EX.1.1.7, Langou

Let ($f : \mathbb{R} \rightarrow \mathbb{R}$) be continuous. Let a and b, and `tol`. Let k be the smallest number of iterations of the Bisection Method that guarantees a forward error bound on the approximate solution within `tol`. Let `numevals` be the smallest number of iterations of the Bisection Method that guarantees a forward error bound on the approximate solution within `tol`.

a. Derive a formula that relates a, b, k and `tol`.

b. Solve for k as a function of a, b, and `tol`.

c. Derive a formula that relates k and `numevals`.

Let a, b, and `tol` as given below. How many function evaluations of the Bisection Method are required to guarantee a forward error bound on the approximate solution within `tol`? Answer with an integer.

d. $a = -11$, $b = 28$, `tol` $=10^{-9}$

e. $a = 0.5$, $b = 0.7$, `tol` $=10^{-14}$

EX.1.1.7, Langou, solution, Langou

- See for a short Colab Noteboook answering (d) and (e) `https://colab.research.google.com/drive/1NdHyu9rQKNS44TM-G5gNn2SWsH84CP90`
- See for a longer Colab Noteboook explaining more `https://colab.research.google.com/drive/1gtEuT7_kJUzySUxTbv2dAsyo_VRIqFuN`

(a) Derive a formula that relates a, b, k and `tol`.

At each step k, we have an approximate solution c_k. At each step k, we define the forward error, $\texttt{true\_err}(k)$, as

$$\texttt{true\_err}(k) = |c_k - x_\star|.$$

The goal is to find k that guarantees

$$\texttt{true\_err}(k) \leq \texttt{tol}.$$

We note that, since $x_\star$ is what we are looking for, it seems reasonnable to assume that we cannot compute $\texttt{true\_err}(k)$.

We will use a bound on the forward error, $\texttt{err\_bound}(k)$, that (1) we can compute and that (2) is such

$$\texttt{true\_err}(k) \leq \texttt{err\_bound}(k).$$

We also would like $\texttt{err\_bound}(k)$ to be "tight" so to be somewhat representative of what $\texttt{true\_err}(k)$ does. Then we will find k such

$$\texttt{err\_bound}(k) \leq \texttt{tol}.$$

We understand that this k will guarantee that

$$\texttt{true\_err}(k) \leq \texttt{tol}.$$

For $k=0$, a bound on the forward error is $\frac{b-a}{2}$. This is because (1) we know that a solution $x_\star$ is in between (a,b), and (2) we take our approximate solution to be $c_0=\frac{b-a}{2}$, so that

$$|c_0 - x_\star| \leq \frac{b-a}{2}.$$

Setting

$$\texttt{err\_bound}(0) = \frac{b-a}{2}.$$

This guarantees that

$$\texttt{true\_err}(0) = |c_0 - x_\star| \leq \texttt{err\_bound}(0).$$

So that $\texttt{err\_bound}(0)$ is an error bound on $\texttt{true\_err}(0)$.
Then, at each step of the Bisection Method, the bound on the error is divided by 2, so that for example for $k=1$, the bound on the error is $\frac{b-a}{4}$, for $k=2$, the bound on the error is $\frac{b-a}{8}$, etc. The general formula is

$$\texttt{err\_bound}(k) = (b-a)\cdot 2^{-(k+1)}.$$

So we want k, the smallest integer such that

$$(b-a)\cdot 2^{-(k+1)} \leq \texttt{tol}.$$

(b) Solve for k as a function of a, b, and $\texttt{tol}$.
We start with

$$(b-a)\cdot 2^{-(k+1)} \leq \texttt{tol}.$$

We rearrange as

$$\frac{b-a}{\texttt{tol}} \leq 2^{k+1}.$$

We apply $\log_2$

$$\log_2(b-a) - \log_2(\texttt{tol}) \leq k+1.$$

So we get that k is the smallest integer such that

$$\log_2(b-a) - \log_2(\texttt{tol}) - 1 \leq k.$$

Another way to write this is

$$k = \lceil \log_2(b-a) - \log_2(\texttt{tol}) - 1 \rceil.$$

(c) Derive a formula that relates k and $\texttt{numevals}(k)$.

Before starting any iteration ($k=0$), we need function evaluation for $f(a)$ and $f(b)$. So, for $k=0$, $\texttt{numevals}(0)$ is 2. Then, at each iteration, we perform one function evaluation. So, for $k=1$, $\texttt{numevals}(1)$ is 3. And so on. So that

$$\texttt{numevals}(k) = k+2.$$

(d) $a=-11$, $b=28$, $\texttt{tol}=10^{-10}$

```
from math import log2
from math import ceil
```

```
a = -11
b = 28
tol = 1e-9
k = ceil( log2( b - a ) - log2(tol) - 1 )
numevals = k + 2
print( numevals )
```

```
37
```

(e) $a = 0.5$, $b = 0.7$, `tol` $=10^{-14}$

```
a = 0.5
b = 0.7
tol = 1e-14
k = ceil( log2( b - a ) - log2(tol) - 1 )
numevals = k + 2
print( numevals )
print( numevals )
```

```
46
```