ass3q1

February 17, 2018

In [1]:
```
# load libraries and set plot parameters
from pylab import *
from scipy.integrate import quad
%matplotlib inline

from IPython.display import set_matplotlib_formats
set_matplotlib_formats('pdf', 'png')
plt.rcParams['savefig.dpi'] = 75

plt.rcParams['figure.autolayout'] = False
plt.rcParams['figure.figsize'] = 10, 6
plt.rcParams['axes.labelsize'] = 18
plt.rcParams['axes.titlesize'] = 20
plt.rcParams['font.size'] = 16
plt.rcParams['lines.linewidth'] = 2.0
plt.rcParams['lines.markersize'] = 4
plt.rcParams['legend.fontsize'] = 14
plt.rcParams['legend.numpoints'] = 2
plt.rcParams['legend.loc'] = 'best'
plt.rcParams['legend.fancybox'] = True
plt.rcParams['legend.shadow'] = True
plt.rcParams['text.usetex'] = True
plt.rcParams['font.family'] = "serif"
plt.rcParams['font.serif'] = "cm"
plt.rcParams['text.latex.preamble'] = r"\usepackage{subdepth}, \usepackage{type1cm}"
```

1 Question 1

- Define Python functions for the two functions e^x and $cos(cos(x))$ which return a vector (or scalar) value.

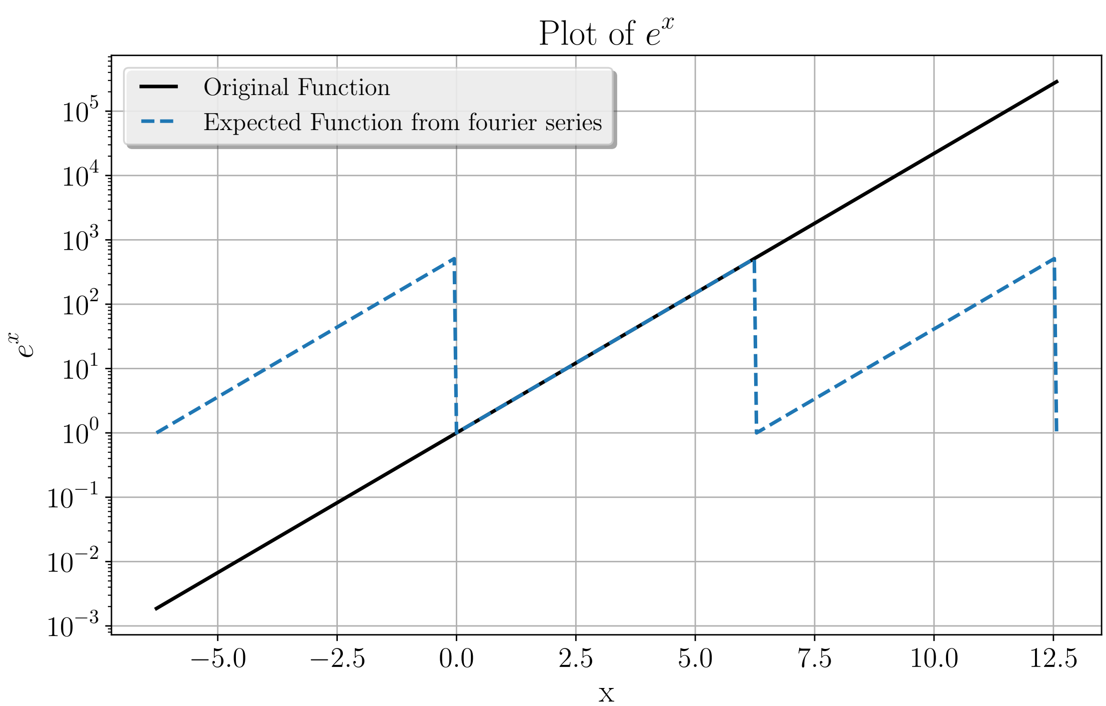
- Plot the functions over the interval $[2\pi, 4\pi)$.
- Discuss periodicity of both functions
- Plot the expected functions from fourier series

Functions for e^x and $cos(cos(x))$ is defined

```
In [2]: def fexp(x):
            return exp(x)

        def fcoscos(x):
            return cos(cos(x))
```

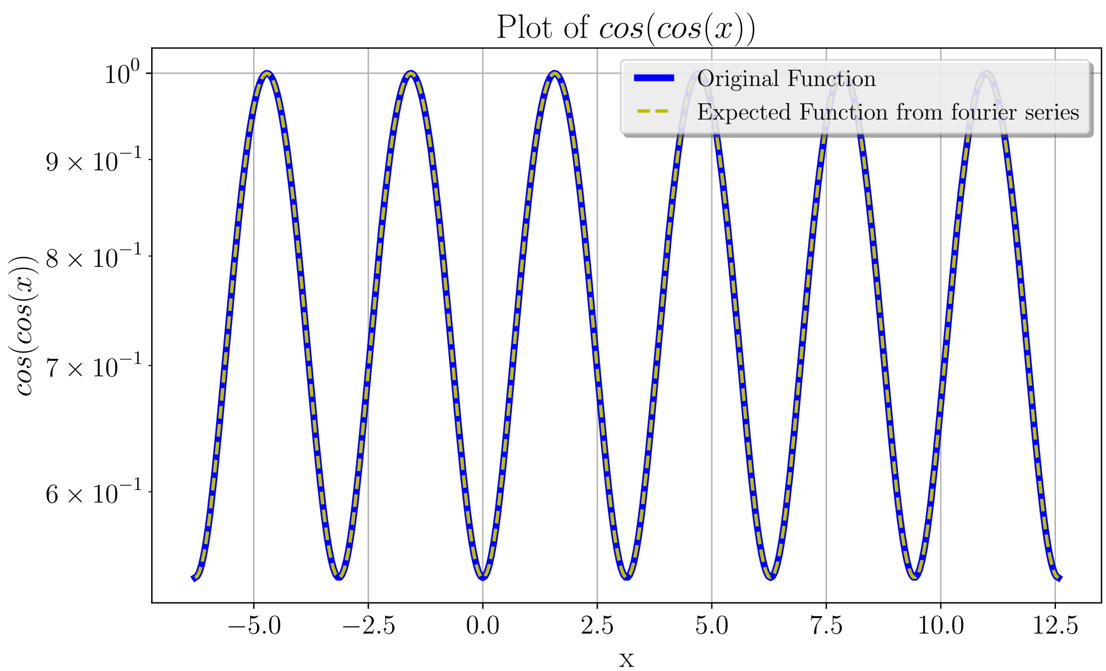
```
In [3]: x = linspace(-2*pi, 4*pi,400)
        period = 2*pi

        exp_fn = fexp(x)
        cos_fn = fcoscos(x)
```

```
In [4]: fig1 = figure()
        ax1 = fig1.add_subplot(111)
        ax1.semilogy(x,exp_fn,'k',label="Original Function")
        ax1.semilogy(x,fexp(x%period),'--',label="Expected Function from fourier series")
        ax1.legend()
        title("Plot of $e^{x}$")
        xlabel("x")
        ylabel("$e^{x}$")
        grid()
        savefig("Figure1.jpg")
```




```
In [5]: fig2 = figure()
        ax2 = fig2.add_subplot(111)
```

```
ax2.plot(x,cos_fn,'b',linewidth=4,label="Original Function")
ax2.semilogy(x,fcoscos(x%period),'y--',label="Expected Function from fourier series")
ax2.legend(loc='upper right')
title("Plot of $cos(cos(x))$")
xlabel("x")
ylabel("$cos(cos(x))$")
grid()
savefig("Figure2.jpg")
show()
```



2 Question 2

Obtain the first 51 coefficients for the two functions

```
In [6]: def createAmatrix(nrow,ncol,x):
            A = zeros((nrow,ncol)) # allocate space for A
            A[:,0]=1 # col 1 is all ones
            for k in range(1,26):
                A[:,2*k-1]=cos(k*x) # cos(kx) column
                A[:,2*k]=sin(k*x) # sin(kx) column
            #endfor
            return A

In [7]: def fourier_an(x,k,f):
            return f(x)*cos(k*x)
```

```
def fourier_bn(x,k,f):
    return f(x)*sin(k*x)
```

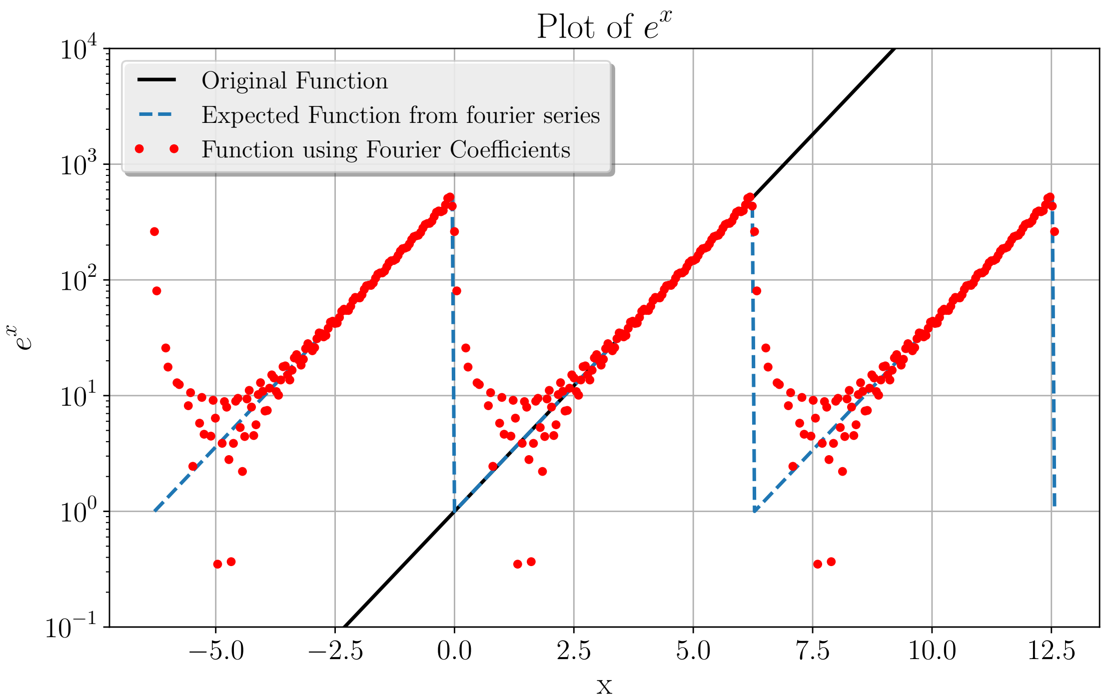
In [8]:
```
def find_coeff(f):

    coeff = []
    coeff.append((quad(f,0,2*pi)[0])/(2*pi))
    for i in range(1,26):
        coeff.append((quad(fourier_an,0,2*pi,args=(i,f))[0])/pi)
        coeff.append((quad(fourier_bn,0,2*pi,args=(i,f))[0])/pi)

    return coeff
```

In [9]:
```
def computeFunctionfromCoeff(c):
    A = createAmatrix(400,51,x)
    f_fourier = A.dot(c)
    return f_fourier
```

In [10]:
```
exp_coeff = []
coscos_coeff = []

exp_coeff1 = find_coeff(fexp)
coscos_coeff1 = find_coeff(fcoscos)

exp_coeff = np.abs(exp_coeff1)
coscos_coeff = np.abs(coscos_coeff1)

fexp_fourier = computeFunctionfromCoeff(exp_coeff1)
fcoscos_fourier = computeFunctionfromCoeff(coscos_coeff1)
```

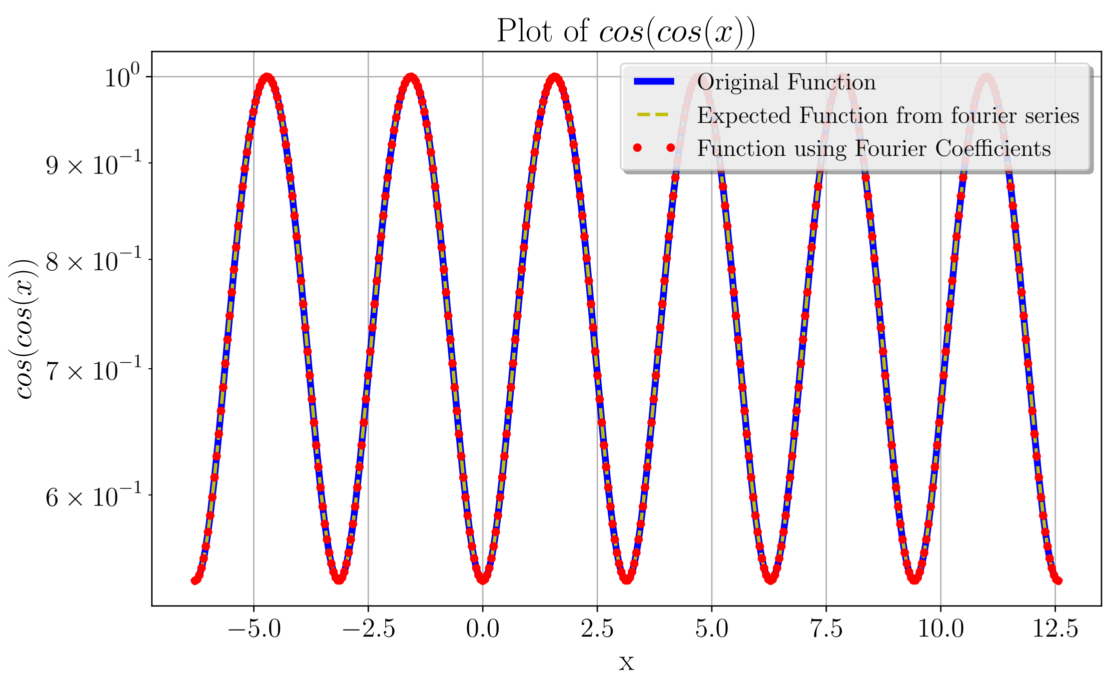
In [11]:
```
ax1.semilogy(x,fexp_fourier,'ro',label = "Function using Fourier Coefficients")
ax1.set_ylim([pow(10,-1),pow(10,4)])
ax1.legend()
fig1
```

Out[11]:




```
In [12]: ax2.plot(x,fcoscos_fourier,'ro',label = "Function using Fourier Coefficients")
         ax2.legend(loc='upper right')
         fig2
```

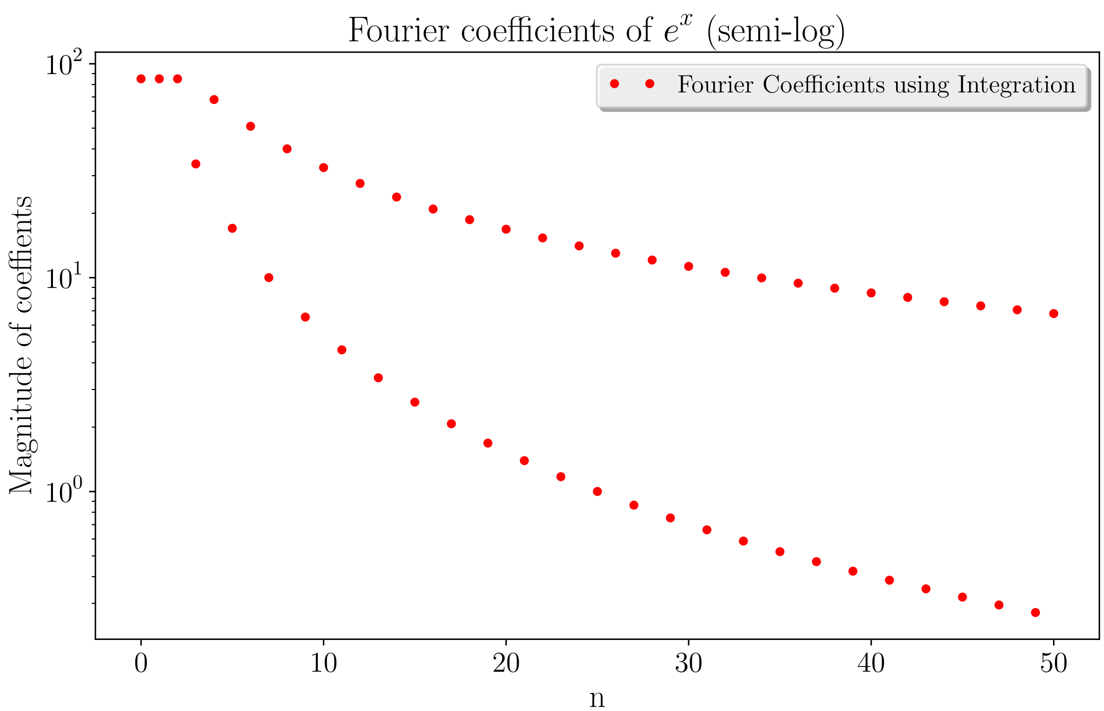
Out[12]:



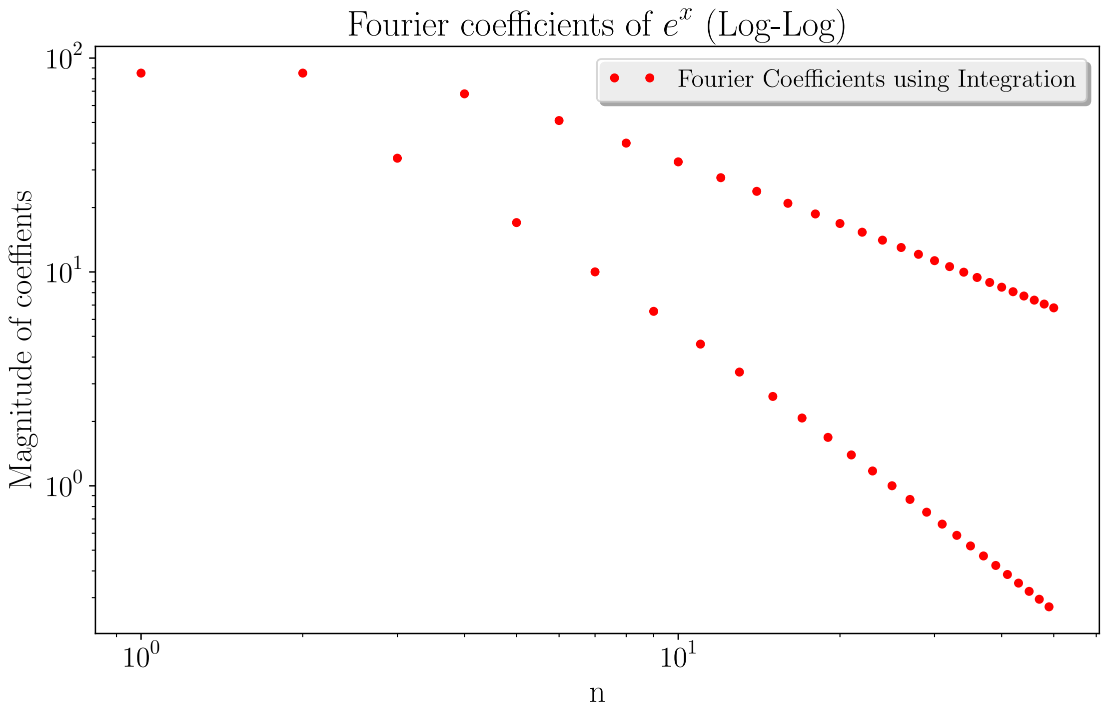
3 Question3

Two different plots using "semilogy" and "loglog" and plot the magnitude of the coefficients vs n

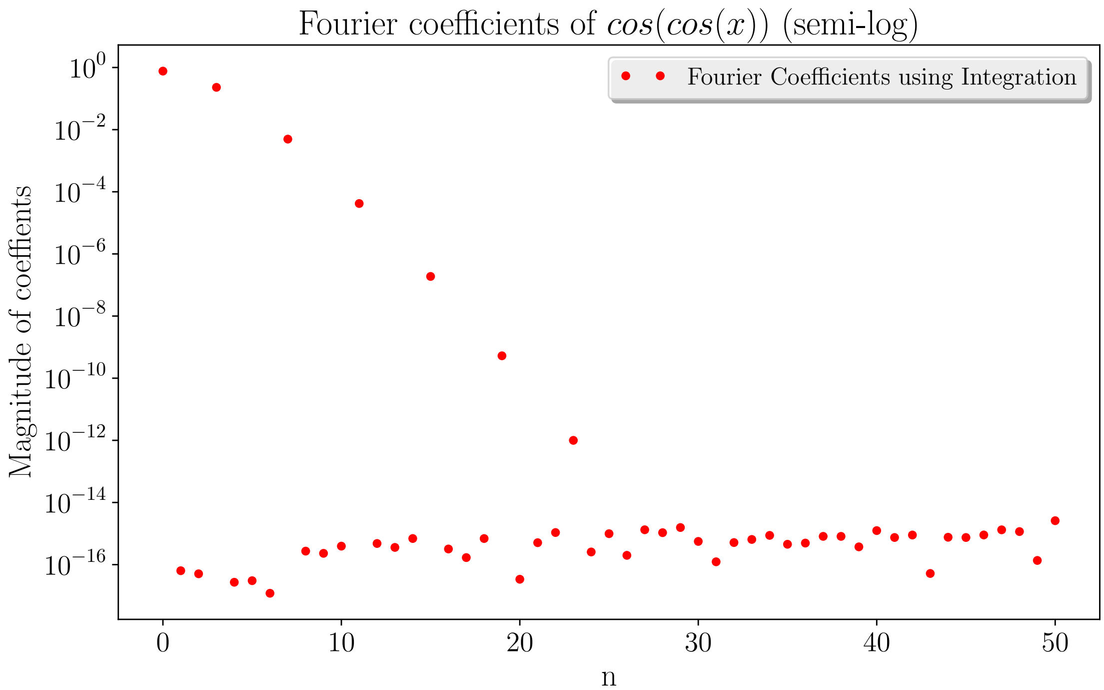
In [13]:
```
fig3 = figure()
ax3 = fig3.add_subplot(111)
ax3.semilogy((exp_coeff),'ro',label = "Fourier Coefficients using Integration")
ax3.legend()
title("Fourier coefficients of $e^{x}$ (semi-log)")
xlabel("n")
ylabel("Magnitude of coeffients")
show()
```



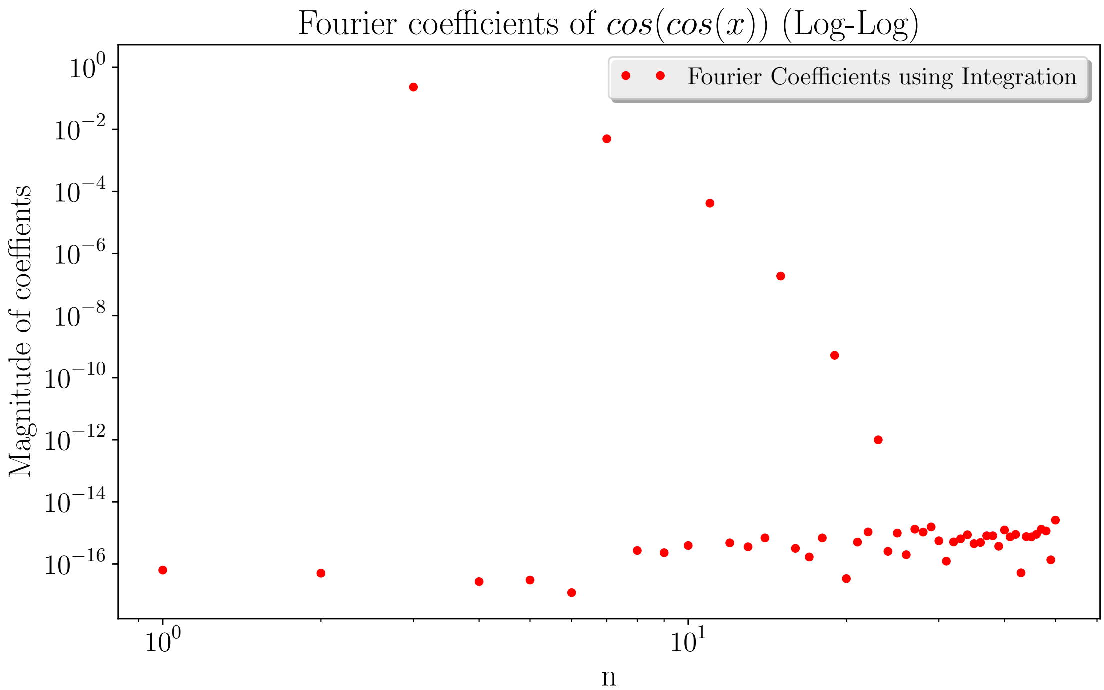
In [14]:
```
fig4 = figure()
ax4 = fig4.add_subplot(111)
ax4.loglog((exp_coeff),'ro',label = "Fourier Coefficients using Integration")
ax4.legend(loc='upper right')
title("Fourier coefficients of $e^{x}$ (Log-Log)")
xlabel("n")
ylabel("Magnitude of coeffients")
show()
```



In [15]:
```
fig5 = figure()
ax5 = fig5.add_subplot(111)
ax5.semilogy((coscos_coeff),'ro',label = "Fourier Coefficients using Integration")
ax5.legend(loc='upper right')
title("Fourier coefficients of $cos(cos(x))$ (semi-log)")
xlabel("n")
ylabel("Magnitude of coeffients")
show()
```




```
In [16]: fig6 = figure()
         ax6 = fig6.add_subplot(111)
         ax6.loglog((coscos_coeff),'ro',label = "Fourier Coefficients using Integration")
         ax6.legend(loc='upper right')
         title("Fourier coefficients of $cos(cos(x))$  (Log-Log)")
         xlabel("n")
         ylabel("Magnitude of coeffients")
         show()
```



4 Question 4 & 5

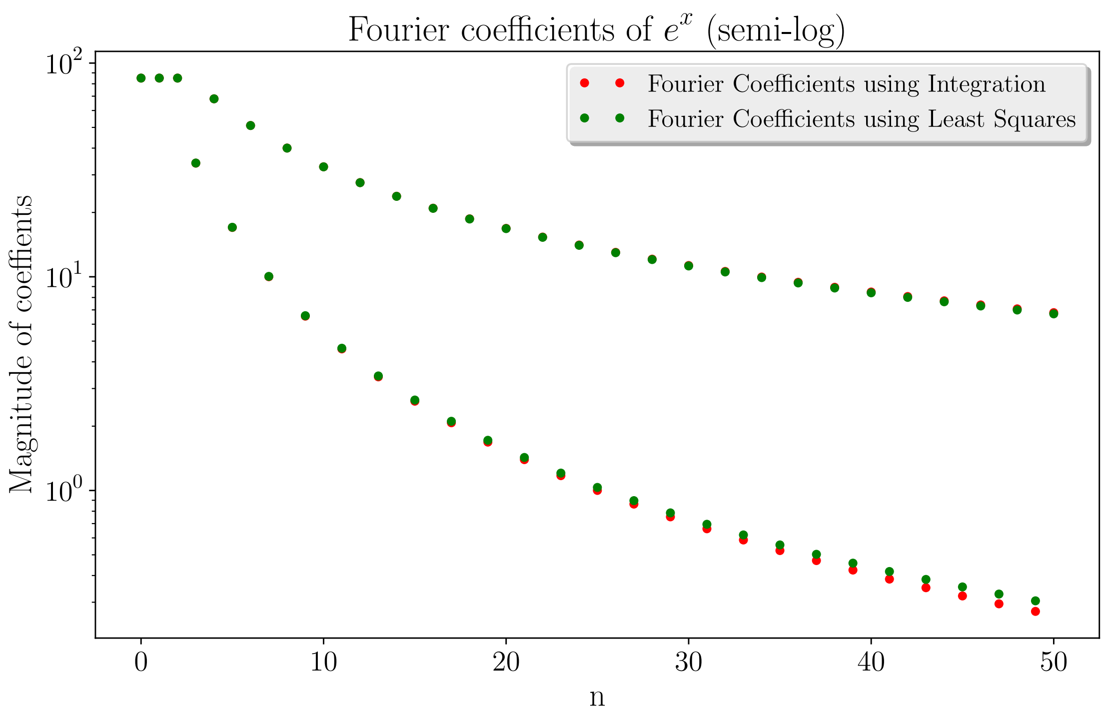
Uses least square method approach to find the fourier coefficients

```
In [17]: def getCoeffByLeastSq(f,low_lim,upp_lim,no_points):
             x1 = linspace(low_lim,upp_lim,no_points)
             b = f(x1)
             A = createAmatrix(400,51,x1)
             c = []
             c=lstsq(A,b)[0] # the [0] is to pull out the
             # best fit vector. lstsq returns a list.
             return c
```

```
In [18]: coeff_exp = getCoeffByLeastSq(fexp,0,2*pi,400)
         coeff_coscos = getCoeffByLeastSq(fcoscos,0,2*pi,400)

         c1 = np.abs(coeff_exp)
         c2 = np.abs(coeff_coscos)
```

```
In [19]: ax3.semilogy(c1,'go',label = "Fourier Coefficients using Least Squares")
         ax3.legend(loc='upper right')
         fig3
```

Out[19]:




```
In [ ]: legend?
```

```
In [ ]: ax4.loglog(c1,'go',label = "Fourier Coefficients using Least Squares")
        ax4.legend(loc=0)
        fig4
```

```
In [ ]: ax5.semilogy(c2,'go',label = "Fourier Coefficients using Least Squares")
        ax5.legend(loc='upper right')
        fig5
```

```
In [ ]: ax6.loglog(c2,'go',label = "Fourier Coefficients using Least Squares")
        ax6.legend()
        fig6
```

5 Question 6

Comparing the answers got by least squares and by the direct integration.And finding deviation between them and find the largest deviation using Vectors

```
In [ ]: def compareCoeff(f):
            deviations = []
            max_dev = 0
            if(f==1):
                deviations = np.abs(exp_coeff1 - coeff_exp)
```

```
        elif(f==2):
            deviations = np.abs(coscos_coeff1 - coeff_coscos)

        max_dev = np.amax(deviations)
        return deviations,max_dev
```

```
In [ ]: dev1,maxdev1 = compareCoeff(1)
        dev2,maxdev2 = compareCoeff(2)

        plot(dev1,'g')
        title("Deviation between Coefficients by lstsq & by Integration method for $e^{x}$")
        xlabel("n")
        ylabel("Magnitude of Deviations")
        show()
```

```
In [ ]: plot(dev2,'g')
        title("Lstsq Vs Integration method (Deviation) for $cos(cos(x))$")
        xlabel("n")
        ylabel("Magnitude of Deviations")
        show()
```

6 Question 7

- Computing Ac i.e multiplying Matrix A and Vector C from the estimated values of Coeffient Vector C by Least Squares Method.
- To Plot them (with green circles) in Figures 1 and 2 respectively for the two functions.

```
In [ ]: x1 = linspace(0,2*pi,400)
```

```
In [ ]: def computeFunctionbyLeastSq(c):
            f_lstsq = []
            A = createAmatrix(400,51,x1)
            f_lstsq = A.dot(c)
            return f_lstsq
```

```
In [ ]: fexp_lstsq = computeFunctionbyLeastSq(coeff_exp)
        fcoscos_lstsq = computeFunctionbyLeastSq(coeff_coscos)

        ax1.semilogy(x1,fexp_lstsq,'go',
                     label = "Inverse Fourier Transform From Least Squares")
        ax1.legend()
        ax1.set_ylim([pow(10,-2),pow(10,5)])
        ax1.set_xlim([0,2*pi])
        fig1
```

```
In [ ]: ax2.plot(x1,fcoscos_lstsq,'go',markersize=4,
                 label = "Inverse Fourier Transform From Least Squares")
        ax2.set_ylim([0.5,1.3])
```

```
ax2.set_xlim([0,2*pi])
ax2.legend()
fig2
```