Question 2

In [1]:
```
import numpy as np
```

In [2]:
```
#function to round to 5 sig figs
def round_5_sigfigs(x):
    x = '{:.5g}'.format(x)
    x = float(x)
    return x
```

In [3]:
```
#2 (a)

exp=5.5
k=30 # number of terms in series
#use exponent notation so readable
numerator = ['{:.4e}'.format(1.)] #create lists and append to them
denominator = ['{:.4e}'.format(1.)]
seriesterm = ['{:.4e}'.format(1.)]

for n in range(1,k+1):
    #treat exponentiation as repeated multiplication, so multiply by
 numerator of previous term
    tempnum=float(numerator[n-1])*round_5_sigfigs(exp) #previous nume
rator already has 5 sigfigs
    numerator.append('{:.4e}'.format(tempnum)) #append to list of num
erators

    #same trick for denomenator
    tempden=float(denominator[n-1])*round_5_sigfigs(float(n))
    denominator.append('{:.4e}'.format(tempden))

    tempseries=round_5_sigfigs(float(numerator[n])/float(denominator[
n] ))
    seriesterm.append('{:.4e}'.format(tempseries))


print('Numerators:\n {} \n'.format(numerator))
print('Denominators:\n {}\n'.format(denominator))
print('Terms in series:\n {}\n'.format(seriesterm))
```

```
Numerators:
 ['1.0000e+00', '5.5000e+00', '3.0250e+01', '1.6638e+02', '9.1509e+0
2', '5.0330e+03', '2.7682e+04', '1.5225e+05', '8.3738e+05', '4.6056e+
06', '2.5331e+07', '1.3932e+08', '7.6626e+08', '4.2144e+09', '2.3179e
+10', '1.2748e+11', '7.0114e+11', '3.8563e+12', '2.1210e+13', '1.1666
e+14', '6.4163e+14', '3.5290e+15', '1.9410e+16', '1.0676e+17', '5.871
8e+17', '3.2295e+18', '1.7762e+19', '9.7691e+19', '5.3730e+20', '2.95
51e+21', '1.6253e+22']

Denominators:
 ['1.0000e+00', '1.0000e+00', '2.0000e+00', '6.0000e+00', '2.4000e+0
1', '1.2000e+02', '7.2000e+02', '5.0400e+03', '4.0320e+04', '3.6288e+
05', '3.6288e+06', '3.9917e+07', '4.7900e+08', '6.2270e+09', '8.7178e
+10', '1.3077e+12', '2.0923e+13', '3.5569e+14', '6.4024e+15', '1.2165
e+17', '2.4330e+18', '5.1093e+19', '1.1240e+21', '2.5852e+22', '6.204
5e+23', '1.5511e+25', '4.0329e+26', '1.0889e+28', '3.0489e+29', '8.84
18e+30', '2.6525e+32']

Terms in series:
 ['1.0000e+00', '5.5000e+00', '1.5125e+01', '2.7730e+01', '3.8129e+0
1', '4.1942e+01', '3.8447e+01', '3.0208e+01', '2.0768e+01', '1.2692e+
01', '6.9805e+00', '3.4902e+00', '1.5997e+00', '6.7679e-01', '2.6588e
-01', '9.7484e-02', '3.3510e-02', '1.0842e-02', '3.3128e-03', '9.5898
e-04', '2.6372e-04', '6.9070e-05', '1.7269e-05', '4.1297e-06', '9.463
8e-07', '2.0821e-07', '4.4043e-08', '8.9715e-09', '1.7623e-09', '3.34
22e-10', '6.1274e-11']
```

In [4]:
```
#2 (b)

exp=5.5
k=30 # number of terms in series

partialsums = ['{:.4e}'.format(1.)]
prevsum=1. #keep track of previous value of sum
n_converge=None
for n in range(1,k+1):
    currentsum = round_5_sigfigs(prevsum+float(seriesterm[n]))
    partialsums.append('{:.4e}'.format(currentsum))
    #check if prevsum is the same, if so, series has converged
    if prevsum == currentsum and n_converge is None:
        n_converge = n #starts from zero

    prevsum = currentsum #update previous sum


print('Partial sums:\n {} \n'.format(partialsums))

print('Series converged at   k={} \n'.format(n_converge))

percise_exp=np.exp(5.5)
print('This method computes exp(5.5)={}, np.exp(5.5)={:.10e} \n'.form
at(
    partialsums[n_converge],percise_exp ))

rel_error=np.abs(percise_exp-float(partialsums[n_converge]))/percise_
exp
print('The relative error is {} \n'.format(rel_error))
```

```
Partial sums:
 ['1.0000e+00', '6.5000e+00', '2.1625e+01', '4.9355e+01', '8.7484e+0
1', '1.2943e+02', '1.6788e+02', '1.9809e+02', '2.1886e+02', '2.3155e+
02', '2.3853e+02', '2.4202e+02', '2.4362e+02', '2.4430e+02', '2.4457e
+02', '2.4467e+02', '2.4470e+02', '2.4471e+02', '2.4471e+02', '2.4471
e+02', '2.4471e+02', '2.4471e+02', '2.4471e+02', '2.4471e+02', '2.447
1e+02', '2.4471e+02', '2.4471e+02', '2.4471e+02', '2.4471e+02', '2.44
71e+02', '2.4471e+02']

Series converged at   k=18

This method computes exp(5.5)=2.4471e+02, np.exp(5.5)=2.4469193226e+0
2

The relative error is 7.383870654188337e-05
```

In [5]:
```
#2 (c)

exp=5.5
k=30 # number of terms in series

partialsums_b = ['{:.4e}'.format(1.)]
prevsum=1. #keep track of previous value of sum
n_converge_b=None

for n in range(1,k+1):
    tempsum=0.
    for ind in range(n, -1, -1): #do sum backwards for each partial s
um
        tempsum =round_5_sigfigs(tempsum+float(seriesterm[ind]))

    currentsum = round_5_sigfigs(tempsum)
    partialsums_b.append('{:.4e}'.format(currentsum))
    #check if prevsum is the same, if so, series has converged
    if prevsum == currentsum and n_converge_b is None:
        n_converge_b = n


    prevsum = currentsum #update previous sum

print('Partial sums from right to left:\n {} \n'.format(partialsums_b
))


if n_converge_b is None:
    n_converge_b=k
print('Series converged at  k={} \n'.format(n_converge_b))

percise_exp=np.exp(5.5)
print('right to left computes exp(5.5)={}, np.exp(5.5)={:.10e} \n'.fo
rmat(partialsums_b[n_converge_b],percise_exp ))

rel_error_b=np.abs(percise_exp-float(partialsums_b[n_converge_b]))/pe
rcise_exp
print('The relative error right to left is {} \n'.format(rel_error_b
))

print('Partial sums coverge 1 k sooner,'
      ' and final result is closer to percise value and relative erro
r is thus reduced')
```

```
Partial sums from right to left:
 ['1.0000e+00', '6.5000e+00', '2.1625e+01', '4.9355e+01', '8.7484e+0
1', '1.2942e+02', '1.6788e+02', '1.9809e+02', '2.1885e+02', '2.3154e+
02', '2.3851e+02', '2.4201e+02', '2.4362e+02', '2.4428e+02', '2.4456e
+02', '2.4466e+02', '2.4469e+02', '2.4469e+02', '2.4469e+02', '2.4469
e+02', '2.4469e+02', '2.4471e+02', '2.4471e+02', '2.4471e+02', '2.447
1e+02', '2.4471e+02', '2.4471e+02', '2.4471e+02', '2.4471e+02', '2.44
71e+02', '2.4471e+02']

Series converged at  k=17

right to left computes exp(5.5)=2.4469e+02, np.exp(5.5)=2.4469193226e
+02

The relative error right to left is 7.896722227439787e-06

Partial sums coverge 1 k sooner, and final result is closer to percis
e value and relative error is thus reduced
```

In [6]:

```
#2 (d) setup
exp_n=-5.5
series_n=[]
#first change partial sums
for n in range(k+1):
    tempnegseries=float(seriesterm[n])
    if n & 1: #if odd, flip sign
        tempnegseries=-tempnegseries
    series_n.append('{:.4e}'.format(tempnegseries))

#2 (d) i, adding left to right
partialsumsf_n = ['{:.4e}'.format(1.)]
prevsum=1. #keep track of previous value of sum
n_convergef_n=None
for n in range(1,k+1):
    currentsum = round_5_sigfigs(prevsum+float(series_n[n]))
    partialsumsf_n.append('{:.4e}'.format(currentsum))
    #check if prevsum is the same, if so, series has converged
    if prevsum == currentsum and n_convergef_n is None:
        n_convergef_n = n #starts from zero
    prevsum = currentsum #update previous sum

print('Partial sums:\n {} \n'.format(partialsumsf_n))
print('Series converged at   k={} \n'.format(n_convergef_n))
percise_exp=np.exp(-5.5)
print('This method computes exp(-5.5)={}, np.exp(-5.5)={:.10e} \n'.fo
rmat(
    partialsumsf_n[n_convergef_n],percise_exp ))
rel_error=np.abs(percise_exp-float(partialsumsf_n[n_convergef_n]))/pe
rcise_exp
print('The relative error is {} \n'.format(rel_error))
```

```
Partial sums:
 ['1.0000e+00', '-4.5000e+00', '1.0625e+01', '-1.7105e+01', '2.1024e+
01', '-2.0918e+01', '1.7529e+01', '-1.2679e+01', '8.0890e+00', '-4.60
30e+00', '2.3775e+00', '-1.1127e+00', '4.8700e-01', '-1.8979e-01',
'7.6090e-02', '-2.1394e-02', '1.2116e-02', '1.2740e-03', '4.5868e-0
3', '3.6278e-03', '3.8915e-03', '3.8224e-03', '3.8397e-03', '3.8356e-
03', '3.8365e-03', '3.8363e-03', '3.8363e-03', '3.8363e-03', '3.8363e
-03', '3.8363e-03', '3.8363e-03']

Series converged at   k=26

This method computes exp(-5.5)=3.8363e-03, np.exp(-5.5)=4.0867714385e
-03

The relative error is 0.06128834025477125
```

In [7]:
```
#2 (d) ii, adding right to left
partialsums_nb = ['{:.4e}'.format(1.)]
prevsum=1. #keep track of previous value of sum
n_converge_nb=None

for n in range(1,k+1):
    tempsum=0.
    for ind in range(n, -1, -1): #do sum backwards for each partial s
um
        tempsum =round_5_sigfigs(tempsum+float(series_n[ind]))

    currentsum = round_5_sigfigs(tempsum)
    partialsums_nb.append('{:.4e}'.format(currentsum))
    #check if prevsum is the same, if so, series has converged
    if prevsum == currentsum and n_converge_nb is None:
        n_converge_nb = n

    prevsum = currentsum #update previous sum

print('Partial sums from right to left:\n {} \n'.format(partialsums_n
b))
if n_converge_nb is None:
    n_converge_nb=k
print('Series converged at  k={} \n'.format(n_converge_nb))
percise_exp=np.exp(-5.5)
print('right to left computes exp(-5.5)={}, np.exp(-5.5)={:.10e} \n'.
format(partialsums_nb[n_converge_nb],percise_exp ))
rel_error_nb=np.abs(percise_exp-float(partialsums_nb[n_converge_nb]))
/percise_exp
print('The relative error right to left is {} \n'.format(rel_error_nb
))
print('Partial sums again converge sooner,'
      ' the final result is closer to percise value and relative erro
r is thus reduced')
```

```
Partial sums from right to left:
 ['1.0000e+00', '-4.5000e+00', '1.0625e+01', '-1.7105e+01', '2.1024e+
01', '-2.0918e+01', '1.7529e+01', '-1.2679e+01', '8.0890e+00', '-4.60
30e+00', '2.3770e+00', '-1.1130e+00', '4.8700e-01', '-1.9000e-01',
'7.6000e-02', '-2.1000e-02', '1.2000e-02', '1.0000e-03', '5.0000e-0
3', '4.0000e-03', '4.0000e-03', '4.0000e-03', '4.0000e-03', '4.0000e-
03', '4.0000e-03', '4.0000e-03', '4.0000e-03', '4.0000e-03', '4.0000e
-03', '4.0000e-03', '4.0000e-03']

Series converged at  k=20

right to left computes exp(-5.5)=4.0000e-03, np.exp(-5.5)=4.086771438
5e-03

The relative error right to left is 0.021232270943118334

Partial sums again converge sooner, the final result is closer to per
cise value and relative error is thus reduced
```

In [8]:

```
#2 (d) iii, adding positive and negative terms seperately
partialsums_ns = ['{:.4e}'.format(1.)]
prevsum=1. #keep track of previous value of sum
n_converge_ns=None
for n in range(1,k+1):
    tempsumev=0. #do < 0 and >0 sums seperately
    tempsumodd=0.
    for ind in range(n+1):
        if float(series_n[ind])<0: #if odd, term is less than zero, a
dd to odd sum
            tempsumodd=round_5_sigfigs(tempsumodd+float(series_n[ind
]))
        else:
            tempsumev=round_5_sigfigs(tempsumev+float(series_n[ind]))

    currentsum =round_5_sigfigs(tempsumodd+tempsumev)
    partialsums_ns.append('{:.4e}'.format(currentsum))
    if prevsum == currentsum and n_converge_ns is None:
        n_converge_ns = n #starts from zero

    prevsum = currentsum #update previous sum

print('Partial sums:\n {} \n'.format(partialsums_ns))
print('Series converged at   k={} \n'.format(n_converge_ns))
percise_exp=np.exp(-5.5)
print('This method computes exp(-5.5)={}, np.exp(-5.5)={:.10e} \n'.fo
rmat(
    partialsums_ns[n_converge_ns],percise_exp ))
rel_error=np.abs(percise_exp-float(partialsums_ns[n_converge_ns]))/pe
rcise_exp
print('The relative error is {} \n'.format(rel_error))
```

```
Partial sums:
 ['1.0000e+00', '-4.5000e+00', '1.0625e+01', '-1.7105e+01', '2.1024e+
01', '-2.0918e+01', '1.7529e+01', '-1.2679e+01', '8.0900e+00', '-4.60
00e+00', '2.3800e+00', '-1.1100e+00', '4.9000e-01', '-1.9000e-01',
'8.0000e-02', '-2.0000e-02', '1.0000e-02', '0.0000e+00', '0.0000e+0
0', '0.0000e+00', '0.0000e+00', '0.0000e+00', '0.0000e+00', '0.0000e+
00', '0.0000e+00', '0.0000e+00', '0.0000e+00', '0.0000e+00', '0.0000e
+00', '0.0000e+00', '0.0000e+00']

Series converged at   k=18

This method computes exp(-5.5)=0.0000e+00, np.exp(-5.5)=4.0867714385e
-03

The relative error is 1.0
```

In [9]:
```
#2 (d) iv, adding positive and negative terms seperately and going ba
ckwards
partialsums_nsb = ['{:.4e}'.format(1.)]
prevsum=1. #keep track of previous value of sum
n_converge_nsb=None
for n in range(1,k+1):
    tempsumev=0. #do < 0 and >0 sums seperately
    tempsumodd=0.
    for ind in range(n, -1, -1):
        if float(series_n[ind])<0: #if odd, term is less than zero, a
dd to odd sum
            tempsumodd=round_5_sigfigs(tempsumodd+float(series_n[ind
]))
        else:
            tempsumev=round_5_sigfigs(tempsumev+float(series_n[ind]))

    currentsum =round_5_sigfigs(tempsumodd+tempsumev)
    partialsums_nsb.append('{:.4e}'.format(currentsum))
    if prevsum == currentsum and n_converge_nsb is None:
        n_converge_nsb = n #starts from zero

    prevsum = currentsum #update previous sum

print('Partial sums:\n {} \n'.format(partialsums_nsb))
print('Series converged at   k={} \n'.format(n_converge_nsb))
percise_exp=np.exp(-5.5)
print('This method computes exp(-5.5)={}, np.exp(-5.5)={:.10e} \n'.fo
rmat(
    partialsums_nsb[n_converge_nsb],percise_exp ))
rel_error=np.abs(percise_exp-float(partialsums_nsb[n_converge_nsb]))/
percise_exp
print('The relative error is {} \n'.format(rel_error))
```

```
Partial sums:
 ['1.0000e+00', '-4.5000e+00', '1.0625e+01', '-1.7105e+01', '2.1024e+
01', '-2.0918e+01', '1.7529e+01', '-1.2679e+01', '8.0900e+00', '-4.60
00e+00', '2.3700e+00', '-1.1200e+00', '4.9000e-01', '-1.9000e-01',
'7.0000e-02', '-3.0000e-02', '0.0000e+00', '-1.0000e-02', '1.0000e-0
2', '1.0000e-02', '1.0000e-02', '1.0000e-02', '1.0000e-02', '1.0000e-
02', '1.0000e-02', '1.0000e-02', '1.0000e-02', '1.0000e-02', '1.0000e
-02', '1.0000e-02', '1.0000e-02']

Series converged at   k=19

This method computes exp(-5.5)=1.0000e-02, np.exp(-5.5)=4.0867714385e
-03

The relative error is 1.4469193226422041
```

2(d) closing notes: method iii converged most quickly but failed, giving $e^{-5.5} = 0$. method ii, adding from right to left, had the lowest error. This is consistent with what was seen with $e^{+5.5}$.

2(e) It is more accurate to simply invert the answer from 2(c). In particular, compute $e^{5.5}$ adding terms in the partial sum from right to left, then computing $e^{-5.5} = 1/e^{-5.5}$. This is shown below

In [10]:
```
#2(e)
#e^5.5 as calculated summing right to left, as in 2(c)
exp5p5_est=float(partialsums_b[n_converge_b])

exp_est=round_5_sigfigs(1./exp5p5_est)

percise_exp=np.exp(-5.5)
print('This method computes exp(-5.5)={:.4e}, np.exp(-5.5)={:.10e} \n
'.format(
    exp_est,percise_exp ))
rel_error=np.abs(percise_exp-exp_est)/percise_exp
print('The relative error is {}, which is much better than all previo
us methods\n'.format(rel_error))
```

```
This method computes exp(-5.5)=4.0868e-03, np.exp(-5.5)=4.0867714385e
-03

The relative error is 6.988777415931331e-06, which is much better tha
n all previous methods
```