，室内和室外等）、人脸的很多遮盖物（例如口罩、墨镜、头发、胡须等）、年龄等多方面因素的影响，因此区分个体变得异常困难[12]。

增量学习可以利用原有训练样本已得到的计算结果和新增数据来计算新的参数，不再需要原来的样本数据，从而降低学习的开销，提高算法处理数据的运行效率。

3 文献调研

3.1 国内外研究现状

人脸识别研究，起源于l9世纪末法国人Sir Franis Gahon的工作。到20世纪90年代，开始作为一个独立学科快速发展起来。人脸识别研究的发展大致分成三个阶段：第一阶段以Allen和Parke为代表，主要研究人脸识别所需要的面部特征。研究者用计算机实现了较高质量的人脸灰度图模型。这阶段的工作特点是识别过程全部依赖于操作人员。第二阶段是人机交互式识别阶段，其中用几何特征参数来表示人脸正面图像是以Harmon和Lesk为代表，将人脸面部特征用多维特征矢量表示出来，并设计了基于这一特征表示法的识别系统。而以Kaya和Kobayashi为代表，则采用了统计识别的方法、用欧氏距离来表示人脸特征。这两类方法都摆脱不了人的干预。第三阶段是真正的机器自动识别阶段，近十余年来，随着高速度高性能计算机的发展，人脸模式识别方法有了较大的突破，提出了多种机器全自动识别系统，人脸识别技术进人了实用化阶段。如Eyematic公司研发的人脸识别系统。我国清华大学的“十五”攻关项目《人脸识别系统》也通过了由公安部主持的专家鉴定。

目前人脸识别的难点主要存在于以下几个方面：

（1）光照变化是影响人脸识别性能的最关键因素，对该问题的解决程度关系着人脸识别实用化进程的成败，在人脸图像预处理或者归一化阶段，尽可能地补偿乃至消除其对识别性能的影响。

（2）成像角度及成像距离等因素的影响，即人脸的姿态的变化，会垂直于图像平面的两个方向的深度旋转，会造成面部信息的部分缺失。

（3）不同年龄的人脸有着较大的差别。身份证是以前照的，在逃犯的照片也是以前的，因此在公安部门的实际应用中，年龄问题是一个最突出的问题。

（4）采集图像的设备较多，主要有扫描仪、数码相机、摄像机等。由于成像的机理不同，形成了同类人脸图像的识别率较高，而不同类间人脸图像识别率较低的情况。随着人脸识别技术的发展，这一问题也将逐步得到解决。

（5）人脸图像的数据量巨大。目前由于计算量的考虑，人脸定位和识别算法研究大多使用尺寸很小的灰度图像。一张64*64像素的256级灰度图像就有4096个数据，每个数据有256种可能的取值。定位和识别算法一般都很复杂，在人脸库较大的情况下，计算量十分大，很多情况下速度令人难以忍受。而灰度数据事

实上是丧失了色彩、运动等有用信息的。如果使用全部的有用信息，计算量就更大了[13]。

3.2 常用人脸识别算法

人脸识别的方法主要有：基于几何特征的人脸识别方法、基于模板匹配的人脸识别方法、基于小波特征的人脸识别方法和基于模型的人脸识别方法。其中基于模型的人脸识别方法包括特征脸法(Eigenface)、神经网络法(NN)、隐马尔可夫模型方法(HMM)等方法[14]。本次实验我们采用的是基于特征脸的人脸识别方法[15]。

3.2.1 基于回归模型的人脸识别方法

LRC 算法是 Naseem 等人 2010 年提出的一种用于人脸识别的线性回归分类器(Linear Regression Classification，LRC)，它本质上是一个最近子空间分类器(Nearest Space, NS)[16]。基于稀疏表示的方法在人脸识别方面体现出优异的性能，然而随后的研究显示，稀疏性对于人脸识别的识别效果并不起主要作用。有研究认为是协同表示(Collaborative Representation,CR)的机制而非稀疏性约束在该类算法中起主体作用，并将稀疏表示分类器(Sparse Representation based Classification,SRC)视为协同表示分类器(Collaborative Representation based Classification,CRC)的一个特例[17]。

LRC 算法认为：

（1）同一个人的人脸图像位于一个线性子空间中，这个线性子空间可由训练样本张成；

（2）任何一个人的人脸图像可以由这个人的训练图像线性表示；

（3）待检测人脸图像应当与训练集中同一类图像张成的子空间距离最小。

3.2.2 基于神经网络的人脸识别方法

基于神经网络的人脸识别方法就是利用神经网络的学习能力和分类能力对人脸进行特征提取与识别，见图 1 所示。目前常用的人工神经网络方法是 BP 神经网络、自组织神经网络、径向基函数神经网络。径向基函数神经网络与 BP 网络一样都是多层前向网络，它以径向基函数作为基准，以高斯函数作为隐含层的

激励函数。这种网络的学习速率快、函数逼近、模式识别等能力均优于 BP 神经网络，并广泛应用于模式识别、图像处理等方面。但是这种网络比 BP 网络所用的神经元数目要多得多，使它的应用受到了一定的限制[18]。

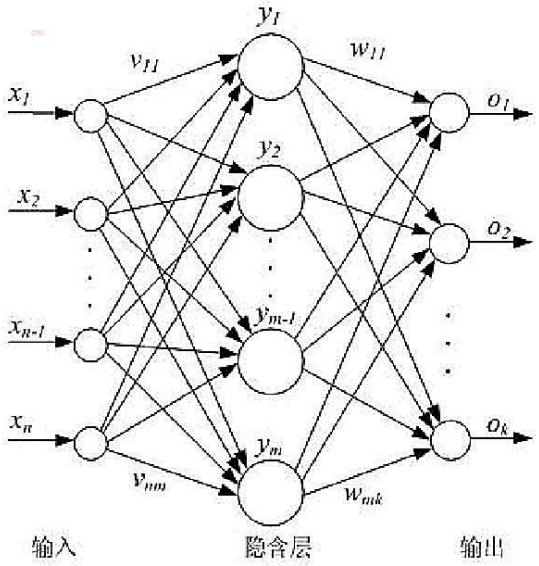


图 1 神经网络示意图

3.2.3 基于特征脸的人脸识别方法

特征脸方法是从主成分分析(PCA)导出的一种人脸识别和描述技术。它将包含人脸的图像区域看作一随机向量，采用 K-L 变换得到正交 K-L 基，对应其中较大特征值的基具有与人脸相似的形状，因此又被称为特征脸。利用这些基的线性组合可以描述、表达和逼近人脸图像，所以可进行人脸识别与合成。识别过程就是将人脸图像映射到由特征脸组成的子空间上，并比较其在特征脸空间中的位置，然后利用对图像的这种投影间的某种度量来确定图像间的相似度，最常见的就是选择各种距离函数来进行度量分类实现人脸识别。

主成分分析原理：

对于一个样本资料，观测 p 个变量 $x_1, x_2, \cdots x_p$， n 个样品的数据资料阵为：

$$X = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \vdots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{np} \end{pmatrix} = \left(x_1, x_2, \cdots x_p\right)$$

其中：$x_j=\begin{pmatrix}x_{1j}\\x_{2j}\\\vdots\\x_{nj}\end{pmatrix}$，　$j=1,2,\cdots p$

主成分分析就是将 p 个观测变量综合成为 p 个新的变量（综合变量），即

$$\begin{cases}F_1=a_{11}x_1+a_{12}x_2+\cdots+a_{1p}x_p\\F_2=a_{21}x_1+a_{22}x_2+\cdots+a_{2p}x_p\\\cdots\\F_p=a_{p1}x_1+a_{p2}x_2+\cdots+a_{pp}x_p\end{cases}$$

简写为：

$$F_j=\alpha_{j1}x_1+\alpha_{j2}x_2+\cdots+\alpha_{jp}x_p$$

$$j=1,2,\cdots,p$$

要求模型满足以下条件：

①F_i,F_j 互不相关（$i\neq j$，　$i,j=1,2,\cdots,p$ ）

②F_1 的方差大于 F_2 的方差大于 F_3 的方差，依次类推

③$a_{k1}{}^2+a_{k2}{}^2+\cdots+a_{kp}{}^2=1$　　$k=1,2,\cdots p.$

于是，称 F_1 为第一主成分，F_2 为第二主成分，依此类推，有第 p 个主成分。主成分又叫主分量。这里 a_{ij} 我们称为主成分系数。

上述模型可用矩阵表示为：

$F=AX$，其中

$$F=\begin{pmatrix}F_1\\F_2\\\vdots\\F_p\end{pmatrix}\qquad X=\begin{pmatrix}x_1\\x_2\\\vdots\\x_p\end{pmatrix}$$

$$A=\begin{pmatrix}a_{11}&a_{12}&\cdots&a_{1p}\\a_{21}&a_{22}&\cdots&a_{2p}\\\vdots&\vdots&\vdots&\vdots\\a_{p1}&a_{p2}&\cdots&a_{pp}\end{pmatrix}=\begin{pmatrix}a_1\\a_2\\\vdots\\a_p\end{pmatrix}$$

A 称为主成分系数矩阵。

本实验将采用基于主成分分析的方法实现人脸识别。

3.3 利用增量学习改进的人脸识别

增量学习可以利用原有训练样本已得到的计算结果和新增数据来计算新的参数，不再需要原来的样本数据，从而降低学习的开销，提高算法处理数据的运行效率。

在 3.2.3 节提到了基于主成分分析的人脸识别算法。在这里，主要有三个增量的模型：

模型 1：对平均像素矩阵使用增量计算：

$$M = M1 \times \frac{n1}{n1+n2} + M2 \times \frac{n2}{n1+n2}$$

上式中，M1、M2 分别为原训练集平均像素矩阵和新增训练集平均像素矩阵，n1、n2 分别为原训练集图片数目和新增训练集图片数目。

```
m2 = mean(Tz,2); %  平均图像/列平均（每一副图像的对应象素求平均）m=(1/P)*sum(Tj's) (j=1 : P)
Train_Number2 = size(Tz,2);%列数，训练图片张数
Train_Number=Train_Number1+Train_Number2;     %增量计算得到新的图片数目
m=m1*Train_Number1/Train_Number+m2*Train_Number2/Train_Number; %增量计算训练均值
```

模型 2：对 PCA 过程中每一张图像与均值图像的方差矩阵 A 使用增量计算：

```
Az = [];
  for i = 1:Train_Number2 %对每一列
       temp = double(Tz(:,i))-m; %每一张增加图与均值的差异
        Az=[Az temp]; %新增方差矩阵
  end
A=[A1 Az];       %增量计算新的差值矩阵
```

模型 3：对协方差矩阵的特征向量使用增量计算：

```
Lz= Az'*Az;%计算新的协方差矩阵
[V,D] = eig(Lz); %对角线上的元素是 L|C 的特征值.V:以特征向量为列的满秩矩阵，D：特征值对角矩阵。即 L*V = V*D.
```

```
L_eig_vec = []; %特征值向量
for i = 1 : size(V,2)    %对每个特征向量
%        if( D(i,i)>0 )    %特征值大于 0 时
            L_eig_vec = [L_eig_vec V(:,i)];     %集中对应的特征向量
%        end
end
Eigenfaces2 = Az*L_eig_vec; % 计算协方差矩阵 C 的特征向量，得到降维了的特征,A 为每一张图像与均值图像的方差构成的矩阵，
Eigenfaces=[Eigenfaces1 Eigenfaces2];     %增量计算新的协方差矩阵的特征向量；
```

上述三个模型被封装在函数 ADDEigenfaceCore 中。

4 算法与编程

4.1 编程流程

完整的 IPCA 人脸识别的应用包括以下几个步骤：

⑥ 人脸图像预处理；

⑦ 读入人脸库；

⑧ 训练形成特征子空间；

⑨ 把训练图像和测试图像投影到特征子空间上；

⑩ 选择一定的距离函数进行识别。

⑪ 输入新的训练图片；

⑫ 利用 IPCA 更新特征子空间；

⑬ 选择一定的距离函数进行识别；

⑭ 重复第 6 步到第 8 步。

因为我们在本实验侧重于人脸识别这一过程，所以，对于人脸图像预处理这一部分不用进行编程实现，而是从老师给定的数据库：Yale_32x32，Yale_64x64，ORL_32 x32，ORl_64 x64，YaleB_32x32 中选择数据库进行测试。

所以本实验的算法实现流程如下图 2、图 3 所示：

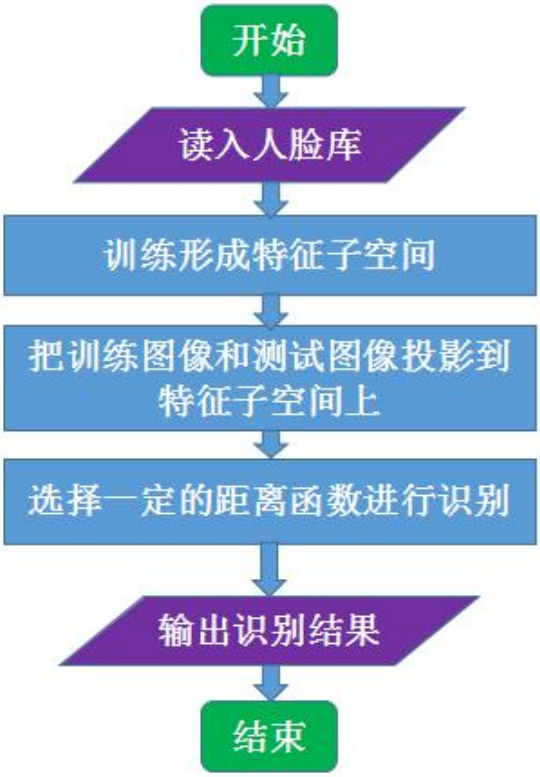


图 2 本实验 PCA 算法流程图

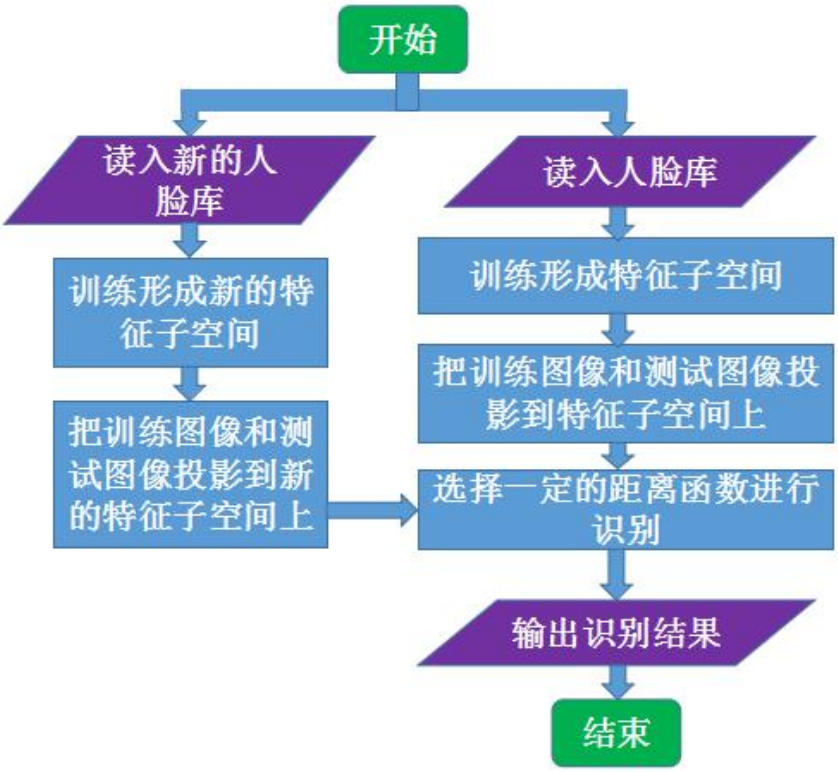


图 3 本实验 IPCA 算法流程图

训练过程如下图所示：

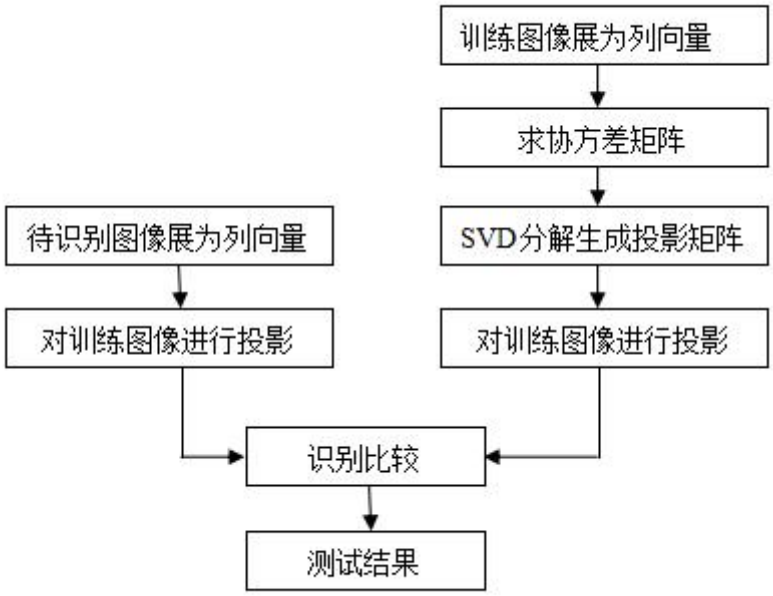


图 4 本实验人脸识别训练过程

4.2 文件结构

实验编写函数 M 文件：

/ADDEigenfaceCore.m　　增量人脸识别训练函数

/EigenfaceCore.m　　人脸识别训练函数，生成训练数据

/imageshow.m　　显示图片命令行文件

/Loaddata　　载入数据库函数

/main　　本次实验主函数命令行文件

/Recognition　　人脸识别识别函数，输出识别结果

/testfunction　　检测用的命令行文件

其中，人脸识别数据库、实验结果、训练结果这三个文件夹结构如下：

/人脸识别数据库

/ORL_32x32.mat　　400 张照片，共 40 个人

/ORL_64x64.mat　　400 张照片，共 40 个人

/Yale_32x32.mat　　165 张照片，共 15 个人

/Yale_64x64.mat　　165 张照片，共 15 个人

/YaleB_32x32.mat　　2414 张照片，共 38 个人

/实验结果　　实验结果存储文件夹

/ORL_32x32

/ORL_64x64

/Yale_32x32

/Yale_64x64

/YaleB_32x32

/训练结果　　训练结果存储文件夹

/ORL_32x32

/ORL_64x64

/Yale_32x32

/Yale_64x64

/YaleB_32x32

整个项目的文件结构如下图所示。

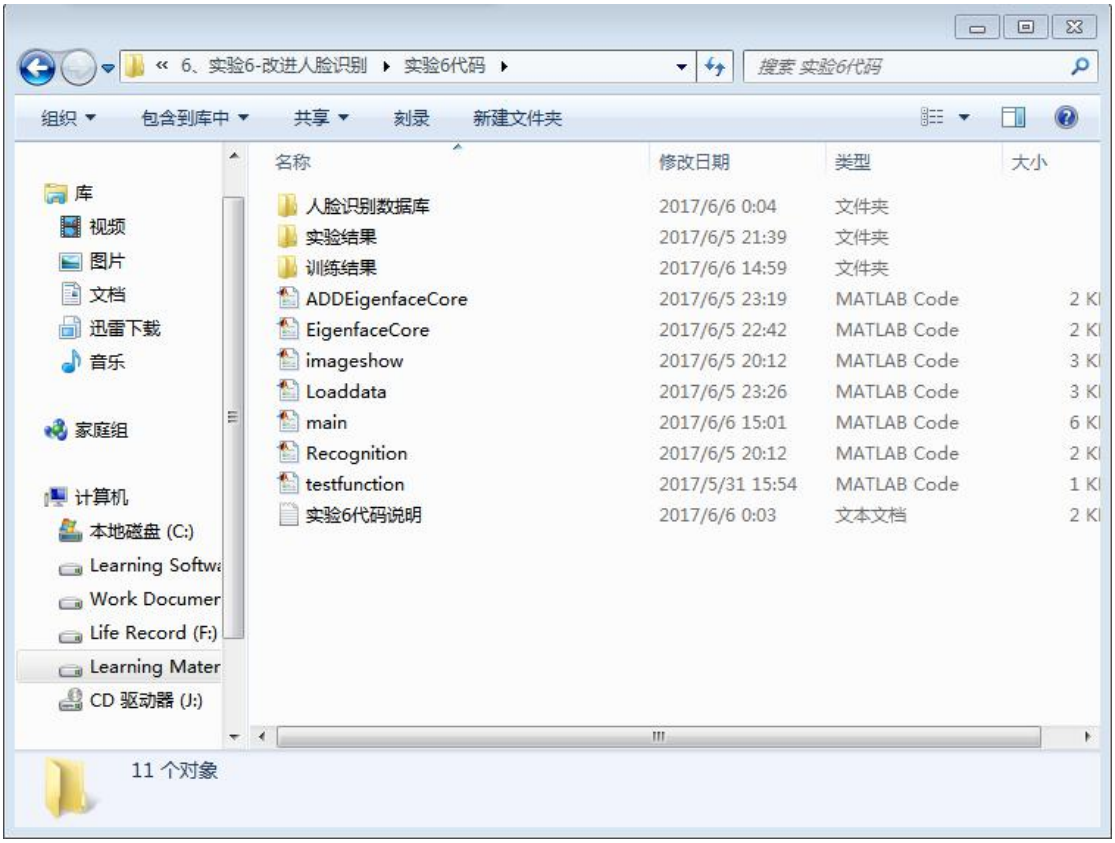


图 5 人脸识别代码存储文件夹

4.3 编程细节

1、规范函数命名和变量命名：在实验 5，我们两个人合作，在函数命名、变量命名过程中欠缺考虑。本次实验代码均进行了修正：以下是部分变量命名示例。

```
%     TrainDatabase          训练数据集
%     TrainArray             训练图片编号索引矩阵，第一列为图片编号，第二列为人物编号
%     TestDatabase           测试数据集
%     TestArray              测试图片编号索引矩阵，第一列为图片编号，第二列为人物编号
%     linenum                 fea 矩阵的行数
%     rownum                   fea 矩阵的列数，也是 gnd 矩阵的列数
%     fenbianlv              图片像素矩阵维数
%     Testnum                 被测试图片数目
%     accuracy               识别正确率
%     time                   运行时间
```

2、尽可能提高函数和代码的普适性，将功能更具体地划分模块并用函数封装起来。

下面这个函数实现对数据库的读取，而只需输入数据库的路径，例如：G:\3 作业\2017 数学实验作业-陈艺荣\6、实验 6-改进人脸识别\实验 6 代码\人脸识别数据库\ORL_32x32.mat。

```
function
[ TrainDatabase,TrainArray,TestDatabase,TestArray,linenum,rownum,fenbianlv,Testnum,Trainnum ] = Loaddata( DataSetName )
%Loaddata.m  读取 DataSetName 指定的路径的数据
```

下面这个函数实现增量 PCA 的训练过程，封装了整个训练过程。

```
function [Train_Number,m,A,Eigenfaces]
=ADDEigenfaceCore(Train_Number1,m1,A1,Eigenfaces1,Tz)
% ADDEigenfaceCore.m
```

下面的代码实现了读取指定数据库并作预处理的功能。

datasetsnum;

datasets = {'Yale_32x32','Yale_64x64','YaleB_32x32','ORL_32x32','ORL_64x64'}; %所有测试数据集

DataSetName=['人脸识别数据库\',datasets{ datasetsnum },'.mat']; %载入数据库路径，在命令行窗口显示正在测试的数据集

[TrainDatabase,TrainArray,TestDatabase,TestArray,linenum,rownum,fenbianlv,Testnum,Trainnum]=Loaddata(DataSetName); %载入 DataSetName 指定的数据库

4.4 实现代码

实验编写函数 M 文件：

/ADDEigenfaceCore.m	增量人脸识别训练函数
/EigenfaceCore.m	人脸识别训练函数，生成训练数据
/imageshow.m	显示图片命令行文件
/Loaddata	载入数据库函数
/main	本次实验主函数命令行文件
/Recognition	人脸识别识别函数，输出识别结果

1、/main 本次实验主函数命令行文件

```
%%
%main.m        人脸识别测试脚本文件
%华南理工大学  电子与信息学院  数学实验 6
%15 电科卓越班  陈艺荣  何晨晖
%文件说明：本文件是人脸识别测试用 M 文件
%运行前请确保 EigenfaceCore.m 文件和 main.m 文件和本文件在同一路径下
%本次测试使用
'Yale_32x32.mat','Yale_64x64.mat','YaleB_32x32.mat','ORL_32x32.mat','ORL_64x6
```

```
4.mat'6 个数据库当中的数据
%其中，在测试前应确保
'Yale_32x32.mat','Yale_64x64.mat','YaleB_32x32.mat','ORL_32x32.mat','ORL_64x6
4.mat'6 个数据库已
%经和本文件在同一路径下
%  人脸识别数据库
% /ORL_32x32.mat          400 张照片，共 40 个人
% /ORL_64x64.mat          400 张照片，共 40 个人
% /Yale_32x32.mat         165 张照片，共 15 个人
% /Yale_64x64.mat         165 张照片，共 15 个人
% /YaleB_32x32.mat        2414 张照片，共 38 个人
%     TrainDatabase         训练数据集
%     TrainArray            训练图片编号索引矩阵，第一列为图片编号，第二列
为人物编号
%     TestDatabase          测试数据集
%     TestArray             测试图片编号索引矩阵，第一列为图片编号，第二列
为人物编号
%     linenum               fea 矩阵的行数
%     rownum                fea 矩阵的列数，也是 gnd 矩阵的列数
%     fenbianlv             图片像素矩阵维数
%     Testnum               被测试图片数目
%     accuracy              识别正确率
%     time                  运行时间
%     Percent               初始 PCA 的训练图片数目和增量 PCA 训练图片数目
的比例

%   Chen Yirong and He Chenhui  修改于 2017-06-05
%   15 电科卓越班  陈艺荣  何晨晖
%   代码编辑 matlab 版本：MATLAB R2014a
```

```
%%
%****准备工作****%
clear all   %清空工作空间
clc         %清空命令行窗口
close all   %关闭其他 figure 窗口

%%
%****读入数据库****%
for datasetsnum = 1: 4   %可以自己修改，使其只运行一个或几个数据集
clearvars -except datasetsnum;
    for Percentnum = 1: 9
    %       clear all   %清空工作空间
    Percent=0.1*Percentnum;     %取值范围为 0 到 1，修改此处改变
        datasetsnum;
        datasets =
{'Yale_32x32','Yale_64x64','ORL_32x32','ORL_64x64','YaleB_32x32'};     %所有测试数据集
        Percentsets={'0.1','0.2','0.3','0.4','0.5','0.6','0.7','0.8','0.9'};
        DataSetName=['人脸识别数据库\',datasets{ datasetsnum },'.mat'];     %载入数据库路径，在命令行窗口显示正在测试的数据集

[TrainDatabase,TrainArray,TestDatabase,TestArray,linenum,rownum,fenbianlv,Testnum,Trainnum]=Loaddata(DataSetName);   %载入 DataSetName 指定的数据库
        Judge=zeros(1,Testnum);   %创建存储判断值矩阵，元素取值只有 1 和 0 两种可能，判断正确则为 1，错误则为 0
        temresult=zeros(1,Testnum); %创建存储测试识别结果矩阵：人物编号
        temoutputname=zeros(1,Testnum); %存储测试识别结果矩阵：图片编号
    %%
    %****训练人脸图像****%
```

```
tic;    %记录时间
Tnum=floor(Percent*Trainnum);
T=TrainDatabase(:,1:Tnum);    %T 是一个二维矩阵，若有 P 张 M*N 的图像，则 T 为 M*N 行 P 列的矩阵。修改此处改变训练图片数目
Tz=TrainDatabase(:,Tnum+1:Trainnum);  %Tz 是新增的训练集
[Train_Number1,m1,A1,Eigenfaces1] = EigenfaceCore(T); %用 PCA 原理决定人脸图像的最优特征，得到一个二维矩阵，包含训练图像向量，返回三个输出
[Train_Number,m,A,Eigenfaces]
=ADDEigenfaceCore(Train_Number1,m1,A1,Eigenfaces1,Tz);  %利用增量 PCA
% 返回值：m：(M*Nx1)训练均值;
%          Train_Number:训练图片张数
%          Eigenfaces：(M*Nx(P-1))训练集协方差矩阵的特征向量；
%          A：(M*NxP) 每一张图像与均值图像的方差矩阵
 DataSetName=['训练结果
\',datasets{ datasetsnum },'\',datasets{ datasetsnum },'_',Percentsets{Percentnum},'_Tr
ainresult','.mat'];  %存储训练结果的路径
 save(DataSetName,'m','A','Eigenfaces');    %存储训练结果
%%
%****进行人脸识别****%
    for shibienum=1:Testnum
    TestNumber=TestArray(shibienum,1);  %读取测试图片编号
    RealNumber=TestArray(shibienum,2);  %确定所读取图片是第几个人，真实值
    OrignImage=TestDatabase(:,shibienum);  %读出待测试图片
    originim=reshape(OrignImage,fenbianlv,fenbianlv);  %行向量转变为像素点的矩阵
    im=originim;  %待测试图片矩阵
    OutputName = Recognition(im ,m ,A,Eigenfaces);    %输出人物索引
    temoutputname(shibienum)=TrainArray(OutputName,1);  %该数组用于
```

```
存储每一个测试样例的识别结果：图片编号
        temresult(shibienum)=TrainArray(OutputName,2);  %该数组用于存储每一个测试样例的识别结果：人物编号
        ShibieNumber=TrainArray(OutputName,2);

            if(ShibieNumber==RealNumber)  %比较获取判断矩阵用于计算识别准确率
                Judge(shibienum)=1;  %人脸识别正确，赋值 1
            else
                Judge(shibienum)=0;   %人脸识别错误，赋值 0
            end
        end
    time=toc;
    %%
    %****分析识别效果****%
    CorrectNumber=0;  %记录正确测试人数
        for j=1:Testnum
            if( Judge(j)==1 )
                CorrectNumber=CorrectNumber+1;
            end
        end
    accuracy=CorrectNumber/Testnum;       %计算识别正确率，存储在 accuracy 中
     DataSetName=['实验结果\',datasets{ datasetsnum },'\',datasets{ datasetsnum },'_',Percentsets{Percentnum},'_Testresult','.mat'];  %存储实验结果的路径

save(DataSetName,'TestArray','temoutputname','temresult','Judge','accuracy','time');
 %存储训练结果
```

```
        fprintf(datasets{ datasetsnum },'\n');     %输出数据集名字
        fprintf('原始训练集占比：');     %原始训练集占比：
        fprintf(Percentsets{Percentnum},'\n');     %Percentsets{Percentnum}
        fprintf('识别的正确率为:%i\n',accuracy);     %输出识别正确率，百分数格式
        fprintf('代码运行时间为:%i\n',time);     %输出识别正确率，百分数格式
        end
end    %for datasetsnum = 1: 5    %可以自己修改，使其只运行一个或几个数据集
 save('实验结果\TestResult.mat','TestResult_Accuracy','TestResult_Time');     %存储训练结果
%%
%****绘制测试结果****%
%绘制不同情形的精确度曲线
Percent=[0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9];    %占比
figure('Name','实验 6 精确度','Position',[80,80,500,300]);    %创建 1 个新显示图形的窗口
hold on;
plot(100*Percent,100*TestResult_Accuracy(1,1:9),'r',100*Percent,100*TestResult_Accuracy(2,1:9),'y',100*Percent,100*TestResult_Accuracy(3,1:9),'b',100*Percent,100*TestResult_Accuracy(4,1:9),'g');    %绘制正确率曲线
legend('Yale\_32x32','Yale\_64x64','ORL\_32x32','ORL\_64x64');
xlabel('Percent/%');
ylabel('Accuracy/%');
%绘制不同情形的时间曲线
figure('Name','实验 6 时间','Position',[580,80,500,300]);    %创建 1 个新显示图形的窗口
hold on;
plot(100*Percent,TestResult_Time(1,1:9),'r',100*Percent,TestResult_Time(2,1:9),'y',100*Percent,TestResult_Time(3,1:9),'b',100*Percent,TestResult_Time(4,1:9),'g');    %绘制时间曲线
legend('Yale\_32x32','Yale\_64x64','ORL\_32x32','ORL\_64x64');
```

```
xlabel('Percent/%');
ylabel('Time/s');
%end for main function
% by Chen Yirong and He Chenhui
```

2、/Loaddata　　载入数据库函数

```
function
[ TrainDatabase,TrainArray,TestDatabase,TestArray,linenum,rownum,fenbianlv,Testn
um,Trainnum ] = Loaddata( DataSetName )
%Loaddata.m  读取 DataSetName 指定的路径的数据
%TrainDatabase          训练数据集
%TrainArray             训练图片编号索引矩阵
%TestDatabase           测试数据集
%TestArray              测试图片编号索引矩阵
%linenum                fea 矩阵的行数
%rownum                 fea 矩阵的列数，也是 gnd 矩阵的列数
%fenbianlv              图片像素矩阵维数
%Testnum                被测试图片数目

%   Chen Yirong and He Chenhui  修改于 2017-06-05
%   15 电科卓越班  陈艺荣  何晨晖
%   代码编辑 matlab 版本：MATLAB R2014a

fea=[];
gnd=[];
load(DataSetName);   %读取数据库
    % /ORL_32x32.mat          400 张照片，共 40 个人
    %fea:400x1024        gnd:400x1
    % /ORL_64x64.mat          400 张照片，共 40 个人
    %fea:400x4096        gnd:400x1
```

```
% /Yale_32x32.mat        165 张照片，共 15 个人
%fea:165x1024         gnd:165x1
% /Yale_64x64.mat        165 张照片，共 15 个人
%fea:165x4096         gnd:165x1
% /YaleB_32x32.mat       2414 张照片，共 38 个人
%fea:2414x1024          gnd:2414x1
%     linenum           fea 矩阵的行数
%     rownum            fea 矩阵的列数，也是 gnd 矩阵的列数
[linenum,rownum]=size(fea);       %获取数据库维数
fenbianlv=sqrt(rownum);   %获取图片像素矩阵维数

%****创建训练集和数据测试集****%
trainnum=1;       %创建训练集时用到的临时编号变量
testnum=1;       %创建测试集时用到的临时编号变量
%   TrainDatabase       训练集
%   TrainArray          训练集图片索引
%   TestDatabase        测试集
%   TestArray           测试集图片索引
        for temnum=1:linenum-1     %检索每一张图片
                if(gnd(temnum)==gnd(temnum+1))   %创建训练集
                        TEMarrayline=fea(temnum,:);     %提取图片
                        TEMarrayrow=reshape(TEMarrayline,rownum,1);   %行向量
转为列向量
                        TrainDatabase(:,trainnum)=TEMarrayrow(:,1);   %创建训练
数据集
                        TrainArray(trainnum,1)=temnum;   %训练图片编号索引矩阵
                        TrainArray(trainnum,2)=gnd(temnum);   %训练图片人物索引
矩阵
                        trainnum=trainnum+1;
```

```
                else    %创建测试集，每一个人的最后一张图片用作测试数据
                    TEMarrayline=fea(temnum,:);     %提取图片
                    TEMarrayrow=reshape(TEMarrayline,rownum,1);    %行向量转为列向量
                    TestDatabase(:,testnum)=TEMarrayrow(:,1);       %创建测试数据集
                    TestArray(testnum,1)=temnum;    %测试图片编号索引矩阵
                    TestArray(testnum,2)=gnd(temnum);    %测试图片人物索引矩阵
                    testnum=testnum+1;
                end
            end
        Trainnum=trainnum-1;    %训练图片总数目
        %特殊情形：处理最后一张图片
        TEMarrayline=fea(linenum,:);     %提取图片
        TEMarrayrow=reshape(TEMarrayline,rownum,1);    %行向量转为列向量
        TestDatabase(:,testnum)=TEMarrayrow(:,1);       %创建测试数据集
        TestArray(testnum,1)=linenum;    %测试图片编号索引矩阵
        TestArray(testnum,2)=gnd(linenum);    %测试图片人物索引矩阵
Testnum=testnum;     %被测试图片数目

end
```

3、/EigenfaceCore.m　　人脸识别训练函数，生成训练数据

```
function [Train_Number,m, A, Eigenfaces] = EigenfaceCore(T)
% 用 PCA 原理决定人脸图像的最优特征，得到一个二维矩阵，包含训练图像向量，返回三个输出
% 参数:T 包含训练集中所有的图像信息集合，
% 返回值：m：(M*Nx1)训练均值;
```

```
%            Eigenfaces：(M*Nx(P-1))训练集协方差矩阵的特征向量；
%            A：(M*NxP) 每一张图像与均值图像的方差矩阵

%  Chen Yirong and He Chenhui 修改于 2017-06-05
%  15 电科卓越班 陈艺荣 何晨晖
%  代码编辑 matlab 版本：MATLAB R2014a

m = mean(T,2); % 平均图像/列平均（每一副图像的对应象素求平均）
m=(1/P)*sum(Tj's) (j=1 : P)
Train_Number = size(T,2);%列数，训练图片张数
%计算机每一张图片到均值图像的方差
A = [];
   for i = 1 : Train_Number%对每一列
        temp = double(T(:,i))-m; %每一张图与均值的差异
         A=[A temp]; %方差矩阵
   end
%降维
L = A'*A; % L 是协方差矩阵 C=A*A'的转置.
[V,D] = eig(L); %对角线上的元素是 L|C 的特征值.V:以特征向量为列的满秩矩阵，
D：特征值对角矩阵。即 L*V = V*D.
L_eig_vec = []; %特征值向量
for i = 1 : size(V,2)   %对每个特征向量
%       if( D(i,i)>0 )   %特征值大于 0 时
          L_eig_vec = [L_eig_vec V(:,i)];     %集中对应的特征向量
%       end
end
Eigenfaces = A*L_eig_vec; % 计算协方差矩阵 C 的特征向量，得到降维了的特
征,A 为每一张图像与均值图像的方差构成的矩阵，
end
```

4、/ADDEigenfaceCore.m　　增量人脸识别训练函数

```
function [Train_Number,m,A,Eigenfaces]
=ADDEigenfaceCore(Train_Number1,m1,A1,Eigenfaces1,Tz)
% ADDEigenfaceCore.m
% 用增量 PCA 原理决定人脸图像的最优特征，得到一个二维矩阵，包含训练图像向量，返回三个输出
% 参数:Train_Number1     原训练集图片数目
%         m1                  原训练集图片均值
%         A1                  原训练集每一张图像与均值图像的方差矩阵
%         Eigenfaces1       原训练集(M*Nx(P-1))训练集协方差矩阵的特征向量；
%         Tz                  包含新增训练集中所有的图像信息集合，
% 返回值：Train_Number：训练图片总数
%           m：(M*Nx1)训练均值;
%           Eigenfaces：(M*Nx(P-1))训练集协方差矩阵的特征向量；
%           A：(M*NxP) 每一张图像与均值图像的方差矩阵

%   Chen Yirong and He Chenhui 修改于 2017-06-05
%   15 电科卓越班 陈艺荣 何晨晖
%   代码编辑 matlab 版本：MATLAB R2014a

m2 = mean(Tz,2); % 平均图像/列平均（每一副图像的对应象素求平均）
m=(1/P)*sum(Tj's) (j=1 : P)
Train_Number2 = size(Tz,2);%列数，训练图片张数
Train_Number=Train_Number1+Train_Number2;     %增量计算得到新的图片数目
m=m1*Train_Number1/Train_Number+m2*Train_Number2/Train_Number; %增量计算训练均值
Az = [];
   for i = 1:Train_Number2 %对每一列
         temp = double(Tz(:,i))-m; %每一张增加图与均值的差异
```

```
        Az=[Az temp]; %新增方差矩阵
    end
A=[A1 Az];      %增量计算新的差值矩阵
Lz= Az'*Az;%计算新的协方差矩阵
[V,D] = eig(Lz); %对角线上的元素是 L|C 的特征值.V:以特征向量为列的满秩矩阵，D：特征值对角矩阵。即 L*V = V*D.
L_eig_vec = []; %特征值向量
for i = 1 : size(V,2)   %对每个特征向量
%       if( D(i,i)>0 )   %特征值大于 0 时
            L_eig_vec = [L_eig_vec V(:,i)];     %集中对应的特征向量
%       end
end
Eigenfaces2 = Az*L_eig_vec; %  计算协方差矩阵 C 的特征向量，得到降维了的特征,A 为每一张图像与均值图像的方差构成的矩阵，
Eigenfaces=[Eigenfaces1 Eigenfaces2];     %增量计算新的协方差矩阵的特征向量;
end
```

5、/Recognition　　人脸识别识别函数，输出识别结果

```
function OutputName = Recognition(TestImage, m, A, Eigenfaces)
%本函数用于进行人脸识别并且输出识别结果
%使用本函数之前需要使用 function [m, A, Eigenfaces] = EigenfaceCore(T)获取图像特征数据
%           m：(M*Nx1)训练均值;
%           Eigenfaces：(M*Nx(P-1))训练集协方差矩阵的特征向量;
%           A：(M*NxP)  每一张图像与均值图像的方差矩阵

%   Chen Yirong and He Chenhui  修改于 2017-06-05
%   15 电科卓越班  陈艺荣  何晨晖
%   代码编辑 matlab 版本：MATLAB R2014a
```

```
%读入测试图像，测试前的准备工作
ProjectedImages=[];%映射图像
Train_Number=size(Eigenfaces,2);%列，降维后，
   for i =1:Train_Number       %对于每一个训练特征
          temp = Eigenfaces'*A(:,i);
          ProjectedImages=[ProjectedImages temp];   %得到  L_eig_vec;
   end
InputImage = TestImage;%读入测试图片
temp = InputImage;   %用临时矩阵装载图片
[irow , icol] = size(temp);%测试图片大小
InImage = reshape(temp, irow*icol ,1);%转置后转为一维
Difference = double(InImage)-m; % L_eig_vec'
%进行测试，特征比较
ProjectedTestImage = Eigenfaces'*Difference; %  测试图像的特征向量
Euc_dist = [];
for i = 1 : Train_Number     %对每列
      q = ProjectedImages(:,i);%取出训练图像
      temp=(norm(ProjectedTestImage-q))^2;%欧氏距离
      Euc_dist = [Euc_dist temp];
end
[Euc_dist_min , Recognized_index] = min(Euc_dist);%得到差值最小的图像的索引
号
OutputName = Recognized_index;%返回索引
end
```

5 实验结果

5.1 命令行输出

运行 main.m 文件，在 matlab 命令行中输出以下结果：

Yale_32x32 原始训练集占比：0.1 识别的正确率为:8.000000e-01

代码运行时间为:2.241166e-01

Yale_32x32 原始训练集占比：0.2 识别的正确率为:8.000000e-01

代码运行时间为:2.071952e-01

Yale_32x32 原始训练集占比：0.3 识别的正确率为:7.333333e-01

代码运行时间为:2.022604e-01

Yale_32x32 原始训练集占比：0.4 识别的正确率为:7.333333e-01

代码运行时间为:2.077413e-01

Yale_32x32 原始训练集占比：0.5 识别的正确率为:7.333333e-01

代码运行时间为:2.065555e-01

Yale_32x32 原始训练集占比：0.6 识别的正确率为:8.000000e-01

代码运行时间为:2.063981e-01

Yale_32x32 原始训练集占比：0.7 识别的正确率为:8.000000e-01

代码运行时间为:2.093359e-01

Yale_32x32 原始训练集占比：0.8 识别的正确率为:8.000000e-01

代码运行时间为:2.165223e-01

Yale_32x32 原始训练集占比：0.9 识别的正确率为:8.000000e-01

代码运行时间为:2.091166e-01

Yale_64x64 原始训练集占比：0.1 识别的正确率为:8.666667e-01

代码运行时间为:1.275626e+00

Yale_64x64 原始训练集占比：0.2 识别的正确率为:7.333333e-01

代码运行时间为:1.187121e+00

Yale_64x64 原始训练集占比：0.3 识别的正确率为:8.000000e-01

代码运行时间为:1.163476e+00

Yale_64x64 原始训练集占比：0.4 识别的正确率为:8.000000e-01

代码运行时间为:1.160848e+00

Yale_64x64 原始训练集占比：0.5 识别的正确率为:8.000000e-01

代码运行时间为:1.182465e+00

Yale_64x64 原始训练集占比：0.6 识别的正确率为:8.000000e-01

代码运行时间为:1.214061e+00

Yale_64x64 原始训练集占比：0.7 识别的正确率为:7.333333e-01

代码运行时间为:1.183454e+00

Yale_64x64 原始训练集占比：0.8 识别的正确率为:7.333333e-01

代码运行时间为:1.142595e+00

Yale_64x64 原始训练集占比：0.9 识别的正确率为:7.333333e-01

代码运行时间为:1.109929e+00

ORL_32x32 原始训练集占比：0.1 识别的正确率为:8.250000e-01

代码运行时间为:2.671855e+00

ORL_32x32 原始训练集占比：0.2 识别的正确率为:7.250000e-01

代码运行时间为:2.468880e+00

ORL_32x32 原始训练集占比：0.3 识别的正确率为:7.500000e-01

代码运行时间为:2.639563e+00

ORL_32x32 原始训练集占比：0.4 识别的正确率为:7.000000e-01

代码运行时间为:2.368847e+00

ORL_32x32 原始训练集占比：0.5 识别的正确率为:7.000000e-01

代码运行时间为:2.439601e+00

ORL_32x32 原始训练集占比：0.6 识别的正确率为:8.500000e-01

代码运行时间为:2.432003e+00

ORL_32x32 原始训练集占比：0.7 识别的正确率为:8.750000e-01

代码运行时间为:2.397536e+00

ORL_32x32 原始训练集占比：0.8 识别的正确率为:8.500000e-01

代码运行时间为:2.538822e+00

ORL_32x32 原始训练集占比：0.9 识别的正确率为:8.250000e-01

代码运行时间为:2.451671e+00

ORL_64x64 原始训练集占比：0.1 识别的正确率为:8.500000e-01

代码运行时间为:1.187890e+01

ORL_64x64 原始训练集占比：0.2 识别的正确率为:7.000000e-01

代码运行时间为:1.190482e+01

ORL_64x64 原始训练集占比：0.3 识别的正确率为:7.750000e-01

代码运行时间为:1.184104e+01

ORL_64x64 原始训练集占比：0.4 识别的正确率为:7.250000e-01

代码运行时间为:1.214329e+01

ORL_64x64 原始训练集占比：0.5 识别的正确率为:7.000000e-01

代码运行时间为:1.237598e+01

ORL_64x64 原始训练集占比：0.6 识别的正确率为:8.000000e-01

代码运行时间为:1.250795e+01

ORL_64x64 原始训练集占比：0.7 识别的正确率为:8.750000e-01

代码运行时间为:1.253277e+01

ORL_64x64 原始训练集占比：0.8 识别的正确率为:8.250000e-01

代码运行时间为:1.244203e+01

ORL_64x64 原始训练集占比：0.9 识别的正确率为:8.250000e-01

代码运行时间为:1.271669e+01

5.2 结果分析

本实验中，对各个数据集进行了如下处理：

对于每一个人的所有图片，只取最后一张图片作为测试集，其余图片用作训练集，其中，训练集又被划分为初始训练集和新增训练集。对初始训练集作 PCA 处理，对新增训练集和初始训练集作 IPCA 处理。其中，在分割初始训练集和新增训练集的时候，按 Percent 的不同划分，最后得到四个数据集不同分割比的识别精确率和代码运行时间。

用 matlab 绘制得到结果图：

四个数据集不同分割比（Percent）下的精确度（Accuracy）变化，如下图 6 所示。

四个数据集不同分割比（Percent）下的运行时间（Time）变化，如下图 7 所示。

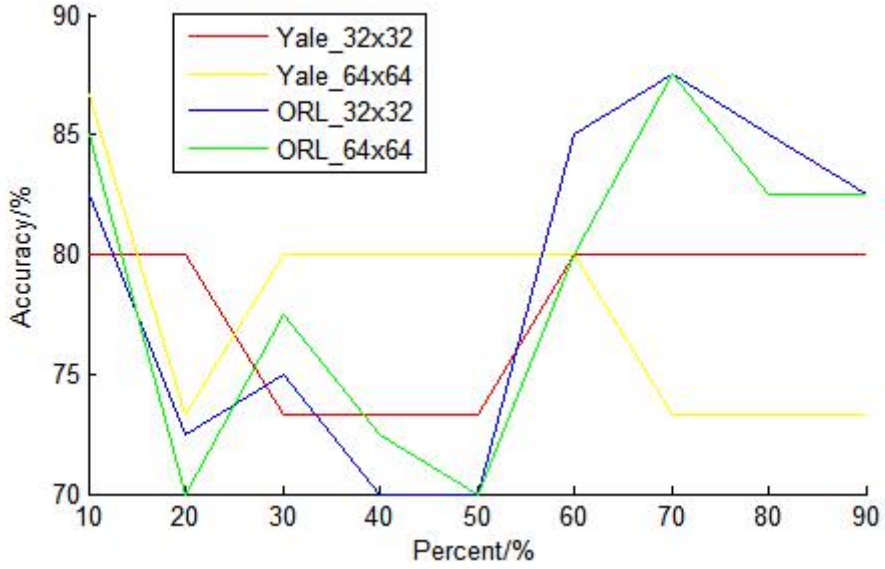


图 6 四个数据集不同分割比（Percent）下的精确度（Accuracy）

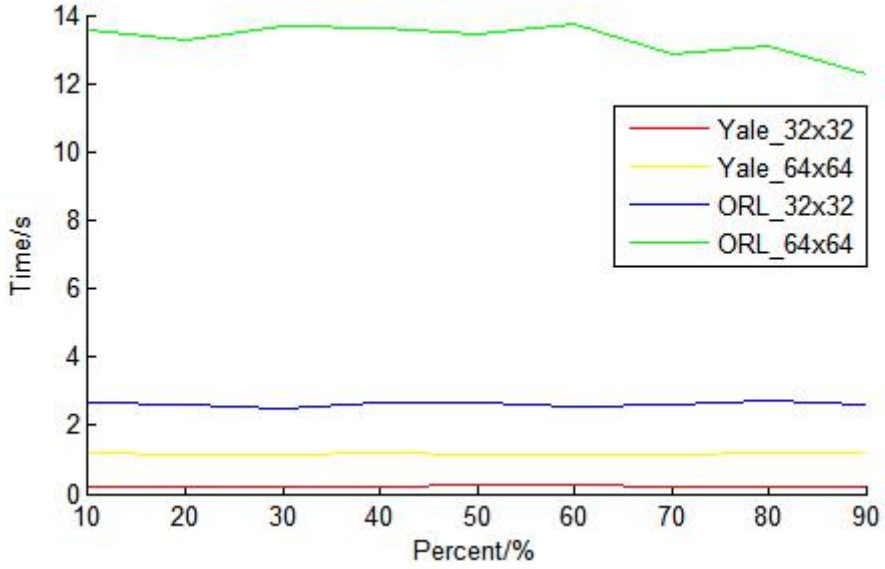


图 7 四个数据集不同分割比（Percent）下的运行时间（Time）

结论：由图 6 分析，Percent 在 60%到 70%的时候，各个数据集的人脸识别的精确度基本达到最大值。由此可以说明，每做一次增量 PCA，新增样本占总样本数目应该在 30%到 40%。或者说，已经存在的样本占新的总样本数目理应在 60%到 70%。由图 7 分析，Percent 在 60%到 70%的时候，各个数据集的人脸识别的时间基本达到最小值。

图 8、图 9 为本实验结果输出情况。

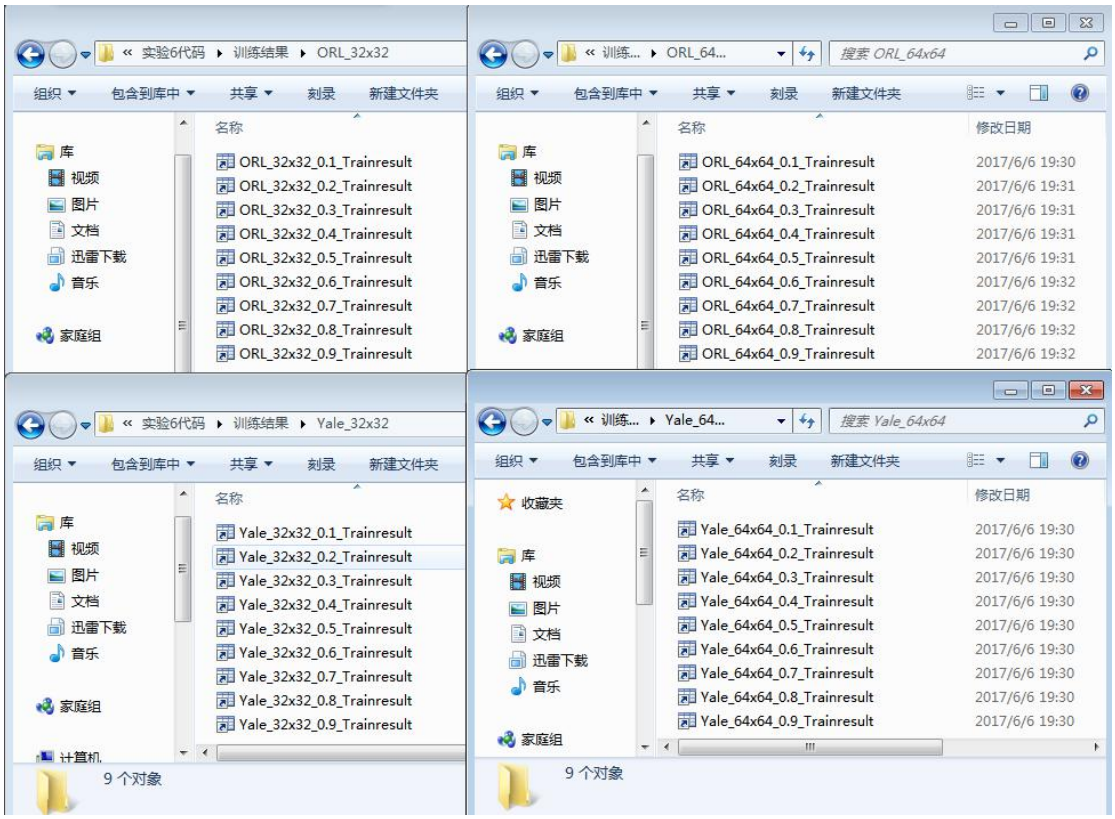


图 8 四个数据集的训练结果存储形式

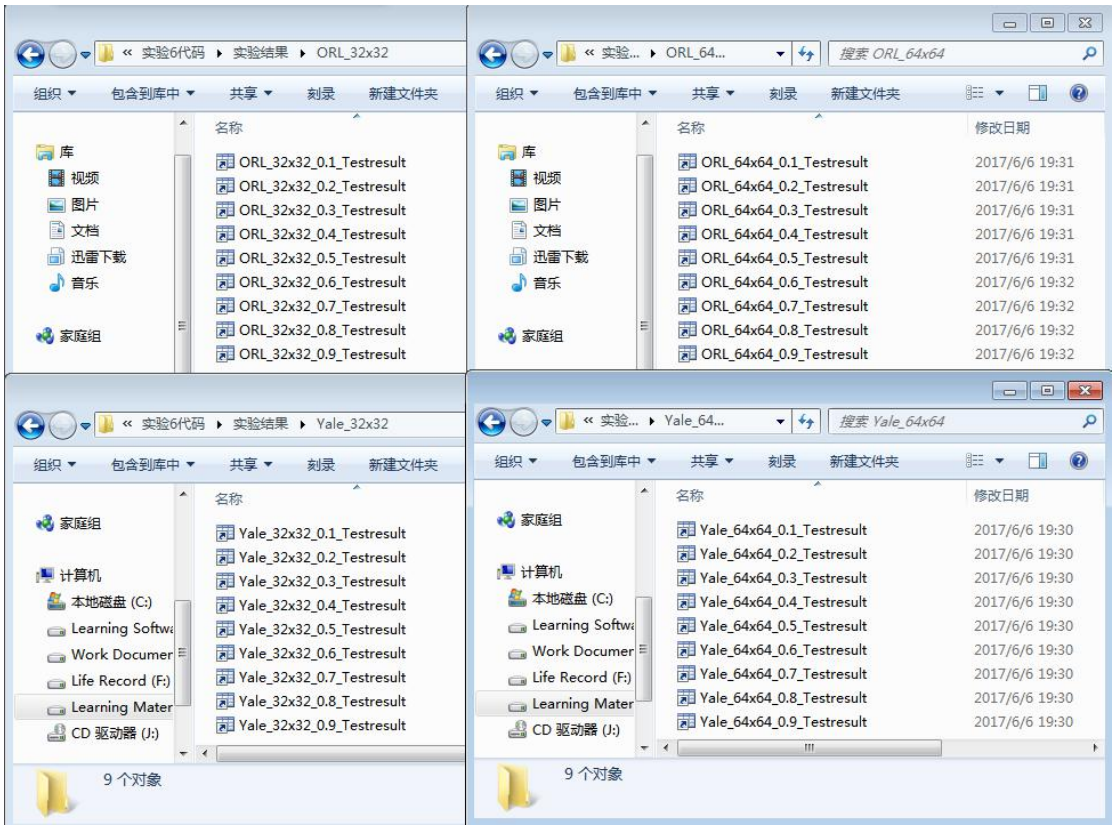


图 9 四个数据集的实验结果存储形式

6 实验总结和实验感悟

6.1 实验总结

1、在本次数学实验中，我了解了人脸识别技术的应用及其难点，发展与现状，研究内容与主要方法，及常用的人脸识别标准数据库。

2、通过对求解，我掌握了人脸识别的流程、KL 变换、特征脸方法（PCA），使用特征脸方法设计出人脸识别程序并计算出识别成功率。

3、在整个实验的设计以及代码的调试中，我们学会了模块化的思想，使得整个实验条理更清楚，同时，在编写代码的过程中，我和队友要考虑到自己所编写的代码还要给队友看，因此我们都尽量规范自己的代码并且加入了大量的注释帮助理解，这使得我们的沟通更加方便，合作效率高。

4、通过本次实验，我和队友学会了文献检索方法，包括使用华工图书馆提供的各种数据库以及使用谷歌学术搜索对人脸识别进行文献检索，这为我们的实验提供了很大的帮助。

5、本次实验，我们对代码作了大量的规范工作，对函数的接口作了进一步处理，特别考虑到函数的通用性、移植性。

6、本次实验，我们加强了代码的可扩充性，对每一个函数文件的代码都进行了严格的模块化处理。

7、本次实验，我们对实验结果作了可视化处理和规范统一的保存处理（包括训练结果和实验结果）。

6.2 实验感悟

1、整个实验过程中，通过多种识别方法的比较与测试，我发现每种人脸识别方法都各有优缺点，如何充分利用现有的各种人脸识别方法，发挥某一类方法的优点，克服某一类方法的缺点，将它们进行有效的综合和组合，也将是以后一个探索的方向。

2、同时，此次实验成功也得益于与队友的团结协作，也让我认识到了团队合作的重要性，还有也要感谢老师的大力指导与帮助，希望在以后的实验中能够收获更多知识和方法。

3、对于 matlab 编程，大局观和细节观都至关重要。

实验七　线性相关性

地　　点：	4 号楼　　4104 房；	**实验台号：**	66
实验日期与时间：	2017 年 06 月 07 日	**评　　分：**	
预习检查纪录：		**实验教师：**	刘小兰
电子文档存放位置：			
电子文档文件名：	卓越班-66-陈艺荣实验七		

批改意见：

1 实验目的

- 理解向量、向量组的线性组合与线性表示、向量组的线性相关与无关、最大线性无关组的概念；
- 掌握向量组线性相关和无关的有关性质及判别法；
- 掌握向量组的最大线性无关组和秩的性质和求法；
- 通过调味品配制问题理解上述知识在实际中的应用；
- 学会用 matlab 处理线性相关问题。

2 问题描述

某中药厂用 9 种中草药（A-I），根据不同的比例配制成了 7 种特效药，各用量成分见表 1（单位：克）

试解答：

（1）某医院要购买这 7 种特效药，但药厂的第 3 号药和第 6 号药已经卖完，请问能否用其他特效药配制出这两种脱销的药品。

（2）现在该医院想用这 7 种草药配制三种新的特效药，表 2 给出了三种新的特效药的成分，请问能否配制？如何配制？

表 1 7 种特效药的成分

中药	1 号成药	2 号成药	3 号成药	4 号成药	5 号成药	6 号成药	7 号成药
A	10	2	14	12	20	38	100
B	12	0	12	25	35	60	55
C	5	3	11	0	5	14	0
D	7	9	25	5	15	47	35
E	0	1	2	25	5	33	6
F	25	5	35	5	35	55	50
G	9	4	17	25	2	39	25
H	6	5	16	10	10	35	10
I	8	2	12	0	2	6	20

表 2 3 种新特效药的成分

中药	1 号新药	2 号新药	3 号新药
A	40	162	88
B	62	141	67
C	14	27	8
D	44	102	51
E	53	60	7
F	50	155	80
G	71	118	38
H	41	68	21
I	14	52	30

3 实验原理

1、线性相关和线性无关

对向量组 $\alpha_1,\alpha_2,\cdots,\alpha_m$ ，如果存在一组不全为零的数 $k_1,k_2,\cdots,k_m$，使得 $k_1\alpha_1+k_2\alpha_2+\cdots+k_m\alpha_m=0$，那么，称向量组 $\alpha_1,\alpha_2,\cdots,\alpha_m$ 线性相关。如果这样的 m 个数不存在，即上述向量等式仅当 $k_1=k_2=\cdots=k_m=0$ 时才能成立，就称向量组 $\alpha_1,\alpha_2,\cdots,\alpha_m$ 线性无关。

含零向量的向量组 $\mathbf{0},\alpha_1,\alpha_2,\cdots,\alpha_m$ 一定线性相关，因为 $1\mathbf{0}+0\alpha_1+0\alpha_2+\cdots+0\alpha_m$，其中，$1,0,0,\cdots,0$ 不全为零.

只有一个向量 α 组成的向量组线性无关的充分必要条件是 $\alpha\neq 0$，线性相关的充分必要条件是 $\alpha=0$.

考虑齐次线性方程组

$$\begin{cases} a_{11}x_1+a_{12}x_2+\cdots+a_{1m}x_m=0 \\ a_{21}x_1+a_{22}x_2+\cdots+a_{2m}x_m=0 \\ \vdots \\ a_{n1}x_1+a_{n2}x_2+\cdots+a_{nm}x_m=0 \end{cases} \qquad (*)$$

它可以写成

$$AX=0,$$

或

$$x_1\alpha_1+x_2\alpha_2+\cdots+x_m\alpha_m=0,$$

其中

$$A=(\alpha_1,\alpha_2,\cdots,\alpha_m),\alpha_j=\begin{pmatrix} a_{1j} \\ a_{2j} \\ \vdots \\ a_{nj} \end{pmatrix},j=1,2,\cdots,m,\mathbf{x}=\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{pmatrix}.$$

由此可见，向量组 $\alpha_1,\alpha_2,\cdots,\alpha_m$ 线性相关的充分必要条件是齐次线性方程组 (*)

有非零解. 也就是说, 向量组 $\alpha_1,\alpha_2,\cdots,\alpha_m$ 线性无关的充分必要条件是齐次线性方程组 (*) 只有零解.

2、最大线性无关组

最大线性无关组：在 $\alpha_1,\alpha_2,\cdots,\alpha_s$ 中，存在 $\alpha_{i1},\alpha_{i2},\cdots,\alpha_{ip}$ 满足：

（1）$\alpha_{i1},\alpha_{i2},\cdots,\alpha_{ip}$ 线性无关；

（2）在 $\alpha_{i1},\alpha_{i2},\cdots,\alpha_{ip}$ 中再添加一个向量就线性相关。

则称 $\alpha_{i1},\alpha_{i2},\cdots,\alpha_{ip}$ 是 $\alpha_1,\alpha_2,\cdots,\alpha_s$ 的一个最大线性无关组，

注：

Ⅰ、不难看出条件（2）等价的说法还有 $\alpha_1,\alpha_2,\cdots,\alpha_s$ 中任一向量均可由 $\alpha_{i1},\alpha_{i2},\cdots,\alpha_{ip}$ 线性表示；或者亦可以说成 $\alpha_1,\alpha_2,\cdots,\alpha_s$ 中任意 $p+1$ 个向量均线性相关；

Ⅱ、从最大线性无关组的定义可以看出最大线性无关组与原先的向量组可以相互线性表示，进而最大线性无关组与原先的向量组是等价的（即有效的最少的方程构成的方程组与原先的方程组是等价的）；

Ⅲ、从上面的方程组可以看出同解的有效方程组可以是第 1、2 两个方程构成，也可以是第 2、3 两个方程构成（因为第 1 个方程可以看成第 2、3 两个方程的和），因此从其对应的向量组来说，向量组的最大线性无关组是不唯一的；

Ⅳ、可以发现，虽然同解的有效方程组的形式可以不一样，但是同解的有效方程组中所含的方程的个数是唯一的，即从其对应的向量组来说，最大线性无关组虽然不唯一，但是最大线性无关组中所含向量的个数唯一的。

3、rref 命令

matlab 将矩阵化成行最简形的命令是 rref 或 rrefmovie。

函数 rref 或 rrefmovie

格式

R = rref(A)　　%用高斯—约当消元法和行主元法求 A 的行最简行矩阵 R

[R,jb] = rref(A)

%jb 是一个向量，其含义为：r = length(jb)为 A 的秩；A(:, jb)为 A 的列向量基；

jb 中元素表示基向量所在的列。

[R,jb] = rref(A,tol)　　%tol 为指定的精度

rrefmovie(A)　　%给出每一步化简的过程

4、线性方程组的求解命令

AX=B 或 XA=B

在 MATLAB 中，求解线性方程组时，主要采用前面章节介绍的除法运算符“/”和“\”。如：

X=A\B 表示求矩阵方程 AX＝B 的解；

X＝B/A 表示矩阵方程 XA=B 的解。

对方程组 X＝A\B，要求 A 和 B 用相同的行数，X 和 B 有相同的列数，它的行数等于矩阵 A 的列数，方程 X＝B/A 同理。

如果矩阵 A 不是方阵，其维数是 m×n，则有：

m＝n 恰定方程，求解精确解；

m>n 超定方程，寻求最小二乘解；

m<n 不定方程，寻求基本解，其中至多有 m 个非零元素。

针对不同的情况，MATLAB 将采用不同的算法来求解。

恰定方程组

恰定方程组由 n 个未知数的 n 个方程构成，方程有唯一的一组解，其一般形式可用矩阵，向量写成如下形式：

Ax=b 其中 A 是方阵，b 是一个列向量；

在线性代数教科书中，最常用的方程组解法有：

（1）利用 cramer 公式来求解法；

（2）利用矩阵求逆解法，即 x=A-1b；

（3）利用 gaussian 消去法；

（4）利用 lu 法求解。

一般来说，对维数不高，条件数不大的矩阵，上面四种解法所得的结果差别不大。前三种解法的真正意义是在其理论上，而不是实际的数值计算。MATLAB 中，出于对算法稳定性的考虑，行列式及逆的计算大都在 lu 分解的基础上进行。

在 MATLAB 中，求解这类方程组的命令十分简单，直接采用表达式：x=A\b。

在 MATLAB 的指令解释器在确认变量 A 非奇异后，就对它进行 lu 分解，并最终给出解 x；若矩阵 A 的条件数很大，MATLAB 会提醒用户注意所得解的可靠性。

如果矩阵 A 是奇异的，则 Ax=b 的解不存在，或者存在但不唯一；如果矩阵 A 接近奇异时，MATLAB 将给出警告信息；如果发现 A 是奇异的，则计算结果为 inf，并且给出警告信息；如果矩阵 A 是病态矩阵，也会给出警告信息。

注意：在求解方程时，尽量不要用 inv(A)*b 命令，而应采用 A\b 的解法。因为后者的计算速度比前者快、精度高，尤其当矩阵 A 的维数比较大时。另外，除法命令的适用行较强，对于非方阵 A，也能给出最小二乘解。

超定方程组

对于方程组 Ax=b，A 为 n×m 矩阵，如果 A 列满秩，且 n>m。则方程组没有精确解，此时称方程组为超定方程组。线性超定方程组经常遇到的问题是数据的曲线拟合。对于超定方程，在 MATLAB 中，利用左除命令（x=A\b）来寻求它的最小二乘解；还可以用广义逆来求，即 x=pinv(A),所得的解不一定满足 Ax=b，x 只是最小二乘意义上的解。左除的方法是建立在奇异值分解基础之上，由此获得的解最可靠；广义逆法是建立在对原超定方程直接进行 householder 变换的基础上，其算法可靠性稍逊与奇异值求解，但速度较快。

4 算法与编程

4.1 编程描述

第一步：将 7 种特效药的成分保存在 7 种特效药成分.txt 文件当中，每列数据代表 1 种药的配方；将 3 种新药的成分保存在 3 种新药成分.txt 文件当中。

第二步：创建一个 FileOpenYao.m 函数文件用于打开第一步所创建的两个 txt 文件，读取数据并将获得的矩阵返回给两个变量。

第三步：创建一个主 M 文件：shiyan7.m，在该 M 文件中调用函数 FileOpenYao.m 函数获取 7 种特效药的成分、3 种新药的成分。

第四步：利用线性无关求解第一问。

第五步：利用线性无关求解第二问。

实验代码和数据文件存放如下图所示。

图 1 实验代码和实验数据存放

4.2 实现代码

1、主 M 文件：shiyan7.m

```
%%
%shiyan7.m        数学实验 7 文件
%文件说明：
%某中药厂用 9 种中草药（A-I），根据不同的比例配制成了 7 种特效药，各用量成分见表 TYAO 矩阵（单位：克）
%试解答：
%（1）某医院要购买这 7 种特效药，但药厂的第 3 号药和第 6 号药已经卖完，请问能否用其他特效药配制出这两种脱销的药品。
%（2）现在该医院想用这 7 种草药配制三种新的特效药，NYAO 矩阵给出了三种新的特效药的成分，请问能否配制？如何配制？

%    Chen Yirong  修改于 2017-06-07
%    代码编辑 matlab 版本：MATLAB R2014a

%%
%****准备工作****%
```

```
clear all   %清空工作空间
clc           %清空命令行窗口
close all   %关闭其他 figure 窗口
%%
%****读取数据文件****%
[ TYAO, NYAO ] = FileOpenYao();   %打开 7 种特效药成分.txt、3 种新药成分.txt 这两个文件获取数据
%    TYAO        7 种特效药成分比例，每一列为一种药的药方，行分别代表 A 到 I 这 9 种中草药
%    NYAO        3 种新药成分比例，每一列为一种药的药方，行分别代表 A 到 I 这 9 种中草药

%%
%第 1 问求解
A=TYAO;
B=NYAO;

[A0, jd] = rref(A)    %A 的行最简形和一组最大线性无关组
R1 = length(jd)          %A 的秩

A1=[A(:,1) A(:,2) A(:,4) A(:,5) A(:,7)];
fprintf('第 1 问求解结果:');
fprintf('3 号药的配制比例:');
xishu1=A1\A(:,3)        %求 3 号药由 1、2、4、5、7 号药线性表示配制的比例系数
fprintf('6 号药的配制比例:');
xishu2=A1\A(:,6)        %求 6 号药由 1、2、4、5、7 号药线性表示配制的比例系数

%%
```

```
%第 2 问求解
% 找出矩阵 A 的所有最大线性无关组
num = 0;
[m,n]= size(A);
p = (combntns([1:1:n],R1))';
qq = [];
for k=1: nchoosek(n,R1)
    q = A(:, p(:,k))';
    if rank(q) == R1
        num = num+1;
        qq = [qq; p(:,k)'];
    end
end
fprintf('第 2 问求解结果:');
fprintf('所有的最大无关组:');
qq                  % 所有的最大无关组：每行为一最大无关对应的序号
fprintf('最大无关组的个数:');
num                 % 最大无关组的个数

C1=[A(:,1) A(:,2) A(:,4) A(:,5) A(:,6) A(:,7)];    %矩阵 A 的三个最大线性无关组
C2=[A(:,1) A(:,3) A(:,4) A(:,5) A(:,6) A(:,7)];    %从之前输出的矩阵 qq 可以得知
C3=[A(:,2) A(:,3) A(:,4) A(:,5) A(:,6) A(:,7)];
fprintf('1 号新药的配制:');
peizhi11=C1\B(:,1)
peizhi12=C2\B(:,1)
peizhi13=C3\B(:,1)
fprintf('2 号新药的配制:');
peizhi21=C1\B(:,2)
peizhi22=C2\B(:,2)
peizhi23=C3\B(:,2)
```

```
fprintf('3 号新药的配制:');
peizhi31=C1\B(:,3)
peizhi32=C2\B(:,3)
peizhi33=C3\B(:,3)

%%
%end
%15 电科卓越班
%陈艺荣
```

2、打开数据 M 文件：FileOpenYao.m

```
function [ TYAO, NYAO ] = FileOpenYao()
%FileOpenYao.m  该函数打开 7 种特效药成分.txt、3 种新药成分.txt 这两个文件
%     TYAO          7 种特效药成分比例，每一列为一种药的药方，行分别代表 A 到 I 这 9 种中草药
%     NYAO          3 种新药成分比例，每一列为一种药的药方，行分别代表 A 到 I 这 9 种中草药

%    Chen Yirong  修改于 2017-06-07
%    代码编辑 matlab 版本：MATLAB R2014a

%打开 7 种特效药的成分数据
TXT=fopen('7 种特效药成分.txt');   %打开数据文件
A=textscan(TXT,'%f %f %f %f %f %f %f');          %把每一列的数据读入到读入到单元数组 A 中
TYAO=[A{1} A{2} A{3} A{4} A{5} A{6} A{7} ];   %矩阵赋值
%打开 3 种新药的成分数据
TXT=fopen('3 种新药成分.txt');   %打开数据文件
A=textscan(TXT,'%f %f %f');          %把每一列的数据读入到读入到单元数组 A
```

```
中
NYAO=[A{1} A{2} A{3}];   %矩阵赋值
end
```

5 实验结果

在 matlab 中运行 shiyan7.m 文件，在命令行窗口输出如下：

```
A0 =

     1     0     1     0     0     0     0
     0     1     2     0     0     0     0
     0     0     0     1     0     0     0
     0     0     0     0     1     0     0
     0     0     0     0     0     1     0
     0     0     0     0     0     0     1
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0

jd =

     1     2     4     5     6     7

R1 =

     6
```

第 1 问求解结果:3 号药的配制比例:

xishu1 =

1.0000
2.0000
-0.0000
0.0000
-0.0000

6 号药的配制比例:

xishu2 =

-0.0690
3.0192
1.0025
1.0403
-0.0044

第 2 问求解结果:所有的最大无关组:

qq =

1	2	4	5	6	7
1	3	4	5	6	7
2	3	4	5	6	7

最大无关组的个数:

num =

3

1 号新药的配制:

peizhi11 =

1.0000
3.0000
2.0000
0.0000
-0.0000
0.0000

peizhi12 =

-0.5000
1.5000
2.0000
-0.0000
0.0000
0.0000

peizhi13 =

1.0000
1.0000
2.0000
-0.0000
0.0000
0.0000

2 号新药的配制:

peizhi21 =

3.0000
4.0000
2.0000
0.0000
-0.0000
1.0000

peizhi22 =

1.0000
2.0000
2.0000
0.0000
-0.0000
1.0000

peizhi23 =

-2.0000
3.0000
2.0000
0.0000
-0.0000
1.0000

3 号新药的配制:

peizhi31 =

1.1322
7.4379
2.1718
2.3827
-2.0645
0.6844

peizhi32 =

-2.5867
3.7189
2.1718
2.3827
-2.0645
0.6844

peizhi33 =

5.1734
1.1322
2.1718
2.3827
-2.0645
0.6844

数据分析：

对于第 1 问，从实验结果中 xishu1 和 xishu2 的输出可以看出，3 号药可以使用其他特效药配制，每一份 3 号药可以用一份 1 号药和一份 2 号药制得；由于求得的 xishu2 中存在负数，所以 6 号药无法用其他特效药配制。

对于第 2 问，最大无关组的个数 num =3，所以可以采用三种方案配制新药。

对于 1 号新药，分析 peizhi11、peizhi12、peizhi13，可知有 2 种不同的配制方案，1 份 1 号新药可由 1 份 1 号特效药、3 份 2 号特效药、2 份 4 号特效药制得；或者由 1 份 2 号特效药、1 份 3 号特效药、2 份 4 号特效药制得。

对于 2 号新药，分析 peizhi21、peizhi22、peizhi23，可知有 2 种不同的配制方案，1 份 2 号新药可由 3 份 1 号特效药、4 份 2 号特效药、2 份 3 号特效药制得；或者由 1 份 1 号特效药、2 份 2 号特效药、2 份 3 号特效药制得。

对于 3 号新药，分析 peizhi31、peizhi32、peizhi33，可知没有可行的配制方案。

6 实验总结和实验感悟

线性相关和无关为研究产品的相似性和不可替代性以及开发新的品种提供了理论基础。这次实验相当于对整个数学实验进行阶段性总结，所以总体感觉比较简单，同时通过这次实验，我简单地对大一所学的线性代数的知识进行了简单的复习。

参考文献

[1] 叶洪伟.人脸识别的特征描述方法研究[D].重庆大学,2013.

[2] 董建华.人脸识别算法的研究及实现[D].山东大学,2015.

[3] 朱怀毅.人脸检测与识别的多特征分析与系统实现[D].上海交通大学,2008.

[4] 程正东等.基于子空间的人脸识别(第一版) [D]. 北京:清华大学出版社, 2009.

[5] 孙继文.人脸识别算法的研究[D].沈阳工业大学,2014.

[6] 沈锐.基于 PCA 的整体与局部特征融合的人脸识别方法[D].武汉理工大学,2008.

[7] Linear regression for face recognition. I Naseem,R Togneri,M Bennamoun. IEEE Transactions on Pattern Analysis and Machine Intelligence . 2010

[8] 张哲来.基于协同表示和线性回归的人脸识别算法研究[D].苏州大学,2015.

[9] A Pyramidal Neural Network For Visual Pattern Recognition. Son Lam Phung,Abdesselam Bouzerdoum. IEEE TRANSACTIONSON NEURAL NETWORKS . 2007

[10] 叶洪伟.人脸识别的特征描述方法研究[D].重庆大学,2013.

[11] 董建华.人脸识别算法的研究及实现[D].山东大学,2015.

[12] 朱怀毅.人脸检测与识别的多特征分析与系统实现[D].上海交通大学,2008.

[13] 程正东等.基于子空间的人脸识别(第一版) [D]. 北京:清华大学出版社, 2009.

[14] 孙继文.人脸识别算法的研究[D].沈阳工业大学,2014.

[15] 沈锐.基于 PCA 的整体与局部特征融合的人脸识别方法[D].武汉理工大学,2008.

[16] Linear regression for face recognition. I Naseem,R Togneri,M Bennamoun. IEEE Transactions on Pattern Analysis and Machine Intelligence . 2010

[17] 张哲来.基于协同表示和线性回归的人脸识别算法研究[D].苏州大学,2015.

[18] A Pyramidal Neural Network For Visual Pattern Recognition. Son Lam Phung,Abdesselam Bouzerdoum. IEEE TRANSACTIONSON NEURAL NETWORKS . 2007