Simulated Real-Time Controller for Tuning Algorithm Using Modified Hill Climbing Approach Based on Model Reference Adaptive Control System

Ahmed Abdulelah Ahmed, Azura Che Soh,
Mohd Khair Hassan, Samsul Bahari Mohd Noor
and Hafiz Rashidi Harun

Abstract

In this chapter, an intelligent algorithmic tuning technique suitable for real-time system tuning based on hill climbing optimization algorithm and model reference adaptive control (MRAC) system technique is proposed. Although many adaptive control tuning methodologies depend partially or completely on online plant system identification, the proposed method uses only the model that is used to design the original controller, leading to simplified calculations that do not require neither high processing power nor long processing time, as opposed to identification technique calculations. Additionally, a modified hill climbing algorithm that is developed in this research is specifically designed, configured and tailored for the automatic tuning of control systems. The modified hill climbing algorithm uses a systematic movement when searching for new solution candidates. The algorithm measures the quality of the solution candidate based on error function. The error function is generated by comparing the system response with a desired reference response. The algorithm tests new solution candidates using step signals iteratively. The results showed the algorithm effectiveness to drive the system response. The simulation results illustrate that the method schemes proposed in this study show a viable and versatile solution to deal with controller tuning for systems with model inaccuracies as well as controller real-time calibration problem.

Keywords: real-time controller, auto-tuning algorithm, optimization, modified hill climbing approach, model reference adaptive control system (MRAC)

1. Introduction

The increasing complexity of industrial processes is always pushing for advancements and innovations in technologies involved in industrial processing, including

control systems engineering. The invention of computers paved the way for new and intelligent possibilities in control system technologies [1]. The last four decades witnessed a rapid movement towards the field of intelligent control [2–5]. This movement was being coupled with what were already growing new ways and paradigms of applying automatic control for even more decades prior to computer inception [6]. Control systems advanced from having a sensor-actuator relationship moulded together in feedback paradigm to complex multilayer interrelated sets of systems [2].

In designing control systems, system modelling can be considered the backbone of the design process. Many modelling techniques were introduced by researchers through time. The introduction of computers pushed towards better models [7]. However, no matter how good modelling techniques are getting, whether being based on analytical analysis or identification methods, no model would be a perfect match to the system it is depicting. There would always be an accuracy factor affecting the model quality.

The controller design is dependent upon the system model. As a result, the end-product controller quality is hugely dependent upon the model quality, and, to a degree, the model mimics the behaviour of the system it represents [8]. However, due to the vast and deep complexity of most systems humans encounter in the world, whether physical, economical or any other types of systems, humans tend to build mathematical models for control purposes as simplified as possible in order to simplify the controller design process. That simplification comes with a major flaw, which is a decreased accuracy of system presentation by the model. The effects of this flaw, however, are often not important or of insignificant consideration for the control system. However, sometimes it is effective to a degree of generating a degraded control quality over the system. That is especially present when models encounter even more degradation in quality due to system parameter fluctuation over time due to various operational effects.

However, sometimes control strategy should be built to deal with a high or variable model inaccuracy. Complex systems would often be represented with simplified models to reduce calculation time and efforts [9], which result in less accurate models [8], while in many cases, the model accuracy degrades over time due to either sudden system variable change due to undisclosed reasons or gradual variable change from normal wear and tear of system components [10].

Due to control system design procedures being highly dependent on the mathematical model of the controlled plant [9], a high model quality and accuracy is critically needed. However, in some dynamic systems, it is very difficult (or even impossible) to have models with good accuracy that are sufficient to predict the plant behaviour in a way that an acceptably controlled performance can be produced.

On the other hand, sometimes even if mathematical models are sufficiently accurate in a way that a good controlled performance can be obtained, long-term operation (or even short term in some cases) gradually increases the difference between the plant and its dynamical model, which, in turn, would lead to a degraded performance.

It is a common task in industrial applications to recalibrate the control system periodically, as the plant parameters suffer various fluctuations from their original values that were used in designing the control system. The calibration procedure usually requires professional attendance, which adds up to more maintenance costs. Also, the experimental nature of the manual calibration often requires at least part of the plant operations to be halted [2, 11].

Instead of relying on manual calibration, this research proposes an automatic tuning scheme and algorithm specifically designed for control systems to deal with model inaccuracies and parameter fluctuations during real-time operation without the need to halt plant operation.

2. Model reference adaptive control (MRAC)

Adaptive control systems were among the first techniques in automatic control to tackle the problem of parameter fluctuation [12]. The various techniques required a continuous system observation or identification to improve the model quality while adapting the controller parameters according to the new model [13, 14]. However, the general approach of adaptive control is different to that of real-time optimization. Real-time optimization techniques attempt to tune or optimize the control system generally by changing the operating conditions in order to get the desired response. There is no online modelling involved. It was introduced mainly in chemical processes and plants to deal with the autonomous calibration problem [15]. However, it is possible to generalize its framework to deal with other types of control systems [16].

Model reference adaptive control (MRAC) is one of the most important adaptive techniques in control engineering [12, 13], as it provides a good viewpoint to thoroughly analyse adaptive systems [12].

The MRAC system structure is generally considered to have two feedback loops [12, 17], an inner feedback loop that is the ordinary feedback that is compared with the set point and fed to the controller and an outer feedback loop that modifies controller parameters based on the adaptation mechanism. In this technique, a reference response is to be followed by the system. The reference response is generated by using a prebuilt reference model [12]. The controller parameters are adjusted based on the difference error between the reference model response and the controlled process response as shown in **Figure 1**.

The adaptation algorithm adjusts the controller parameters so that the error is reduced to zero [12]. Parameter estimation or process measurements help in forming the adaptation rule that the controller would be adjusted according to.

MRAC uses one of three mechanics to adjust controller parameters by finding an adaptation law. The Massachusetts Institute of Technology (MIT) rule is a gradient method that was the first used mechanic [12]. However, the MIT rule could sometimes lead to unstable system response [13]. Later, the Lyapunov stability theory and the passivity theory were also used to generate the adaptation law that was proved to be robust and stable [12, 13].

It should be noted that there is a large correlation between MRAC and self-tuning regulator (STR). Actually, some STRs can be considered as MRAC and vice versa [12, 17].
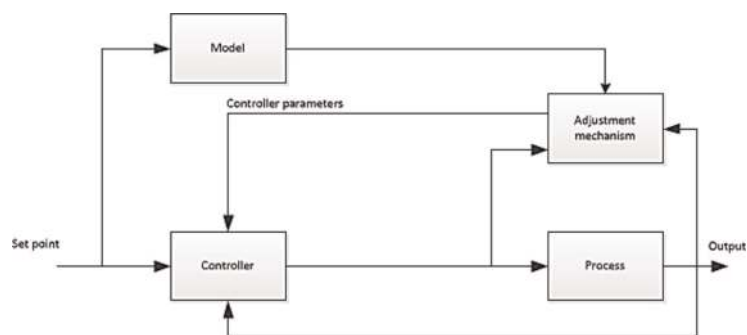

Figure 1.
Model reference adaptive control schematic diagram [12–13, 17].

3. Overview of hill climbing algorithm

It was proved that although process models are often inaccurate, they are sufficient to provide general knowledge for real-time optimization problem [18], while real-time optimization problem can be solved iteratively [16, 19]. The tuning operation includes searching for new actuator settings (e.g. gains) to achieve the desired performance [16]. Many optimization algorithms are available, but most of these algorithms are not suitable for real-time tuning due to several reasons such as time-consuming calculation or the algorithm's inability to provide a transitional solution until a satisfactory solution is found [20]. Obviously, the latter is essential for real-time operation.

In this research, a real-time controller tuning algorithm based on the principles of hill climbing optimization algorithm is proposed. The algorithm is suitable for controller automatic tuning in real-time. Hill climbing is one of the oldest traditional optimization algorithms. The procedure begins by selecting an arbitrary solution in the searching space and then testing the surrounding solutions, one at a time. If the algorithm finds a better solution than the currently selected solution, the better solution is selected, and the procedure is done again until the algorithm cannot find a solution that is better than the currently selected one, and at that point, the optimization stops, and the best found solution is considered the best solution [21].

In **Figure 2**, searching for a maximum height point is assumed. Let's suppose that the arbitrary starting point was point a. When the optimization starts, hill climbing would check the sides of point a and decide to move in the shown direction, since the left side gives a larger (height) value than the value of point a. On the other hand, the algorithm would not select to move to the right, since the value is less than that of point a. This movement would continue until the selection reaches the maximum point denoted as 'maximum height (global)' at which the optimization is considered finished and the selected point is the optimum, since no more movement is possible because points on both sides hold less value than the selected one.

However, starting arbitrarily at point b of **Figure 2** would result in the optimization to stop when reaching the point denoted as 'maximum (local)'. When the optimization starts from point b, the selection would take the direction shown in the figure. But that movement would stop upon reaching the local optimum point because the algorithm would be trapped at a point that is better than all its neighbours, although that point is not the overall best point of the search space.

The above example demonstrates a major flaw of the hill climbing algorithm, which is that the choice of the first solution at beginning of optimization can lead to a huge impact on the final result in case the function being optimized has multiple
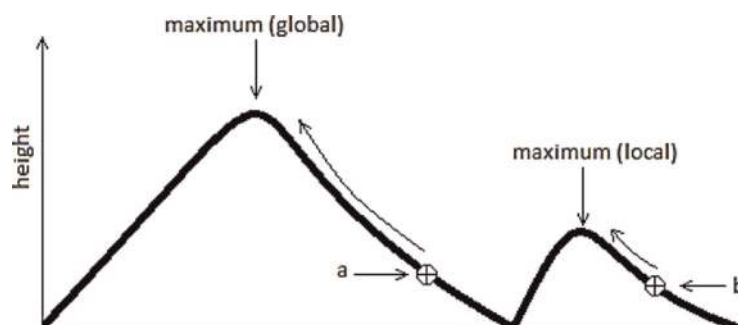

Figure 2.
An illustration of how hill climbing works.

optimum points [22, 23]. Therefore, hill climbing can easily be trapped in local optimum if applied to functions with non-uni-model landscapes [21]. In the same way, it can be trapped in plateaux or ridges also [21–23].

Even with its shortcomings, hill climbing is considered one of the best local searching algorithms [21] because it tends to move towards the local optimum quite fast [21, 23]. Therefore, many modifications to simple hill climbing were proposed by researchers to overcome its negatives and making it usable for a wider scope [21].

Stochastic hill climbing uses a random choice on the next uphill movement, which can give better results in some cases. Another variant is first-choice hill climbing, which is a modified form of stochastic hill climbing. In this algorithm, random steps are generated, and the movement occurs when a solution better than the current solution is found [21]. One of the other modifications is random-restart hill climbing. In this algorithm, simple hill climbing is used but with restarting from a new arbitrary point after each local search finds the optima. Hence, the probability of finding the global optimum is increased after several repetitions [21, 23].

Hill climbing, at its simple form, is very useful for real-time controller tuning, since the algorithm moves in the search space with small steps by changing one variable at a time while promoting a transitional solution until a final solution can be found [6, 10, 20]. On the other hand, all modifications to hill climbing that could be used to overcome its shortcomings are based on a random selection or movement in the search space [21]. Therefore, these variants produce the same consequences of genetic algorithms regarding controller tuning in real time. Unpredictability and instability of the response are not far from being possible.

4. The modified hill climbing algorithm

A systematic movement modification to hill climbing algorithm is suggested. Our modification is targeted at real-time tuning of control systems.

The main concept suggested as modification is to keep exploring the neighbourhood of the current best solution and keep moving candidates further from the best solution in both directions alternately (moving towards a bigger and a smaller value for each variable, one variable at a time), even if the surrounding candidates yield a degraded quality, until a satisfactory solution is found after searching for best values to all the elements of the controller. This is done by changing one variable at a time and testing the system by applying a testing step input iteratively [16].

An offset value must be specified that represents the minimum movement size from the current best value to the next candidate. This offset will be used as the discretization interval to find the movement amount for each variable in each iteration. That is done by simply increasing the movement amount by the value of the offset. **Figure 3** shows the general tuning operation workflow.

Instead of moving in one direction when searching for new candidates in classic hill climbing (the direction of the best neighbourhood of starting position), and keeping on moving until no further improvement in the fitness function is met, searching in the proposed algorithm would keep moving alternately between two directions and will not stop at maxima points. The searching for new candidates would stop when fitness (error function result) achieves the designer's requirements.

To demonstrate the moving mechanics when searching for candidates, it will be assumed that a function that has a single variable parameter is required to be tuned in order to achieve certain function fitness. It is also assumed that an arbitrary starting value of 'a' is assigned to the variable parameter, which gives a fitness value
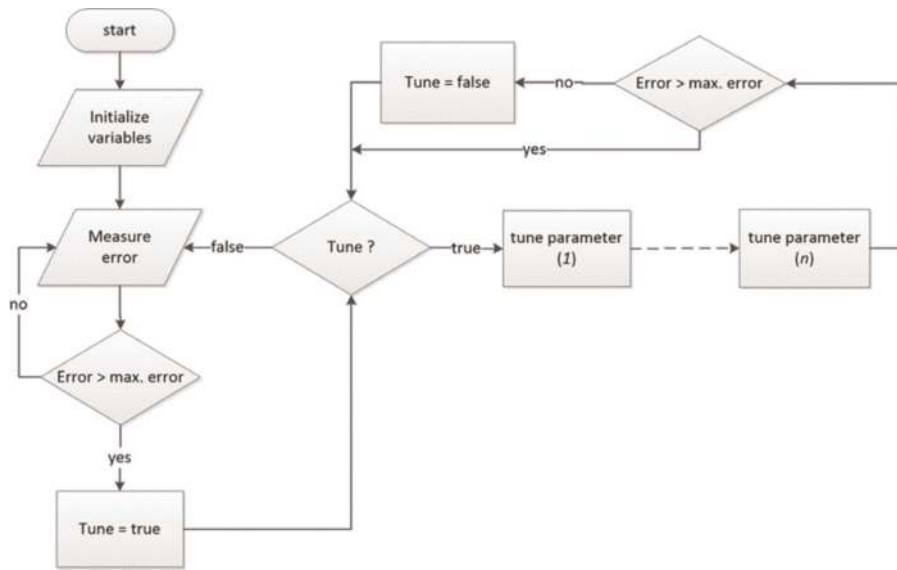
Figure 3.
Tuning algorithm schematic diagram for system with (n) variables.
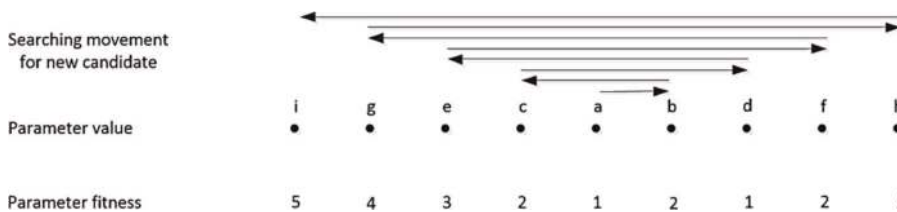

Figure 4.
Demonstration of movement when searching for new candidates.

of '1', as demonstrated in **Figure 4**. It is also assumed that the performance would be acceptable if fitness is equal or greater than '4', and therefore, tuning will stop if a fitness value of '4' or greater is reached, regardless if that value is the best value possible or not.

Each point in **Figure 4** is larger than the point to its left by an amount, called 'offset', and smaller than the point to its right by the same amount. The offset is the smallest movement the tuning algorithm is allowed to move when searching for candidates. The offset value must be determined by the designer depending on the system sensitivity to gain changes.

Considering tuning the function of **Figure 4** again, if the tuning started from point 'a', where tuning is required due to it having fitness of '1', the algorithm moves the candidate selection to the right by the amount of offset, so point 'b' is chosen as the new candidate. When testing fitness of point 'b', the algorithm finds it better than that of point 'a', and therefore, point 'b' will be promoted as the current best solution. However, fitness is still less than the minimum required value of '4'; hence, tuning continues. The movement amount is increased by the amount of offset and reversed in direction, which will result in selecting point 'c' as the new candidate. Testing the fitness of point 'c' shows that it is equal to the current best fitness, and as a result, no promoting is required, since no advantage will be acquired. Next, the movement amount is increased by the amount of offset again and reversed in direction.

The next candidate will therefore be point 'd', which has fitness less than the current best fitness, so this point will not be promoted. Increasing the movement amount by offset and reversing direction will select point 'e' as the next candidate.

There is still a fitness of point 'b' as the best, so point 'b' remains promoted as the best parameter value until now, even though it doesn't satisfy the design requirements. When checking the fitness of the last selected candidate, point 'e', the algorithm finds that this fitness is better than the last best fitness. Therefore, point 'e' is promoted as the best current solution, although it also doesn't satisfy the design requirements. The performance of point 'b' (and all other points other than the best point) is forgotten and won't be memorized. The same procedure will continue until a parameter value with a satisfactory fitness is found (point 'g' in this example) by the algorithm, and then the tuning will stop, and the last best parameter selection is fixed to the system. If fitness is degraded for any reason once again, the tuning procedure will start again until it finds new satisfactory parameters. It is noticed that the algorithm doesn't necessarily lead to finding the global best solution (point 'i' in the example), but if a solution exists, then it will be found, since the algorithm would theoretically pass through every possible candidate unless a solution is found.

Steps being done when tuning consists of the following:

1. Calculate variable candidate according to the following equation:

$$newV_c = oldV_c + (V_{md} * V_m)$$
(1)

2. Store current variable value as the best.

3. Change the variable value to the candidate value.

4. Check performance.

    i. If not improved, restore the best value to the variable, and set flag to change next variable.

    ii. If improved, reset all other variables movements around their best values.

5. Calculate the next direction of movement.

$$newV_{md} = -1 * oldV_{md}$$
(2)

6. Calculate the next amount of movement.

$$newV_m = oldV_m + offset$$
(3)

7. Check flag to decide on what variable to tune.

Notes
V_c: variable candidate
V_m: variable candidate movement amount
V_{md}: variable candidate movement direction

The following **Figure 5** below demonstrates operations to be done when tuning each single variable.

Like most optimization algorithms in general, and hill climbing specifically, the use of the algorithm does not guarantee the best solution for certain systems under certain conditions. However, if some solutions do exist, it is guaranteed that this modified algorithm will find one of them. The solution under the same circumstances would always be unique assuming similar starting conditions. However,
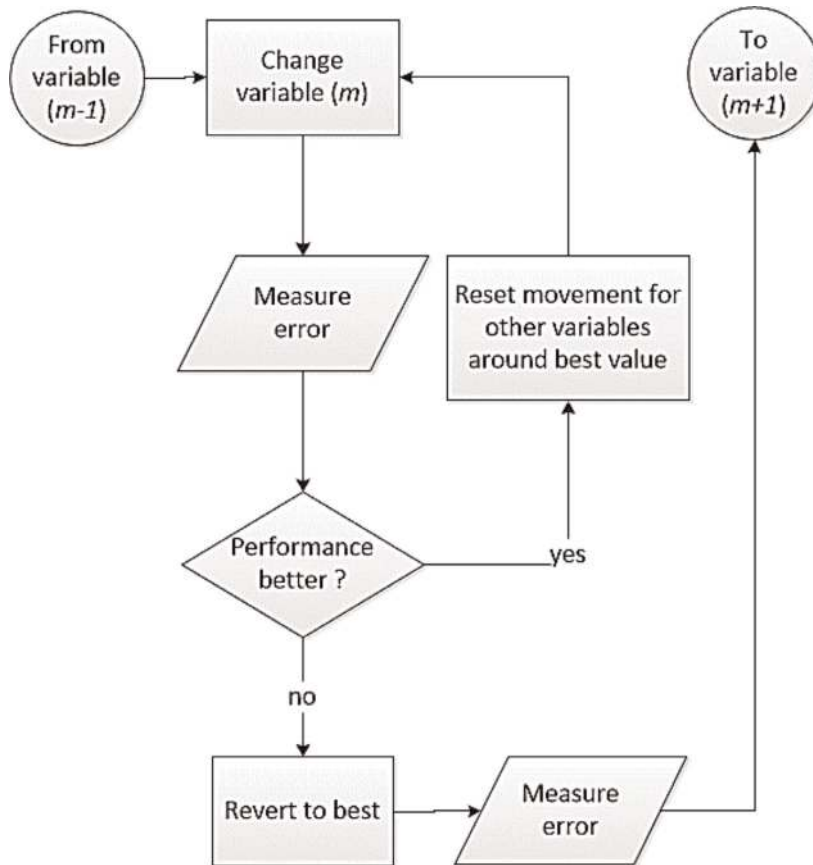
Figure 5.
Operations when tuning the m-th variable.

during the experiments at least one solution was assumed to be available within a certain error margin.

5. Control paradigm

The suggested control paradigm is to compare the actual system response with a desired reference response and then generate an error function that decides on whether to tune the system controller or not as demonstrated in **Figure 6**. This paradigm is inspired by the adaptive control paradigm, although it is not a complete replication, since no parameter estimation is involved. The desired reference response is generated by designing a controller for the system model.

The tuning drives the actual system response to act the same way as that of the modelled system. This is to be done by tuning the actual system controller [24].

In order to make a decision on whether it is required to tune the process controller or to check whether the latest modification improved the performance, a comparison between the desired reference response and the process response must be conducted.

This comparison generates an error signal using the error function block in **Figure 5**. If the error signal is smaller than the maximum allowed error, then tuning is not needed. If the error signal is larger than the maximum allowed error, then tuning is needed. This is also done after each tuning iteration. If the error signal becomes smaller than the maximum allowed error after making a tuning pass over all the tuneable parameters, tuning will stop. Otherwise, tuning will continue.
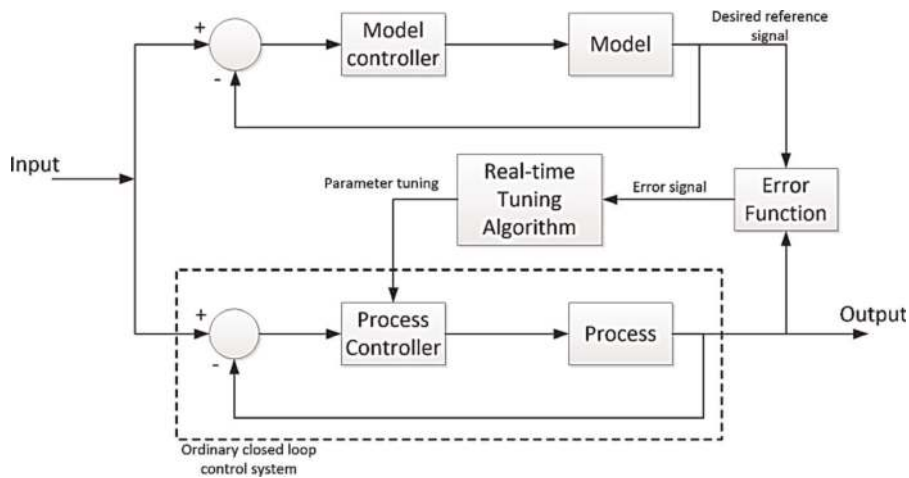
Figure 6.
Schematic diagram of the proposed design.

The error function itself is not a mere subtraction equation. In order to get sufficient clarification about the amount of the difference between the process response and the desired reference response, the designer needs to consider a period that covers both transient response to the experimental step input change and a steady-state response. Consequently, the designer must specify an inspection period so error measurements can be collected. Next, the error function can be the average value of the errors measured over a specified period of time, or it can be the summation of those error values or any other suitable function. The period can be identified from prior knowledge of the system, since the designer has a model of the process; though not accurate, it can give a general assessment of the required period. Meanwhile, error reading should be put in discrete form as pairs of model and process outputs that were taken simultaneously.

Hence, it can be considered that the error function is a cost function that is to be minimized below a certain value. That value is the maximum allowed error, which is also to be specified by the designer, reflecting the closeness required between process response and the desired reference response.

If tuning is needed (when the error generated by the error function is larger than the maximum allowed error), then the tuning algorithm will start to change the controller variables (e.g. K_p, K_i and K_d in a PID controller) one by one and measure the error after each single variable change. If the error is reduced, the algorithm would keep the new value for the variable and move on to change the next variable. Otherwise the algorithm would reset the variable to its best value and move on to change the next variable.

If a value substitution happened (when finding a better value for one of the variables), the movement amount for each other variable is reset to the minimum, which is the offset value. This is to make sure that the algorithm would search the neighbourhoods of the best values of variables again when any change happens to one of them in order to find the best possible values for them, since they could be strongly related to each other dynamically. For example, if we have a PID controller being optimized and the value of K_p is increased, then the system will be faster but will generate a higher overshoot. So, K_d should be tuned a little to lower the overshoot.

This procedure should be repeated until the error becomes less than the maximum allowed error. Then, the tuning will be stopped until the error fluctuates outside the allowed margins again. The model controller can be of any type.

5.1 Algorithm properties

To emphasize the advantages and usability of the proposed modified hill climbing algorithm, this section discusses the properties of the algorithm under real-time controller tuning considerations.

Firstly, the algorithm uses very simple calculations during system monitoring and controller tuning. All equations involved in the design consist of basic combinations of principal operations such as addition or division only. Therefore, the algorithm retains high applicability for real-time execution, since the calculation time of such simple equations can be considered irrelevant in current electronics; hence, it is very easy to be implemented using embedded computer systems. The shear simplicity of the calculations is largely attributed to the lack of online parameter estimation and to the simplicity of searching for a candidate mechanism which is partly inherited from hill climbing.

Secondly, due to its hill climbing inheritance, the algorithm is very applicable to tune multivariable controllers. Dealing with controller variables one at a time is one of the core principles taken from hill climbing that serves very well to overcome the tuning task in a systematic and procedural approach, without the need for randomness in candidates' selection which nullifies the ability to tune the controller based on the dynamic relation between the controller variables.

Thirdly, in principle, the algorithm is not tied to a specific type of controller. Any type of controllers can be tuned theoretically, using any heuristic tuning method. Although certain algorithms can be more suitable for certain types of controllers than to others, some controllers are better not be tuned heuristically. However, in the case study included in this paper, PID controllers were used. This is a result of PID controllers themselves being suitable for heuristic controller tuning in general, and real-time tuning specifically, as PID has few parameters to tune, reducing tuning time and effort.

Fourthly, an important property of the proposed algorithm, which is also inherited from classic hill climbing, is the ability to promote transitional solutions until a satisfactory solution can be found. In fact, this transitional solution is part of the overall tuning procedure, since the proposed algorithm repeatedly compares last candidate performance with the last best available solution performance. This property is essential for real-time tuning, as it keeps the controller running with the best possible variable selection periodically between new candidate testing instances, sustaining the best possible output performance for the furthest possible period and, in turn, reducing the chances of instability or other undesirable possibilities.

5.2 Important aspects to consider when applying the proposed design

Many design considerations should be taken during the design of any control system. These considerations are dependent on the performance requirements and design methodology. Here the concentration is put on the considerations that are related to the real-time controller tuning paradigm using the modified hill climbing algorithm.

Quality of the controller (model controller in **Figure 6**) used to generate desired reference response is a very sensitive aspect. Great care should be taken as this step will decide the shape of the response that we are aiming for. The desired reference response should be physically viable (within an allowed error margins).

The error function must be made so that it can detect the variable changing effect on the response. That is also directly related to the maximum error value and

how the error is (difference between model and process outputs) being recorded (the period being considered after the change and the timing of collecting the signal). Consequently, the period spent on recording the outputs of the model and process should start from, or before applying, the experimental step input and end sometime after reaching a steady state.

The offset value should be chosen in the middle ground between the system sensitivity to variable or gain changes and the speed at which the candidates will move in the search space. To clarify this, it should be noted that the larger the offset value, the faster the algorithm reaches a solution, since the offset value would affect the new candidate movement amount. However, that would be on the expense of a reduced fine tuning accuracy, since no smaller steps than the offset is possible, which could mean missing a better parameter value point close to a relatively good point.

If necessary, some simple conditions can be applied to avoid well-known undesired candidates (e.g. negative gains for PID controller).

6. Simulation experiment

In order to demonstrate the design success, it was applied to a twin rotor MIMO system (TRMS). The results were obtained using simulations on MATLAB and Simulink software. The TRMS system used in the experiment is provided by Quanser [25]. This system is a two degree of freedom (2-DOF) mechanical system. It resembles a helicopter system using two rotors, one as a main rotor and one as a tail rotor, as seen in **Figure** 7.

It has two degrees of freedom: one for pitch angle and one for yaw angle. The device manual includes a linearized state-space model of the system [25], while the software bundled with the device includes a nonlinear Simulink model of the system. The linearized model was used to generate the desired reference signal, which was used for tuning the nonlinear model controller in Simulink.

Figure 7.
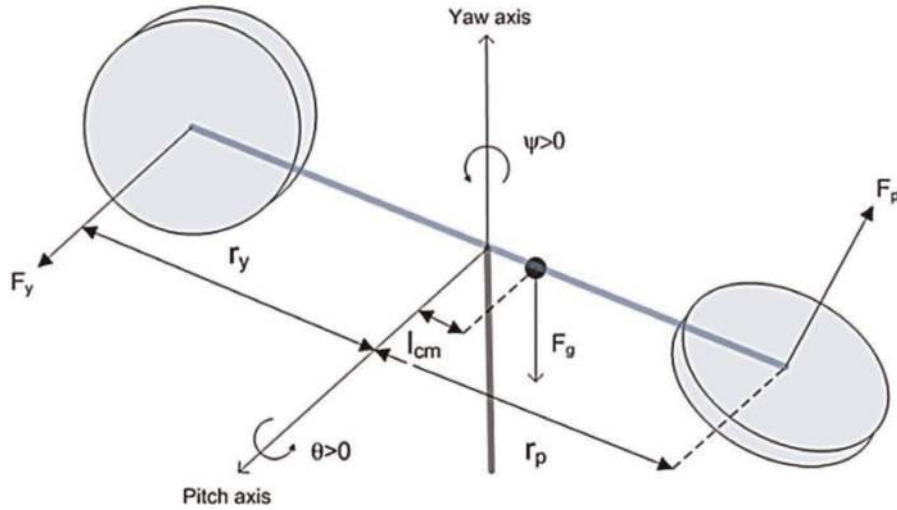*Quanser 2-DOF helicopter [25].*

Figure 8.
Free-body diagram of Quanser helicopter [25].

Each degree of freedom is driven using a DC motor. The front head of the helicopter moves in the pitch angle and is driven by a 24 volt motor. The tail of the helicopter moves in the yaw angle and is driven by a 12 volt motor. The process is a highly nonlinear system with a very strong cross coupling between pitch and yaw angles.

The model uses the free-body diagram of **Figure 8** below.
The following modeling conventions were assumed [25]:

- The helicopter is horizontal when $\theta = 0$.

- The pitch angle increases positively, $\frac{d\theta(t)}{dt} > 0$ when rotating in the counter-clockwise direction.

- The yaw angle increases positively, $\frac{d\psi(t)}{dt} > 0$ when rotating in the clockwise direction.

- The pitch angle increases when pitch thrust force is positive, $F_p > 0$.

- The yaw angle increases when the yaw thrust force is positive, $F_y > 0$.

The model is a state-space representation of the system using the standard form of

$$\dot{x} = Ax + Bu \tag{4}$$

$$\dot{y} = Cy + Du \tag{5}$$

where the state vector and output vector are defined by

$$x^T = \left[\theta, \lambda, \dot{\theta}, \dot{\lambda}\right]$$

$$y^T = [\theta, \lambda]$$

where **A, B, C** and **D** are defined as (all parameters are defined in [25]):

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -\dfrac{B_p}{J_{eq\_p} + m_{heli}l_{cm}^2} & 0 \\ 0 & 0 & 0 & -\dfrac{B_y}{J_{eq\_y} + m_{heli}l_{cm}^2} \end{bmatrix}$$

$$B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \dfrac{K_{pp}}{J_{eq\_p} + m_{heli}l_{cm}^2} & \dfrac{K_{py}}{J_{eq\_p} + m_{heli}l_{cm}^2} \\ \dfrac{K_{yp}}{J_{eq\_y} + m_{heli}l_{cm}^2} & \dfrac{K_{yy}}{J_{eq\_y} + m_{heli}l_{cm}^2} \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

$$D = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

The system exhibits a high cross coupling between the pitch and yaw angles. The main feature of the suggested controller is to use two separated PID controllers, one for each degree of freedom, so that a simple design procedure can be applied.

Each of the fuzzy controllers, a PID controller and an LQR, was used to generate the desired reference response in order to tune PID controllers that govern the nonlinear model for both pitch and yaw. In all the cases, an arbitrary PID was used in the beginning. The values of Kp, Ki and Kd were all set to a gain value of 1. Both the fuzzy controller and PID controller were designed for the system, while the LQR is taken from a suggested design in the device manual [25].

7. Results and discussions

In addition to observing how much the process will follow the desired reference response in the twin rotor system, it will be checked whether the decoupling strategy is successful enough, since the twin rotor system inherits strong coupling between pitch and yaw angles.

In all the following cases, the initial parameter values of the process PID controller were set arbitrarily to 1 before the start of the tuning process. Stable responses were already achieved but with unacceptable performances. The performances of the initial PID controllers are shown when compared to the tuned systems below.

7.1 TRMS process controller tuning using PID controller to generate the desired reference response

In this case, a huge improvement was achieved in the process response after tuning. The tuning algorithm steered the process output to follow the desired reference response closely. Pitch angle controller tuning results are shown in **Figure 9**. The tuning results for the pitch were nearly perfect.
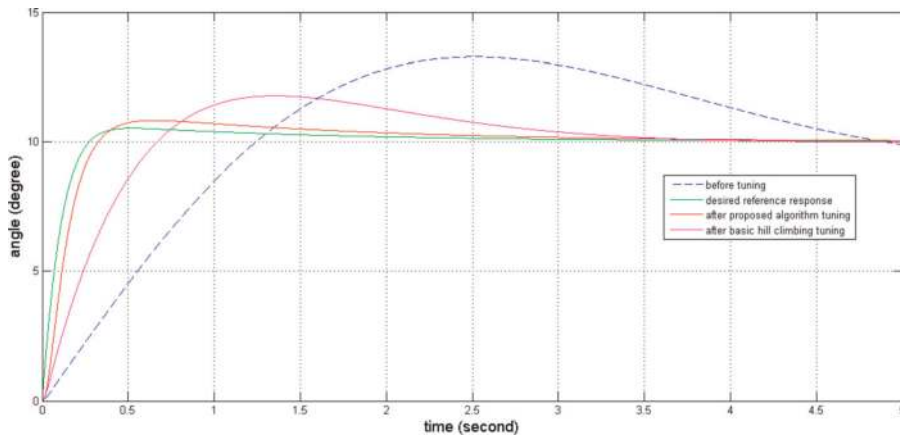
Figure 9.
TRMS pitch angle response before and after tuning with desired reference response generated by PID.

Tuning using the basic hill climbing did not achieve the same level of success in following the reference response; although the overshoot was reduced, the system response was slower in comparison.

The above result was achieved because the desired reference response was physically viable and that, in turn, it helped to relatively reduce the maximum allowed error margins; therefore, a very close following was achieved. The final tuned parameters for pitch angle PID were 1.9 for proportional gain, 1.2 for integral gain and 0.2 for derivative gain.

Figure 10 shows the error signal (difference between desired reference response and process output) for the pitch angle before and after tuning. The excellent tuning results are obvious in the error signal graph.

The yaw angle controller tuning also achieved very good results in bringing the process response closer to the desired reference response than the initial response, as seen in **Figure 11**.

Following the reference response was very good, and the overshoot was reduced dramatically, although not perfectly as in the pitch response. The small overshoot seen at the fifth second is due to coupling. This brings the researcher to mention that the yaw angle looks more difficult to be stripped from coupling effects than the pitch angle. Hence, there is no considerable effect when changing yaw angle on pitch angle; however, changing the pitch angle still has a small effect on the yaw
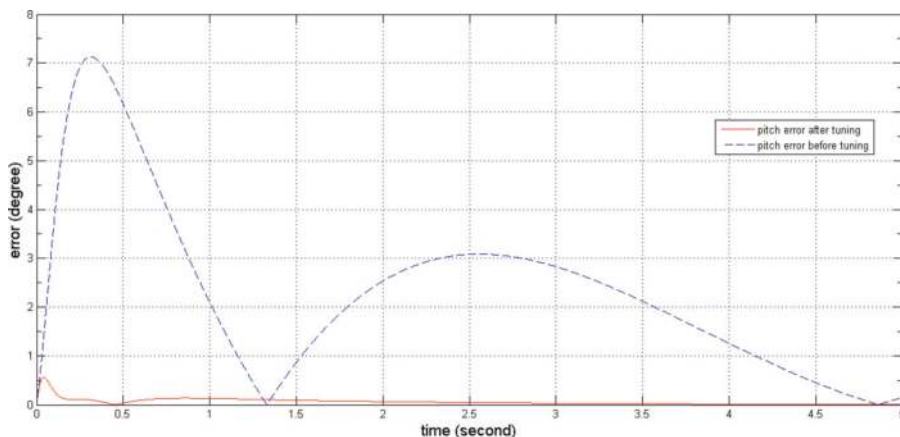

Figure 10.
Pitch angle response error signal to step set-point change before and after tuning when using PID to generate the desired reference response.
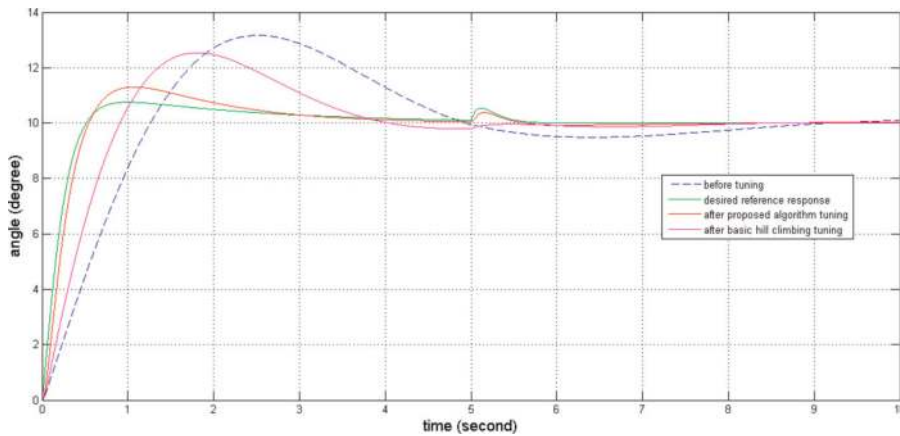
Figure 11.
TRMS yaw angle response before and after tuning with desired reference response generated by PID.
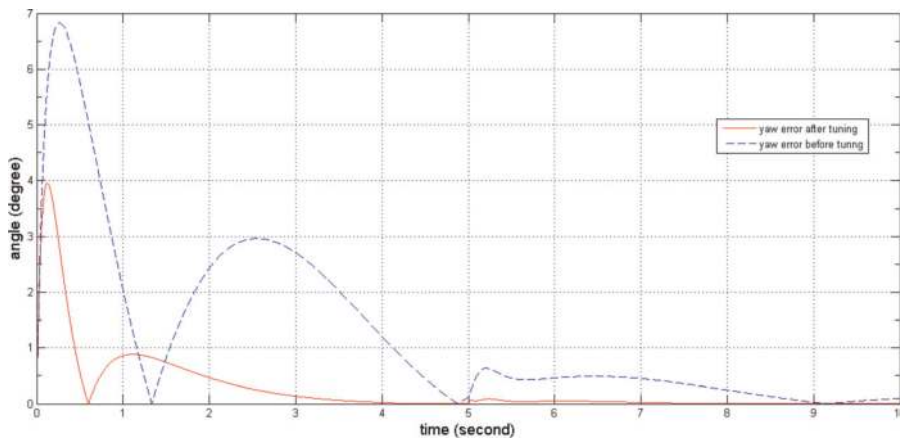

Figure 12.
Yaw angle response error signal to step set-point change before and after tuning when using PID to generate the desired reference response.

angle. The effect is very minimal, and the system decoupling efforts can be considered marginally successful. The final tuned parameters for the yaw angle PID were 1.7 for proportional gain, 1.2 for integral gain and 0.4 for derivative gain.

The error signal for the yaw angle before and after tuning is shown in **Figure 12**, where improvement in the response can clearly be seen.

7.2 TRMS process controller tuning using fuzzy controller to generate the desired reference response

The pitch angle response tuning, when using the fuzzy controller, was successful, as expected. Since the researchers had designed the desired reference response with an emphasis on reducing the overshoot with high speed performance, the tuned response of the process followed the design guidelines properly. It can be observed in **Figure 13** below how the process response moved from the initial PID controlled response to the final tuned PID controlled response quite well.

The result shows successful tuning of the process controller using the proposed algorithm. The effects of coupling were unnoticed, denoting the successful decoupling strategy that was designed, which was suggested earlier. The final parameters for the pitch angle process PID after tuning were 2.1 for proportional gain, 0.4 for integral gain and 0.7 for derivative gain.
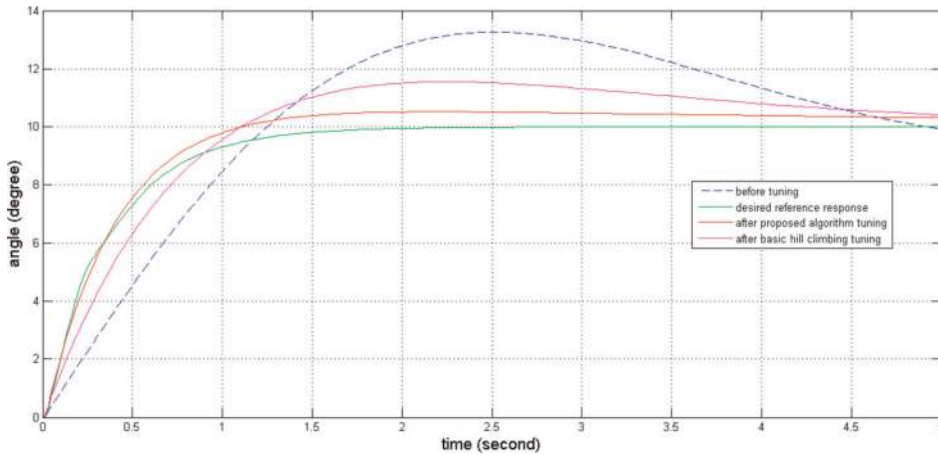
Figure 13.
TRMS pitch angle response before and after tuning with desired reference response generated by fuzzy controller.
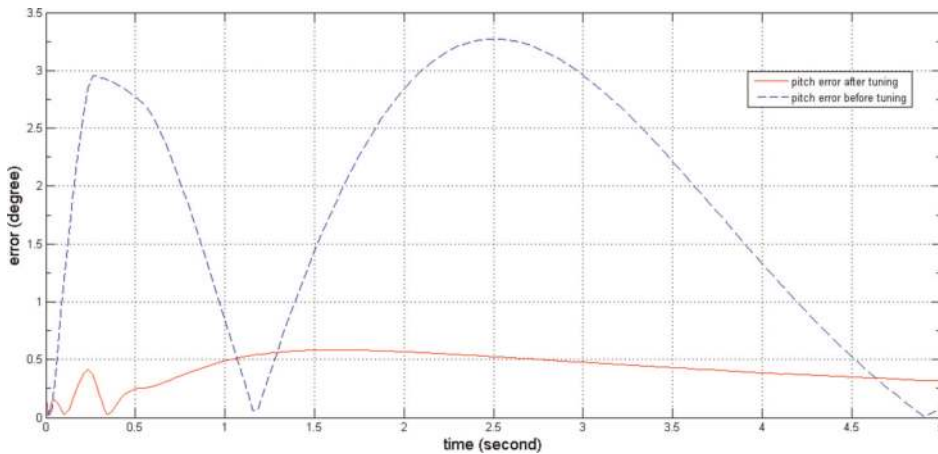

Figure 14.
Pitch angle response error signal to step set-point change before and after tuning when using fuzzy controller to generate the desired reference response.

The error signal between the desired reference response and the process response was reduced largely by the tuning algorithm. **Figure 14** shows the error signal graph of the pitch angle, both before and after tuning.

The yaw angle controller tuning did steer the process response from the initial towards the desired reference response, resulting in better performance; however, the response did not follow the reference perfectly. Namely, the response did have a considerable overshoot, unlike the desired reference response, as shown in **Figure 15** below. This came as a result of pushing the desired reference response beyond the physical nature of the actual process. The actual yaw angle system is inherently highly under-damped. Consequently, obtaining a well-damped response with decent speed seems unrealistic.

The final parameters for the yaw angle process PID after tuning were 1.7 for proportional gain, 0.7 for integral gain and 0.9 for derivative gain.

The error signal between the desired reference response and the process response was reduced by the tuning algorithm. A sizable error can be observed from the start of second second until the fourth second in **Figure 16**, showing overshoot difference as observed above. However, the error function was low enough to grant the response as acceptable, since the error at other areas is minimal.
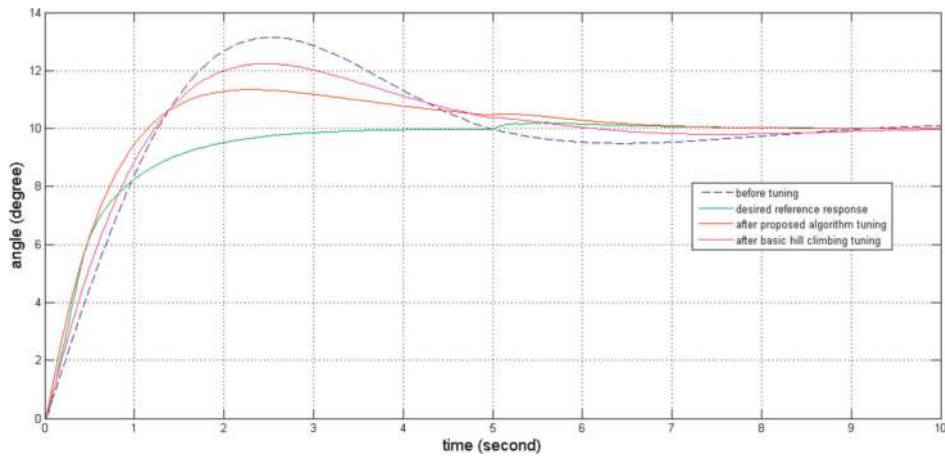
Figure 15.
TRMS yaw angle response before and after tuning with a desired reference response generated by fuzzy controller.
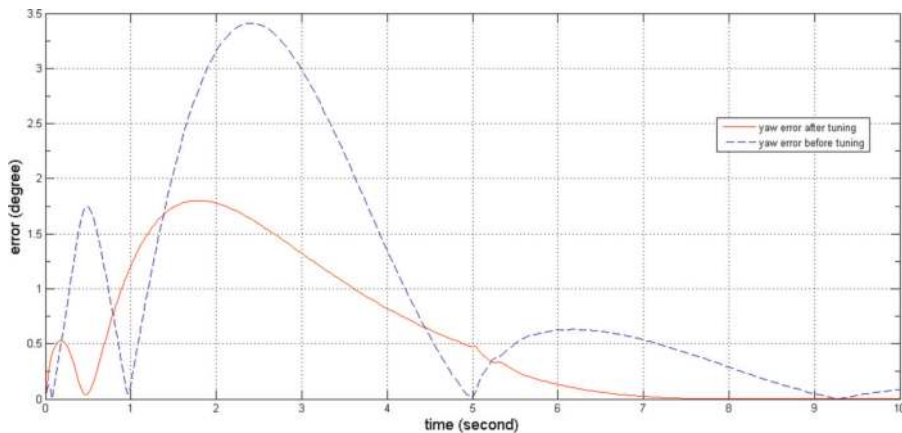

Figure 16.
Yaw angle response error signal to step set-point change before and after tuning when using fuzzy controller to generate the desired reference response.

Both pitch and yaw outputs that were tuned using the basic hill climbing gave an acceptable response speeds, but the overshoot amounts were high comparatively.

7.3 TRMS process controller tuning using LQR to generate the desired reference response

The tuning algorithm, as in other cases, succeeded in reshaping the process response so that it followed the desired reference response closely. This case gives evidence that the performance of the tuned process controller is highly determined by the performance of the desired reference response. In **Figure 17**, it can be seen that the tuned response exhibits a relatively high overshoot with slow response, because the desired reference response gave similar results.

The final PID parameters for pitch angle process after tuning were 1.5 for proportional gain, 0.6 for integral gain and 0.9 for derivative gain.

Although the response was not the best when compared to the other cases, the approach has again proven to be successful, as the error plot in **Figure 18** shows the improvement in reference following from the initial non-tuned response to the final tuned response. Tuning will always depend on error to the reference response, regardless of acquired performance, as shown in this case.
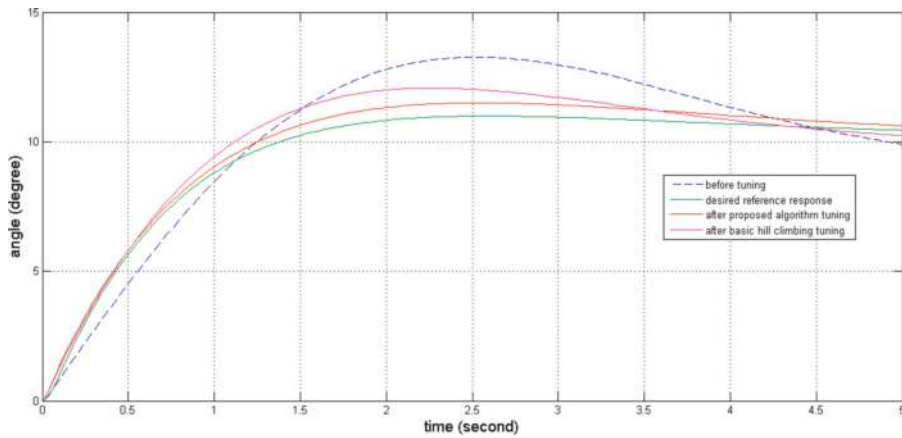
Figure 17.
TRMS pitch angle response before and after tuning with desired reference response generated by LQR.

Figure 18.
Pitch angle response error signal to step set-point change before and after tuning when using LQR to generate the desired reference response.

The yaw angle response can be found in **Figure 19**.

The response is no different from the observations taken from the pitch response, as the low-quality model controller generated a low-performance desired reference response that, in turn, shadowed the performance of the tuned process

Figure 19.
TRMS yaw angle response with desired reference response generated using LQR.
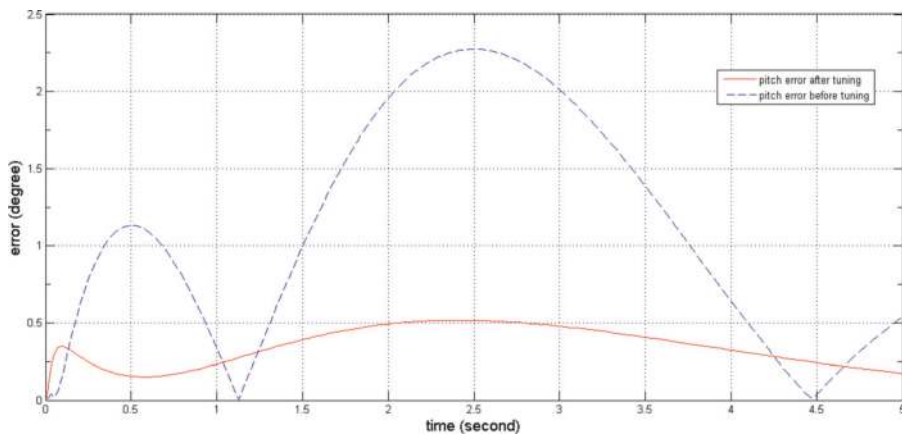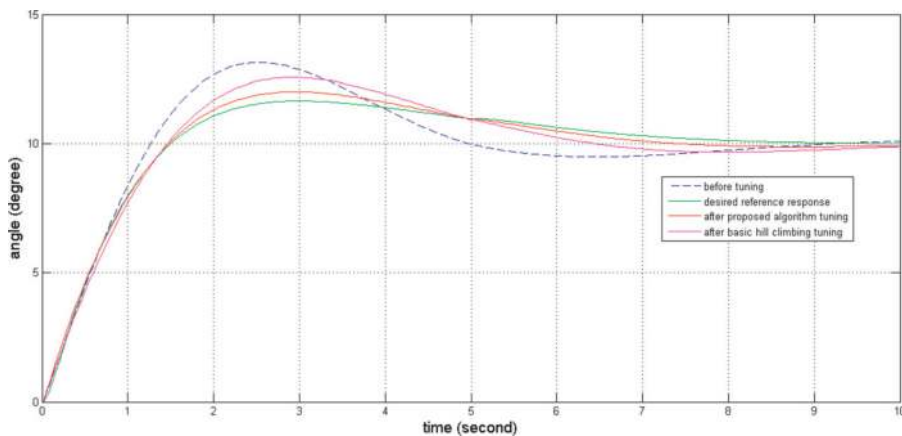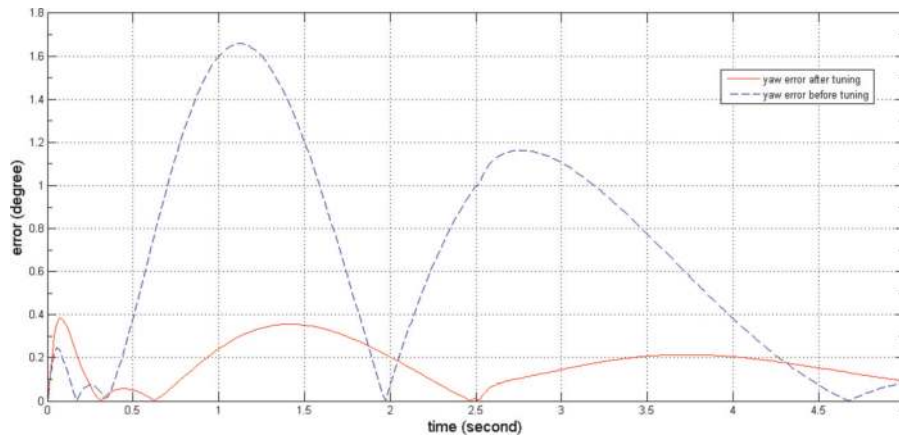
Figure 20.
Yaw angle response error signal to step set-point change before and after tuning when using LQR to generate the desired reference response.

PID controller. The final PID parameters for the yaw angle process after tuning were 1.4 for proportional gain, 0.7 for integral gain and 1.2 for derivative gain.

The error plot of the response before and after tuning is shown in **Figure 20**.

Although the responses tuned using basic hill climbing were very close to the desired reference response, they always lagged behind the proposed modified hill climbing algorithm responses in almost every performance index.

7.4 Comparison among the TRMS cases

In all the cases presented for the TRMS, which is a highly nonlinear, strongly inherited coupling system, the tuning algorithm proved to be highly successful, regardless of the desired reference response performance in each case. The tuning always brought the process response to act in a similar fashion to that of the desired reference response. Therefore, the process controller performance expectations should be relevant to the model controller performance on the one hand and to the applicability of that model controller performance to the process controller on the other hand. That can be clearly evident when checking the comparative results in **Table 1**.

Generally, the performance values have shown improvement on that of the initial response, except for the LQR case, where some performances were lower due to the low quality of the desired reference response (settling time for pitch and rising time for yaw). This, once again, confirms the importance of the desired reference response preparation to achieve the best possible results.

By examining **Table 1**, it can be seen that forcing the system to achieve a certain milestone in one area may lead to degradation in quality in other areas. For example, when using a fuzzy controller as a model controller, the desired reference response was pushed to the extreme to fully eliminate overshoot. Although overshoot in the model responses was eliminated, the results were different for the actual process. For the pitch angle, the overshoot was decreased by 3.2 degrees when using a fuzzy controller as a model controller, compared to using PID controller as model controller. However, both rising time and settling time were increased by 0.77 and 1.13 s, respectively. Here, it is up to the system designer to make a decision on which path to take, according to the operation preferences. For the yaw angle, however, the outcome is different. As the yaw angle in the actual process is naturally highly under-damped, trying to force it to achieve non-overshooting response brought negative effects. All performance indexes were

19

System		Model controller type	Final process PID gains			Model controller performance			Process controller performance		
			P	I	D	Rise time	Settling time	Overshoot (%)	Rise time	Settling time	Overshoot (%)
Pitch	Before tuning	N/A	1	1	1	N/A	N/A	N/A	1.25	4.5	33
	After tuning	PID (max. error = 0.2)	1.9	1.2	0.2	0.3	0.71	5.1	0.33	1.47	8.2
		Fuzzy (max. error = 0.5)	2.1	0.4	0.7	3	1.1	0	1.1	2.6	5
		LQR (max. error = 0.4)	1.5	0.6	0.9	1.3	4.7	10	1.25	5.35	15
Yaw	Before tuning	N/A	1	1	1	N/A	N/A	N/A	1.26	6.75	31.6
	After tuning	PID (max. error = 0.3)	1.7	1.2	0.4	0.48	0.35	4.6	0.54	2.43	13
		Fuzzy (max. error = 0.5)	1.7	0.7	0.9	5	2	0	1.15	4.85	13.4
		LQR (max. error = 0.2)	1.4	0.7	1.2	1.5	6.3	16.5	1.5	6	20

Table 1.
Comparative results of TRMS cases.

System	Model controller type	Improvement in	
		Overshoot (%)	Settling time (s)
Pitch	PID	9.6	1.34
	Fuzzy	10.5	2.15
	LQR	5.8	−0.86
Yaw	PID	12.3	1.02
	Fuzzy	9	−0.05
	LQR	5.6	−0.38

Table 2.
Performance improvement when using the proposed design in comparison to the basic hill climbing with TRMS cases.

reduced in quality in comparison to using a PID controller as model controller, even the overshoot itself. Rising time and settling time were increased by 0.92 and 3.88 s, respectively, while overshoot was increased by 0.4%. This sheds some light on the priority of not straining the process with unachievable performances.

In order to compare the proposed algorithm performance with that of basic hill climbing, the following **Table 2** is created.

Some of the settling time performances are slightly slower than basic hill climbing (<1 s), but that was a result to reducing the overshoot amount significantly. This is because the proposed modified hill climbing algorithm followed the reference response better than the basic hill climbing.

8. Conclusions

In this chapter, a paradigm for real-time tuning of control systems is proposed. The proposed paradigm facilitates the application of a tuning algorithm based on simple error cost function calculations. Additionally, a modified hill climbing algorithm is developed. The control paradigm allows the system controller to be tuned iteratively to enhance the control performance using the suggested tuning algorithm. The algorithm does not require complex calculations to move in the searching space looking for suitable candidates. The algorithm always promotes the best solution to be used until a better one is found, which makes it suitable for real-time tuning problem. Simulation results clearly showed that the system response is following the desired reference response after tuning. However, the reference response should always be physically viable within an acceptable error margins.

Results have shown that the algorithm provides a very useful tool for control systems' engineers and specialists. The proposed system, overall, was proved to be quite successful in driving controlled plant response by using an inaccurate model of that plant. Therefore, the proposed design is suitable for tuning controllers based on inaccurate models or for calibrating controllers of systems with parameters deviation automatically.

This research introduced a very effective methodology to tune or calibrate control systems in real time, without the need to intervene with the system's operation. The proposed methodology is very flexible and can be used to achieve various performance targets. It was implemented with control systems specifically in mind; hence, the depth of achievement possibilities is high.

Acknowledgements

Appendix A

List of parameters that were used in expressing example models with their values (if available) is presented in this appendix (**Table A**).

Parameter	Brief explanation	Value
θ	Pitch angle	
ψ	Yaw angle	
F_p	Pitch thrust force	
F_y	Yaw thrust force	
B_p	Equivalent viscous damping about pitch axis	0.8 N/V
J_{eq_p}	Total moment of inertia about pitch axis	0.0384 kg m^2
m_{heli}	Total moving mass of the helicopter	1.3872 kg
lcm	Centre of mass length along helicopter body from pitch axis	0.186 m
B_y	Equivalent viscous damping about yaw axis	0.318 N/V
J_{eq_y}	Total moment of inertia about yaw axis	0.0432 kg m^2
K_{pp}	Thrust force constant of yaw motor/propeller	0.204 N m/V
K_{py}	Thrust toque constant acting on pitch axis from yaw motor/propeller	0.0068 N m/V
K_{yp}	Thrust toque constant acting on yaw axis from pitch motor/propeller	0.0219 N m/V
K_{yy}	Thrust toque constant acting on yaw axis from yaw motor/propeller	0.072 N m/V
α	Angle of attack	
q	Pitch rate	
δ	Elevator deflection angle	
μ	$\frac{\rho S \bar{c}}{4m}$	
ρ	Density of air	
S	Platform area of the wing	
$\bar{c}$	Average cord length	
m	Mass of the aircraft	
Ω	$\frac{2U}{\bar{c}}$	
U	Equilibrium flight speed	
C_D	Coefficient of drag	
C_L	Coefficient of lift	
C_w	Coefficient of weight	
C_M	Coefficient of pitch moment	
γ	Flight path angle	
σ	$\frac{1}{1+\mu C_L}$	

Parameter	Brief explanation	Value
i_{yy}	Normalized moment of inertia	
η	$\mu\sigma C_M$	
$V(t)$	Motor input voltage	
$e(t)$	Back electromotive force	
L_a	Motor armature electric inductance	0.15 H
$i(t)$	Electric current	
R_a	Motor armature electric resistance	0.2 Ohm
$T_m(t)$	Mechanical torque	
K_m	Motor torque constant	9.14×10^{-5} N m/A
K_b	Electromotive force constant	0.055 V/rad/sec
$w(t)$	Angular velocity of motor shaft	
$T_L(t)$	Load torque	
J_m	Moment of inertia of the rotor	1.36×10^{-5} kg m^2
B_m	Motor shaft viscous friction coefficient	0.5×10^{-5} N m sec

Table A.
List of parameters used in example models.

Author details

Ahmed Abdulelah Ahmed[1,2], Azura Che Soh[1]*, Mohd Khair Hassan[1],
Samsul Bahari Mohd Noor[1] and Hafiz Rashidi Harun[1]

1 Control System and Signal Processing (CSSP) Research Centre, Department of
Electrical and Electronic Engineering, Universiti Putra Malaysia, Selangor, Malaysia

2 Faculty of Engineering, University of Kufa, Iraq Kufa, Najaf Governarate, Iraq

*Address all correspondence to: azuracs@upm.edu.my

IntechOpen

References

[1] Jantzen J. Foundations of Fuzzy Control. New Jersey, USA: John Wiley & Sons; 2007

[2] Thomas BS. Telerobotics, Automation, and Human Supervisory Control. 1st ed. Cambridge, UK: MIT Press; 1992

[3] Sands TA. Physics-based control methods. In: Ghadawala R, editor. Advancements in Spacecraft Systems and Orbit Determination. Rijeka: In-Tech Publishers; 2012. pp. 29-54

[4] Sands T. Space system identification algorithms. Journal of Space Exploration. 2017;**6**:138

[5] Sands T. Nonlinear-adaptive mathematical system identification. Computational Engineering, Special Issue of Computation. 2017;**5**(4):47

[6] Ogata K. Modern Control Engineering. 5th ed. New Jersey, USA: Prentice Hall; 2010

[7] Reedy J, Lunzman S, Mekari B. Model-based design accelerates development of mechanical locomotive controls. Off-Highway Engineering. 2011;**19**(2):18-22

[8] Ogata K. System Dynamics. 4th ed. New Jersey, USA: Prentice Hall; 2015

[9] Miller TW. Modeling Techniques in Predictive Analytics. 1st ed. New Jersey, USA: Pearson Education, FT Press; 2013

[10] Burns R. Advanced Control Engineering. 1st ed. Oxford, UK: Butterworth-Heinemann; 2001

[11] Sands T. The catastrophe of electric vehicle sales. Mathematics. 2017;**5**:46

[12] Astrom KJ, Wittenmark B. Adaptive Control. 2nd ed. New York, USA: Dover Publications; 2008

[13] Ioannou PA, Sun J. Robust Adaptive Control. 1st ed. New York, USA: Courier Dover Publications; 2012

[14] Wittenmark B. Adaptive Dual Control. In: Control Systems, Robotics and Automation. Encyclopedia of Life Support Systems (EOLSS), Developed under the auspices of the UNESCO; 2002

[15] Kartik BA, Krstic M. Real-Time Optimization by Extremum-Seeking Control. 1st ed. New Jersey, USA: Wiley-Interscience; 2003

[16] Bunin GA, François G, Bonvin D. A real-time optimization framework for the iterative controller tuning problem. PRO. 2013;**1**(2):203-237

[17] Mosca E. Optimal, Predictive and Adaptive Control. 1st ed. New Jersey, USA: Prentice Hall; 1994

[18] Bonvin D, Srinivasan B. On the use of models for real-time optimization. In: Chemical Process Control-VIII, No. EPFL-CONF-181259; 2012

[19] Tan KK, Zhao S, Xu J-Z. Online automatic tuning of a proportional integral derivative controller based on an iterative learning control approach. IET Control Theory and Applications. 2007;**1**(1):90-96

[20] Kuo BC, Golnaraghi F. Automatic Control Systems. 9th ed. New Jersey, USA: John Wiley & Sons; 2009

[21] Russell SJ, Norvig P. Artificial Intelligence: A Modern Approach. 3rd ed. New Jersey, USA: Prentice Hall; 2010

[22] Katalin MH, Lakner R, Gerzson M. Intelligent Control Systems: An Introduction with Examples. Dordrecht, Netherlands: Kluwer Academic Publisher; 2004

[23] Charbonneau P. An Introduction to Genetic Algorithms for Numerical Optimization. Boulder, Colorado: NCAR Technical Note, Institute of Global Environment and Society (IGES); 2002

[24] Naysmith MR, Douglas PL. Review of real time optimization in the chemical process industries. Developments in Chemical Engineering and Mineral Processing. 1995;**3**(2):67-87

[25] Quanser. 2-DOF Helicopter. Reference Manual: Quanser, revision 2.11