Tarea: Metodos de solucion de raices

Sergio Andres Vargas Mendez

Descripcion

A partir de los cuatro metodos vistos en clase (Biseccion de Bolzano, Newton-Rapshon, Secante y Posicion falsa) encontrar las raices para la funcion $f(x)$.

$$f(x) = 1.04ln(x) - 1.26cos(x) + 0.0307e^x$$

Modulos

Se importan los modulos requisito y los creados para usarlos en el notebook.

In [1]:

```
!pip install sympy
import sys, os
sys.path.append(os.path.dirname(sys.path[0]))

import numpy as np
from num_sol import roots
from num_sol import utils
```

```
Defaulting to user installation because normal site-packages is not writeabl
e
Collecting sympy
  Downloading sympy-1.11.1-py3-none-any.whl (6.5 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 6.5/6.5 MB 6.0 MB/s eta 0:00:0
000:0100:01
Collecting mpmath>=0.19
  Downloading mpmath-1.2.1-py3-none-any.whl (532 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 532.6/532.6 kB 4.9 MB/s eta 0:0
0:00a 0:00:01
Installing collected packages: mpmath, sympy
  WARNING: The script isympy is installed in '/home/FoxyLuxe/.local/bin' whi
ch is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this
 warning, use --no-warn-script-location.
Successfully installed mpmath-1.2.1 sympy-1.11.1
```

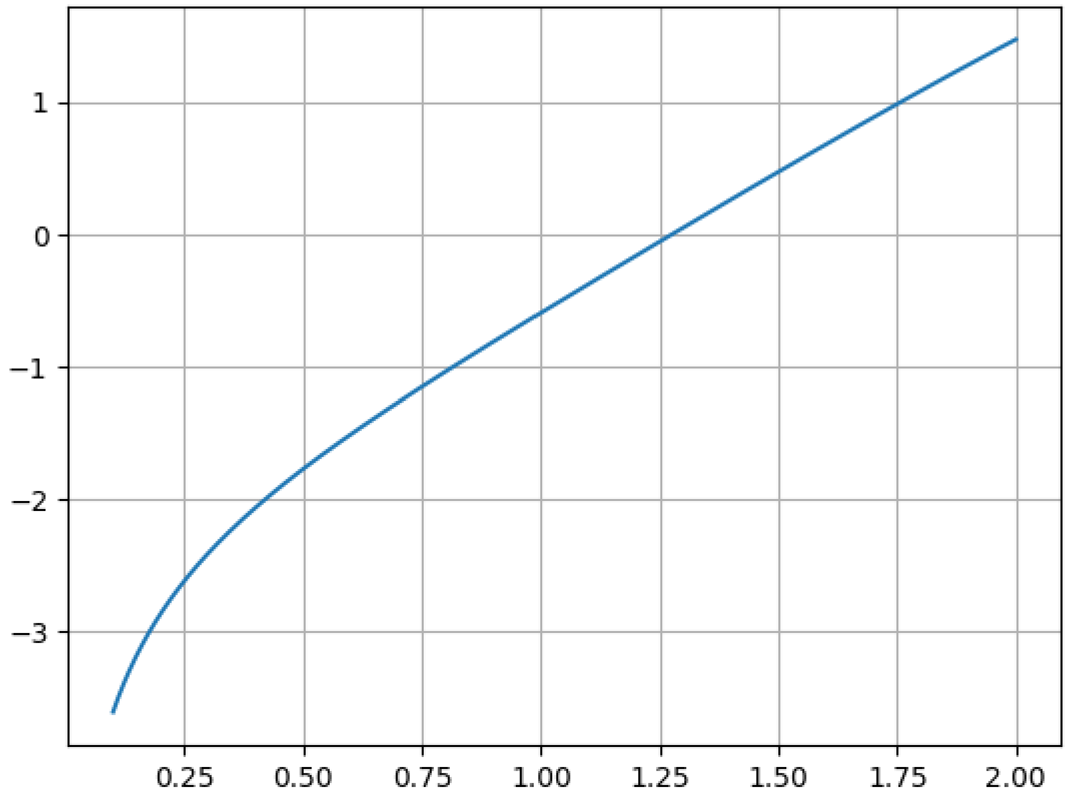
Metodo visual

Antes de intentar cualquier deduccion de las raices para la funcion $f(x)$ es necesario inspeccionar la grafica para encontrar intervalos donde existen raices porque son una exigencia de convergencia. Por referencia se tendra en cuenta la raiz calculada con Wolfram

$r = 1.27723498267385...$, por lo tanto se escogera el intervalo $[0.1,\ 2]$ para todas las pruebas.

In [2]:
```
def f(x): return 1.04*np.log(x)-1.26*np.cos(x)+0.0307*np.exp(x)

inter = [0.1, 2]
utils.plot_function(f, interval = inter)
```



Metodo de biseccion

El metodo de biseccion requiere de un intervalo $[a, b]$ para el cual se conoce de antemano que existe una raiz en ese intervalo. El intervalo se acota con cada iteracion haciendo que la nueva cota sea $x_{i+1} = \frac{a+b}{2}$ considerando $x_0 = a$, dicha cota x_i reemplazara a o b segun la condicion de polaridad.

$$si\ f(a)f(x_i) < 0,\ b = x_i$$

$$si\ f(a)f(x_i) > 0,\ a = x_i$$

$$si\ f(a)f(x_i) = 0,\ r = x_i$$

Definicion del error.

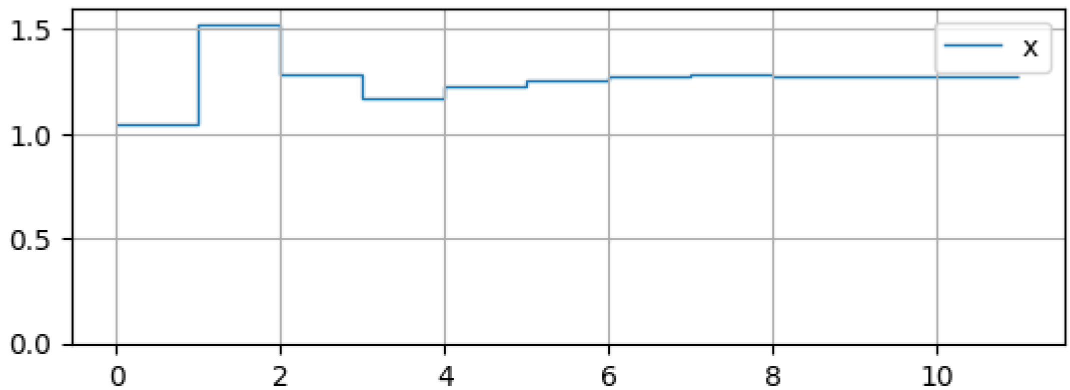
$$err_i = \frac{b-a}{2}$$

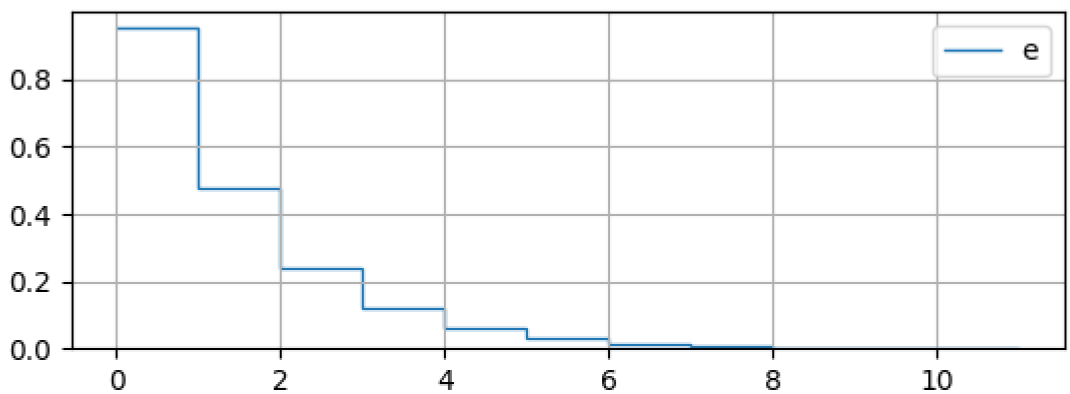
En base a la definicion anterior se puede calcular las raices como una iteracion.

Utilizando la implementacion propuesta se obtuvieron los siguientes resultados.

In [3]:
```
r = roots.interval_method(f, interval = inter, history = True)
```

```
 i        a        b        xi       err
 0 0.100000 2.000000 1.050000 0.950000
 1 1.050000 2.000000 1.525000 0.475000
 2 1.050000 1.525000 1.287500 0.237500
 3 1.050000 1.287500 1.168750 0.118750
 4 1.168750 1.287500 1.228125 0.059375
 5 1.228125 1.287500 1.257812 0.029687
 6 1.257812 1.287500 1.272656 0.014844
 7 1.272656 1.287500 1.280078 0.007422
 8 1.272656 1.280078 1.276367 0.003711
 9 1.276367 1.280078 1.278223 0.001855
10 1.276367 1.278223 1.277295 0.000928
root = 1.277294921875
```

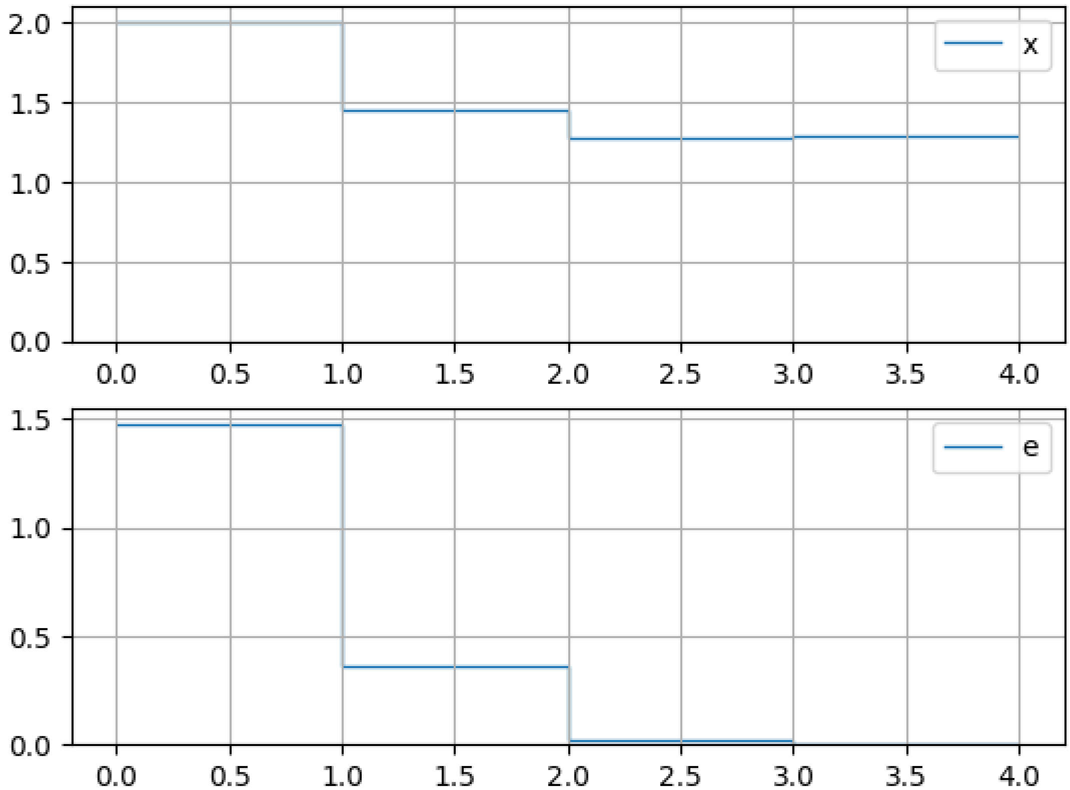




Metodo secante

In [4]:
```
r = roots.secant_method(f, points = inter, history = True)
```

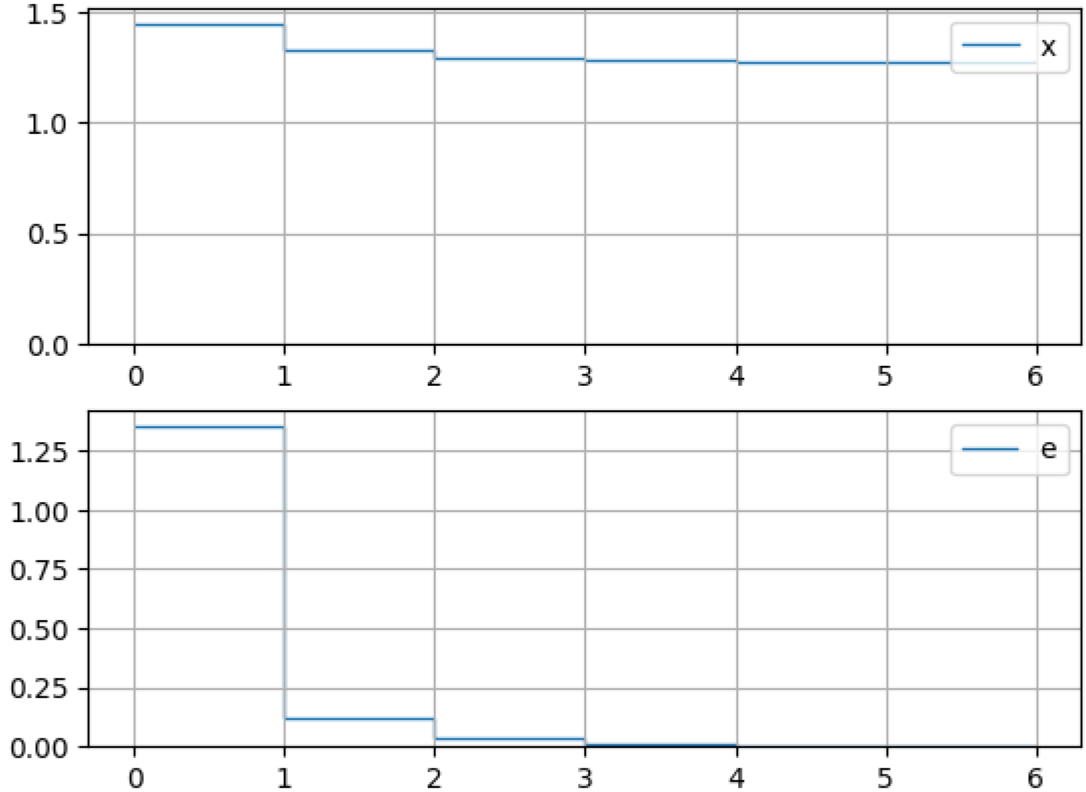
```
 i       xn   xnPlus      err
 0 0.100000 2.000000 1.472062
 1 2.000000 1.450132 0.365747
 2 1.450132 1.268346 0.018944
 3 1.268346 1.277298 0.000134
root = 1.2772980439454413
```



Metodo posicion falsa

In [5]:
```
r = roots.false_position_method(f, interval = inter, history = True)
```

```
 i   a        b        xi       err
 0 0.1 2.000000 1.450132 1.350132
 1 0.1 1.450132 1.326067 0.124066
 2 0.1 1.326067 1.291827 0.034239
 3 0.1 1.291827 1.281669 0.010158
 4 0.1 1.281669 1.278589 0.003080
 5 0.1 1.278589 1.277649 0.000940
root = 1.277649274316975
```


1.5
1.0
0.5
0.0
0
1
2
3
4
5
6
x
1.25
1.00
0.75
0.50
0.25
0.00
0
1
2
3
4
5
6
e