Q1:

% 用自带函数 integral 计算

```
>> format long
>> F1=@(x)exp(-abs(x)).*abs(sin(x));
>> Q1=integral(F1,-5*pi,10*pi,'AbsTol',1e-9)
Q1 =
    1.090331540770800
```

% 用 trapz，梯形公式计算

```
>> t=-5*pi:pi/10:10*pi;
>> F2=exp(-abs(t)).*abs(sin(t));
>> Q2=trapz(F2) *pi/10
Q2 =
     1.0725
```

% 用符号运算，保留 16 位有效数字作为真实值

```
>> syms m F Q
>> F(m)=exp(-abs(m)).*abs(sin(m));
>> Q= vpa( int(F,m,-5*pi,10*pi) ,16)
ans =
    1.090331328569942
```

```
% 比较两种算法的绝对误差
>> vpa(abs(Q1-Q),10)
ans =
   0.0000002122008578
>> vpa(abs(Q2-Q),10)
ans =
   0.01787595098
```

% 可见，trapz 梯形公式计算误差较自带函数 integral 大，因为 intergral 使用的是代数精度更高的算法，如自适应辛普森公式

Q2:

```
>> format long
>> M=rand(2,10^6);
>> P=(M(1,:)-0.5).^2+(M(2,:)-0.5).^2;
% 计数落在圆内的点，除以总点数作为 pi/4 的预测值
>> pipredict=length(find(P<=0.25))/10^6*4
pipredict =
   3.141156000000000
```

Q3：

```
>> format long
>> t=0:10^(-5):2;
```

>> f=@(x)log(1+x);

% 用 diff 差分求近似导数

>> D1=diff(f(t))/10^(-5);

% 用 gradient 差分求近似导数

>> D2=gradient(f(t))/10^(-5);

% 输出两种算法下的 x=1 处导数（真实值是 0.5）

>> D1(end/2)

ans =

    0.500001250003379

>> D2((end+1)/2)

ans =

    0.500000000008827

% 可见，gradient 算法更加准确，因为它使用了更多的点来拟合，因此受极端值影响更小

Q4（选做）：

1)>> format long

>> syms x y F f

>> f=(x + y)/(x^2 + y^2 + 1)^(3/2);

>> F=int(int(f,x,0,1),y,0,1)

F =

    0.445789277114269

2)>> format long

% 运用二维中矩形公式

```
>> f=@(x,y)(x + y)./(x.^2 + y.^2 + 1).^(3/2);
>> a=(0.5:1999.5)/2000;
a=repmat(a,2000,1);
b=a';
Fsqu=sum(sum(f(a,b)))/2000^2*1
Fsqu =
    0.445789297206669
```

% 运用二维辛普森复合公式

```
>> s=0;
for i=1:2000
for j=1:2000
s=s...
+1/36*(f((i-1)/2000,(j-1)/2000)+f(i/2000,(j-1)/2000)+f((i-1)/2000,j/2000)+f(i/2000,j/2000))...
+4/36*(f((i-0.5)/2000,(j-1)/2000)+f((i-1)/2000,(j-0.5)/2000)+f(i/2000,(j-0.5)/2000)+f((i-0.5)/2000,j/2000))...
+16/36*f((i-0.5)/2000,(j-0.5)/2000);
end
end
>> Fsim =s*1/2000^2
```

Fsim =

0.445789277114280

% 查看计算误差

>> vpa(abs(F-Fsqu),10)

ans =

0.00000002009240024

>> vpa(abs(F-Fsim),10)

ans =

1.089579815e-14

% 可以看出，辛普森公式比中矩形公式要精确得多，因为它使用了更多的点，代数精度更高

矩阵运算方法 credit by 唐文萱 16342159