# 2-parameter GAINTUNER

Dariusz Horla, Wojciech Giernacki
Poznan University of Technology
dariusz.horla@put.poznan.pl
dhorla.put.poznan.pl
act.put.poznan.pl

## 1 Sketch of the algorithm for optimal tuning of a two-parameter controller

Suppose we need to tune two parameters of a controller, i.e. $P_1$ and $P_2$, as an example $k_P$ and $T_i$ in a continuous-time PI controller. The algorithm is composed of the steps:

0) for the given system calculate/estimate allowable ranges of $P_1$ and $P_2$ ensuring stability of the closed-loop system,
1) define initial values for $P_1^{(0)}$ and $P_2^{(0)}$,
2) using a sequence Fibonacci iterations for defined tolerance $\epsilon$ and $k = 0$ implement the following **bootstrapping** technique (put $P_2^{(k+1)} = P_2^{(k)}$):

   2a) starting with the initial range for $P_1$ and fixed $P_2^{(k+1)}$ find by means of Fibonacci method the optimal $\hat{P}_1^{*(k+1)}$, and proceed to step 2b,

   2b) starting with the initial range for $P_2$ and fixed $P_1^{(k+1)} = \hat{P}_1^{*(k+1)}$ find by means of Fibonacci method the optimal $\hat{P}_2^{*(k+1)}$, and proceed to step 2c,

   2c) if the updated point $\left(\hat{P}_1^{*(k+1)}, \hat{P}_2^{*(k+1)}\right)$ has already been found in the past iterations, stop the algorithm (no improvement is possible anymore); otherwise, put $k := k+1$ and proceed to step 2a.

The step 2c does not have to be implemented, when number of bootstraps is assumed, as well as the number of iterations $N$ of the Fibonacci method is known.

## 2 Outline of the implementation of the algorithm

The method is iterative-based and collects information about performance index (cost function value) at sample time instants, spaced every $T_S$ seconds. By assuming the sampling period to be sufficiently small, a single main iteration of the method is initialized $N_{\max}$ times where:

- for $n = 1, \ldots, N_c - 1$ for the controller parameters updated in the previous iteration the performance index is evaluated
  $$J^{(n)} = J^{(n-1)} + \Delta J^{(n)} ,$$
  where for example $\Delta J^{(n)}$ might be chosen as $\Delta J^{(n)} = |e_n|$ corresponding to the sample of the tracking error at time $t = nT_S$ or $\Delta J^{(n)} = e_n^2 + u_n^2$ corresponding to the sum squared values of the tracking error and control input to the plant;

- for $n = N_c$ a single iteration of Fibonacci method is initialized – the cost function (performance index) value is stored, and if it is possible to compare the values of cost functions in a given range, i.e. either if the two intermediate points have been evaluated, the range is reduced or the optimal solution is found and bootstrapping takes place – either way at this point controller parameters change (are updated), what results in a transient behaviour of the dynamical system,

- for $n = N_c{+}1, \ldots, N_{\max}$ no action is taken (the parameters of the controller have been updated, no performance index is collected), and these steps are intended to allow the closed-loop system to stabilize at some point (to decay the transients), to allow performance index evaluation on the next main iteration.

Thus, the experiment-based tuning process takes $N_{\max} \cdot T_S$ seconds, what should be correlated with the reference signal changes.
Having assumed that a complete run for the Fibonacci method takes $N$ steps, and two parameters are changes, every bootstrap stage takes $2N \cdot N_{\max} \cdot T_S$ seconds.
This time horizon can be reduced if past the first iteration of the Fibonacci method, only a single intermediate point is evaluated, with the other taken from one of the previous iterations!

## 3 Sample tuning process results

The written Matlab function stores all data in a matrix with the following columns:

1) iteration number,
2) ID of the changed parameter (either 1 or 2),
3) left limit of the range of the changed parameter,
4) right limit of the range of the changed parameter,
5) current value of the 1st parameter,
6) current value of the 2nd parameter,
7) value of the performance index (allowing range reduction decision)

Based on the performance indexes from two consecutive rows, a decision is made how to reduce the current range of the tuned parameter.

The first $N$ rows correspond to the tuned 1st parameter, and fixed 2nd parameters, the second $N$ rows correspond to the tuned 2nd parameter, and fixed 1st parameters, what forms a single bootstrap.

## 4 C-code for ROS implementation of the algorithm

The described algorithm has been implemented for tuning of the two parameters, namely $k_P$ and $k_D$ of a PD controller, given their ranges, and the initial value of the second parameter.
The syntax of the method is as follows

```
update_parameters_FIB(input,main_iteration_counter,P1_range,P2_range,...
N_c,N_max,Par_initial,method,comments,output)
```

with the following parameters:

- `input` – current value of the increment of the performance index, i.e. tracking error or the low-pass filtered tracking error (requires editing of the `main.cpp` file),
- `main_iteration_counter` – variable referring to the number of iterations of the algorithm, here: 48,
- `P1_range` – range for the first tuned parameter, here $[4,\ 12]$,
- `P2_range` – range for the second tuned parameter, here $[2,\ 7]$,
- `N_c` – number of samples when performance index is collected (see Figure), here 50,
- `N_max` – number of samples corresponding to the length of the trajectory primitive, here 60,
- `Par_initial` – initial value for the second parameter, here 5,
- `method`:

  0 – performance index calculated every time it is needed

  1 – performance index is averaged over all past measurements with the same parameters

  2 – performance index is evaluated only for parameters that have been unconsidered before (the length of the tuning procedure is reduced)
- `comments`:

  0 – all comments are turned off (silent mode)

  1 – comments are turned on

The designed method has the following features:

- comprises two bootstraps,
- $\epsilon = 0.05$,
- number of iterations in a single bootstrap $2 \times 2 \times 6 = 24$ (each iteration is composed of two evaluations of performance indexes),

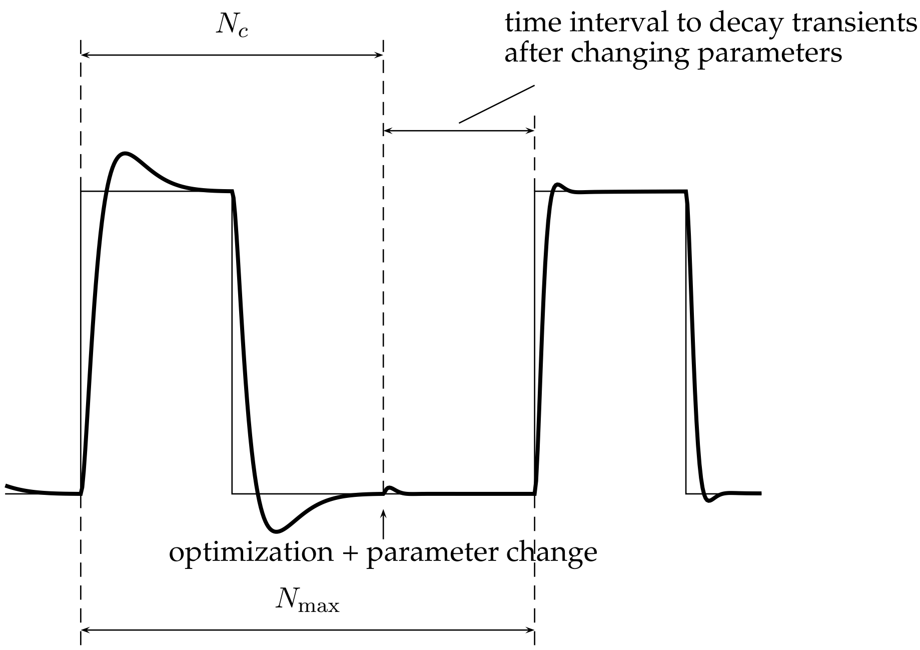


Fig. 1: Explanation of quantities: $N_c$ and $N_{\max}$ (depicted: reference vs. actual output)

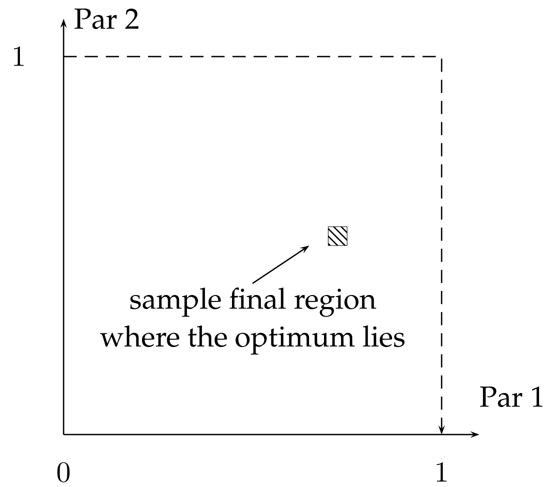


Fig. 2: Visualisation of $\epsilon = 0.05$ accuracy for sample arbitrary ranges $[0,\ 1]$ of the both parameters

- $\delta = 0.01$ (Fibonacci search),
- requires 48 loops of a trajectory primitive.

**In order to change the trajectory primitive, the `N_max` multiplicity of the time interval (multiplicity of the inverse of the frequency of executions of the tuning method) must be equal to the duration of the trajectory primitive.** In this way it can be changed, e.g. extended, for other regimes of work of the UAV, such as tuning at high speeds.
For example, when trajectory primitive has $12\,\mathrm{s}$ duration, and the experiment consists of 48 iterations (2 bootstraps in 2 parameters), the total length of the experiment is $578\,\mathrm{s}$. For the sampling period $T_S = 0.2\,\mathrm{s}$ there are 60 samples for one period of the trajectory primitive ($N_c = 50$, $N_{\max} = 60$).

# 5 Sample result of a ROS simulation in a tracking task

```
*************************************************************
 (c) Model-free autotuner (2 parameters)
     based on zero-order Fibonacci search method
                            Dariusz Horla, Wojciech Giernacki
                            Poznan University of Technology
                            dariusz.horla@put.poznan.pl
*************************************************************
 accuracy of calculations (epsilon) = 0.05
 no. of bootstrap cycles            = 2
 total number of main iterations    = 48 (for method==0 or 1)
                                    < 48 (for method==2)

Iter.   Par  |     Par(-)     Par(+)  |      Par1        Par2  |        J
-------------------------------------------------------------------------------
   1      1  |     4.0000    12.0000  |     7.0476      5.0000 |      0.9672
   2      1  |     4.0000    12.0000  |     8.9524      5.0000 |      0.5669
   3      1  |     7.0476    12.0000  |     8.9524      5.0000 |      0.7908
   4      1  |     7.0476    12.0000  |    10.0952      5.0000 |      0.9254
   5      1  |     7.0476    10.0952  |     8.1905      5.0000 |      0.3890
   6      1  |     7.0476    10.0952  |     8.9524      5.0000 |      0.5433
   7      1  |     7.0476     8.9524  |     7.8095      5.0000 |      0.5348
   8      1  |     7.0476     8.9524  |     8.1905      5.0000 |      0.6737
   9      1  |     7.0476     8.1905  |     7.4286      5.0000 |      0.5823
  10      1  |     7.0476     8.1905  |     7.8095      5.0000 |      0.6638
  11      1  |     7.0476     7.8095  |     7.4210      5.0000 |      1.1914
  12      1  |     7.0476     7.8095  |     7.4362      5.0000 |      0.5332
-------------------------------------------------------------------------------
  13      2  |     2.0000     7.0000  |     7.5742      3.9048 |      0.5852
  14      2  |     2.0000     7.0000  |     7.5742      5.0952 |      0.6452
  15      2  |     2.0000     5.0952  |     7.5742      3.1905 |      0.4044
  16      2  |     2.0000     5.0952  |     7.5742      3.9048 |      0.7961
  17      2  |     2.0000     3.9048  |     7.5742      2.7143 |      0.5868
  18      2  |     2.0000     3.9048  |     7.5742      3.1905 |      1.1901
  19      2  |     2.0000     3.1905  |     7.5742      2.4762 |      0.5964
  20      2  |     2.0000     3.1905  |     7.5742      2.7143 |      0.5382
  21      2  |     2.4762     3.1905  |     7.5742      2.7143 |      0.7170
  22      2  |     2.4762     3.1905  |     7.5742      2.9524 |      0.3506
  23      2  |     2.7143     3.1905  |     7.5742      2.9476 |      0.9388
  24      2  |     2.7143     3.1905  |     7.5742      2.9571 |      0.5167
-------------------------------------------------------------------------------
  25      1  |     4.0000    12.0000  |     7.0476      3.0434 |      0.9102
  26      1  |     4.0000    12.0000  |     8.9524      3.0434 |      0.9664
  27      1  |     4.0000     8.9524  |     5.9048      3.0434 |      0.4600
  28      1  |     4.0000     8.9524  |     7.0476      3.0434 |      0.5556
  29      1  |     4.0000     7.0476  |     5.1429      3.0434 |      0.5077
  30      1  |     4.0000     7.0476  |     5.9048      3.0434 |      0.6984
  31      1  |     4.0000     5.9048  |     4.7619      3.0434 |      0.5425
  32      1  |     4.0000     5.9048  |     5.1429      3.0434 |      0.6495
  33      1  |     4.0000     5.1429  |     4.3810      3.0434 |      1.2190
  34      1  |     4.0000     5.1429  |     4.7619      3.0434 |      0.4581
  35      1  |     4.3810     5.1429  |     4.7543      3.0434 |      0.6300
  36      1  |     4.3810     5.1429  |     4.7695      3.0434 |      0.6039
-------------------------------------------------------------------------------
  37      2  |     2.0000     7.0000  |     4.9076      3.9048 |      0.4552
  38      2  |     2.0000     7.0000  |     4.9076      5.0952 |      0.7258
  39      2  |     2.0000     5.0952  |     4.9076      3.1905 |      0.6217
  40      2  |     2.0000     5.0952  |     4.9076      3.9048 |      1.1278
  41      2  |     2.0000     3.9048  |     4.9076      2.7143 |      0.5624
  42      2  |     2.0000     3.9048  |     4.9076      3.1905 |      0.5592
  43      2  |     2.7143     3.9048  |     4.9076      3.1905 |      0.6744
  44      2  |     2.7143     3.9048  |     4.9076      3.4286 |      0.3145
  45      2  |     3.1905     3.9048  |     4.9076      3.4286 |      0.8688
  46      2  |     3.1905     3.9048  |     4.9076      3.6667 |      0.6031
  47      2  |     3.4286     3.9048  |     4.9076      3.6619 |      1.0398
  48      2  |     3.4286     3.9048  |     4.9076      3.6714 |      0.8223
-------------------------------------------------------------------------------
[ WARN] [1493725214.531450171]: Tuning finished!!
[ WARN] [1493725214.531542558]: The final results are:
[ WARN] [1493725214.531579542]: gain1: 4.91, gain2: 3.78
```