Python. Szukanie pierwiastków

B.Kowal

17 listopada 2021

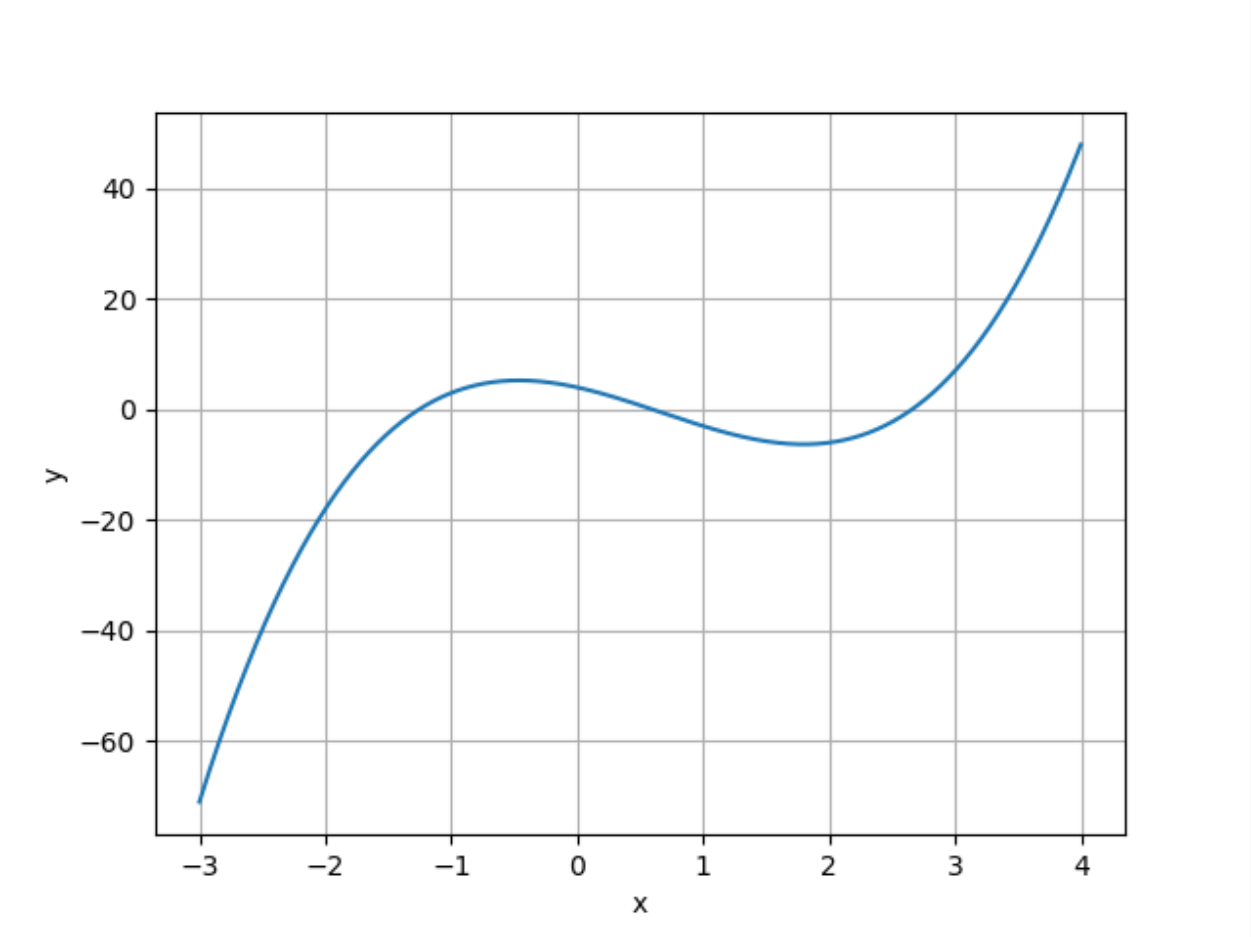
Pierwiastek funkcji w przedziale, metoda Riddera

```python
from scipy import optimize
def f(x):
  return 2*x**3 - 4*x**2 - 5*x + 4

root1 = optimize.ridder(f,-2,-1)
root2 = optimize.ridder(f,-1,2)
root3 = optimize.ridder(f,2,3)
print(root1,'\n',root2,'\n',root3)
print(f(root1),'\n',f(root2),'\n',f(root3))
```

```
-1.2564880255530644
 0.5989528218683059
 2.6575352036837 92
-1.4504841772122745e-11
 -3.3058000781238 66e-12
 -6.4623861817381 11e-12
```

Pierwiastek funkcji w przedziale, metoda Riddera



Pierwiastek funkcji w okolicy punktu, metoda Newtona

```python
from scipy import optimize
def f(x):
  return 2*x**3 - 4*x**2 - 5*x + 4
def fp(x):
  return 6*x**2 - 8*x - 5

root1 = optimize.newton(f,-2,fprime=fp)
root2 = optimize.newton(f,0,fprime=fp)
root3 = optimize.newton(f,2,fprime=fp)
print(root1,'\n',root2,'\n',root3)
print(f(root1),'\n',f(root2),'\n',f(root3))
```

```
-1.2564880255520656
 0.5989528218678731
 2.657535203684193
1.776356839400250 5e-15
 0.0
 3.552713678800501e-15
```

Pierwiastki funkcji wielu zmiennych

```
import numpy as np
from scipy import optimize
def f(x):
  return [4*x[0]**2  - x[1] - 3,  6*x[1]**2 - 2*x[0] - 3]

d=0.1; a=-2.0; b=2.0
x = np.arange(a,b,d)
y = np.arange(a,b,d)
xyarray = np.array([[a,b] for a in x for b in y])
roots=np.empty((0,2))
for b in xyarray:
  root=optimize.root(f,b)
  if root.success:
    roots=np.append(roots,[np.round(root.x,5)],axis=0)
unique_roots = np.unique(roots, axis=0)
for a in unique_roots:
  print(a,f(a))
```