In [1]:
```
%matplotlib notebook

from matplotlib.pylab import *
from numpy import *
```

Homework #7

Problem #1 (TB 11.2, page

Find the best-fit coefficients $\mathbf{c} = (c_0, c_1, c_2)$ so that

$$g(x) = c_0 e^x + c_1 \sin(x) + c_2 \Gamma(x) \text{ "} \tag{1}$$

approximates the function $f(x) = 1/x$ as closely as possible. To find $\mathbf{c}$, find $\overline{\mathbf{c}}$ that minimizes the function

$$\mathbf{F}(c) = \| f(\mathbf{x}) - g(\mathbf{x}) \|_2^2 \tag{2}$$

over an interval $x \in [a, b]$.

Task

We solve this problem by discretizing the interval $[a, b]$ using discrete nodes $\mathbf{x} = [x_0, x_1, \ldots x_{m-1}]$. We then solve the system of equations

$$c_0 e^{x_i} + c_1 \sin(x_i) + c_2 \Gamma(x_i) = f(x_i), \qquad i = 0, 1, \ldots m-1 \tag{3}$$

This will in general be an overdetermined system that can be expressed as the linear system

$$A\mathbf{c} = \mathbf{F} \tag{4}$$

where $A \in \mathbb{R}^{m \times 3}$ and $\mathbf{F} \in \mathbb{R}^{m \times 1}$.

- Solve this resulting linear system using Algorithm 11.2 in TB (page 83).
- Choose m so that the relative norm satisfies

$$\frac{\|\mathbf{f} - \mathbf{g}\|_2}{\|\mathbf{f}\|_2} \leq 10^{-2} \tag{5}$$

where vectors $\mathbf{f}$ and $\mathbf{g}$ are defined as $\mathbf{f} = f(\mathbf{x})$ and $\mathbf{g} = g(\mathbf{x})$. Your value of m should not be very large.

- Display the coefficients $\mathbf{c} = (c_0, c_1, c_2)$ you obtain for each interval (a, b).
- Plot the function $f(x)$ and your approximation $g(x)$ on the same graph.

Hints

- Use `linspace` to construct discrete points $x_i, i = 0, 1, \ldots m-1$.
- Use the SciPy function `scipy.special.gamma` to get the function $\Gamma(x)$.
- Use the QR function provided below.
- For the interval (0,1), you can use an interval $(\varepsilon, 1)$, where $\varepsilon \ll 1$.
- Use the notebook "Hmwk7_exact" to compare the coefficients you get to the exact solution computed using SymPy.

In [2]:
```
def display_mat(msg,A):
    print(msg)
    fstr = {'float' : "{:>16.8f}".format}
    with printoptions(formatter=fstr):
        display(A)
    print("")
```

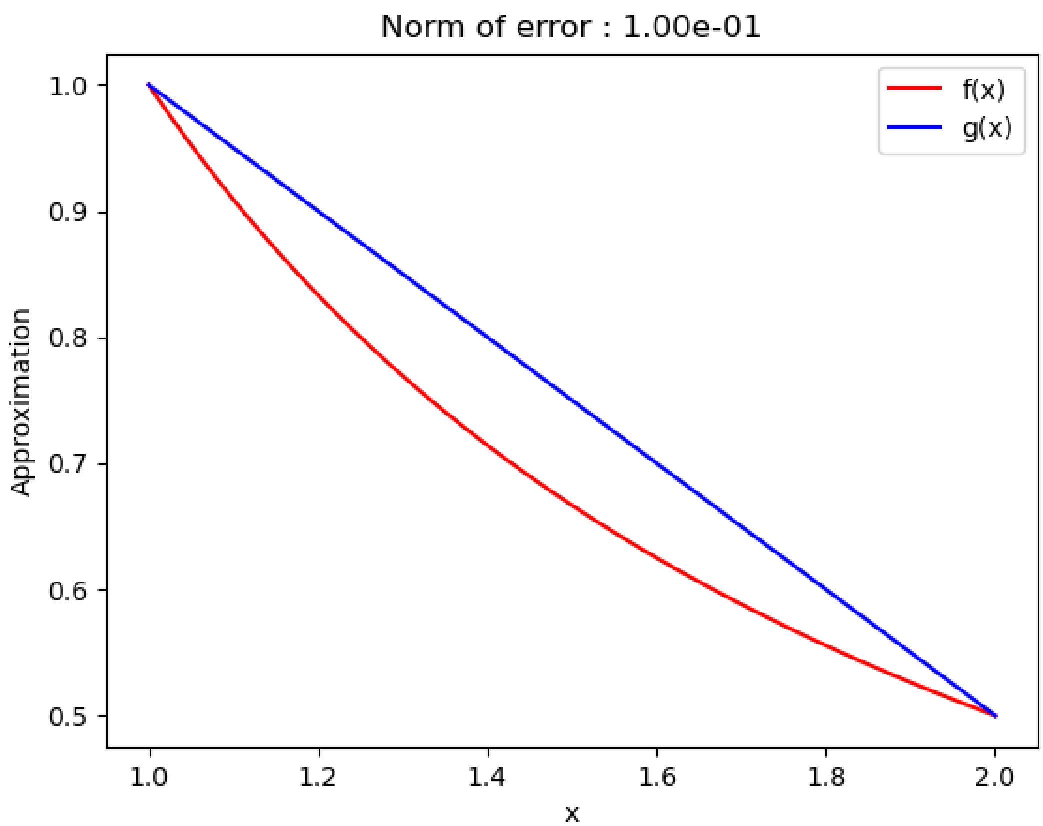
In [3]:
```
# Use this QR algorithm, or you may use your own from Homework #6

def QR_House(A):
    m,n = A.shape

    R = A.copy()
    Qt = eye(m)
    p = min([m,n])
    for j in range(p):
        x = R[j:m,j:j+1]
        I = eye(m-j)
        s = 1 if x[0] >= 0 else -1      # sign function, with sign(0)  = 1
        v = s*linalg.norm(x,2)*I[:,0:1] + x
        v = v/linalg.norm(v,2)
        F = I - 2*(v@v.T)
        R[j:m,j:n] = F@R[j:m,j:n]
        Qt[j:m,:] = F@Qt[j:m,:]    # Solution to Homework #6 !!!

    Q = Q.T
    return Q, R
```

In [4]:
```
from scipy.special import gamma

def f(x):
    return 1/x

# Define g(x)
```

Problem 1a

Compute the coefficients $\mathbf{c}$ for the interval (a,b).

In [7]:
```
figure(1)
clf()

# Solve on interval (1,2)

a = 1
b = 2
x = linspace(a,b,200)

plot(x,f(x),'r',label='f(x)')

# TODO : Set up linear system to be solved.  Choose 'm' to get desired error

# TODO : Compute QR decomposition of A

# TODO : Solve least squares problem using QR

# TODO : Plot g(x) using coefficients you found above.

plot([1,2],[1,0.5],'b',label='g(x)')     # Not a good approximation!

# TODO : Compute relative norm

rel_norm = 0.1    # TODO : Compute relative norm here
str = "Norm of error : {:.2e}".format(rel_norm)
xlabel('x')
ylabel('Approximation')
title(str)

legend()

show()
```



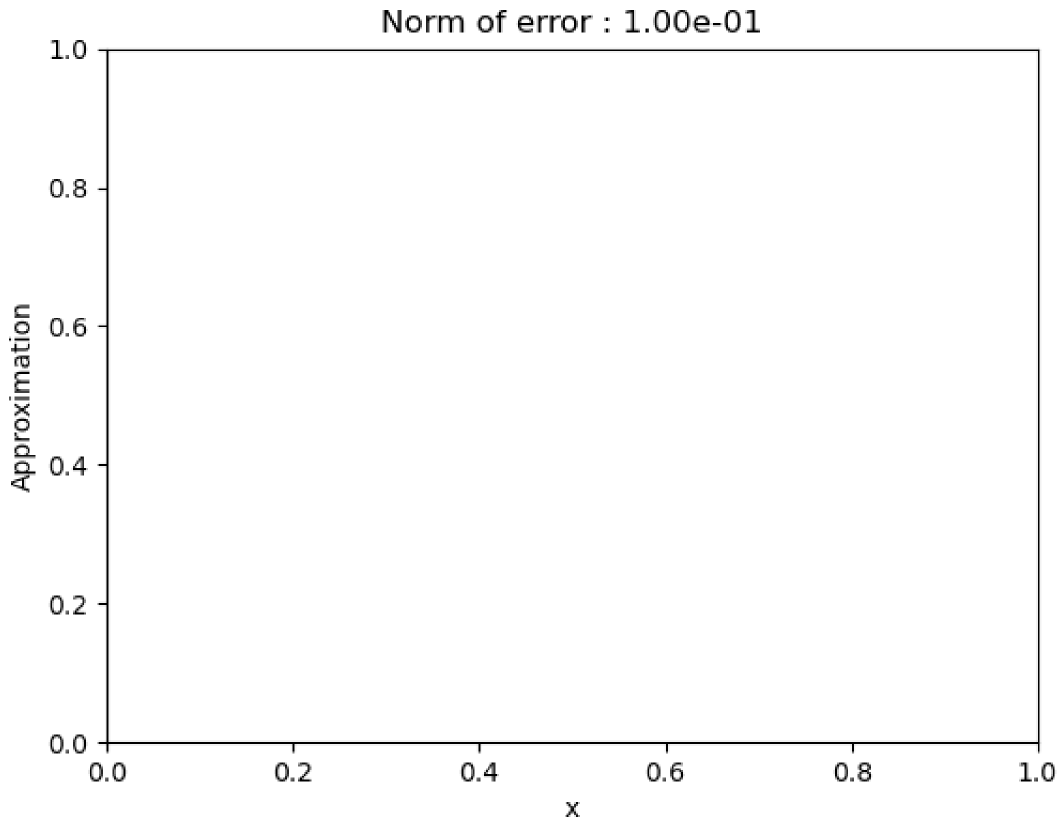
Problem 1b

Compute the coefficients $\mathbf{c}$ for the interval $(0, 1)$.

Question :

Is there one function used to define $g(x)$ that seems to be the closest match to $f(x)$?

In [11]:
```
# Repeat the above for interval (0,1)

figure(2)
clf()

# TODO : Your work goes here


rel_norm = 0.1     # TODO : Compute relative norm here
str = "Norm of error : {:.2e}".format(rel_norm)
xlabel('x')
ylabel('Approximation')
title(str)

# legend()

show()
```


Norm of error : 1.00e-01
1.0
0.8
0.6
0.4
0.2
0.0
Approximation
0.0
0.2
0.4
0.6
0.8
1.0
x

In [ ]:

```

```