Q1

用符号积分

```
syms x
tic
f = exp(-abs(x))*abs(sin(x));
a_s = vpa(int(f, 5*pi, 10*pi),16);
toc
```

```
Elapsed time is 0.045077 seconds.
```

用integral

```
tic
F = @(x)exp(-abs(x)).*abs(sin(x));
a_int = integral(F, 5*pi, 10*pi, 'AbsTol', 1e-9);
toc
```

```
Elapsed time is 0.015513 seconds.
```

```
err_int = double(abs(a_s-sym(a_int)))
```

```
err_int = 1.8078e-11
```

用trapz

```
d = pi/120;
tic
t = 5*pi:d:10*pi;
y = exp(-abs(t)).*abs(sin(t));
a_tra = trapz(t, y);
toc
```

```
Elapsed time is 0.011775 seconds.
```

```
err_tra = double(abs(a_s-sym(a_tra)))
```

```
err_tra = 9.3848e-12
```

可见当把integral的绝对精度设为1e-9时，实际的误差会更小（在这个问题中为1e-11）

trapz的精度显然与步长有关，而调整步长使得与integral的精度相同后，发现trapz会比integral稍快

符号计算最慢

Q2

```
clear
sample = rand(2, 1e6); % 在正方形内均匀采样
in_cir_sample = (sample(1,:)-(1/2)).^2+(sample(2,:)-(1/2)).^2 < (1/4); % 圆内的样本点
num = sum(in_cir_sample);
pi_appr = 4*num/1e6;
err = abs(pi-pi_appr)
```

```
err = 4.3665e-04
```

Q3

```
clear
d = 1e-5;
t = 0:1e-5:2;
f = log(1+t);
df_d = diff(f)/d;
df_g = gradient(f)/d;
err_d=abs(df_d(t==1)-(1/2))
```

```
err_d = 1.2500e-06
```

```
err_g=abs(df_g(t==1)-(1/2))
```

```
err_g = 8.8267e-12
```

可见gradient比diff要精确

Q4

(1)

$$\int_0^1 \int_0^1 \frac{x+y}{\sqrt{(1+x^2+y^2)^3}} dxdy$$

$$= \int_0^1 \int_0^1 \frac{x}{\sqrt{(1+x^2+y^2)^3}} dxdy + \int_0^1 \int_0^1 \frac{y}{\sqrt{(1+x^2+y^2)^3}} dxdy$$

$$= \int_0^1 \int_0^1 \frac{x}{\sqrt{(1+x^2+y^2)^3}} dxdy + \int_0^1 \int_0^1 \frac{y}{\sqrt{(1+x^2+y^2)^3}} dydx$$

$$= 2\int_0^1 \int_0^1 \frac{x}{\sqrt{(1+x^2+y^2)^3}} dxdy$$

$$= \int_0^1 \int_0^1 (1+y^2+x^2)^{-\frac{3}{2}} dx^2 dy$$

$$= 2\int_0^1 \frac{1}{\sqrt{1+y^2}} - \frac{1}{\sqrt{2+y^2}} dy$$

$$= 2\ln\frac{\sqrt{2}+2}{\sqrt{3}+1}$$

(2)

符号积分

```
clear
syms x y
f = (x+y)/(1+x^2+y^2)^(3/2);
a = int(int(f,x,[0,1]),y,[0,1])
```

```
a =
```

$$\log\left(\left(\frac{\sqrt{2}}{2}-\sqrt{3}-\frac{\sqrt{6}}{2}+1\right)^2\right)$$

中矩形法

```
N = 2000; d = 1/N;
x = d/2:d:1;
x_1 = repmat(x, N, 1);
x_2 = repmat(x.^2, N, 1);
f = (x_1+x_1')./(1+x_2+x_2').^(3/2); % 利用被积函数的对称性，类似于生成Hilbert矩阵
a_rec= sum(reshape(f,1,[]))*d*d;
err_rec = double(abs(a-sym(a_rec)))
```

```
err_rec = 2.0092e-08
```

Simpson法

```
t = 0:d:1; % 取样点为网格点，于是采样点数目与中矩形法大致相同
M = length(t);
coeff_x = ones(1, M);
coeff_x(2:2:M) = 4;
coeff_x(3:2:M) = 2;
coeff_x(end) = 1;
coeff = coeff_x'*coeff_x; % Simpson公式的系数
t_1 = repmat(t, M, 1);
t_2 = repmat(t.^2, M, 1);
g = (t_1+t_1')./(1+t_2+t_2').^(3/2);
a_sim = sum(reshape(g.*coeff,1,[]))*d*d/9;
err_sim = double(abs(a-sym(a_sim)))
```

```
err_sim = 4.8590e-16
```

可见Simpson法比中矩形法更精确

实际上在这个问题中若把Simpson法中的取样点数目 定为4001*4001，误差约为1e-14，反而没有只在网格点处取样更精确