Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture,
Volume 231, Issue, 13, pp2291-2301.
DOI:10.1177/0954405415598924

Self-repairing design process applied to a 4-bar linkage mechanism

Colin Bell[1], Michael Farnsworth[2], James Knowles[3] and Ashutosh Tiwari[2]

[1]Department of Mechanical Engineering and Mathematical Sciences,

Oxford Brookes University, Oxford, UK

[2]School of Aerospace, Transport Systems and Manufacturing,

Cranfield University, Cranfield, UK

[3]Department of Aeronautical and Automotive Engineering,

Loughborough University, Loughborough, UK

Abstract

Despite significant advances in modelling and design, mechanical systems almost inevitably fail at some point during their operative life. This can be due to a pre-existing design flaw, which is usually overcome in a revision, or more commonly due to some unexpected damage during operation. To overcome a failure during operation, a new method of designing machines or systems is proposed that creates a result that is resilient to both expected and unexpected failure. By shifting the focus from a detailed assessment of the underlying cause of failure to how that failure will manifest, a system becomes inherently resilient against a wide range of failure modes. The proposed process involves five steps: Cause, Detection, Diagnosis, Confirmation, and Correction. This is demonstrated with an application to a generic four-bar linkage mechanism. Through this process the system is able to return to a near perfect state even after a permanent deformation occurs in the mechanism. These results show the potential that this self-repairing design process has for applications including robotics, manufacturing and other systems.

Keywords

Self-repairing, linkage mechanism, design, optimisation, mechanical systems

Introduction

In an era of rapid technological development, largely driven by significant advances in computational power and global collaboration, engineering systems are becoming exponentially more complex and less reliable. Although improvements in maintenance regimes and through-life services can enhance the reliability of systems, it is almost inevitable that failure will occur at some point during their operation and these failures are often due to unexpected events [1]. Providing reactive maintenance following an unexpected failure is often expensive, and can be impossible to achieve quickly (if at all) in certain circumstances, e.g. during space exploration. Several current design approaches exist to mitigate the risks associated with failures in operation: system redundancy can be built in [16]; controllers can be employed to cope with damaged hardware [17]; systems can be pre-programmed to cope with certain types of specific failures [18]. Although these methods are adequate for designing against anticipated failures, achieving robust performance under uncertainty is far more difficult [2]. Self-healing is a phenomena most commonly associated with biological systems, for example mammalian skin and its ability to repair after serious injury to a fully functional state. An alternative to this is the term 'self-repairing' often commonly associated with physical systems such as electronics or mechanical systems that can similarly exhibit some ability to regain a fully functional state. Self-healing can be seen as a bottom-up approach, where the components of the system heal any damage from the inside [12]. Much work conducted into self-healing technologies has focused upon the material level, such as self-healing composites [13, 14]. Here, passive systems provide the structure with regenerative properties in an attempt to restore its strength following a structural failure. Such passive systems offer potential for coping with small-scale structural damage, however large deviations from the undamaged structural state may require additional input to re-align severely damaged structures in order to maintain a reasonable level of functionality (cf. broken bones in animals). For a system to repair under these circumstances, it would have to identify damage before deciding on the best course of action to correct the damage.

Much like adaptable design [15], self-repairing systems require specific design methodologies to maximise the benefits offered to the mechanical system under consideration. Designing for self-repair is a top-down approach where the system is designed to have the ability to maintain its own function through external factors such as diagnosis and reconfiguration. Self-repairing autonomous systems or electronics are able to detect and diagnose faults for example and either isolate the fault and replace them or through external action repair the damage [12]. Such characteristics provide a system that can therefore identify and correct unanticipated damage. Recent work by Koos et. al. proposes an algorithm that allows machines to adapt to failure modes by learning from an internal model of themselves. They test their algorithm with application to a hexapod robot, and show that it offers significant performance advantages over other robust design methodologies, such as optimised control based methods [1]. Although the proposed algorithm can maintain certain functionalities, it does not explore the possibility of the machine repairing the physical damage. This is because in general terms, what is often desired is a system that is able to "Maintain some degree of functionality after a failure has occurred" [3]. For a further discussion on the taxonomy of self-healing and self-repairing systems and their differences can be found in [12]

To achieve a self-repairing system, it is clear that the system must have an element of self-awareness. Amor-Segan et al. [4] originally stated that what is desired is a system that has "the ability to autonomously predict or detect and diagnose failure conditions, confirm any given diagnosis, and perform appropriate corrective intervention(s)". From this description, the 'design for self-repairing' process has previously been broken down into five steps [5]:

1. Cause of fault: This is numbered as such because in an ideal 'self-repairing' system the underlying cause of fault is irrelevant. Instead the system should be designed in such a way that all underlying causes are mitigated against by instead focusing on how these causes might manifest.

2. Detection of fault: All underlying faults within a system will inevitably lead to a fundamental change in the behaviour or output (else it could be argued that a fault hasn't occurred).
3. Diagnosis: Once a fault has been detected the system must then determine where and how that fault has occurred
4. Confirmation of diagnosis: Any fault that is diagnosed will have an associated confidence level based on how certain the diagnosis is. In some cases this will be sufficient to instigate a corrective action, in other cases, where multiple points of failure could be the culprit, it is important to confirm the diagnosis to avoid rerouting or replacing potentially unfaulty components.
5. Corrective action: Perhaps the most significant aspect of the self-repairing process, this will be application specific; however a number of possible approaches are available, which are discussed in more detail later.

It should be highlighted that closed-looped systems mirror much of the steps highlighted in above, with monitoring and diagnosis abilities used to control and optimize certain functions of a component or device, for example in an HVAC system. However there are a number of differences to these systems and the field of self-repair.

Firstly self-repair looks to overcome the loss or degradation of function whereas most control systems focus upon maintaining an optimised state in response to some kind of variation, often environmental in the case of HVACs. Self-repair often goes further than feedback systems when it comes to repair with actions resulting in the removal of damaged components, for example found in systems with many redundant parts such as electronics, or physical components such as self-repairing robotics as discussed previously. Closed-looped systems often monitor simple parameters such as environmental change, while self-repairing systems have to cope with higher levels of

complexity with regards to detection and diagnosis as the source of damage is often irregular and non-deterministic. The complication of most repairing strategies being one-off solutions makes taking the correct action crucial as it is not always possible to go back after corrective action has been employed.

To demonstrate the above process, this paper uses an example of a simulated 4-bar linkage mechanism. Mechanical linkages are, at a basic level, an assembly of rigid elements connected via joints to translate motion or force. Planar linkage mechanisms with revolute joints are widely used in industry either transmit torque, motion and power, or to transform one type of motion or force to another [6].

From a theoretical point of view, 4-bar linkages have been most extensively covered in literature due to their relative simplicity, making them an ideal case-study for a proposed self-repairing design approach. A simple example of a 4-bar linkage is shown in Figure 1. It is assumed that the mechanism is designed to trace a particular pattern. The apex of the triangular coupler link will follow a particular pattern when the input link is rotated one complete revolution. The specific pattern will depend on the geometry/length of the four links and the geometry of the triangular float link, which can be described completely by the length of l_1 and the two other tracer edge lengths (t_1 and t_2).

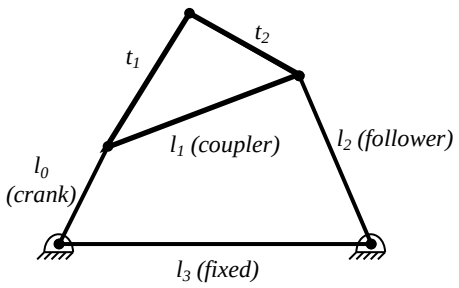

Figure 1. Simple 4-bar linkage mechanism.

It is assumed that the 4-bar linkage mechanism is designed to trace a particular pattern. Hence, the mechanism is deemed to have failed should its traced path deviate from the desired motion.

In the next section, the 'design for self-repairing' process is outlined for a four-bar link mechanism. The subsequent case studies section presents the process of applying this to a four-bar link mechanism that has experienced a fault over a number of the self-repairing process steps, before concluding remarks are provided.

Methodology: application of "design for self-repairing" to a four-bar mechanism

Productive and efficient design is a critical segment of the product lifecycle, and various methodologies have been proposed to enhance this. One such methodology is 'Design for X', which is a philosophy intended to focus a design around a particular parameter [7]. In this paper the proposed philosophy is 'Design for Self-repairing', which inherently covers a number of other parameters such as 'Design for reliability' or even 'Design for Maintenance' in that the system must be designed to be self-aware and hence maintenance can shift from being reactive to preventative. The final aim of such an approach is to remove the need for maintenance altogether. To demonstrate this, the steps previously stated are broken down and applied to a 4-bar linkage mechanism.

Cause of fault

It has already been stated that the cause of fault should not be the primary focus when designing a self-repairing system, but nevertheless it can serve as a useful tool in determining where faults can manifest. In the 4-bar mechanism considered here, a particular rod or element's dimension could be altered due to shock loading, manufacture defect, thermal expansion etc. The specific cause of the change in mechanism dimensions is irrelevant – the significant factor is that a change in mechanism dimensions will lead to a change in the behaviour of the system. In addition to the cause is the scope and scale of the fault, particularly in relation to its effect on the function of

the system. Wherever possible common faults that manifest in a particular design, component or system should be accounted for as a guide to designers when looking to build and integrate a self-repairing schema. In our example the loss of function of the tracer element through alteration in the 4-bar mechanism structure can result from an alteration to the element dimension or free play within the joint connectors. Anticipating that these areas are common to loss of function through the act of some form of damage or defect can lead designers to focus upon self-repairing solutions that perform corrective actions that act upon or mitigate this damaged area. For this case-study it is simply assumed that one or more of the rigid elements changes its length, why it occurred does not need to be considered here.

Detection

Perhaps the best way of detecting a fault is to look for a deviation in the prescribed behaviour, which can be achieved by utilising either internal or external telemetric data. Existing technology already allows for a number of externalised or embedded sensors and a number of high-end industries such as motorsport and aerospace have already began to implement this. By utilising these sensors the health and status of a system can be continuously monitored, with any deviation in expected behaviour used to identify that a fault has occurred.

The manifestation of failure within a 4-bar linkage mechanism could perhaps be most easily interpreted as a deviation from the prescribed tracer pattern. Hence it is assumed for this example that the system is aware of the tracer path, but not of a change in element dimensions.

Diagnosis

Whilst the detection of faults could be considered within the realms of current technology, the autonomous diagnosis of a fault is perhaps a more difficult concept. An analogy for this is that any user could potentially tell if something has gone wrong, but it often requires the input of an expert to say *why*. One of the reasons for this is the difficulty in validating large, complex system models that can exhibit a vast number of possible states [4].

Current methods for diagnosis include:

- Model-based: Abductive reasoning: compare observation with predicted observation: I expect 'X' but get 'Y', therefore I must correct 'Y' to get it to match.
- Bayesian belief networks: probabilistic graphical model (or statistical model) that represents a set of random variables and their conditional dependencies: If 'X' and 'Y' happen, it's likely a failure with 'Z'
- Case-based reasoning methods: anecdotal evidence, if 'X' happens, do 'Y'. – (Simplest, but only accounts for expected failure)

Since a precise mathematical relationship exists between the element dimensions and the tracer pattern, the most obvious choice for a 4-bar linkage mechanism is model-based reasoning. It is assumed that the system knows that the tracer pattern is no longer following the designed trajectory, so now it must infer the most likely point of failure(s) based purely on the information available.

Confirmation of diagnosis

One might reasonably ask if the 4th step, Confirmation of the Diagnosis, is required. To demonstrate the importance of this step, an analogy of a car repair is proposed. Currently the on-board diagnostics systems can present the most likely fault (as a fault code) given the available sensor data; however it is left to the mechanic to either:

- Look at the indicated faulty part to confirm it is indeed 'broken'
- Recreate the fault to confirm the diagnosis
- Use additional tools (multi-meter, etc.) to pinpoint the precise fault

It would seem rather ridiculous for the mechanic to blindly rely on the fault code to instigate a particular action without confirmation. In many cases this is the step that currently takes the longest to perform manually and consequently has significant room for improvement. Furthermore

there is an issue of confidence in diagnosis, i.e. how much certainty must be present to initiate a corrective action? If the initial diagnosis is incorrect this can lead to undesirable situations such as 'good' components being unnecessarily removed or bypassed. Both steps 3 and 4 are important as they differentiate the self-repairing strategy proposed to designed in characteristics such as 'robustness' which act only to mitigate damage rather than diagnose and repair it.

For the 4-bar linkage mechanism, the confirmation of the diagnosis could be as simple as manipulating one of the controllable elements to determine if the diagnosis is correct. If it is found to be incorrect then it must revisit the initial diagnosis and choose from the next most probable point of failure.

Corrective action

The difference between self-healing and self-repair as discussed is open to interpretation but it is assumed here that the primary difference can be defined by the action that takes place to bring the system back to operation. In essence a self-repairing system is capable of fixing a given fault to continue satisfactory operation, whereas a self-healing system has the ability to physically bring itself back to an initial state after a fault has occurred [5].

A simple example of a self-repairing strategy is having adaptable redundant components that are able to alter their function to stand-in for whichever component is diagnosed as at fault. This concept of 'self-repairing through self-reconfiguration' does not necessarily require additional redundant materials; instead performance could be sacrificed to ensure continued functionality utilizing only the currently available resources. This approach would use degenerate modules that have the ability to perform the same function or yield the same output even if they are structurally different [8].

If it is assumed there is a failure in an element then it can be reasonably assumed that we don't wish to affect these further. It is preferable therefore to change only the attached 'tool' – in this

case a simple pen designed to draw a particular pattern. Hence, once the system has been remodelled with revised dimensions, it can then attempt to return back to original desired path through changing dimensions of tracer element (t_1 and t_1 in Figure 1). An alternative approach would be to have a system with open variables (through the use of linear actuators for example) that would enable the system to alter the mechanism dimensions such as the link lengths. Whilst this would reduce the innate reliability of each component it would allow far more control over the desired output and therefore corrective action. The results in the following section will analyse both approaches.

Problem domain and case studies

Damage and subsequent faults or failure can occur in complex mechanical systems from a number of sources and result in a varying degree of loss of function or characteristic change in expected output. In the 4-bar linkage mechanism described in the previous sections this failure or loss of function can result from the consequence of failure in one or more of the linkage bars that make up the system. The self-repairing process outlined previously consists of many steps before a corrective action is even taken, each one important to the success of the overall operation. As a result a number of case studies are outlined which discuss and demonstrate how a number of these steps can be undertaken for a 4-bar linkage mechanism. Firstly the need to diagnose the fault within the system using the information available, in this instance inferring the point of failure from the current damaged tracer path. The second case study focuses upon confirmation of this diagnosis, while the final case study looks at a series of solutions for implementing some form of corrective action to regain system function.

Case study: Diagnosis

Before any corrective action can be taken it is important to be able to gauge the extent, degree and position of damage. Under the assumption the system has detected damage through information on the change in the original tracer pattern it can begin to infer the source of the failure. Outlined in Table 1 are the base dimensions for the 4-bar linkage mechanism along with the post failure values, in this case a simple reduction in l_2, whilst a comparison of the base and failed tracer paths is given in Figure 2.

Table 1. 4-Bar mechanism dimensions and bounds

Link	*Base Dimensions*	*Post-failure Dimensions*	*Lower Bounds*	*Upper Bounds*
l_0	3	3	1.0	3.0
l_1	5	5	3.0	5.0
l_2	6	5.75	3.0	6.0
l_3	7	7	4.0	7.0
t_1	3	3	1.0	3.0
t_2	3	3	1.0	3.0

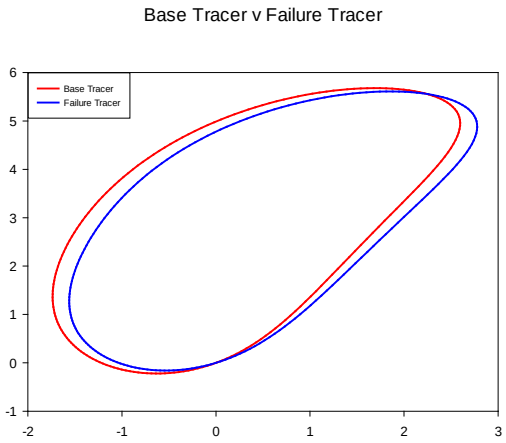

Figure 2. Comparison of 4-bar tracer mechanism paths for initial (red) and damaged (blue) tracer

In order to diagnose the source of the fault (i.e. which element is responsible), one solution is to replicate the new failed tracer path through an incremental change of the linkage length values that make up the entire 4-bar linkage mechanism. In essence the self-repairing process can explore the design space of the 4-bar mechanism to try and derive a solution which matches the current failed tracer path and infer the current dimensions as a result. This assumes that there is a one to one matching between the current failed tracer path and its dimensions and that there exist no other possible dimensions which could give rise to the same failed tracer path (i.e. every tracer path has a single, unique set of element dimensions that will produce it).

One class of algorithm capable of searching such a design space are those that utilise local gradient-based optimisation, such as the Nelder-Mead simplex algorithm [9]. Using local information taken from a single solution or design it is capable of deriving a search direction and generating new solutions that may prove to be more optimal. To demonstrate the process of diagnosis the Nelder-Mead algorithm was applied to the base design in order to diagnose the source of the original fault that gives rise to the current failed tracer path. Using a single solution over the course of 1500 functional evaluations, the Nelder-Mead algorithm acts upon the all the open design

variables within Table 1 to find a solution (design dimensions) which matches the failed tracer path. Framed as a simple optimisation problem, in order to evaluate each new design a simple objective looks to minimize the deviation between the current failed tracer path and the new solution tracer path. This is calculated as the average of the Euclidean distance between each point on the tracer path at intervals of 1° changes in the input angle α, as shown in Figure 3.

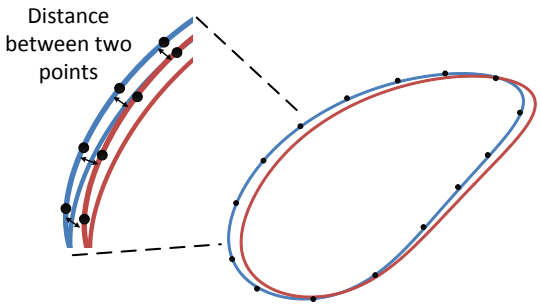

Figure 3. Euclidean distance between tracer path points

In addition a single constraint is used to ensure the tracer element lengths remain in contact, both objective and constraint information is listed in Table 2. The final solutions dimensions are shown in Table 3 and match the failed dimensions exactly showing the ability for the algorithm to diagnose what is the source of failure in this instance.

Table 2. Diagnosis - objectives and constraints

Objective/Constraint	*Type*	*Expression*
Total Tracer Error	Minimize	$\sum_{360}(B_x - S_x)^2 + (B_y - S_y)^2$
Constraint	$\geq$	$(S_{t1} + S_{t2}) - S_{l1}$

Table 3. Diagnosis – dimensions and objective values

Variable Name	*Target*	*Variable Value*
l_0	3	2.999999
l_1	5	5.000001
l_2	5.75	5.750003
l_3	7	6.999999
t_1	3	2.999999
t_2	3	2.999999
Tracer Error	0	0.000119

Case study: Confirmation of diagnosis

The confirmation of diagnosis a critical step in the self-repairing process as any changes to the systems open variables as a result of the corrective action must not further degrade the systems function. There is also the possibility that error is simply carried over and any information from the previous steps passed on or utilised during corrective may result in the wrong action being taken. If an assessment from the previous diagnosis step was incorrect and future corrective actions alter variables, particularly those deemed as damaged or failed, this could lead to further system failure.

In the simple case-study used in this paper with only a single point of failure the algorithm was able to find the precise single solution on every occasion. With more advanced applications such as robotics or multiple simultaneous point of failure this is not always the case. Furthermore depending on the computational power and time available the algorithm can provide a close, but not precise predictions for the failed dimensions. In these situations a trade-off has to be made between accuracy and computational demand, and if an inaccurate prediction is used then the system will have to confirm the result before attempting any correction.

To simulate this step the system was artificially provided with the wrong failure point. It then systematically manipulated one of its controllable elements to determine if the diagnosis was correct. Ideally in this situation the system would not have to complete a full motion range check to confirm the diagnosis so only 10° of rotation of the input link were allowed. Even with this partial knowledge the algorithm was able to determine the initial diagnosis was incorrect and it returned to

the diagnosis stage to seek another solution. The use of a quarantine or tabu list [10] prevented the algorithm from suggesting the same solutions again. By utilising this step and only partial range checking, further damage to the system can be minimised or eliminated.

Case study: Corrective action

In order to facilitate some level of self-repairing and regain full or partial function the system has to undergo some level of corrective action. Whilst this builds on and requires information learned from the previous steps, it is the step that is perhaps of most interest. The corrective action can either be deterministic and often integrated into the system at design time, acting almost reactively to the damage or fault that has occurred, or it can be a non-deterministic, online process. In a non-deterministic process the response or corrective action is directed as a result of information gathered both currently and from the past while some degree of intelligence or heuristic is utilizes this information to form what is hoped to be an optimal response. For example let us look at a simple degradation of function from the result of damage to one of the linkage bars (in this instance l_2). A reduction of its size from its base length leads to a change in the tracer path as demonstrated in previously in Figure 2. In order to facilitate some level of self-repairing and regain full or partial function the system has to undergo some level of corrective action. One possible course of action is in the alteration of the physical linkage bars that make up the 4-Bar linkage mechanism, so as to alter the current damaged state into a future state which exhibits better functional performance.

In Figure 4 an example is shown of a mechanism that can alter the tracer elements dimensions and alter its output tracer path. By employing an algorithm or heuristic which can calculate the optimal reconfiguration of the dimensions of this mechanism it is possible to return back the original function of the system, in this case our 4-bar mechanism tracer path. The algorithm chosen to perform this action is a once again the simple local optimisation Nelder-Mead Simplex method, acting upon the open variables described in Table 4 under the same objectives and constraints used in the previous steps in Table 1 and run over 1000 functional evaluations.

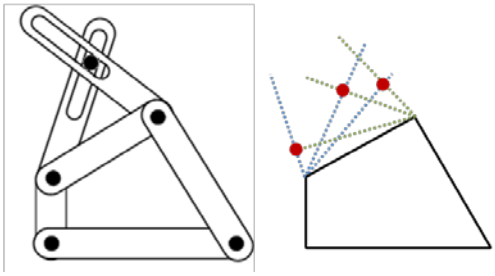

Figure. 4. Simple mechanism for varying effective tracer element lengths

Table 4. Corrective action - 4-Bar mechanism dimensions and bounds

Link	*Base Dimensions*	*Post-failure Dimensions*	*Lower Bounds*	*Upper Bounds*
l_0	3	3	Fixed	Fixed
l_1	5	5	Fixed	Fixed
l_2	6	5.75	Fixed	Fixed
l_3	7	7	Fixed	Fixed
t_1	3	3	1.0	5.0
t_2	3	3	1.0	5.0

The outcome of such an approach can be seen in both Figures 5 and 6 where the algorithm is able to produce a near optimal reconfiguration of the tracer path. In Figure 5 the solution tracer path is indistinguishable from the original base tracer path as a result of the dimensional changes made by the algorithm on t_1 and t_2 shown in Table 5.

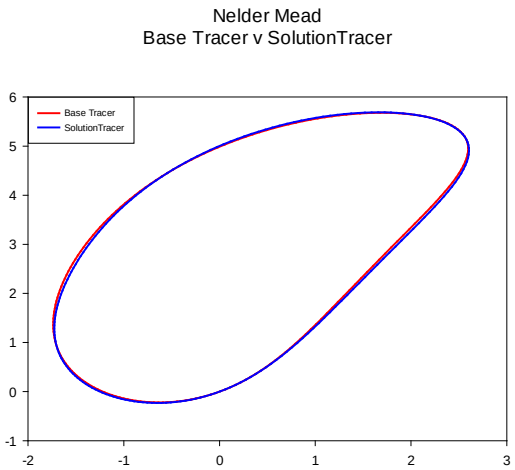

Figure 5. Corrective action - Comparison of 4-bar mechanism tracer paths for base (red) and Nelder-Mead fixed solution (blue)

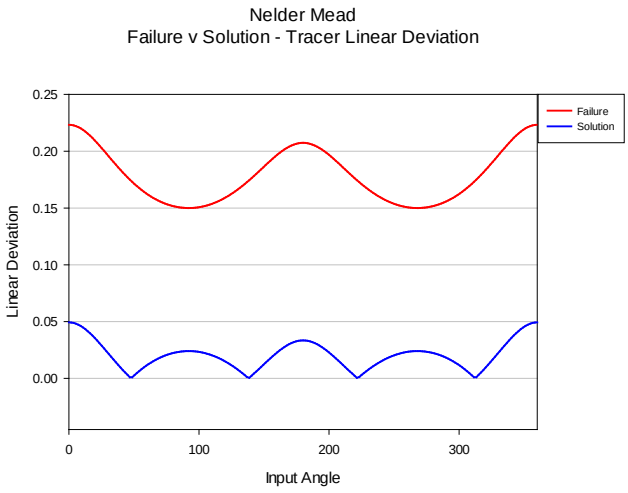

Figure 6. Corrective action - Comparison of 4-bar mechanism tracer linear deviation for failure (red) and Nelder-Mead fixed solution (blue)

Table 5. Corrective action - dimensions and objective values

Variable Name	Variable Value
l_0	3.0
l_1	5.0
l_2	5.75
l_3	7.0
t_1	2.999975
t_2	3.162831
Tracer Error	4.064422

Sometimes alternative objectives are required in order to maintain some degree of functionality even if it is different from the original function. For example if damage to the 4-bar mechanism were to occur in a different linkage element, in this case l_0 as seen in Table 6 then a tracer shape may arise that the self-repairing process is not able to return to its original path.

Table 6. Corrective action - 4-Bar mechanism dimensions and bounds

Element	*Base Dimension*	*Post-failure Dimension*	*Lower Bound*	*Upper Bound*
l_0	3	2	Fixed	Fixed
l_1	5	5	3.0	7.0
l_2	6	6	3.0	9.0
l_3	7	7	4.0	10.0
t_1	3	3	1.0	5.0
t_2	3	3	1.0	5.0

Shown in Figure 7 a comparison between the standard base tracer path and the failed or damage tracer path is considerably larger then our previous examples. Running our standard Nelder-Mead algorithm using the values in Table 6 to derive a corrective action based upon the same tracer error objectives unfortunately does not yield a very optimal solution as seen in Figure 8, even when we open up more variables to alteration.

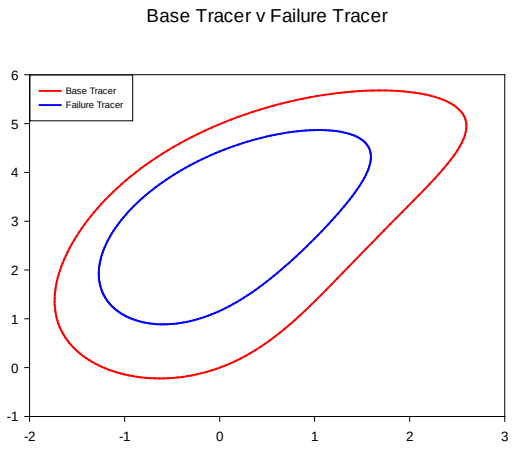

Figure 7. Corrective action - Comparison of 4-bar tracer mechanism paths for base (red) and damaged (blue) for trajectory example

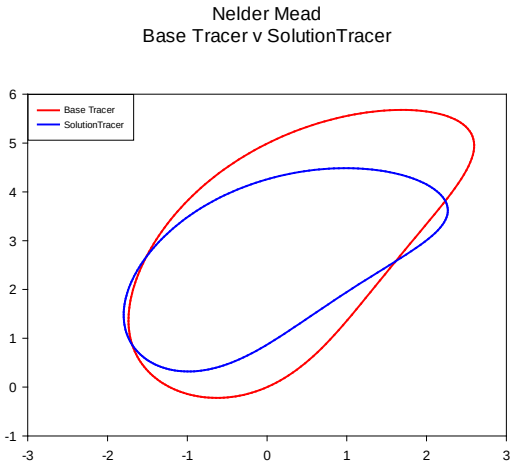

Figure 8. Corrective action - Comparison of 4-bar tracer mechanism paths for base (red) and Nelder-Mead solution (blue) tracer for trajectory example

A different perspective on the overall aim of the corrective action could provide a separate solution that to some degree matches the original function shape but not position. If it is not possible to regain the original tracer path then perhaps it may be possible to replicate its original trajectory shape. Shown in Figure 9 is a new objective for the self-repairing process, to replicate the original tracer trajectory shape regardless of whether it has been translated along one of the axes or has shrunk or grown in ratio to the original tracer path.

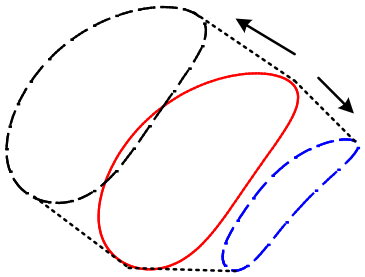

Figure 9. Comparison of tracer path trajectory either through translation or altered ratio / size

The objective becomes the simple task of minimizing the change of deviation from one input angle to the next for all input angles sampled as shown in Table 7. Running the Nelder-Mead algorithm using the values in Table 5 again but with this new objective yields the following solution in Figure 10 and its linear deviation in Figure 11.

Table 7. Corrective action - trajectory objective

Objective	*Type*	*Expression*
Total Trajectory Error	Minimize	$\sum_{360}^{n} Abs\left(\left((B_x - S_x)^2 + (B_y - S_y)^2\right)_n - \left((B_x - S_x)^2 + (B_y - S_y)^2\right)_{n-1}\right)$

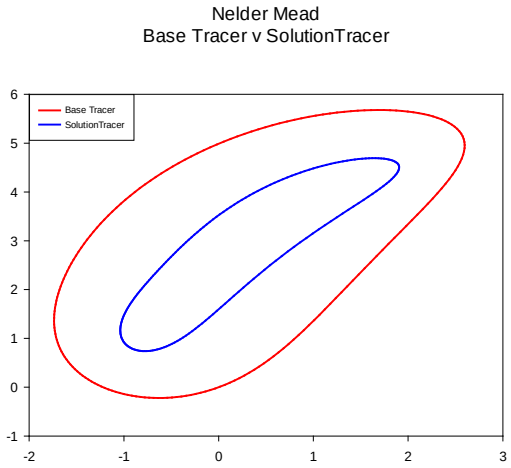

Figure 10. Corrective action - Comparison of 4-bar tracer mechanism paths for base (red) and Nelder-Mead solution (blue) for trajectory example

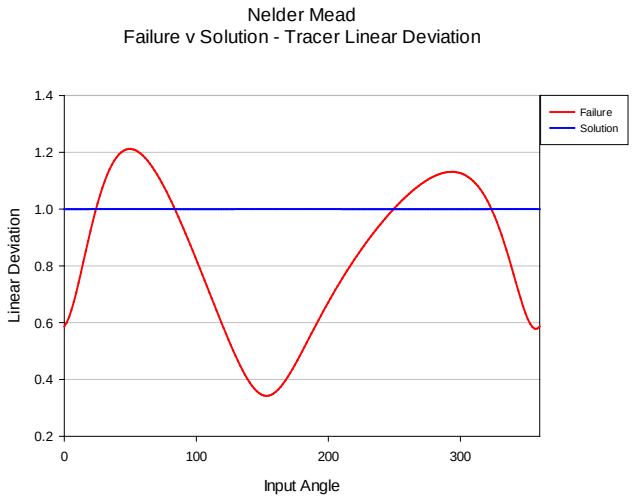

Figure 11. Corrective action - Comparison of 4-bar tracer mechanism linear deviation for failure (red) and Nelder-Mead solution (blue) for trajectory example

Though containing a higher tracer error its trajectory is the same shape as the original with each point having the same deviation as the next. The dimensions and objective values are shown in Table 8.

Table 8. Corrective action - dimensions and objective values for trajectory example

Variable Name	*Variable Value*
l_0	2.0
l_1	3.333324
l_2	3.999967
l_3	4.666657
t_1	2.999998
t_2	1.855946
Trajectory Error	2.794213E-05

The examples given previously focus upon a single objective, the need to regain the original tracer path function. However there may be situations within a particular system that require multiple objectives to be considered when developing the optimal corrective action. For example in our 4-bar mechanism the system may wish to account for future damage or faults in conjunction with regaining the original tracer path. Therefore it may wish to reduce the amount of reconfiguration (dimension change) placed upon the system, while reducing tracer path error. Conversely the system may wish to act upon two functional objectives, for example our tracer and trajectory objectives. This is described as a multi-objective problem and is often tackled in one of two ways, the first to combine all objectives into a single weighted 'sum' objective function, or to utilise an algorithm which houses multiple solutions or often described as a population of solutions which can be used to form a Pareto set. An example of such an algorithm can be found in the field of evolutionary computation, in a heuristic called NSGAII [11].

This multi-objective genetic algorithm exploits the concept of Pareto optimality in order to partition the population of solutions into a number of ranked sets. This Pareto ranking of a population set works by utilising Pareto dominance to define a set of solutions which either

dominate or are equal to all other solutions for each objective within the design problem. The first set to meet these criteria is given a rank and is defined as the Pareto optimal set. The process is repeated for the remaining solutions within the population until all ranks are filled as shown in Figure 12. Here the two objectives f_1 and f_2 are often antagonistic and work against each other, for example performance against cost.

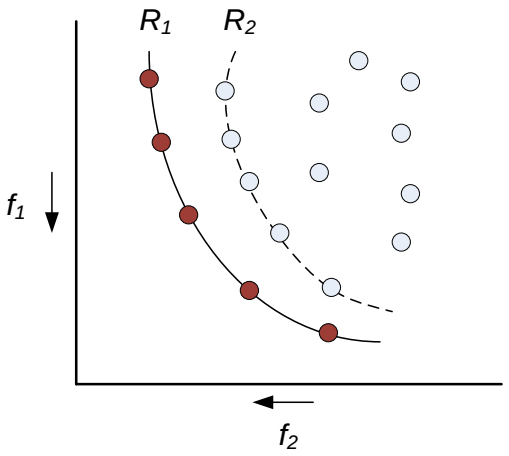

Figure 12. Pareto ranking within a set of solutions

Applying the multi-objective algorithm to the 4-bar mechanism under the same conditions as described in table 6 but with both the tracer error and trajectory error objectives allows for a choice of solutions to apply for corrective action. Using the parameters outlined in Table 9, over 5 separate trials NSGAII was able to produce an optimal Pareto set of solutions the best of which is shown in Figure 13 produced by the first trial.

Table 9. Default Algorithm Parameters NSGAII

NSGAII Parameter	Default Value
Population Size	100
Offspring Size	100
Selection Size	100
Replacement Size	200
SBX Distribution Index	20
Polynomial Mutation Distribution Index	20
Probability of SBX Crossover	0.8
Probability of Mutation	0.1
Generations	100
Tests	5

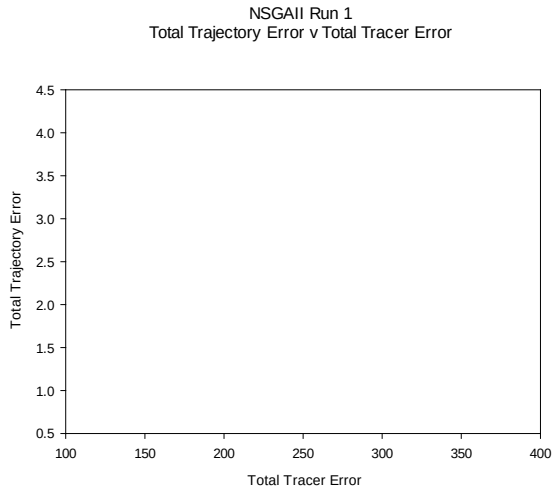

Figure 13. Corrective action – Final population set for NSGAII run 1

An example of the best solution found for a specific objective is described in Table 10 for the tracer error objective and Table 11 for the trajectory objective. The set of solutions provided give the self-repairing process access to a number of alternative solutions however in this instance there is a loss in optimality. Looking at the best solution found regarding the trajectory objective it is apparent that it is much worse than the solution found using the single objective Nelder-Mead algorithm shown in Table 8. This is perhaps not surprising considering the heuristic NSGAII is designed to be more of a global search algorithm providing good but perhaps not optimal solutions, while the local optimisation method is able to significantly improve upon solutions that are found within a local search space. Future work could investigate how to utilise both algorithms in order to gain the ability for more global search and a set of solutions with more powerful local search from the Nelder-Mead algorithm.

Table 10. Corrective action - NSGAII Best Result for Tracer Error Run 1

Variable Name	Variable Value
l_0	2.0
l_1	4.589180
l_2	3.021640
l_3	4.345781
t_1	3.371373
t_2	2.178347
Tracer Error	0.892857
Trajectory Error	364.2624

Table 11. Corrective action - NSGAII Best Result for Trajectory Error Run 1

Variable Name	Variable Value
l_0	2.0
l_1	4.931317
l_2	4.147293
l_3	4.426633
t_1	2.664856
t_2	3.371498
Tracer Error	3.871128
Trajectory Error	128.0485

Conclusions

Even with extensive planning and fault analysis it is practically impossible to create a perfectly reliable machine. Current approaches to improving reliability are largely based around advancing areas of modelling and detection to include specific methods designed to overcome particular failure modes. Although this has led to an increase in the operational life of a system and hence improved reliability, it is becoming more and more difficult to predict and mitigate against all possible failure modes. Hence rather than focusing on specific failure modes a new philosophy is proposed that allows systems to reconfigure themselves to overcome both expected and unexpected failure but focusing on how a failure manifests rather than the failure itself. This approach has been demonstrated in the design of a self-rectifying 4-bar linkage mechanism. This was achieved by breaking the self-repairing process into five individual steps that can be applied to any system, with three of these steps being analysed in more detail.

During this analysis a number of potential issues were encountered:

- Step 0: Cause of Fault. Whilst the cause of fault shouldn't be of primary concern, it is important to investigate common failure modes as a starting point for subsequent steps. Furthermore it is important to focus on a system (or sub-system) that has homogeneity between elements. The rectification of a failed motor for example would be vastly

different from the rectification of a rigid element. If a system has non-homogenous elements then it should be broken down into sub-systems and each one assessed individually.

- Step 1: Detection of Fault. Perhaps the simplest way of detecting a fault is through user intervention however this is not always feasible and somewhat defeats the objective of having a fully autonomous system. The ideal solution is to continuously monitor the desired output of a mechanism or machine to observe any deviation from the status quo. This information can subsequently be used to make an assessment of the damage and monitor the effect of any corrective action.
- Step 2: Diagnosis of Fault. There are several options that can be used to diagnose a fault including probabilistic, model-based or case-based methods. The choice of method will depend on the pre-existing knowledge and complexity of the system. In general model-based can be preferable but this can also leads the designer to only focus on particular expected modes of failure. It is better therefore to infer the most likely failure mode from a change in output behaviour.
- Step 3: Confirmation of diagnosis. In simple application or those with known, precise diagnoses this step can perhaps be ignored. However in complex systems or those with limited computational resources it is critical to ensure that any diagnosis made is correct to avoid damaging the system or output further. In these situations rather than seeking an fully confirmed diagnosis, a suspect diagnosis can be used as a starting point to test a possible corrective action. If the system behaves as expected than the diagnosis is confirmed, else the algorithm must tabu the proposed diagnosis and seek another.
- Step 4: Corrective Action. The corrective action is perhaps of most interest to designers as it alone ultimately will govern the extent to which a system is able to recover. Where possible it is preferable to avoid changes that fundamentally alter the basic system mechanism. For example in the 4-bar mechanism it is preferable not to alter the rigid

link elements and instead focus on changing the tracer element dimensions. This alone might be sufficient to return the system to near perfect working order, or alternatively as shown in the results, it may be necessary to further alter the system to maintain functionality. Furthermore it is important to determine what the desired output should be. The results show that if reduced or translated functionality is desired (similar to a 'limp home' mode), then a greater degree of failure can be overcome. Maintaining perfect functionality whilst only manipulating some elements within a system is not always possible an a trade-off has to be made.

Although applied to a simple 4-bar mechanism, the proposed process could be applied to a number of different applications including robotics. In this instance a manipulator or end effector could develop a fault that limits its motion. By determining the precise limit of motion, the system could determine the most likely point of failure (diagnosis), and then adapt itself stochastically to confirm this diagnosis. Finally it would utilise other elements within itself to adapt and maintain as wide an operating envelope as possible.

Systems with additional self-repairing mechanisms will inevitably be more complex and hence can become intrinsically less 'reliable', even though it has the ability to bring itself back to normal operating conditions. However in these situations one must view the system from the perspective of the end-user: a system with an integral resilience-mechanism would appear to be more 'reliable' – it is able to maintain operation for a longer period of time than would otherwise have been possible – and this should be the primary aim of any self-repairing system.

Acknowledgements

The authors would like to thank the EPSRC Centre for Innovative Manufacturing in Through-life Engineering Services (EPSRC Reference: EP/I033246/1) for funding this research.

References

1. Bongard J, Zykov V, and Lipson H. Resilient machines through continuous self-modeling *Science*, 2006, 314(5802), pp.1118-1121.

2. Thrun S, Burgard W, and Fox D. *Probabilistic Robotics*. MIT Press, Cambridge, MA, USA. 2005.

3. Bell C, Farnsworth M, Tiwari A, and Dorey R.. Theoretical Design of a Self-rectifying 4-bar Linkage Mechanism. Procedia CIRP, 2013, 11, 385-389.

4. Amor-Segan ML, McMurran R, Dhadyalla G, and Jones RP. Towards the Self-Healing Vehicle. *3rd Institution of Engineering and Technology Conference*, IET, pp. 1-7, June 2007

5. Bell C, McWilliam R, Purvis A, and Tiwari A. Concepts of Self-Repairing Systems. *Measurement and Control*, 2013. 46(6), 176-179.

6. Zinn M. Lock-up failure of a four-bar linkage deployment mechanism. In *27th Aerospace Mechanisms Symposium* (Vol. 1, pp. 283-298), 1993, May

7. Desai A, and Mital A. Design simplification and innovation through adoption of a time-based design for maintenance methodology. *Human Work Productivity: A Global Perspective*, 2013, 89.

8. Edelman GM, and Gally JA. Degeneracy and complexity in biological systems. *Proceedings of the National Academy of Sciences*, 2001. 98(24), 13763-13768.

9. Powell MJD. On Search Directions for Minimization Algorithms. *Mathematical Programming 4*. 1973. 193–201. doi:10.1007/bf01584660.

10. Glover F. Future Paths for Integer Programming and Links to Artificial Intelligence. *Computers and Operations Research*. 1986. 13 (5): 533–549. doi:10.1016/0305-0548(86)90048-1.

11. Deb K, Agrawal S, Pratap A, and Meyarivan T. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II". In *Proceedings of the 6th International Conference Parallel Problem Solving from Nature (PPSN-VI)*. 2000. pp 849-858.

12. Frei, R., McWilliam, R., Derrick, B., Purvis, A., Tiwari, A., Di Marzo Serugendo, G., 2013 “Self-healing and self-repairing technologies”, International Journal of Advanced Manufacturing Technology (JAMT), Springer. DOI 10.1007/s00170-013-5070-2

13. Williams, H. R., R. S. Trask, and I. P. Bond. "Self-healing sandwich panels: restoration of compressive strength after impact." *Composites Science and Technology* 68, no. 15 (2008): 3171-3177.

14. Kessler, M. R. "Self-healing: a new paradigm in materials design." *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering* 221, no. 4 (2007): 479-495.

15. Gu, P., D. Xue, and A. Y. C. Nee. "Adaptable design: concepts, methods, and applications." *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture* 223, no. 11 (2009): 1367-1387.

16. Chow, E., and Alan S. Willsky. "Analytical redundancy and the design of robust failure detection systems." *Automatic Control, IEEE Transactions on* 29, no. 7 (1984): 603-614.

17. Kawabata, Kuniaki, Shinnosuke Okina, Teruo Fujii, and Hajime Asama. "A system for self-diagnosis of an autonomous mobile robot using an internal state sensory system: fault detection and coping with the internal condition."*Advanced Robotics* 17, no. 9 (2003): 925-950.

18. Zhang, Youmin, and Jin Jiang. "Bibliographical review on reconfigurable fault-tolerant control systems." *Annual reviews in control* 32, no. 2 (2008): 229-252.

19. Lai, Xiongming, Yong Zhang, and Cheng Wang. "A new robust design method for path-generating linkages with multi-stochastic noises." *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*(2014): 0954405414534435.