Auto-Tuning for High Performance Autopilot Design

J. M. Counsell[1], O. S. Zaher[2], J. Brindley[3]

University of Strathclyde, Scotland, UK

ABSTRACT

A novel auto-tuning method for the RIDE controller algorithm is presented. The RIDE controller is applied to a high performance aircraft model. The tuner utilises a constrained genetic algorithm to automate the tuning process. The results of the tuner are compared with that of another tuning method which utilises unconstrained optimisation so as to highlight the efficacy of constrained optimisation for this application. It is shown from the results that the constrained genetic algorithm optimisation scheme offers a highly effective tuning solution which can be used to attain safe and high performance control with the RIDE control algorithm.

NOMENCLATURE

A	State matrix
B	Control matrix
C	Output matrix
$\mathbf{G_A(s)}$	Actuator transfer function
$\mathbf{K_I}$	Integral gain
$\mathbf{K_p}$	Proportional gain
$\mathbf{\Omega_n}$	Diagonal matrix of natural frequencies
$\vec{u}$	Input vector
$\vec{y}$	Output vector
$\mathbf{y_A}$	Actual model response
$\mathbf{y_I}$	Ideal model response
$\mathbf{Z_d}$	Diagonal matrix of damping ratios

1. INTRODUCTION

In all control systems, it is vitally important to tune the controller parameters in order to obtain the optimum controller performance. Incorrect selection of controller parameters will not only result in poor controller performance, it can also be the source of system instability. Thus, careful selection of these parameters must be made in order to avoid this problem. The process of tuning can be done in an iterative fashion i.e. changing the values until the desired response is achieved, however, this process can be extremely time consuming. Consequently, there are some proven tuning methods which have been developed in order to aid in this process.

One such method which is commonly used in the design of PID controllers is the Zeigler-Nichols Ultimate Cycle method [1]. This method has proven to be reliable, and so is widely used for determining controller gains. However, it is not applicable to all controllers, particularly nonlinear MIMO controllers. For the controllers to which it is applicable the tuning process can often still be rather tedious, particularly when lengthy simulations are involved. A further drawback of the Zeigler Nichols tuning method is that, although it generally produces tenable results, the results are not always optimal. This is due to the fact that it was originally designed to produce a "quarter wave decay" response, that is, each peak in the oscillatory response is a quarter magnitude of its predecessor. In view of these problems, the benefit of automating the tuning process is apparent.

This paper aims to elucidate the development of a novel method which can be used to tune controller parameters automatically. One of the main benefits of this tuning method, besides eliminating the need for lengthy iterative tuning processes, is that it can be applied to any controller setup, be it SISO or MIMO, to tune any parameter. The performance of the algorithm, which utilises constrained optimisation with the genetic algorithm, is compared to that of an unconstrained optimisation algorithm accentuating the advantages of constrained optimisation for this application. The performance of both auto-tuners is demonstrated and analysed through application to a high performance autopilot system using the RIDE control algorithm [2] to control an F-18/HARV fighter aircraft model.

[1] Professor (BRE Centre), Mechanical Engineering
[2] PhD Student, Mechanical Engineering
[3] PhD Student, Mechanical Engineering

2. AIRCRAFT MODEL

The F-18/HARV - a modified version of the F/A-18 fighter aircraft - is the aircraft model used in the testing of the auto-tuning algorithm. The computational model used in this paper was developed in [3] which presents the model description in greater detail than shown here. A linearised model of the aircraft operating at a speed and altitude of Mach 0.4 and 6000ft respectively was used. The model is controlled using only thrust vectoring commands since the dynamic pressure at the operating point is relatively low (approx. 180 psf). The linear model used is expressed in state-space as shown in equation 7.

$$\dot{x}(t) = Ax(t) + Bu(t)$$

$$y(t) = Cx(t) \qquad \ldots(1)$$

Where the state vector x(t) = $[\alpha\ q\ \beta\ p\ r]^T$ and the input vector u(t) = $[\delta_{PTV}\ \ \delta_{RTV}\ \delta_{YTV}]^T$. Only p, q and r, the state variables describing the roll, pitch and yaw rates respectively, are controlled in this model. The actuator inputs δ_{PTV}, δ_{RTV} and δ_{YTV} are the pitch, roll and yaw thrust vectoring positions respectively. The numerical state space model is given in *Appendix A.* A diagram of the general F-18 aircraft is shown in Fig.1.

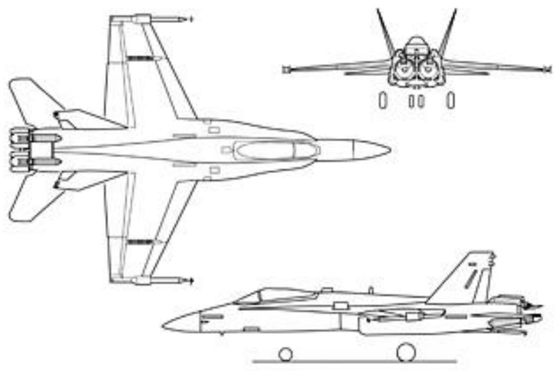

Fig.1: F-18 fighter aircraft

2.1 Actuator Dynamics

When modelling any controlled system, in order to capture reality as accurately as possible, it is essential to include models of the systems actuators and their associated nonlinear characteristics. All actuators have specific physical characteristics which define their performance and capability thus, in order to obtain a realistic model response, it is necessary that these characteristics are preserved in the model. Detailed models of the actuators present in the F-18 model can be reduced and represented simply as spring-mass damper systems with second order transfer functions [4]. The transfer function used to represent the thrust vectoring nozzles is shown below in equation 2.

$$\frac{1}{\left(\frac{s}{20}\right)^2 + \frac{2(0.6)}{20}s + 1} \quad \ldots(2)$$

The actuator discontinuity that is included in this model is the deflection limits. For the thrust vectoring nozzles this is ±30 deg.

3. RIDE CONTROLLER

As previously mentioned, the RIDE control algorithm was used to control the aircraft model and thus it is the controller on which the auto-tuner was tested. This section provides a brief overview of the RIDE controller set up used so as to clarify which parameters require tuning. A more detailed explanation of the RIDE algorithm can be found in [2]. The RIDE control algorithm is given by the equation below:

$$\vec{u}(t) = r - K_P\vec{y}(t) + \hat{\vec{u}}_{eq}(t) \quad \ldots(3)$$

Where

$$\dot{r} = K_I e(t) \ \ldots(4)$$

The equivalent control term (u_{eq}) in (3) uses inverse dynamics to determine the actuator inputs that are required to ensure zero rate of change of the outputs and thus works effectively to diminish any disturbances and reduce coupling. The equivalent control term is determined by the following equation:

$$\hat{\vec{u}}_{eq}(t) = -[CB]^{-1}\dot{\vec{y}}(t) + \vec{u}(t) \ \ \ldots(5)$$

The RIDE controller is shown in diagrammatic form in Fig.2[5].

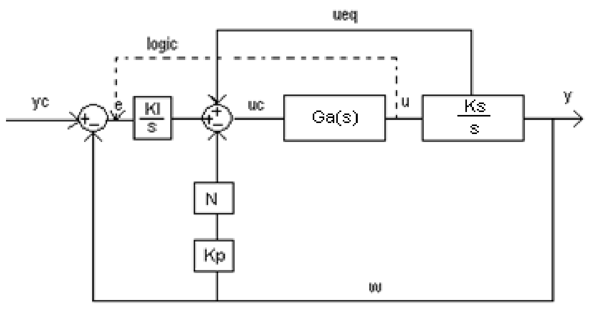

Fig.2: RIDE controller setup

The transfer function of the above system is given by

$$G_{(s)} = \frac{K_I K_S G_{A(s)}}{s^2 + (K_p K_S G_{A(s)})s + K_I K_S G_{A(s)}} \qquad \ldots(6)$$

$G_A(s)$ can be shown to be approximately equal to 1 if the bandwidth of the inner loop is slow in comparison to the outer loop bandwidth. If (6) is compared to a second order transfer function, K_P and K_I can be selected such that

$$K_p = [K_s]^{-1} 2 Z_d \Omega_n \quad \ldots(7)$$

$$K_I = [K_s]^{-1} \Omega_n^2 \quad \ldots(8)$$

Where $K_s = [CB]$. This way, the controller gains can be tuned by altering the natural frequency and damping ratios of the controller. When tuning the controller, it is important to note that only small excitations should be used when observing the system response so that the actuator discontinuities are not encountered. The RIDE algorithm is used with the Optimal and Safe Control Algorithm (OSCA) which deals with actuator nonlinearities [6]. This way the gains which are obtained with small excitations to the system remain suitable for larger excitations which cause discontinuities to be encountered.

4. AUTO-TUNING

Both auto-tuning algorithms mentioned previously were implemented using MATLAB however the algorithmic concepts can be extended to any coding platform. The algorithms work by comparing the actual model response to that of an ideal reference model which is represented by an arbitrary second order transfer function. The reference model is connected in parallel to the actual control system as shown in *Fig.3.*

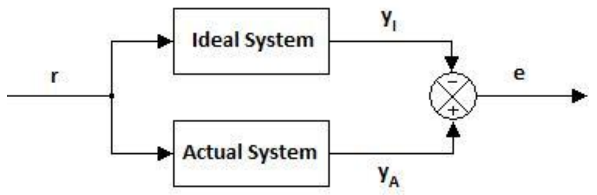

Fig.3: Actual to ideal system comparison

Based on the requested output (r), the error signal (e) is calculated from the difference between both output signals ($y_A - y_I$) in order to determine the deviation of the actual output from the ideal output. The requested output can be altered by the user giving the user the freedom to tune the controller for a specific desired output e.g. a step or pulse excitation. The second order transfer function representing the ideal system can also be designed with specific attributes which suit the user's needs e.g. response time can be altered by varying the natural frequency and damping of the transfer function. In the case of tuning a MIMO control system, the calculation of the error signal requires that all of the system outputs are measured to allow for a cumulative error signal to be determined. It is clear that less deviation of the actual output signal from the ideal output signal i.e. smaller error signal means better controller performance. Therefore the controller can be said to have achieved its optimum performance when the error signal is at its minimum. Hence, in order to achieve the optimum tuning for the controller, the controller parameters that minimise the error signal must be determined.

4.1 Optimisation

In order to achieve the aforementioned, the entire control system is treated as a nonlinear function whose inputs are the parameters to be tuned and the output is the error signal. The optimisation of this function can be performed in two different manners, namely constrained and unconstrained nonlinear optimisation. Both of these types of optimisation were applied and tested in this paper so as to highlight the advantages and disadvantages of each method, and more importantly, to demonstrate the superiority of constrained optimisation for this application.

4.1.1 Unconstrained Optimisation

For the unconstrained auto-tuner, the Nelder-Mead simplex minimisation algorithm implemented in MATLAB (fminsearch function) was used. This algorithm requires user defined initial estimates of the parameters to be tuned as a starting point. It then minimises the objective function, that is, the control system with the error signal as the output by altering the tuning parameters – starting from the user defined initial estimates - until the parameters which correspond to a local minimum are found. It is important to bear in mind that the algorithm is only capable of finding local minima as opposed to the global minimum. The likelihood of the local minimum found corresponding to the global minimum is unknown and is highly dependant on the user defined initial estimates. Hence choosing suitable initial estimates proves problematic in that suboptimal results can be produced based on the initial specification of the parameters [7]. Another drawback of unconstrained optimisation for this application is that system stability is not guaranteed with the results produced, as is demonstrated in the results section. The auto-tuner does however allow up to a maximum of five parameters to be tuned simultaneously which is particularly advantageous when the user desires to tune many parameters e.g. controller specific parameters other than the conventional proportional, integral and derivative gains.

4.1.2 Constrained Optimisation

It is clear from the problems associated with unconstrained optimisation mentioned a priori that an optimisation algorithm that is able to determine global minima is desirable. The Genetic Algorithm (GA), well known for its global optimisation capability [8], therefore was used for the constrained Auto-tuning algorithm. It is an evolutionary algorithm which uses techniques inspired by evolutionary biology (natural selection). The genetic algorithm, unlike classical optimisation algorithms, generates a population of points at each iteration which converge towards an optimal solution as opposed to a sequence of single points approaching an optimal solution. Due to the design of recombination that exists within the GA, the population generated moves away from local minima which traditional algorithms are likely to get caught in [9]. Fig.4 illustrates the concept of local and global minima with a nonlinear function which has more than one minimum.

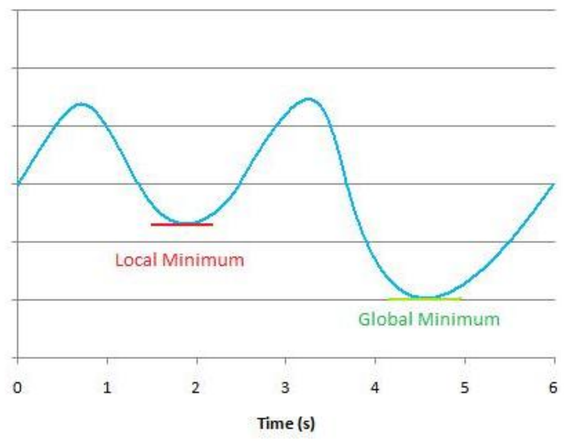

Fig.4: Local & global minimum

In order to understand how the GA can be configured for the auto-tuning algorithm, it is imperative that the reader is familiarised with the concepts behind the GA.

Initially the algorithm creates a random initial population based on the initial range which can be specified by the user. It is important that the initial range be selected carefully as the population diversity has a large bearing on how well the GA performs. The optimal initial range is usually found by trial and error. It is known however that if the initial range is specified close to the optimal solution or if the optimal solution lies within the initial range, the GA's performance will be greatly improved and the global minimum is more likely to be found. Once the initial range has been specified, a random initial population is created within this range. The next population is then created based on the members in the current population. It consists of members who are taken directly from the current population, members which have been modified using mutations and others can be from crossovers - combined vector entries from a pair of members. The members which are taken directly from the current population are selected based on their fitness – the members which correspond to the lowest outputs from the nonlinear function being optimised have higher fitness. The number of members which are taken directly from the current population, known as the elite count, can be specified by the user. The number of new members which are created using crossovers can also be defined by the user through the crossover fraction. Based on the elite count and the crossover fraction, the number of new members created from mutations is determined. To illustrate this, consider a population size of 15 with the elite count and crossover fraction set as 2 and 0.7 respectively. This would mean that 13 members other than elite members remain from the population. The number of crossover members is then determined by 0.7 x 13 = 9.1 which would be rounded to 9. This would leave four members to be created from mutations. In order to control the amount of mutation applied to members, the scale and shrink options can be defined. The scale option allows the standard deviation of the mutation at the initial population to be controlled. The shrink option controls the amount by which the mutation decreases with each generation [10]. The algorithm continues to generate new populations until one of the stopping criteria is met.

4.1.2.1 Establishing Constraints

In order to ensure system stability and increase the likelihood of a global minimum being found, it is necessary to establish upper and lower bounds within which the system is stable and the optimum gain is known to lie. In many cases however, the user may not know where the optimum gain lies, in which case the gains that correspond to the boundaries of the stable region for the system can be used as reasonable constraints so as to ensure that the auto-tuner maintains the system within stable limits. As was stated in previous section, the parameters which are being tuned are the system natural frequency and damping ratio. Therefore it is necessary to establish suitable constraints for both of these parameters. Since the closed loop system with the RIDE controller can be considered to be second order, a simple classical root locus analysis of a second order system can be related to the actual system in order to determine the constraints for the damping ratio. A root locus plot for a second order system can be seen in *Fig.5* where the damping ratio is varied from 0.8 to 0.1.

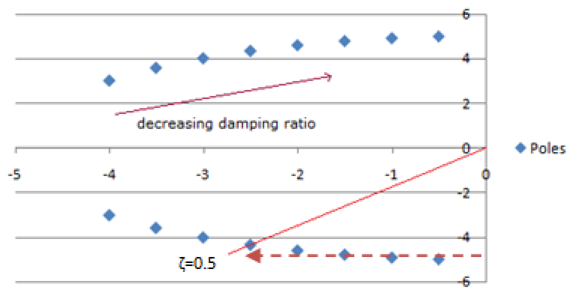

Fig.5: Root Locus with varying damping ratio

A minimum damping ratio of **0.5** was chosen as the lower constraint so as to leave a reasonable margin for stability and to ensure that the level of overshoot does not become too high (*Fig.6*). The upper constraint was chosen to be **1** so that the response is not over damped or unnecessarily slow.

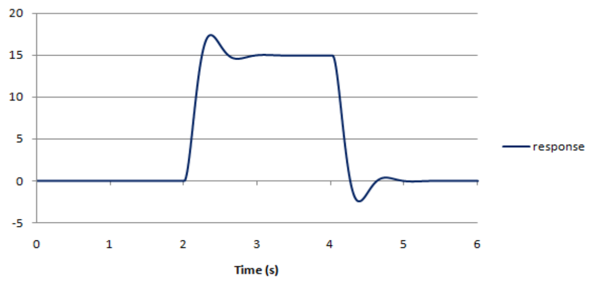

Fig.6: Second order response with ζ=0.5

When finding constraints for the natural frequency, the lower constraint could be selected as any value above zero which is known to be stable. A value of **3 rad/s** was chosen. The upper limit for the natural frequency, ω_{crit}, can be difficult to determine when the system is nonlinear as classical stability analysis cannot be applied. Thus, an alternative method of finding these boundaries is required. To this effect, a program which automatically establishes the upper constraint by rapidly searching through the gains was developed. This program, for a fixed damping ratio, returns the upper boundary of the stable region accurate to the number of decimal places desired by the user. The main benefit of this program is that it can be used for nonlinear and linear systems.

The program works by searching through the gains starting at a user defined value. This should be a value which the user knows exists within the stable operating region for the system. If this is not known, an initial value of zero can always be used. The program then increases the gain by one unit with each iteration and analyses the corresponding system response. It is important to bear in mind that, as with tuning the controller, small excitations are used so as not to engage the actuator discontinuities. This process is repeated until the system response exhibits unstable characteristics i.e. the response is oscillatory with each successive peak increasing in magnitude (*Fig.7*). This is expressed mathematically in equation 8

$$X(n) > X(n\text{-}1) \quad \ldots(8)$$

Where X is the peak amplitude, $n = 0,t,1t,2t,3t\ldots$ where n is the discrete time step and t is the sampling period.

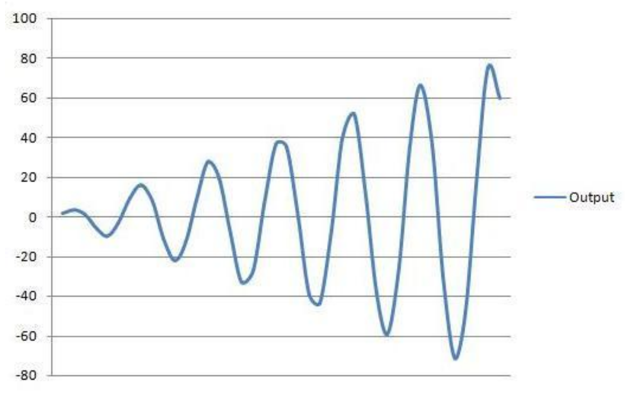

Fig.7: Unstable response

When the above condition (equation 8**)** becomes true, the program returns to the last gain value at which the system was stable i.e. the value at the preceding iteration, and repeats the iterative search but increasing the gain by one tenth of the initial unit size used hence increasing the accuracy by one decimal place. The above process is repeated until a value of gain accurate to four decimal places which corresponds to the upper limit of the stable operating region is returned. This was performed with the damping ratio fixed at the upper and lower damping ratio constraints i.e. 1 and 0.5 respectively. The values of ω_{crit} corresponding to the aforementioned damping ratios were **9.413 rad/s** and **16.86 rad/s** respectively. Hence the lower value of ω_{crit} (9.413 rad/s) was used as the upper constraint for the natural frequency.

4.2 MIMO optimisation

For a MIMO system, it is necessary to adapt the algorithm in order to account for the errors occurring in multiple signals. In this case, a summation of the error signals for all of the outputs is used to produce a cumulative error signal. The controller set up, however, remains the same as for the SISO system as shown in *Fig.3*. The difference lies in the output signals y_A and y_I which become vector signals with a number of elements equal to the number of outputs of the system. Consequently, the error signal also becomes a vector signal calculated by

$$e = (y_I(1) - y_A((1)) + (y_I(2) - y_A(2)) + (y_I(k) - y_A(k)) \quad \ldots(9)$$

Where k is the number of outputs of the system. Thus, for a system with three outputs, the error signal would consist of the addition of three separate error signals. In order to obtain the true magnitude of the error however, it is necessary that absolute values of the error signals are taken so as to prevent any cancellation of error. Consider a MIMO system with three outputs producing the three error signals A, B and C shown in *Fig.8*. As both errors A an B are positive and C is negative, the magnitude of the cumulative error would be erroneous if absolute values of the errors were not taken since the overall error would be reduced. Thus, absolute values are required to produce correct results as shown in *Fig.8*. Hence, the error function now becomes:

$$e = |y_I(1) - y_A((1)| + |y_I(2) - y_A(2)| + |y_I(k) - y_A(k)| \quad \ldots(10)$$

The same optimisation procedure as described above for a SISO system is subsequently performed on the cumulative error signal.

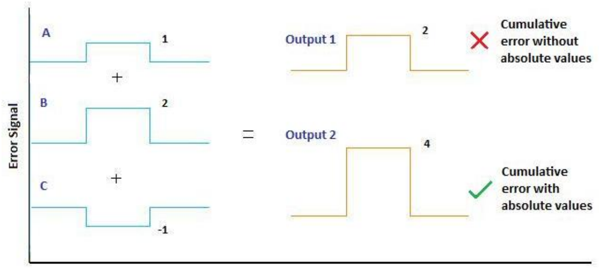
Error Signal
A
1
+
B
2
=
+
C
-1
Output 1
2
Cumulative error without absolute values
Output 2
4
Cumulative error with absolute values

Fig. 8: Cumulative error signal

5. RESULTS

As was shown in Section 3, the controller can be tuned by altering the values of the damping ratio and natural frequency matrices. For the results presented in this paper, the damping ratios and the natural frequencies for each channel were kept the same i.e. the matrices were replaced with scalar values in order to keep the values the same across all three channels. Initially, however, the controller was tuned using the Ziegler Nichols tuning method so as to illustrate the problems which can be encountered when using manual tuning for nonlinear MIMO systems.

5.1 Zeigler Nichols Tuning

In order for the Zeigler Nichols tuning method to work, the gains were required to be tuned differently. The gain equations (6) and (7) can be represented as:

$$K_p = [K_s]^{-1}p \quad \ldots(11)$$

$$K_I = [K_s]^{-1}g \quad \ldots(12)$$

Where $p = 2Z_d\Omega_n$ and $g = \Omega_n^2$. The gains could then be tuned by altering p and g as opposed to the controller natural frequency and damping ratio. As this tuning method requires that the integral action on the controller to initially be zero, g was set to 1. The matrix $[K_s]^{-1}$ in both of the gain equations must remain however, since it is part of the RIDE theory. Through increasing the value of p, the value which corresponded to the critical proportional gain (K_{pc} in Table.1) was found to be 26.448 and the corresponding period of oscillation of the response signal (T_c) was 0.08s. The formulae in Table.1 were subsequently used to determine the values of g and p.

Control	p	g
P	0.5 K_{PC}	
PI	0.45 K_{PC}	1.2 K_P/T_C

Table.1: Zeigler Nichols gain formulae

The values of g and p were found to be 178.524 and 11.902 respectively. These values, when used in the model, yielded a highly undesirable system response as shown in *Fig.9*

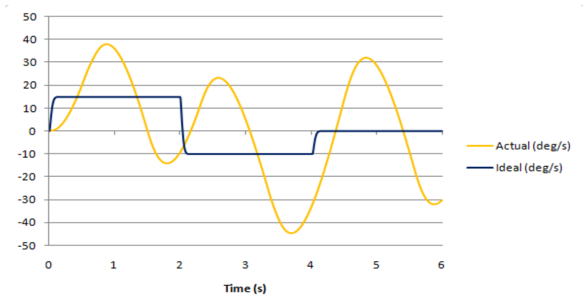

Fig.9: Response after Zeigler Nichols Tuning

It can be seen that after the tedious manual tuning process, the resulting gains produce an undesirable system response. The following results from the auto-tuner demonstrate the benefits that are achieved from auto-tuning.

5.2 Auto-tuning

Both auto-tuning algorithms were tested with a pulse roll rate request of 2 deg/s so as not to excite the actuator deflection limit. The initial values for natural frequency and damping ratio for the unconstrained optimiser were set to **3 rad/s** and **0.5** respectively. The upper and lower constraints for the constrained optimiser can be seen in Table.2.

	ζ	**ω (rad/s)**
Lower Constraint	0.5	3
Upper Constraint	1	9.413

Table.2: Constraints

Fig.10 and *Fig.11* show the performance of the controller for the same roll rate command after constrained and unconstrained tuning respectively.

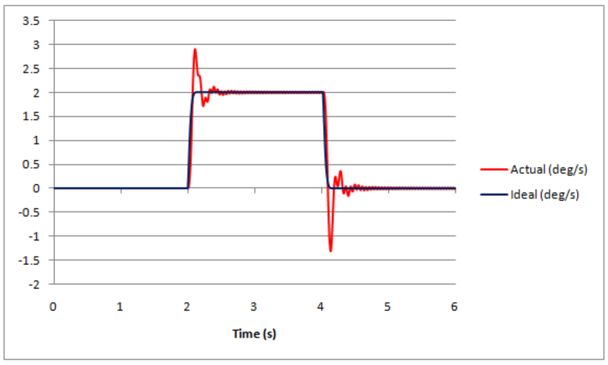

Fig.10: Response after unconstrained optimisation

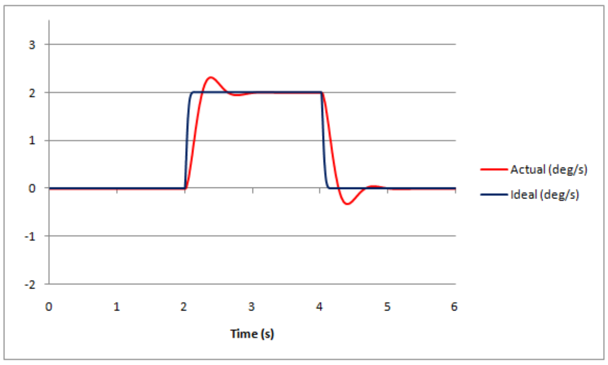

Fig.11: Response after constrained optimisation

The values returned for the natural frequency and damping ratio for both tuning algorithms are shown in Table.3.

	ζ	**ω (rad/s)**
Unconstrained	0.3591	24.4305
Constrained	0.505	9.41

Table.3: Tuned values of ζ and ω

From *Fig.10* and *Fig.11* it can be seen that the response after unconstrained tuning is faster than after constrained tuning. However, the response in *Fig.*10 also shows a much higher degree of overshoot than that in *Fig.*11. When a higher roll rate was requested, 10 deg/s, from the model so that the actuator deflection limit would be reached, the system became completely unstable when the gains obtained from unconstrained tuning were used (*Fig.12*). This shows that the system was being pushed too hard and that safe control cannot be guaranteed when using the unconstrained tuning algorithm. The system response when using the gains obtained by constrained tuning however can be seen to be stable and to essentially exhibit the same response characteristics as when the deflection limit was not being reached (*Fig.13*). This demonstrates that the constrained auto-tuning process presented in this paper offers an effective safe tuning solution whilst extracting the optimum performance from the control system.

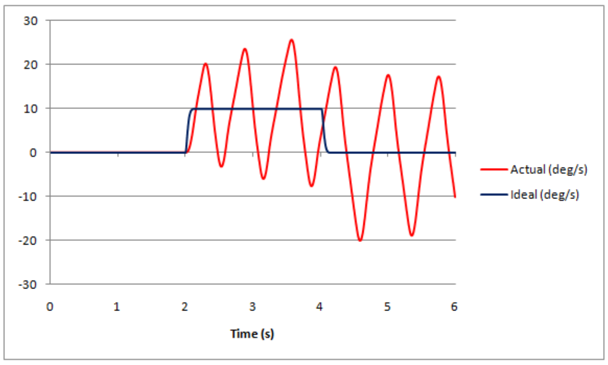

Fig.12: Response for high roll rate request (unconstrained tuning)

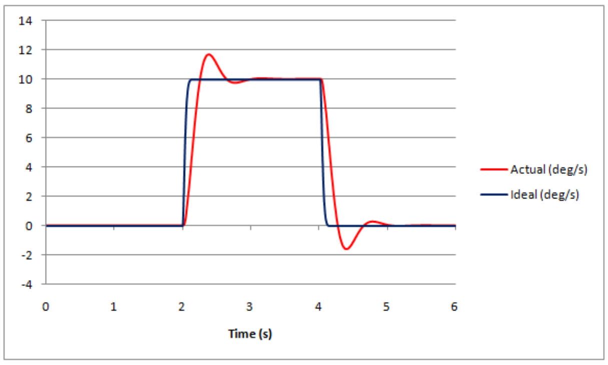

Fig.13: Response for high roll rate request (constrained tuning)

6. CONCLUSION

A novel tuning method for the RIDE controller algorithm has been presented with application to a high performance F-18 aircraft model. A comparison was made between an unconstrained and a constrained tuning algorithm when used to tune the controller's natural frequency. The results show that the constrained algorithm is more effective for tuning as it is able to achieve safe and optimal controller performance. This is due to its ability to locate the global optimum within the stable operating region as opposed to the unconstrained algorithm which cannot guarantee safe control. Future work will include optimising the damping ratio and natural frequency of the controller across all three channels separately so as to investigate the improvement in performance which can be achieved through driving different channels harder. A nonlinear constraint for the u_{eq} term [5] which is required to ensure stability when actuator limits are reached will also be implemented.

REFERENCES

[1] Bolton W., "Control Engineering", *Published by Prentice Hal,* May 1998

[2] Bradshaw A., Counsell J.M, "Design of Autopilots for High Performance Missiles", *Journal of Systems and Control Engineering,"* 1992 pp.75-84

[3] Haiges K.R et al, "Robust Control Law Development for Modern Aerospace Vehicles", WL-TR-91-3105, Aug. 1991.

[4] Adams R.J., Buffington J.M., Sparks A.G., Banda S.S., "Robust Multivariate Flight Control Advances in Industrial Control", *Published by Springer-Verlag London Limited,* 1994 Printed in Great Britain.

[5] Counsell J.M., Brindley J., Macdonald M., "Non-Linear Autopilot Design Using the Philosophy of Variable Transient Response", *Published in AIAA Guidance, Navigation and Control Conference,* August 2009

[6] Counsell J.M., "Optimum and Safe Control Algorithm for Modern Missile Autopilot Design", PhD thesis, *Lancaster University,* 1992

[7] MATLAB Documentation, "Optimisation Toolbox^{TM} User's Guide", *Copyright 1990 - 2009 by The MathWorks, Inc.*

[8] Fielding C., Varga A., Bennani S., Selier M., "Advanced Techniques for Clearance of Flight Control Laws", *Published by Springer-Verlag,* 2002 Printed in Germany

[9] Man K. F., Tang K. S., Kwong S., "Genetic Algorithms: Concepts and Designs", *Published by Springer-Verlag London Limited,* 1999 Printed in Great Britain

[10] MATLAB Documentation, "Genetic Algorithm and Direct Search Toolbox^{TM} User's Guide", *Copyright 2004 - 2009 by The MathWorks, Inc*

APPENDIX A – Numerical State Space Aircraft Model

$$A=\begin{bmatrix} -0.8018 & 0.9847 & 0 & 0 & 0 \\ -1.5210 & -0.5944 & 0 & 0 & 0 \\ 0 & 0 & -0.1702 & 0.0896 & -0.9935 \\ 0 & 0 & -10.2840 & -1.9481 & 0.6339 \\ 0 & 0 & 2.3506 & -0.0226 & -0.1649 \end{bmatrix}$$

$$B=\begin{bmatrix} -0.0278 & 0 & 0 \\ -1.7510 & 0 & 0 \\ 0 & 0 & 0.0027 \\ 0 & 0.0750 & 0.0104 \\ 0 & -0.0000 & -0.1342 \end{bmatrix}$$

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