

Skill Assessment

Paco Castaneda Ruan

April 23, 2025

The algorithm contained inside the function `segLengths` considers three variables: S_1 , S_2 and L , which represent the length of the left side element, the length of the right side element and the length of the middle segment to be filled, respectively. Moving forward, we will assume that $S_1 < S_2$, due to the nature of the problem, however the function can receive and process $S_1 \geq S_2$.

The algorithm fills the gap L with a list of n elements ℓ . We will call the length of these elements $\ell_1, \ell_2, \dots, \ell_n$. For now we will not impose conditions on n , as those will arise naturally when discussing the algorithm.

In order to ensure a smooth transition between S_1 and S_2 , we assume the lengths to have the form

$$\ell_i = (S_2 - S_1)x_i - S_1, \quad (1)$$

where $0 \leq x_i \leq 1$ to ensure that ℓ_i is between S_1 and S_2 . We want the total length of the segments to be equal to L , which means

$$\sum_{i=1}^n \ell_i = \sum_{i=1}^n (S_2 - S_1)x_i - S_1 = L. \quad (2)$$

We can reorder this expression to obtain

$$\sum_{i=1}^n x_i = \frac{L - nS_1}{S_2 - S_1}, \quad (3)$$

which gives us a set of conditions on the x_i 's. To ensure that our elements are of increasing length, we assume that these variables have the form

$$x_i = \alpha i, \quad (4)$$

where α is a positive constant. Substituting equation (4) into (3) we find that α has the form

$$\alpha = \frac{2(L - nS_1)}{n(n+1)(S_2 - S_1)}. \quad (5)$$

From this expression we can obtain two conditions on the number of segments n . First, to ensure that α is positive, we need

$$n < \frac{L}{S_1}. \quad (6)$$

Second, we need to ensure that all the elements have a length smaller than St . Due to the nature of the x_i 's, it is enough to check that the last segment $\ell_n < S_2$, which gives us the condition.

$$\alpha n = \frac{2(L - nS_1)}{(n+1)(S_2 - S_1)} < 1, \quad (7)$$

$$\frac{2L + (S_1 - S_2)}{S_2 + S_1} < n. \quad (8)$$

Any n satisfying the conditions in equations (6) and (8) (if there are any) gives us a valid list of element lengths. We choose

$$n = \left\lceil \frac{2L + (S_1 - S_2)}{S_2 + S_1} \right\rceil \quad (9)$$

to use the minimum number of elements possible.

If the parameters are such that

$$\left\lceil \frac{2L + (S_1 - S_2)}{S_2 + S_1} \right\rceil > \frac{L}{S_1}, \quad (10)$$

then there are no eligible n values and the algorithm cannot proceed. In this case, the suggested solution is to rerun the algorithm with a slightly smaller S_1 value, which we obtain by dividing $S_1/2$ (this parameter can be tweaked inside the function to take values such as 1.5 with the argument `divCoeff`).