1)

$$C_{n-k}^{(n)} = \frac{(-1)^k}{nk!(n-k)!}\int_0^n \prod_{i=0,i\neq n-k}^{n}(t-i)dt = \frac{(-1)^k}{nk!(n-k)!}\int_0^n \prod_{i=0,i\neq n-k}^{n} -(s-(n-i))ds$$

$$= \frac{(-1)^{n+k}}{nk!(n-k)!}\int_0^n \prod_{i=0,i\neq k}^{n}(s-i)ds$$

$\because (-1)^{n+k} = (-1)^{n-k}$

$\therefore$得证

2)

令$c = (a+b)/2$

$$\int_a^b f(x)dx = -\int_c^a f(x)dx + \int_c^b f(x)dx$$

由带Lagrange余项的Taylor展开式可知

$$\int_c^b f(x)dx = \int_c^b f(c) + (x-c)f'(c) + \frac{(x-c)^2}{2}f''(\xi)dx$$

$$= (b-c)f(c) + \frac{(b-c)^2}{2}f'(c) + \frac{(x-c)^3}{6}f''(\xi)$$

$$\therefore \int_a^b f(x)dx = (b-a)f(c) + \frac{(b-a)^3}{48}(f''(\xi_1) + f''(\xi_2))$$

$\because$二次导数连续

$$\therefore \int_a^b f(x)dx = (b-a)f(\frac{a+b}{2}) + \frac{(b-a)^3}{24}f''(\xi)$$

3)

对于$f(x) = ax^2 + bx + c$

$$\int_{-1}^1 f(x)dx = \frac{2}{3}a + 2c$$

$$\begin{pmatrix} 1 & 1/9 & 1/9 \\ -1 & -1/3 & 1/3 \\ 1 & 1 & 1 \end{pmatrix}\begin{pmatrix} A_1 \\ A_2 \\ A_3 \end{pmatrix} = \begin{pmatrix} 2/3 \\ 0 \\ 2 \end{pmatrix}$$

解得

$A_1 = 1/2, A_2 = 0, A_3 = 3/2$

4)

对于$f(x) = ax^2 + bx + c$

$$\int_{-1}^{1} f(x)dx = \frac{2}{3}a + 2c$$

$$\begin{pmatrix} 1 & x_1^2 & x_2^2 \\ -1 & x_1 & x_2 \\ 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} 1/3 \\ 2/3 \\ 1 \end{pmatrix} = \begin{pmatrix} 2/3 \\ 0 \\ 2 \end{pmatrix}$$

解得

$x_1 = \frac{-2+3\sqrt{6}}{10}, x_2 = \frac{-1-\sqrt{6}}{5}$ （舍去）

$x_1 = \frac{-2-3\sqrt{6}}{10}, x_2 = \frac{-1+\sqrt{6}}{5}$

5)

利用python编程计算

```
import numpy as np
def f1(x):
    return np.exp(x)
def f2(x):
    return np.sin(x) ** 2
def f3(x):
    return np.exp(-x**2)
def f4(x):
    if x == 0:
        return 1
    return np.sin(x)/x

def f1_int(a, b):
    return f1(b) - f1(a)
def f2_int(a, b):
    return 1/2*(b-a) - 1/4*np.sin(2*b) + 1/4*np.sin(2*a)

def solve(fx, a, b):
    ix1 = (fx(a) + fx(b))/2 * (b-a)
    print(ix1)
    ix2 = (fx(a) + fx((a+b)/2) + fx(b))/3 * (b-a)
    print(ix2)
    return

solve(f1, 1.1, 1.8)
print(f1_int(1.1, 1.8))
solve(f2, 0, np.pi/2)
print(f2_int(0, np.pi/2))
solve(f3, 1, 2)
solve(f4, 0, np.pi/2)
'''
Out:
3.1688347209257826  梯形
3.10728320082324 6   Simpson
3.045481440466513   准确值

0.7853981633974483
0.7853981633974483
0.7853981633974483
```

```
0.19309754003008825
0.16386476820734694

1.2853981633974483
1.3283366297226638
'''
```

结果见输出

再利用误差公式计算得

$R_{11}(f) = (4x^2 - 2)e^{-x^2}/12$

$R_{12}(f) = (16x^4 - 48x^2 + 12)e^{-x^2}/2880$

$R_{21}(f) = ((x^2 - 2)sinx + 2xcosx)/12x^3$

$R_{22}(f) = ((x^4 - 12x^2 + 24)sinx + (4x^3 - 24x)cosx)/2880x^5$

通过python求得

```
def r1(x):
    return np.exp(-x**2)*(4*x**2-2)/12
def r2(x):
    return np.exp(-x**2)*(16*x**4-48*x**2+12)/2880
def r3(x):
    return ((x**2-2)*np.sin(x)+2*x*np.cos(x))/12/x**3
def r4(x):
    return ((x**4-12*x**2+24)*np.sin(x)+(4*x**3-24*x)*np.cos(x))/2880/x**5

def solve2(rx, a, b):
    r = 0
    for i in np.arange(a, b, 0.001):
        temp = np.fabs(rx(i))
        if r < temp:
            r = temp
    print(r)
    return

solve2(r1, 1, 2)
solve2(r2, 1, 2)
solve2(r3, 0, np.pi/2)
solve2(r4, 0, np.pi/2)
'''
Out:
0.07437670552882485
0.0025547183414683493
0.027777769450940126
6.987080667128806e-05
'''
```