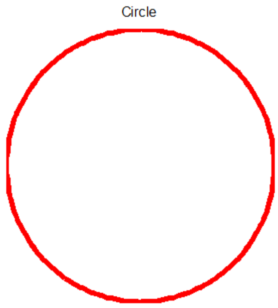
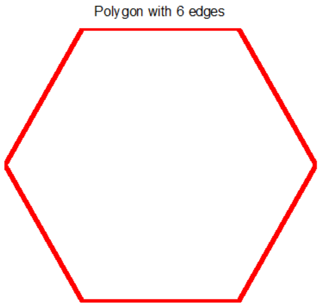
Q1. 习题 6 第 3 题(写出 M 函数源代码即可)

编写一个函数 M 文件，它的功能：没有输入量时，画出单位圆（见图 p6-1）；输入量是大于 2 的自然数 N 时，绘制正 N 边形，图名应反映显示多边形的真实边数（见图 p6-2）；输入量是“非自然数”时，给出“出错提示”。此外，函数 M 文件应有 H1 行、帮助说明和程序编写人姓名。（提示：nargin, error, int2str）





答：这里仅提供电子书所提供的官方标准答案，并附加注释（题目不要求颜色、线性、上述字符串一定要和上图完全一致，但是需要为绘图增加与 n 有关的标题，否则扣 1 分）

```
function ex6_3(n)
% ex6_3.m              画出正n边型或圆，此行即位H1行，下面三行称为帮助说明区
% ex6_3(n)              画出正n边型
% ex6_3                画出单位圆
% n                    应为大于2的自然数或0，以上内容均在help命令中可见
%
% Coded By ZZY, 2006-2-15
% Modified By Li Jia, 2018-11-4，H1行，help，姓名缺失均可以扣分
if nargin==0                %如果没有输入的变量，则绘制圆形
    t=0:pi/100:2*pi;
    y=sin(t);
    x=cos(t);
    str='Circle';
else
    if nargin~=0 && n<=2    %输入的边数不合理（小于等于2）
        error('多边形边数需要大于2')
    end
    if n-round(n)~=0        %输入的边数不为自然数
        error('请输入自然数')
    end                     %正确的多边形绘制
    t=0:2*pi/n:2*pi;
    y=sin(t);
    x=cos(t);
    str=['Polygon with ',int2str(n),' edges'];%字符串未显示边数可扣分
end
plot(x,y,'r','linewidth',4);
title(str);                 %绘图的标题由前述的字符串分别定义
axis off image;
%横纵无放缩（无需加equal），去掉坐标轴与边框，绘图边界紧贴窗口边缘
```

Q2. 分别使用 MATLAB 函数 fminbnd，自己完成的黄金分割法与自己完成的牛顿迭代法，完成一元函数$f(x)=x^2-\cos x+\mathrm{e}^{-x}$在$x\in[0,1]$之间最小值的计算。注意$f(x)$在$x\in[0,1]$是单峰函数并且是凸函数，所以各种方法均应收敛。

答：(1) fminbnd 函数求最小值的方法:

```
clear;
f=@(x)x^2-cos(x)+exp(-x);
[xmin,fmin] = fminbnd(f,0,1);
fprintf(' MATLAB的fminbnd函数得到的极值点为%f,极小值为%f\n',xmin,fmin);
```

MATLAB 的 fminbnd 函数得到的极值点为 0.258386,极小值为-0.127743

(2) 黄金分割法（若同学在改代码中循环体内不再比较和更新 fmin 提高效率可酌情加分）

```
x1=0;x2=1;
fmin = min(f(x1),f(x2));
while(1)
    alpha1 = x1*0.618+x2*0.382;
    alpha2 = x1*0.382+x2*0.618;
    if(f(alpha1)>f(alpha2))
        x1 = alpha1;
    else
        x2 = alpha2;
    end
    if(alpha2 - alpha1<1e-8) break; end
end
xmin = alpha1;fmin = f(xmin);
fprintf('黄金分割法得到的极值点为%f,极小值为%f\n',xmin,fmin);
```

黄金分割法得到的极值点为 0.258387,极小值为-0.127743

(3) 牛顿迭代法

```
x_old = 0.5;               %迭代初始点设为中点
fp = @(x) 2*x+sin(x)-exp(-x); %一阶导数函数
fp2 = @(x) 2+cos(x)+exp(-x);% 二阶导数函数

while(1)
x_new = x_old - double(fp(x_old))/double(fp2(x_old));
    if(abs(x_new - x_old)<1e-6)
        break
    end
    x_old = x_new;
end
xmin= x_new;fmin = double(f(xmin));
fprintf('牛顿迭代法得到的极值点为%f,极小值为%f\n',xmin,fmin);
```

牛顿迭代法得到的极值点为 0.258387,极小值为-0.127743

Q3. 多元函数的最小值问题有时会加上各种不同的约束条件（等式或不等式条件），从而演化为条件极值问题。请尝试利用 MATLAB 帮助系统理解函数 `fmincon` 的基本使用方法，并利用这个函数求解最优化问题 $\begin{cases}\min\limits_{x,y,z} \sin x + e^y + z \\ x^2 + y^2 + z^2 = 1\end{cases}$，初始点可以设为 $(x_0, y_0, z_0) = (1, 0, 0)$。

提示：这个问题没有线性约束，需要用空矩阵[]作为线性约束参数

答：首先利用帮助菜单或命令行键入 `doc fmincon` 来查看函数 `fmincon` 的引用页：即有约束的多元最小值计算的函数：

fmincon

Find minimum of constrained nonlinear multivariable function

collapse all in pa

Nonlinear programming solver.

Finds the minimum of a problem specified by

$$\min_x f(x) \text{ such that } \begin{cases} c(x) \le 0 \\ ceq(x) = 0 \\ A \cdot x \le b \\ Aeq \cdot x = beq \\ lb \le x \le ub, \end{cases}$$

b and beq are vectors, A and Aeq are matrices, $c(x)$ and $ceq(x)$ are functions that return vectors, and $f(x)$ is a function that returns a scalar. $f(x)$, $c(x)$, and $ceq(x)$ can be nonlinear functions.

x, lb, and ub can be passed as vectors or matrices; see Matrix Arguments.

Syntax

```
x = fmincon(fun,x0,A,b)
x = fmincon(fun,x0,A,b,Aeq,beq)
x = fmincon(fun,x0,A,b,Aeq,beq,lb,ub)
x = fmincon(fun,x0,A,b,Aeq,beq,lb,ub,nonlcon)
x = fmincon(fun,x0,A,b,Aeq,beq,lb,ub,nonlcon,options)
```

对于我们的问题，并不存在线性不等式 $A \cdot x \le b$ 与线性等式 $Aeq \cdot x = beq$，也没有 $lb \le x \le ub$ 这样的上下界限制。仅有的约束条件 $x^2 + y^2 + z^2 - 1 = 0$ 是一个非线性约束条件。根据下面的函数格式，A,b,Aeq,beq,lb,ub 均应该设为空矩阵[]。而非线性约束需要以函数句柄的形式进行输入。格式在帮助文档显示如下：

```
x = fmincon(@myfun,x0,A,b,Aeq,beq,lb,ub,@mycon)
```

where `mycon` is a MATLAB function such as

```
function [c,ceq] = mycon(x)
c = ...     % Compute nonlinear inequalities at x.
ceq = ...   % Compute nonlinear equalities at x.
```

因此根据文档描述，可以按如下方式完成代码：

```
clear,clc
f = @(x) sin(x(1)) + exp(x(2)) + x(3);
x0 = [-1,0,0];
A = []; b = []; %无线性不等式约束
Aeq = []; beq = []; %无线性等式约束
lb = []; ub = [];%无上下界约束
options = optimoptions(@fmincon,'OptimalityTolerance',1e-8);
%设定迭代终止条件
[x,fval] = fmincon(f,x0,A,b,Aeq,beq,lb,ub,@mycon,options)

function [c,ceq] = mycon(x)
```

```
c = 0; %无不等式约束，可设为0，恒成立
ceq = x(1)^2+x(2)^2+x(3)^2-1;    %计算非线性等式约束函数
end
x =
   -0.5764   -0.4419   -0.6874
fval =
   -0.5896
```

Q4. (开放性问题，选做)l_1正则化问题在计算与应用数学的最优化领域，数据科学的压缩感知，统计的 Lasso 问题中都有很重要的应用价值。本题请利用函数 fminunc 与自己编写的迭代算法分别求解多元问题$\min_x \frac{1}{2}\|Ax-b\|_2^2+\lambda\|Bx\|_1$的解。其中，$A, B$为$5\times 5$矩阵,$b$为已知5维列向量,$\lambda = 0.1$为常参数。$x$为 5 维列向量，迭代初值自定。具体定义格式如下：

```
rng default
A = rand(5); B = rand(5); b = rand(5,1); lambda = 0.1;
```

注意事项：(1)使用函数 fminunc 时要注意如果使用行向量来定义 x0 与 x 时表达式要进行必要的变化或转置。

(2)自己写算法会很难，主要是 1-范数是不可导的所以无法直接使用梯度下降法或牛顿迭代法。因此这一问量力而行，可以使用的算法思想主要有 ADMM 算法或基于 Split 思想的 Proximal 算法

答：本题使用 MATLAB 函数 fminunc 解答其实并不困难，只需要利用范数来定义好目标函数即可。具体代码如下：

```
clear, clc;
rng default
A = rand(5); B = rand(5); b = rand(5,1); lambda = 0.1;

f = @(x) 1/2 * norm(A*x'-b,2)^2 + lambda*norm(B*x',1);
x0 = zeros(1,5);
options = optimoptions(@fminunc,'OptimalityTolerance',1e-8);
[x,fval] = fminunc(f,x0,options)

Local minimum possible.

fminunc stopped because it cannot decrease the objective function
along the current search direction.

x =
   -0.0001   -0.0621    0.7729   -0.3578    0.0713
fval =
    0.1945
```

这个输出结果表明函数可能获取到了最小值 0.1945 与对应的解向量 x。因为函数 fminunc 只支持行向量的初始值及求解，因此在 f 的定义中要注意 x 需要转置。

该问题$\min\limits_x \frac{1}{2}\|Ax-b\|_2^2+\lambda\|Bx\|_1$是一个凸问题，所以梯度下降类算法可以获得全局最优解。但是，$\lambda\|Bx\|_1$的不可导导致无法直接使用梯度下降法或牛顿法。这里介绍一下交替方向乘子迭代法（ADMM），详细资料可见相应附件。该方法的核心是设计辅助变量$u=Bx$，而后将原问题转化为增广问题$\min\limits_{\substack{x\\Bx=u}} \frac{1}{2}\|Ax-b\|_2^2+\lambda\|u\|_1+\frac{1}{2}\|Bx-u\|_2^2$，这里的最后一项看起来没什么意义，不过可以一定程度上优化算法（不加这一项后面对增广拉格朗日函数做梯度下降法也是可以的）。此后，我们可以将限制条件$Bx=u$与原问题混合定义出增广拉格朗日函数（拉格朗日乘数法的推广）的 minimax 问题

$$\min_{x,u}\max_{d}\frac{1}{2}\|Ax-b\|_2^2+\lambda\|u\|_1+\frac{1}{2}\|Bx-u\|_2^2+\langle Bx-u,d\rangle$$

这里的d是对偶变量，我们尽量使其增大拉格朗日函数，从而达到“强迫$Bx=u$”的目的。然后对三个变量交替迭代，可以得到如下的算法：

$$\begin{cases} x^{k+1}=\min\limits_x \frac{1}{2}\|Ax-b\|_2^2+\frac{1}{2}\left\|Bx-u^k\right\|_2^2+\left\langle Bx-u^k,d^k\right\rangle \\ u^{k+1}=\min\limits_u \lambda\|u\|_1+\frac{1}{2}\left\|Bx^{k+1}-u\right\|_2^2+\left\langle Bx^{k+1}-u,d^k\right\rangle \\ d^{k+1}=\max\limits_d\left\langle Bx^{k+1}-u^{k+1},d\right\rangle \end{cases}$$

实际算法中，我们可以对第一步和第二步的目标函数进行变形，使其成为更加简单清晰的问题，第三步我们将其改为最大值问题的一步迭代。

$$\begin{cases} x^{k+1}=\min\limits_x \frac{1}{2}\|Ax-b\|_2^2+\frac{1}{2}\left\|Bx-u^k+d^k\right\|_2^2 \\ u^{k+1}=\min\limits_u \lambda\|u\|_1+\frac{1}{2}\left\|Bx^{k+1}-u+d^k\right\|_2^2 \\ d^{k+1}=d^k+\left(Bx^{k+1}-u^{k+1}\right) \end{cases}$$

第一步本质是一个 10 行 5 列线性方程组（变分问题），因此可用 MATLAB 运算符\完成最小二乘法。第二步可以用软阈值算法$u^{k+1}=\mathrm{sign}\left(Bx^{k+1}-u+d^k\right)\cdot\max\left(\left|Bx^{k+1}-u+d^k\right|-\lambda,0\right)$，相应的算法原理可以参见附加的源自 CSDN 的资料。

因此，我们的算法代码如下：

```
x = zeros(5,1); %初始x估计值
x_old = ones(5,1); % x旧值假设初始与x并不相同
u = B*x; %辅助变量为B*x的近似值
d = u-B*x; %初始对偶变量的值（可直接设为0向量）
while(norm(x_old - x)>1e-8)
    x_old = x;
    x = [A;B]\[b;u-d]; %获取最小二乘解更新x
    %或用x = [A'*A+B'*B]\[A'*b'+B'*(u-d)]达到相同效果
    u = sign(B*x + d) .* max(abs(B*x+d)-lambda,0);
    %软阈值算法获取辅助变量u的值
    d = d + (B*x - u);
end
x_ADMM = x
fval_ADMM = f(x')
```

运算结果：

```
x_ADMM =
   -0.0305
   -0.0744
    0.7988
   -0.4112
    0.1455

fval_ADMM =
    0.1936
```

运算结果的目标函数比 `fminunc` 计算的更小，且解向量并不接近。事实上，提高 `fminunc` 的精度仍然无法使其获得更小的解。这说明 `fminunc` 并未获得正确的最优解，而 ADMM 算法更有可能获得正确的全局最优解。