Developing a fuzzy proportional-derivative controller optimization engine for engineering design optimization problems

Hsu, Yeh-Liang and Liu, Tzu-Chi (2007) "Developing a fuzzy proportional-derivative controller optimization engine for engineering design optimization problems", Engineering Optimization, 39:6, 679 - 700

URL: http://dx.doi.org/10.1080/03052150701252664

© Taylor and Francis 2007
Developing a fuzzy proportional–derivative controller optimization engine for engineering design optimization problems

YEH-LIANG HSU* and TZU-CHI LIU

Department of Mechanical Engineering, Yuan Ze University, 135 Yuan-Tung Road, Chungli, Taiwan

(Received 28 September 2006; in final form 17 January 2007)

In real world engineering design problems, decisions for design modifications are often based on engineering heuristics and knowledge. However, when solving an engineering design optimization problem using a numerical optimization algorithm, the engineering problem is basically viewed as purely mathematical. Design modifications in the iterative optimization process rely on numerical information. Engineering heuristics and knowledge are not utilized at all. In this article, the optimization process is analogous to a closed-loop control system, and a fuzzy proportional–derivative (PD) controller optimization engine is developed for engineering design optimization problems with monotonicity and implicit constraints. Monotonicity between design variables and the objective and constraint functions prevails in engineering design optimization problems. In this research, monotonicity of the design variables and activities of the constraints determined by the theory of monotonicity analysis are modelled in the fuzzy PD controller optimization engine using generic fuzzy rules. The designer only needs to define the initial values and move limits of the design variables to determine the parameters in the fuzzy PD controller optimization engine. In the optimization process using the fuzzy PD controller optimization engine, the function value of each constraint is evaluated once in each iteration. No sensitivity information is required. The fuzzy PD controller optimization engine appears to be robust in the various design examples tested.

Keywords: Design optimization; Monotonicity analysis; Fuzzy control

1. Introduction

Engineering design optimization problems often have two characteristics. First, monotonicity between design variables and the objective and constraint functions prevails in engineering design optimization problems. The theory of monotonicity analysis (Papalambros and Wilde 2000) was developed to analyse this type of problem to identify which constraints must be active at the optimum design point. However, numerical optimal solution cannot be obtained by monotonicity analysis alone.

Secondly, most real world engineering design problems display complex phenomena, and the objective and constraint functions often cannot be expressed analytically in terms of design variables. These so-called ‘implicit functions’ are evaluated by computer simulation or physical

*Corresponding author. Email: mehsu@saturn.yzu.edu.tw
experiments, which are usually the major cost of the optimization process. Moreover, the analytical forms of the first derivatives of the implicit functions are often not available. This fact hinders the use of formal numerical optimization techniques to solve these engineering design optimization problems.

An iterative optimization process can be analogous to a closed-loop control system. Figure 1 shows the block diagram of a closed-loop control system. The measured response of the system process being controlled is fed back and compared with a desired response. The control actions generated by the controller are determined in an attempt to fix the error between the system response and the desired response.

Figure 2 shows a block diagram for an optimization process. Initial parameters are input to the optimization algorithm, which in turn generates a trial design point according to its search rules. The optimization model is then evaluated at this trial design point, and information such as objective and constraint function values and sensitivities is fed back to the termination test. If the termination test fails, the optimization algorithm is triggered again to generate the next design point, using the numerical information from previous iterations. Finally, while a control system attempts to achieve a stable predefined output, the optimization process pursues a converging objective function value. Comparison of figures 1 and 2 shows that an optimization model in an optimization process is analogous to the system process in a control system, while an optimization algorithm is analogous to the controllers.

Traditional numerical optimization algorithms are analogous to direct digital controllers. The algorithms are usually ‘crisply’ designed for well-defined mathematical models, and their numerical rules for generating the next design point are exact and definite. In real world engineering design problems, decisions for design modifications are often based on engineering heuristics and knowledge. However, when solving an engineering design optimization problem using a numerical optimization algorithm, the engineering problem is basically viewed as a pure mathematical problem. Design modifications in the iterative optimization process rely on numerical information. Engineering heuristics and knowledge are not utilized at all.

This suggests that, in addition to crisp numerical rules, engineering heuristics and knowledge should also be modelled in an optimization algorithm using fuzzy rules. As suggested in figure 2, the ‘controller’ in the optimization process may as well be a fuzzy controller!

In this research, a fuzzy proportional–derivative (PD) controller optimization engine is developed to deal with general engineering design optimization problems with monotonicity and implicit functions. In particular, monotonicity of the design variables and activities of
the constraints determined by monotonicity analysis are modelled in the fuzzy PD controller optimization engine using generic fuzzy rules. The structure of an optimization algorithm is still maintained to guide the engineering decision process.

The remainder of the article is organized as follows. In section 2 the concept of the fuzzy PD controller optimization engine is discussed further. The process of using the fuzzy PD controller optimization engine is described in section 3, and in section 4 the process is illustrated with an example. In section 5, the fuzzy PD controller optimization engine is used to solve several engineering design optimization examples commonly seen in literature. Finally, section 6 concludes the article.

2. The concept of the fuzzy PD controller optimization engine

2.1 Application of fuzzy theory in optimization problems

Fuzzy theory is primarily concerned with quantifying and reasoning using natural language in which many words have ambiguous meanings. Since the introduction of the basic theory of fuzzy sets by Zadeh (1965), fuzzy theory has been extended and applied to many different fields, including engineering design optimization.

The application of fuzzy theory in optimization problems can be categorized in two ways.

(i) Using fuzzy theory to formulate the uncertainty of the optimization model: Bellman and Zadeh (1970) introduced the fuzzy-set-based optimization of decision-making in a fuzzy environment which includes the concept of fuzzy constraint, fuzzy objective, and fuzzy decision. For fuzzy optimization, the objective and constraint functions are characterized by the membership functions in a fuzzy system. The decision can be viewed as the intersection of the fuzzy objective and constraint functions. Once the membership functions are known, the optimization problem can be viewed as a crisp optimization problem. Different kinds of mathematical models have been proposed to solve fuzzy optimization problems in various engineering fields. Rao et al. (1992) and Xiong and Rao (2003, 2005) applied the concept of fuzzy optimization to the design optimization of mechanical and structural systems. Guan et al. (1995) described the application of a fuzzy optimization technique to optimal power flow calculations. Sarma and Adeli (2000) presented a fuzzy discrete multi-objective optimization model for space steel structure design, subject to constraints of commonly used design codes. Inuiguchi and Ramik (2000) reviewed some fuzzy optimization methods and techniques from a practical point of view.

(ii) Implementing fuzzy theory into the optimization algorithms: in this research area, fuzzy theory is used to adjust or control the parameters of the numerical optimization algorithm such that engineering knowledge and human supervision process is integrated into the optimization process. Arakawa and Yamakawa (1990) demonstrated an optimization method using qualitative reasoning, which makes use of qualitative information to give an approximate direction of the optimum search. Trabia and Lu (2001) proposed a fuzzy adaptive simplex search optimization algorithm to minimize a function of \( n \) variables. This method uses fuzzy logic to decide the next move of the simplex. Hsu et al. (1995a, b) proposed a fuzzy algorithm for determining the move limits in a sequential linear programming algorithm. Arabshahi et al. (1996) pointed out that many optimization techniques involve parameters that are often adapted by the user through trial and error, experience, and other insights. They applied neural and fuzzy ideas to adaptively select these parameters. Mulkay and Rao (1998) proposed a modified sequential
linear programming algorithm using fuzzy heuristics to control the optimization parameters. Other researchers have tried to develop intelligent optimization algorithms based on fuzzy theory to solve optimization problems. Xiong and Rao (2003) combined fuzzy $\lambda$-formulation with a hybrid genetic algorithm to solve the mixed-discrete design optimization problem. Mukuda et al. (2005) combined a fuzzy logic controller with a hybrid genetic algorithm and a local search technique to solve the multi-objective optimization problems.

The research presented in this article falls into the second category. In implementing fuzzy theory in optimization algorithms, most research has focused on implementing optimization process knowledge into the numerical optimization algorithms using fuzzy theory. Few researchers have placed emphasis on implementing engineering knowledge to develop optimization algorithm specifically for solving certain types of engineering optimization problems.

### 2.2 Fuzzy PD controller

Fuzzy control is similar to the classic closed-loop control approach, but differs in that it substitutes imprecise symbolic notions for precise numerical measures. The fuzzy controller takes input values from the real world. These crisp input values are mapped to the linguistic values through membership functions in the fuzzification step. A set of rules which emulates the decision-making process of the human expert controlling the system is then applied using certain inference mechanism to determine the output. Finally, the output is mapped into crisp control actions required in practical applications in the defuzzification step.

In the closed-loop control system shown in figure 1, the measured response of the system process being controlled is fed back and compared with the desired response. The control actions generated by the controller are determined in part by the system response in an attempt to fix the error. The output of a proportional controller is a control signal $u$ which is proportional to the error $e$:

$$u = K_p e$$  \hspace{1cm} (1)

where $K_p$ is the gain.

Derivative control is used to anticipate the future behaviour of the error signal by using corrective actions based on the rate of change in the error signal. The output of a derivative controller is a control signal $u$ which is proportional to the derivative of error:

$$u = K_d \frac{de}{dt}$$  \hspace{1cm} (2)

where $K_d$ is the gain.

The control action in a PD controller is described by

$$u = K_p e + K_d \frac{de}{dt}$$  \hspace{1cm} (3)

which combines proportional and derivative control modes. The PD controller makes a control loop respond faster and with less overshoot, and is by far the most popular method of control.

The fuzzy counterpart of the PD controller also has two inputs: system error $e$ and error change $\Delta e$. Fuzzy inference is used to compute the control signal $u$. Table 1 shows a typical rule base for a fuzzy PD controller with 25 rules. Five linguistic terms are used for each variable, NB (negative big), NS (negative small), ZE (zero), PS (positive small), and PB (positive big). For example, the fifth rule in table 1 (row 1, column 5) states: ‘IF $e$ is NB AND $\Delta e$ is PB, THEN the control action is NS’.
Table 1. A typical rule base for a fuzzy PD controller (25 rules).

<table>
<thead>
<tr>
<th>Δe</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td>PB</td>
<td>PB</td>
</tr>
<tr>
<td>PS</td>
<td>PB</td>
</tr>
<tr>
<td>ZE</td>
<td>PS</td>
</tr>
<tr>
<td>NS</td>
<td>PS</td>
</tr>
<tr>
<td>NB</td>
<td>PS</td>
</tr>
</tbody>
</table>

2.3 The concept of the fuzzy PD controller optimization engine

As discussed in the previous section, the iterative optimization process can be analogous to a close-loop control system. In previous work (Hsu et al. 2005), the fuzzy PD controller was used as the optimization engine in an optimization process to handle the specific type of objective function

\[ \min f = \left( \frac{\sum_{j=1}^{n} (y_j - Y_j)^2}{n - 1} \right)^{0.5} \]

i.e. to minimize the differences between system outputs \( y_i \) and target values \( Y_i \). As shown in figure 3, initial errors \( e \) and change of errors \( \Delta e \) are input to the fuzzy PD controller, which generates the changes of design variables \( \Delta x \) for the next iteration. Then the system inputs are updated \( x^{q+1} = x^q + \Delta x \) and the new system process outputs \( y^{q+1} \) are fed back and compared with the set point \( Y \) again.

System outputs \( y \) are functions of design variables \( x \). In engineering design optimization problems, the monotonicity of design variables in system outputs is often known empirically. This monotonicity is modelled in the fuzzy PD controller optimization engine in order to correctly update the design variables in the next iteration. The optimization process terminates when \( e \) and \( \Delta e \) approach zero, i.e. when no change in design variables \( \Delta x \) will be generated by the fuzzy PD controller optimization engine.

The objective function in equation (4) is rather restricted. In the next section, the application of the fuzzy PD controller optimization engine is extended to more general engineering optimization problems with monotonicity and implicit functions.

3. Using the fuzzy PD controller optimization engine for engineering design optimization problems

As discussed earlier, monotonicity between design variables and the objective and constraint functions prevails in engineering design optimization problems. Monotonicity is often known through engineering knowledge or experience without knowing the explicit
The idea of analysing the monotonicity of objective function and constraints was first introduced by Wilde (1975) for checking model boundedness. Papalambros (1979) then developed monotonicity analysis as a generalized systematic methodology. Monotonicity analysis seeks to identify rigorously, in advance of any extensive numerical computation, which combinations of inequality constraints can be active. The active constraints are to be satisfied with strict equality at the optimum.

There are two monotonicity principles (Papalambros and Wilde, 2000). The first monotonicity principle (MP1) states that: ‘In a well-constrained minimization problem, every (strictly) increasing (decreasing) variable is bounded below (above) by at least one active constraint’. The second monotonicity principle (MP2) deals with variables which are not in the objective function: ‘Every monotonic non-objective variable in a well-bounded problem is either (a) irrelevant and can be deleted from the problem together with all constraints in which it occurs, or (b) relevant and bounded by two active constraints, one from above and one from below’.

Figure 4 shows the conceptual structure of how to use the output of monotonicity analysis to construct the fuzzy PD controller optimization engine. In general, monotonicity analysis can be carried out by explicit algebraic substitution for the constraints that are identified as active. In this research, monotonicity analysis is done using the computer program MONO developed by Hsu (1993), which implements monotonicity analysis into an automated computerized process.

The input of MONO is a monotonicity table which contains only the monotonicity signs of the design variables with respect to the objective function and constraints. The explicit mathematical form of the optimization model is not required. In MONO, the monotonicity principles are applied to the design variables one by one. When a constraint is identified as a critical constraint by the monotonicity principles, MONO uses ‘implicit elimination’ to

Figure 4. The conceptual structure of using the output of monotonicity analysis to construct the fuzzy PD controller optimization engine.
eliminate the critical constraint using only the monotonicity signs in the monotonicity table. The monotonicity table is then updated according to the implicit elimination rules, and the size of the table is reduced. The new table is then passed to the next analysis cycle.

In some cases, the monotonicity of a design variable with respect to the objective function or certain constraint cannot be determined after implicit elimination. The monotonicity sign of the variable in the monotonicity table is marked $i$ for ‘indeterminate’. MONO terminates when all design variables have been analysed, or the monotonicity table has only $i$ in the objective function row and cannot be analysed further. Details of implicit elimination are described in Hsu (1993).

The complete output from MONO includes rigorous monotonicity analysis steps, the reduced monotonicity tables after each step, and the monotonicity analysis results. After monotonicity analysis, some of the constraints are identified as ‘critical constraints’ by MP1. These constraints have to be satisfied with strict equality at the optimum. If more than one constraint can be critical for a design variable $x$ by MP1, these constraints form a ‘conditionally critical set’ (Papalambros and Wilde 2000), i.e. at least one of the constraints in the set must be critical at the optimal design point. Some constraints are identified as ‘uncritical’ by MP1, while other constraints are identified as ‘irrelevant’ by MP2. These constraints can be temporarily eliminated from the optimization model. Finally, some constraints may remain undecided by either MP1 or MP2.

As shown in figure 4, the inactive (uncritical or irrelevant) constraints are temporarily eliminated from the optimization model. It is necessary to check the inactive constraints again after the optimal design point is obtained. The critical, conditionally critical, and undecided constraints, as well as the monotonicity signs of the design variables in the original objective function, are implemented into the fuzzy PD controller optimization engine.

There are four possible situations for implementing the analysis results from MONO into the fuzzy PD controller optimization engine.

(i) **Design variables with only one critical constraint.** For a design variable $x$ with only one critical constraint $g_k$, the change of the design variable $\Delta x$ in the next iteration is decided by the current value of $g_k$. Since the target value for $g_k$ is zero, $e$ and $\Delta e$ of constraint $g_k$ can be calculated and are converted into fuzzy membership in the fuzzification step using a ‘quantization table’ defined by the user. The definition of the quantization table will be described in detail in the next section. From the two inputs, the membership of control action contributed by constraint $g_k$ can be determined using a rule base similar to table 1. Finally, $\Delta x$ in the next iteration is calculated by:

$$\Delta x = \text{defuzzification}(\mu_{g_k}(x))$$  \hspace{1cm} (5)

where $\mu_{g_k}$ is the membership of control action contributed by the critical constraint $g_k$, and ‘defuzzification’ defuzzifies the membership function into a crisp value of $\Delta x$, again using a quantization table.

(ii) **Design variables with a conditionally critical set.** For a design variable $x$ with a conditionally critical set, at least one of the constraints in the set must be critical at the optimal design point. In this situation, the change in the design variable $\Delta x$ in the next iteration is decided by the current values of these constraints. At the current design point, if constraints $g_1, g_2, \ldots, g_i$ in the conditionally critical set are violated, $\Delta x$ is decided by

$$x = \text{defuzzification} (\max(\mu_{g_1} \cup \mu_{g_2} \cup \ldots \cup \mu_{g_i})).$$  \hspace{1cm} (6)

The max and union operators are used to pick the strongest membership from all violated constraints in order to force the design point back to the feasible domain. However, if
all constraints are satisfied, $\Delta x$ is decided by the monotonicity sign of $x$ in the original objective function. In this case, $\Delta x$ will be equal to the move limit defined by the user in the proper direction which reduces the value of the objective function.

(iii) **Design variables whose monotonicity signs are ‘indeterminate’**. For a design variable $x$ whose monotonicity sign is ‘indeterminate’ in the objective function of the final monotonicity table output by MONO, all constraints in which the variable appears can be active. Thus the change of the design variable $\Delta x$ in the next iteration is decided by the current values of these constraints in the same fashion as described in (2).

(iv) **Design variables which do not appear in the objective function**. If a design variable does not appear in the objective function, and is proved to be ‘relevant’ by MP2, all constraints in which the design variable appears can be active according to MP2. Thus, if some constraints are violated, $\Delta x$ is decided by the current values of the violated constraints in the same fashion as described in equation (6). However, if all constraints in which the non-objective variable appears are satisfied, the amount of change of the design variable $\Delta x$ for the next iteration is zero to maintain the current value of the design variable.

In the next section, an air tank design optimization example is used to illustrate the complete process of applying the fuzzy PD controller optimization engine to general engineering optimization problems with monotonicity and implicit constraints.

4. **An air tank design optimization problem**

4.1 **Monotonicity analysis using MONO**

The optimal design problem for the cylindrical air tank shown in figure 5 (Unklesbay et al. 1972; Papalambros and Wilde 2000) is considered next. The objective is to minimize the quantity of material used, which depends on the inner radius $r$, the shell thickness $s$, the shell length $l$, and the head thickness $h$. The volume of the tank has to be larger than the specified volume (constraint $g_1$), the thicknesses of the head and the wall have to satisfy the ASME code ($g_2, g_3$), and there are constraints on the size of the tank ($g_4, g_5, g_6$). The problem can be expressed as follows:

\[
\min F(r, s, l, h) = \pi [(2rs + s^2)l + 2(r + s)^2h] \\
\text{s.t. } g_1 = \frac{2.12 \times 10^7}{\pi r^2 l} - 1 \leq 0 \\
g_2 = \frac{130 \times 10^{-3}r}{h} - 1 \leq 0 \\
g_3 = \frac{9.59 \times 10^{-3}r}{s} - 1 \leq 0 \\
g_4 = \frac{10}{l} - 1 \leq 0 \\
g_5 = \frac{(r + s)}{150} - 1 \leq 0 \\
g_6 = \frac{l}{610} - 1 \leq 0
\] (7)
Figure 5. The air tank design problem.

Table 2. Monotonicity table for the air tank example.

<table>
<thead>
<tr>
<th></th>
<th>( r )</th>
<th>( s )</th>
<th>( l )</th>
<th>( h )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( F )</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>( g_1 )</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>( g_2 )</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>( g_3 )</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>( g_4 )</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>( g_5 )</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>( g_6 )</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 2 is the corresponding monotonicity table for equation (7), which is the only input required for the computer program MONO. In the monotonicity table, a plus (minus) sign indicates that the variable is monotonically increasing (decreasing) in the function, while a full point means that the variable does not appear in the function. In the air tank design optimization example, MONO concludes that constraints \( g_1 \), \( g_3 \), and \( g_2 \) are critical to the design variables \( r \), \( s \), and \( h \), respectively. The monotonicity sign of design variable \( l \) is ‘indeterminate’ in the objective function of the final monotonicity table in the output from MONO. Note that this analysis result is obtained using only the monotonicity table in table 2. Explicit formulations of the objective function and the constraints are not required. The complete monotonicity analysis of this example using explicit substitution of the active constraints can be found in Papalambros and Wilde (2000).

4.2 Preparing the fuzzy PD controller

Four simple generic rules are used to describe how to adjust the values of the monotonic design variables in order to find the design point which satisfies the critical (active) constraints with strict equality.

For monotonically increasing variables, the rules are:

- Rule I: ‘IF the constraint function value is positive, THEN the variable must be decreased’.
- Rule II: ‘IF the constraint function value is negative, THEN the variable must be increased’.

Similarly, for monotonically decreasing variables, the rules are:

- Rule III: ‘IF the constraint function value is positive, THEN the variable must be increased’.
• Rule IV: ‘IF the constraint function value is negative, THEN the variable must be decreased’.

For example, constraint $g_1$ is critical for design variable $r$, which is monotonically decreasing with respect to $g_1$. At the initial design point, if $g_1$ is positive, design variable $r$ must be increased by Rule III in the next iteration so that $g_1$ will be pushed to zero.

The function value of the critical constraint (such as $g_1$) is the errors $e$ in the fuzzy PD controller optimization engine. The error change $\Delta e$ (the change in the constraint function values) reflects the trend of the constraint function. Considering the error change, the rules can be extended as follows. For monotonically increasing variables:

• Rule Ia: ‘IF the constraint function value is positive and is increasing, THEN decrease the variable strongly’.
• Rule Ib: ‘IF the constraint function value is positive and is decreasing, THEN decrease the variable softly’.
• Rule IIa: ‘IF the constraint function value is negative and is increasing, THEN increase the variable softly’.
• Rule IIb: ‘IF the constraint function value is negative and is decreasing, THEN increase the variable strongly’.

For monotonically decreasing variables:

• Rule IIIa: ‘IF the constraint function value is positive and is increasing, THEN increase the variable strongly’.
• Rule IIIb: ‘IF the constraint function value is positive and is decreasing, THEN increase the variable softly’.
• Rule IVa: ‘IF the constraint function value is negative and is increasing, THEN decrease the variable softly’.
• Rule IVb: ‘IF the constraint function value is negative and is decreasing, THEN decrease the variable strongly’.

Similar to the typical rule base for a fuzzy PD controller in table 1, the rules discussed above are further interpreted into the fuzzy rule base in tables 3 and 4 using five linguistic terms: negative big (NB), negative small (NS), zero (ZE), positive small (PS), and positive big (PB).

<table>
<thead>
<tr>
<th>$\Delta e$</th>
<th>PB</th>
<th>PS</th>
<th>ZE</th>
<th>NS</th>
<th>NB</th>
</tr>
</thead>
<tbody>
<tr>
<td>PB</td>
<td>NB</td>
<td>NB</td>
<td>ZE</td>
<td>PS</td>
<td>PS</td>
</tr>
<tr>
<td>PS</td>
<td>NB</td>
<td>NS</td>
<td>ZE</td>
<td>PS</td>
<td>PS</td>
</tr>
<tr>
<td>ZE</td>
<td>NB</td>
<td>NS</td>
<td>ZE</td>
<td>PS</td>
<td>PB</td>
</tr>
<tr>
<td>NS</td>
<td>NS</td>
<td>NS</td>
<td>ZE</td>
<td>PS</td>
<td>PB</td>
</tr>
<tr>
<td>NB</td>
<td>NS</td>
<td>NS</td>
<td>ZE</td>
<td>PB</td>
<td>PB</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$\Delta e$</th>
<th>PB</th>
<th>PS</th>
<th>ZE</th>
<th>NS</th>
<th>NB</th>
</tr>
</thead>
<tbody>
<tr>
<td>PB</td>
<td>PB</td>
<td>PB</td>
<td>ZE</td>
<td>NS</td>
<td>NS</td>
</tr>
<tr>
<td>PS</td>
<td>PB</td>
<td>PS</td>
<td>ZE</td>
<td>NS</td>
<td>NS</td>
</tr>
<tr>
<td>ZE</td>
<td>PS</td>
<td>PS</td>
<td>ZE</td>
<td>NS</td>
<td>NB</td>
</tr>
<tr>
<td>NS</td>
<td>PS</td>
<td>PS</td>
<td>ZE</td>
<td>NS</td>
<td>NB</td>
</tr>
<tr>
<td>NB</td>
<td>PS</td>
<td>PS</td>
<td>ZE</td>
<td>NB</td>
<td>NB</td>
</tr>
</tbody>
</table>
In the air tank design optimization example, design variables \( r, s, \) and \( h \) have only one critical constraint \((g_1, g_3, \) and \( g_2, \) respectively\), which is situation (1) in the four possible situations discussed in section 3. Therefore the change of design variable \( r, s, h \) in the next iteration can be defined as follows:

\[
\Delta r = \text{defuzzification}(\mu_{g_1}(r)) \\
\Delta s = \text{defuzzification}(\mu_{g_3}(s)) \\
\Delta h = \text{defuzzification}(\mu_{g_2}(h)).
\] (8)

The monotonicity sign of design variable \( l \) is ‘indeterminate’ in the objective function of the final monotonicity table output from MONO, which is situation (3) discussed in section 3. Therefore the change of design variable \( l \) in the next iteration is decided by constraints \((g_1, g_4, \) and \( g_6, \) in which \( l \) appears using equation (6), and can be written as

\[
\Delta l = \text{defuzzification}\left\{ \begin{array}{ll}
\mu_{g_1}, & \text{if } g_1 \text{ is violated} \\
\mu_{g_4}, & \text{if } g_4 \text{ is violated} \\
\mu_{g_6}, & \text{if } g_6 \text{ is violated} \\
\max(\mu_{g_1} \cup \mu_{g_4}), & \text{if } g_1 \text{ and } g_4 \text{ are violated} \\
\max(\mu_{g_1} \cup \mu_{g_6}), & \text{if } g_1 \text{ and } g_6 \text{ are violated} \\
\max(\mu_{g_1} \cup \mu_{g_4} \cup \mu_{g_6}), & \text{if } g_1, g_4 \text{ and } g_6 \text{ are violated}
\end{array}\right.
\] (9)

4.3 Defining the initial values and move limits of the design variables

The inputs of the fuzzy PD controller optimization engine are the constraint function values \((\Delta \Delta e)\) and the change of the constraint function values \((\Delta \Delta e)\). Table 5 gives the quantitative definitions for the error inputs \((e : g_1, g_2, g_3, g_4, \) and \( g_6, \) and table 6 gives the quantitative definitions for the error change inputs \((\Delta e : \Delta g_1, \Delta g_2, \Delta g_3, \Delta g_4, \) and \( \Delta g_6). \) As discussed in the previous

<table>
<thead>
<tr>
<th>Quantized level</th>
<th>( g_1 )</th>
<th>( g_2 )</th>
<th>( g_3 )</th>
<th>( g_4 )</th>
<th>( g_6 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>(</td>
<td>g_1</td>
<td>_{\text{initial}} )</td>
<td>(</td>
<td>g_2</td>
</tr>
<tr>
<td>1</td>
<td>(</td>
<td>g_1</td>
<td>_{\text{initial}} )/2</td>
<td>(</td>
<td>g_2</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>−1</td>
<td>−(</td>
<td>g_1</td>
<td>_{\text{initial}} )/2</td>
<td>−(</td>
<td>g_2</td>
</tr>
<tr>
<td>−2</td>
<td>−(</td>
<td>g_1</td>
<td>_{\text{initial}} )</td>
<td>−(</td>
<td>g_2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Quantized level</th>
<th>( \Delta g_1 )</th>
<th>( \Delta g_2 )</th>
<th>( \Delta g_3 )</th>
<th>( \Delta g_4 )</th>
<th>( \Delta g_6 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>( \Delta g_1(x_0)_{\text{max}} )</td>
<td>( \Delta g_2(x_0)_{\text{max}} )</td>
<td>( \Delta g_3(x_0)_{\text{max}} )</td>
<td>( \Delta g_4(x_0)_{\text{max}} )</td>
<td>( \Delta g_6(x_0)_{\text{max}} )</td>
</tr>
<tr>
<td>1</td>
<td>( \Delta g_1(x_0)_{\text{max}}/2 )</td>
<td>( \Delta g_2(x_0)_{\text{max}}/2 )</td>
<td>( \Delta g_3(x_0)_{\text{max}}/2 )</td>
<td>( \Delta g_4(x_0)_{\text{max}}/2 )</td>
<td>( \Delta g_6(x_0)_{\text{max}}/2 )</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>−1</td>
<td>−( \Delta g_1(x_0)_{\text{max}}/2 )</td>
<td>−( \Delta g_2(x_0)_{\text{max}}/2 )</td>
<td>−( \Delta g_3(x_0)_{\text{max}}/2 )</td>
<td>−( \Delta g_4(x_0)_{\text{max}}/2 )</td>
<td>−( \Delta g_6(x_0)_{\text{max}}/2 )</td>
</tr>
<tr>
<td>−2</td>
<td>−( \Delta g_1(x_0)_{\text{max}} )</td>
<td>−( \Delta g_2(x_0)_{\text{max}} )</td>
<td>−( \Delta g_3(x_0)_{\text{max}} )</td>
<td>−( \Delta g_4(x_0)_{\text{max}} )</td>
<td>−( \Delta g_6(x_0)_{\text{max}} )</td>
</tr>
</tbody>
</table>
section, these quantization tables are used in the fuzzification step to convert $e$ and $\Delta e$ into fuzzy membership for the rule base in tables 3 and 4.

As shown in table 5, the values of the error inputs at different quantized levels are determined by the initial values of the constraint functions. In this example, the initial values of the variables are

$$h = 10 \text{ cm} \quad l = 800 \text{ cm} \quad r = 150 \text{ cm} \quad s = 3 \text{ cm}.$$  \hspace{1cm} (10)

The constraint function values can be evaluated at the initial design point:

$$g_{1\text{initial}} = -0.63 \quad g_{2\text{initial}} = 0.95 \quad g_{3\text{initial}} = -0.52$$

$$g_{4\text{initial}} = -0.99 \quad g_{6\text{initial}} = 0.31.$$  \hspace{1cm} (11)

In table 6, the quantization level value of error change ($\Delta e : \Delta g_1, \Delta g_2, \Delta g_3, \Delta g_4$ and $\Delta g_6$) is determined by

$$\Delta g_i(x_0)_{\text{max}} = g_i(x_0 \pm \Delta x_{\text{max}}) - g_i(x_0)$$  \hspace{1cm} (12)

where $x_0$ is a vector of initial values of the design variable, and $\Delta x_{\text{max}}$ is the vector of ‘move limits’ of the variables. The $\pm$ sign depends on the monotonicity of the variable to ensure that $\Delta g_i(x_0)_{\text{max}}$ is always positive. If the variable is monotonically increasing in the constraint function, the sign will be $+$, and vice versa.

The move limit of a design variable is the maximum amount of change of the variable allowed in one iteration. In this example, the move limits are

$$\Delta h_{\text{max}} = 2 \quad \Delta l_{\text{max}} = 160 \quad \Delta r_{\text{max}} = 30 \quad (\Delta s)_{\text{max}} = 0.6.$$  \hspace{1cm} (13)

Therefore, from equation (13),

$$\Delta g_1(r, l)_{\text{max}} = 0.36 \quad \Delta g_2(r, h)_{\text{max}} = 0.98 \quad \Delta g_3(r, s)_{\text{max}} = 0.24$$

$$\Delta g_4(l)_{\text{max}} = 3.1 \times 10^{-3} \quad \Delta g_6(l)_{\text{max}} = 0.26.$$  \hspace{1cm} (14)

The outputs from the fuzzy PD optimization engine are the changes of the design variables in the next iteration. Table 7 shows the quantization level of the outputs, which are also determined by the move limits defined by the designer. As discussed in the previous section, this quantization table is used to ‘defuzzify’ the outputs into crisp values of $\Delta h$, $\Delta l$, $\Delta r$, and $\Delta s$ for the next iteration (equations (8) and (9)).

### 4.4 The optimization results

Using the initial design point and move limits described above, the fuzzy PD controller optimization engine terminates after 49 iterations when the change in objective function value in consecutive iterations is less than 0.01% and all constraints are satisfied within a

<table>
<thead>
<tr>
<th>Quantized level</th>
<th>$\Delta h$</th>
<th>$\Delta l$</th>
<th>$\Delta r$</th>
<th>$\Delta s$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>((\Delta h)_{\text{max}})</td>
<td>((\Delta l)_{\text{max}})</td>
<td>((\Delta r)_{\text{max}})</td>
<td>((\Delta s)_{\text{max}})</td>
</tr>
<tr>
<td>1</td>
<td>((\Delta h)_{\text{max}}/2)</td>
<td>((\Delta l)_{\text{max}}/2)</td>
<td>((\Delta r)_{\text{max}}/2)</td>
<td>((\Delta s)_{\text{max}}/2)</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>−1</td>
<td>(-(\Delta h)_{\text{max}}/2)</td>
<td>-(\Delta l)_{\text{max}}/2)</td>
<td>-(\Delta r)_{\text{max}}/2)</td>
<td>-(\Delta s)_{\text{max}}/2)</td>
</tr>
<tr>
<td>−2</td>
<td>((\Delta h)_{\text{max}})</td>
<td>-(\Delta l)_{\text{max}})</td>
<td>-(\Delta r)_{\text{max}})</td>
<td>-(\Delta s)_{\text{max}})</td>
</tr>
</tbody>
</table>
tolerance of 0.01% of the initial values of the constraints. The numerical results $h = 13.67$ cm, $l = 610.00$ cm, $r = 105.18$ cm, and $s = 1.01$ cm are identical to the analytical solution. At this optimal design point, the values of the active constraints are $g_1 = 4.02 \times 10^{-10}$, $g_2 = 9.17 \times 10^{-5}$, $g_3 = 2.20 \times 10^{-10}$, and $g_6 = 7.83 \times 10^{-10}$. The values of other constraints are $g_4 = -0.98$ and $g_5 = -0.29$. The value of the objective function is $1.38 \times 10^6$.

Note that in this optimization process the function value of each constraint is evaluated 51 times, including two evaluations to construct tables 5 and 6. No sensitivity information is required. The designers only need to define the initial values and the move limits of the design variables in the optimization process.

Figure 6 shows the iteration history of the objective function of the air tank example. The objective function value drops rapidly in the first several iterations, and then shows a stable converging trend. Different initial values and different move limits were also tested for this example (tables 8 and 9). As expected, the number of iterations increases when the starting design point is farther from the optimal design point. In some tests larger move limits help to reduce the number of iterations, while in other tests larger move limits cause overshoot and the number of iterations increases. In all 19 tests shown in tables 8 and 9, the fuzzy PD controller optimization engine appears to be robust. The same numerical solutions are obtained although

![Figure 6. The iteration history of the objective function of the air tank example.](image)

<table>
<thead>
<tr>
<th>Iteration</th>
<th>$r$ (cm)</th>
<th>$s$ (cm)</th>
<th>$h$ (cm)</th>
<th>$l$ (cm)</th>
<th>$g_1$ (0.01%)</th>
<th>$g_5$ (0.1%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original</td>
<td>150</td>
<td>3</td>
<td>10</td>
<td>800</td>
<td>49</td>
<td>34</td>
</tr>
<tr>
<td>Test 1</td>
<td>200</td>
<td>3</td>
<td>10</td>
<td>800</td>
<td>48</td>
<td>21</td>
</tr>
<tr>
<td>Test 2</td>
<td>50</td>
<td>3</td>
<td>10</td>
<td>800</td>
<td>51</td>
<td>38</td>
</tr>
<tr>
<td>Test 3</td>
<td>150</td>
<td>5</td>
<td>10</td>
<td>800</td>
<td>49</td>
<td>24</td>
</tr>
<tr>
<td>Test 4</td>
<td>150</td>
<td>0.1</td>
<td>10</td>
<td>800</td>
<td>138</td>
<td>81</td>
</tr>
<tr>
<td>Test 5</td>
<td>150</td>
<td>3</td>
<td>15</td>
<td>800</td>
<td>17</td>
<td>13</td>
</tr>
<tr>
<td>Test 6</td>
<td>150</td>
<td>3</td>
<td>5</td>
<td>800</td>
<td>154</td>
<td>104</td>
</tr>
<tr>
<td>Test 7</td>
<td>150</td>
<td>3</td>
<td>10</td>
<td>1000</td>
<td>63</td>
<td>45</td>
</tr>
<tr>
<td>Test 8</td>
<td>200</td>
<td>5</td>
<td>15</td>
<td>1000</td>
<td>44</td>
<td>29</td>
</tr>
<tr>
<td>Test 9</td>
<td>50</td>
<td>0.1</td>
<td>5</td>
<td>100</td>
<td>214</td>
<td>135</td>
</tr>
<tr>
<td>Optimal</td>
<td>$105.0$</td>
<td>$1.0$</td>
<td>$13.6$</td>
<td>$610.0$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
692

Y.-L. Hsu and T.-C. Liu

Table 9. Using different move limits in the air tank example. Bold type indicates a difference from the initial values.

<table>
<thead>
<tr>
<th>Variables</th>
<th>Iteration</th>
</tr>
</thead>
<tbody>
<tr>
<td>r (cm)</td>
<td>s (cm)</td>
</tr>
<tr>
<td>150</td>
<td>3</td>
</tr>
</tbody>
</table>

Move limits

<table>
<thead>
<tr>
<th>Move limits</th>
<th>Δr (cm)</th>
<th>Δs (cm)</th>
<th>Δh (cm)</th>
<th>Δl (cm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original</td>
<td>30</td>
<td>0.6</td>
<td>2</td>
<td>160</td>
</tr>
<tr>
<td>Test 1</td>
<td>15</td>
<td>0.6</td>
<td>2</td>
<td>160</td>
</tr>
<tr>
<td>Test 2</td>
<td>45</td>
<td>0.6</td>
<td>2</td>
<td>160</td>
</tr>
<tr>
<td>Test 3</td>
<td>30</td>
<td>0.3</td>
<td>2</td>
<td>160</td>
</tr>
<tr>
<td>Test 4</td>
<td>30</td>
<td>0.9</td>
<td>2</td>
<td>160</td>
</tr>
<tr>
<td>Test 5</td>
<td>30</td>
<td>0.6</td>
<td>1</td>
<td>160</td>
</tr>
<tr>
<td>Test 6</td>
<td>30</td>
<td>0.6</td>
<td>3</td>
<td>160</td>
</tr>
<tr>
<td>Test 7</td>
<td>30</td>
<td>0.6</td>
<td>2</td>
<td>80</td>
</tr>
<tr>
<td>Test 8</td>
<td>30</td>
<td>0.6</td>
<td>2</td>
<td>200</td>
</tr>
<tr>
<td>Test 9</td>
<td>15</td>
<td>0.3</td>
<td>1</td>
<td>80</td>
</tr>
<tr>
<td>Test 10</td>
<td>45</td>
<td>0.9</td>
<td>3</td>
<td>200</td>
</tr>
</tbody>
</table>

the number of iterations required varies. The iteration histories of all 19 tests also show a converging trend similar to that in figure 6. Table 8 also compares the number of iterations required for different termination criteria. Note that in these tests, a third to a half of the iterations are used to drive the optimal solution from a 0.1% tolerance to a 0.01% tolerance.

The fuzzy PD controller optimization engine has been tested in various engineering optimization problems with implicit functions and monotonicity. In the next section three representative design examples are described to demonstrate the practicality of this approach.

5. Design examples

5.1 A tension–compression spring design optimization problem

The weight of a tension–compression spring (figure 7) is to be minimized subject to constraints on minimum deflection ($g_1$), shear ($g_2$), and surge frequency ($g_3$), and limits on outside diameter ($g_4$) (Belegundu and Arora 1985; Arora 1989; Coello and Monts 2002). The design variables are the mean coil diameter $D$, the wire diameter $d$, and the number $N$ of active coils. The problem can be expressed as follows:

\[
\begin{align*}
\text{min.} \quad & F = (N + 2)Dd^2 \\
\text{s.t.} \quad & g_1 = \frac{71785xd^4}{D^3N} - 1 \leq 0 \\
& g_2 = \frac{4D^2 - dD}{12566(Dd^3 - d^4)} + \frac{1}{5108d^2} - 1 \leq 0 \\
& g_3 = \frac{D^2N}{140.45d} - 1 \leq 0 \\
& g_4 = \frac{d + D}{1.5} - 1 \leq 0.
\end{align*}
\]

The monotonicity table for this example is shown in table 10. After monotonicity analysis by the program MONO, it is concluded that the design variable $D$ has only one critical constraint $g_1$. One of the constraints in the conditionally critical set $g_2$ and $g_3$ must be critical for design...
variable \( d \). The monotonicity sign of design variable \( N \) is ‘indeterminate’ in the objective function at the end of monotonicity analysis.

The initial values of the design variables are

\[
d = 0.1 \quad D = 0.25 \quad N = 13
\]

and the move limits are

\[
(\Delta d)_{\text{max}} = 0.001 \quad (\Delta D)_{\text{max}} = 0.01 \quad (\Delta N)_{\text{max}} = 0.1. 
\]

Similar to tables 5–7, these user-defined values are used to determine the quantization levels of the error inputs, change of error inputs, and outputs.

The fuzzy PD controller optimization engine terminates after 225 iterations, when the change in objective function value in consecutive iterations is less than 0.01% and all constraints are satisfied within a tolerance of 0.01% of the initial values. Table 11 compares the optimization results obtained using the fuzzy PD controller optimization engine with those reported in the literature. Note that, although there are three design variables, only constraints

Table 11. Comparison of the results for the spring design example.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Objective function value</td>
<td>0.0126503</td>
<td>0.0126810</td>
<td>0.0127303</td>
<td>0.0128334</td>
</tr>
<tr>
<td>Iteration</td>
<td>225</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>( D )</td>
<td>0.052362</td>
<td>0.051989</td>
<td>0.053396</td>
<td>0.050000</td>
</tr>
<tr>
<td>( D )</td>
<td>0.373153</td>
<td>0.363965</td>
<td>0.399180</td>
<td>0.315900</td>
</tr>
<tr>
<td>( N )</td>
<td>10.36486</td>
<td>10.89052</td>
<td>9.185400</td>
<td>14.25000</td>
</tr>
<tr>
<td>( g_1 )</td>
<td>0.001987</td>
<td>-0.000013</td>
<td>0.000019</td>
<td>-0.000014</td>
</tr>
<tr>
<td>( g_2 )</td>
<td>0.000084</td>
<td>-0.000021</td>
<td>-0.000018</td>
<td>-0.003782</td>
</tr>
<tr>
<td>( g_3 )</td>
<td>-0.803753</td>
<td>-4.061338</td>
<td>-4.123832</td>
<td>-3.938302</td>
</tr>
<tr>
<td>( g_4 )</td>
<td>-0.716237</td>
<td>-0.722698</td>
<td>-0.698283</td>
<td>-0.756067</td>
</tr>
</tbody>
</table>
Figure 8. The iteration history of objective function of the spring design example.

$g_1$ and $g_2$ are active at the optimum design point. Figure 8 shows the iteration histories of the objective function. In this process, the function value of each constraint is evaluated 227 times, and no sensitivity information is required.

The purpose of the comparison in table 11 is to confirm the quality of the optimum solution obtained by the fuzzy PD controller optimization engine. The information required (and therefore the computational cost) in one iteration is different in every algorithm listed in table 11. Therefore the number of iterations of the literature algorithms cannot be compared directly.

5.2 Tubular column design optimization problem

Figure 9 shows an example for designing a uniform column of tubular section to carry a compressive load $P = 2500$ kgf at minimum cost (Rao 1996). The column is made of a material that has a yield stress $\sigma_y$ of $500$ kgf cm$^{-2}$, a modulus of elasticity $E$ of $0.85 \times 10^6$ kgf cm$^{-2}$, and

![Figure 9. The tubular column under compression design problem.](image-url)
a density $\rho$ of 0.0025 kgf cm$^{-3}$. The length $L$ of the column is 250 cm. The stress included in the column should be less than the buckling stress (constraint $g_1$) and the yield stress (constraint $g_2$). The mean diameter of the column is restricted between 2 and 14 cm (constraint $g_3$ and $g_4$), and columns with thickness outside the range 0.2–0.8 cm are not commercially available (constraint $g_5$ and $g_6$). The cost of the column includes material and construction costs and can be taken as $9.82dt + 2d$, where $d$ is the mean diameter of the column in centimetres and $t$ is tube thickness. The optimization model of this problem can be expressed as follows:

$$\min f(d, t) = 9.82dt + 2d$$

s.t. $g_1 = \frac{P}{\pi dt\sigma_y} - 1 \leq 0,$

$$g_2 = \frac{8PL^2}{\pi^3 Edt(d^2 + t^2)} - 1 \leq 0,$$

$$g_3 = \frac{2.0}{d} - 1 \leq 0,$$

$$g_4 = \frac{d}{14} - 1 \leq 0,$$

$$g_5 = \frac{0.2}{t} - 1 \leq 0,$$

$$g_6 = \frac{t}{0.8} - 1 \leq 0.$$

(18)

After monotonicity analysis by the program MONO, it is concluded that there are two conditionally critical sets. One of the constraints in the conditionally critical set $\{g_1, g_2, g_3\}$ must be critical for design variable $d$, and one of the constraints in the other conditionally critical set $\{g_1, g_2, g_5\}$ must be critical for design variable $t$.

In this example, the initial values of the design variables are

$$d = 14.0\,\text{cm} \quad t = 0.8\,\text{cm}$$

and the values of the move limits are:

$$\Delta d_{\text{max}} = 1.0 \quad \Delta t_{\text{max}} = 0.1.$$

(20)

Using the same termination criteria of the previous example, the fuzzy PD controller optimization engine terminates after 53 iterations. Table 12 compares the results obtained from

<table>
<thead>
<tr>
<th></th>
<th>Fuzzy PD</th>
<th>Rao (1996)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Objective function</td>
<td>25.5316</td>
<td>26.5323</td>
</tr>
<tr>
<td>Iteration</td>
<td>53</td>
<td>3</td>
</tr>
<tr>
<td>$d$</td>
<td>5.4507</td>
<td>5.4400</td>
</tr>
<tr>
<td>$T$</td>
<td>0.2920</td>
<td>0.2930</td>
</tr>
<tr>
<td>$g_1$</td>
<td>$-7.8 \times 10^{-5}$</td>
<td>$-0.8579$</td>
</tr>
<tr>
<td>$g_2$</td>
<td>$-7.8 \times 10^{-5}$</td>
<td>$-0.9785$</td>
</tr>
<tr>
<td>$g_3$</td>
<td>$-0.6331$</td>
<td>$-0.8571$</td>
</tr>
<tr>
<td>$g_4$</td>
<td>$-0.6107$</td>
<td>0.0000</td>
</tr>
<tr>
<td>$g_5$</td>
<td>$-0.3151$</td>
<td>$-0.7500$</td>
</tr>
<tr>
<td>$g_6$</td>
<td>0.6350</td>
<td>0.0000</td>
</tr>
</tbody>
</table>
the fuzzy PD controller optimization engine with those reported in the literature. Figure 10 shows the iteration histories of the objective function.

5.3 The speed reducer design optimization problem

The design of the speed reducer (Golinski 1973) shown in figure 11 is considered with the face width \( b \), module of teeth \( m \), number of teeth on pinion \( z \), length of shaft 1 between bearings \( l_1 \), length of shaft 2 between bearings \( l_2 \), diameter of shaft 1 \( d_1 \), and diameter of shaft 2 \( d_2 \). The objective is to minimize the total weight of the speed reducer. The constraints include limitations on the bending stress of the gear teeth, surface stress, transverse deflections of shafts 1 and 2 due to transmitted force, and stresses in shafts 1 and 2. The design optimization model can be summarized as follows:

\[
\begin{align*}
\text{min } f(b, m, z, l_1, l_2, d_1, d_2) &= 0.7854bm^2(3.3333z^2 + 14.9334z - 43.0934) \\
&\quad - 1.508b(d_1^2 + d_2^2) + 7.477(d_1^3 + d_2^3) + 0.7854(l_1d_1^2 + l_2d_2^2) \\
\text{s.t. } g_1 &= \frac{27}{bm^2z} - 1 \leq 0, \\
g_2 &= \frac{397.5}{bm^2z^2} - 1 \leq 0, \\
g_3 &= \frac{1.93}{mzl_1^3d_1^4} - 1 \leq 0, \\
g_4 &= \frac{1.93}{mzl_2^3d_2^4} - 1 \leq 0, \\
g_5 &= \sqrt{(745l_1/mz)^2 + 1.69 \times 10^6} - \frac{110d_1^3}{100} - 1 \leq 0, \\
g_6 &= \sqrt{(745l_1/mz)^2 + 157.5 \times 10^6} - \frac{85d_2^3}{85} - 1 \leq 0,
\end{align*}
\]
There are seven design variables and 25 constraints in this model. After monotonicity analysis by the program MONO, it is concluded that there are seven conditionally critical sets for the seven design variables.

The initial values of the design variables are

\[
\begin{align*}
    b &= 3.6, \\
    m &= 0.8, \\
    z &= 28, \\
    l_1 &= 8.3, \\
    l_2 &= 8.3, \\
    b_1 &= 3.9, \\
    b_2 &= 5.5
\end{align*}
\]
and the values of the move limits are

\[
\begin{align*}
(\Delta b)_{\text{max}} & = 0.1 \\
(\Delta m)_{\text{max}} & = 0.01 \\
(\Delta z)_{\text{max}} & = 1.0 \\
(\Delta l_{1})_{\text{max}} & = 0.1 \\
(\Delta l_{2})_{\text{max}} & = 0.01 \\
(\Delta b_{1})_{\text{max}} & = 0.01 \\
(\Delta b_{2})_{\text{max}} & = 0.01.
\end{align*}
\] (23)

Following the same process, the fuzzy PD controller optimization engine terminates after 302 iterations. Table 13 compares the results obtained from the fuzzy PD controller

<table>
<thead>
<tr>
<th></th>
<th>Fuzzy PD</th>
<th>Golinski (1973)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Objective function value</td>
<td>3007.8</td>
<td>2985.2</td>
</tr>
<tr>
<td>Iteration</td>
<td>302</td>
<td>–</td>
</tr>
<tr>
<td>$x_{1}$</td>
<td>3.5197</td>
<td>3.5</td>
</tr>
<tr>
<td>$x_{2}$</td>
<td>0.7039</td>
<td>0.7</td>
</tr>
<tr>
<td>$x_{3}$</td>
<td>17.3831</td>
<td>17.0</td>
</tr>
<tr>
<td>$x_{4}$</td>
<td>7.3000</td>
<td>7.3</td>
</tr>
<tr>
<td>$x_{5}$</td>
<td>7.7152</td>
<td>7.3</td>
</tr>
<tr>
<td>$x_{6}$</td>
<td>3.3498</td>
<td>3.35</td>
</tr>
<tr>
<td>$x_{7}$</td>
<td>5.2866</td>
<td>5.29</td>
</tr>
<tr>
<td>$g_{1}$</td>
<td>-0.1095</td>
<td>-0.0739</td>
</tr>
<tr>
<td>$g_{2}$</td>
<td>-0.2458</td>
<td>-0.1980</td>
</tr>
<tr>
<td>$g_{3}$</td>
<td>-0.5127</td>
<td>-0.4990</td>
</tr>
<tr>
<td>$g_{4}$</td>
<td>-0.9073</td>
<td>-0.9194</td>
</tr>
<tr>
<td>$g_{5}$</td>
<td>0.0000</td>
<td>0.0001</td>
</tr>
<tr>
<td>$g_{6}$</td>
<td>0.0000</td>
<td>-0.0020</td>
</tr>
<tr>
<td>$g_{7}$</td>
<td>-0.6941</td>
<td>-0.7025</td>
</tr>
<tr>
<td>$g_{8}$</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>$g_{9}$</td>
<td>0.5833</td>
<td>-0.5833</td>
</tr>
<tr>
<td>$g_{10}$</td>
<td>-0.2613</td>
<td>-0.2571</td>
</tr>
<tr>
<td>$g_{11}$</td>
<td>-0.0223</td>
<td>-0.0278</td>
</tr>
<tr>
<td>$g_{12}$</td>
<td>-0.0056</td>
<td>0.0000</td>
</tr>
<tr>
<td>$g_{13}$</td>
<td>-0.1201</td>
<td>-0.1250</td>
</tr>
<tr>
<td>$g_{14}$</td>
<td>-0.0220</td>
<td>0.0000</td>
</tr>
<tr>
<td>$g_{15}$</td>
<td>-0.3792</td>
<td>-0.3929</td>
</tr>
<tr>
<td>$g_{16}$</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>$g_{17}$</td>
<td>-0.1205</td>
<td>-0.1205</td>
</tr>
<tr>
<td>$g_{18}$</td>
<td>-0.0538</td>
<td>0.0000</td>
</tr>
<tr>
<td>$g_{19}$</td>
<td>-0.0705</td>
<td>-0.1205</td>
</tr>
<tr>
<td>$g_{20}$</td>
<td>-0.1343</td>
<td>-0.1343</td>
</tr>
<tr>
<td>$g_{21}$</td>
<td>-0.1411</td>
<td>-0.1410</td>
</tr>
<tr>
<td>$g_{22}$</td>
<td>-0.0542</td>
<td>-0.0548</td>
</tr>
<tr>
<td>$g_{23}$</td>
<td>-0.0388</td>
<td>-0.0382</td>
</tr>
<tr>
<td>$g_{24}$</td>
<td>-0.0514</td>
<td>-0.0514</td>
</tr>
<tr>
<td>$g_{25}$</td>
<td>0.0000</td>
<td>0.0574</td>
</tr>
</tbody>
</table>
optimization engine with those reported in the literature. Figure 12 shows the iteration history of the objective function.

6. Conclusions and discussion

In real world engineering design problems, engineering heuristics often relate to the monotonic behaviour of certain design variables. Indeed, monotonicity between the design variables and the objective and constraint functions prevails in engineering design optimization problems. Monotonicity analysis can generate valuable qualitative information for this type of optimization model, such as the activity of the constraints at the optimum design point. However, numerical optimal solutions cannot be obtained by monotonicity analysis alone.

The fuzzy PD controller optimization engine developed in this research utilizes the monotonicity of the design variables and the activity of the constraints concluded from monotonicity analysis in the optimization process to obtain the numerical optimum solution. The basic structure of this optimization process is actually very similar to the traditional line search algorithms. In a numerical line search algorithm, the search direction and step length are determined by numerical information, such as the gradients of the objective function and constraints. In the optimization process presented here, the search direction and step length are generated by the fuzzy PD controller optimization engine using the information on the monotonicity of the design variables and the activity of the constraints, as well as the initial values and move limits of the design variables defined by the user. In the optimization process, the objective and constraint functions only need to be evaluated once in each iteration. No sensitivity information is required. This characteristic makes the optimization algorithm particularly suitable for engineering optimization problems with implicit constraints. The fuzzy PD controller optimization engine appears to be robust in the various design examples tested in this research.

In this research, the optimization process is analogous to a closed-loop control system, and the fuzzy PD controller optimization engine was developed following the general concept of developing a fuzzy PD controller for such a system. In future developments, concepts and methods commonly used for improving fuzzy PD controllers (e.g. how to speed up the response, reduce the overshoot, improve the transient response, etc.) could also be tested to
further improve the performance of the fuzzy PD controller optimization engine. Moreover, many design variables in engineering optimization problems are discrete in nature. This should also be considered in the future development of the fuzzy PD controller optimization engine.

References


