Exercise 1

Write a method that converts a zero coupon bond curve (given as an array of `doubles`) in a Libor rate curve, and return the Libor rate curve again as an array of `doubles`. You can write this method also in the test class you have created for Handout 4.

Do then the following test, possibly again in the test class above: take $T = 3$, a time step for the tenure structure $\Delta_i = \delta = 0.5$ and the zero coupon bond curve of Exercise 3 of Handout 4. Call the method you have written above with these data, and then give the resulting vector of `doubles` to the constructor of the class `SwapWithoutFinmath`, specifying that it is indeed a Libor rate curve. Compute the value of the swap rate for such a curve and for the tenure structure above, and check if the value is the same as the one you get giving directly the original zero coupon bond curve.

Exercise 2

This exercise (or some points of this exercise) may be done during the Tutorium

Suppose you have a tenure structure

$$T_0 = 0 < T_1 < \cdots < T_{n-1} < T_n = T$$

of evenly distributed points (i.e., $T_i - T_{i-1} = \Delta$ for every $i = 1, \dots, n$), and call S_i, $i = 3, \dots, n$, the par swap rate for the partial tenure structure

$$T_0 = 0 < T_1 < \cdots < T_{i-1} < T_i,$$

i.e., the value S_i such that

$$\sum_{j=1}^{i-1} (L(T_j, T_{j+1}; 0) - S_i)\, P(T_{j+1}; 0)(T_{j+1} - T_j) = 0.$$

Also suppose that you are given the first two bonds $P(T_1; 0)$, $P(T_2; 0)$.

(a) Write a class `Bootstrap` in order to do the following:

(i) get the value of $P(T_i; 0)$ knowing the value of $P(T_1), \dots, P(T_{i-1}; 0)$ and the par swap rate S_i, $i > 2$;

(ii) get the value of $P(T_i; 0)$ and $P(T_{i-1}; 0)$ knowing the value of $P(T_1; 0), \dots, P(T_{i-2}; 0)$ and the par swap rate S_i, $i > 2$.

Try to give an implementation of this class such that these calculations can be done recursively, i.e., the newly computed values can be stored and used to compute the next ones.

(b) Write a test where you call the methods of this class in order to compute the value of the bonds $P(T_i; 0)$, $i > 2$ when you have $\Delta = 0.5$, $P(T_1; 0) = 0.98$, $P(T_2; 0) = 0.975$, and

$$S_3 = 0.0086, \quad S_4 = 0.0077, \quad S_5 = 0.0073, \quad S_6 = 0.0084,$$
$$S_8 = 0.0075, \quad S_{10} = 0.0085, \quad S_{12} = 0.0095, \quad S_{14} = 0.0092.$$

You should get the following results:

$$P(0.5; 0) = 0.9800, \quad P(1; 0) = 0.9750, \quad P(1.5; 0) = 0.9716, \quad P(2; 0) = 0.9688$$
$$P(2.5; 0) = 0.9658, \quad P(3; 0) = 0.9597, \quad P(3.5; 0) = 0.9572, \quad P(4; 0) = 0.9547,$$
$$P(4.5; 0) = 0.9490, \quad P(5; 0) = 0.9433, \quad P(5.5; 0) = 0.9366, \quad P(6; 0) = 0.9301,$$
$$P(6.5; 0) = 0,9266, \quad P(7; 0) = 0,9231,$$

Hints:

- Note that, whereas you can address point (a) just looking at the formulas in the script, things are more difficult for point (b): this is for example the case when you have semi-annual bonds but you are given only annual par swap rates.
- You can solve this point by using a *root finder* algorithm, as the one implemented in `net.finmath.rootfinder.BisectionSearch`, by which you can find the zero of a function f.
- In particular, here $f(x)$ can be the difference $S - S(x)$, where S is the given par swap rate and $S(x)$ is the value of the par swap rate if the value of the new bond is x: at every iteration of the algorithm, $f(x)$ can be computed by a method which returns the value of the difference above.
- However, there are here two bonds whose value you don't know, i.e., $P(T_{i-1};0)$ and $P(T_i;0)$. For this reason, in order to compute $f(x)$ at the point above, you have to get the value of $P(T_{i-1};0)$ if $P(T_i;0) = x$. In order to do this, you can write a method which performs an interpolation between the values of two bonds, and call this method inside the method computing $f(x)$. A nice interpolation is for example

$$P(T_{i-1};0) = e^{0.5(\ln P(T_{i-2};0)+\ln P(T_i;0))}.$$