1)

$$lim_{n\to\infty}T_n = lim_{n\to\infty}\frac{b-a}{n}\sum_{k=1}^{n-1}f(x_k) = \int_a^b f(x)dx$$

$$lim_{n\to\infty}S_n = lim_{n\to\infty}\frac{2}{3}\frac{b-a}{n}\sum_{k=0}^{n-1}f(x_{k+1/2}) + \frac{1}{3}\frac{b-a}{n}\sum_{k=1}^{n-1}f(x_k) = \int_a^b f(x)dx$$

2)

$$T_n = \frac{h}{2}(f(a) + 2\sum_{k=1}^{n-1}f(x_k) + f(b))$$

$$S_n = \frac{h}{6}(f(a) + 4\sum_{k=0}^{n-1}f(x_{k+1/2}) + 2\sum_{k=1}^{n-1}f(x_k) + f(b))$$

直接代入计算可得

$$\frac{4}{3}T_{2n}(f) - \frac{1}{3}T_n(f) = S_n(f)$$

3.a)

$$\begin{aligned} & T_m(x)T_n(x) \\ =& cos(marccos(x))cos(narccos(x)) \\ =& \frac{1}{2}(cos((m+n)arccos(x)) + cos((m-n)arccos(x))) \\ =& \frac{1}{2}(T_{m+n}(x) + T_{m-n}(x)) \end{aligned}$$

3.b)

$$T_m(T_n(x)) = cos(marccos(cos(narccos(x)))) = cos(mnarccos(x)) = T_n(T_m(x))$$

$$T_{mn}(x) = cos(mnarccos(x)) = T_m(T_n(x))$$

得证

3.c)

$$\because T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x), T_0(x) = 1$$

$$\therefore T_n(x) = 2^{n-1}x^n + \dots$$

得证

4)

不妨设$M=1$

令$T_n(x)=1,-1$

得到n+1个解$x_k=cos\frac{k\pi}{n}$

令$f_n(x)=P_n(x)-T_n(x)$

不妨设$f_n(x)$不恒为零

则$f_n(x_{2k})\leq 0$

$f_n(x_{2k+1})\geq 0$

从而f_n在$[x_0,x_n]$上有n个零点（含重数）

由于$f_n(x)=P_n(x)-T_n(x)$不大于n次

所以$f_n(x)$在$[-1,1]$外没有零点

得到$f_n(1)\leq 0$

所以$y>1,P_n(x)-T_n(x)=f_n(y)\leq 0$

得证

5)

$$R(f)=-\frac{b-a}{12}h^2f''(\xi)=-\frac{1}{6}h^2e^{\xi}cos\xi\leq 10^{-6}$$

$$\therefore n\geq 1110\geq\sqrt{\frac{e^{\xi}cos\xi}{6*10^{-6}}}$$

用python编程计算得

```
import numpy as np

def fun(x):
    return np.exp(x)*np.sin(x)

def tx(a,b,n):
    h=(b-a)/n
    x=a
    s=fun(x)-fun(b)
    for k in range(1,n+1):
        x=x+h
        s=s+2*fun(x)
    res=(h/2)*s
    return res

print(tx(1,2,1110))
#Out: 4.487560327818263
```

$\therefore I\approx 4.4875603$

6)

用python编程计算得

```
import math
import numpy as np

def fun(x):
    return np.exp(x)*np.sin(x)

def T_2n(a, b, n, T_n):
    h = (b - a)/n
    sum_f = 0.
    for k in range(0, n):
        sum_f = sum_f + fun(a + (k + 0.5)*h)
    T_2n = T_n/2. + sum_f*h/2.
    return T_2n

def Romberg(a, b, err_min):
    kmax = 99
    tm = np.zeros(kmax,dtype = float)
    tm1 = np.zeros(kmax,dtype = float)
    tm[0] = 0.5*(b-a)*(fun(a) + fun(b))
    err = 1.
    k = 0
    while(err>err_min):
        n = 2**k
        m = 1
        tm1[0] = T_2n(a, b, n, tm[0])
        while(err>err_min and m <= (k+1)):
            tm1[m] = tm1[m-1]+(tm1[m-1]-(tm[m-1]))/(4.**m-1)
            result = tm1[m]
            err1 = abs(tm1[m]-tm[m-1])
            err2 = abs(tm1[m]-tm1[m-1])
            err = min(err1,err2)
            m = m+1
        tm = np.copy(tm1)
        k = k+1
    return result

print(Romberg(1, 3, 1.e-6))
#Out: 10.950170352167321
```

$\therefore I \approx 10.9501704$