**Matlab第二次作业**

1. 编写一个M文件，首先生成一个具有20行的杨辉三角，保存在名为yanghui.txt的文本文件中，要求三角形左右对称；然后从这个文本文件中读取其第10行，将其中的数字保存在名为line\_10的向量中。，

**解：M文件的源代码为：**

**clear all**

**clc**

**fid=fopen('yanghui.txt','w');**

**A=zeros(20,39);**

**A(1,20)=1;**

**A(20,1)=1;**

**A(20,39)=1;**

**for i=2:20;**

**for j=2:38;**

**if i+j==21|j-i==19;**

**A(i,j)=1;**

**else A(i,j)=A(i-1,j-1)+A(i-1,j+1);**

**end**

**end**

**end**

**for i=1:20;**

**for j=1:39**

**if A(i,j)==0&j~=39;**

**fprintf(fid,' ');**

**elseif A(i,j)~=0;**

**fprintf(fid,'%5d',A(i,j));**

**else**

**fprintf(fid,' \n');**

**end**

**end**

**end**

**fclose(fid);**

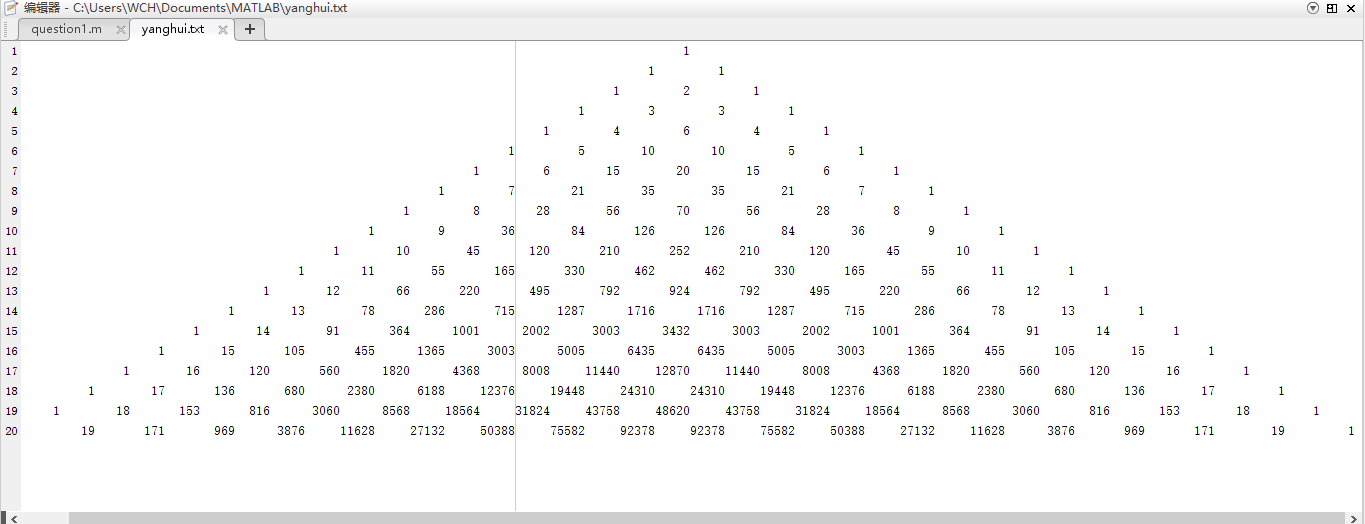
**fid=fopen('yanghui.txt');**

**C=fscanf(fid,'%d',[1,inf]);**

**line\_10=C(46:55);**

**fclose(fid);**

**对称的杨辉三角形截图为：**

****

1. 编写M文件求解一元二次方程：，要求对参数的取值情况加以讨论。

**解：源代码为：**

**clear all**

**clc**

**a=input('a=');**

**b=input('b=');**

**c=input('c=');**

**d=b^2-4\*a\*c;**

**if d<0**

**disp('无解')**

**elseif d==0**

**x=-2\*a/b;**

**a=['x=',num2str(x)];**

**disp(a)**

**else**

**x=[(-b+sqrt(d))/(2\*a),(-b-sqrt(d))/(2\*a)];**

**b=['x1=',num2str(x(1)),' x2=',num2str(x(2))];**

**disp(b)**

**end**

1. 编写M文件实现阶乘函数factor()，要求用至少两种方式实现，比较递归调用和其它方法的程序执行效率。

**解：第一种方式（不使用递归）的源代码为：**

**function [ y ] = Factor( x )**

**y=1;**

**for n=x:-1:1**

**y=y\*n;**

**end**

**end**

**第二种使用递归的源代码为：**

**function [ y ] = Factor1( x )**

**if x<=1**

**y=1;**

**else**

**y=Factor1(x-1)\*x;**

**end**

**end**

**对比两者的计算执行效率：**

**clear all**

**clc**

**x=input('x=');**

**tic**

**Factor(x);**

**time1=toc**

**tic**

**Factor1(x);**

**time2=toc**

**对比结果为：**

**time1 =0.0020**

**time2 =0.0618**

**所以非递归算法执行效率更高**

1. 绘制饱和非线性函数 的曲线，并用字体“Times New Roman”标注横坐标*x*、纵坐标*y*，和图的标题“Saturated nonlinear function”。

**解：M文件源代码为：**

**clear all**

**clc**

**x=linspace(-5,5);**

**y=1.1.\*sign(x).\*(abs(x)>1.1)+x.\*(abs(x)<=1.1);**

**plot(x,y);**

**xlabel('x','fontname','Times New Roman');**

**ylabel('y','fontname','Times New Roman');**

**title('Saturated nonlinear function','fontname','Times New Roman');**

1. 编写M文件实现两个任意阶数的矩阵的克罗内克积。把这个函数的句柄作为参数传递给另一个函数，该函数求解矩阵所有元素之和，克罗内克积的定义如下：

If A is an m × n matrix and B is a p × q matrix, then the Kronecker product  is the mp × nq block matrix:

**解：克罗内克积函数为：**

**function [Y] = KLNK( A,B )**

**[m,n]=size(A);**

**[p,q]=size(B);**

**for x=1:m**

**for y=1:n**

**Y((x-1)\*p+1:x\*p,(y-1)\*q+1:y\*q)=A(x,y).\*B;**

**End**

**end**

**end**

**求元素之和的函数为：  
function [ x ] = Sum( A,B)**

**f=@(A,B) KLNK(A,B);**

**Y=f(A,B)-f(B,A);**

**[m,n]=size(Y);**

**y(1)=0;**

**for a=2:m\*n+1**

**y(a)=y(a-1)+Y(a-1);**

**end**

**x=y(m\*n);**

**end**

1. 由矩阵理论可知，如果一个矩阵可以写成, 则矩阵的逆矩阵可以由下面算法给出：



试根据该算法编写函数对矩阵求逆，并编写测试程序，和直接求逆方法进行求解精度上的比较。

**解：源代码为：**

**clear all**

**clc**

**A=input('请输入矩阵A:');**

**B=input('请输入矩阵B:');**

**C=input('请输入矩阵C:');**

**M=A+B\*C\*B.';**

**M1=inv(M);**

**M2=inv(A)-inv(A)\*B\*(inv(C)+B.'\*inv(A)\*B)\*B.'\*inv(A);**

**error=norm(M2-M1)**