In [1]:

```
%matplotlib notebook
import numpy as np
from scipy import optimize
import matplotlib.pyplot as plt
```

Homework 2 Writeup

AMATH 301 B

Christian Diangco

Problem 1

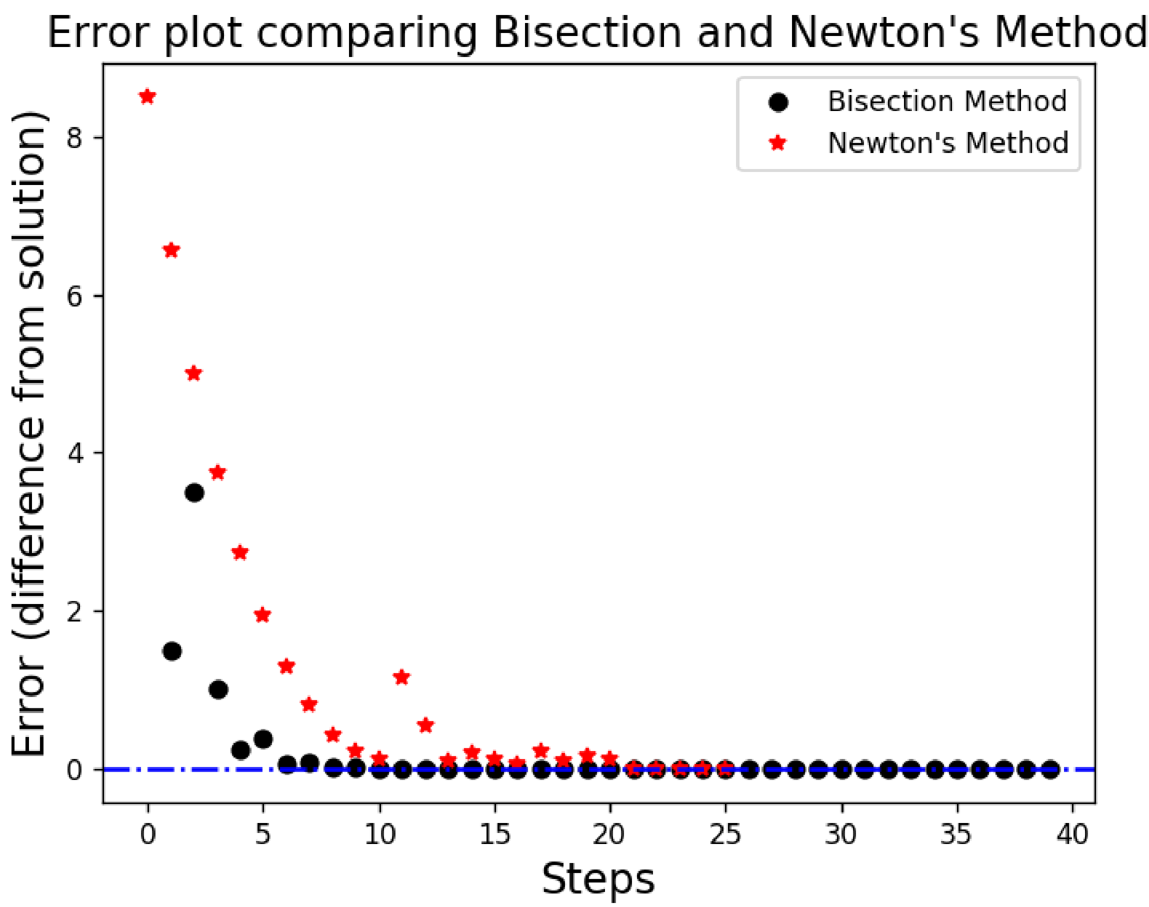
The root problem we must solve is $x^5 - 2x^4 + 4x^3 - 2x^2 + x - 1 - cos(30x) - 2\pi = 0$. The exact solution to the root problem is $x = 1.4964284338589289$

In [5]:

```
# Problem 1
def bisection(fun, left, right, tol):
    # fun - the function that we are trying to find the root of
    # left - the left endpoint (intially)
    # right - ""
    # tol - the tolerance for finding the root.

    midpoints = np.array([[]])

    for k in range(2000):
        mid = (left+right)/2
        f_left = fun(left)
        f_mid = fun(mid)
        f_right = fun(right)
        midpoints = np.append(midpoints,mid)
        if np.abs(f_mid) < tol:
            root = mid
            print('We found the root! And it is '+str(root))
            return root, k+1, midpoints
        elif f_mid*f_left < 0:
            right = mid
        elif f_mid*f_right <0:
            left = mid
        else:
            print('No root found in this interval')
            return np.nan, np.nan

f = lambda x: x**5 - 2 * x**4 + 4 * x**3 - 2 * x**2 + x - 1 - np.cos(30 * x) - 2 * np.pi
sol = float(optimize.fsolve(f,1.5))
midpoints = bisection(f,-10,10,1e-9)[2]
bi_err = abs(sol - midpoints)

# Newton's Method
df = lambda x: 5 * x**4 - 8 * x**3 + 12 * x**2 - 4 * x + 1 + 30 * np.sin(30 * x)
temp = 10
x = temp
guesses = np.array([[x]])
for i in range(100):
    if abs(f(x)) < 1e-9:
        break
    x = temp - f(temp) / df(temp)
    guesses = np.append(guesses, x)
```

```
    temp = x
newton_err = abs(guesses-sol)

# Plotting
fig, ax = plt.subplots()
length = len(midpoints)
ax.plot(np.arange(1,length + 1), bi_err, "ko", label="Bisection Method")
ax.plot(np.arange(i + 1), newton_err, "r*", label="Newton's Method")
ax.axhline(y=1e-9, color="blue", linestyle="-.")
ax.legend()
ax.set_title("Error plot comparing Bisection and Newton's Method", fontsize=15)
ax.set_xlabel("Steps", fontsize=15)
ax.set_ylabel("Error (difference from solution)", fontsize=15)
```

```
We found the root! And it is 1.4964284338566358
```



Out[5]:

```
Text(0, 0.5, 'Error (difference from solution)')
```

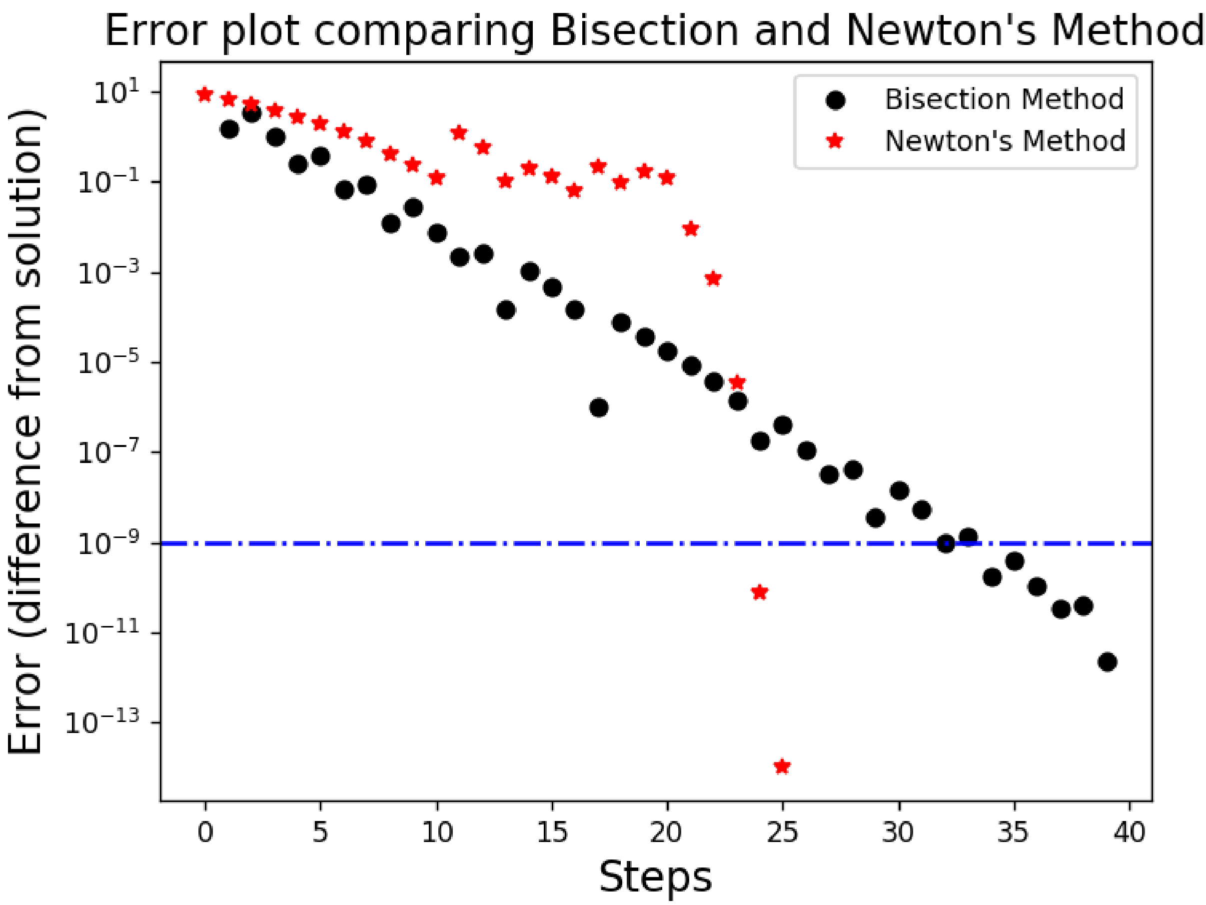
Problem 2

From the plot shown below, Newton's Method performed **better** than the Bisection Method. It is shown in the plot that Newton's Method was able to get below our error tolerance in fewer steps than the Bisection Method.

In [6]:

```
# Problem 2
fig, ax = plt.subplots()
length = len(midpoints)
ax.semilogy(np.arange(1,length + 1), bi_err, "ko", label="Bisection Method")
ax.semilogy(np.arange(i + 1), newton_err, "r*", label="Newton's Method")
ax.axhline(y=1e-9, color="blue", linestyle="-.")
ax.legend()
ax.set_title("Error plot comparing Bisection and Newton's Method", fontsize=15)
```

```
ax.set_xlabel("Steps", fontsize=15)
ax.set_ylabel("Error (difference from solution)", fontsize=15)
```



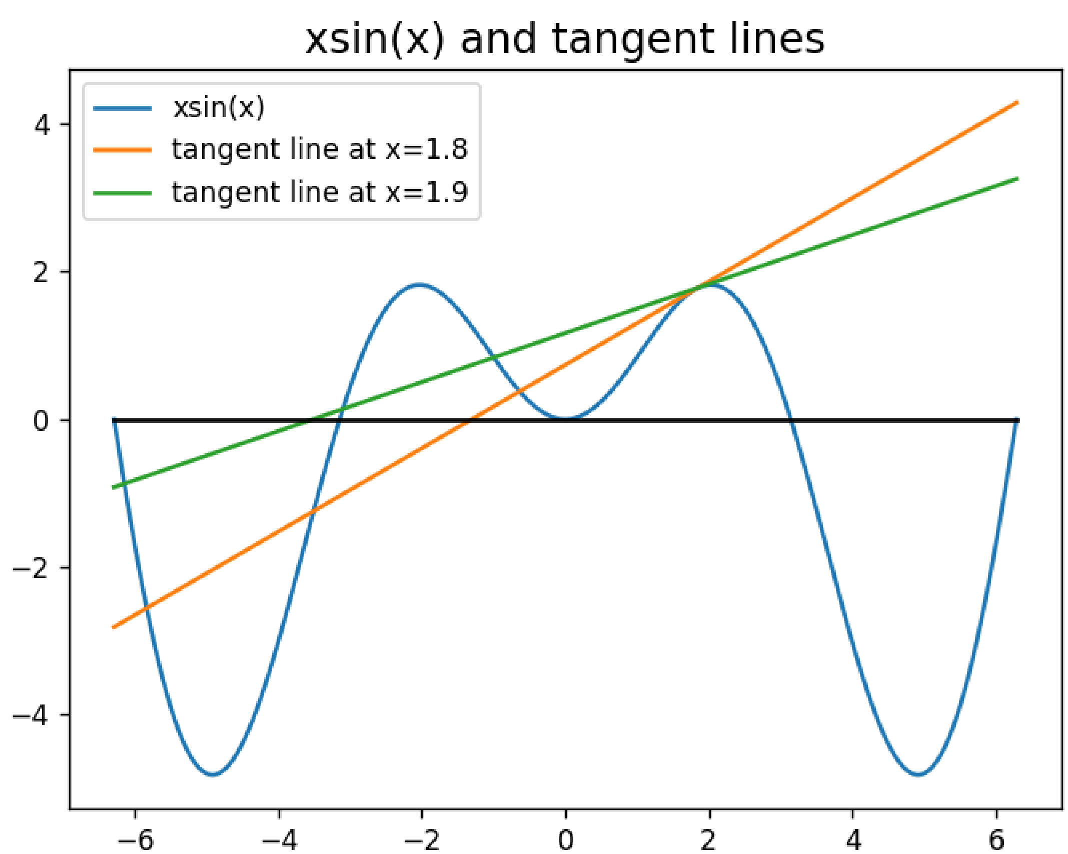
Out[6]:
```
Text(0, 0.5, 'Error (difference from solution)')
```

Problem 3

The error using an initial guess of $x_0 = 1.8$ is equal to $2.038853501266102e-05$.
The error using an initial guess of $x_0 = 1.9$ is equal to $3.141592653589793 = \pi$.
The reason that these answers aren't the same is because $f(x) = xsin(x)$ has **multiple roots** (x-values where y=0). We can see in the plot below that from $[-2\pi, 2\pi]$, there are 4 roots. Looking at the tangent line at $x = 1.8$, we can see that the next tangent line to be drawn will be closest to the root at $x = 0$, which is why $x_0 = 1.8$ gave us that root. In contrast, looking at the tangent line at $x = 1.9$, the next tangent line to be drawn will be closest to the root at $x = -\pi$, which is why our error for $x_0 = 1.9$ ends up equaling π.

In [68]:
```
# Problem 3
f = lambda x: x * np.sin(x)
df = lambda x: np.sin(x) + x * np.cos(x)
sol = 0

temp = 1.8 #initial guess
x = temp
for i in range(50):
    if abs(f(x)) < 1e-9:
        break
    x = temp - f(temp) / df(temp)
    guesses = np.append(guesses, x)
    temp = x
err1 = abs(x-sol)
```

```python
temp = 1.9 #initial guess
x = temp
for i in range(50):
    if abs(f(x)) < 1e-9:
        break
    x = temp - f(temp) / df(temp)
    guesses = np.append(guesses, x)
    temp = x
err2 = abs(x-sol)

# Plotting
x = np.linspace(-2 * np.math.pi, 2 * np.math.pi, 1000)
t1 = lambda x: df(1.8) * (x - 1.8) + f(1.8)
t2 = lambda x: df(1.9) * (x - 1.9) + f(1.9)  # tangent line @ x=1.9
fig, ax = plt.subplots()
ax.plot(x, f(x), label="xsin(x)")
ax.plot(x, t1(x), label="tangent line at x=1.8")
ax.plot(x, t2(x), label="tangent line at x=1.9")
ax.plot(x, np.zeros(len(x)), "k")
ax.legend()
ax.set_title("xsin(x) and tangent lines", fontsize="15")
```



Out[68]:
```
Text(0.5, 1.0, 'xsin(x) and tangent lines')
```