ICAS-96-2.3 (R)

A NEW RECONFIGURATION CONCEPT FOR FLIGHT CONTROL SYSTEMS IN CASE OF ACTUATOR AND CONTROL SURFACE FAILURES

G. Baumgarten* and W. Heine**

1. Abstract

A new concept for the reconfiguration of a flight controller in case of control effector failures is proposed. Failure types of reduced control surface effectiveness, stuck surface, reduced actuator rate limit, and reduced actuator deflection limit are considered. A comprehensive process model forms the basis for the controller and the reconfiguration. It is divided up in an aircraft model describing the flight dynamics and a control effector model describing the dynamics of the actuator system and the control surface system. Every failure case can be described by changing the corresponding parameters in the control effector model.

The controller contains both the control effector model and the aircraft model. The control system has to be adapted in order to have intact actuators take over the lost functionality of the impaired ones. The inherent redundancy in most modern actuator systems is the main prerequisite for such a substitution. The qualitative and quantitative information about the fault is supposed to be available from a failure detection and isolation process, called FDI process in the following. The information is used to adapt the control system via predefined knowledge based means. During reconfiguration the model parameters are updated based on the failure information.

A genetic algorithm is applied to take care of those impacts of the fault that cannot be quantitatively gauged. Such an adaptation basically involves an optimization to alter the model parameters via a specific strategy within predefined limits in order to maximize control performance [1]. Linear and nonlinear failure types can be estimated by measuring the aircraft inputs and states. The conservation of the estimation results during missing excitation of the aircraft can be realized easily.

* Dasa, Militärflugzeuge, Postfach 801160, D-81663 München, Tel.:(0)89/607-29419
** DLR, Inst. für Flugmechanik, Lilienthalplatz 7, D-38108 Braunschweig, Tel.:(0)531/295-2664

2. Nomenclature

Matrices are written in capital bold letters, vectors in bold letters, and functions in italic letters.

Symbols:	*Description:*
Δ	difference
γ	flight path angle
Φ	bank angle
α	angle of attack
β	angle of sideslip
ρ	air density
ξ	aileron deflection
Θ	pitch angle
c	cost value
g	transfer function
k	gain
L_p	roll damping coefficient
L_ξ	aileron roll effectiveness
p, q, r	roll rate, pitch rate, yaw rate
T	time constant of first order lag
u	input variable
V_A	true air speed
x	state variable
y	output variable

Indices:	*Description:*
b	body fixed coordinates
C	command
FE	failure case
i, j, k	counting indices
M	model
NF	normal case, no failure
FB	feedback
TR	translator
FF	feedforward
R	controlled output variable
FBM	model feedback
CM	command for actuator model
RC	command for controlled output variable

3. Introduction

Overviewing existing reconfiguration concepts [2] mainly linear failure types are investigated. These are described by changing the input matrix or the dynamic matrix of a model state space representation. Often a changed control surface effectiveness or a stuck surface are taken into account without considering nonlinear failure types for example a reduced rate limit or a reduced deflection limit.

A second disadvantage of the existing concepts is the time for the calculation of new controller parameters after the fault has occured. Further the convergence of the calculation must be guaranteed. The risk of missing convergence is avoided by determining the parameter sets for each failure case in advance and storing them in the flight computer [3]. The disadvantage of this method is the great amount of storing space required for the vast number of possible failure cases.

Our reconfiguration approach tries to avoid the disadvantages listed above. There are three goals for the development:

1. The amount of controller parameters to be stored in advance has to be as small as possible.

2. The calculation time for the new controller parameters after a fault has occured should be as short as possible.

3. The variety of failure types that can be considered should be as large as possible. Nonlinear failure types such as a reduced rate limit should be included.

The first part of the paper introduces the controller structure and its reconfiguration meeting the objectives compiled above. The second part presents the application of the genetic algorithm optimizing the reconfiguration.

4. Control Effector Model

Besides the nonlinear aircraft model the control effector model in the state space representation provides the basis for the controller structure, the reconfiguration, and the genetic algorithm.

A single component of the control effector system is shown in Fig. 1. An actuator drives a control surface.

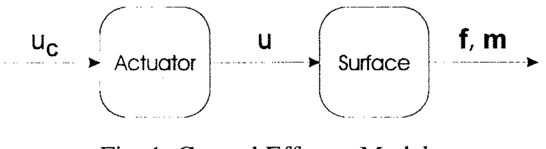

Fig. 1: Control Effector Model

u_C is the controller command for the surface deflection. The actuator model comprises a first order lag including a rate limit $\dot{u}_{max}$ and a travel limit u_{max}. u is the actual surface position.

$$\dot{u} = \frac{1}{T}(-u + k \cdot u_C) \quad |\dot{u}| \le \dot{u}_{max}, \quad |u| \le u_{max} \qquad (1)$$

The actuator failure types taken into account are a reduced rate limit, a reduced travel limit, and a total actuator breakdown with the surface stuck at an arbitrary position.

The control surface produces the force vector **f** and the moment vector **m** (about the center of gravity of the aircraft). Both depend on the surface position u and the magnitude of the surface area a. The air density ρ and the true air speed V_A are additional influences:

$$\begin{pmatrix} \mathbf{f} \\ \mathbf{m} \end{pmatrix} = \boldsymbol{b}(\rho, V_A, a) \cdot k_{FE} \cdot u, \quad k_{FE} \in [0 \ldots 1] \qquad (2)$$

Concerning the failure types only a reduced surface area is considered modelled by the factor k_{FE}.

5. Basic Controller Structure

5.1. Introduction to the Structure

The reconfiguration concept is founded on a special controller structure. This chapter presents the basic ideas underlying this structure.

A reconfiguration of a controller means making it execute the following task:

If there is a damaged component in the control effector system, forward that part of the command that cannot be fulfilled to other components in order to fulfil the pilot's commands as in the no-failure-case.

Many reconfiguration concepts come up with a standard controller structure to execute this task - a simple feedback structure generating control commands from the error between measured and commanded values. So the reconfiguration has to focus on the controller gains. The drawback is the necessary reconfiguration effort in case of nonlinear failure types. These failure types may require a high order controller with many parameters difficult to design. This means much calculation time for redesigning the controller in such failure cases.
Our approach wants to avoid this by building up a special controller structure that can automatically do the reconfiguration task mentioned above. This solution considerably reduces the reconfiguration effort.

The comprehensive aircraft model is used to generate control commands by a *feedforward* part

[4]. Using the explicit actuator model of Eq. (1) such a feedforward structure can easily realize the reconfiguration task even for nonlinear faults. If the feedforward part has sufficiently precise information about the process it can overtake the control task of

- the generation of satisfactory command following.

So there remain for the controller *feedback* part the tasks of

- stabilization, and

- disturbance rejection.

By this the influence of the actuator rate and travel limits on the feedback commands is considerably reduced because the feedback command amplitudes are smaller. In this way a robust but simple feedback structure can be found that need not be reconfigured in case of changed travel or rate limits.

5.2. Single Output Case

The following simple example illustrates the development of the controller structure and is shown in Fig. 2:

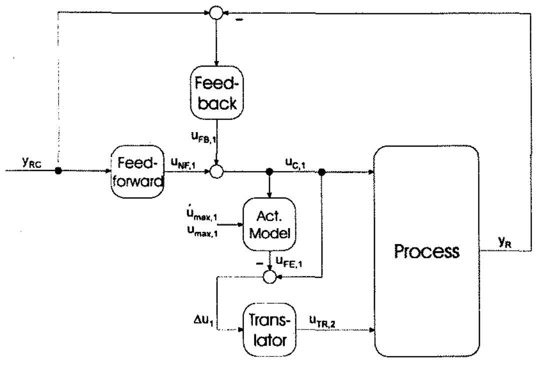

Fig. 2: Basic Controller Structure for Single Output Case

Consider a system with one controlled output y_R that is controllable by a primary input u_1 and a secondary input u_2. The secondary input is only used in the failure case to support the impaired effector system of u_1. The transfer function form is:

$$y_R = \begin{pmatrix} g_1 & g_2 \end{pmatrix} \cdot \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} \tag{3}$$

Building up the structure a feedforward element is introduced to transform the pilot's commands into inputs for the failure free case. In the failure free case the commands are fulfilled by the actuators without rate or deflection saturation. To leave the illustration simple the commands are generated by the inverted transfer function comprising the process information:

$$u_{NF,1} = g_1^{-1} \cdot y_{RC} \quad . \tag{4}$$

The command $u_{NF,1}$ is filtered by an actuator model according to Eq. (1) that is updated by the actual values for rate- and travel limits $\dot{u}_{max,1}$ and $u_{max,1}$. The redistribution of commands according to the reconfiguration task works as follows: A daisy-chain is installed around the actuator model of u_1. The difference of $u_{FE,1}$ and $u_{C,1}$ is the part Δu_1 of the command that has to be transformed into a command for the secondary input u_2 in order to fulfil y_{RC}. The missing contribution of u_1 to the output is:

$$\Delta y_{RC} = g_1 \cdot \Delta u_1 \tag{5}$$

As soon as the actuator model is saturated because of the reduced deflection or rate limit in the failure case, the signal Δu_1 is calculated. If the actuator model is not saturated Δu_1 is set to zero. Otherwise the dynamic of the first order lag would cause a Δu_1 signal despite the absence of failures. u_2 can be calculated by

$$u_2 = g_2^{-1} \cdot g_1 \cdot \Delta u_1 \quad . \tag{6}$$

The function $g_2^{-1} \cdot g_1$ "translates" $\Delta u_1(t)$ into $u_2(t)$ to compensate the actuator failure. Therefore this component is called *translator* in the following text. Assuming that u_2 fulfils the command the desired output is fulfilled as in the failure free case despite the rate and travel saturations of u_1.

The feedback commands are added to the command for the primary actuator model. The feedback is not activated because of the equality $y_R(t) = y_{RC}(t)$. If u_2 cannot carry out the commands of the translator, the output equality doesn't apply, $y_R(t) \neq y_{RC}(t)$. In this case the feedback is activated to make the process follow the desired output as good as possible.

If there are other secondary inputs available this controller structure can be augmented easily to integrate them. Defining a sequence order for the secondary inputs, components of an actuator model, a daisy-chain, and the respective translator are formed and connected.

5.3. Multi Output Case

The structure concept built up in the chapter above can be easily augmented for a greater number of controlled output variables, see Fig. 3. The translators and the actuator models were united into banks of translators and actuator models:

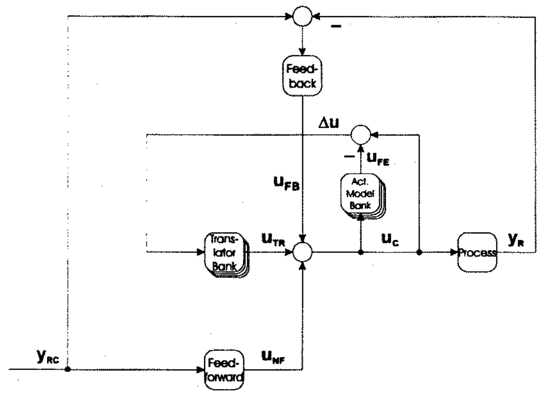

Fig. 3: Basic Controller Structure for Several Control Outputs

The model of the failure free aircraft is used in the feedforward block. m controlled output variables and l input variables $u_i, u_j, u_k, \ldots$ have to be chosen as primary inputs for the failure free case ($m < l$). Every input that can be replaced by a group of other inputs creating the same trajectory $\mathbf{y}_R$ (t) has to be connected to those inputs via a translator. The differences $\Delta u_i, \Delta u_j, \Delta u_k, \ldots$, summarized by $\Delta\mathbf{u}$, are generated by the daisy-chains and directed to the respective translators. Every translator uses a group of inputs that must not contain the input to be replaced. The feedback commands are added to the inputs of the primary actuator models.

In the failure free case every primary actuator model can fulfil the command $\mathbf{u}_{NF}$; $\Delta\mathbf{u} = \mathbf{0}$, $\mathbf{u}_{TR} = \mathbf{0}$ and $\mathbf{u}_{FB} = \mathbf{0}$. If some saturated actuator models are not able to fulfil their commands the translators are activated by the signal $\Delta\mathbf{u}$, such that $\mathbf{u}_C = \mathbf{u}_{NF} + \mathbf{u}_{TR}$. The output trajectory is similar to that of the normal case, $\mathbf{y}_R$ (t)= $\mathbf{y}_{RNF}$ (t)= $\mathbf{y}_{RC}$ (t). If $\Delta\mathbf{u}$ cannot be translated successfully into secondary inputs because of travel or rate saturations in the secondary actuators, the output differs from the desired output $\mathbf{y}_R$ (t)$\neq$ $\mathbf{y}_{RC}$ (t). By this the feedback is activated making the process follow $\mathbf{y}_{RC}$ as good as possible. The feedback contributes to the control command such that $\mathbf{u}_C = \mathbf{u}_{NF} + \mathbf{u}_{TR} + \mathbf{u}_{FB}$.

5.4. Realization of the Concept

The realization of the concept presented in the section above makes a modification of the controller structure necessary. According to the structure the choice of the feedback variables is restricted to the controlled output variables. They are chosen as

$$\mathbf{y}_R = [V_A \quad \gamma \quad \Phi]^T \tag{7}$$

Flight control experiences [5] show the necessity of feeding back additional variables such as the roll velocities p_h, q_h, r_h, angle of attack α, and sideslip angle β in order to gain sufficient control performance. For determining the feedback error command values have to be generated to compare with the measured values. For this reason the controller structure is augmented by an aircraft model, the block "A/C Model", generating the desired commanded values for the feedback, see Fig. 4. It contains the control surface model of Eq. 2 and the nonlinear flight dynamics. Thus this block together with the block "Act. Model Bank" represents the comprehensive aircraft model. The actuator model outputs $\mathbf{u}_{FE}$ are fed into the a/c model. The feedback of Fig. 3 has changed into a model feedback.

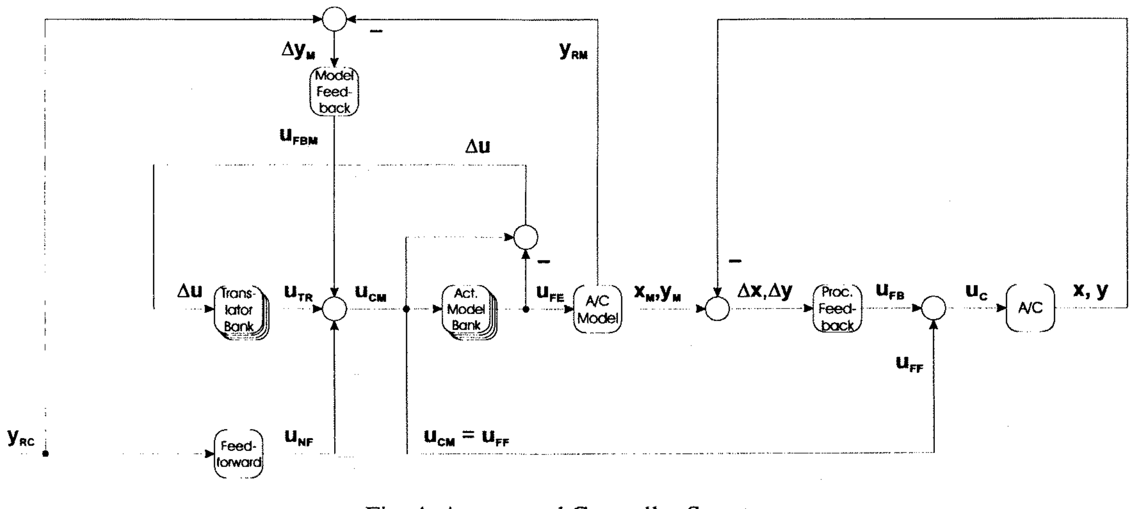

Fig. 4: Augmented Controller Structure

The left half of the structure up to the block "A/C Model" forms the *feedforward* part of the controller. The commanded values for the actuator models are now the feedforward commands for the aircraft. Thus an explicit model following structure is created comprising a comprehensive model of the aircraft in the feedforward part. By this the trajectories $\mathbf{x}$ and $\mathbf{y}$ of the aircraft should follow the state and output trajectories $\mathbf{x}_M$ and $\mathbf{y}_M$ of the model. The model feedback and the process feedback have proportional and integral parts.

6. Controller Reconfiguration

The types of failure were introduced during the control effector modelling. Every failure type requires different reconfiguration actions. They have one aspect in common: the model updating relying on the information from the FDI procedure.

The second step is to compensate the failure by modifying the input commands. After that the genetic algorithm is activated to optimize the model parameters to maximize controller performance.

6.1. Reduced Surface Effectiveness

In the aircraft model the failure parameter k_{FE} of the corresponding control surface is updated (Eq. (2)). This type of failure can be simply compensated by increasing the surface deflection. Hence the modified command for the actuator model is:

$$u_{C.FE} = \frac{1}{k_{FE}} \cdot u_{C.NF} \tag{8}$$

The gains of the process feedback referring to the impaired control input have to be multiplied by $1/k_{FE}$.

6.2. Stuck control surface

The control surface and the actuator remain in an arbitrary position. The actuator model is updated by setting the actuator state u of Eq. (1) to the fixed surface position and the gain k and $\dot{u}$ to zero. The failure is compensated automatically by generating the Δu signal. By this the respective translator is activated and generates the commands for the secondary inputs. The neutralization of the disturbance input of the stuck surface is automatically included.

The process feedback matrix has to be replaced by a new one using a secondary input instead of the damaged one. The failure gain matrices are determined in advance and stored in the flight computer.

6.3. Reduced rate limits

The affected actuator model is updated by changing the parameter $\dot{u}_{max}$. The failure is compensated automatically by generating the Δu signal as soon as there is a rate saturation in the actuator model. By this the respective translator is activated and transforms the commands for the secondary inputs. The process feedback need not be reconfigured.

6.4. Reduced travel limits

The affected actuator model is updated by changing the parameter u_{max}. The failure compensation is done automatically by generating the Δu signal as soon as the actuator model is saturated. By this the respective translator is activated and transforms the commands for the secondary inputs. Again the process feedback need not be changed.

7. Identification of Model Parameters by the Genetic Algorithm

The automatic parameter identification begins after the information of the FDI process has been exploited for reconfiguration. It adapts the A/C model to the real aircraft. The A/C model, not necessarily linear, is given as

$$\dot{\mathbf{x}}_M = f[\mathbf{x}_M, \mathbf{u}_M, \mathbf{p}] \tag{9}$$

with the parameters $\mathbf{p}$ to be tuned by the genetic algorithm. The task is to:

1. Measure $\mathbf{u}$, $\mathbf{x}$, and $\dot{\mathbf{x}}$ of the aircraft

2. Optimize $\mathbf{p}$ so that the dynamics of the model correspond to the dynamics of the aircraft

3. Use $\mathbf{p}$ in the model and the controller

The cost function the genetic algorithm has to minimize is derived from Eq. 9:

$$\Delta\dot{\mathbf{x}} = \left|\dot{\mathbf{x}} - f(\mathbf{x}, \mathbf{u}, \mathbf{p})\right| \tag{10}$$

7.1. Introduction to Genetic Algorithms

Mother nature has had time over billions of years to develop an algorithm to optimize her creatures. This means that the genetic optimization algorithm itself is the outcome of the profoundest optimization procedure ever applied on earth. This genetic algorithm, called evolution, has created systems like the albatross, dynamically soaring hours and hours over the sea without a single wing beat, or the skin of a shark, that generates micro-turbulence for significant drag reduction. Both are examples of fascinating biological phenomena that we scientists are just beginning to understand and utilize to make our aircraft more efficient. So - if life successfully uses the genetic algorithm to optimize and adapt its systems, it seems to be a good idea to apply the same strategy to technical optimization problems as well.

7.2. Selection, Crossover, and Mutation

Based on the biological model the technical genetic algorithm defines a population of individuals, all of them having different properties. A certain fitness (or cost, depending on whether a maximum or a minimum is looked for), has to be calculated for every individual in a user supplied fitness function. This fitness function is the mathematical description of the optimization problem, gets the characteristics of an individual as its input, calculates the fitness of this individual with regard to the problem and forwards this fitness to the genetic algorithm.

Under the control of the genetic algorithm every individual gains certain access to a mating pool. As the fitness of an individual represents the probability to appear in the mating pool, "good" individuals get a higher chance to reproduce than individuals with a lower fitness (*survival of the fittest, selection*).

In the recombination phase members of the mating pool are chosen arbitrarily two by two. They exchange their characteristics and generate children that have properties similar but not identical to those of their parents (*crossover*). Since individuals with greater fitness have a higher probability to be selected for reproduction, the children's generation is very likely to have a higher mean fitness than their parents. Looked at from the geometric point of view the new population as a whole has converged one step closer to the optimum. This does not automatically mean, that the best individual of the new generation has a higher fitness than the best of the parents. In general, it is just the mean fitness of the whole generation that is increased [6].

Starting from a widespread initial population, that ideally covers the whole parameter space (Fig. 5 Left), the population converges, contracts, and shrinks towards the optimum in the middle (Fig. 5 Middle), until finally nearly all individuals are concentrated close to the optimum (Fig. 5 Right):

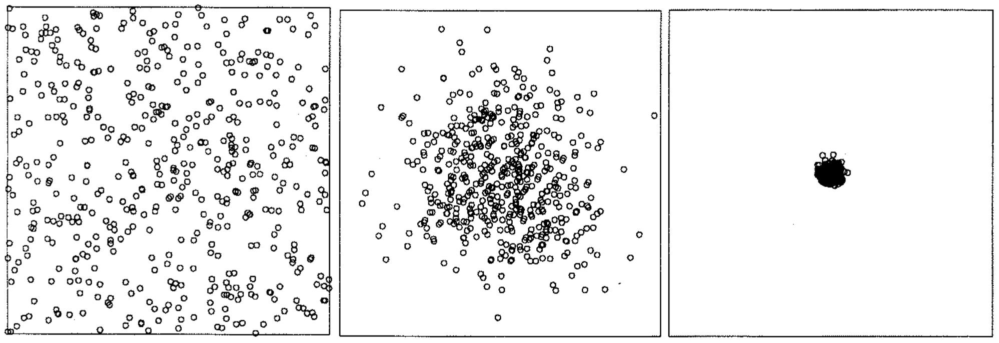
Fig. 5: Population Development

Unfortunately the described procedure of selection and crossover has one great drawback: Along with the convergence the population loses properties. If for example one property to be optimized is the color of the individuals, all colors (blue, red, white, black, ...) should be present in the initial population. If now the optimum itself is blue, all non-blue individuals will be substituted by blue individuals in the end. This is okay, as long as the optimum is static and does not move. It becomes fatal, if the optimum *changes* its properties. Sticking to the example, all of a sudden blue is out, and red would be optimal. Unfortunately there *are* no more red individuals left and it is impossible to generate red children from blue parents. Such a pedigree population would have no chance to find the new red optimum "out of the blue".

The usual trick to get round this problem is to introduce *mutation*. By mutation in every generation a few individuals have their properties changed arbitrarily, positioning them more or less far away from the latest optimum. If now a new red optimum has cropped up anywhere else in the parameter space, unreachable for those formerly optimal blue masses, a few individuals get their color changed to red by mutation. These individuals suddenly have a much higher fitness than those blue ones and therefore get a much higher chance to produce children. A few generations later most of the blue individuals have died out and the whole population gathers round the new red optimum.

8. Implementation of the Genetic Algorithm

During the last 30 years hundreds of different implementations of genetic algorithms have shown that the algorithm itself is a very flexible tool that can be modified and adapted to fit about every optimization problem. One of its greatest benefits is the fact that you can find a biological analogy for almost any implementation question. Nature, society, and even politics are full of more or less perfectly working optimization strategies that can successfully be implemented into genetic algorithms.

For our on-line identification problem there are further advantages of the genetic algorithm in opposition to conventional methods like least square methods:

- The model to be identified need not be linear.
- The problem of missing excitation with the loss of identification results can be solved easily.

8.1. Different Optimization Tasks

Four main different optimization problems can be solved by the genetic algorithm:

8.1.1. Static Optimum

In general this is the easiest task. The optimum is constant, it does not change or move. The population starts from a uniform distribution, converges, finds the global optimum, and quits. A one-dimensional example would be to find the independent variable x that minimizes the cost-function $c(x)$:

$$c(x) = 1 + x + x^2 + x^3 + x^4 \tag{11}$$

8.1.2. Moving Optimum

If the location of the optimum is not constant, but moves around or suddenly appears somewhere else (for example after a sudden fault), the optimization algorithm must be able to follow the optimum on its path through the parameter space. The genetic algorithm mainly uses the mutation to send out a certain percentage of the population far away from the present optimum having those individuals search the whole parameter space for new optima. Slowly moving optima are mainly followed by recombination of present high-fitness individuals. Eq. (12) shows a simple example of a cost function with a moving (t = time) optimum.

$$c(x,t) = 1 + tx + x^2 + x^3 + x^4 \tag{12}$$

8.1.3. Identification of Dynamic Systems

If we want to use the genetic algorithm for the adaptation of a feedforward controller or a model, we run into the general on-line identification problems. Consider the simplified linear roll acceleration differential equation of an aircraft [5].

$$\dot{p} = L_p p + L_\xi \xi \tag{13}$$

Roll acceleration $\dot{p}$, roll rate p, and aileron deflection ξ can all be measured in every sample interval; the task is to identify the unknown constant aircraft parameters L_p and L_ξ. This looks like a simple underdetermined equation (system) with one equation for two unknowns. So all we need is two equations from two different times to solve for the two unknowns.

$$\begin{aligned} \dot{p}(t_1) &= L_p p(t_1) + L_\xi \xi(t_1) \\ \dot{p}(t_2) &= L_p p(t_2) + L_\xi \xi(t_2) \end{aligned} \tag{14}$$

Unfortunately the measurements in a real-world aircraft are not at all ideal. Dropouts can occur, signal noise is always present; the latest two measurements could be totally disturbed. As a consequence the two calculated parameters would be absolutely wrong and useless. If used in the actual feedforward controller, they could even harm the performance of the aircraft.

The usual way to deal with noise is to filter. One possible alternative is to filter the signals directly, before they are used in the parameter identification process. The second way stores more than two old measurements in an information matrix and forwards the corresponding overdetermined equation system to an equation solver [7],[8].

$$\begin{aligned} \dot{p}(t_1) &= L_p p(t_1) + L_\xi \xi(t_1) \\ \dot{p}(t_2) &= L_p p(t_2) + L_\xi \xi(t_2) \\ &\vdots \\ \dot{p}(t_n) &= L_p p(t_n) + L_\xi \xi(t_n) \end{aligned} \tag{15}$$

In such a simple linear case the most effective regression algorithm would be a singular value decomposition of the information matrix to solve the overdetermined equation system in a least squares sense [9].

$$\begin{aligned} &\dot{\mathbf{p}} = L_p \mathbf{p} + L_\xi \boldsymbol{\xi} \quad \text{with} \quad \dot{\mathbf{p}} = \begin{bmatrix} \dot{p}(t_1) & \dot{p}(t_2) & \cdots & \dot{p}(t_n) \end{bmatrix}^T \ldots \\ &\dot{\mathbf{p}} = \begin{bmatrix} \mathbf{p} & \boldsymbol{\xi} \end{bmatrix} \begin{bmatrix} L_p \\ L_\xi \end{bmatrix} \\ &\left\| \dot{\mathbf{p}} - \begin{bmatrix} \mathbf{p} & \boldsymbol{\xi} \end{bmatrix} \begin{bmatrix} L_p \\ L_\xi \end{bmatrix} \right\|_2 \to \text{Min} \end{aligned} \tag{16}$$

The genetic algorithm offers a third, more elegant way to take care of underdetermined equation systems and disturbances in the measured data [9]. No external storing or filtering of the data is necessary, the algorithm per se has enough inertia to directly handle one set of measurement data at a time and to suppress noise or dropouts. If for example a short signal peak occurs because of a measurement error, the optimum will abruptly jump to another location in the parameter space and jump back a few time samples later. Even if, by accident, one individual should find this new, but "wrong" optimum, it would take more than just a few generations to build up a new population in that region. As soon as the fake optimum disappears, all mistaken individuals will die out, i.e. the renegade part of the population will move back to the true optimum.

The same low pass filter characteristics of the genetic algorithm that successfully damp the disturbances are automatically utilized to handle the underdetermined Eq. (13). Let the cost function to be minimized be the absolute error of Eq. (13).

$$c(L_p, L_\xi) = |\dot{p} - L_p p - L_\xi \xi| \rightarrow \text{Min} \tag{17}$$

For every new $[\,\dot{p}\ p\ \xi\,]$ measurement data set Eq. (17) can be geometrically interpreted as another cost valley:

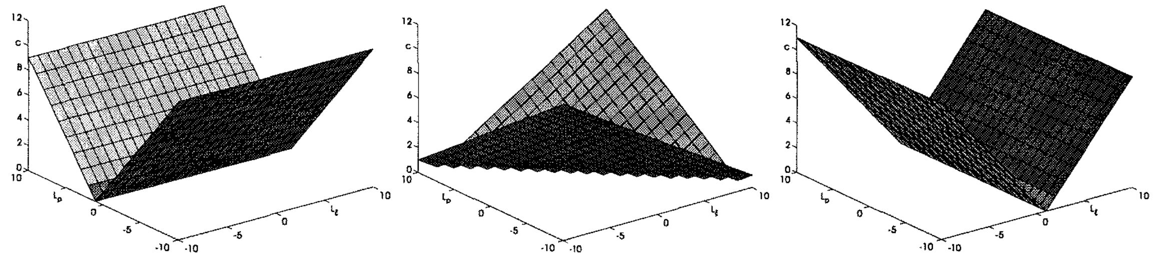

Fig. 6: Different Cost Valleys

All three valley bottoms in Fig. 6 have only one point in common, which is the unknown optimal parameter vector.

$$[L_p \quad L_\xi]_{opt} = [1 \quad 1] \tag{18}$$

With every excitation of the aircraft the measurement vector $[\,\dot{p}(t)\ p(t)\ \xi(t)]$, that represents the dynamic response of the aircraft, generates an oscillation of the valley around its fixpoint. The higher the frequency of the excitation and the resulting aircraft response are, the faster the oscillation of the valley will be. The genetic algorithm will now initially position his individuals all across the two-dimensional parameter space. Quickly all individuals on the hillside will be sorted out because of their high cost. Only those individuals around the optimum will survive and produce children. Their cost is constantly low, no matter how the valley is oriented.

The convergence speed of the genetic algorithm has to be carefully coordinated with the frequency of the aircraft response. If the genetic algorithm is much faster than the motion of the valley, the whole population will converge like a drop of water at a random point of the valley bottom, not necessarily at the optimum. With the slow motion of the valley this point might wander around in the valley bottom, without becoming stationary at the optimum.

In a real-time genetic algorithm the convergence speed can easily be controlled by the number of generations per sample interval. One can calculate hundreds of generations whenever one gets new information about the process to identify, or one can wait decades of sample intervals before using a new measurement for the next generation. The genetic algorithm should on the one hand suppress the influence of disturbances and not try to follow excitation and fast system dynamics. On the other hand it should willingly track longterm parameter variations. Its convergence speed (frequency) therefore should be obtained by the traditional tradeoff between the highest parameter variation frequency and the lowest noise frequency (and the system response respectively).

8.2. Conservation of the Identification Results During Missing Excitation

Sufficient excitation is necessary for identifying the dynamics of a system. Using the genetic algorithm the impossibility to gain further information about the system during missing excitation can be interpreted as a stillstanding cost valley. The optimum is no longer visible and identifiable. All the individuals lying at the bottom of the valley have the same optimal cost value zero. The problem are situations like in Fig. 7. The cost valley from (Fig. 6 Left) is seen from above:

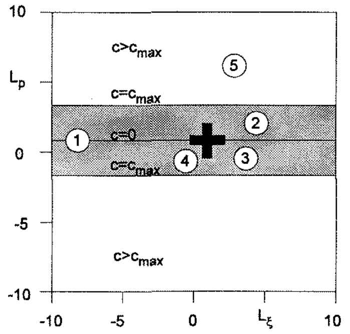

Fig. 7: Frozen Individuals During Missing Excitation

There is no excitation so that the valley does not move. The bottom line of the valley runs from the left side across the optimum at the point [1 1] (black cross) to the right side. A big part of the population, namely the individuals No. 2, 3, and 4, have approached the optimum very closely. Unfortunately individual No. 1 has a better cost than the rest of the population. Creating new generations this would wipe out the well converged individuals No. 2, 3, and 4 and making up new individuals near to the alleged best individual No. 1.

This can be avoided by a simple measure: The possible best cost value is zero. Defining a threshold c_{max} all individuals with cost values between zero and c_{max} are considered as "good enough". So this part of the population represents the acquired information about the system parameters and is excluded from the *crossover* and *mutation process* respectively copied into the next generation. These individuals "get frozen". The grey area in Fig. 7 marks the area holding the individuals that have to be copied because they are better than c_{max}. Individual No. 5 is not good enough and may converge into the grey area.

The compromise of this measure can be found in the reachable accuracy of the identification setting up such a "freezing area". As soon as an individual is better than c_{max}, the convergence is stopped. Such an individual is prevented from a better convergence to the real optimum.

9. Flight Experiments

DLR's in-flight simulator ATTAS (Fig. 8) was used as test bed for the validation of the concept. In September 1993 flight tests were conducted to examine the controller adaptation by the genetic algorithm and the explicit model following controller concept for reconfiguration. Tracking tasks were designed with and without elevator failures such as a reduced control effectiveness and a stuck elevator. Further flight tests are planned for the end of August 1995 to verify the complete reconfiguration concept.

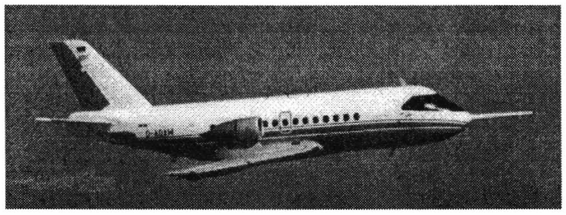

Fig. 8: DLR's In-Flight Simulator ATTAS (Advanced Technologies Testing Aircraft System)

9.1. Flight Test with Genetic Algorithm

Tracking task experiments were designed to compare a fixed and an adaptive feedforward controller after a 50% elevator effectiveness reduction [10], [11]. The pilot's task was to follow a moving pitch bar on his flight display as accurate as possible. The parameters of the fixed feedforward were calculated offline prior to the flight experiment by pseudo-inverting a linearized model of the undamaged aircraft. The adaptive controller on the other hand used a genetic reconfiguration algorithm to tune these parameters in order to minimize the control error. No extra knowledge based structural reconfiguration took place ahead of the adaptation. The parameter boundaries, limiting the parameter space the genetic algorithm could search, were chosen to cover any reasonable fault.

Fig. 9 shows the time histories of two flight experiments. In both experiments the elevator effectiveness reduction takes place after 100 seconds. During the first experiment the pilot has to manage the reconfiguration "in his head"; he has to increase his commands in order to compensate for the reduced actuator effectiveness. In the second experiment the genetic algorithm identifies the actuator fault and optimizes the feedforward gain matrix. Obviously (Fig. 9) the pilot has much less difficulties to fulfill his task, after the genetic algorithm has adapted the control system.

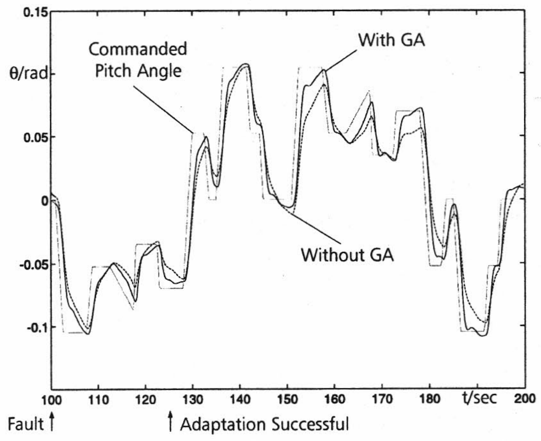

Fig. 9: Flight Test Results with and without Genetic Algorithm

The above observation is underpinned by Fig. 10, which indicates the integrated (mean) error between pitch angle task and actual pitch angle, both for the fixed feedforward (no genetic algorithm) and the adaptive feedforward under the control of the genetic algorithm. While the errors of fixed and adapted feedforwards are about equal, as long as the actuator fault is not present (0 - 100 sec.), the fixed feedforward error increases significantly after the occurance of the fault at 100 seconds. The error with the adapted feedforward shows only a slight increase which is due to the fact that the feedback controller is not adapted.

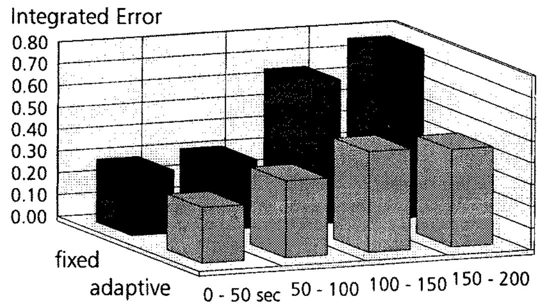

Fig. 10: Error of Fixed and Adaptive Feedforwards with Elevator Fault after 100 Seconds

During the flight experiments the pilots were asked to comment on their workload and the way the adaptation process influenced the handling qualities of the aircraft:

Tab. 1: Pilot Comments during Flight Experiment

Fixed Feedforward Controller	Adaptive Feedforward Controller
"Bit more work"	"Distinct learning process"
"Tends to shoot over"	"Controller is getting faster"
"Works slower"	"Almost like the intact elevator"
"You can learn to live with it"	

Even though the pilots have not been asked in this first flight test to quantitatively assess aircraft handling qualities via a scale, such as the Cooper-Harper-rating scale, their comments clearly underline the fact that genetic algorithms can definitely reconfigure control system parameters after a fault in the actuator system.

9.2. Flight Test with Model Following Controller

In a second flight test the reconfiguration capabilities of the explicit model following controller were examined using Cooper-Harper-Ratings. The failure was an elevator stuck in the trim position. The flight task was to execute a so-called "bathtub manoeuvre" (Fig. 11). Starting the test run at an altitude of 15,000 ft with an indicated airspeed of 190 kts the pilot had to deccelerate the aircraft down to 160 kts and keep the altitude constant. In a second phase a descent with a vertical speed of -500 ft/min with constant speed down to 13,000 ft was to be done. After the pilot had stabilized the aircraft at this altitude the manoeuvre continued in a reversed manner:

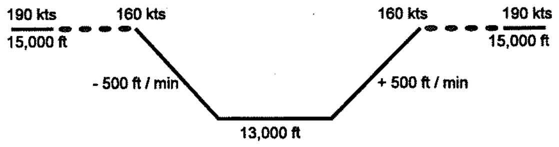

Fig. 11: Bathtub Manoeuvre

In the first run the pilot had to manage the reconfiguration by himself using the trim button on his sidestick generating a stabilizer command. The thrust levers were available as second input. The stabilizer had a rate limit of 0.125 deg/s rendering the stabilization of the aircraft difficult what can be seen in the altitude time history of Fig. 12:

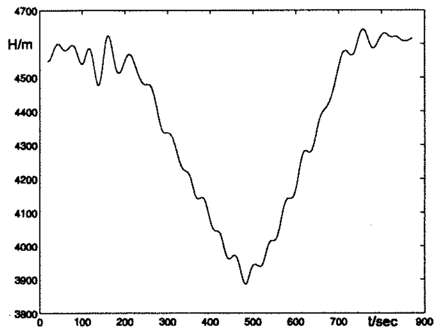

Fig. 12: Altitude Trajectory Without Controller

Obviously the weakly damped phugoid mode was the main difficulty for the pilot. An adequate performance was not attainable with a tolerable workload expressed by a Cooper-Harper-Rating of 7 for this run. In the second run the pilot was supported by the reconfigured controller. Its controller structure compared to the structure that was used for the genetic algorithm. For reconfiguration purposes it was augmented by a control mixer that redistributed the commmands of the damaged elevator to the stabilizer. The control law was a "pitch rate command-attitude hold-law". By deflecting the sidestick a pitch rate was commanded. By returning the stick to the neutral position the controller reduced the pitch rate and switched over to the hold mode as soon as the pitch rate had sufficiently decreased. The controller commanded the stabilizer while the pilot set the thrust levers. Fig. 13 shows the altitude time history.

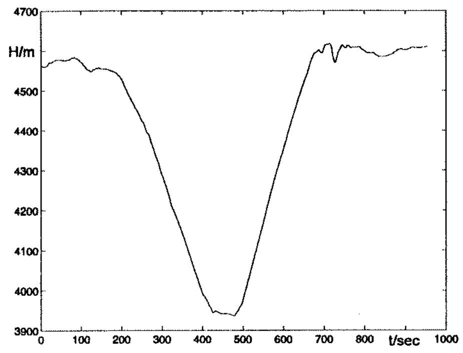

Fig. 13: Altitude Trajectory with Controller

The phugoid mode was well damped now. The execution of the bathtub manoeuvre was considerably improved by the controller. This resulted in a Cooper-Harper-Rating of 5. The runs with and without the controller were repeated several times to confirm the results. Concerning speed tracking there was no improvement by the controller to be found in comparison to the runs without controller.

10. Conclusions

A controller concept was developed based on a special controller structure. The three objectives set up were sufficiently fulfilled:

1. Only some gain matrices for the failure type of a stuck surface have to be stored in advance.

2. The calculation time for new controller gains is limited to simple multiplications of gains.

3. Linear and nonlinear failure types like reduced effectiveness, stuck surface, reduced rate limit, and a reduced travel limit were taken into consideration.

The implementation of a genetic algorithm turns out to be a useful tool for controller reconfiguration. Linear and nonlinear failure types can easily be identified.

The first flight test campaign showed the considerable capacities of the chosen controller structure and the genetic algorithm with regard to reconfiguration.

11. References

[1] Buchholz, J. J., *Genetic Algorithms for Reconfiguration*; to be published in ZFW, Journal of Flight Sciences and Space Research, Springer Verlag, Berlin

[2] Baumgarten, G., *Rekonfiguration - Literaturrecherche*, IB 111-93/59, Inst. f. Flugmechanik, DLR Braunschweig, 1993

[3] Moerder, D. D., Halyo, N., *Application of Precomputed Control Laws in a Reconfigurable Aircraft Flight Control System*, Journal of Guidance, Control and Dynamics, Vol. 12, No. 3., 1988

[4] Baumgarten, G., *Rekonfiguration eines Flugreglers bei Stellflächenfehlern - Erste Ergebnisse*, DGLR, Sitzung des Fachausschusses T 5.3, Braunschweig, 1994

[5] Brockhaus, R., *Flugregelung*, Springer-Verlag, Berlin, 1994

[6] Goldberg, D. E., *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading, MA, 1989

[7] Ljung, L., *System Identification, Theory for the User*, Prentice-Hall, Englewood Cliffs, NJ, 1987

[8] Becker,T. T., *Methoden kleinster Fehlerquadrate zur parametrischen Identifikation dynamischer Systeme, Fortschritt-Berichte VDI, Reihe 8, Nr. 176*, VDI-Verlag, Düsseldorf, 1989

[9] Porter, B., Hicks, D. L., *Genetic Design of Digital Model-Following Flight-Control Systems*, Proc. AIAA Guidance, Navigation and Control Conference, Monterey,1993

[10] Heine, W., *Genetischer Algorithmus - Theorie und Anwendung in einem Flugregler*, Aspekte der Informatik im Arbeitsbereich der Ingenieure, FF - FM Workshop, DLR Braunschweig, 1994

[11] Heine, W., *Rekonfiguration der Vorsteuer-parameter einer Modellfolgeregelung mit Hilfe eines genetischen Algorithmus*, Informatik Aktuell, 4. Dortmunder Fuzzy Tage (Poster Session), Springer Verlag, Berlin, 1994