第 10 讲　线性方程组和代数方程数值求解

（第 8 章　MATLAB 方程数值求解）

目的：

一、掌握矩阵的分解与线性方程组的解

二、代数方程数值求解的方法。

一、掌握线性方程组和代数方程数值求解的方法。

（一）矩阵的分解

1、化简矩阵的计算工作量

当方程组 $AX=b$ 的系数矩阵为方阵且可逆时，方程组有唯一解。

$X=A^{-1}b$，求 A^{-1} 或者 $A^{-1}b$ 的方法是将 $(A,E)\overset{r}{\sim}(E,A^{-1})$ 或者 $(A,b)\overset{r}{\sim}(E,A^{-1}b)$

$$A=\begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ & \cdots & \cdots & \cdots & \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} \end{pmatrix}$$

这两个方法都需要经过将矩阵 A 化为单位矩阵 E 这一过程，而将矩阵 A 化为单位矩阵 E 通常采用高斯消元法。假设 $a_{11}\neq 0$ 利用 a_{11} 将红框内的元素化为 0，在计算化简的过程中，蓝框内的元素也要一起同步计算。重复这一过程，可以将 A 从上往下化简为上三角矩阵：

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ 0 & b_{22} & b_{23} & \cdots & b_{2n} \\ 0 & 0 & b_{33} & \cdots & b_{3n} \\ & \cdots & \cdots & \cdots & \\ 0 & 0 & 0 & \cdots & b_{nn} \end{pmatrix}, \quad (1)$$

可以继续将这个矩阵从下往上化简为对角矩阵，在往上化简时，比如用 b_{nn} 化简红框元素，由于主对角线下元素是零，此时蓝框内的元素不会一起同步计算。

$$\begin{pmatrix} a_{11} & 0 & 0 & \cdots & 0 \\ 0 & b_{22} & 0 & \cdots & 0 \\ 0 & 0 & b_{33} & \cdots & 0 \\ & \cdots & \cdots & \cdots & \\ 0 & 0 & 0 & \cdots & b_{nn} \end{pmatrix}, \quad (2)$$

对比从上往下，和从下往上化简的过程可知，化简一般矩阵为单位矩阵的计算工作量远远

大于将三角矩阵化简为单位矩阵的计算工作量。

2、方阵的 LU 分解与方程组 $A_n X = b$ 的求解

矩阵的 LU 分解是指将矩阵 A 分解为下三角矩阵 L 与上三角矩阵 U 的乘积，即 $A = L \cdot U$，方程组 $AX = b$ 等同于 $LUX = b$，解得 $X = U^{-1}L^{-1}b$，由于 L,U 是三角矩阵，求逆时运算量比较小。所以如果 L,U 已知，用 $X = U^{-1}L^{-1}b$ 求解比用 $X = A^{-1}b$ 更加有效。

但如果 L,U 未知，用 $X = U^{-1}L^{-1}b$ 求解方程组，还需要将 A 分解为 L,U，如果求出的 L,U 只用于求解一个方程组 $AX = b$，显然并不比用 A^{-1} 求解更有效。只有当需要求解一系列的方程组 $AX = b_1, AX = b_2 \cdots$ 时，将 A 分解为 L,U 只需做一次，然后就可以将 L,U 用于每一个方程的求解中，这将大大减少计算量。

Matlab 中调用 **lu(A)**函数将矩阵 A 进行 LU 分解，其格式为：**[L,U]=lu(A)**。

3、矩阵的奇异值分解（SVD 分解）

矩阵 $A_{m\times n}$ 的奇异值分解是指**存在正交矩阵** U,V **和对角矩阵 S，使** $A_{m\times n} = U_m S_{m\times n} V_n^T$。

S 为矩阵 A 的奇异值的根植构成的矩阵（即 AA^T 的特征值的根植构成的矩阵，它同时也是 $A^T A$ 的特征值的根植）。 U 是 AA^T 的特征向量构成的正交矩阵，V 是 $A^T A$ 的特征向量构成的正交矩阵。

矩阵奇异值分解（svd）的应用：

svd 分解主要用于图像压缩、信号处理等方面。

Svd 分解用于图像压缩的原理：

设有一副图像，由于图像其实只是不同颜色的显示，所以只需掌握图像上每个像素点的颜色，就掌握了该图像。设每个像素点都由三种颜色 R，G，B（红绿蓝）混合产生，可以用三个矩阵 R，G，B 用于存放图像上像素点对应的 R，G，B 三色，只要颜色矩阵一定，图像就一定。注意到，同样是红色，数字 150 对应的红色和数字 148 对应的红色，人眼看来差别不大，也就是相近的颜色对应的颜色数字差不多。从而假设红色矩阵 R 是 $m \times n$ 阶的（$m \times n$ 由图像的像素

决定），将矩阵 R 进行 svd 分解设$[U,S,V]=svd(R)$，$S_{m\times n}$是$\left(RR^T\right)_m$对应的特征值$\sqrt{\lambda_i}$（按从大到小排列）做主对角线的矩阵。设$m>n$从而

$$R=U_mSV_n^T=\left(u_1,u_2,\cdots,u_m\right)\begin{pmatrix}\Lambda_n\\O\end{pmatrix}\begin{pmatrix}v_1^T\\v_2^T\\\cdots\\v_n^T\end{pmatrix}$$

$$=\sqrt{\lambda_1}u_1v_1^T+\sqrt{\lambda_2}u_2v_2^T+\cdots+\sqrt{\lambda_n}u_nv_n^T$$

上式把R分解为了系数为$\sqrt{\lambda_i}$的n个矩阵之和。如果上面分解式中有一半的$\sqrt{\lambda_i}$都很小或者为零，我们把这些$\sqrt{\lambda_i}$对应的矩阵全舍掉，此时

$$R\approx\sqrt{\lambda_1}u_1v_1^T+\sqrt{\lambda_2}u_2v_2^T+\cdots+\sqrt{\lambda_{n/2}}u_{n/2}v_{n/2}^T=R_{n/2}$$

用颜色矩阵$R_{n/2}$替换颜色矩阵R显示出来的效果就差不多。注意到，我们只需保留U和V^T矩阵中的前面$n/2$列和行就可以据此算出$R_{n/2}$，显然$U_{n/2}$和$V_{n/2}^T$比R矩阵占用的存储空间要小。用同样的方式处理G,B两矩阵，得到对应的压缩矩阵，于是就起到了压缩整个图片的作用。

Matlab 中调用 **svd 函数**对矩阵 A 进行 svd 分解，其格式为**[U,S,V]=svd(A)**。

练习 1：使用 **I=imread('文件路径\文件名 ')** 读入一个图片的颜色矩阵（该矩阵是一个三维矩阵，每层代表一个颜色），要求：

（1）对该矩阵每层的颜色矩阵进行 svd 分解，

（2）使用分解原理，对颜色矩阵进行压缩处理，

（3）使用 **imshow(I)**函数显示压缩后的颜色矩阵所对应的图片。

```
clear;
I = imread('E:\...\...\...\*.jpg');  %读取文件路径所对应图片，I 是一个三维矩阵
R = I(:,:,1);  % R 是 I 的第一层颜色矩阵，其尺寸就是图片的像素尺寸
G = I(:,:,2);  % G 是 I 的第二层颜色矩阵，其尺寸就是图片的像素尺寸
B = I(:,:,3);  % B 是 I 的第三层颜色矩阵，其尺寸就是图片的像素尺寸
```

```
%现在的颜色矩阵数字是 uint8 图片型，需用 im2double 转为实数型后才能进行 svd 计算。
R1 = im2double(R);
G1 = im2double(G);
B1 = im2double(B);
%进行 svd 运算
[Ur,Sr,Vr]=svd(R1);
[Ug,Sg,Vg]=svd(G1);
[Ub,Sb,Vb]=svd(B1);

n = 200;% 打算取 U，V 的前面 n 个特征向量来作为近似，这个数不能大于维度。
%取 U 前面 n 个特征向量、S 中的前 n 个特征值、V 中的前 n 个特征向量
%进行压缩，公式是 A=USV'.
R2=Ur(:,1:n)*Sr(1:n,1:n)*Vr(:,1:n).';
G2=Ug(:,1:n)*Sg(1:n,1:n)*Vg(:,1:n).';
B2=Ub(:,1:n)*Sb(1:n,1:n)*Vb(:,1:n).';

I2(:,:,1) = R2; %产生新的图片矩阵 I2
I2(:,:,2) = G2;
I2(:,:,3) = B2;
I2=im2uint8(I2);  %将颜色矩阵由 double 型转回 uint8 图片型
imshow(I2); %显示压缩后的图片
figure(2);
imshow(I);  %显示原图片对比效果。
```

（补充知识点）Svd 分解原理：

设 A 为 $m\times n$ 阶矩阵，则 A^TA 与 AA^T 分别为 n 阶和 m 阶的对称方阵，关于这两矩阵，有如下结论：

（1）由对称矩阵理论，A^TA 与 AA^T 一定可以找到它们各自的特征值矩阵和特征向量构成的正交矩阵 $V=(v_1,v_2,\cdots,v_n)$ 和 $U=(u_1,u_2,\cdots,u_m)$ 使得它们相似对角化。

（2）设 $A^TAv_i=\lambda_i v_i$，由于 $AA^T(Av_i)=A\left(A^TAv_i\right)=A\left(\lambda_i v_i\right)=\lambda_i\left(Av_i\right)$，所以如果 v_i 是 A^TA 对应于 λ_i 的特征向量，则当 $Av_i\neq 0$ 时，Av_i 就是 AA^T 对应于 λ_i 的特征向量。且当 v_i,v_j 为 A^TA 不同的彼此正交的特征向量时，由于

$(Av_i)^T(Av_j)=v_i^TA^TAv_j=v_i^T\lambda_jv_j=\lambda_jv_i^Tv_j=0$，

所以 AA^T 对应的特征向量 Av_i 和 Av_j 也彼此正交。

（3）因为 $U=(u_1,u_2,\cdots,u_m)$ 是 AA^T 的单位正交向量构成的矩阵，所以当 $Av_i\neq0$ 时，

$u_i=\frac{1}{\|Av_i\|}Av_i$，其中 $\|Av_i\|^2=(Av_i)^T(Av_i)=v_i^TA^TAv_i=v_i^T\lambda_iv_i=\lambda_i$，所以首先可以得出 $\lambda_i>0$，所以 AA^T 与 A^TA 的特征值一定不小于 0，即它们至少是半正定的。其次可以得出 $u_i=\frac{1}{\sqrt{\lambda_i}}Av_i$ 即 $Av_i=\sqrt{\lambda_i}u_i$。

（4）从而 $(Av_1,Av_2,\cdots,Av_n)=(\sqrt{\lambda_1}u_1,\sqrt{\lambda_2}u_2,\cdots,\sqrt{\lambda_m}u_m,0\cdots,0)$，其中 $\lambda_1,\cdots,\lambda_m$ 是 A^TA 的非零特征值。此时

即 $A(v_1,v_2,\cdots,v_n)=(u_1,u_2,\cdots,u_m)\begin{pmatrix}S&0\\0&0\end{pmatrix}$，即 $AV=US$，其中

$S=\begin{pmatrix}\sqrt{\lambda_1}&&&\\&\sqrt{\lambda_2}&&\\&&\ddots&\\&&&\sqrt{\lambda_m}\end{pmatrix}$，从而 $A=USV^{-1}=USV^T$。

4、矩阵的 QR 分解

QR 分解同样将矩阵分解为正交矩阵 Q 与上三角矩阵 R 的乘积，$A=QR$，$A^{-1}=R^{-1}Q^{-1}$，故方程组的解为：$X=R^{-1}Q^{-1}b$。Matlab 中调用 qr 函数将矩阵 A 进行 QR 分解，其格式为：$[Q,R]=qr(A)$。

（二）方阵对应的线性方程组的迭代法求解

对方程组 $A_nX=b$ 而言，如果 A_n 尺寸很大，那么不管用上面的哪一种分解法或者直接用高斯消元法求解该方程组，计算工作量都非常大。因此为了迅速找到 $A_nX=b$ 的一个解，可以采用下面的迭代法进行求解。注意：如果 A 的尺寸不大，那么迭代法还不如消元法快。

雅可比(jacobi)迭代法：对 $A_nX=b$

假设 $A=\begin{pmatrix}1&5&3&2\\2&6&1&2\\3&4&2&1\\4&1&2&5\end{pmatrix}$，则 A 可以做如下分解：

令 $D=\begin{pmatrix}1&&&\\&6&&\\&&2&\\&&&5\end{pmatrix}$，$L=\begin{pmatrix}0&0&0&0\\-2&0&0&0\\-3&-4&0&0\\-4&-1&-2&0\end{pmatrix}$，$U=\begin{pmatrix}0&-5&-3&-2\\0&0&-1&-2\\0&0&0&-1\\0&0&0&0\end{pmatrix}$

可以验证：$A=D-L-U$。可以用命令 $D=diag(diag(A))$、$L=-tril(A,-1)$、$U=-triu(A,1)$ 求这三个矩阵。

tril(A)和 triu(A)：求矩阵 A 的下上三角矩阵；

diag(A)：如果 A 是矩阵，则结果是 A 的对角线构成的向量；如果 A 是向量，则生成以这个向量为对角线的矩阵。

此时，方程组 $AX=b$ 可以做下面的变形：

$$AX=b\Rightarrow(D-L-U)X=b\Rightarrow DX=(L+U)X+b$$

$$\Rightarrow X=D^{-1}(L+U)X+D^{-1}b=GX+d\text{ ,即 }X=GX+d$$

其中 $G=D^{-1}(L+U),\ d=D^{-1}b$

将 $X=GX+d$ 左右分别当两个东西看待，左边是算式 X，右边是算式 $GX+d$。我们将某个 X_0 代入右边算式，如果求出的 X_1 等于（或约等于）X_0，则说明 X_0 就是方程组的解

（或近似解）；如求出的 X_1 不等于（或不近似等于） X_0（可以用 X_1-X_0 的长度是否小于容差来判断，即 $\| X_1-X_0 \| < tol$），那么 X_0 就不是方程组的解，此时将 X_1 作为新的 X_0，代入右边算出新的 X_1，不断这样做，直到 X_1 等于（或约等于） X_0 为止。即有递推公式 $X_{n+1}=GX_n+d$。

这种用来求方程组的解的方法，叫做雅可比(jacobi)迭代法。

高斯塞德尔(Gauss-Serel)迭代法：

雅可比迭代法是将方程组移项为：$AX=b \Rightarrow (D-L-U)X=b \Rightarrow DX=(L+U)X+b$

高—塞迭代法是将其移项为：$AX=b \Rightarrow (D-L-U)X=b \Rightarrow (D-L)X=UX+b$

此时 $X=(D-L)^{-1}UX+(D-L)^{-1}b$，令 $G_1=(D-L)^{-1}U$，$d_1=(D-L)^{-1}b$

则 $X=G_1X+d_1$，结构和雅可比迭代法是一样的。

关于迭代法的关键问题是将初始点 X_0 代入后产生的系列点必须要是收敛的才能保证方法可行。即需要 $\lim\limits_{n\to\infty}\| X_{n+1}-X_n \|=0$。迭代法 $X_{n+1}=GX_n+d$ 的收敛条件请自行查阅资料。

练习 2：设计雅可比迭代法的对应函数，求解方程组 $\begin{cases}10x_1-x_2=9\\-x_1+10x_2-2x_3=7\\-2x_2+10x_3=6\end{cases}$，

$X_0=(0,0,0)$。

```
function [y,n]=myjacobi(A,b,x0,tol)  %注意，b，x0是列向量
if nargin==3
    tol=1.0e-6;    % 设置默认容差tol=10^-6
elseif nargin<3
    error('参数数目不够');
end
D=diag(diag(A));
L=-tril(A,-1)  %求主对角线以下的下三角的相反数矩阵
U=-triu(A,1);  %求主对角线以上的上三角的相反数矩阵
```

```
G=D^-1*(L+U);   d=D^-1*b;
y=G*x0+d;
n=1;
while norm(y-x0)>=tol && n<=100     %设置个最大迭代数，防止不收敛时浪费资源
      x0=y;
      y=G*x0+d;
     n=n+1;
     end
end
```

```
>>A=[10,-1, 0;-1,10,-2;0,-2,10], b=[9;7;6], x0=[0;0;0]

>>[x,n]=myjacobi(A,b,x0)

x = 0.9958

    0.9579

    0.7916

n = 11
```

（三）求一般线性方程组的通解

1、齐次方程组 $A_{m\times n}X=0$ 的通解

可以通过调用 **null** 函数求齐次方程组的通解，调用格式为：

x=null(A,'r'): 得到齐次方程的基础解系向量组，自由变量类似取（1，0）、（0，1）等得到的解。

x=null(A): 得到正交单位化的基础解系向量组。

2、非齐次方程组 $A_{m\times n}X=b$ 的通解

方法一：

使用 rref(A,b)将增广矩阵化为行最简型矩阵，然后根据结果手工自己写通解；

方法二：先用 null(A)命令求出对应的 $AX=0$ 的基础解系，再用命令 $X=A\backslash b$ 求出 $AX=b$ 的一个特解，然后组合出通解。

注：由于 A 可能不是方阵，所以上面的 $X=A\backslash b$ **不能写成** $X=inv(A)*b$ **或者** $A^{-1}*b$ **。**

扩展知识——如何求 $A_{m\times n}X=b$ 的特解或近似特解（最小二乘解）。

如果 X 是方程组的特解，则 $AX-b=0$，如果 X 是方程的近似特解，则 $AX-b\approx 0$。可以用求向量的模 $\|AX-b\|$ 来判断一个向量是否为零向量或者近似零向量，如果某个 X 代入求模算式后能使模为 0 或者近似为 0，那么这个 X 就是方程组的特解或近似特解。

我们把使 $f(X)=\|AX-b\|^2=(AX-b)^T(AX-b)$ 取最小值的向量 X 称为方程组 $AX=b$ 的近似特解（即 $\|AX-b\|_{\min}$ 问题）。

$$f(X)=\|AX-b\|^2=(AX-b)^T(AX-b)=(X^TA^T-b^T)(AX-b)$$

$$=X^TA^TAX-X^TA^Tb-b^TAX+b^Tb,$$

注意，这是一个含有 n 个自变量 $X=(x_1,x_2,\cdots,x_n)$ 的多元函数。

其中 X^TA^TAX 是一个二次型多项式，X^TA^Tb 与 b^TAX 是线性一次函数，b^Tb 是常数。

根据多元函数求极值理论，极值点就是使 n 个偏导同时为 0 的点。

所以利用 $\frac{\partial X^TBX}{\partial X}=2BX$ 、$\frac{\partial BX}{\partial X}=B^T$ 、$\frac{\partial X^TB}{\partial X}=B$

对 $f(X)=X^TA^TAX-X^TA^Tb-b^TAX+b^Tb$，令 $\frac{\partial f}{\partial X}=0$

得 $2A^TAX-A^Tb-A^Tb+0=0\Rightarrow A^TAX=A^Tb$，当 A^TA 可逆时，$X=(A^TA)^{-1}A^Tb$ 这就是函数的最小值点，也就是方程组 $A_{m\times n}X=b$ 的近似特解。我们把矩阵 $(A^TA)^{-1}A^T$ 称为矩阵 A 的**伪逆**。

如果 $A_{m\times n}$ 是一个列满秩矩阵，则 A^TA 一定可逆，此时伪逆为 $(A^TA)^{-1}A^T$。

当 $A_{m\times n}$ 是一个行满秩矩阵，此时 AA^T 一定可逆，对方程组 $XA=b$ 进行刚才类似的讨论，可得 $X=bA^T(AA^T)^{-1}$，此时将 $A^T(AA^T)^{-1}$ 称为矩阵 A 的伪逆。

如果 A 既非行满秩又非列满秩，那么将 A 进行 svd 分解，$[U,S,V]=svd(A)$；

则 $A=USV^T$，此时伪逆等于 $V\cdot S1\cdot U^T$， $S1$ 是将 S 转置后对角线上非零值取倒数后的矩阵。

Matlab 中可以使用 **pinv(A)**求矩阵 A 的伪逆，显然如果 A 为可逆方阵，则 pinv(A)=inv(A)。

二、求代数方程的数值解

1、一元函数方程的根

对于一元函数方程 $f(x)=0$，求根的常用方法有二分法、牛顿迭代法、不动点迭代法等。

Matlab 中用于求解一元函数方程根的函数是 **fzero**，调用格式为**[x,fval]=fzero(fun,x0,options),** 其中，fun 是待求解方程的函数句柄，x0 是初始迭代点，options(可缺省)是优化选项，options 需用 optimoptions 或者 optimset 进行指定，比如 options=optimset('属性名称','属性值')。X0 如果是单值标量，那么 fzero 采用牛顿迭代法求根，如果 x0 是两点向量[a,b]，那么 fzero 采用二分法求根，但此时要求输入的 a,b 满足 $f(a)\cdot f(b)<0$。输出[x,fval]中 x 是求出的解，fval 是解对应的函数值，用于验证结果是否近似为 0，可以缺省。

options 的常用设置：options=optimset('Display','off') 不显示计算过程;

option=optimset('Display','iter') 显示迭代过程

op=optimset('Display','final') 仅显示最后结果

abc=optimset('Display','notify') （默认）当函数不收敛时显示提示;

rst=optimset('Tolx', tol) tol 为标量，当误差不大于 tol 时停止迭代。

options 的名字无所谓，只要设置好就可以了。

练习 3：$3x+\sin x-e^x=0$ 在 $x_0=1.5$ 附近的根。

```
function y=fx(x)    %使用m函数文件表示待求解方程，其对应的函数句柄是@fx
y=3*x+sin(x)-exp(x);
end
```

```
>> fplot(@fx,[0,3]); refline(0,0);  % 做[0,3]上的函数图和参考线，观察零点位置。
>>op=optimset('display','off');
>>fzero(@fx,1.5,op)   %求出函数零点，与上图观察结果做对比
```

```
ans = 1.8900    %结果只求出了在1.5附近的零点，试着改变初始点值求另一个零点。
>> fx1=@(x) 3*x+sin(x)-exp(x); %也可以使用匿名函数表示待求解方程
>> fzero(fx1,1.5)   %匿名函数方式定义的fx1本身类型就是函数句柄，前面不用加@符号
ans =1.8900
```

练习 4：求解方程 $f(x)=e^{x}-\sin\frac{\pi x}{3}-10$ 的零点（自己设置初始点）。

2、多元非线性方程组的根

求解多元非线性方程组根的方法有牛顿迭代法、不动点迭代法、信赖域方法等。

练习 5：设初值 $x_0=1, y_0=1, z_0=1$，求方程组的数值解。

$$\begin{cases}\sin x+y^2+\ln z-7=0\\ 3x+2^y-z^3+1=0\\ x+y+z-5=0\end{cases}$$

matlab 中用来求解非线性方程组的函数是 fsolve，调用格式为:

[x，fval]=fsolve(fun,x0,options)

其中 **fun 是待求解方程组所对应的函数句柄**，x0 是迭代法的初始点，options(可缺省)是优化选项，options 需用 optimoptions 或者 optimset 进行指定，比如 options=optimset('属性名称','属性值')。输出[x,fval]中的 x 就是方程组的解向量，fval 是解向量处的函数值向量，用于验证解向量处的函数值向量是否为 0，可以缺省。

求解方程组之前，首先需要将待求解方程组表示成对应的待求解函数。根据迭代法，求解方程就是在求解函数的零点，比如 $x+y+e^{z}=0$，相当于求函数 $F_1(x,y,z)=x+y+e^{z}$ 的零点，上面方程组中有三个方程，因此，对应着三个函数

$$\begin{cases}F_1=\sin x+y^2+\ln z-7\\ F_2=3x+2^y-z^3+1\\ F_3=x+y+z-5\end{cases}$$

于是，matlab 中待求解方程组可以用下面这种方式定义：

```
function F=fx(t)   %对应的函数句柄是@fx
%此处的F就是上面所说的输出值向量(F1,F2,F3)，t就是输入值向量(x,y,z)
F(1)=sin(t(1))+t(2).^2+log(t(3))-7;
F(2)=3*t(1)+2.^t(2)-t(3).^3+1;
F(3)=t(1)+t(2)+t(3)-5;
end
```

设置好方程组函数后，就可以调用 fsolve 求解方程组了：

```
>> option=optimset('Display','off'); %设置为不显示求解过程
 %使用optimset设置优化参数，可缺省，大家可以对比有它无它时的区别
>>x0=[1,1,1];   %指定初始迭代点
>> X=fsolve(@fx,x0,option) %使用fsolve求解
 X =
      0.5991      2.3959      2.0050
>> fx(X)
%验证解向量处的函数值向量是否为 0。也可以用格式[X,fval]=fsolve(@fx,x0,option)将上两步
并成一步。
ans =
    1.0e-10 *
     0.2213      0.3803      0.0009
```

除了用 m 函数文件表示待求解方程组外，也可以使用匿名函数表示待求解方程组，例如，上面的待求解方程组可以表示为：

```
F=@(x)[sin(x(1))+x(2).^2+log(x(3))-7, 3*x(1)+2^x(2)-x(3).^3+1, x(1)+x(2)+x(3)-5];
```

注意，方程间用逗号或者分号隔开，三个方程需要用中括号[ ]括起。

```
>>F=@(x)[sin(x(1))+x(2).^2+log(x(3))-7, 3*x(1)+2^x(2)-x(3).^3+1, x(1)+x(2)+x(3)-5];
>>op=optimset('Display','off');
>>x0=[1,1,1];
>>[X,fval]=fsolve(F,x0,op) %此处F本身类型就是函数句柄，调用时前面不用加@符号
```

写待求解方程组的要点是：需要有输出向量和输入向量。

练习 6：设初始点为$(-5,-5)$，求解方程组$\begin{cases}2x_1-x_2=e^{-x_1}\\-x_1+2x_2=e^{-x_2}\end{cases}$

要求：分别用 m 文件函数和匿名函数两种方式定义方程组并求解。