CP.1.4.1, Sauer3

Each equation has one real root. Use Newton's Method to approximate the root to eight correct decimal places.

a. $x^3 = 2x + 2$;

b. $e^x + x = 7$;

c. $e^x + \sin x = 4$.

CP.1.4.1, Sauer3, solution, Langou

Note: to get the root to "eight correct decimal places", we run Newton's method to "full" convergence. We might do a few iterations more than strictly needed to get "eight correct decimal places".

Colab Notebook: `https://colab.research.google.com/drive/1Y9ASj76NH8tbOy5xOZlVTvBxeoGMZw1f`

```
from math import exp
from math import cos
from math import sin
import scipy.optimize
import numpy as np
```

```
# (a) x^3 = 2x + 2

f = lambda x : ( x ** 3 ) - 2. * x - 2.

x_fsolve = scipy.optimize.fsolve( f, 1. )[0]
print( "  ", f"{x_fsolve:.16f}" )

p = np.roots( [ 1., 0., -2., -2.] )
x_roots = np.real(p[np.isreal(p)])[0]
print( "  ", f"{x_roots:.16f}" )

dfdx = lambda x : 3 * ( x ** 2 ) - 2.

r = x_fsolve

x = 1.
for i in range(0,8):
  x = x - f(x) / dfdx(x)
  true_fwd_rel_error = abs( r - x ) / abs( r )
  print( f"{i+1:2d}", f"{x:.16f}", f"{true_fwd_rel_error:.2e}" )
```

```
   1.7692923542386312
   1.7692923542386312
```

```
 1 4.0000000000000000 1.26e+00
 2 2.8260869565217392 5.97e-01
 3 2.1467190137392356 2.13e-01
 4 1.8423262771400926 4.13e-02
 5 1.7728476364392378 2.01e-03
 6 1.7693013974364495 5.11e-06
 7 1.7692923542973595 3.32e-11
 8 1.7692923542386314 1.25e-16
```

```
# (b) e^x + x = 7

f = lambda x : ( exp(x) ) + x - 7.

x_fsolve = scipy.optimize.fsolve( f, 1. )[0]
print( "   ", f"{x_fsolve:.16f}" )

dfdx = lambda x : ( exp(x) ) + 1.

r = x_fsolve

x = 1.
for i in range(0,5):
  x = x - f(x) / dfdx(x)
  true_fwd_rel_error = abs( r - x ) / abs( r )
  print( f"{i+1:2d}", f"{x:.16f}", f"{true_fwd_rel_error:.2e}" )
```

```
    1.6728216986289064
 1 1.8825899495899661 1.25e-01
 2 1.6906488575705849 1.07e-02
 3 1.6729550688970045 7.97e-05
 4 1.6728217061168933 4.48e-09
 5 1.6728216986289064 0.00e+00
```

```
# (c) e^x + sin x = 4

f = lambda x : exp(x) + sin(x) - 4.

x_fsolve = scipy.optimize.fsolve( f, 1. )[0]
print( "   ", f"{x_fsolve:.16f}" )

dfdx = lambda x : exp(x) + cos(x)

r = x_fsolve

x = 1.
for i in range(0,4):
  x = x - f(x) / dfdx(x)
  true_fwd_rel_error = abs( r - x ) / abs( r )
  print( f"{i+1:2d}", f"{x:.16f}", f"{true_fwd_rel_error:.2e}" )
```

```
    1.1299804986508324
 1 1.1351038268723292 4.53e-03
 2 1.1299886711269971 7.23e-06
```

```
3 1.1299804986716071 1.84e-11
4 1.1299804986508324 0.00e+00
```