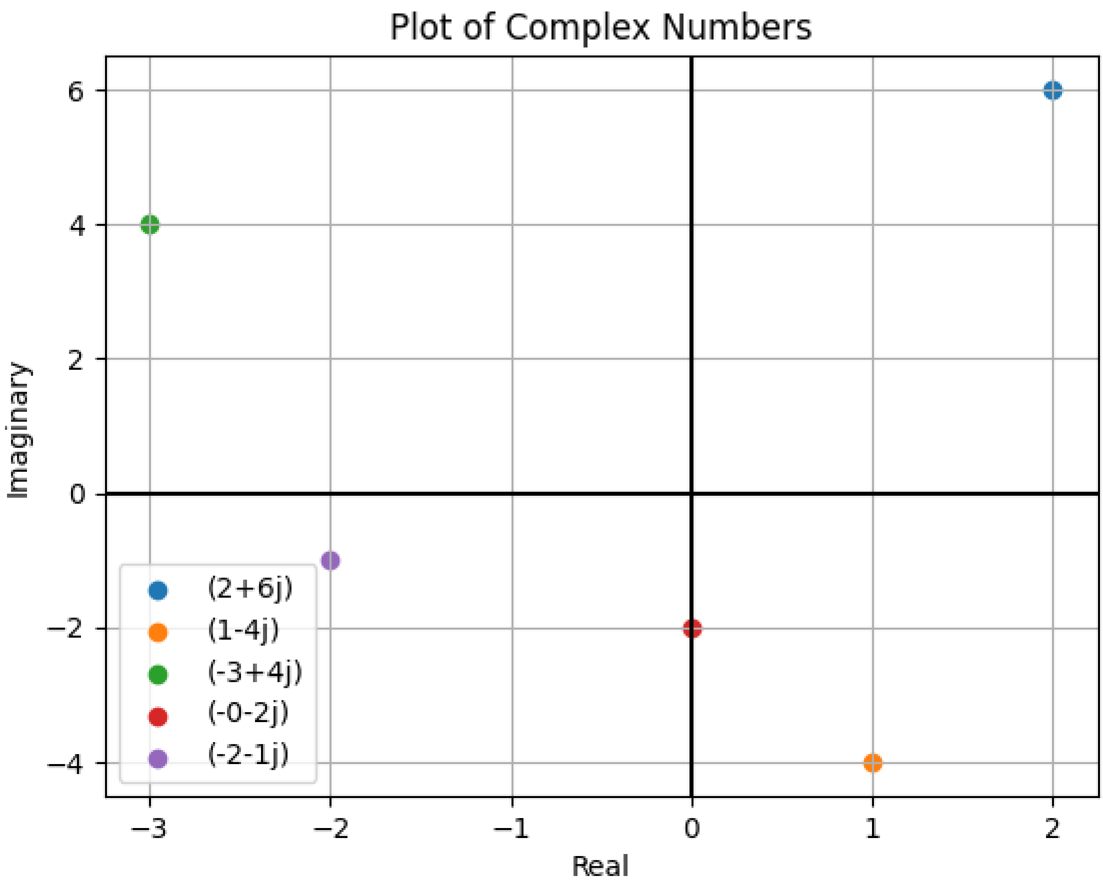
In [20]:
```python
a = eval(input("Enter complex number(arg,amp)$"))
#print(a[1]*exp(a[0]*1j),a[1]*(cos(a[0]) + 1j*sin(a[0])))
True if(a[1]*exp(a[0]*1j) == a[1]*(cos(a[0]) + 1j*sin(a[0]))) else False
```

Out[20]:
```
True
```

> WAP to plot the given set of complex numbers.

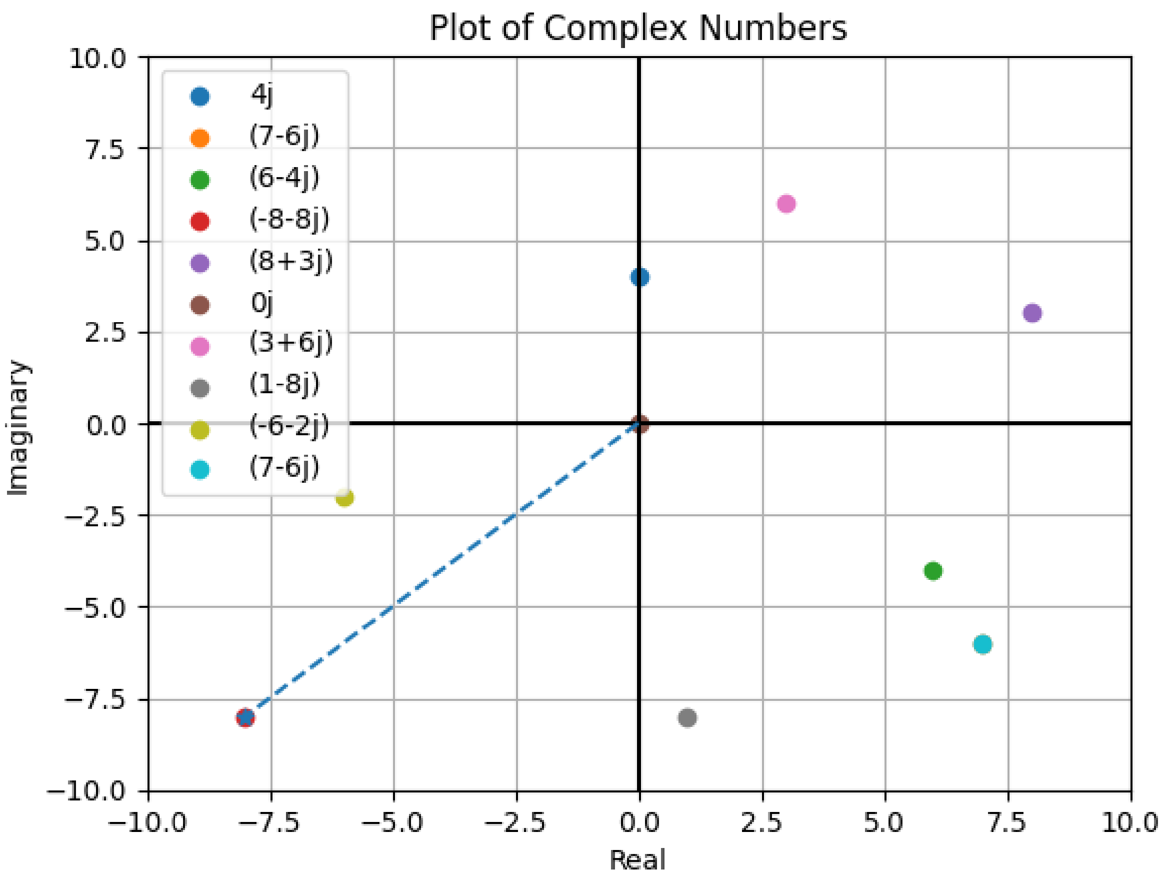
In [3]:
```python
z = eval(input("Enter list of complex numbers:"))
def plotComplex(z,show = True,legend = True):
    f = plt.figure()
    for i in z:
        plt.scatter(i.real,i.imag)
    if(legend):
        plt.legend(z)
    plt.grid()
    plt.axhline(color = "black")
    plt.axvline(color = "black")
    plt.xlabel("Real")
    plt.ylabel("Imaginary")
    plt.title("Plot of Complex Numbers")
    if show:
        plt.show()
    return f
#[2+6j, 1-4j,-3+4j, -2j, -2-1j]
plotComplex(z);
```



> WAP to find the furthest point from the origin

In [4]:
```
#z = eval(input("Enter list of complex numbers:"));
z = [complex(random.randint(-8,8),random.randint(-8,8)) for i in range(10)]
maxz = list(map(lambda z: abs(z),z))
maxz = maxz.index(max(maxz))
maxz = z[maxz]
print("Maximum Distance= ",maxz)
f = plotComplex(z,False)
plt.xlim(-10,10)
plt.ylim(-10,10)
plt.figure(f)
plt.scatter(maxz.real,maxz.imag,marker = "*",label = "Maximum")
plt.plot([0,maxz.real],[0,maxz.imag],linestyle = "--")
#plt.legend()
plt.show()
```

```
Maximum Distance=  (-8-8j)
```



In [5]:
```
z = (3 - 1j)/(2+3j) - (2-2j)/(1-5j)
z
```

Out[5]:
```
(-0.23076923076923073-1.1538461538461537j)
```

> WAP to to plot n^{th} roots of unity

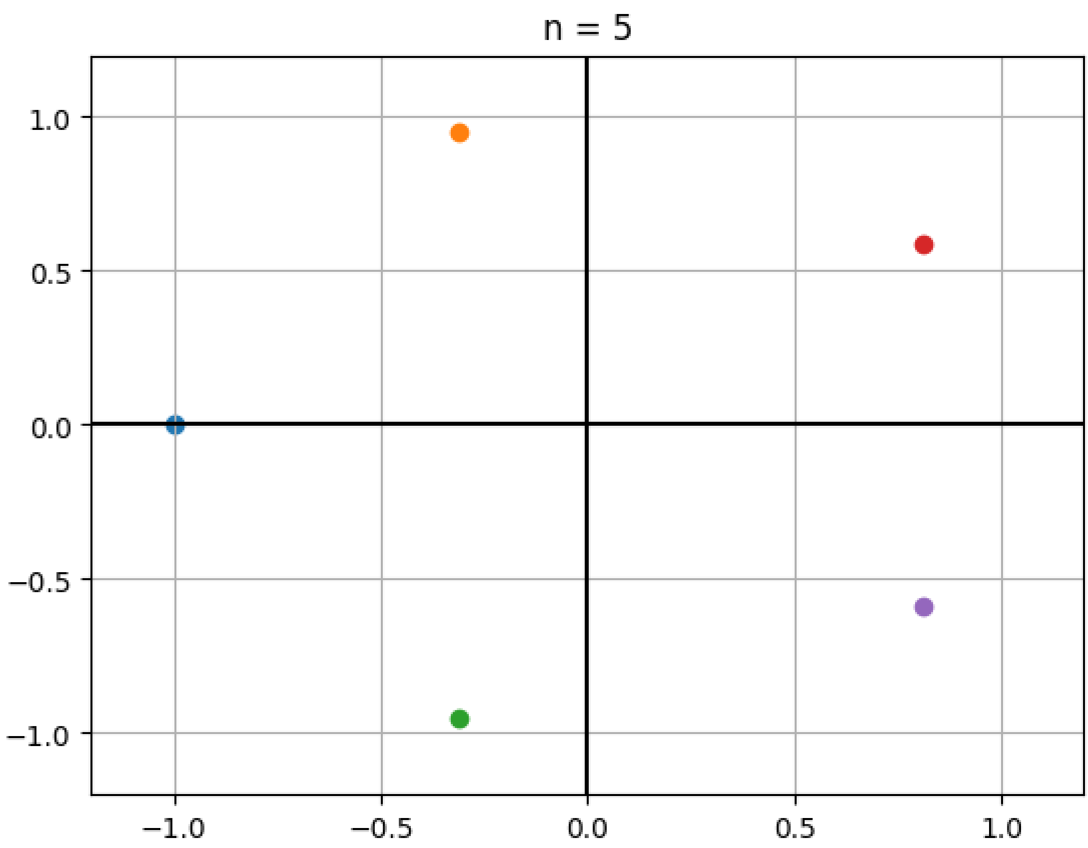
In [6]:
```
def unityRoots(n = 1):
    return np.roots([1]+ [0]*(n-1) + [1])
    #x = sp.Symbol("x")
    #return sp.solve(sp.Eq(x**n,1),x)
unityRoots(4)
```

Out[6]:
```
array([-0.70710678+0.70710678j, -0.70710678-0.70710678j,
        0.70710678+0.70710678j,  0.70710678-0.70710678j])
```

In [7]:

```python
def plotRoots(n):
    roots = unityRoots(n)
    f = plt.figure()
    plt.axhline(color = "black")
    plt.axvline(color = "black")
    plt.grid(which="both")
    plt.xlim(-1.2,1.2)
    plt.ylim(-1.2,1.2)
    for i in roots:
        i = complex(i)
        #pll.polar(1,phase(i))
        plt.scatter(i.real,i.imag)
    plt.title(f"n = {n}")
    plt.show()
widgets.interactive(plotRoots,n = (1,10))
```

Out[7]: n 5



Date: 17-01-24

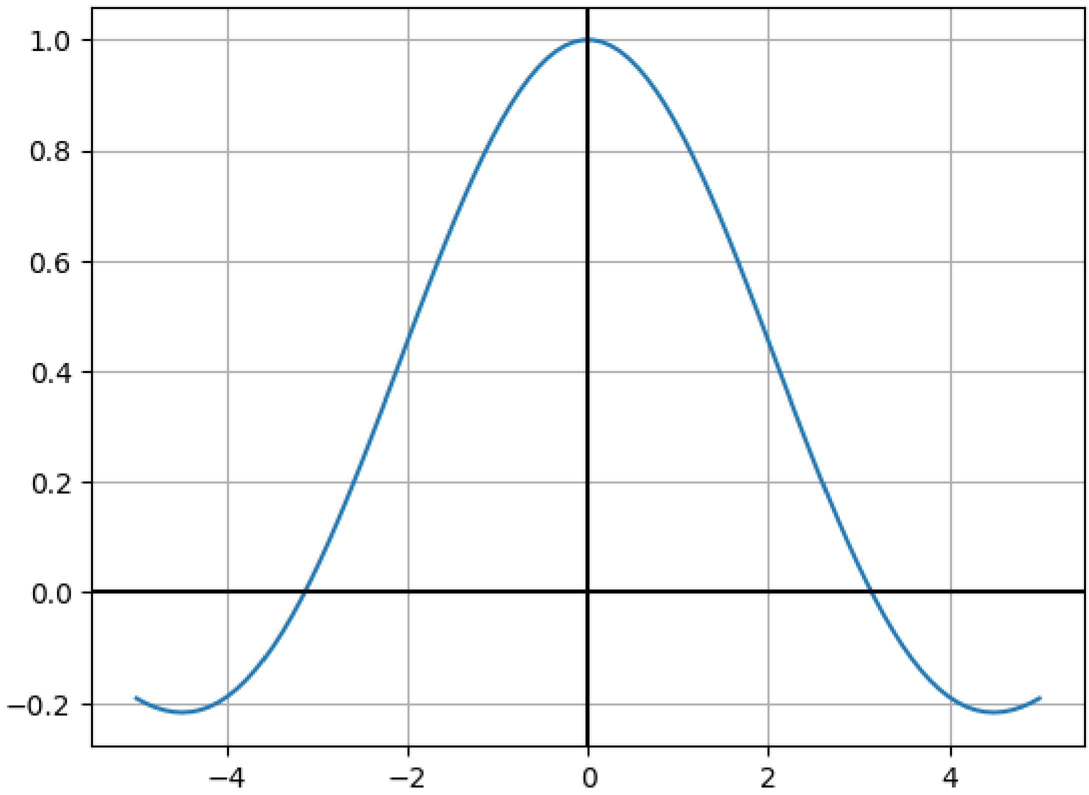
LIMITS OF COMPLEX FUNCTION

In [8]:

```python
def plotF(f:"Callable",x_lim):
    X = np.linspace(-x_lim,x_lim,100)
    plt.plot(X,[f(i) for i in X])
    plt.grid()
    plt.axhline(color = "black")
    plt.axvline(color = "black")
    plt.show()
```

In [9]:

```python
def f(x):
    return np.sin(x)/x
plotF(f,5)
```



In [10]:
```
def getLimit(f:"sympy.Function",z = sp.Symbol("z"), z0 = 0):
    lim = sp.limit(f,z,z0)
    return complex(lim.simplify())
```

In [11]:
```
getLimit("sin(z)/z",z0 = 1+1j)
```

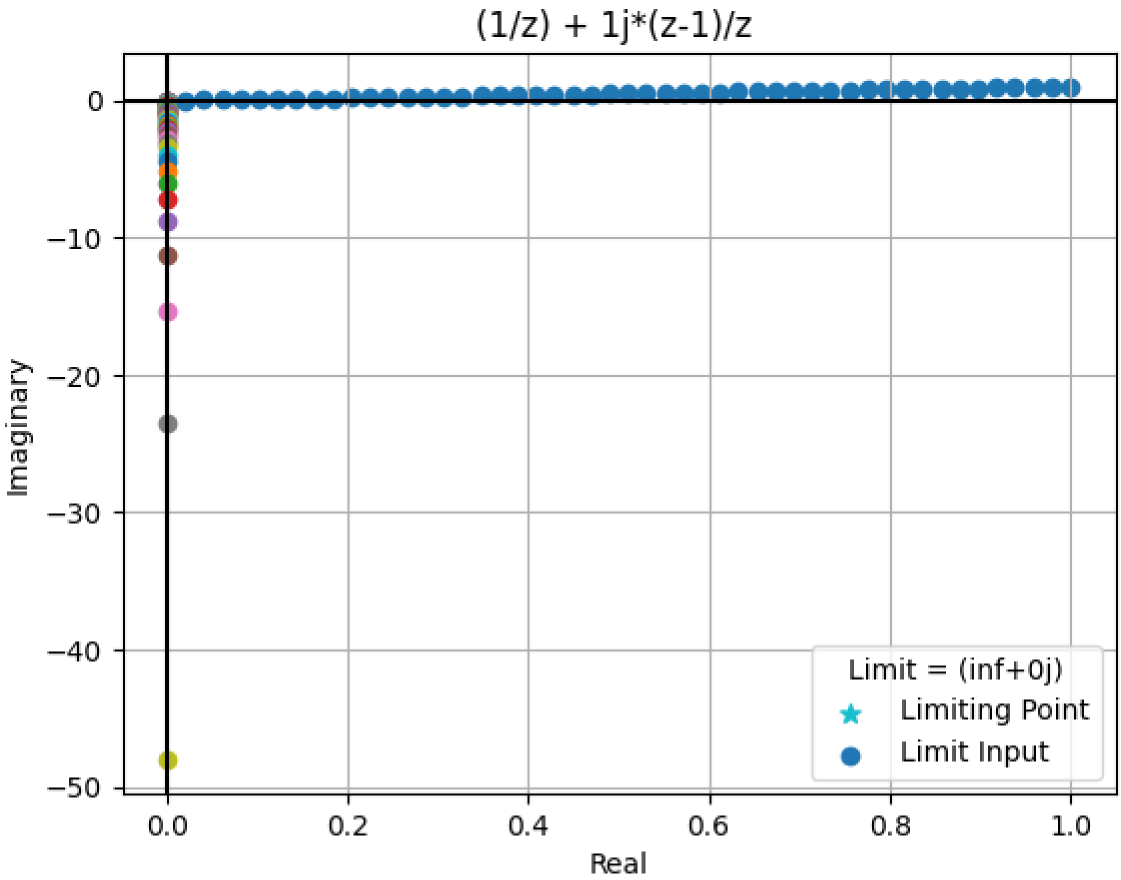
Out[11]:
```
(0.9667107481003567-0.3317468333156206j)
```

Date: 18-01-24

LIMIT OF A COMPLEX SEQUENCE

In [38]:
```
def limitingPoints(f:"sympy.Function",z = sp.Symbol("z"), zStart = 1,zLimit = 0):
    func = sp.lambdify(z,f,"numpy")
    points = np.linspace(zStart,zLimit,50)[:-1]
    fig = plotComplex([func(i) for i in points],show=False,legend=False)
    limit = getLimit(f,z0 = zLimit)
    plt.scatter(limit.real,limit.imag,marker = '*',s = 50,label ="Limiting Point")
    plt.scatter([i.real for i in points],[i.imag for i in points],label = "Limit Ir
    plt.legend(title = f"Limit = {limit}")
    plt.title(f"{f}")
    plt.show()
    return
```

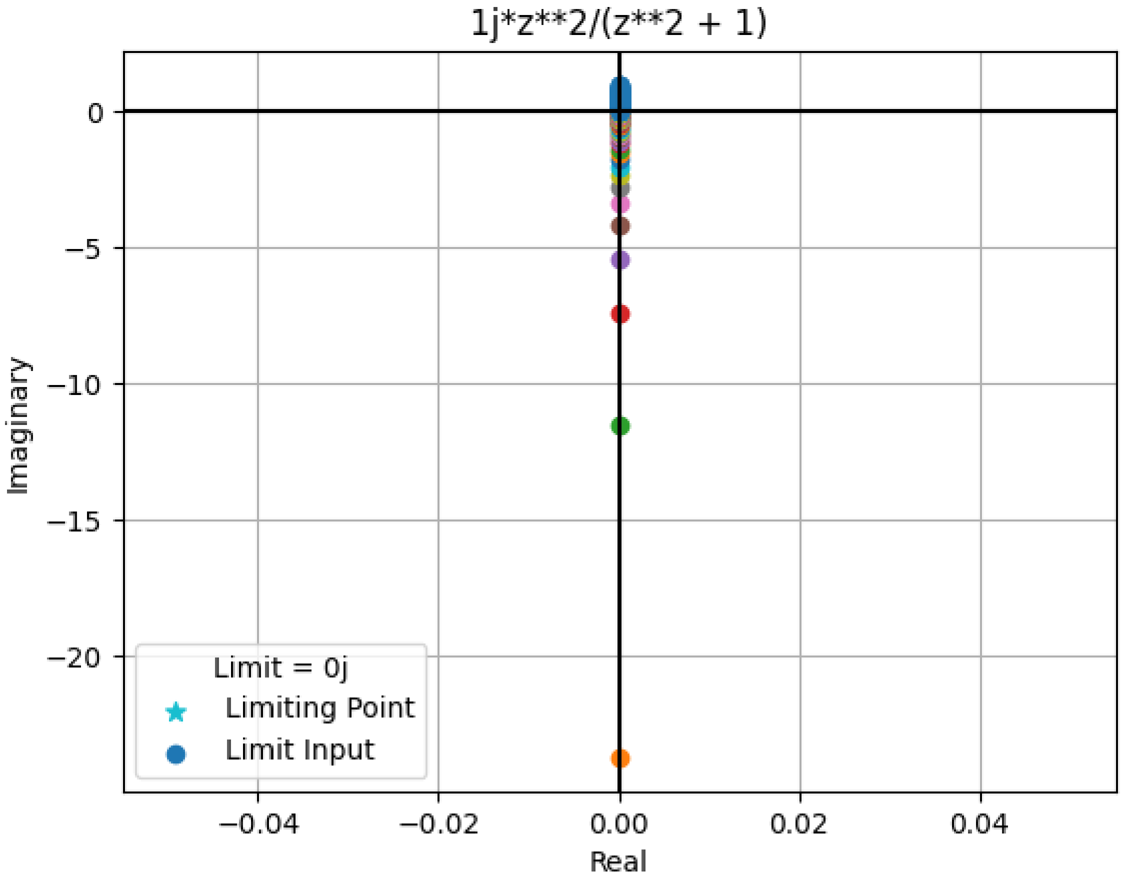
In [39]:
```
limitingPoints("(1/z) + 1j*(z-1)/z ",zStart=complex(1,1),zLimit=0)
```



> $$i\frac{n^2}{n^2+1}$$

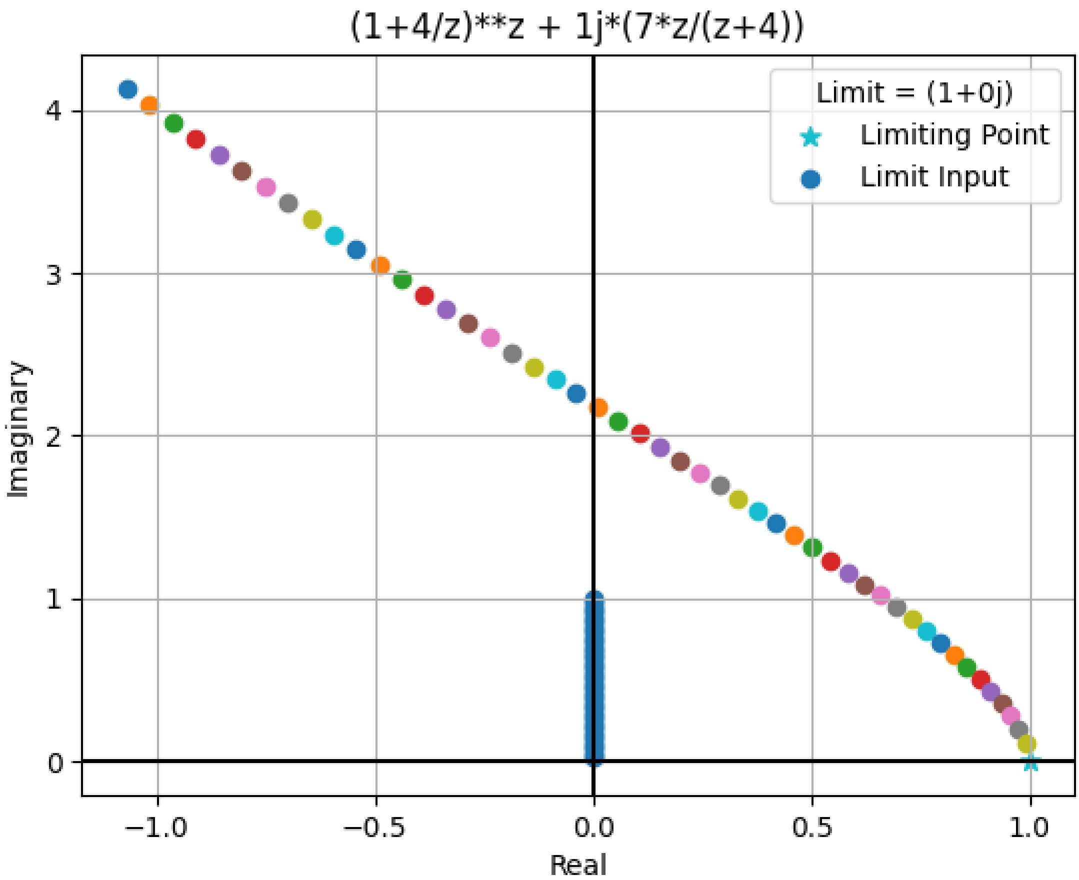
In [40]:
```
limitingPoints("1j*z**2/(z**2 + 1)",zStart=1j,zLimit=0)
```

```
<lambdifygenerated-20>:2: RuntimeWarning: divide by zero encountered in cdouble_sc
alars
  return 1j*z**2/(z**2 + 1)
<lambdifygenerated-20>:2: RuntimeWarning: invalid value encountered in cdouble_sca
lars
  return 1j*z**2/(z**2 + 1)
```



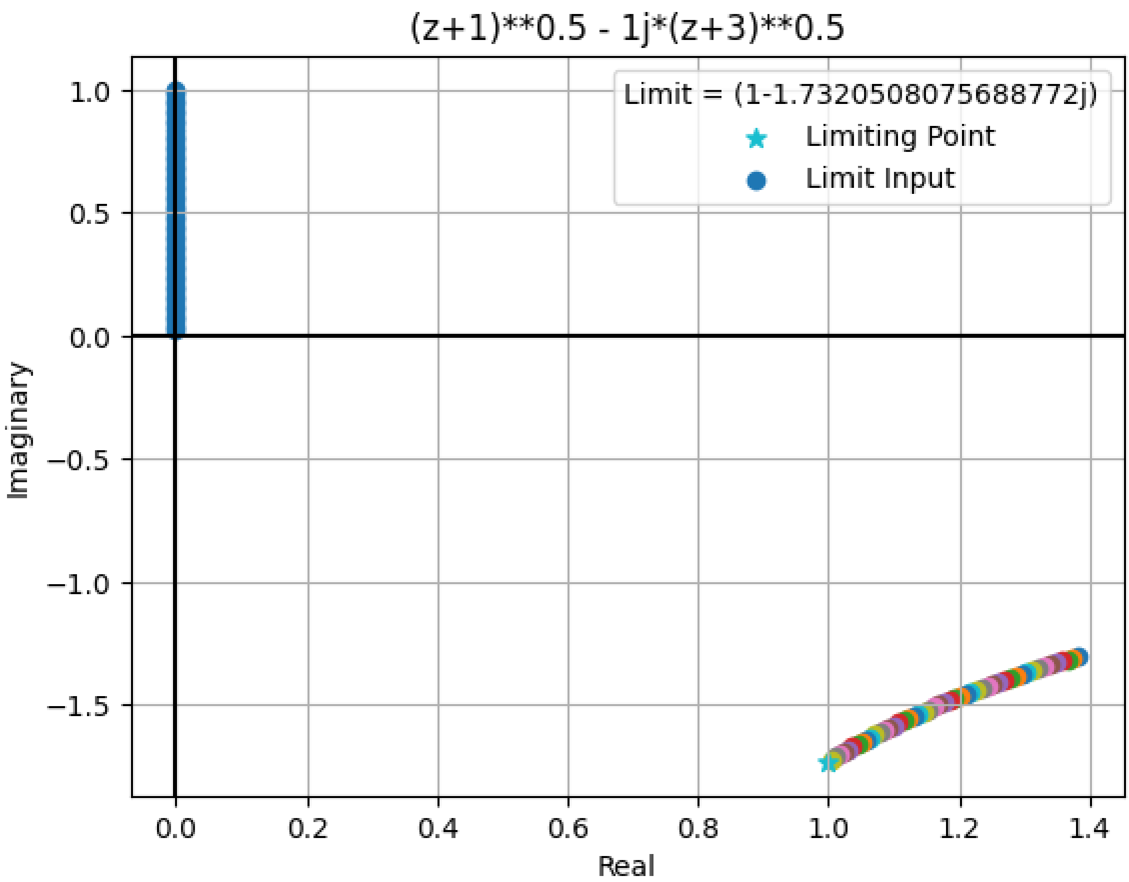
$$(1+\frac{4}{z})^z + i\frac{7z}{z+4}$$

In [41]:
```
limitingPoints("(1+4/z)**z + 1j*(7*z/(z+4))",zStart=1j,zLimit=0)
```



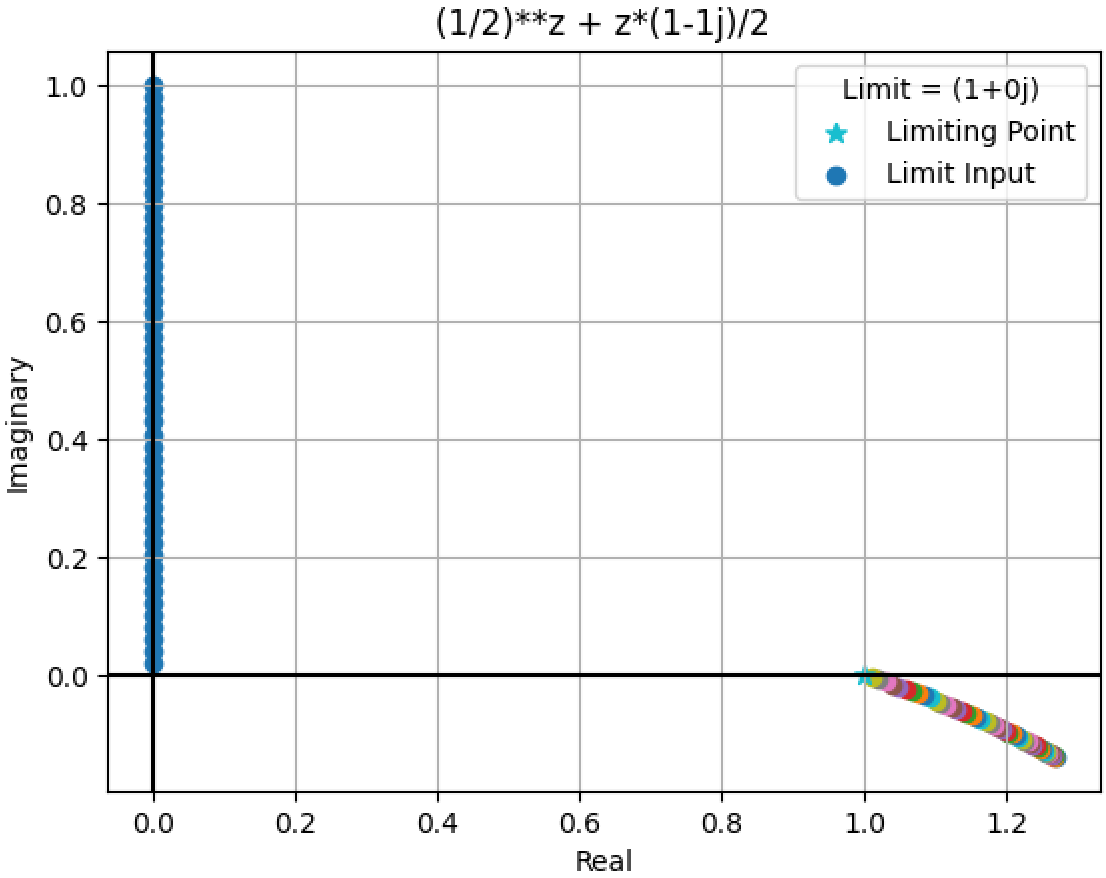
> $\sqrt(z+1) - i\sqrt(z+3)$

In [42]:
```
limitingPoints("(z+1)**0.5 - 1j*(z+3)**0.5",zStart=1j,zLimit=0)
```



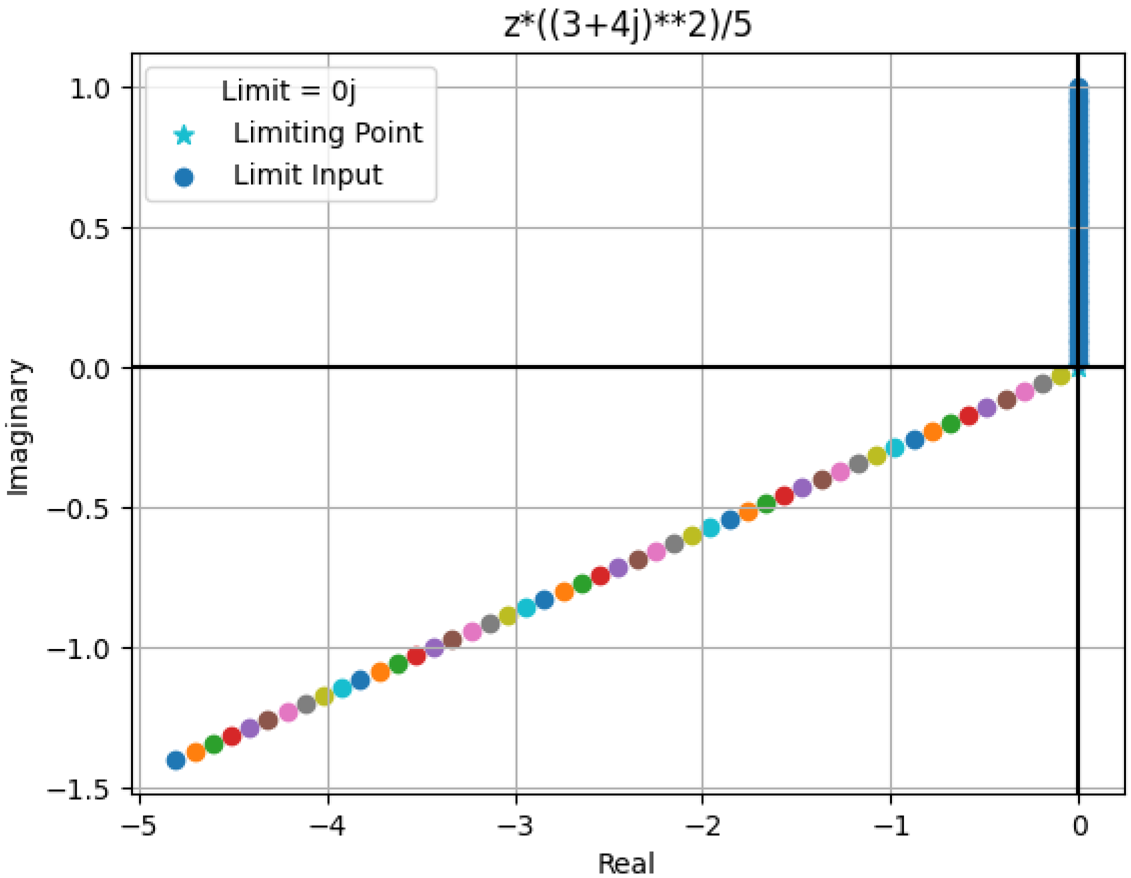
> $\frac{1}{2}^{z} + z\frac{1-i}{2}$

In [43]:
```
limitingPoints("(1/2)**z + z*(1-1j)/2",zStart=1j,zLimit=0)
```



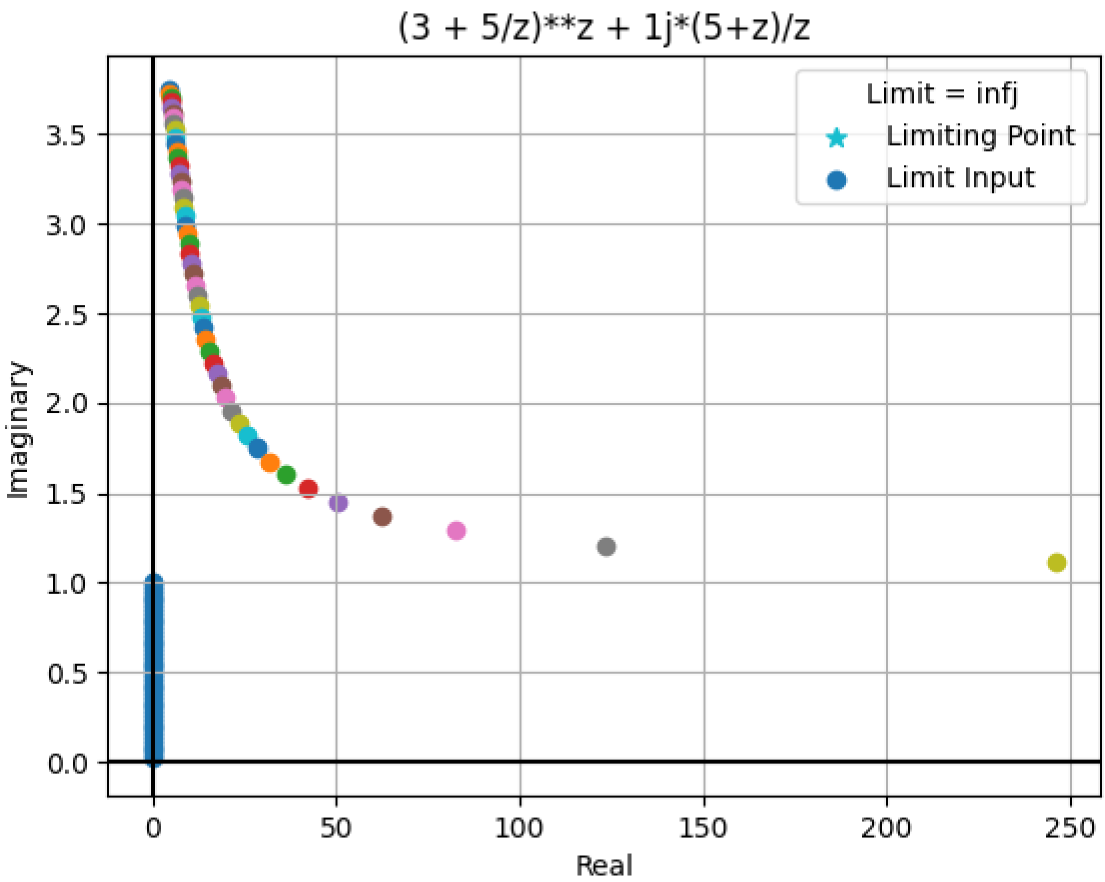
> $$z\frac{(3+4i)^2}{5}$$

In [44]:
```
limitingPoints("z*((3+4j)**2)/5",zStart=1j,zLimit=0)
```



> $$(3+\frac{5}{z})^z+i\frac{5+z}{z}$$

In [45]:
```
limitingPoints("(3 + 5/z)**z + 1j*(5+z)/z",zStart=1j,zLimit=0)
```

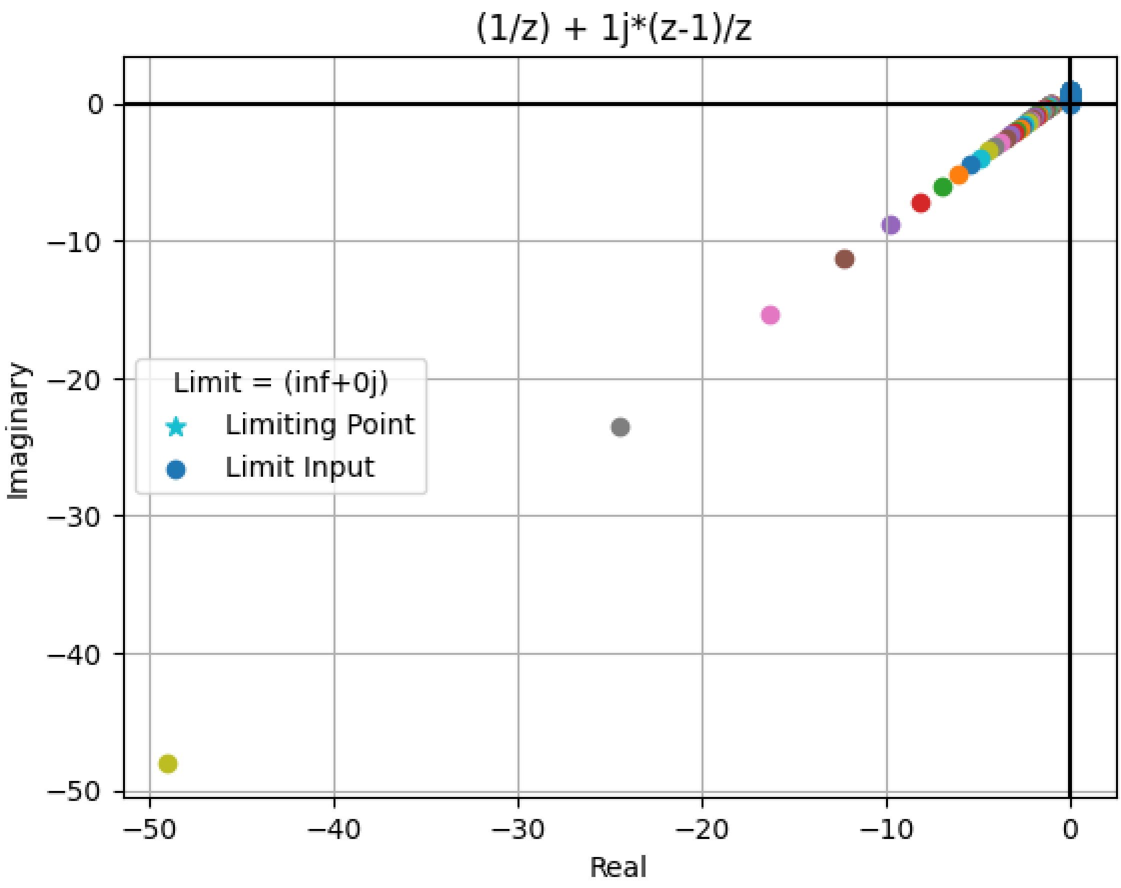


Asynchronous Assignment
Date: 22-01-24

> WAP to find and plot the given limits.

> $$\frac{1}{z}+i\frac{z-1}{z}$$

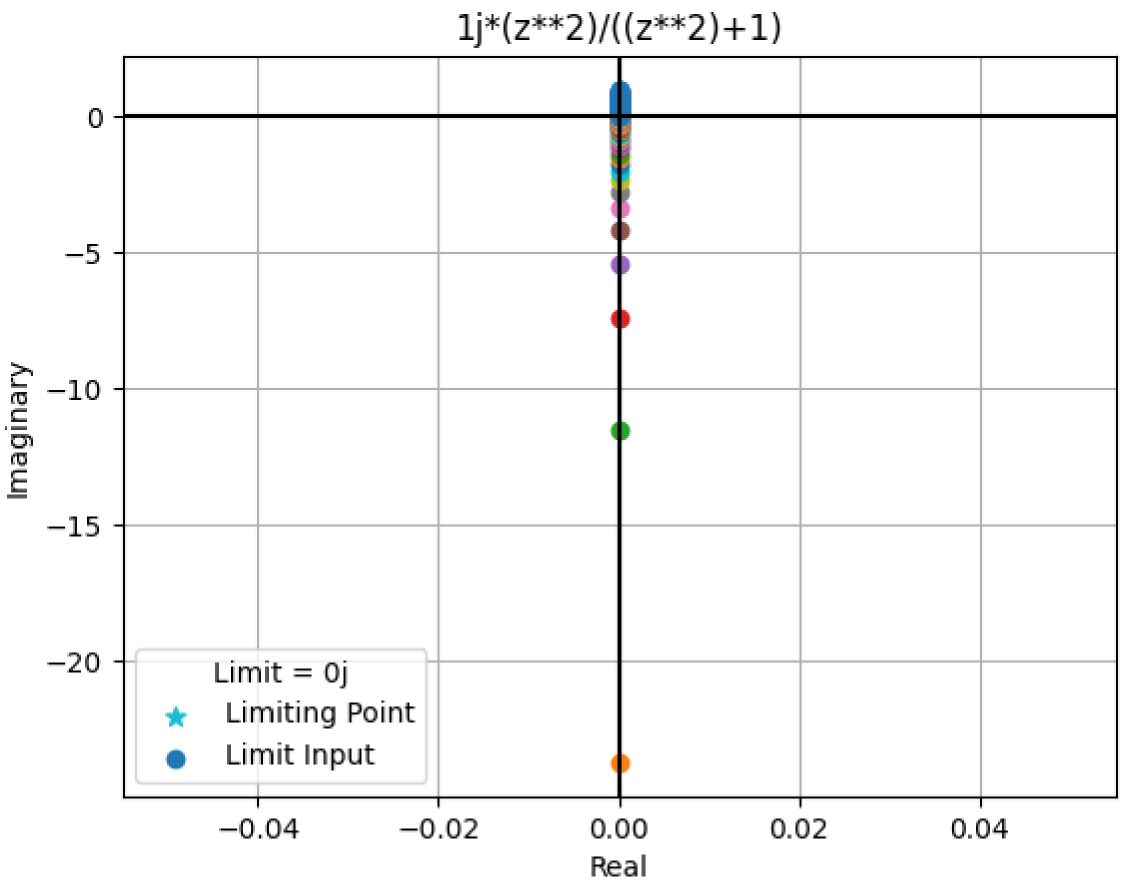
In [57]:
```
limitingPoints("(1/z) + 1j*(z-1)/z ",zStart=1j,zLimit=0)
```



> $$i\frac{z^2}{z^2+1}$$

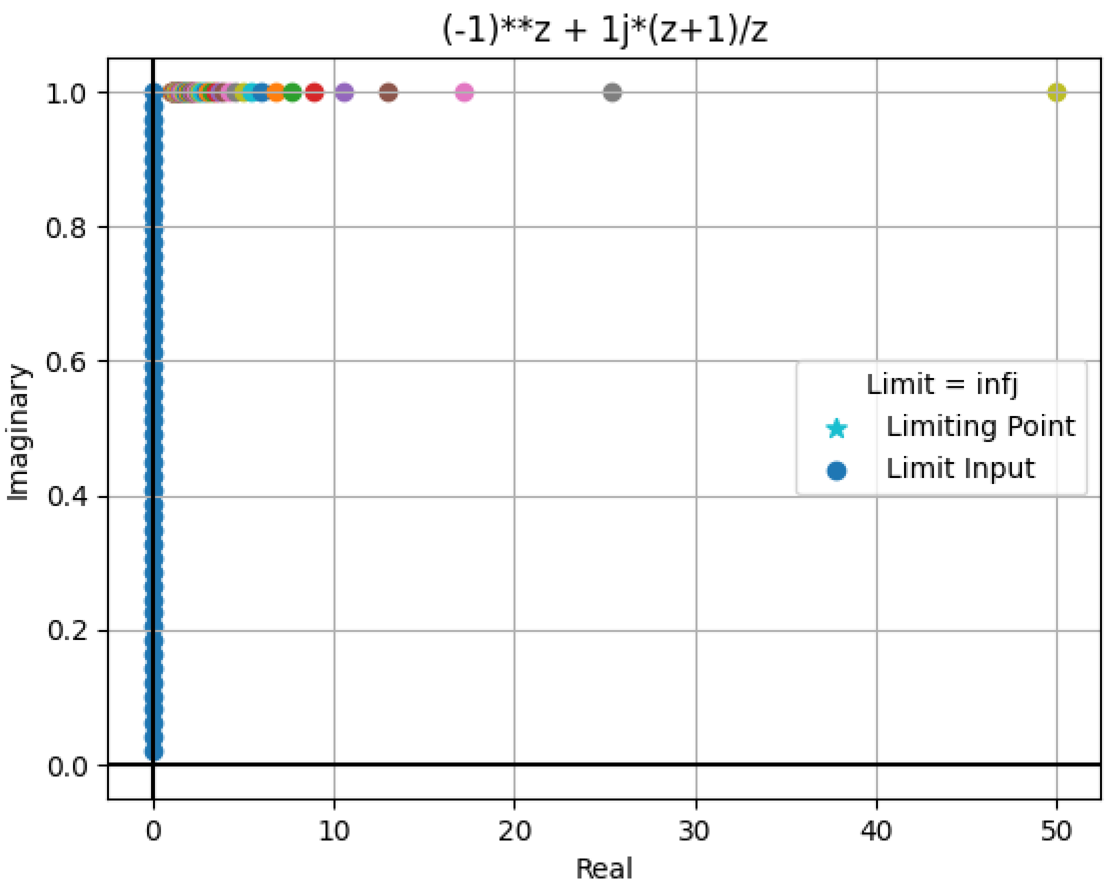
In [58]:
```
limitingPoints("1j*(z**2)/((z**2)+1)",zStart=1j,zLimit=0)
```

```
<lambdifygenerated-36>:2: RuntimeWarning: divide by zero encountered in cdouble_sc
alars
  return 1j*(z**2)/((z**2)+1)
<lambdifygenerated-36>:2: RuntimeWarning: invalid value encountered in cdouble_sca
lars
  return 1j*(z**2)/((z**2)+1)
```



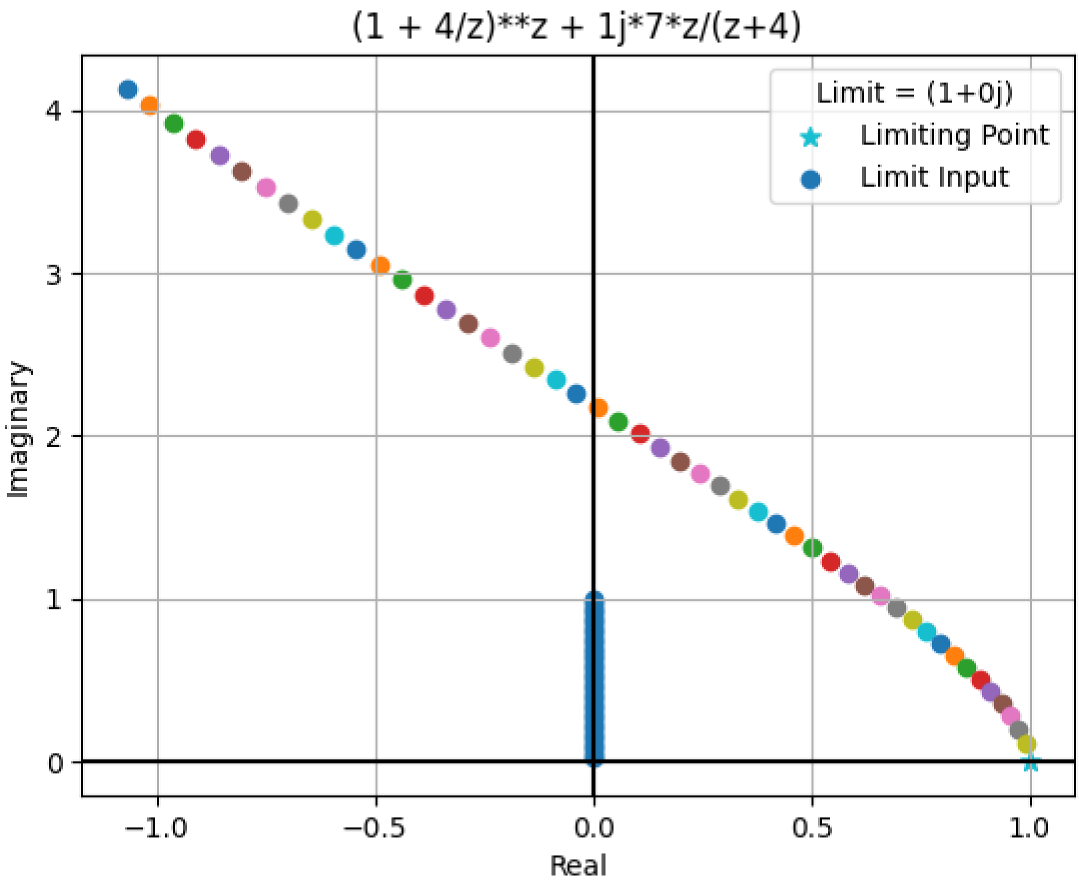
> $(-1)^z + i\dfrac{z+1}{z}$

In [59]:
```
limitingPoints("(-1)**z + 1j*(z+1)/z",zStart=1j,zLimit=0)
```



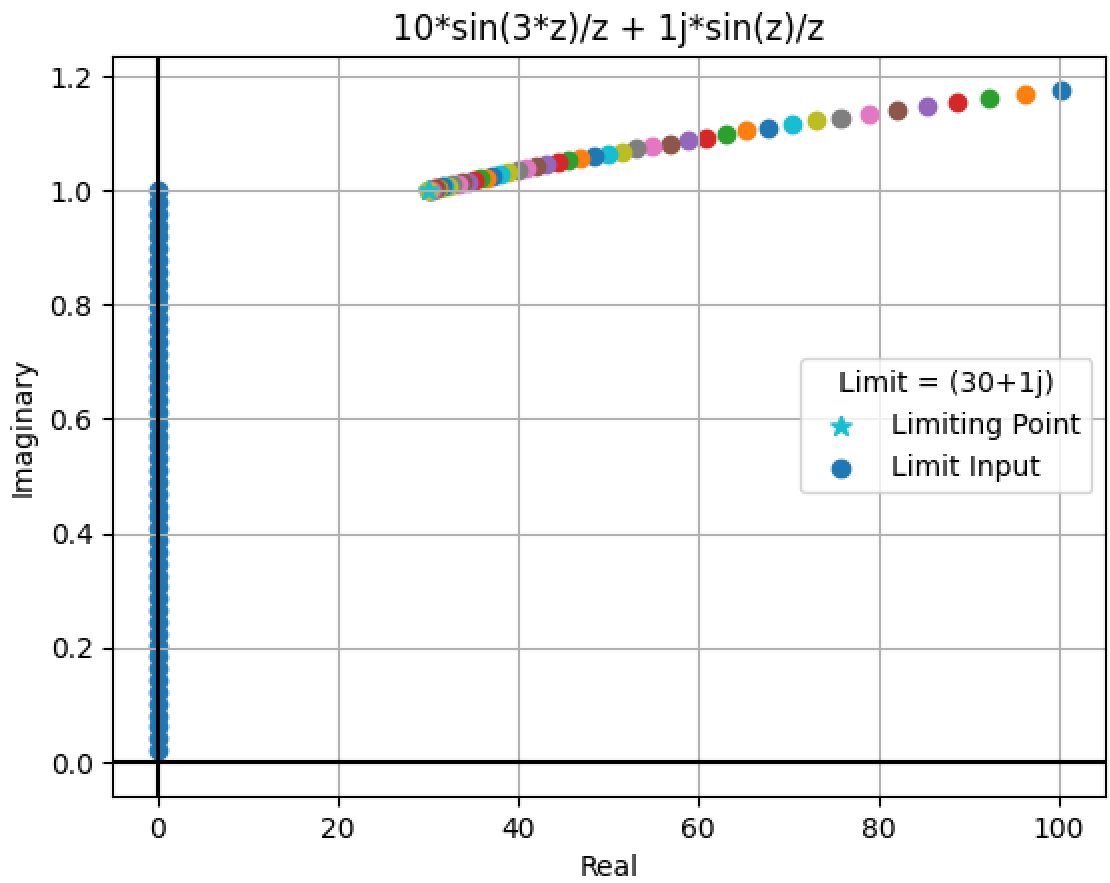
$$(1 + \frac{4}{z})^z + i\frac{7z}{z+4}$$

In [60]:
```
limitingPoints("(1 + 4/z)**z + 1j*7*z/(z+4)",zStart=1j,zLimit=0)
```



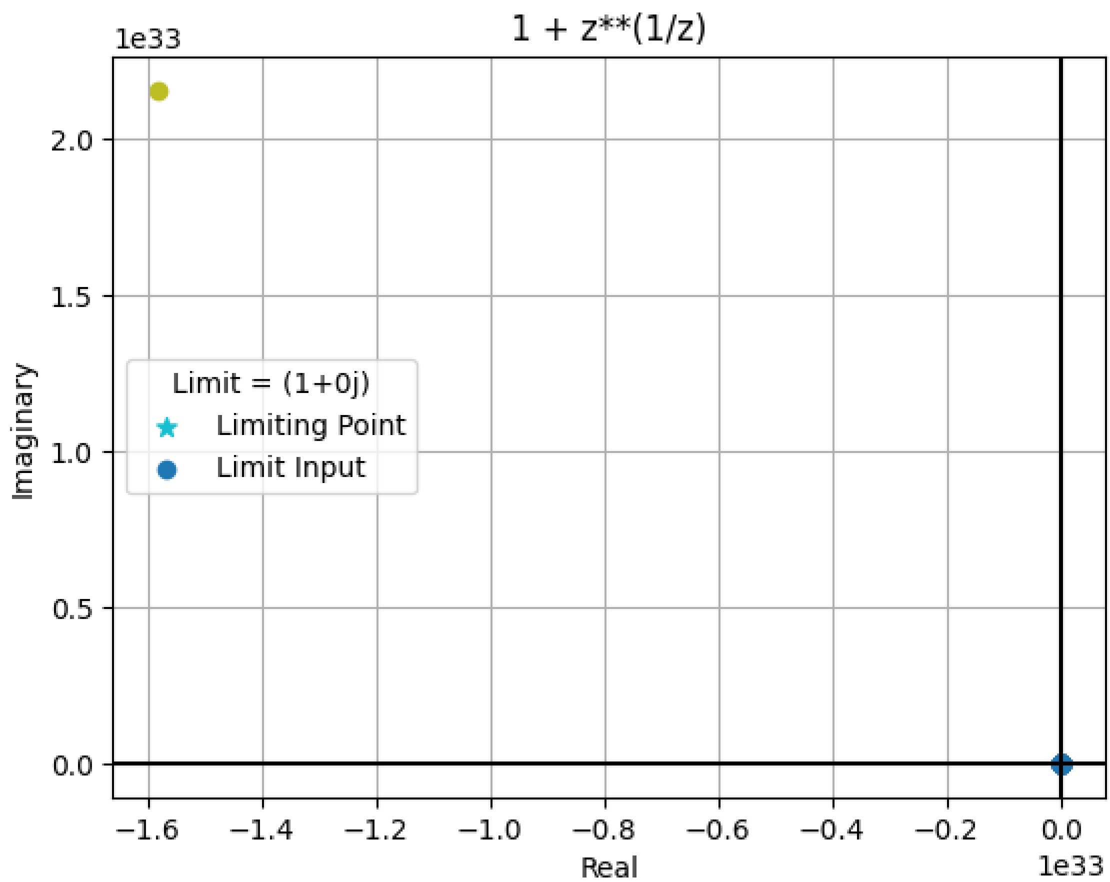
$$10\frac{\sin 3z}{z} + i\frac{\sin z}{z}$$

In [62]:
```
limitingPoints("10*sin(3*z)/z + 1j*sin(z)/z",zStart=1j,zLimit=0)
```



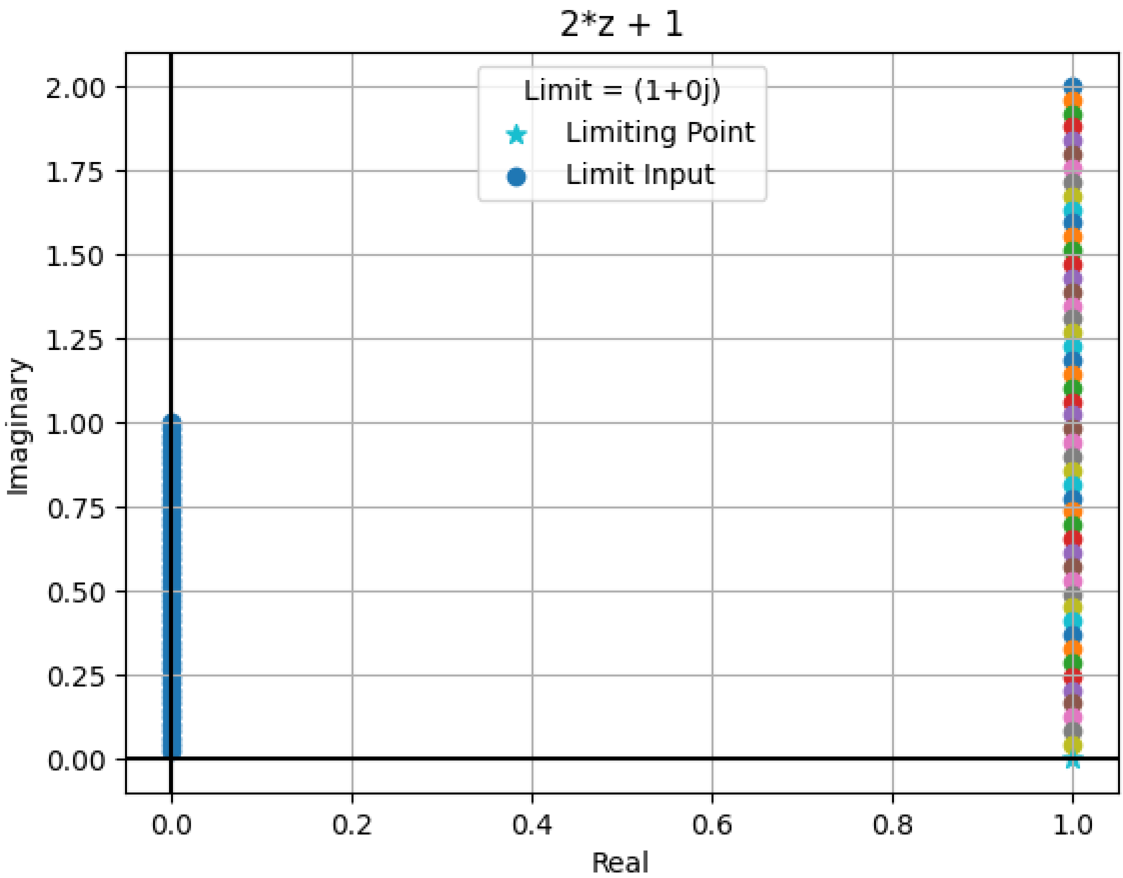
$$1 + z^{\frac{1}{z}}$$

In [63]:
```
limitingPoints("1 + z**(1/z)",zStart=1j,zLimit=0)
```



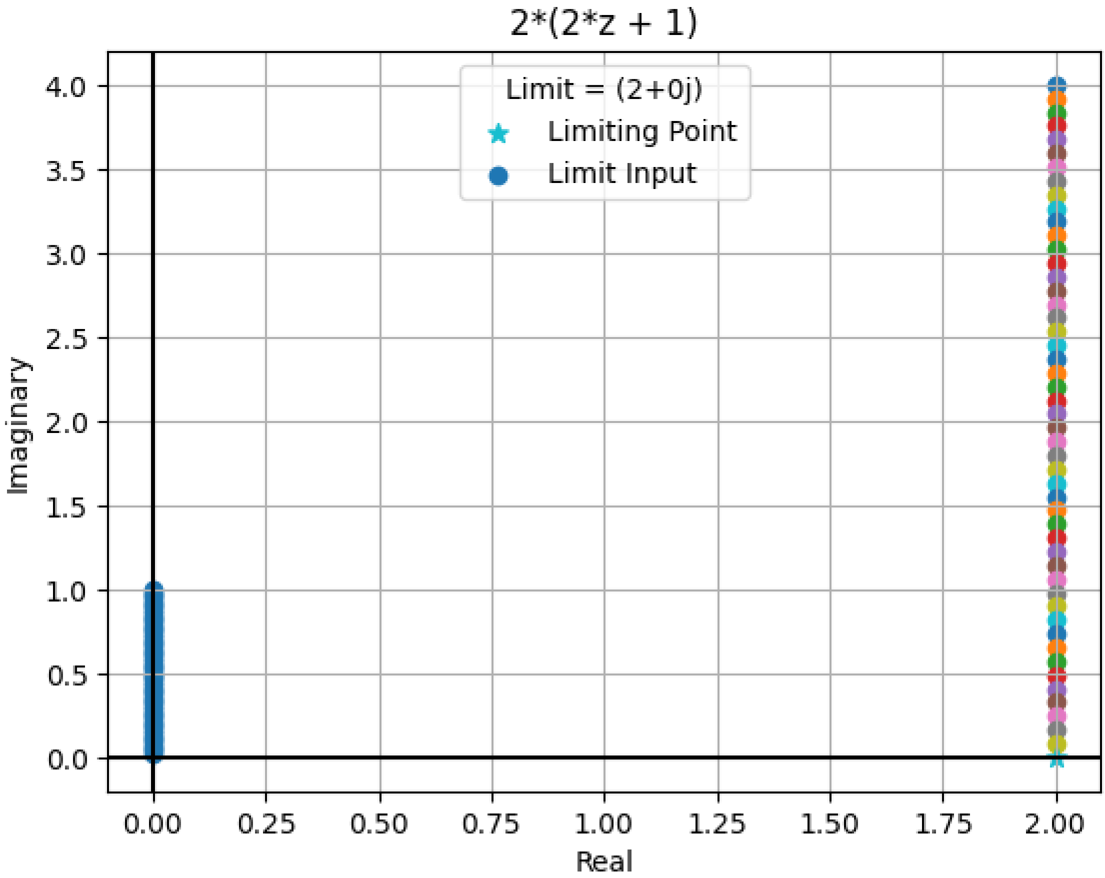
> WAP to verify using examples that, lim(czn) = c lim(zn)

In [65]:
```python
# zn = (2z+1)
limitingPoints("2*z + 1",zStart=1j,zLimit=0)
```



In [68]:
```python
# c = 2
# 2*1 = 2
```

In [69]:
```python
limitingPoints("2*(2*z + 1)",zStart=1j,zLimit=0)
```


2*(2*z + 1)
Limit = (2+0j)
Limiting Point
Limit Input
Imaginary
Real
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

In [ ]: