main

July 4, 2024

[284]:
```python
import numpy as np
import matplotlib.pyplot as plt

def euler(x,y, h, y0, func):
    y[0] = y0
    for i in range(1,len(x)):
        y[i] = y[i-1] + h*func(x[i-1], y[i-1])
        #print(y[i])
    return y

def rk4(x,y,h,y0,func):
    y[0] = y0
    for i in range(1,len(x)):
        k1 = h*func(x[i-1], y[i-1])
        k2 = h*func(x[i-1]+h/2, y[i-1] + k1/2)
        k3 = h*func(x[i-1]+h/2, y[i-1] + k2/2)
        k4 = h*func(x[i-1]+h, y[i-1] + k3)
        y[i] = y[i-1] + (k1 + 2*k2 + 2*k3 + k4)/6
        #print(y[i])
    return y

def func(x,y):
    return -2*y-x**2/4+1/8

def exact(x,y):
    for i in range(0,len(x)):
        y[i] = 1/np.exp(2*x[i]) + x[i]/8 - (x[i]**2)/8
        #print(y[i])
    return y
```

[285]:
```python
h = 0.4
x = np.arange(0.0,4.0+h, h)
y0 = 1.0
y_euler = [0] * len(x)
y_exact = [0]*len(x)
y_rk4 = [0] * len(x)
```

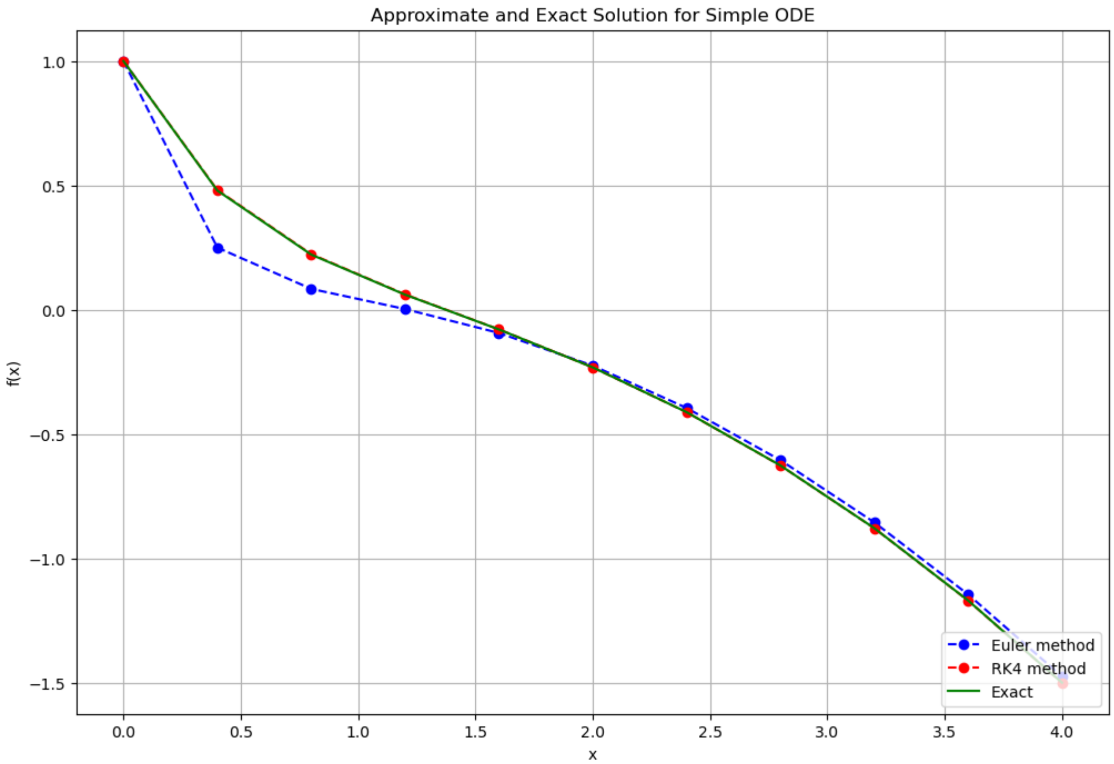
[286]:
```python
y_euler = euler(x,y_euler,h,y0,func)
```

[287]:
```
y_exact = exact(x,y_exact)
```

[288]:
```
y_rk4 = rk4(x,y_rk4,h,y0,func)
```

1 Question 2

[289]:
```
plt.figure(figsize = (12,8))
plt.plot(x,y_euler,'bo--', label='Euler method')
plt.plot(x,y_rk4,'ro--', label='RK4 method')
plt.plot(x, y_exact, 'g', label='Exact')
plt.title('Approximate and Exact Solution \
for Simple ODE')
plt.xlabel('x')
plt.ylabel('f(x)')
plt.grid()
plt.legend(loc='lower right')
plt.savefig('./figures/ode.png')
plt.show()
```



Difference between Exact and Euler method

[290]:
```
np.abs((np.array(y_exact) - np.array(y_euler)).mean())
```

[290]: np.float64(0.02926750179121326)

Difference between Exact and RK4

[291]:
```
np.abs((np.array(y_exact) - np.array(y_rk4)).mean())
```

[291]: np.float64(0.00039775895934885097)

[292]:
```
np.abs(y_euler[np.nonzero(x>=0.4)[0][0]]- y_exact[np.nonzero(x>=0.4)[0][0]])
```

[292]: np.float64(0.22932896411722153)

[293]:
```
np.abs(y_rk4[np.nonzero(x>=0.4)[0][0]]- y_exact[np.nonzero(x>=0.4)[0][0]])
```

[293]: np.float64(0.002191035882778525)

2 Question 3, finding h for euler method

[294]:
```
found = False
h = 0.4
while not found:
    x = np.arange(0.0,4.0+h, h)
    y_euler = [0] * len(x)
    y_exact = [0]*len(x)
    y_euler = euler(x,y_euler,h,y0,func)
    y_exact = exact(x,y_exact)
    if np.abs(y_euler[np.nonzero(x>=0.4)[0][0]]- y_exact[np.nonzero(x>=0.
↪4)[0][0]]) <= 0.001:
        found = True
        print(np.abs(y_euler[np.nonzero(x>=0.4)[0][0]]- y_exact[np.nonzero(x>=0.
↪4)[0][0]]))
        break
    else:
        h = h-0.0001
```

```
0.0009786909834365032
```

[295]:
```
print("h: ", h)
```

```
h:  0.0030000000000277582
```

[295]:

3 Question 3, finding h for rk4

[296]:
```
found = False
h = 0.4
while not found:
    x = np.arange(0.0,4.0+h, h)
    y_rk4 = [0] * len(x)
    y_exact = [0]*len(x)
    y_rk4 = rk4(x,y_rk4,h,y0,func)
    y_exact = exact(x,y_exact)
    if np.abs(y_rk4[np.nonzero(x>=0.4)[0][0]]- y_exact[np.nonzero(x>=0.
 ↪4)[0][0]]) <= 0.001:
        found = True
        print(np.abs(y_rk4[np.nonzero(x>=0.4)[0][0]]- y_exact[np.nonzero(x>=0.
 ↪4)[0][0]]))
        break
    else:
        h = h-0.0001
```

```
0.0009997201746155815
```

[297]:
```
print("h: ", h)
```

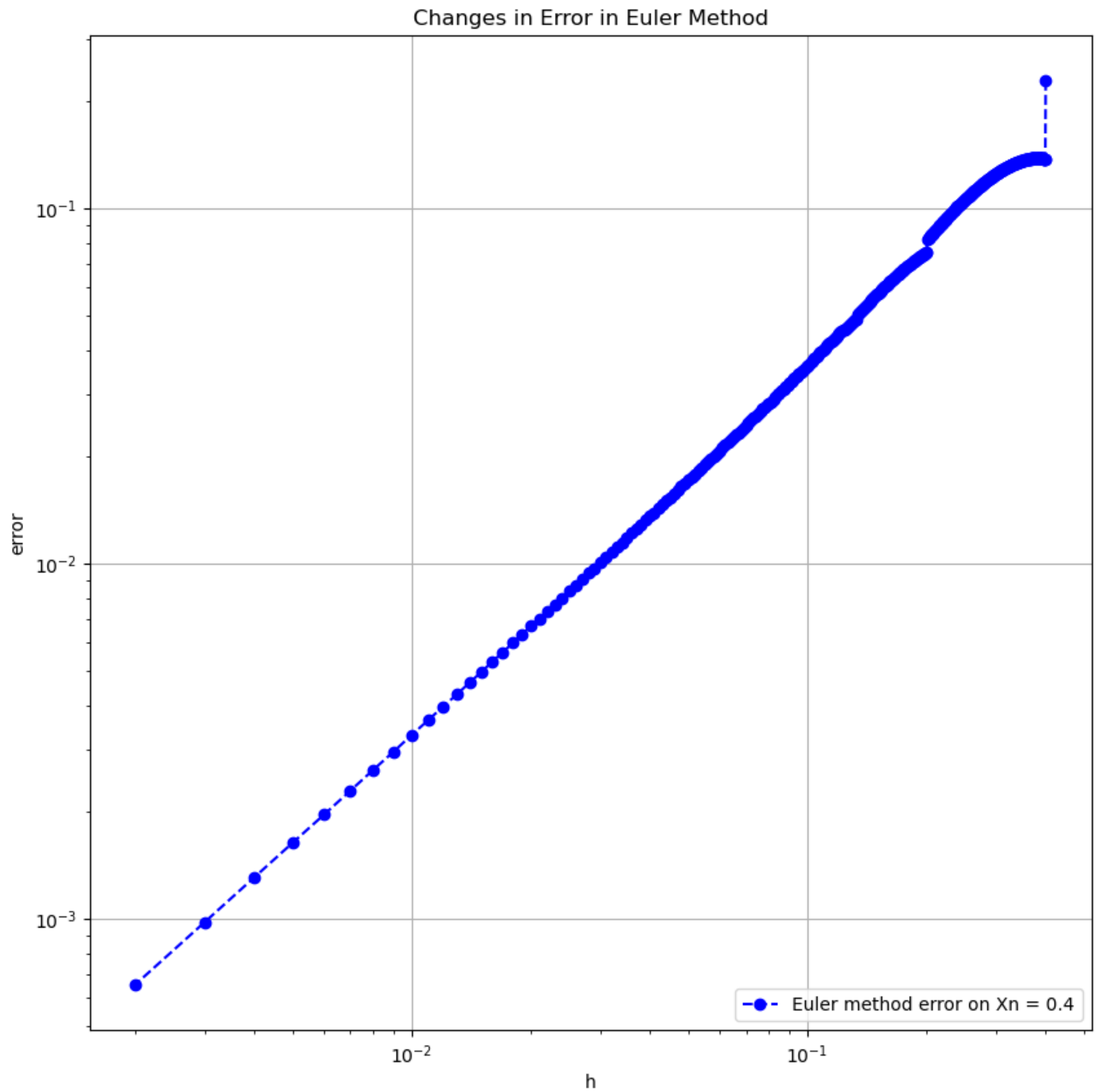
```
h:  0.34340000000000626
```

4 Question 4

[298]:
```
found = False
h = 0.4
h_log = []
err = []
while h >= 0.001:
    h_log.append(h)
    x = np.arange(0.0,4.0+h, h)
    y_euler = [0] * len(x)
    y_exact = [0]*len(x)
    y_euler = euler(x,y_euler,h,y0,func)
    y_exact = exact(x,y_exact)
    err.append(np.abs(y_euler[np.nonzero(x>=0.4)[0][0]]- y_exact[np.
 ↪nonzero(x>=0.4)[0][0]]))
    h = h-0.001
```

[299]:
```
plt.figure(figsize = (10,10))
plt.loglog(h_log,err,'bo--', label='Euler method error on Xn = 0.4')
plt.title('Changes in Error in Euler Method')
plt.xlabel('h')
plt.ylabel('error')
```

```
plt.grid()
plt.legend(loc='lower right')
plt.savefig('./figures/eulererror.png')
plt.show()
```

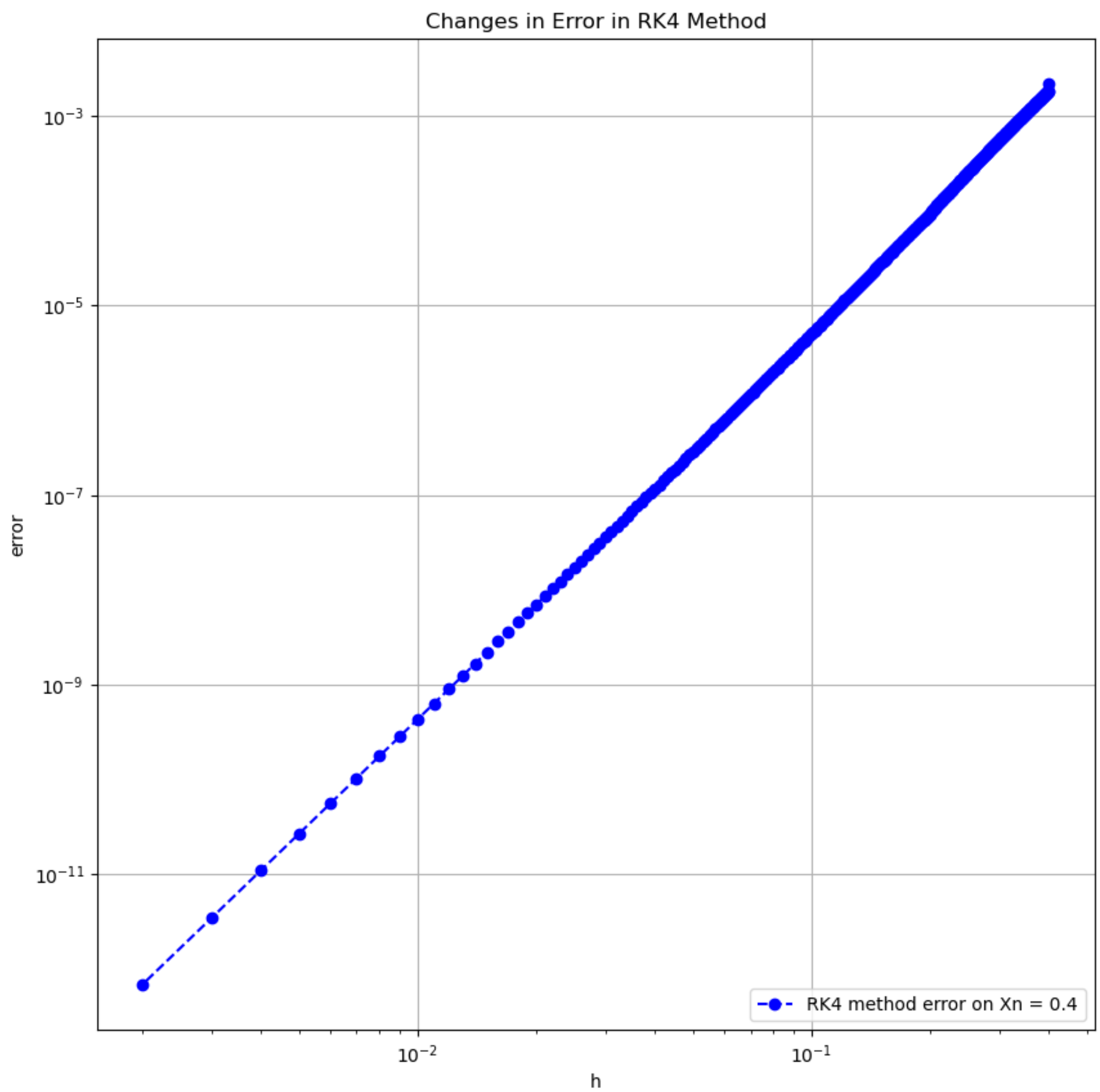


[300]:
```
found = False
h = 0.4
h_log = []
err = []
while h >= 0.001:
    h_log.append(h)
    x = np.arange(0.0,4.0+h, h)
```

```
    y_rk4 = [0] * len(x)
    y_exact = [0]*len(x)
    y_rk4 = rk4(x,y_euler,h,y0,func)
    y_exact = exact(x,y_exact)
    err.append(np.abs(y_rk4[np.nonzero(x>=0.4)[0][0]]- y_exact[np.nonzero(x>=0.
↪4)[0][0]]))
    h = h-0.001
```

[301]:

```
plt.figure(figsize = (10,10))
plt.loglog(h_log,err,'bo--', label='RK4 method error on Xn = 0.4')
plt.title('Changes in Error in RK4 Method')
plt.xlabel('h')
plt.ylabel('error')
plt.grid()
plt.legend(loc='lower right')
plt.savefig('./figures/rk4error.png')
plt.show()
```



5 Question 5

[302]:
```
import math
def substitution(x,y,z):
    return -2*z - 5*y + math.cos(x)
```

note: z = y'

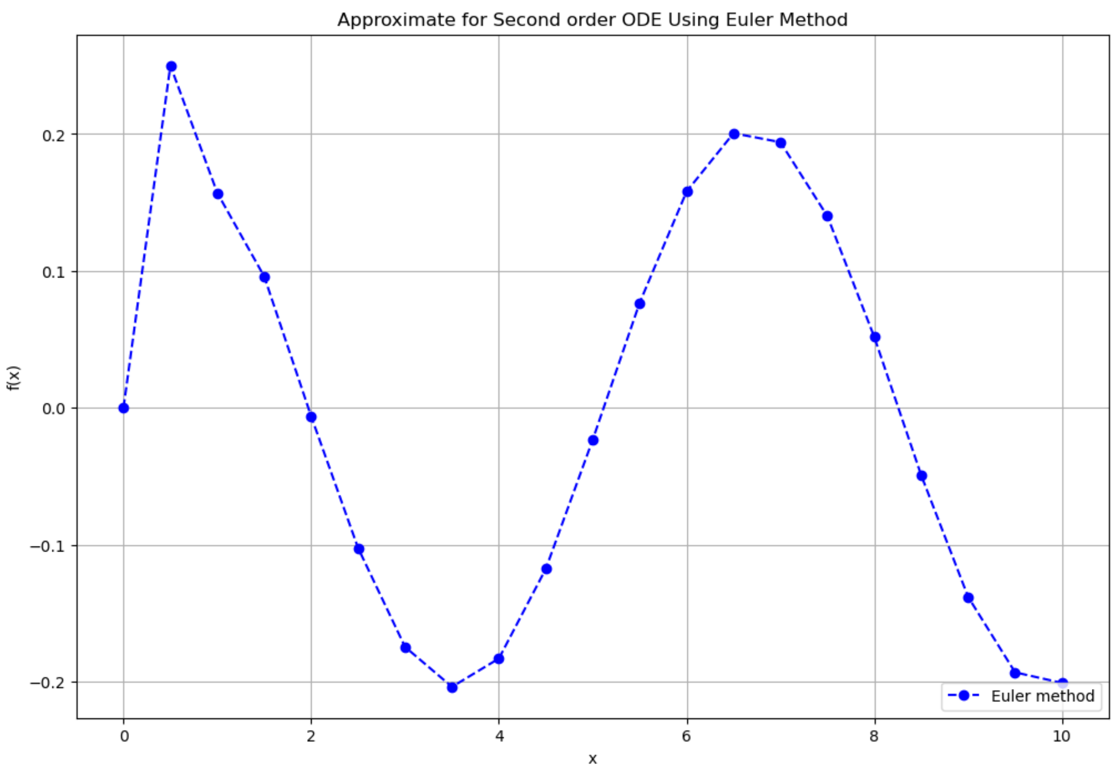
[303]:
```
h = 0.5
x = np.arange(0.0,10.0+h, h)
y0 = 0.0
y_prime0 = 0
```

```
y_euler = [0] * len(x)
y_exact = [0] * len(x)
```

[304]:
```
def eulerSecond(x,y, h, y0, y_prime0, func):
    y[0] = y0
    z = [0] * len(y)
    z[0] = y_prime0
    for i in range(1,len(x)):
        z[i] = z[i-1] + h*func(x[i-1], y[i-1], z[i-1])
        y[i] = y[i-1] + h*z[i]
    return y
```

[305]:
```
y_euler = eulerSecond(x,y_euler,h,y0,y_prime0, substitution)
```

[306]:
```
plt.figure(figsize = (12,8))
plt.plot(x,y_euler,'bo--', label='Euler method')
plt.title('Approximate \
for Second order ODE Using Euler Method')
plt.xlabel('x')
plt.ylabel('f(x)')
plt.grid()
plt.legend(loc='lower right')
plt.savefig('./figures/eulerSecond.png')
plt.show()
```

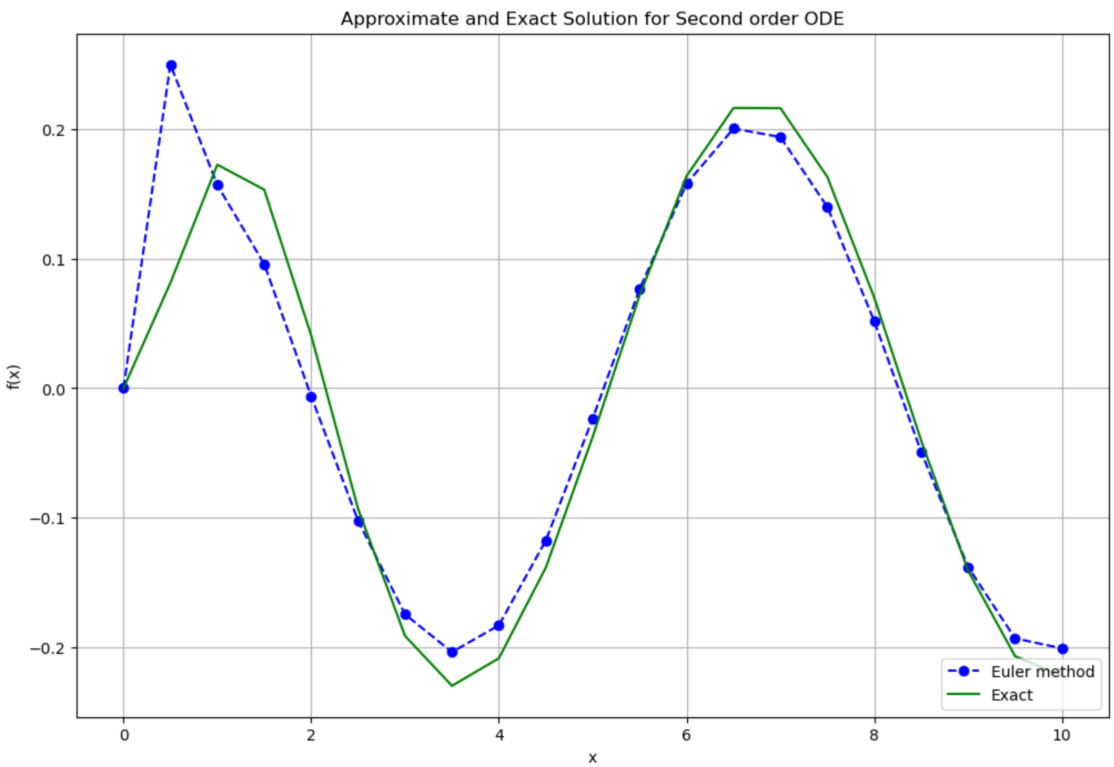


6 Question 6

[307]:
```
def secondODEexact(x,y):
    for i in range(0,len(x)):
        y[i] = np.exp(-x[i])*(-math.cos(2*x[i])/5 - 3*math.sin(2*x[i])/20) +␣
↪math.sin(x[i])/10 + math.cos(x[i])/5
    return y
```

[308]:
```
y_exact = secondODEexact(x,y_exact)
```

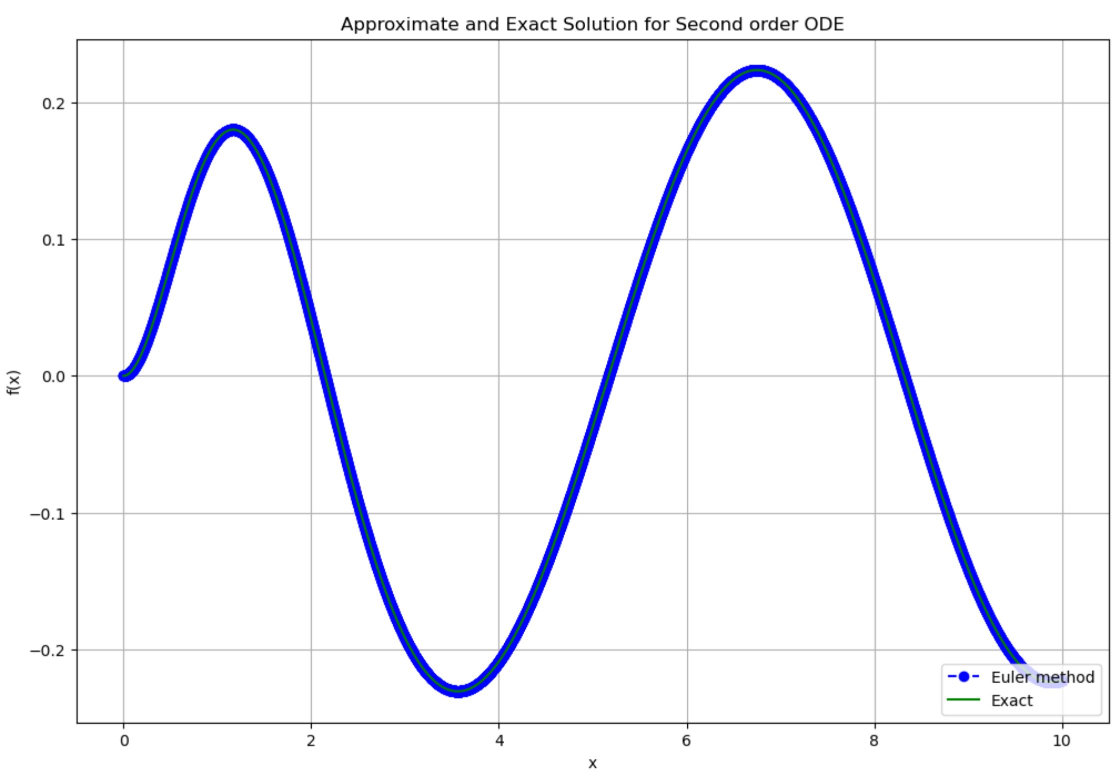
[309]:
```
plt.figure(figsize = (12,8))
plt.plot(x,y_euler,'bo--', label='Euler method')
plt.plot(x, y_exact, 'g', label='Exact')
plt.title('Approximate and Exact Solution \
for Second order ODE')
plt.xlabel('x')
plt.ylabel('f(x)')
plt.grid()
plt.legend(loc='lower right')
plt.savefig('./figures/secondOdeExact.png')
plt.show()
```



6.1 Experimentation on step size

[310]:
```
h = 0.001
x = np.arange(0.0,10.0+h, h)
y0 = 0.0
y_prime0 = 0
y_euler = [0] * len(x)
y_exact = [0] * len(x)
y_euler = eulerSecond(x,y_euler,h,y0,y_prime0, substitution)
y_exact = secondODEexact(x,y_exact)
```

[311]:
```
plt.figure(figsize = (12,8))
plt.plot(x,y_euler,'bo--', label='Euler method')
plt.plot(x, y_exact, 'g', label='Exact')
plt.title('Approximate and Exact Solution \
for Second order ODE')
plt.xlabel('x')
plt.ylabel('f(x)')
plt.grid()
plt.legend(loc='lower right')
plt.show()
```



7 Question 8 Damped Harmonic Oscillator

[312]:
```
def analyticSol(w,a):
    for i in range(0,len(w)):
        a[i] = 1/math.sqrt((4-w[i]**2)**2 + w[i]**2)
    return a, max(a)
```

[313]:
```
w = np.arange(0,5,0.001)
a = [0]*len(w)
```

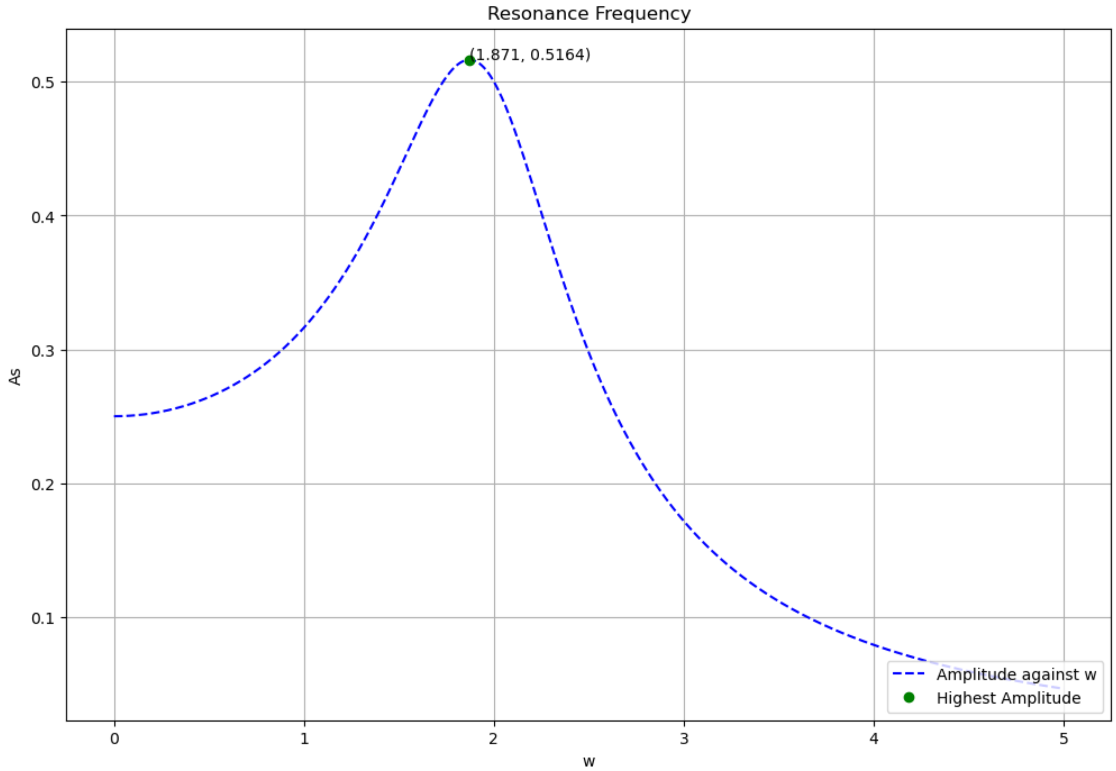
[314]:
```
a, amax = analyticSol(w,a)
```

[315]:
```
A = np.array(a)
```

[316]:
```
at = w[int(np.where(A == amax)[0][0])]

ats = [at]
am = [amax]
```

[317]:
```
plt.figure(figsize = (12,8))
plt.plot(w,a,'b--', label='Amplitude against w')
plt.plot(ats,am,'go', label='Highest Amplitude')
for i, j in zip(ats,am):
    plt.text(i, j, '({}, {})'.format(float('{:.5g}'.format(i)), float('{:.5g}'.
 ↪format(j))))
plt.title('Resonance Frequency')
plt.xlabel('w')
plt.ylabel('As')
plt.grid()
plt.legend(loc='lower right')
plt.savefig('./figures/resonance.png')
plt.show()
```



Resonance Frequency
(1.871, 0.5164)
As
w
0.1
0.2
0.3
0.4
0.5
0
1
2
3
4
5
Amplitude against w
Highest Amplitude

[317]: