2. Linear Harmonic Oscillator

Program:

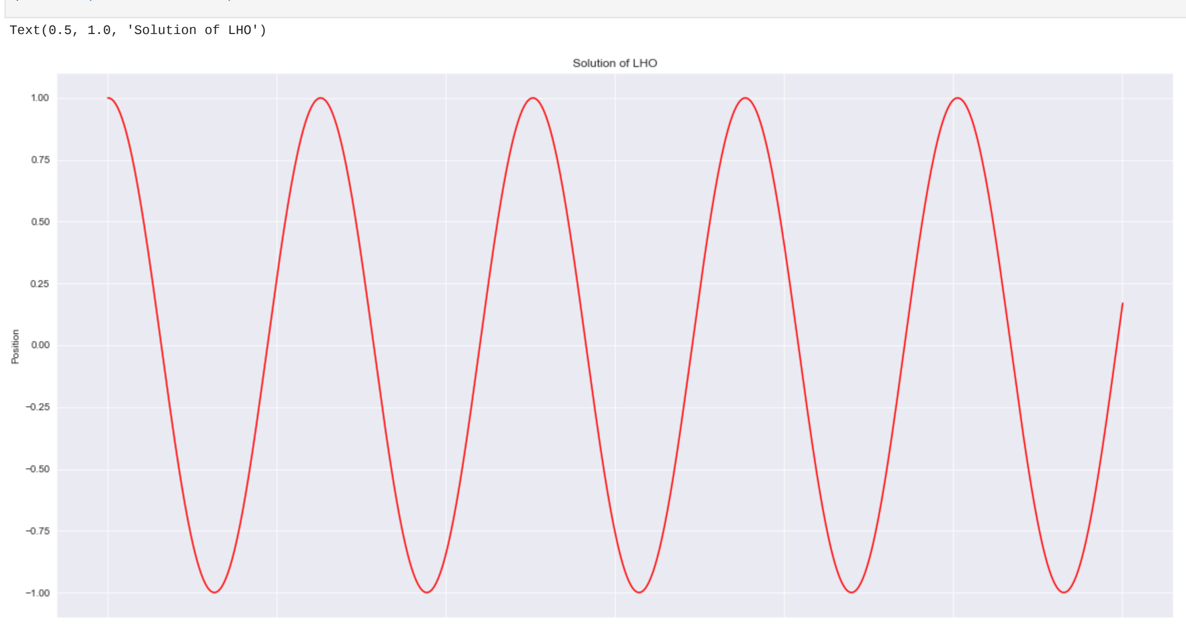
In [1]:
```
import matplotlib.pyplot as plt
import seaborn as sns
import math as mth
sns.set_style('darkgrid')
```

In [ ]:
```
def f(t,x,v):
    return -w0**2*x

w0 = 1
x0 = 1.0
v0 = 0.0
ti = 0.0
tf = 30
n = 1000
h = (tf-ti)/n
t = ti
x = x0
v = v0
x_list = [x0]
v_list = [v0]
t_list = [t]

for i in range(0,n+1):
    print(t,x,v)
    x = x + h*v
    v = v + h* f(t,x,v)
    t = t+h

    x_list.append(x)
    v_list.append(v)
    t_list.append(t)
```

Plotting Solution:

In [16]:
```
plt.plot(t_list, x_list, color='r')
plt.xlabel("Time")
plt.ylabel("Position")
plt.rcParams["figure.figsize"] = (20, 10)
plt.title("Solution of LHO")
```

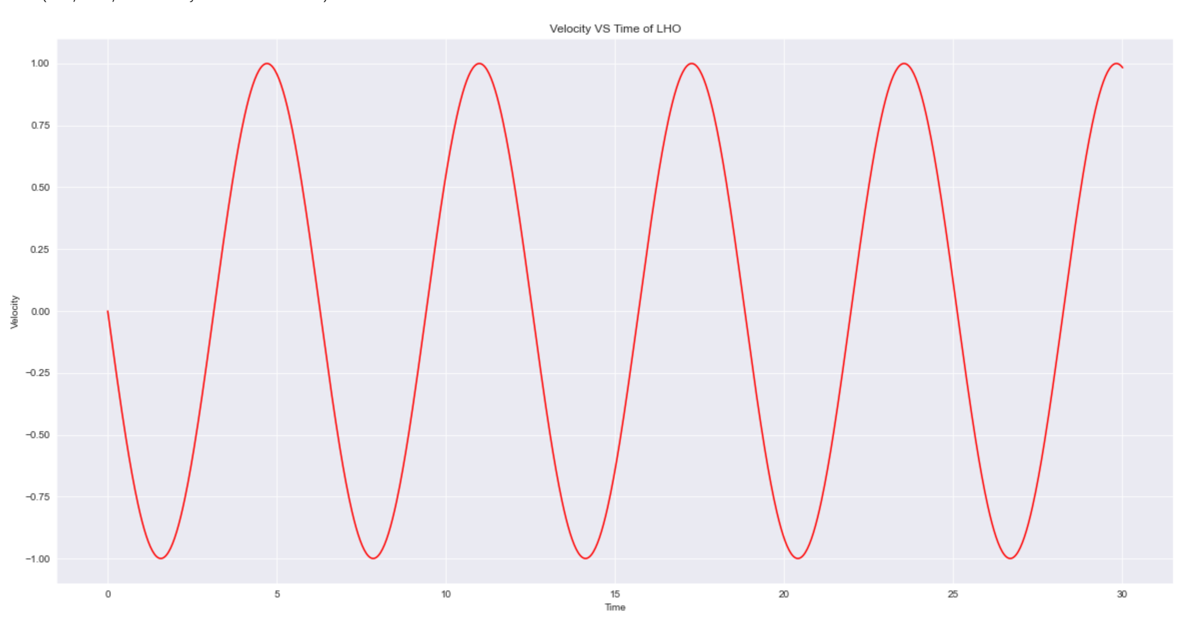
Out[16]:
```
Text(0.5, 1.0, 'Solution of LHO')
```



Plotting Velocity VS Time:

In [4]:
```
plt.plot(t_list, v_list, color = 'r')
plt.xlabel("Time")
plt.ylabel("Velocity")
plt.rcParams["figure.figsize"] = (20, 10)
plt.title("Velocity VS Time of LHO")
```

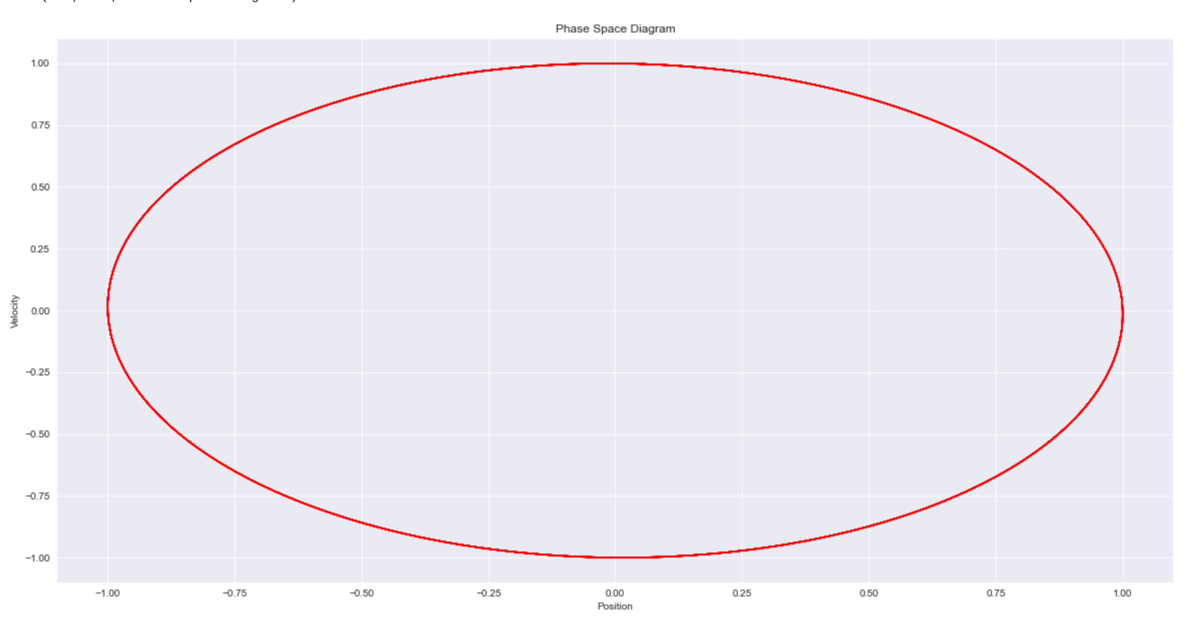
Out[4]:
```
Text(0.5, 1.0, 'Velocity VS Time of LHO')
```



Plotting Phase Space:

In [5]:
```
plt.plot(x_list, v_list, color = 'r')
plt.xlabel("Position")
plt.ylabel("Velocity")
plt.rcParams["figure.figsize"] = (20, 10)
plt.title("Phase Space Diagram")
```

Out[5]:
```
Text(0.5, 1.0, 'Phase Space Diagram')
```



3. Damped Linear Harmonic Oscillator

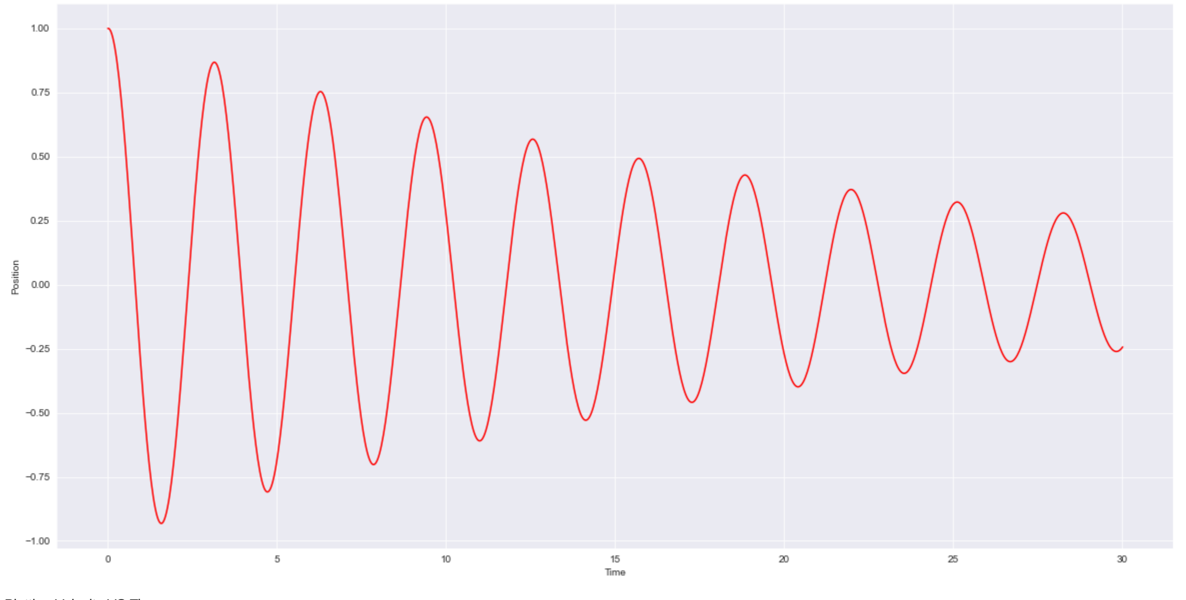
Program:

In [ ]:
```
def f(t,x,v):
    return -w0**2*x - g*v

g = 0.09 # light Damping
w0 = 2
x0 = 1.0
v0 = 0.0
ti = 0.0
tf = 30
n = 1000
h = (tf-ti)/n
t = ti
x = x0
v = v0
xd_list = [x0]
vd_list = [v0]
td_list = [t]

for i in range(0,n+1):
    print(t,x,v)
    x = x + h*v
    v = v + h* f(t,x,v)
    t = t+h

    xd_list.append(x)
    vd_list.append(v)
    td_list.append(t)
```

Plotting Solution:

In [7]:
```
plt.plot(td_list, xd_list, color='r')
plt.xlabel("Time")
plt.ylabel("Position")
plt.rcParams["figure.figsize"] = (20, 10)
plt.title("Solution of Damped LHO")
```

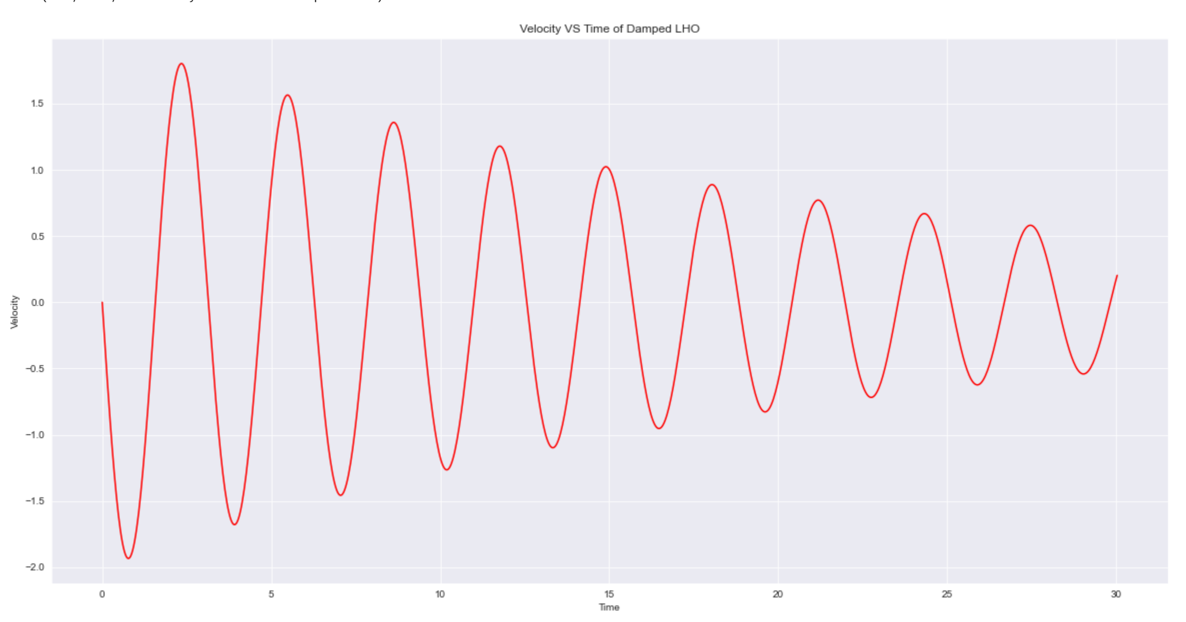
Out[7]:
```
Text(0.5, 1.0, 'Solution of Damped LHO')
```



Plotting Velocity VS Time:

In [8]:
```
plt.plot(td_list, vd_list, color = 'r')
plt.xlabel("Time")
plt.ylabel("Velocity")
plt.rcParams["figure.figsize"] = (20, 10)
plt.title("Velocity VS Time of Damped LHO")
```

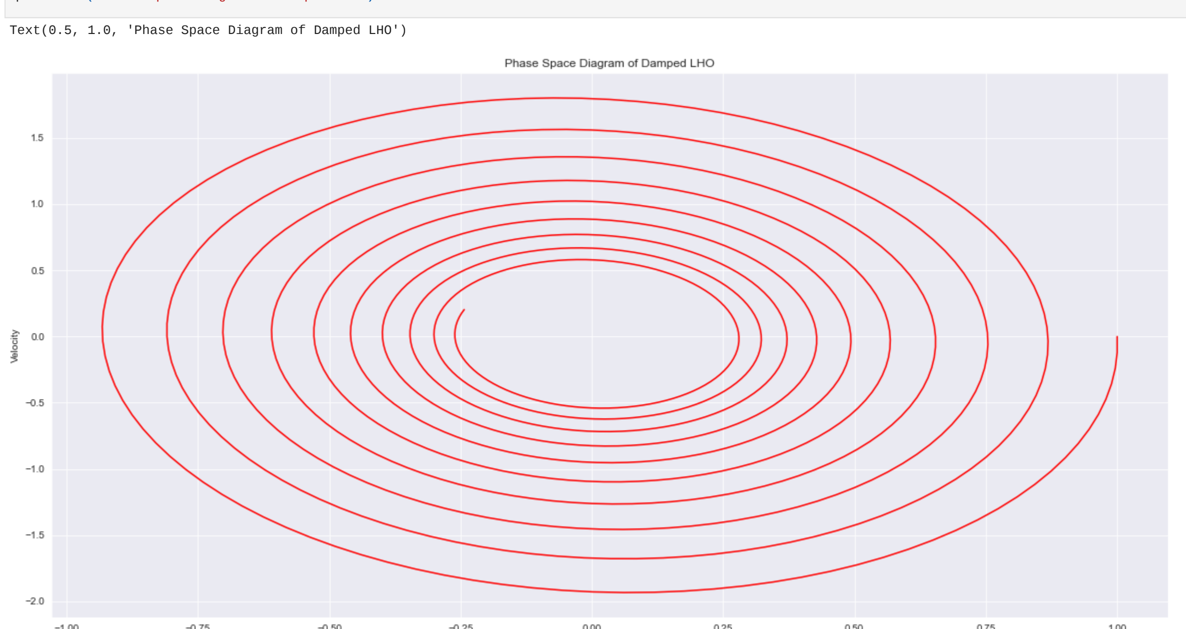
Out[8]:
```
Text(0.5, 1.0, 'Velocity VS Time of Damped LHO')
```



Plotting Phase Space:

In [9]:
```
plt.plot(xd_list, vd_list, color = 'r')
plt.xlabel("Position")
plt.ylabel("Velocity")
plt.rcParams["figure.figsize"] = (20, 10)
plt.title("Phase Space Diagram of Damped LHO")
```

Out[9]:
```
Text(0.5, 1.0, 'Phase Space Diagram of Damped LHO')
```

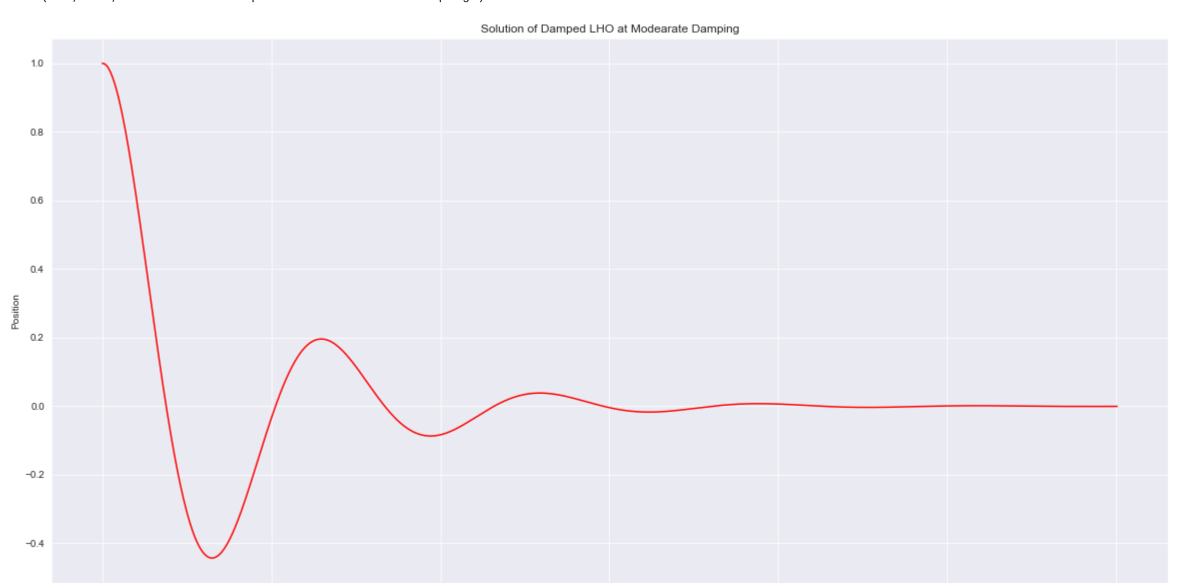


Moderate Damping:

In [ ]:
```
def f(t,x,v):
    return -w0**2*x - g*v

g = 0.5 # Moderate Damping
w0 = 1
x0 = 1.0
v0 = 0.0
ti = 0.0
tf = 30
n = 1000
h = (tf-ti)/n
t = ti
x = x0
v = v0
xd_list = [x0]
vd_list = [v0]
td_list = [t]

for i in range(0,n+1):
    print(t,x,v)
    x = x + h*v
    v = v + h* f(t,x,v)
    t = t+h

    xd_list.append(x)
    vd_list.append(v)
    td_list.append(t)
```

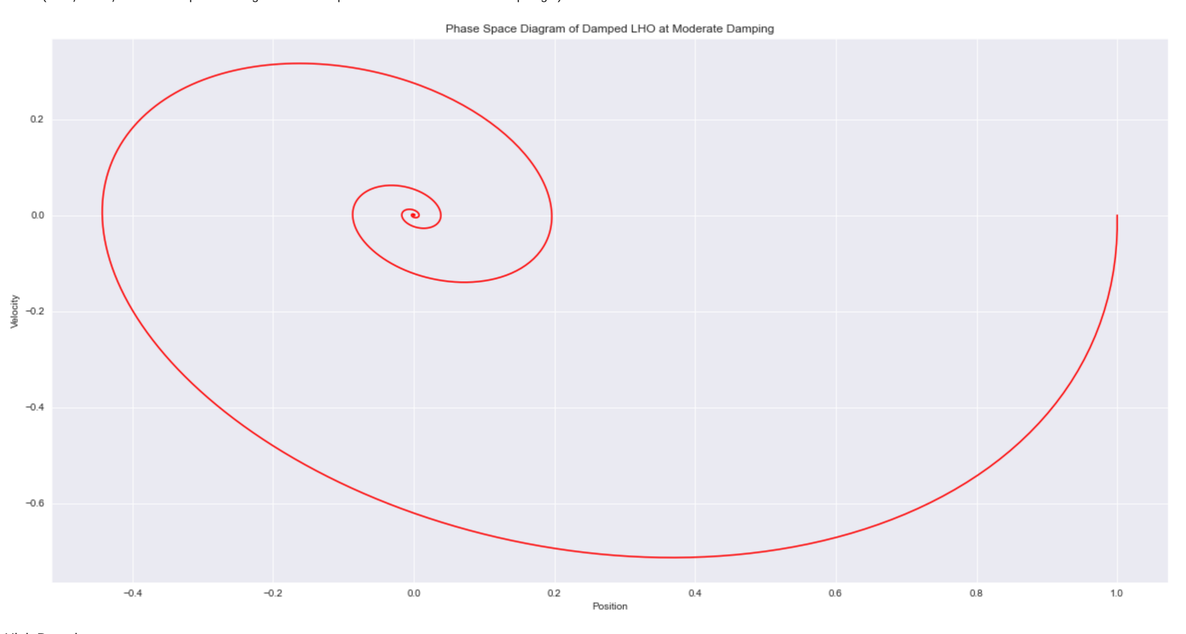
In [11]:
```
plt.plot(td_list, xd_list, color='r')
plt.xlabel("Time")
plt.ylabel("Position")
plt.rcParams["figure.figsize"] = (20, 10)
plt.title("Solution of Damped LHO at Modearate Damping")
```

Out[11]:
```
Text(0.5, 1.0, 'Solution of Damped LHO at Modearate Damping')
```



In [12]:
```
plt.plot(xd_list, vd_list, color = 'r')
plt.xlabel("Position")
plt.ylabel("Velocity")
plt.rcParams["figure.figsize"] = (20, 10)
plt.title("Phase Space Diagram of Damped LHO at Moderate Damping")
```

Out[12]:
```
Text(0.5, 1.0, 'Phase Space Diagram of Damped LHO at Moderate Damping')
```

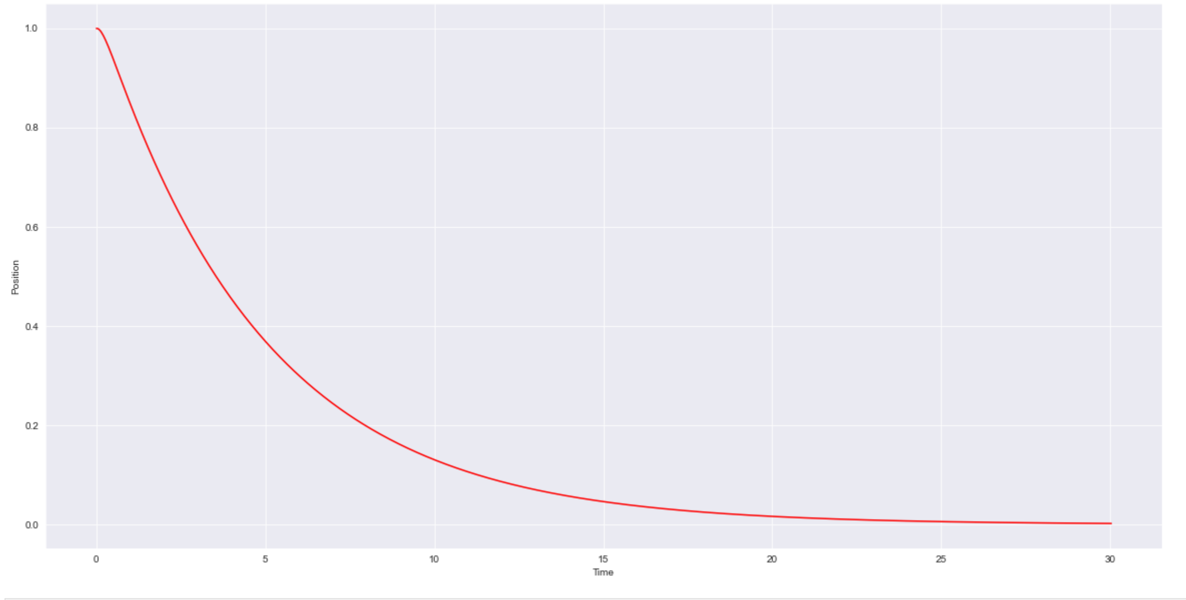


High Damping:

In [ ]:
```
def f(t,x,v,g):
    return -w0**2*x - g*v

g = 5 # High Damping
w0 = 1
x0 = 1.0
v0 = 0.0
ti = 0.0
tf = 30
n = 1000
h = (tf-ti)/n
t = ti
x = x0
v = v0
xd_list = [x0]
vd_list = [v0]
td_list = [t]

for i in range(0,n+1):
    print(t,x,v,g)
    x = x + h*v
    v = v + h* f(t,x,v,g)
    t = t+h

    xd_list.append(x)
    vd_list.append(v)
    td_list.append(t)
```

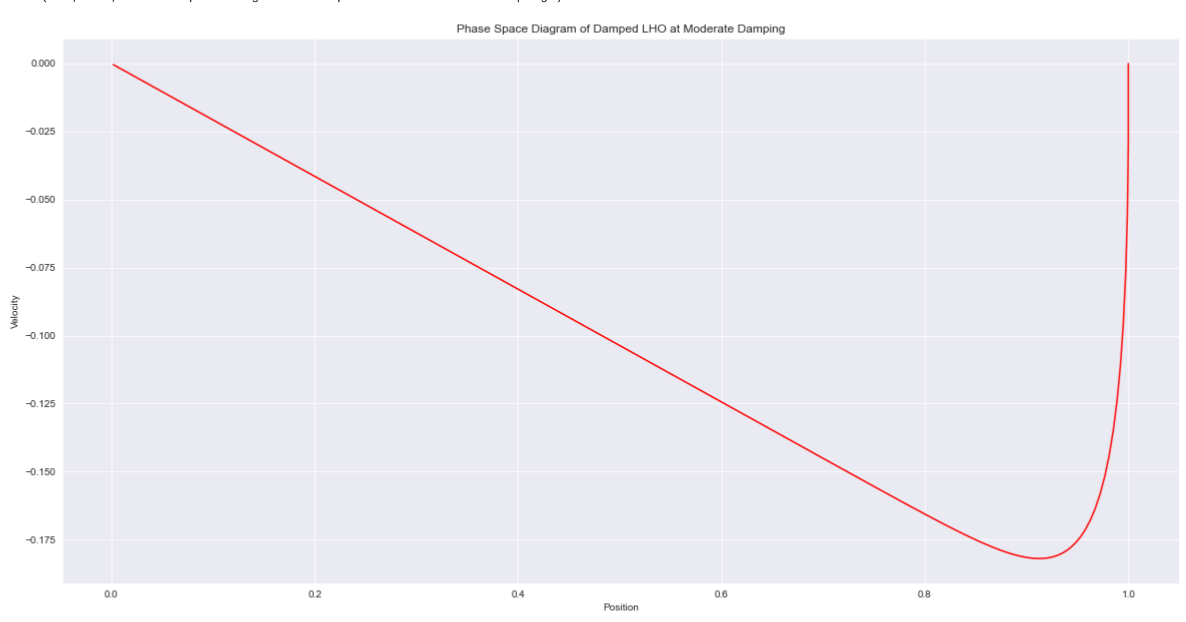
In [14]:
```
plt.plot(td_list, xd_list, color='r')
plt.xlabel("Time")
plt.ylabel("Position")
plt.rcParams["figure.figsize"] = (20, 10)
plt.title("Solution of Damped LHO at High Damping")
```

Out[14]:
```
Text(0.5, 1.0, 'Solution of Damped LHO at High Damping')
```



In [15]:
```
plt.plot(xd_list, vd_list, color = 'r')
plt.xlabel("Position")
plt.ylabel("Velocity")
plt.rcParams["figure.figsize"] = (20, 10)
plt.title("Phase Space Diagram of Damped LHO at Moderate Damping")
```

Out[15]:
```
Text(0.5, 1.0, 'Phase Space Diagram of Damped LHO at Moderate Damping')
```

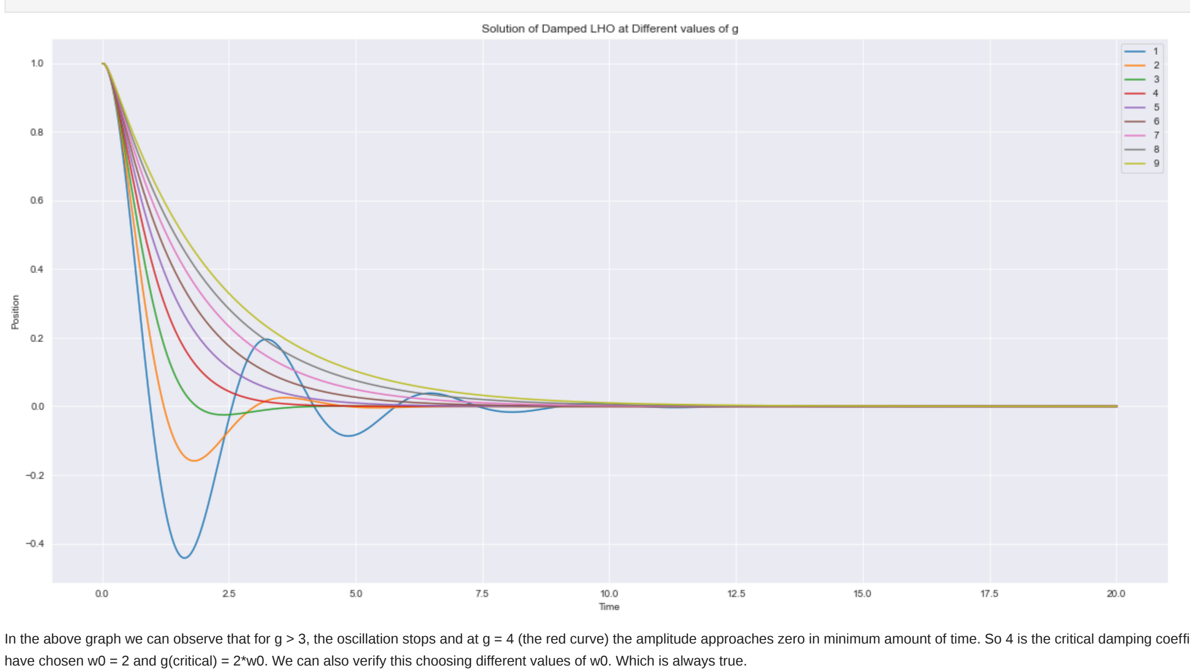


Estimating the Critical Damping Coefficient :

For this we can plot the position vs time graph for different values of damping coefficient and from the graph we can estimate the damping coefficient, which provides the quickest approach to zero amplitude and no oscillation.

program:

In [42]:
```
def f(w,t,x,v,g):
    return -w**2*x - g*v

w0 = 2
x0 = 1.0
v0 = 0.0
ti = 0.0
tf = 20
n = 1000
h = (tf-ti)/n

for g in range(1,10):

    t = ti
    x = x0
    v = v0

    x_list = [x0]
    v_list = [v0]
    t_list = [ti]

    for _ in range(0,n+1):
        #print(t,x,v)
        x = x + h*v
        v = v + h* f(w0,t,x,v,g)
        t = t+h

        x_list.append(x)
        v_list.append(v)
        t_list.append(t)

    plt.plot(t_list, x_list, label = g)
    plt.legend()
    plt.title("Solution of Damped LHO at Different values of g")
    plt.xlabel("Time")
    plt.ylabel("Position")
```



In the above graph we can observe that for g > 3, the oscillation stops and at g = 4 (the red curve) the amplitude approaches zero in minimum amount of time. So 4 is the critical damping coefficient in this case. Note that, we have chosen w0 = 2 and g(critical) = 2*w0. We can also verify this choosing different values of w0. Which is always true.