Q1. 由指令 `rng('default'),A=rand(3,5)`生成二维数组 **A**，试求该数组中所有大于 **0.5** 的元素的位置，分别求出它们的“全下标”和“单下标”。

答：

```
rng('default')
A=rand(3,5)
A =
    0.8147    0.9134    0.2785    0.9649    0.9572
    0.9058    0.6324    0.5469    0.1576    0.4854
    0.1270    0.0975    0.9575    0.9706    0.8003

[ri,cj]=find(A>0.5); %获取下标需要用 find 函数，若一个返回值则直接获取全下标
id=sub2ind(size(A),ri,cj);
ri=ri';cj=cj';
disp('  ')
disp('大于 0.5 的元素的全下标')
disp(['行号  ',int2str(ri)])
disp(['列号  ',int2str(cj)])
disp('  ')
大于 0.5 的元素的全下标
行号  1  2  1  2  2  3  1  3  1  3
列号  1  1  2  2  3  3  4  4  5  5

disp('大于 0.5 的元素的单下标')
disp(id')

大于 0.5 的元素的单下标
     1     2     4     5     8     9    10    12    13    15
```

Q2.先运行指令 `x=-3*pi:pi/15:3*pi; y=x; [X,Y]=meshgrid(x,y); warning off; Z=sin(X).*sin(Y)./X./Y;` 产生矩阵 Z。(1) 请问矩阵 Z 中有多少个“非数”数据？(2) 用指令 `surf(X,Y,Z); shading interp` 观察所绘的图形。(3) 请写出绘制相应的“无裂缝”图形的全部指令。(提示：isnan 用于判断是否非数；可借助 sum 求和；realmin 是最小正数。)

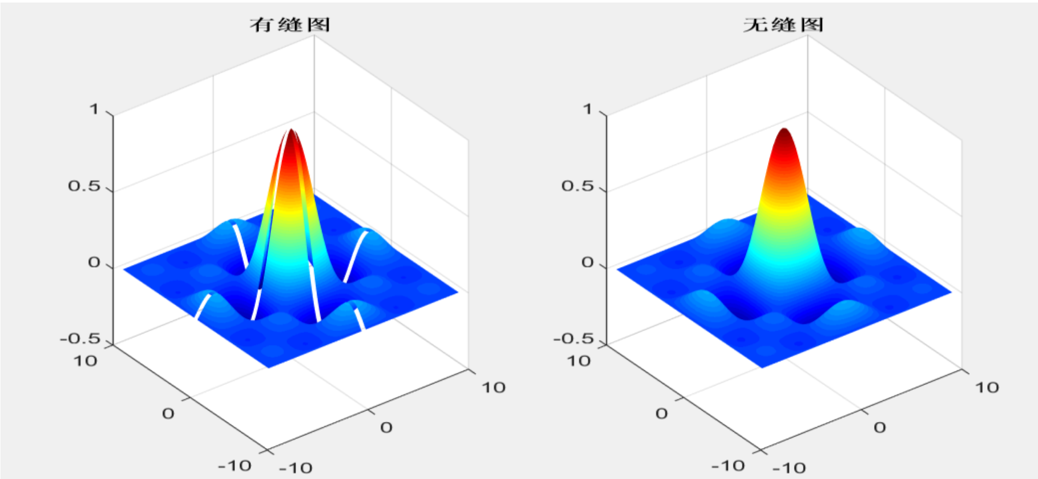
答：(1)

```
x=-3*pi:pi/15:3*pi;
y=x;
[X,Y]=meshgrid(x,y);
warning off
Z=sin(X).*sin(Y)./X./Y;
NumOfNaN=sum(sum(isnan(Z)))      %计算“非数”数目
NumOfNaN =
   181
```

(2，3，摘自配套标准答案)

```
subplot(1,2,1),surf(X,Y,Z),shading interp,title('有缝图')
%产生无缝图
```

```
XX=X+(X==0)*eps;            %将 X=0 的点进行 X 值的一个轻微扰动
YY=Y+(Y==0)*eps;            %对 Y=0 的点同理操作
ZZ=sin(XX).*sin(YY)./XX./YY;%扰动后可以避免除 0 操作，得到的值近似等于极限值
%注：本题亦可以直接人工计算 NaN 点的极限值进行人工代入
subplot(1,2,2),surf(XX,YY,ZZ),shading interp,title('无缝图')
colormap jet
```



Q3.线性方程组问题可以通过高等代数中所学习的初等变换法解决，也可以通过（按列选主元的）高斯消去法解决。容易得知，这些算法复杂度均为$O(n^3)$，3 层循环的代码效率很低。

设方阵 A 可逆，利用 MATLAB 的数组化运算，尝试以尽量少的循环数完成初等变换法或(选主元的)高斯消去法。然后用下列代码生成随机矩阵 A 与随机向量 b，解出 Ax=b 的向量解(无需粘贴解),并与 MATLAB 函数 `c=A\b` 得到的解进行误差分析(函数 `norm(b-b2)` 可以计算向量 b 与 b2 之间的 2-范数差距)

```
rng default, A=rand(1200);b=rand(1200,1);
```

答：(仅提供单层 for 循环通过行初等变换化为最简形（若 A 可逆行最简形即对角阵，比原始的上三角形高斯消去法更为简便），并按列选主元的求解的代码)

```
clear
close all
rng default, A=rand(1200);b=rand(1200,1);

Z = [A,b];  %定义增广矩阵

[m,n] = size(Z);
for i = 1:m
      %以下代码段为鲁棒性代码段（列主元方法），避免Z(i,i)等于0或绝对值太小，导致程序出错（得到Inf）或误差过大，使用此代码段可以将相对误差缩小到10^-13的数量级。注意到A可逆，所以不可能所有列元都为0
    [~,maxi] = max(abs(Z(i:m,i)));
```

```
    temp = Z(i,:);
    Z(i,:) = Z(i-1+maxi,:);
    Z(i-1+maxi,:)=temp;
    ci = Z(:,i)/Z(i,i); %./效果相同, 求出了各行的加减系数
    ci(i) = 0; %第i行本身不变化
    Z = Z - ci*Z(i,:);   %其余行分别减去对应系数ci乘第i行
end
sol = Z(:,m+1)./diag(Z);          %解出对角矩阵方程组对应的解
c = A\b;                          %理论真实解
err = norm(sol-c)/norm(c)          %计算相对误差
err =

   2.5389e-13
```

结果表明，该算法对于可逆方阵构成线性方程组是可行的。

感兴趣的同学们再尝试思考 A 为非方阵、或不可逆时，应该如何设计这个算法。

Q4.（开放性问题，无法实现可写思路）数据降维问题是现今非常流行的数据科学问题。现假设有 1000 个 500 维的列向量，组合成矩阵$F_{500\times1000}$存储于文件“W5Q4.mat”中（将文件复制到当前文件夹，再在窗口内双击,或用 MATLAB 命令 `load W5Q4` 即可打开）。其中这 1000 个向量有随机的 500 个属于一个三维欧氏子空间X_1，其余 500 个向量属于另外一个三维欧氏子空间X_2，但两个空间具体的向量分布未知。

(1)容易证明，这 1000 个向量位于一个 6 维的子空间中，请设计一种算法，将这 1000 个 500 维向量转化为 6 维的“降维表示”。降维后的结果向量保存在矩阵$G_{6\times1000}$中。(提示 :PCA 与函数 `eigs`)
(2)设计一种算法，识别出哪 500 个向量属于X_1，哪 500 个向量属于X_2（分两组），分离两组 index 后，用 `rank` 函数检验你的结果。

答: (1)这一问的代码并不复杂，主要难点在于理解题意。使用特征值分解对数据进行降维是一种很常见的思路。一种解决的思路是使用 PCA（主成分分析），不过进行主成分分析时，并不需要将向量“中心化”(因为原向量已落在 6 维空间内)。对协方差矩阵进行特征值分解，即$Cov = FF^\top = VDV^\top$时，若D的 6 个非零特征值位于左上角。此时设V的对应 6 个特征向量组成子矩阵$W_{500\times6}$，此时$G = FW^\top$就可以作为降维的结果。

```
COV = F*F';                        %协方差矩阵
[V,D] = eigs(COV,500,'lm');
%lm表示特征值按照从大到小排序，V为对应的单位化的特征向量
```

```
G = V(:,1:6)'*F;            %此时，G是一个6X1000的矩阵
Grammian_Error = norm(F'*F-G'*G,'fro')
```
%利用Gram矩阵的误差$F^{\top}F - G^{\top}G$来检测降维准确性

```
Grammian_Error =
   2.8517e-13
```
结果可见，降维是基本准确的。

(2) 如果第一问降维结果准确，可以直接利用降维的向量G来完成这一问。否则，也可以直接对原始的F进行操作，只是效率会略低一点。
这一问是开放式问题，嘉哥提供的一种粗暴思路采用的是抽屉原则，即考虑前 7 个向量，必然有至少 4 个向量属于X_1，或至少 4 个向量属于X_2。这 4 个落在同一个子空间的向量$[\vec{v_1},\vec{v_2},\vec{v_3},\vec{v_4}]_{6\times 4}$的秩为 3。因此，在前 7 个向量寻找 4 个秩为 3 的向量组，即可找到其中一个由这些向量子空间，不妨设为$X_1 \supset [\vec{v_1},\vec{v_2},\vec{v_3},\vec{v_4}]$。
对于任意的其它向量$\vec{w}$，若$\vec{w} \in X_1$，则$[\vec{v_1},\vec{v_2},\vec{v_3},\vec{w}]$秩为 3，否则$[\vec{v_1},\vec{v_2},\vec{v_3},\vec{w}]$秩为 4。
利用这个思路，代码设计如下：

```
for i = combnk(1:7,4)'
%combnk(1:7,4)含义为在1~7任选四个的所有组合，每一种组合为一个行向量，共35行
%转置后作为for循环指标，含义为共35次循环，每次使i等于一种组合的列向量（转置）
    TEMP = G(:,i);              %取出G的对应四列，并检测其秩
    if(rank(TEMP)==3)
        IDX = i;                %秩为3即表示找到了一个子空间,记录前三个
        break;
    end
end
SUBSP = G(:,IDX(1:3));          %取出子空间的前三个向量v1,v2,v3用以判别
for i = 1:1000
    TEMP = [SUBSP,F2(:,i)];     %组成临时矩阵[v1,v2,v3,w]
    if(rank(TEMP)==3 && ~ismember(i,IDX))     %IDX的元素不重复加入
        IDX = [IDX;i];          %将IDX添加进新的向量w
    end
end
%循环结束后，IDX表示X1的所有向量的index
RANK1 = rank(F(:,IDX))          %检测X1的秩
RANK1 =
     3
RANK2 = rank(F(:,setdiff(1:500,IDX))) %检测X2的秩
RANK2 =
     3
```
结果表明，向量分组完全准确，两组向量构成矩阵的秩都是 3。