PHYS 3790 Challenge 2
Newton Basins

X.J.J. Xu

Sept 24, 2018

1 Newton's Method

The program used to solve for the roots is displayed below. For the following sections, the programs are also attached. Since Figure 1 shows the maps of a 5th degree polynomial, there are five distinct colors in the root map, indicating that there 5 roots. However, in Figure 2, there should be an indication of 4 roots existing but the epsilon is too large. If a smaller epsilon is used, then program would fail to execute. Figure 3 shows an attempt to reduce the size of epsilon but failed again, yielding a similar iteration map though. Figures 4 and 5 worked very well, displaying 3 distinct roots for the corresponding cubic functions.

```
# NEWTON'S METHOD

import numpy as np
import matplotlib.pyplot as plt

# def f(x): return x**5 - 1
# def df(x): return 5*x**4

# def f(x): return x**4 - 1
# def df(x): return 4*x**3

def f(x): return (x**2 - 1) / x**2
def df(x): return 2 / x**3

# def f(x): return x**3 + x/2 - 3/2
# def df(x): return 3*x**2 + 1/2

# def f(x): return x**3 + x**2/2 + x/2 - 1/2
# def df(x): return 3*x**2 + x + 1/2

# def f(x): return x**4 * ((5 + 4*x + x**2) / (1 + 4*x + 5*x**2))
# def df(x): return 20*z**3 * ((z**4 + 4*z**3 + 6*z**2 + 4*z + 1) / (5*z**2
```

```
# ------------
epsilon=0.001
# -------------
xmin = -1
xmax = 1
ymin = -1
ymax = 1
nx = 400
ny = 400
iterations=np.zeros((nx,ny))
zroots=np.zeros((nx,ny), dtype=np.complex128)
dx=(xmax-xmin)/(nx - 1)
dy=(ymax-ymin)/(ny - 1)
roots=[]
rootnums = np.zeros((nx,ny))

print("BEGIN...")

for i in range(nx):
    if i == 10: print("UNDERWAY...")
    if i == int(nx/2): print("HALF WAY THRU...")

    for j in range(ny):
        x=xmin+i*dx
        y=ymin+j*dy
        z=x+1j*y
        zold=z
        accuracy = 10**10

        while(accuracy > epsilon): # main loop, get root and record number
            try:
                z = z - f(z) / df(z)
            except:
                print('Error')
            accuracy = abs(z-zold)
            zold = z
            iterations[i][j] += 1

        zroots[i][j] = z # record root
        rootnum = -1 # add to list or classify root

        if len(roots) == 0:
            roots.append(z)
        for k in range(len(roots)):
```

```
            if abs(z - roots[k]) < 10*epsilon:
                rootnum=k
        if(rootnum == -1):
            roots.append(z)
            rootnums[i][j] = len(roots)-1
        else:
            rootnums[i][j] = rootnum

print("FINISHED...")
print("PLOTTING...")

plt.figure()
plt.figure(figsize=(16,8))
X,Y=np.meshgrid(np.arange(-1, 1, (xmax-xmin)/nx),np.arange(-1,1,(xmax-xmin)
plt.subplot(1,2,1)
plt.title('Iterations map')
plt.pcolormesh(X,Y,iterations,cmap=plt.cm.jet)
plt.subplot(1,2,2)
# $f(z) = z^4 - 1$
# $f(z) = (z^2 - 1) / z^2$
# $f(z) = z^3 + z/2 - 3/2$
# $f(z) = z^3 + z^2/2 + z/2 - 1/2$
# $f(z) = z^4 [(5 + 4z + z^2) / (1 + 4z + 5z^2)]$
plt.title('Root map $f(z) = (z^2 - 1) / z^2$')
plt.pcolormesh(X,Y,rootnums,cmap=plt.cm.jet)
plt.show()
```

Figure 1: $\epsilon = 0.001$

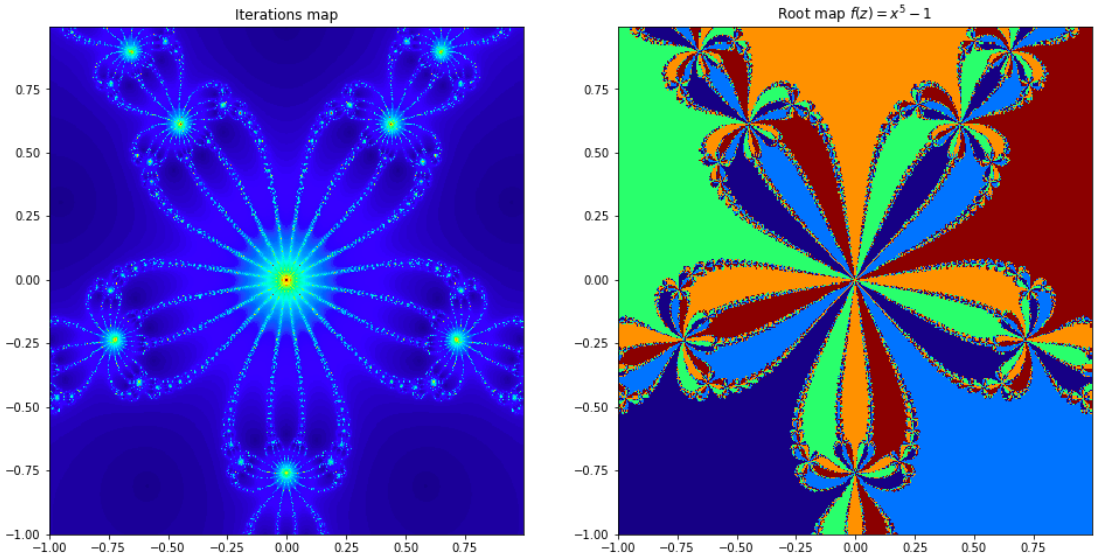


Figure 2: $\epsilon = 0.5$

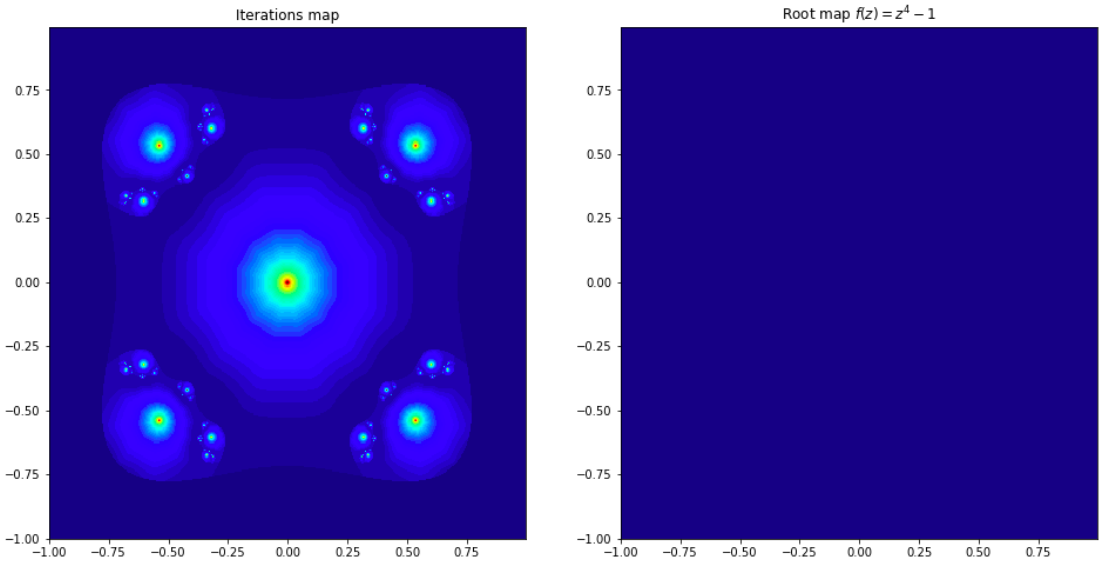


Figure 3: $\epsilon = 0.45$

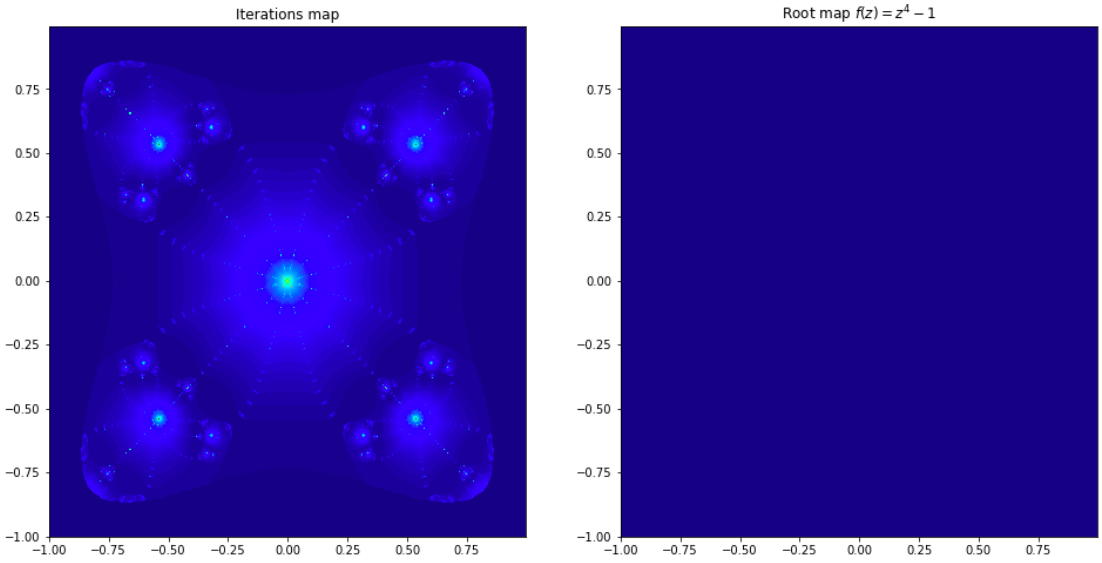


Figure 4: $\epsilon = 0.001$

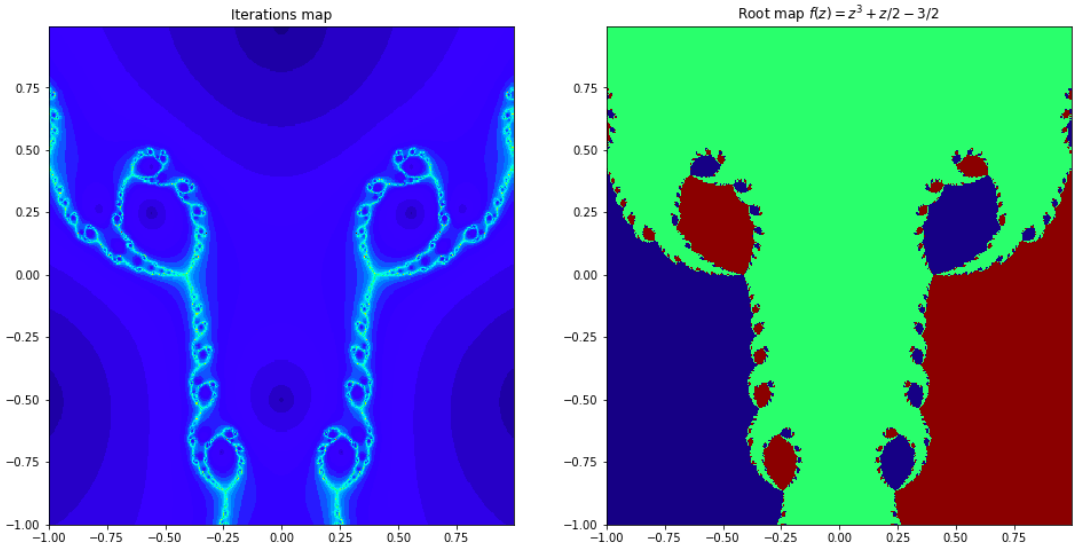


Figure 5: $\epsilon = 0.001$

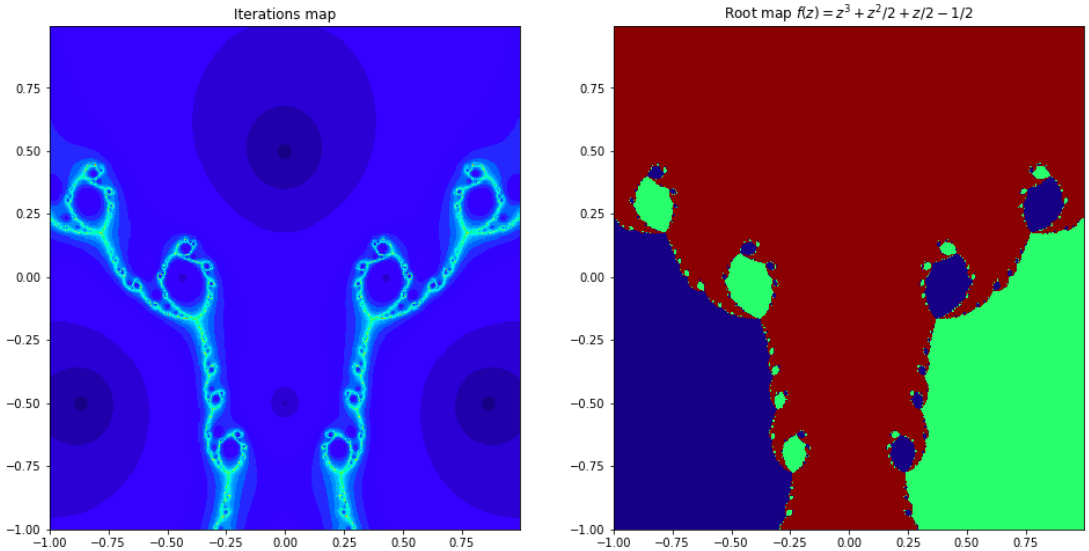
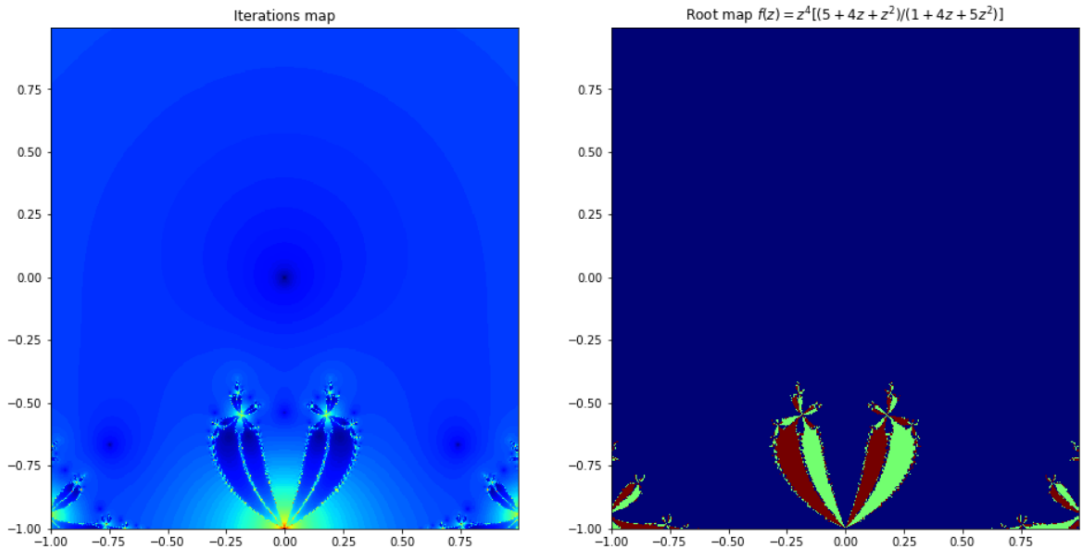


Figure 6: $\epsilon = 0.001$



2 Super Newton's Method

Figure 7 looks somewhat similar to Figure 1, both showing 5 roots. Figure 8 encountered a similar execution issue - the program would not rule with a smaller epsilon value. Figure 9 and Figure 10 are fairly similar, both the iteration and root maps display similar geometric patterns. Figure 11 shows 5 pods-like features whereas Figure 6 shows only 2.

```
# SUPER NEWTON'S METHOD

import numpy as np
import matplotlib.pyplot as plt

# def f(x): return x**5 - 1
# def df(x): return 5*x**4
# def ddf(x): return 20*x**3

# def f(x): return x**4 - 1
# def df(x): return 4*x**3
# def ddf(x): return 12*x**2

def f(x): return (x**2 - 1) / x**2
def df(x): return 2 / x**3
def ddf(x): return -6 / x**4

# def f(x): return x**3 + x/2 - 3/2
# def df(x): return 3*x**2 + 1/2
# def ddf(x): return 6*x

# def f(x): return x**3 + x**2/2 + x/2 - 1/2
# def df(x): return 3*x**2 + x + 1/2
# def ddf(x): return 6*x + 1

# def f(x): return x**4 * ((5 + 4*x + x**2) / (1 + 4*x + 5*x**2))
# def df(x): return 20*z**3 * ((z**4 + 4*z**3 + 6*z**2 + 4*z + 1) / (5*z**2
# def ddf(x): return 20*z**2 * ((15*z**6 + 60*z**5 + 101*z**4 + 96*z**3 + 5

# ----------
epsilon=0.8
# ----------
xmin = -1
xmax = 1
ymin = -1
ymax = 1
nx = 400
```

```
ny = 400
iterations=np.zeros((nx,ny))
zroots=np.zeros((nx,ny), dtype=np.complex128)
dx=(xmax-xmin)/(nx - 1)
dy=(ymax-ymin)/(ny - 1)
roots=[]
rootnums = np.zeros((nx,ny))

All_roots = []

print("BEGIN...")

for i in range(nx):
    for j in range(ny):

        x=xmin+i*dx
        y=ymin+j*dy
        z=x+1j*y
        zold=z
        accuracy=10**10

        while(accuracy > epsilon): # main loop, get root and record number
            try:
                z = z - f(z) / df(z) - (f(z)**2 * ddf(z)) / (2 * df(z)**3)
            except:
                print("Exception")
                print(i,j,x,y,z,df(z))
            accuracy=abs(z-zold)
            zold = z
            iterations[i][j] += 1

        zroots[i][j] = z #record root
        rootnum = -1 # add to list or classify root

        if len(roots) == 0:
            roots.append(z)
        for k in range(len(roots)):
            if abs(z - roots[k]) < 10*epsilon:
                rootnum=k
        if(rootnum == -1):
            roots.append(z)
            rootnums[i][j]=len(roots)-1
        else:
            rootnums[i][j]=rootnum
```

```
print("FINISHED...")
print("PLOTTING...")

plt.figure()
plt.figure(figsize=(16,8))
X,Y=np.meshgrid(np.arange(-1, 1, (xmax-xmin)/nx),np.arange(-1, 1, (ymax - y
plt.subplot(1,2,1)
plt.title('Iterations map')
plt.pcolormesh(X,Y,iterations,cmap=plt.cm.jet)
plt.subplot(1,2,2)
# $f(z) = z^4 - 1$
# $f(z) = (z^2 - 1) / z^2$
# $f(z) = z^3 + z/2 - 3/2$
# $f(z) = z^3 + z^2/2 + z/2 - 1/2$
# $f(z) = z^4 [(5 + 4z + z^2) / (1 + 4z + 5z^2)]$
plt.title('Root map $f(z) = z^4 - 1$')
plt.pcolormesh(X,Y,rootnums,cmap=plt.cm.jet)
plt.show()
```

Figure 7: $\epsilon = 0.001$

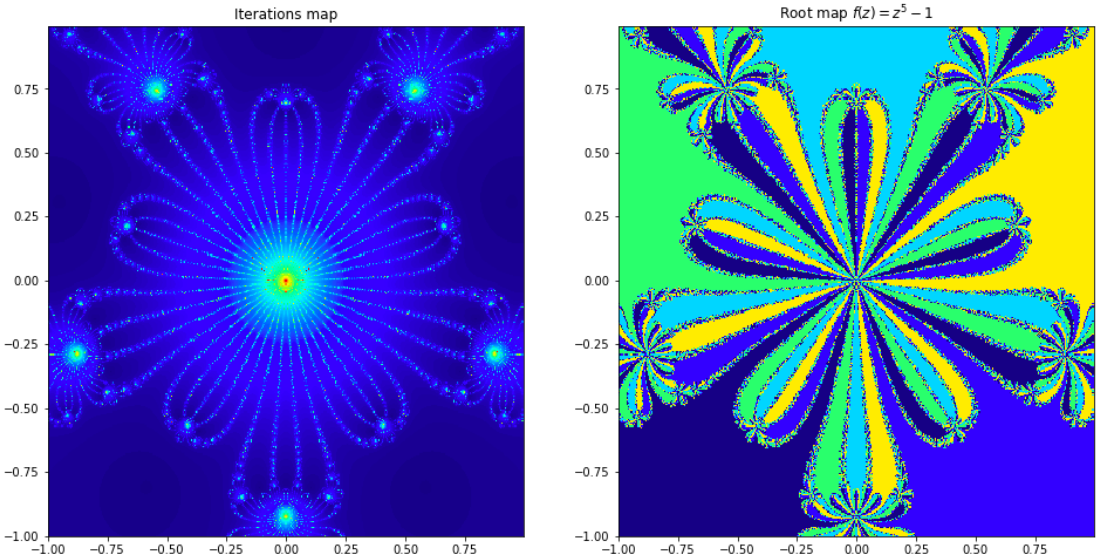


Figure 8: $\epsilon = 0.8$

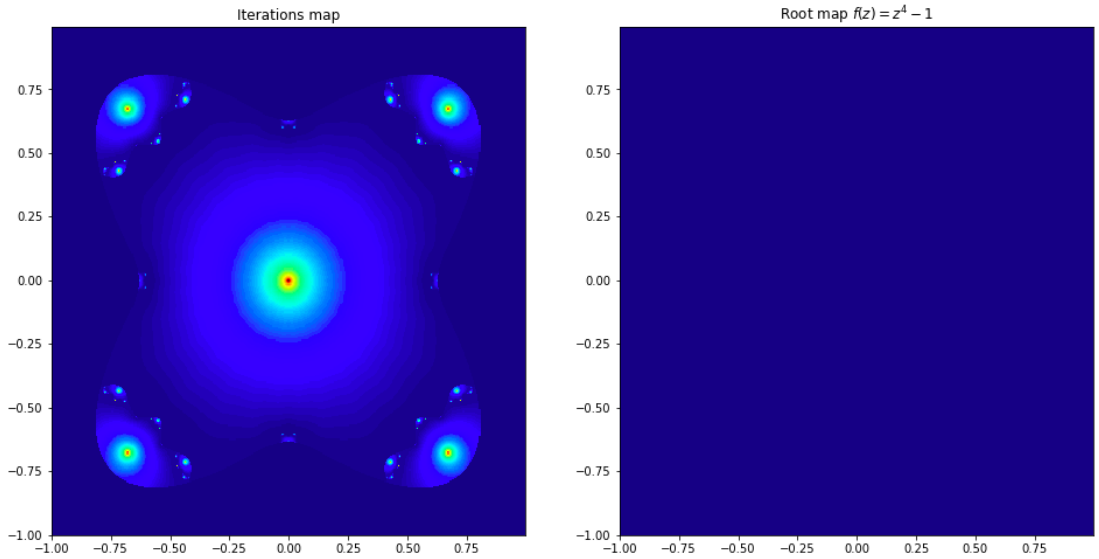


Figure 9: $\epsilon = 0.001$

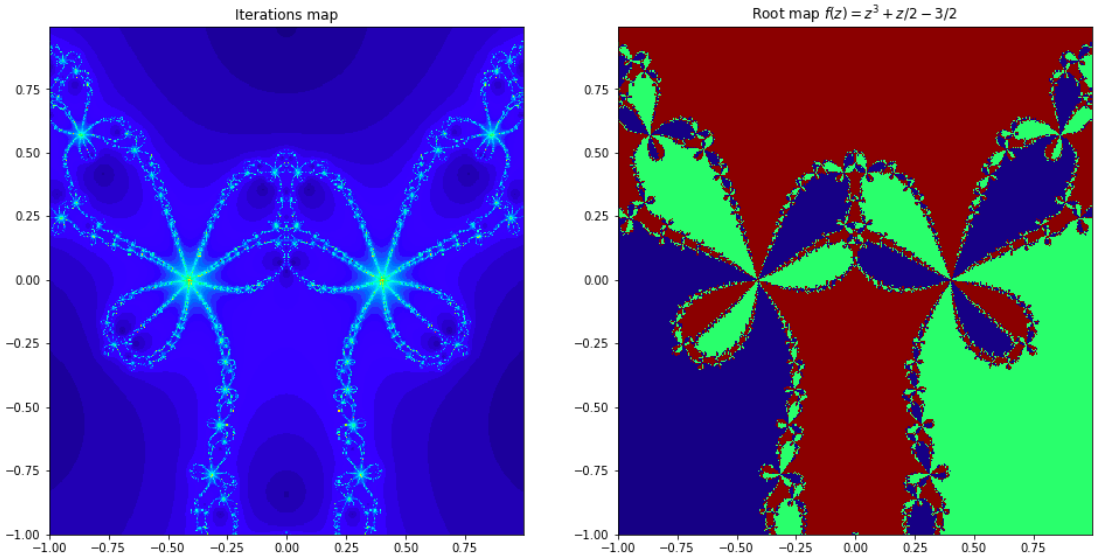


Figure 10: $\epsilon = 0.001$

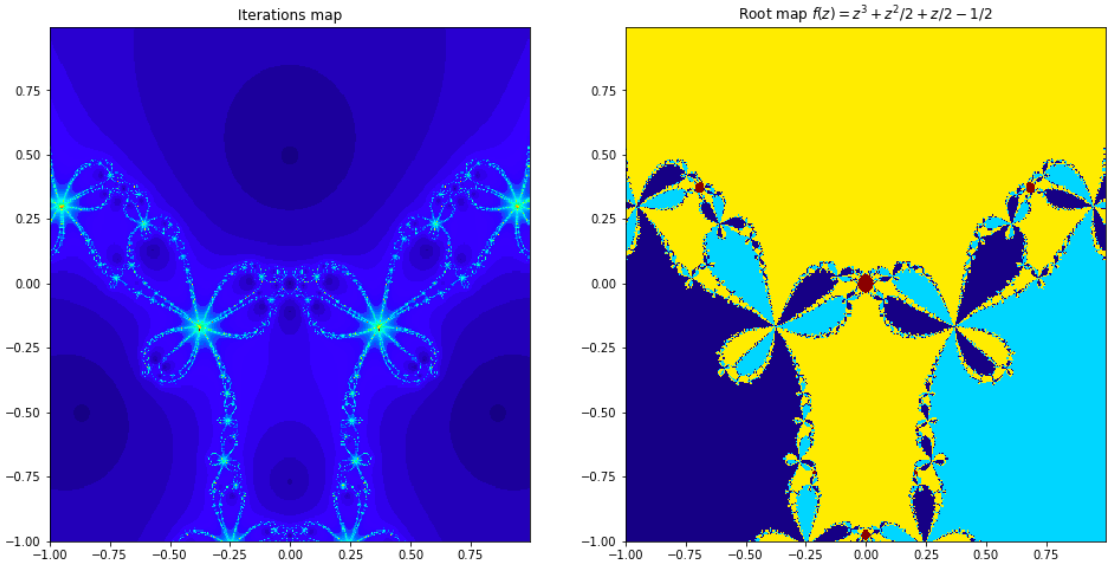
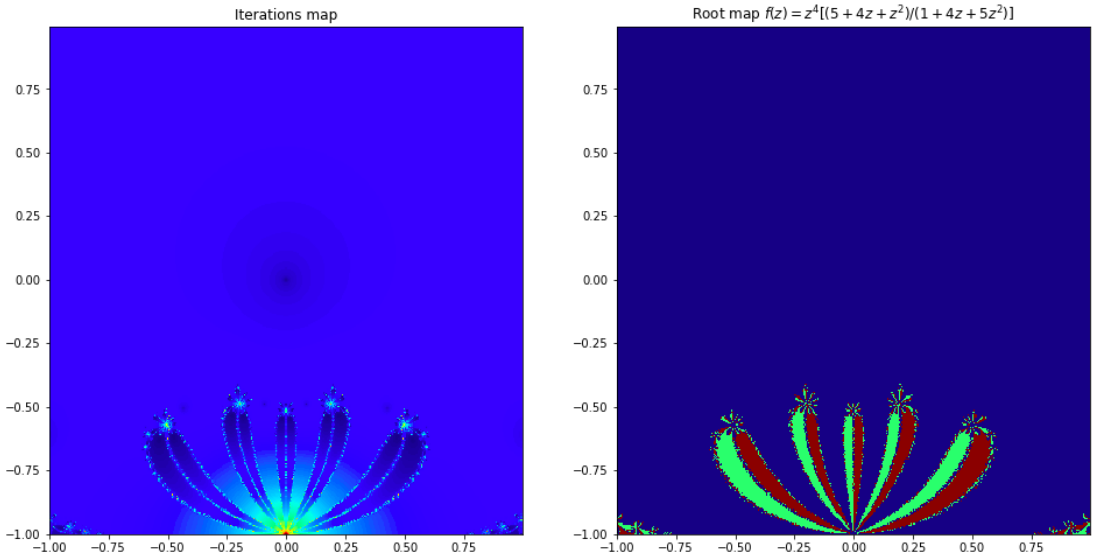


Figure 11: $\epsilon = 0.001$



3 Halley's Method

Figure 12 and 13 are highly distinct from the first two approaches. There are straight lines partitioning the complex space. Figure 14 is quite interesting because only this method worked for the function $f(z) = \frac{z^2-1}{z^2}$ while the programs of the rest of the methods would not execute. The are two roots, as expected, that clearly separated in the root map by a horizontal line. Figure 16 looks highly different from Figure 6 and 11.

```
# HALLEY'S METHOD

import numpy as np
import matplotlib.pyplot as plt
from numpy import sqrt, absolute

# def f(x): return x**5 - 1
# def df(x): return 5*x**4
# def ddf(x): return 20*x**3

# def f(x): return x**4 - 1
# def df(x): return 4*x**3
# def ddf(x): return 12*x**2

# def f(x): return (x**2 - 1) / x**2
# def df(x): return 2 / x**3
# def ddf(x): return -6 / x**4

def f(x): return x**3 + x/2 - 3/2
def df(x): return 3*x**2 + 1/2
def ddf(x): return 6*x

# def f(x): return x**3 + x**2/2 + x/2 - 1/2
# def df(x): return 3*x**2 + x + 1/2
# def ddf(x): return 6*x + 1

# def f(x): return x**4 * ((5 + 4*x + x**2) / (1 + 4*x + 5*x**2))
# def df(x): return 20*z**3 * ((z**4 + 4*z**3 + 6*z**2 + 4*z + 1) / (5*z**2
# def ddf(x): return 20*z**2 * ((15*z**6 + 60*z**5 + 101*z**4 + 96*z**3 + 5

def g(x): return f(x) / sqrt(absolute(df(x)))
def dg(x): return (2*df(x)**2 - f(x)*ddf(x)) / (2*df(x)*sqrt(absolute(df(x)

# ------------
epsilon=1.5
# ------------
```

```
xmin = -1
xmax = 1
ymin = -1
ymax = 1
nx = 400
ny = 400
iterations=np.zeros((nx,ny))
zroots=np.zeros((nx,ny), dtype=np.complex128)
dx=(xmax-xmin)/(nx - 1)
dy=(ymax-ymin)/(ny - 1)
roots=[]
rootnums = np.zeros((nx,ny))

All_roots = []

print("BEGIN...")

for i in range(nx):
    if i == 10: print("UNDERWAY...")
    if i == int(nx/2): print("HALF WAY THRU...")
    for j in range(ny):
        # print(i,j)
        x=xmin+i*dx
        y=ymin+j*dy
        z=x+1j*y
        zold=z
        accuracy=10**10

        while(accuracy > epsilon): # main loop, get root and record number
            try:
                z = z - g(z) / dg(z)
            except:
                print(z, g(z), dg(z))
            accuracy=abs(z-zold)
            zold = z
            iterations[i][j] += 1

        zroots[i][j] = z #record root
        rootnum = -1 # add to list or classify root

        if len(roots) == 0:
            roots.append(z)
        for k in range(len(roots)):
            if abs(z - roots[k]) < 10*epsilon:
```

```
                rootnum=k
            if(rootnum == -1):
                roots.append(z)
                rootnums[i][j]=len(roots)-1
            else:
                rootnums[i][j]=rootnum

print("FINISHED...")
print("PLOTTING...")

plt.figure()
plt.figure(figsize=(16,8))
X,Y=np.meshgrid(np.arange(-1, 1, (xmax-xmin)/nx),np.arange(-1, 1, (ymax - y
plt.subplot(1,2,1)
plt.title('Iterations map')
plt.pcolormesh(X,Y,iterations,cmap=plt.cm.jet)
plt.subplot(1,2,2)
# $f(z) = z^4 - 1$
# $f(z) = (z^2 - 1) / z^2$
# $f(z) = z^3 + z/2 - 3/2$
# $f(z) = z^3 + z^2/2 + z/2 - 1/2$
# $f(z) = z^4 [(5 + 4z + z^2) / (1 + 4z + 5z^2)]$
plt.title('Root map $f(z) = z^3 + z/2 - 3/2$')
plt.pcolormesh(X,Y,rootnums,cmap=plt.cm.jet)
plt.show()
```

Figure 12: $\epsilon = 0.001$

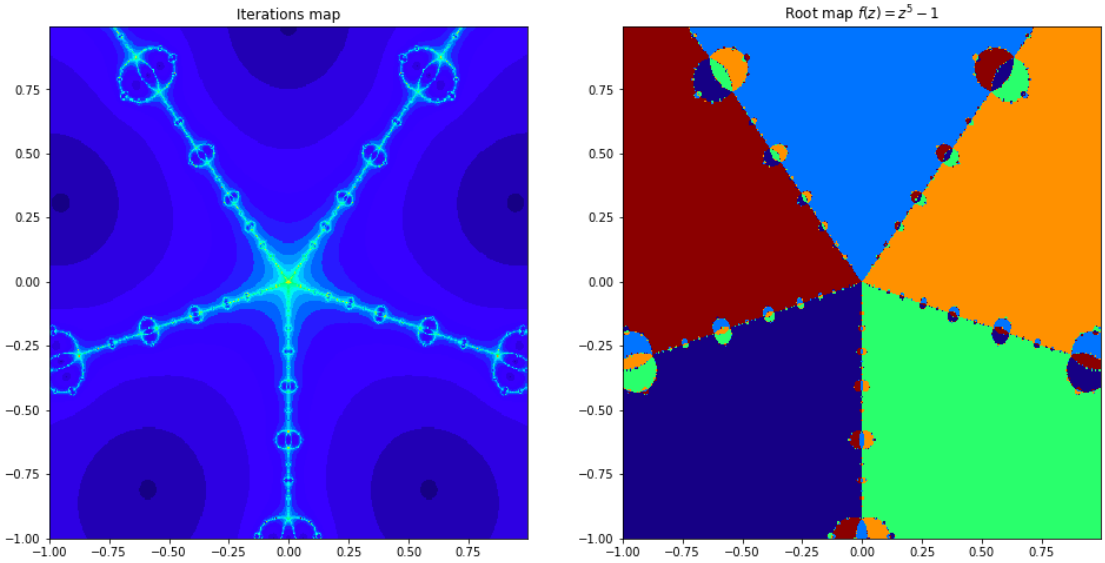


Figure 13: $\epsilon = 0.001$

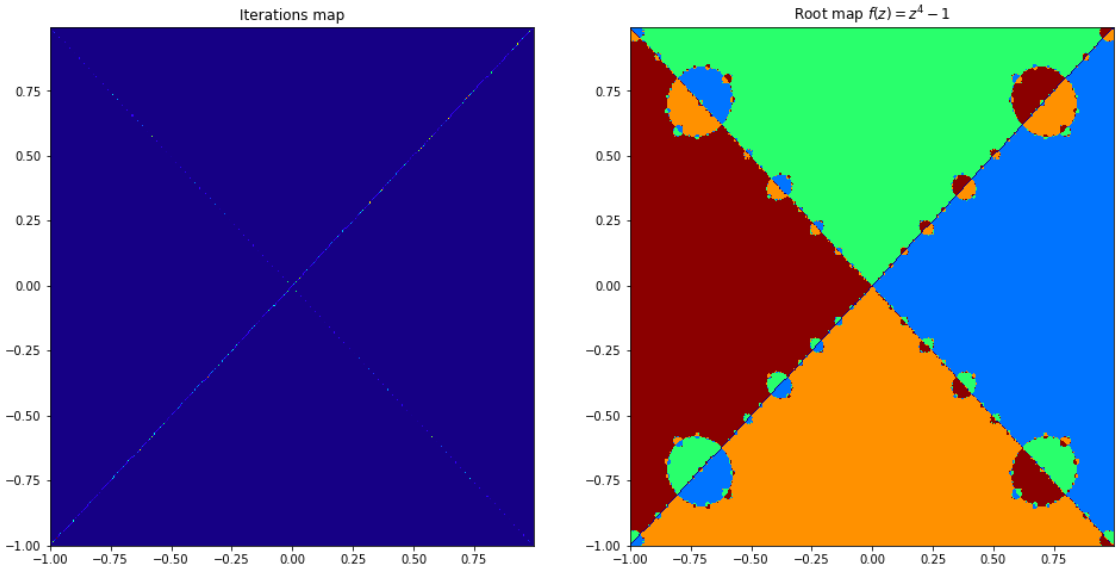


Figure 14: $\epsilon = 0.001$

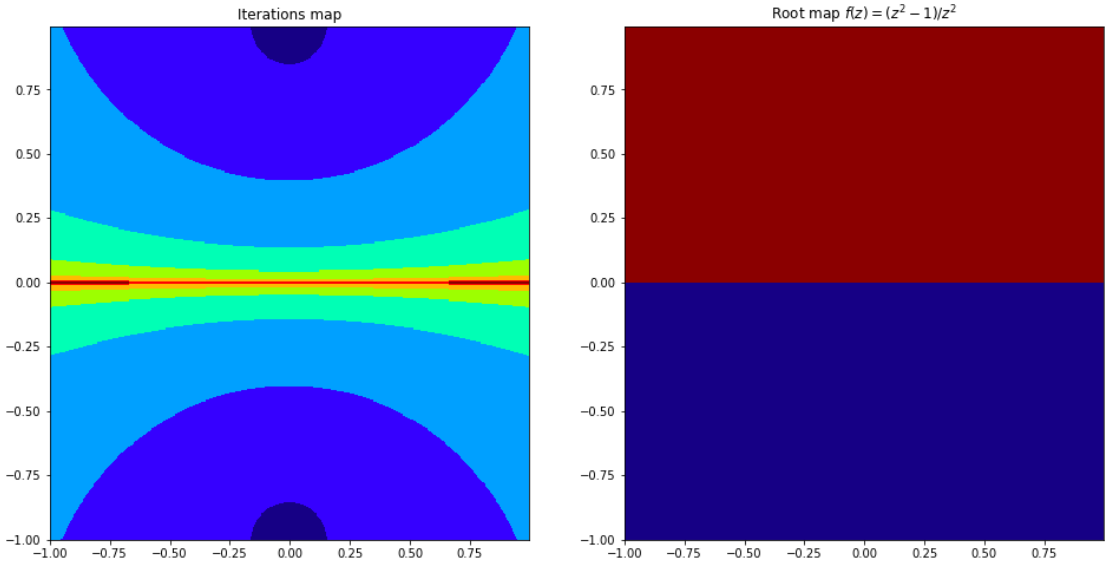


Figure 15: $\epsilon = 0.001$

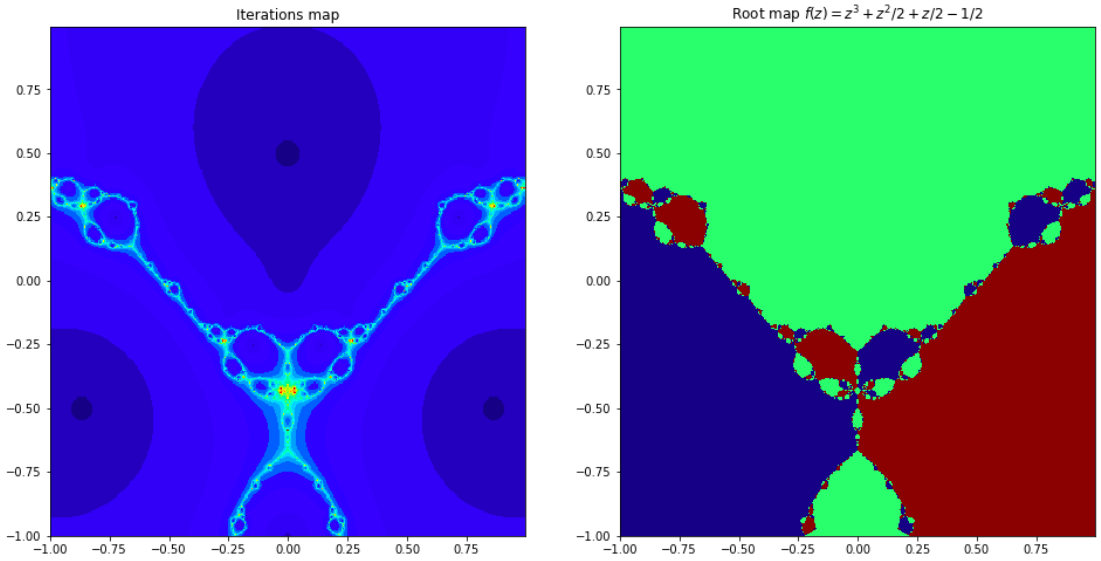
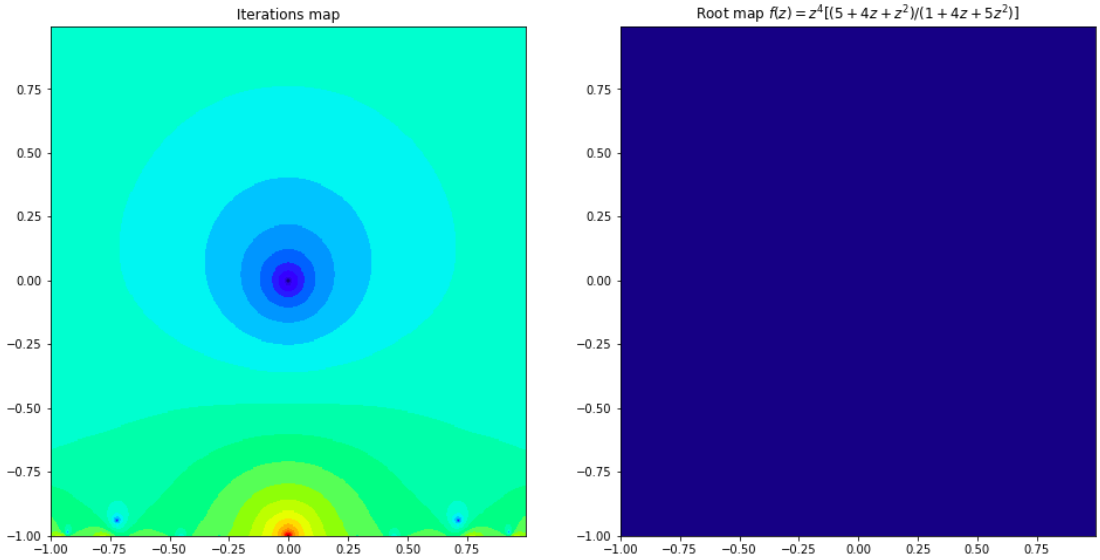


Figure 16: $\epsilon = 0.001$



4 Neta's Method

Unfortunately this method did not work for most of the functions. For the ones that worked, the root map looks very broken up and disrupted - not sure why that is. The iteration maps in Figure 17 - 19 retain some resemblance of each other as they're all cubic functions.

```
# NETA'S METHOD

import numpy as np
import matplotlib.pyplot as plt
from numpy import sqrt, absolute

# def f(x): return x**5 - 1
# def df(x): return 5*x**4
# def ddf(x): return 20*x**3

def f(x): return x**4 - 1
def df(x): return 4*x**3
def ddf(x): return 12*x**2

# def f(x): return (x**2 - 1) / x**2
# def df(x): return 2 / x**3
# def ddf(x): return -6 / x**4

# def f(x): return x**3 + x/2 - 3/2
# def df(x): return 3*x**2 + 1/2
# def ddf(x): return 6*x

# def f(x): return x**3 + x**2/2 + x/2 - 1/2
# def df(x): return 3*x**2 + x + 1/2
# def ddf(x): return 6*x + 1

# def f(x): return x**4 * ((5 + 4*x + x**2) / (1 + 4*x + 5*x**2))
# def df(x): return 20*z**3 * ((z**4 + 4*z**3 + 6*z**2 + 4*z + 1) / (5*z**2
# def ddf(x): return 20*z**2 * ((15*z**6 + 60*z**5 + 101*z**4 + 96*z**3 + 5

def yn(x): return x - f(x) / df(x)
def zn(x): return yn(x) - f(yn(x)) / (df(x) * (1 - f(yn(x))/f(x))**2)

# ------------
epsilon=0.1
# -------------
xmin = -1
xmax = 1
```

```
ymin = -1
ymax = 1
nx = 400
ny = 400
iterations=np.zeros((nx,ny))
zroots=np.zeros((nx,ny), dtype=np.complex128)
dx=(xmax-xmin)/(nx - 1)
dy=(ymax-ymin)/(ny - 1)
roots=[]
rootnums = np.zeros((nx,ny))

All_roots = []

print("BEGIN...")

for i in range(nx):
    if i == 1: print("UNDERWAY...")
    if i == int(nx/2): print("HALF WAY THRU...")
    if i == int(3*nx/2): print("ALMOST THERE..")
    for j in range(ny):

        x=xmin+i*dx
        y=ymin+j*dy
        z=x+1j*y
        zold=z
        accuracy=10**10

        while(accuracy > epsilon): # main loop, get root and record number
            try:
                z = zn(z) - f(zn(x)) / (df(x) * (1 - f(yn(x))/f(x) - f(zn(x
            except:
                print('fuck')
            accuracy=abs(z-zold)
            zold = z
            iterations[i][j] += 1

        zroots[i][j] = z #record root
        rootnum = -1 # add to list or classify root

        if len(roots) == 0:
            roots.append(z)
        for k in range(len(roots)):
            if abs(z - roots[k]) < 10*epsilon:
                rootnum=k
```

```
            if(rootnum == -1):
                roots.append(z)
                rootnums[i][j]=len(roots)-1
            else:
                rootnums[i][j]=rootnum

print("FINISHED...")
print("PLOTTING...")

plt.figure()
plt.figure(figsize=(16,8))
X,Y=np.meshgrid(np.arange(-1, 1, (xmax-xmin)/nx),np.arange(-1, 1, (ymax - y
plt.subplot(1,2,1)
plt.title('Iterations map')
plt.pcolormesh(X,Y,iterations,cmap=plt.cm.jet)
plt.subplot(1,2,2)
# $f(z) = z^4 - 1$
# $f(z) = (z^2 - 1) / z^2$
# $f(z) = z^3 + z/2 - 3/2$
# $f(z) = z^3 + z^2/2 + z/2 - 1/2$
# $f(z) = z^4 [(5 + 4z + z^2) / (1 + 4z + 5z^2)]$
plt.title('Root map $f(z) = z^4 [(5 + 4z + z^2) / (1 + 4z + 5z^2)]$')
plt.pcolormesh(X,Y,rootnums,cmap=plt.cm.jet)
plt.show()
```

Figure 17: $\epsilon = 0.001$

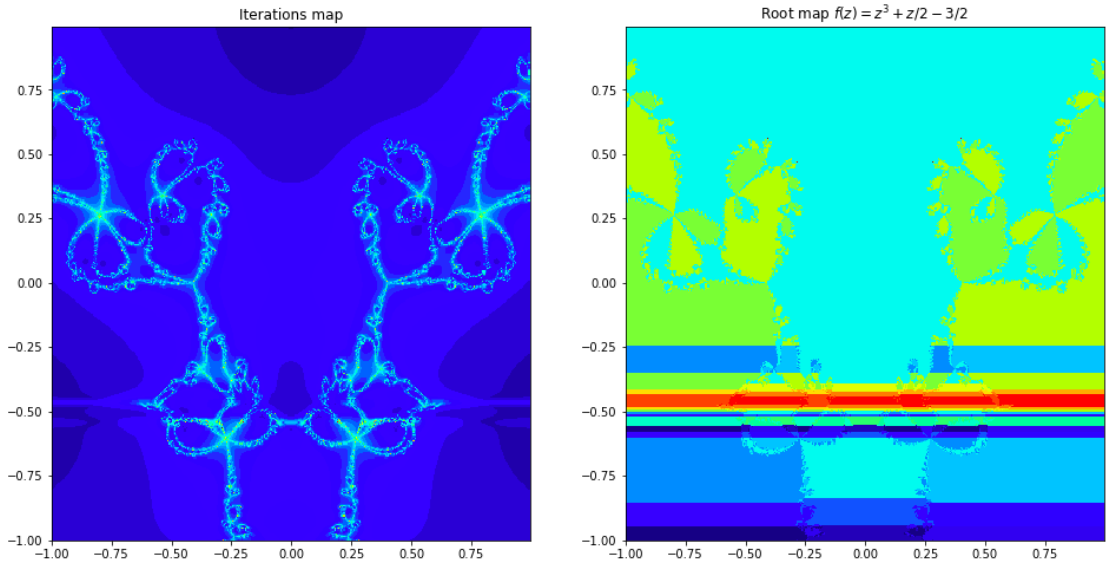


Figure 18: $\epsilon = 0.001$

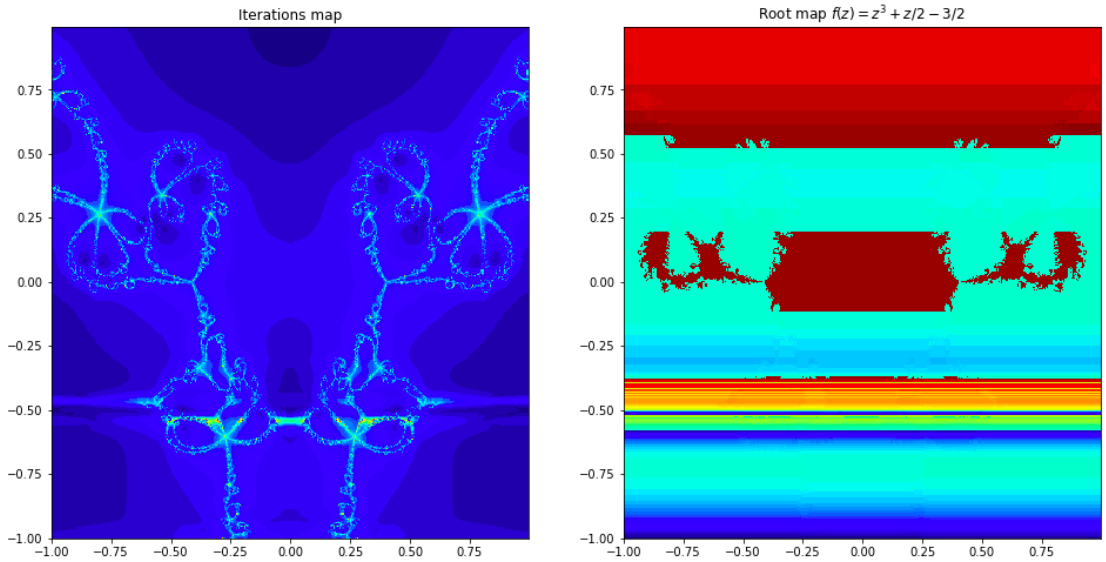


Figure 19: $\epsilon = 0.001$