Fundamental Rootfinding

Matthew Hefner

10/26/2018

Problem 1 (Quadratic Roots)

```
def quad_poly(a,b,c):
  print("For the function:", a, "x^2 +", b, "x +", c)
  if (a == 0) & (b == 0):
    if (c == 0):
      print("All reals")
    else:
      print("No solution")
  elif (a == 0):
    print(- c / b)
  elif (b ** 2 - 4 * a * c < 0):
    print("No real solution")
  elif (b >= 0):
    print((- b - (b ** 2 - 4 * a * c) ** (1/2)) / (2 * a))
    print(2 * c / (-b - (b ** 2 - 4 * a * c) ** (1/2)))
  else:
    print((- b + (b ** 2 - 4 * a * c) ** (1/2)) / (2 * a))
    print(2 * c / (-b + (b ** 2 - 4 * a * c) ** (1/2)))

quad_poly(0,0,2)
```

```
For the function: 0 x^2 + 0 x + 2
No solution
```

```
quad_poly(0,3,-10)
```

```
For the function: 0 x^2 + 3 x + -10
3.3333333333333335
```

```
quad_poly(2,-10,5)
```

```
For the function: 2 x^2 + -10 x + 5
4.436491673103708
0.5635083268962916
```

```
quad_poly(12 / 10, 5 * 10 ** 20, -2 / 1000)
```

```
For the function: 1.2 x^2 + 500000000000000000000 x + -0.002
-4.166666666666667e+20
4.000000000000004e-24
```

```
quad_poly(2, 10, 32)
```

```
For the function: 2 x^2 + 10 x + 32
No real solution
```

$f1 := x \to 2;$
$f2 := x \to 3 \cdot x - 10;$
$f3 := x \to 2 \cdot x^2 - 10 \cdot x + 5;$
$f4 := x \to \left(\frac{12.0}{10.0} \cdot x^2 + \left(5 \cdot 10^{20}\right) \cdot x - \frac{2}{1000} \right);$
$f5 := x \to 2 \cdot x^2 + 10 \cdot x + 32;$

$$f1 := x \mapsto 2$$
$$f2 := x \mapsto 3\,x - 10$$
$$f3 := x \mapsto 2\,x^2 - 10\,x + 5$$
$$f4 := x \mapsto 1.200000000\,x^2 + 500000000000000000000\,x - \frac{1}{500}$$
$$f5 := x \mapsto 2\,x^2 + 10\,x + 32 \qquad \textbf{(1)}$$

$\{ fsolve(f1(x) = 0, x) \}$

$$\varnothing \qquad \textbf{(2)}$$

$\{ fsolve(f2(x) = 0, x) \}$

$$\{3.333333333\} \qquad \textbf{(3)}$$

$\{ fsolve(f3(x) = 0, x) \}$

$$0.5635083269,\ 4.436491673 \qquad \textbf{(4)}$$

$\{ fsolve(f4(x) = 0, x) \}$

$$-4.166666667\ 10^{20},\ 4.000000000\ 10^{-24} \qquad \textbf{(5)}$$

$\{ fsolve(f5(x) = 0, x) \}$

$$\varnothing \qquad \textbf{(6)}$$

Figure 1: Maple Results for Problem 1

Problem 2 (Bisection Method)

```
import math
def f(x):
  return((x**3.0) - (2.0 * math.sin(x)))

def g(x):
  return(2.0 - ((x**2.0) * math.exp(-.385 * x)))
def bisectf(a, b, nmax, eps):
  fa = f(a)
  fb = f(b)
  print("Begin")
  if fa * fb > 0:
    print("\ta =", a)
    print("\tb =", b)
    print("\tfa =", fa)
    print("\tfb =", fb)
    print("\t\tThe function has the same signs at a and b.  End")
    return
  error =  b - a
  for n in range(0, nmax + 1):
    error = error / 2.0
    c = a + error
    fc = f(c)
    print("\tn =", n)
    print("\t\tc = ", c)
    print("\t\tfc =", fc)
    print("\t\tError =", error)
    if abs(error) < eps:
      print("\t\t\tConvergence!  End")
      return
    if fa * fc < 0:
      b = c
      fb = fc
    else:
      a = c
      fa = fc
def bisectg(a, b, nmax, eps):
  fa = g(a)
  fb = g(b)
  print("Begin")
  if fa * fb > 0:
    print("\ta =", a)
    print("\tb =", b)
    print("\tfa =", fa)
    print("\tfb =", fb)
    print("\t\tThe function has the same signs at a and b.  End")
    return
  error =  b - a
  for n in range(0, nmax + 1):
    error = error / 2.0
    c = a + error
    fc = g(c)
```

```python
    print("\tn =", n)
    print("\t\tc = ", c)
    print("\t\tfc =", fc)
    print("\t\tError =", error)
    if abs(error) < eps:
      print("\t\t\tConvergence!  End")
      return
    if fa * fc < 0:
      b = c
      fb = fc
    else:
      a = c
      fa = fc
bisectf(-2.0, 3.0, 100, 10**-10)
```

```
  Begin
    n = 0
        c =  0.5
        fc = -0.833851077208406
        Error = 2.5
    n = 1
        c =  1.75
        fc = 3.391403106252126
        Error = 1.25
    n = 2
        c =  1.125
        fc = -0.38070706319819037
        Error = 0.625
    n = 3
        c =  1.4375
        fc = 0.9882006024674768
        Error = 0.3125
    n = 4
        c =  1.28125
        fc = 0.18655499043684554
        Error = 0.15625
    n = 5
        c =  1.203125
        fc = -0.12479861550947025
        Error = 0.078125
    n = 6
        c =  1.2421875
        fc = 0.023747855460964473
        Error = 0.0390625
    n = 7
        c =  1.22265625
        fc = -0.05228317258085524
        Error = 0.01953125
    n = 8
        c =  1.232421875
        fc = -0.014710216242693086
        Error = 0.009765625
    n = 9
        c =  1.2373046875
```

```
    fc = 0.004407792432213231
    Error = 0.0048828125
n = 10
    c =  1.23486328125
    fc = -0.005178920271853293
    Error = 0.00244140625
n = 11
    c =  1.236083984375
    fc = -0.0003924970675475148
    Error = 0.001220703125
n = 12
    c =  1.2366943359375
    fc = 0.0020059136386210152
    Error = 0.0006103515625
n = 13
    c =  1.23638916015625
    fc = 0.0008062748691961552
    Error = 0.00030517578125
n = 14
    c =  1.236236572265625
    fc = 0.00020678055856349076
    Error = 0.000152587890625
n = 15
    c =  1.2361602783203125
    fc = -9.28853385790962e-05
    Error = 7.62939453125e-05
n = 16
    c =  1.2361984252929688
    fc = 5.6940838785601855e-05
    Error = 3.814697265625e-05
n = 17
    c =  1.2361793518066406
    fc = -1.7973942675331145e-05
    Error = 1.9073486328125e-05
n = 18
    c =  1.2361888885498047
    fc = 1.948302485743625e-05
    Error = 9.5367431640625e-06
n = 19
    c =  1.2361841201782227
    fc = 7.544352920163533e-07
    Error = 4.76837158203125e-06
n = 20
    c =  1.2361817359924316
    fc = -8.609780141277668e-06
    Error = 2.384185791015625e-06
n = 21
    c =  1.2361829280853271
    fc = -3.927679036896947e-06
    Error = 1.1920928955078125e-06
n = 22
    c =  1.236183524131775
    fc = -1.5866235254513583e-06
    Error = 5.960464477539062e-07
```

```
n = 23
    c =  1.2361838221549988
    fc = -4.160945299425123e-07
    Error = 2.980232238769531e-07
n = 24
    c =  1.2361839711666107
    fc = 1.691702777861792e-07
    Error = 1.4901161193847656e-07
n = 25
    c =  1.2361838966608047
    fc = -1.234621518353407e-07
    Error = 7.450580596923828e-08
n = 26
    c =  1.2361839339137077
    fc = 2.285405642510341e-08
    Error = 3.725290298461914e-08
n = 27
    c =  1.2361839152872562
    fc = -5.030404959249779e-08
    Error = 1.862645149230957e-08
n = 28
    c =  1.236183924600482
    fc = -1.3724996916764098e-08
    Error = 9.313225746154785e-09
n = 29
    c =  1.2361839292570949
    fc = 4.5645296431473525e-09
    Error = 4.6566128730773926e-09
n = 30
    c =  1.2361839269287884
    fc = -4.5802335257860705e-09
    Error = 2.3283064365386963e-09
n = 31
    c =  1.2361839280929416
    fc = -7.851941319358957e-12
    Error = 1.1641532182693481e-09
n = 32
    c =  1.2361839286750183
    fc = 2.278338850913997e-09
    Error = 5.820766091346741e-10
n = 33
    c =  1.23618392838398
    fc = 1.135243454797319e-09
    Error = 2.9103830456733704e-10
n = 34
    c =  1.2361839282384608
    fc = 5.636955346943751e-10
    Error = 1.4551915228366852e-10
n = 35
    c =  1.2361839281657012
    fc = 2.779219077098105e-10
    Error = 7.275957614183426e-11
        Convergence!  End
```

```python
bisectg(1.0, 12.0, 100, 10**-10)
```

```
Begin
  a = 1.0
  b = 12.0
  fa = 1.3195493637954123
  fb = 0.5811973671890349
      The function has the same signs at a and b.  End
```

```python
bisectg(1.0, 4.0, 100, 10**-10)
```

```
Begin
  n = 0
      c =  2.5
      fc = -0.38710530814968047
      Error = 1.5
  n = 1
      c =  1.75
      fc = 0.43875812876248776
      Error = 0.75
  n = 2
      c =  2.125
      fc = 0.007443266599291176
      Error = 0.375
  n = 3
      c =  2.3125
      fc = -0.1953596457087885
      Error = 0.1875
  n = 4
      c =  2.21875
      fc = -0.09524229051310185
      Error = 0.09375
  n = 5
      c =  2.171875
      fc = -0.04420676555520808
      Error = 0.046875
  n = 6
      c =  2.1484375
      fc = -0.018456741350282524
      Error = 0.0234375
  n = 7
      c =  2.13671875
      fc = -0.00552525122268932
      Error = 0.01171875
  n = 8
      c =  2.130859375
      fc = 0.000954408899078496
      Error = 0.005859375
  n = 9
      c =  2.1337890625
      fc = -0.002286574586482182
      Error = 0.0029296875
  n = 10
      c =  2.13232421875
```

```
    fc = -0.0006663707350962333
    Error = 0.00146484375
n = 11
    c =  2.131591796875
    fc = 0.00014394716733967527
    Error = 0.000732421875
n = 12
    c =  2.1319580078125
    fc = -0.000261229769820126
    Error = 0.0003662109375
n = 13
    c =  2.13177490234375
    fc = -5.864579681613691e-05
    Error = 0.00018310546875
n = 14
    c =  2.131683349609375
    fc = 4.264956148158916e-05
    Error = 9.1552734375e-05
n = 15
    c =  2.1317291259765625
    fc = -7.998398626973824e-06
    Error = 4.57763671875e-05
n = 16
    c =  2.1317062377929688
    fc = 1.732551118927006e-05
    Error = 2.288818359375e-05
n = 17
    c =  2.1317176818847656
    fc = 4.663538721638716e-06
    Error = 1.1444091796875e-05
n = 18
    c =  2.131723403930664
    fc = -1.6674343426004157e-06
    Error = 5.7220458984375e-06
n = 19
    c =  2.131720542907715
    fc = 1.4980510918416456e-06
    Error = 2.86102294921875e-06
n = 20
    c =  2.1317219734191895
    fc = -8.46918997154944e-08
    Error = 1.430511474609375e-06
n = 21
    c =  2.131721258163452
    fc = 7.06679527562315e-07
    Error = 7.152557373046875e-07
n = 22
    c =  2.131721615791321
    fc = 3.10993796936998e-07
    Error = 3.5762786865234375e-07
n = 23
    c =  2.131721794605255
    fc = 1.1315094439190432e-07
    Error = 1.7881393432617188e-07
```

```
n = 24
    c =  2.1317218840122223
    fc = 1.4229521339004236e-08
    Error = 8.940696716308594e-08
n = 25
    c =  2.131721928715706
    fc = -3.523118907722278e-08
    Error = 4.470348358154297e-08
n = 26
    c =  2.131721906363964
    fc = -1.0500833980131574e-08
    Error = 2.2351741790771484e-08
n = 27
    c =  2.131721895188093
    fc = 1.8643435684140286e-09
    Error = 1.1175870895385742e-08
n = 28
    c =  2.1317219007760286
    fc = -4.318245316881075e-09
    Error = 5.587935447692871e-09
n = 29
    c =  2.131721897982061
    fc = -1.226950541166616e-09
    Error = 2.7939677238464355e-09
n = 30
    c =  2.131721896585077
    fc = 3.1869640260140386e-10
    Error = 1.3969838619232178e-09
n = 31
    c =  2.131721897283569
    fc = -4.5412740234951343e-10
    Error = 6.984919309616089e-10
n = 32
    c =  2.131721896934323
    fc = -6.77151668071474e-11
    Error = 3.4924596548080444e-10
n = 33
    c =  2.1317218967597
    fc = 1.2549050687482577e-10
    Error = 1.7462298274040222e-10
n = 34
    c =  2.1317218968470115
    fc = 2.8887559011536723e-11
    Error = 8.731149137020111e-11
        Convergence!  End
```

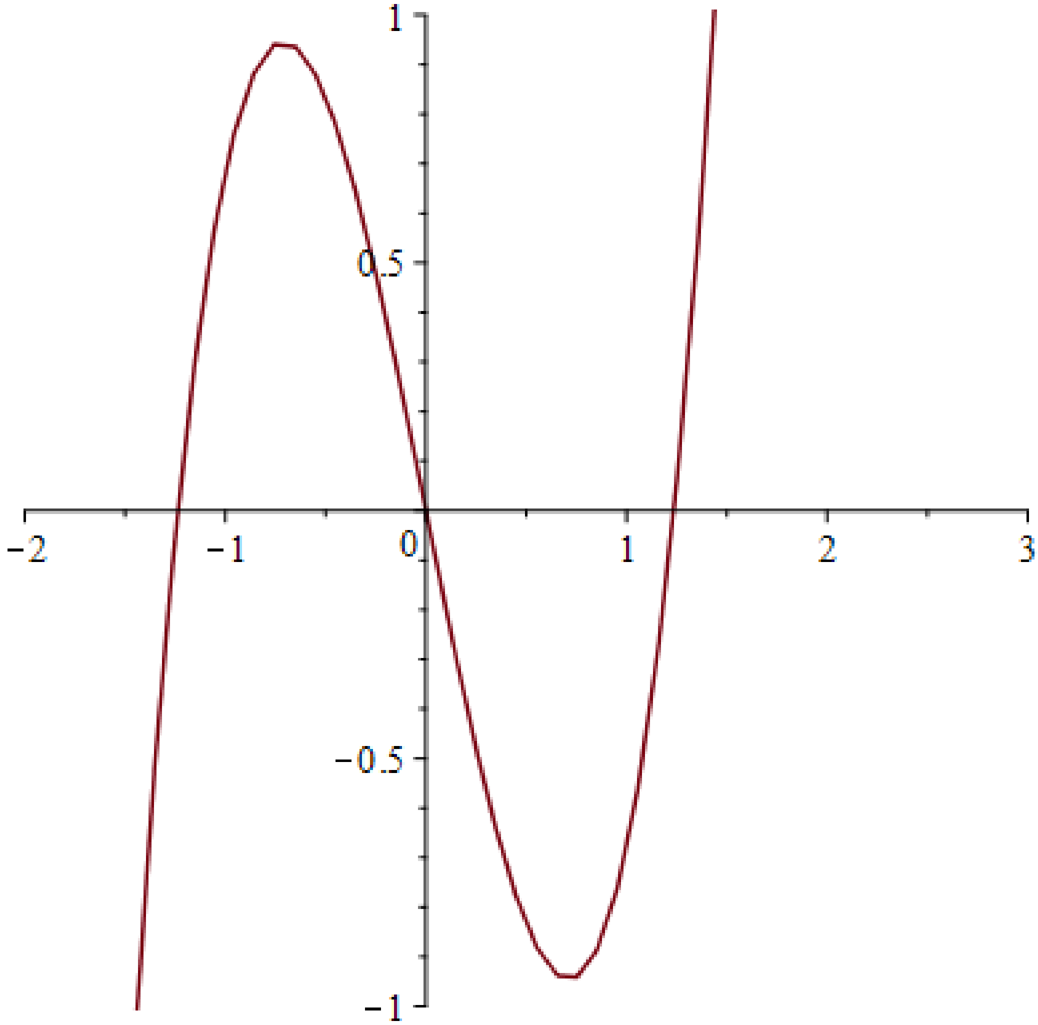

Figure 2: Maple Plot for Bisection 1

Since -2 and 3 have opposite signs, the process is allowed to continue, despite the fact that there are multiple zeros on the interval. The first bisection splits the interval at 0.5, and since -2 and 0.5 share a sign, we select the right half of the interval. Left with only one zero on our new interval, the process slowly converges to that root, at ≈ 1.2.

Maple gives the zeros:

{x = 1.236183928}, {x = -1.236183928}, {x = 0.}

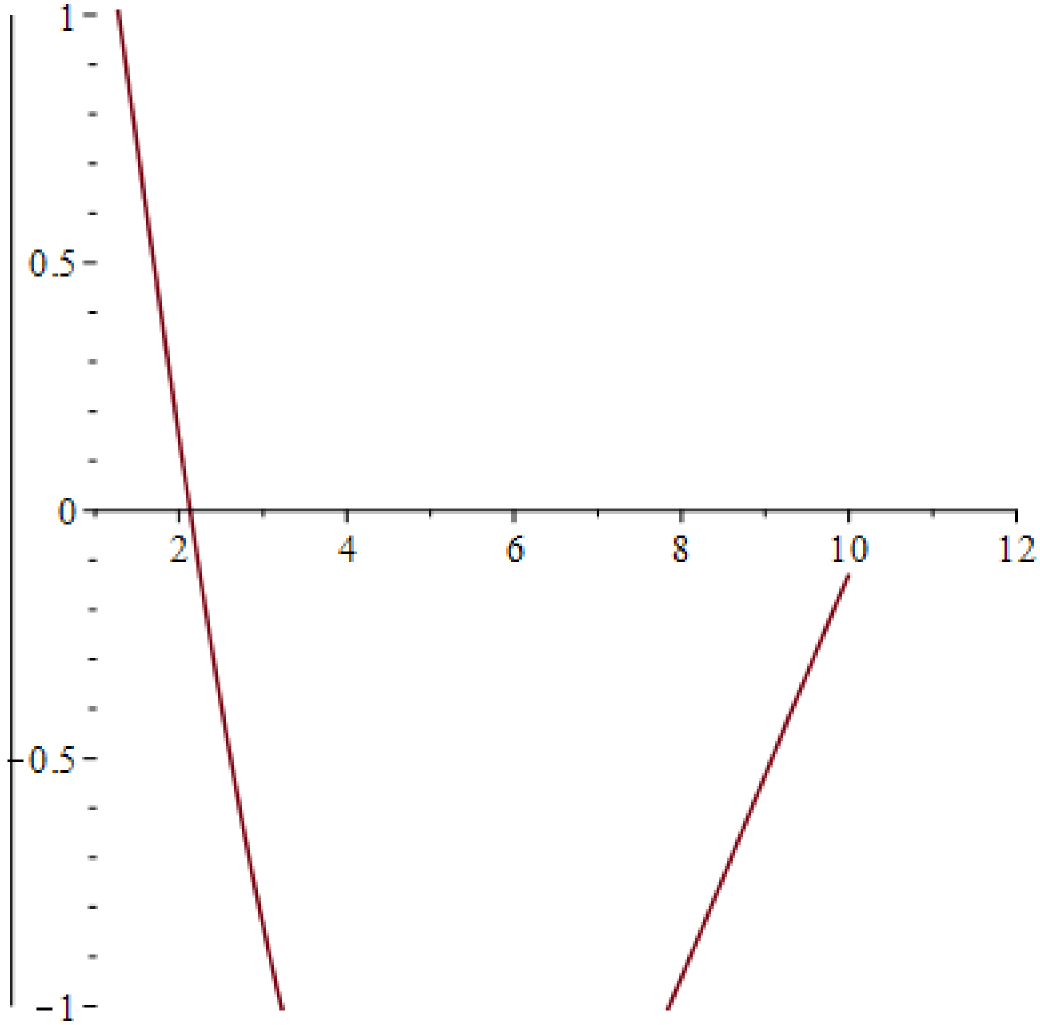

Figure 3: Maple Plot for Bisection 2

Dspite there being zeros on the interval, the endpoints 1 and 12 share the same sign, and we're not allowed to continue further. This highlights a serious limitation to the method.

Maple gives the zeros:

{x = -1.136354516}, {x = 2.131721897}, {x = 10.32951608}

Only the last two of these are within the interval.

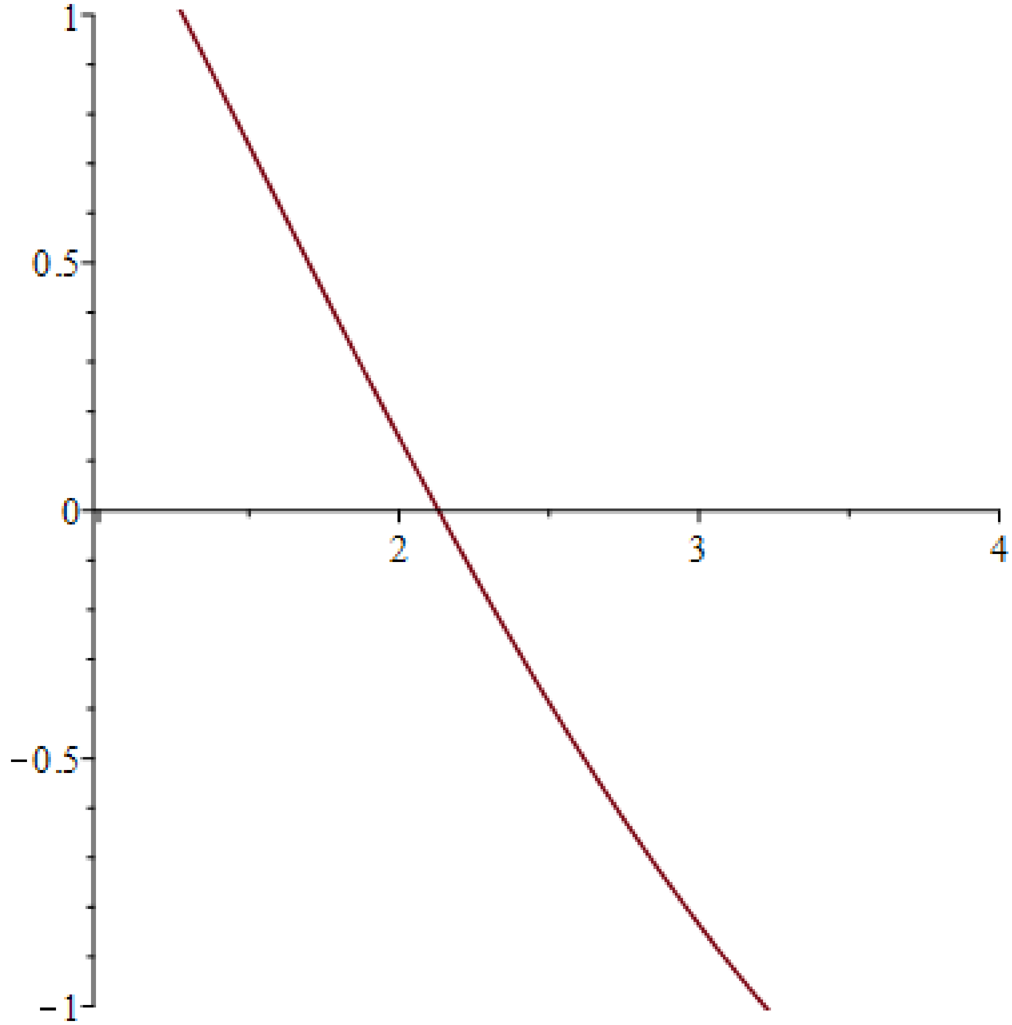

Figure 4: Maple Plot for Bisection 3

With only one zero on our new interval and opposite signs at the endpoints, this interval with this function is ideal for application of the bisection method. We hone in on our one zero around 2.1 exactly as expected.

Maple gives the zeros:

{x = -1.136354516}, {x = 2.131721897}, {x = 10.32951608}

Only the second of these is within our interval.