Fixed Point, Iteration-based, Adaptive Controller Tuning, Using a Genetic Algorithm

István Lovas

Óbuda University - Doctoral School of Applied Informatics and Applied Mathematics, Bécsi út 96/b, H-1034 Budapest, Hungary
E-mail: lovas.istvan@nik.uni-obuda.hu

Abstract: From a control perspective, the exact conditioning of systems with time-varying parameters is still a challenge. Many adaptive control algorithms (Adaptive Inverse Dynamics – AID, Model Reference Adaptive Controllers – MRAC, etc.) exist today. "Fixed-Point Iteration Methods" attempt to offer "alternative" control planning methods to circumvent the application of the Lyapunov function technique. The foundations of the method were developed in 2009. RFPT is an iterative control method, based on the fixed-point theorem of Stefan Banach proved in 1922 [1]. There is usually no specific suggestion for the choice of controller parameters, as the response function also depends on the approximation model parameter used and the actual behavior of the system under control. Adaptive RFPT presupposes strongly nonlinear system models in the first place, so in this case, thinking in frequency image and step inputs is not relevant (it is not advisable to conflict a nonlinear system with step inputs), so it does not have a tuning technique applicable to LTI models. However, there are a number of optimal search methods that can also be used to tune controllers (e.g., PIDs), e.g. the Genetic Algorithm. Using this method, I developed a possible autotuning process for adaptive RFPT controllers.

Keywords: Adaptive Control; Fixed Point Iteration-based Adaptive Control; Banach Space; Genetic algorithm; GA-based RFPT auto-tuning; Auto-tuning method; Control systems

1 Robust Fixed-Point Transformation-based Adaptive Controller

From a control perspective, the exact conditioning of systems with time-varying parameters is still a challenge today. Many adaptive control algorithms (Adaptive Inverse Dynamics – AID, Model Reference Adaptive Controllers – MRAC, etc.) exist today, however, in many cases, their application is difficult, their fine-tuning is not trivial, and they demand significant mathematics knowledge, as in most cases the conditioning algorithm is based on the Lyapunov stability criterion.

Lyapunov in his Ph.D. dissertation defined several stability criteria. Their common characteristic is that they are physically based on a Lyapunov function, a differentiable function of time, that can be interpreted as a scalar error metric, which is not negative and is zero, exclusively in case of zero error, and this function has to be held in control (its time-based derivative must not be positive); this is ordinary stability. If it is accessible, that this derivative is negative enough so that it can make the Lyapunov function to converge to value zero in an infinite amount of time, an asymptotically stable system will be created [2] [3].

The biggest challenge is to select this function for a given specific control task; most of the time it is determined only by “intuitions”. The biggest problem with this is that if it cannot be determined, no information about the stability of the system will be available. Generally, typical Lyapunov function candidates are available for different model tasks, which can be "adapted" to the given task [4].

"Fixed-Point Iteration Methods" attempt to offer "alternative" control planning methods to circumvent the application of the Lyapunov function technique. The essence of the method is to transform the control task, namely, the calculation of the control signal to be given by the control system, into the iterative solution of a fixed-point problem so that one step of this iteration can be performed during a control cycle of a digital controller. Ensuring the convergence of the applied iteration is based on Stefan Banach’s fixed-point theorem. The given control task can be "transformed" into a fixed-point task in several ways, RFPT ("Robust Fixed-Point Transformation") offers a possible solution for this. Later, it became clear that the operation of this method, is related to the Lyapunov function-based technique [5].

The foundations of the method were developed in 2009 [6] [7]. RFPT is an iterative control method based on the fixed-point theorem of Stefan Banach proved in 1922 [1]. The procedure uses the available, usually inaccurate dynamic model of the system to be controlled to try to implement a trajectory tracking strategy based on purely kinetic/kinematic considerations by calculating the control forces. As the model used is inaccurate, the force calculated from it does not realize the desired motion. By observing the realized motion, the method "deforms" the input of the inaccurate model until the realized motion approaches the kinematically prescribed one well enough.

MRAC (Model Reference Adaptive Control) [15] [16] controllers are well suited for nonlinear systems. The model is based on tuning control signals instead of parameters. The essence of the technique is to compare the dynamic behavior of a feedback system to a reference model and control takes place accordingly. RFPT is also suitable for this [8] [9] without the use of Lyapunov functions. In the knowledge of the inverse of the model to be controlled, the input signals for the expected behavior can be determined as follows:

- r_d - expected behavior
- u - input signal

where $u = \phi(r_d)$. The inverse model is usually incomplete, in other words, it gives a different result than expected. The controller uses a deformation function to modify the input signal to compensate for this deviation. Based on [6] in case of SISO system:

$$G(r|r^d) \stackrel{\text{def}}{=} (r + K)[1 + B \tanh(A[f(r) - r^d])] - K \tag{1}$$

$$G(r_*^d|r^d) = r_*^d, \text{ if } f(r_*^d) = r^d \tag{2}$$

$$G(-K|r^d) = -K \text{ if } r_*^d = -K \tag{3}$$

where:

- r_*^d is the deformed signal constituting the solution to the task
- K, A, B are the parameters of the controller

There is usually no specific suggestion for the choice of controller parameters, as the response function $f(r)$ also depends on the approximation model parameter used and the actual behavior of the system under control. Based on Banach's theorem, we should aim to try to give a contractive mapping of the function G in the proximity of the solution, i.e. close to r_*. To do this, the r-based derivative of G must be reduced to a value less than 1 in absolute value, which would require information about the derivative of the function $f(r)$.

The value of B can therefore be +1 or -1 according to the sign of the derivative of $f(r)$, the value of K should be chosen as a large number in comparison with which the values of r in the sum $(K + r)$ are small, and the value of A must be reduced until the iteration becomes convergent. (A very small A value causes slow convergence and inaccurate adaptability.)

In addition to the deforming function, a kinematic block forms part of the controller. Within the kinematic block, the difference between the realized trajectory and the prescribed path is determined as well as the derivative of the error and its integral as follows:

$$e_{int} = \int_{t_0}^{t} \big(q_n(\tau) - q(\tau)\big) \mathrm{d}\tau \tag{4}$$

$$\left(\frac{\mathrm{d}}{\mathrm{d}t} + \Lambda\right)^{n+1} e_{int} = 0 \tag{5}$$

where: Λ is a control parameter

The block diagram of the robust fixed-point transformation-based control is shown in the Figure 1:

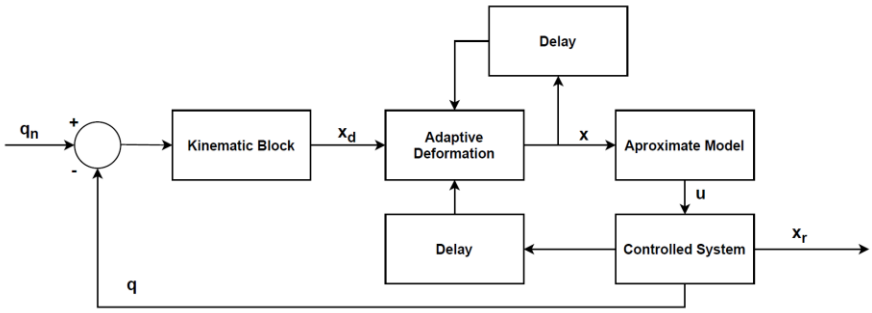


Figure 1
The block diagram of the robust fixed-point transformation-based control

The controller contains a total of four parameters:

- Λ, K, A, B

There is no specific method for determining and tuning these parameters. The value of the parameter $\Lambda \approx 1/\tau$ is roughly determined by the dynamics of the trajectory to be followed. A rapidly changing trajectory with very sluggish dynamics (τ is roughly a time constant) cannot be accurately tracked in the first place. If the dynamics of the track to be followed changes over time, it may also be necessary to tune this parameter.

Many tuning methods have been developed over several decades, for example, in order to set the PID controller correctly e.g. Ziegler-Nichols method (1942), which can be applied to processes where it is possible to operate the control cycle at the limit of stability, or the Oppelt method, which can be used to determine the individual parts based on the response of the process to step input. These methods have been developed for Linear Time-Invariant (LTI) dynamic models [13], mostly using the frequency image.

Adaptive RFPT presupposes strongly nonlinear system models in the first place, so in this case, thinking in frequency image and step inputs is not relevant (it is not advisable to conflict a nonlinear system with step inputs), so it does not have a tuning technique applicable to LTI models.

However, there are a number of optimal search methods that can also be used to tune controllers (e.g., PIDs), e.g. the genetic algorithm (GA) [16] [17] [18] [19]. Using this method, I developed a possible autotuning method for tuning adaptive RFPT.

2 Genetic Algorithm

The genetic algorithm (hereafter GA) is a rather widespread evolutionary strategy, the foundations of which were developed in 1975 by John Holland [14]. It has been used in many fields in recent decades: graph algorithm, extreme value problem, game theory, etc. However, it can also be used in case of tuning regulatory algorithms e.g. PIDs. Following a similar principle, I also developed a procedure for RFPT auto-tuning.

GA algorithms strive to ensure that the result of solving a problem is optimal within a given error threshold and accuracy. They are suitable for global optimization. They are sensitive to local optima, due to their stochastic nature, however, they can handle adequately this problem.

GA is an iterative process based on one initial population. Each population is made up of several individuals. These individuals are possible solutions to the problem under examination. However, there may be identical solutions among them. Individuals can be further broken down into genes. Genes represent the characteristics of an individual. In the process, repeatedly, new populations are created with each iteration. These are called generations. Each new generation is created from the current population using different selection, recombination, and mutation algorithms [10].

2.1 Individuals

In the case of GA, the individuals contain possible solutions to the given problem. There may even be redundancy between individuals. The first step in the process is to design the way individuals are represented. In many cases, an individual is described by a sequence of bits, a vector, or some structure. The parameters of the representation mode, the parameters giving the properties of the individual, are called genes. The algorithm modifies these genes during each iteration based on a given strategy.

2.2 Populations

The first iteration of the genetic algorithm is based on an initial population. The structure of the initial population depends on the particular problem. In many cases, it consists of a few hundred or a few thousand individuals. In the case of an optimum search problem where the location of the optimum can be guessed, this is taken into account when constructing the initial population. In this case, more individuals are created in the search area around the assumed location, otherwise, the individuals are evenly distributed. The reason for this is that by helping GA, the chances of finding a global optimum can be increased and the search time for the optimum can be reduced.

With each iteration, new populations are created from the populations. These are called generations. A generation is created using the current population, based on different selection, recombination, and mutation algorithms. Each individual is described by a "Fitness Function". During selection, aptitude is determined by this function.

2.3 Fitness Function

The individual goodness of each population is determined by a fitness function. The better this value, the closer the solution approximates the expected result, i.e. the global optimum. One of the most difficult tasks is to determine this function. It is one of the main building blocks of GA. It must be chosen with great care for each problem. During each iteration, when new generations are created, individuals with better fitness values are more likely to enter the new population (this may change depending on the selection procedure).

2.4 Selection

During selection, the individuals the genes of which we would like to be further inherited are selected from the current population based on their aptitude and quality. This is one of the conditions for the algorithm to converge towards the global optimum during the iterations. Selected individuals are called parents. There are several strategies for selecting the best individuals. These include random selection, fitness proportional selection, competitive, etc. The first two of these are presented.

2.5 Random Selection

It is less effective, however, it is one of the simplest procedures. In a given iteration, the parents are randomly selected from the given population with the same probability. Considering the Darwinian theory, this solution is not the most obvious, because in this case, the basic idea is that during the various developments, the weak individuals become extinct, and the strongest and most capable ones continue to reproduce. Regardless of fitness, all individuals are equally likely to be parents, thus further reproducing their genes, whether good or bad. The method can be further refined in order to be improved. With each iteration, it can be taken into account that if an individual has already been selected, it cannot participate as a parent later.

2.6 Fitness Proportional Selection (Roulette Method)

The method is based on the fitness function. A parent is selected from the current population in a way that the probability of selecting individuals in each population

is proportional to the fitness value. The same instance can be re-selected at each step. The chance of selection for a given individual is:

$$p(\hat{e}) = \frac{F(\hat{e}\})}{\sum_{e \in p} F(e)} \tag{6}$$

where $F(e)$ means the fitness value of individual e. The roulette wheel method is usually used for selection. The higher the fitness value of a particular individual, the larger the slice you get from the wheel. Rotate the wheel to select a perimeter point. The higher the fitness value of an individual, the more likely it is to be selected by this method.

2.7 Recombination

After selection, the first step in creating new individuals is recombination. Upon recombination, a new individual is created from the selected parents. The new individual inherits the genes of the parents. This is called a crossover. There are several crossover methods, e.g. 1-point crossover, uniform crossover, intermediate recombination, heuristic crossover, etc.

2.8 Single-Point Crossover

In a single-point crossover, an individual is cut at a random point. The parent shall have n-genes and i will be a random number where $1 \pm i < n$. The new individual inherits its genes from one parent from 1 to i and from the other parent from $i + 1$ to n. In this case, the parents are divided into two parts, the individual parts being transferred one by one to the new individual. Continuing with the method, it is possible to cross the two parents not only at one but also at several points. From one parent, n genes are selected, which are transferred to the new individual, and then the missing genes are inherited from the other parent.

2.9 Smooth and Intermediate Crossover

Smooth and intermediate crossovers are based on the same logic. Each gene in the new individual will be one of the same genes in the parents. Each gene is selected with a 50% chance, i.e. half of the genes are exchanged between parents. In the case of an intermediate crossover, the value of the inherited gene changes, the value of which is described by a function. The latter method tries to achieve its larger variety. There is no general method for selecting individual crossover operators. In each case, it needs to adapt to the task. In most cases, each gene is independent. If there is a relationship between the genes, a special “intelligent operator” must be used.

2.10 Mutation

The search space is made possible by the mutation operator. However, this operator should be used with caution. If the genes of an individual are heavily modified, traits inherited from parents with good fitness values can be impaired. Depending on the data describing the individuals (bit sequence, vector, structure), there are several mutation operations (random element per gene or element permutation, inversion operator, neighborhood mutation, sequential mutation, etc.). It is very important that during the mutation, the operator should only change the value of the gene to such an extent that it does not leave the search space. For many generations, if the algorithm approaches the optimum, convergence to the optimum may fail, if the mutation rate is too high.

2.11 Pseudocode of GA Algorithm

Algorithm 1 Genetic algorithm

```
1: Setting strategic parameters
2: t := 0
3: Creating P_0 initial
4: Initial evaluation of P_0 individual based on the fitness function
5: while Not Exit condition(P_t) do
6:     P_t* := Selection(P_t)
7:     P'_t := Recombination(P_t*)
8:     P_{t+1} := Reinstatement(P'_t)
9:     t := t + 1
10:    Calculate the fitness function
11: end while
12: Choosing individual with best fitness values
```

As a first step, the parameters are set and produce the initial population. The number of individuals in the initial population depends on the solution of the problem. That can mean a number from a few hundred to a few thousand individuals. The individual individuals, as I mentioned earlier, can be bit sequences, vectors, or some structures. In the next step, each individual is evaluated based on the fitness function. The algorithm is an iteration process. The exit condition of the process varies. The condition is usually met when a predetermined “error threshold” or iteration (generation) is reached. It is necessary to limit the iteration steps because there may be a case where the GA gets stuck and does not converge towards the optimum. Selection, recombination, and mutation take place within the cycle core. If the exit condition is met, the process stops, and the individual with the best fitness value in the last population is selected.

3 A Demonstration of the Automatic Tuning Method for GA-based Adaptive RFPT Controller Parameters

There is no exact tuning method for selecting the parameters of the adaptive robust fixed-point transformation-based control algorithm. The task of the parameters of the kinematic block and the deformation function in the control circle detailed in Chapter 1 is known, however, the choice of the values of each parameter can be determined experimentally only in a way supported by observations. The expectation for a control task in most cases is to follow a prescribed trajectory with the smallest possible error. Tracking error can be minimized by the optimal selection of each parameter. For RFPT, this means setting four parameters (Λ, K, A, B). The process of the developed method has been extended by one step compared to the genetic algorithm:

Algorithm 2 Genetic algorithm extension with RFPT simulation

1: Setting strategic parameters
2: $t := 0$
3: Creating P_0 initial
4: **Running RFPT simulations using the initial population**
5: Initial evaluation of P_0 individual based on the fitness function
6: **while** Not Exit condition(P_t) **do**
7: $P_t* := Selection(P_t)$
8: $P'_t := Recombination(P_t*)$
9: $P_{t+1} := Reinstatement(P'_t)$
10: $t := t + 1$
11: **Running RFPT simulations using the new population**
12: Calculate the fitness function
13: **end while**
14: Choosing individual with best fitness values

The method is based on a simulation block. Within the simulation block, the RFPT control block visible in figure is realized. The block consists of the following main components:

- Kinematics block
- Deforming function
- Approximating function
- Exact model

During the simulation, the goal for the system is to follow the predefined trajectory as accurately as possible. Genetic algorithm-based tuning uses the result of this simulation block.

3.1 Chromosome, Representation

The genotype assigned to each solution is made up of four numbers, i.e., an individual has four genes. The alleles assigned to the genes correspond to some parameters of the RFPT (Λ, K, A, B). The phenotype generating algorithm is provided by none other than the RFPT simulation block using the genotype.

3.2 Setting Strategic Parameters

After chromosome representation, the first step is to determine the initial population. The parameters of the individuals in the initial population are determined in consideration of the search space. Each allele, as an allele within a different range, is given a value by generating evenly distributed random numbers. Each domain is as follows:

- Λ: $0.1 < x < 6$
- K: $1e2 < x < 1e13$
- A: $1 > x > 1e-6$
- B: $[-1,1]$

As the parameter space is too large for the initial population, except for Λ, the possible values can only be multiplies of 10. There is no specific method for determining population size [11]. Most of the time it depends on the problem. In this case, the simulation was performed with populations of 50 to 450 individuals. Each new generation is created from the elite of the current population. I selected a 10% elite rate [12].

3.3 Crossover and Mutation

The first step in the crossover is to choose the parents. The parents come from the elites. I chose single-point crossover as the crossover operator. Each parent has 4 genes. I used a randomly distributed random number generator to determine the point of intersection. $P_1 = [\Lambda_1, K_1, A_1, B_1]$ and $P_2 = [\Lambda_2, K_2, A_2, B_2]$, as well as C_p should be a random number generated in the range $[1, 2, 3, 4]$. If for instance $C_p = 2$, then $C = [\Lambda_1, K_1, A_2, B_2]$. The last strategic parameter is the choice of mutation rate. As in the case of the initial population formation, I limited the values of each parameter, the mutation rate helps to broaden the search space again. I used a random mutation per gene. 6% of all genes in the population as well as the value of each gene can be modified proportionately by generating a random number between $\pm 10\%$.

3.4 Stopping Criteria

The generic algorithm is an iterative method. At the end of each iteration, it should be examined whether it is worth continuing the optimal search or not. Usually, the stop condition consists of two parameters, an expected fitness value, and a maximum iteration number. Optimally, the first condition is met, in which case the individual with the best fitness value in the last population provides the best solution to the problem. However, due to the heuristic nature of the algorithm, this is not guaranteed at all. In the case of the autotuning method developed by me, it is difficult or impractical to set a fitness value as a stopping criterion. In the case of control, the goal is to minimize trajectory tracking error or even reduce it to zero. For the latter physical systems, it can be concluded that it will never be satisfied. Since in reality, the interfering signals are limited, e.g. a system displaced from rest would have to be displaced with an unrealistically large intervention signal and then slowed down to follow the prescribed trajectory immediately. As this is not met, there will always be a tracking error. I determined only the maximum iteration number for the exit condition of the genetic algorithm. To determine this maximum number of iterations, I performed several tests, which I will detail in a later chapter.

3.5 Fitness Definition

The aptitude of each specimen is described by the fitness function. The global optimum of this function is to be found. In the case of GA-based RFPT tuning, the fitness function is determined from the results of the simulation performed for each iteration. The goal is to minimize tracking error, ideally to zero. Achieving the latter is impossible, the reason for which has already been explained in the previous chapter. However, minimizing the error is a realistic goal.

It can be assumed that the smaller the error integral, the more accurately the system follows the required trajectory. However, as with most control algorithms (e.g., PID), the system can oscillate or even immediately diverge to infinity due to the selection of unsuitable parameters. As the genes of the specimens in the initial population are determined by evenly distributed random numbers, the latter case is also highly likely to occur. However, this does not mean that during further mutation, a specimen that initially provides an erroneous simulation result cannot converge to a good solution. In the articles below [23] [24], the authors used the following performance indicators to minimize the error generated during the simulations performed with each specimen:

- MSE: Mean of the Squared Error
- IAE: Integral of the Absolute Magnitude of the Error
- ITAE: Integral of Time multiplied by Absolute Error
- ISE: Integral of the Squared Error
- ITSE: Integral of Time multiplied by the Squared Error

$$MSE = \frac{1}{t}\int_0^t (e(t))^2 dt \quad (7)$$

$$IAE = \int_0^t |e(t)| dt \quad (8)$$

$$ITAE = \int_0^t |e(t)| dt \quad (9)$$

$$ISE = \frac{1}{t}\int_0^t e(t)^2 dt \quad (10)$$

$$ITSE = \frac{1}{t}\int_0^t t * e(t)^2 dt \quad (11)$$

Based on the performance indicators, the actual fitness value is determined based on the following equation [24] [25]:

$$Fitness\ value\ =\ 1/\ Performance\ index \quad (12)$$

The same performance indicators and fitness value calculations were used to tune the RFPT.

3.6 Dynamic Models used in Testing

I tested the tuning method using the following three nonlinear SISO type dynamic models:

- Van der Pol Oscillator
- Duffing Oscillator
- Inverted pendulum

The dynamic equation of the Van de Pol Oscillator is [20]:

$$m\ddot{q} - \mu(1 - q^2)\dot{q} + \omega_0^2 q + \alpha q^3 + \lambda q^5 = g \quad (13)$$

where, the values of each exact model parameter are:

- $m = 1.2$
- $\mu = 0.4$
- $\omega_0 = 0.2$
- $\alpha = 4$
- $\lambda = 0.3$

The dynamic equation of the Duffing Oscillator is [21]:

$$\ddot{x} + \sigma\dot{x} + \alpha x + \beta x^3 - \gamma\cos(\omega t) = u \quad (14)$$

, where the values of each exact model parameter are:

- $\alpha = 1$
- $\beta = 5$
- $\sigma = 0.02$

- $\gamma = 8$
- $\omega = 0.5$

The dynamic equation of the Inverted Pendulum [22]:

$$(m + M)\sin^2\theta\, l\dot{\theta} + ml\theta^2 \sin\theta\cos\theta - (m + M)g\sin\theta = -F\cos\theta \qquad (15)$$

Where the values of each exact model parameter are:

- $m = 0.8kg$
- $M = 1kg$
- $l = 0.6m$

The number of parameters of the approximating models (approx. model) within the adaptive RFPT control block was identical during each test, only the values of the parameters were modified compared to the exact model.

4 Experimental Results

The simulation was implemented in MATLAB software. I divided the simulation into an inner and an outer block. Inside the inner block is the RFPT block, which contains the kinematic block and the deformation function, as well as one of the exact and approximated models presented in Chapter 1. During each test, the controller must follow a sinusoidal trajectory. The genetic algorithm is built around the inner block. The genetic algorithm performs the following main steps for a given population number as shown in Chapter 3:

- Fitness calculation
- Selection
- Crossover
- Mutation

Only the iteration number was specified as the exit condition. To determine the optimal population size and the maximum number of iterations, I performed several measurements based on the following parameters:

- Population size: 50, 60, 70, ... 450
- Maximum iteration: 100

Additional parameters of GA:

- Elite rate: 0.1%
- Population mutation rate: 0.06%
- Gene mutation rate: 10%

For each population, I performed 100 tests, a total of 41×100 measurements. The measurements were performed on the same computer. In the following figures, I used the average of the measurements.

Figure 2 shows the integral of the error associated with the simulation of each population. The goal is to tune the RFPT so that during the simulation, the error integral is minimized (MSE, IAE, ITAE, ISE and ITSE produces similar results). The lower this number, the more accurately the controller works. It can be seen that after the 50th iteration no significant change occurs, for all population sizes RFPT-tuning is successful, the integral of the error approaches zero. As GA is heuristics-based, the larger the population, the more likely it is to reach the number of iterations sooner, after which tuning is no longer necessary. However, when choosing a population size, run time must also be taken into consideration, which increases in proportion to the size of the population.

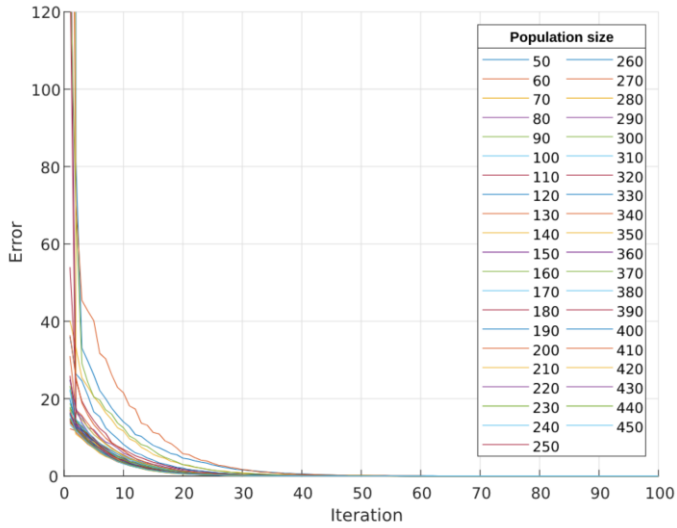
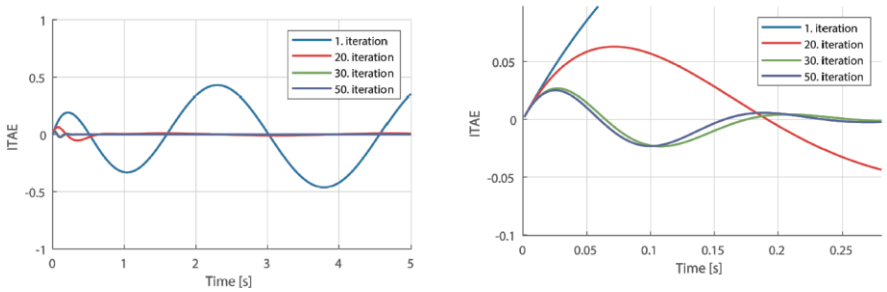


Figure 2

The integral of the error associated with the simulation of each population

Figure 3 (a) shows the process of tuning. In the case of the initial population, the system will most likely include specimens that may for instance lead to oscillations, but also, of course, those that already have the appropriate genes (RFPT parameters) at the beginning. Specimens with inappropriate parameters were not plotted, as the error rate was unrealistically large at the outset, so it was not used for plotting, but they were, of course, used for the performance of the genetic algorithm.



(a) ITAE simulation result

(b) Settling time and overshoot (details of Fig. (a))

Figure 3

Integral of Time multiplied by Absolute Error of the Van der Pol Oscillator

Figure 3 (b) shows that the trajectory tracking error decreases during tuning, at each time the controller is able to intervene faster, thus settling time and overshoot become smaller. However, the fact that in physical reality the intervening forces are limited should not be ignored. The further these values decrease, the greater the intervention signal required! This must be taken into consideration during tuning and a threshold value must be set for the magnitude of the interfering signal.

Figure 4 represents the result of the operation of the successfully tuned controller. The system had to follow a sinusoidal trajectory. The nominal trajectory is shown with a blue line while the tracking trajectory with a dashed red line. The nominal trajectory $q^N(t) = A_0 \sin(\omega_0 t)$, $A_0 = 3$ and initial state $q_0 = 0$, $\dot{q}_0 = 0$

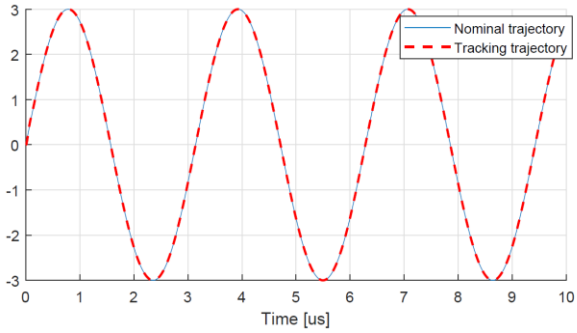


Figure 4

Trajectory tracking of the Van der Pol oscillator under tuned RFPT control

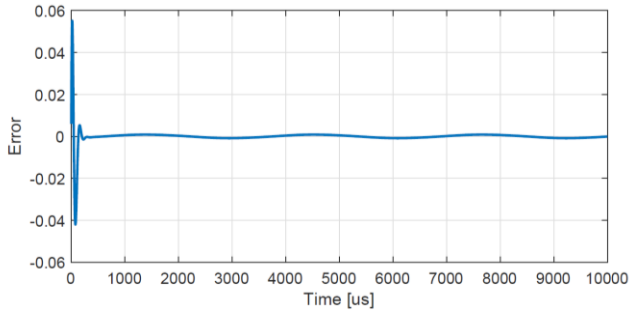


Figure 5
Trajectory tracking error of the Van der Pol oscillator under tuned RFPT control

Figure 6 shows the phase trajectory; Figure 7 shows the total control force while tracking of the Van der Pol oscillator under tuned RFPT control.

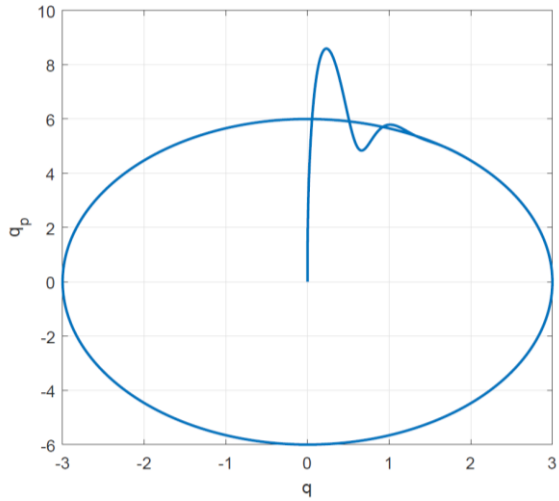


Figure 6
Phase trajectory tracking of the Van der Pol oscillator under tuned RFPT control

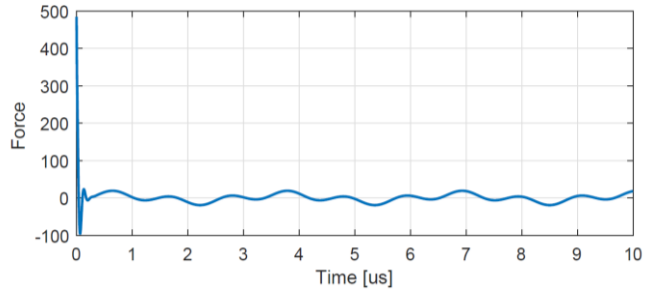


Figure 7
The total control force under tuned RFPT control

Conclusion

Based on the tests, it can be stated that the genetic algorithm is suitable for tuning the RFPT. Unlike PID, in the case of RFPT, there are no well-established, exact tuning methods, only individual parameters can be inferred from observations. Tuning can also be done manually, by monitoring the behavior of the system. However, the search area for each parameter is very large, it is possible that the right values can only be found after several attempts, however, this is not necessarily optimal. Thanks to GA, the whole operation can be automated, accelerated, and tuned correctly, under the right exit conditions. During the tests, I performed several measurements and tested the method on several dynamic models. In all cases, the controller was successfully tuned for the tracking error to nearly approximate zero.

References

[1] S. Banach: Surles opérations dansles ensembles abstraits et leur application aux équations intégrales (About the Operations in the Abstract Sets and Their Application to Integral Equations), Fund. Math., 1922, Vol. 3, pp. 133-181

[2] A. M. Lyapunov: A General Task about the Stability of Motion, Ph.D. Thesis, University of Kazan, Tatarstan (Russia), 1892

[3] A. M. Lyapunov: Stability of Motion, Academic Press, New-York and London, 1966

[4] P. Gahinet, P. Apkarian, M. Chilali: Affine parameter-dependent Lyapunov functions for real parametric uncertainty, In Proc. of Conference on Decision Control, 1994, pp. 2026-2031

[5] B. Csanádi, P. Galambos, J. K. Tar, Gy. Györök, A. Serester: Revisiting Lyapunov's Technique in the Fixed Point Transformation-based Adaptive Control, 2018 IEEE 22nd International Conference on Intelligent Engineering Systems (INES), Las Palmas de Gran Canaria, Spain, 2018 June 21-23, doi: 10.1109/INES.2018.8523923

[6] J. K. Tar: Application of local deformations in adaptive control - a comparative survey (invited plenary lecture). In: Proc. of the 7th IEEE International Conference on Computational Cybernetics (ICCC 2009), Palma de Mallorca, Spain, November 26-29, 2009, pp. 25-38, 2009

[7] J. K. Tar, J. F. Bitó, L. Nádai, J. A. Tenreiro Machado: Robust Fixed Point Transformations in adaptive control using local basin of attraction. Acta Polytechnica Hungarica, 6 (1): 21-37, 2009

[8] J. K. Tar: Towards replacing Lyapunov's 'direct' method in adaptive control of nonlinear systems (invited plenary lecture). In: Proc. of the 2010 Mathematical Methods in Engineering International Symposium (MME 2010), October 21-24, 2010, Coimbra, Portugal, 2010

[9] J. K. Tar, K. Eredics: Simulation studies on various tuning methods for convergence stabilization in a novel approach of model reference adaptive control based on robust fixed point transformations. Acta Technica Jaurinensis, 4 (1): 37-57, 2010

[10] M. Mitchell: An introduction to genetic algorithms, Comput. Math. with Appl., Vol. 32, No. 6, p. 133, 1996

[11] Y. R. Tsoy: The influence of population size and search time limit on genetic algorithm, Sci. Technol. 2003. Proc. KORUS 2003, 7th KoreaRussia Int. Symp. Vol. 3, No. 1, pp. 181-187, 2003

[12] Sándor Szénási, Imre Felde: Configuring Genetic Algorithm to Solve the Inverse Heat Conduction Problem. Acta Polytechnica Hungarica, Vol. 14, No. 6, pp. 133-152, 2017

[13] Dániel Stojcsics, András Molnár: AirGuardian UAV Hardware and Software System for Small Size UAVs, Int J Adv Robotic Sy, 2012, Vol. 9, 174:2012

[14] J. Holland: Adaptation in Natural and Artificial Systems, The MIT Press [Reprint edition 1992] (originally published in 1975)

[15] P. Parks: Liapunov redesign of model reference adaptive control systems, in IEEE Transactions on Automatic Control, Vol. 11, No. 3, pp. 362-367, July 1966, doi: 10.1109/TAC.1966.1098361

[16] G. Kreisselmeier, B. Anderson: Robust model reference adaptive control, in IEEE Transactions on Automatic Control, Vol. 31, No. 2, pp. 127-133, February 1986, doi: 10.1109/TAC.1986.1104217

[17] X. Meng, B. Song: Fast Genetic Algorithms Used for PID Parameter Optimization, 2007 IEEE International Conference on Automation and Logistics, Jinan, 2007, pp. 2144-2148, doi: 10.1109/ICAL.2007.4338930

[18] X. Shao, L. Xiao, C. Han: Optimization of PID parameters based on genetic algorithm and interval algorithm, 2009 Chinese Control and Decision Conference, Guilin, 2009, pp. 741-745, doi: 10.1109/CCDC.2009.5191861

[19] G. Lin, G. Liu: Tuning PID controller using adaptive genetic algorithms, 2010 5th International Conference on Computer Science and Education, Hefei, 2010, pp. 519-523, doi: 10.1109/ICCSE.2010.5593559

[20] J. K. Tar, I. J. Rudas, J. F. Bitó, J. A. T. Machado, K. R. Kozłowski: Decoupled fixed point transformation based adaptive control of the generalized 2 DOF 6-type Van der Pol oscillator}, 2009 European Control Conference (ECC), Budapest, 2009, pp. 579-584, doi: 10.23919/ECC.2009.7074465

[21] J. Kabziński: Adaptive tracking control of a Duffing oscillator with hard error constraints, 2016 21st International Conference on Methods and Models in Automation and Robotics (MMAR), Miedzyzdroje, 2016, pp. 1176-1181, doi: 10.1109/MMAR.2016.7575305

[22] T. A. Várkonyi, J. K. Tar, I. J. Rudas: Robust Fixed Point Transformations-based model reference adaptive control of inverted pendulums, 2011 IEEE 12th International Symposium on Computational Intelligence and Informatics (CINTI), Budapest, 2011, pp. 591-596, doi: 10.1109/CINTI.2011.6108471

[23] Zwe-Lee Gaing: A Particle Swarm Optimization Approach for Optimum Design of PID Controller in AVR System, IEEE Transactions on Energy Conversion, Vol. 19, No. 2, pp. 384-391, 2004

[24] T. O. Mahony, C. J. Downing, K. Fatla: Genetic Algorithm for PID Parameter Optimization: Minimizing Error Criteria}, Process Control and Instrumentation 2000 26-28 July, University of Stracthclyde, pp. 148-153

[25] T. K. Teng, J. S. Shieh, C. S. Chen: Genetic algorithms applied in online autotuning PID parameters of a liquid-level control system}, Transaction of the Institute of Measurement and control 25, 5 (2003) pp. 433-450