Problem 8

In [18]:
```
import numpy as np
import sys
```

In [30]:
```
#8(c)



epsilon=sys.float_info.epsilon
k = 20
Nfact=np.math.factorial(20)/epsilon
Nguess=0
Nguessf=np.math.factorial(Nguess)
while Nguessf<Nfact:
    Nguess+=1
    Nguessf=np.math.factorial(Nguess)

N=Nguess
# np.math.factorial(Nguess-1)-Nfact #test that we have the right N
print('Need N={} to acheive this error in y_20'.format(N))
```

```
Need N=32 to acheive this error in y_20
```

In [62]:
```
# 8(d)


y20back=0 #initial value at N=32
for k in range(N, 20, -1): #loop backwards, updating y each time
#     print(k)
    y20back=(np.exp(1)-y20back)/(k) #using k=n+1, so at final step co
mputing n=20 using k=20+1

print('\n The backwards recurrance relation yeilds y_20 = {} \n'.form
at(y20back))

x=np.linspace(0., 1., int(1e5))
y20int=np.trapz(np.exp(x)*x**20, x=x)
print('\n Numerical integration: y_20 = {} \n'.format(y20int))

y20wolfram = 0.12380383076256994869 #value from wolfram alpha

print('\n Wolfram alpha: y_20 = {} \n'.format(y20wolfram))

print('\n backwards integration y_20 - wolfram y_20: {} \n eps: {}'.f
ormat(abs(y20wolfram-y20back), epsilon))
```

```
 The backwards recurrance relation yeilds y_20 = 0.12380383076256993


 Numerical integration: y_20 = 0.12380383123827879


 Wolfram alpha: y_20 = 0.12380383076256996


 backwards integration y_20 - wolfram y_20: 2.7755575615628914e-17
 eps: 2.220446049250313e-16
```

We see that the backwards recurrence relation is very accurate even starting with an incorrect seed of $y_{32} = 0$. The recurrence relation computed y_{20} whose error with respect to the precise answer from Wolfram Alpha is less than the machine epsilon, and is much better than the y_{20} calculated via numerical integration.