A Genetic Search Approach to Unfalsified PI Control Design for a Weigh Belt Feeder

by

Emmanuel G. Collins, Jr. and Yanan Zhao
Department of Mechanical Engineering
Florida A&M University - Florida State University
Tallahassee, FL 32310
ecollins@eng.fsu.edu, yzhao@eng.fsu.edu

Richard Millett
Merrick Industries, Inc.
Lynn Haver, FL 32444
rmillett@merrick-inc.com

Abstract

This paper proposes and experimentally demonstrates an approach to automated PI tuning for an industrial weigh belt feeder that is based on unfalsified control concepts. Unfalsified control is used here as a means of using either open or closed loop test data to identify a subset of controllers (from an initial set) that is not proved to violate the multiple objectives specified by the control engineer. A novel feature of the unfalsified approach is that it allows controllers to be eliminated from consideration by predicting their performance without actually inserting the controllers in the loop. In addition, this methodology does not require an explicit model. However, in practice it does require some closed-loop experimentation to determine the cost functions used to perform the unfalsification. When the unfalsified PI autotuning approach is applied to the industrial weight belt feeder, it is able to successfully identify a subset of PI control laws that meets the performance specs. A key feature of this paper is the use of a genetic search algorithm to reduce the computational requirements of unfalsified control, especially when the initial set of controllers is large.

Keywords: unfalsified control, genetic algorithms, PI autotuning, weigh belt feeder

---

This research was supported in part by the National Science Foundation under Grant CMS-9802197.
1. Introduction

An industrial weigh belt feeder is designed to transport solid materials into a manufacturing process at a constant feedrate, usually in kilograms or pounds per sec. The weigh belt feeder used in this research was designed and manufactured by Merrick Industries, Inc. of Lynn Haven, Florida and is a process feeder that is typically used in a food, chemical or plastics manufacturing process. To ensure a constant feedrate in industrial operation, a PI control law is designed and implemented in the Merrick controller. In current practice the PI tuning process is performed manually by an engineering technician. However, for better and more consistent quality, it is desired to use automated PI tuning. In this research an automated tuning procedure is proposed and implemented using unfalsified control concepts.\textsuperscript{7,7} Controller falsification concepts based on model reference control are given in \textsuperscript{7}.

Controller unfalsification is a natural extension of model unfalsification and leads to less conservative control laws.\textsuperscript{7,7} It points out that for a set of causally-left-invertible candidate controllers, such as a set of PI control laws, decisions about which control laws should be falsified (i.e., deemed unsuitable) or left unfalsified (i.e., deemed suitable) can be made based on stored sensor output signals and actuator input signals, possibly representing open-loop or closed-loop data obtained with another controller in the feedback loop. Consequently, it is not necessary for a controller to be actually inserted in the feedback loop to be falsified. This is important because it means that adaptive unfalsified control processes may be significantly less susceptible to poor transient response than adaptive or tuning processes that require inserting controllers in the loop one-at-time to determine their suitability.

A particularly attractive feature of unfalsified control is the ability to find control laws that can meet multiple objectives. Since most real world engineering problems have more than one objective, this offers an improvement over standard optimal and adaptive control approaches that are based on optimizing or constraining only a scalar cost function. In addition, unfalsified control is simple to implement, typically involving only real-time integration of algebraic functions of the observed data, with one set of integrators for each candidate controller. However, in practice the cost functions used for the unfalsified control procedure must be chosen carefully based on closed-loop experimentation. This increases the effort needed to develop effective unfalsified control procedures.

Instead of implementing the controller unfalsification concept in real-time,\textsuperscript{7} this research applied the concept off-line. The original implementation essentially involved a brute force search over the original set of controllers. To reduce the computational requirements of the unfalsified control process, a genetic search algorithm was used.
The paper is organized as follows. Section 2 reviews the unfalsified control concept. Section 3 uses the unfalsified control concept to develop an automated PI tuning method for an industrial weigh belt feeder. Section 4 develops a genetic search algorithm for more efficient implementation of the unfalsified control concept. Section 5 presents actual experimental results showing the effectiveness of the unfalsification algorithm. Section 6 demonstrates the computational benefits of using the genetic algorithm instead of an exhaustive search. Finally, Section 7 presents some conclusions.

2. Review of Concepts from Unfalsified Control Theory

Consider the feedback control system of Figure 1. Given $K = \{K_1, K_2, \ldots, K_N\}$, a finite, initial set of causally-left-invertible control laws, the goal of unfalsified control is to determine a $K \in K$ such that for a set of plants $P$, representing the variations of the real system, the closed-loop system response satisfies a set of performance specs involving the command signal $r(t)$, the command input $u(t)$, and the measured output $y(t)$. Unfalsified control concepts use real test data to determine $K \in K$ that are not predicted by data from the plant to violate the performance specs.

![Figure 1: Control System](image)

Let $(\bar{y}, \bar{u})$ denote a set of open or closed-loop test data, taken with sample period $h$ in the time interval $[t_0, t_0 + ph]$ where $p$ is some positive integer. Specifically,

$$
\bar{y} = \begin{bmatrix}
y(t_0) \\
y(t_0 + h) \\
y(t_0 + 2h) \\ \\
y(t_0 + ph) 
\end{bmatrix}, \quad \bar{u} = \begin{bmatrix}
u(t_0) \\
u(t_0 + h) \\
u(t_0 + 2h) \\ \\
u(t_0 + ph) 
\end{bmatrix}.
$$

(1)

Since a controller $K_i \in K$ is causally-left-invertible, it is possible to determine a unique set of error
signals, represented by \( \bar{e}_K \), where

\[
\bar{e}_K = \begin{bmatrix}
  e_{K_i}(t_0) \\
  e_{K_i}(t_0 + h) \\
  e_{K_i}(t_0 + 2h) \\
  \vdots \\
  e_{K_i}(t_0 + ph)
\end{bmatrix}
\]

(2)

that will result in the control signals \( \bar{u} \) if \( K_i \) is in the loop. It follows that the corresponding command signals \( \bar{r}_K \) are given by

\[
\bar{r}_K = \bar{e}_K + \bar{y}.
\]

Hence, \( \bar{r}_K \) is the set of command signals that would have yielded the signals \( \bar{u} \) and \( \bar{y} \) if the controller \( K_i \) were in the loop. Therefore, associated with each \( K_i \) \( (i = 1, 2, \ldots, N) \) is a triple \( (\bar{r}_K, \bar{y}, \bar{u}) \).

Now assume that there exist scalar cost functions \( J_j : \mathbb{R}^{(p+1)} \times \mathbb{R}^{(p+1)} \times \mathbb{R}^{(p+1)} \rightarrow \mathbb{R}, \ j = 1, \ldots, q, \) and associated performance specifications

\[
J_j(\bar{r}_K, \bar{y}, \bar{u}) \leq \bar{\sigma}_j, \ j = 1, \ldots, q.
\]

(3)

Then, for each control law \( K_i \) \( (i = 1, 2, \ldots N) \) one can compute the set of costs \( \{J_j(\bar{r}_K, \bar{y}, \bar{u})\}_{j=1}^q \). A controller is falsified if there exists \( j \in \{1, 2, \ldots, q\} \) such that

\[
J^*_j(\bar{r}_K, \bar{y}, \bar{u}) > \bar{\sigma}_j
\]

(4)

It is important to note that in practical application of unfalsified control the construction of useful cost functions \( J_j(\cdot) \) requires some closed-loop experimentation.

The remaining set of unfalsified controllers, \( K_u \subseteq K \) can now be reevaluated by using a new set of open or closed loop data \( (\bar{u}^{(new)}, \bar{y}^{(new)}). \) This process may be repeated as long as the set of unfalsified controllers is nonempty. The reader is referred to\(^7\) for more information regarding unfalsified control concepts.

3. Formulation of Unfalsified Control the Weigh Belt Feeder

The weigh belt feeder used in this research, shown in Figure ??, is a product of Merrick Industries, Inc. It is a typical process feeder that would be used in a food, chemical or plastics manufacturing process. To weigh the material, a weigh deck is mounted on a precision strain gauge load cell. This directly gives a belt load that is measured in kg/m. A 1000 pulse per revolution optical encoder is mounted on the tail pulley of the feeder and is used to measure the distance the belt has travelled. By taking the first derivative of the belt travel, we obtain the belt speed in
m/sec. The feedrate is calculated by multiplying the belt speed and the material load on the belt. This product gives us a feedrate in kg/sec. To control the feedrate, the feeder has a field wound DC motor and a silicon controlled rectifier (SCR) motor controller combination. The motor is coupled to the head pulley of the feeder through a reducer and chain drive combination. The belt speed and hence the overall system feedrate is controlled by varying the rotational rate of the motor. The plant in Figure ?? presents a schematic description of the feeder.

![Figure 2: The Merrick Weigh Belt Feeder](image)

### 3.1. Experimental System Setup

Using REALoop, a software and hardware kit from XANALOG Corp., real-time experiments were performed. REALoop software is a single Simulink block with its own dialog box, which can be dragged into any Simulink block model. In this dialog box, the user may define the real-time sample time and indicate the number of PC A/D and D/A board channels to communicate with the real world. The experimental control system configuration used to implement an automated PI tuning procedure is shown in Figure ??.

### 3.2. Nonlinear Dynamics and Standard Tuning

The dynamics of the weigh belt feeder are dominated by the motor. To protect the motor, the control signal is restricted to lie in the interval [0,10] volts. The motor also has significant friction. In addition, the sensors exhibit significant quantization noise. Hence, the weigh belt feeder exhibits nonlinear behavior.

The system nonlinearities makes standard tuning methods difficult to apply. PI tuning was attempted using both Ziegler Nichols tuning and relay feedback auto-tuning. In attempting to apply Ziegler Nichols tuning, the system saturated before the ultimate gain was obtained. When relay feedback tuning was attempted, because of the sensor quantization noise and the motor friction,
the desired square wave with symmetric positive and negative half cycles could not be achieved even after considering the relay hysteresis and compensating for part of the load disturbance.

3.3. Performance Specifications

The cost functions used in the unfalsification process are based on the actual engineering goals and the observed closed-loop performance of the feeder. As previously mentioned, the control signal $u(t)$ is restricted to lie in the interval $[0, 10]$ volts. Also, in practice, $r(t) \leq 5$ volts (corresponding to a belt speed of $2.54 \times 10^{-2}$ m/sec ($5$ ft/min)). It was experimentally observed that saturation occurs if the following constraint is violated:

$$J_1(r_{K_1}, \bar{u}) = \rho \max_{t \in T} |u(t)| \leq \min_{t \in T} |r(t)| \leq 2,$$

(5)


where $T = (t_0, t_0 + h, \ldots, t_0 + ph)$ and $\rho$ takes on different values for different setpoints. In particular, $\rho$ increases as the setpoint increases. The number 2 in (5) is the ratio of the absolute value of the saturation control signal (10 volts) and the maximum reference signal (5 volts). Since actuator saturation is a hard constraint (i.e., it must be satisfied for proper operation of the controlled system), the constraint (5) is also treated as a hard constraint.

To achieve a good step response (i.e., low overshoot and fast settling time) another cost function is needed for unfalsification. The cost function constructed using experimental observations is

$$J_2(r_{K_1}, \bar{y}) = \frac{\|\bar{r}_{K_1} - \bar{y}\|_2}{\|\bar{r}_{K_1}\|_2} + \rho_1 K_P + \rho_2 K_I,$$

(6)

where $K_P$ and $K_I$ are respectively the proportional and integral gains of the PI controller. The terms $\rho_1 K_P$ and $\rho_2 K_I$ are used to take into account the transient overshoot of the system. The parameters $\rho_1$ and $\rho_2$ are selected to make the latter two terms in (6) comparable in size to the first term.
In the unfalsified tuning process for the weigh belt feeder a controller is falsified if (??) is violated. Among \( K_u \), the current set of unfalsified controllers, the “best” controller is considered to be the controller that satisfies

\[
\min_{K \in K_u} J_2(\bar{r}_K, \bar{y}).
\]

(7)

4. A Genetic Algorithm for Unfalsified Control

A genetic algorithm (GA), which may be viewed as a random search process, can be used to avoid computing the costs for each of the unfalsified controllers and hence can increase the computational efficiency of the unfalsified control process. The unfalsified control problem described above can be viewed as a constrained optimization problem, i.e., solve the optimization problem (??) subject to the constraint (??). Hence the GA must be suitable for such a constrained optimization problem. Below, the GA chosen for the unfalsification is described and then several factors affecting the efficiency are discussed.

4.1. A Genetic Algorithm for Constrained Optimization

GAs were originally developed for unconstrained optimization problems. To solve constrained optimization problems using a GA, both the problem and the GA have to be adapted.

There are generally two ways to develop GAs for constrained optimization. One is the penalty function method. The other is to try to enforce the constraints within the GA itself. In the penalty function method, a new objective function is constructed by adding the original one and a second term that measures the amount that the constraints are violated. This method will not be used here.

Three ways can be used to maintain the constraints within GAs by genetic operators (i.e., crossover and mutation operators): (1) filtering, i.e., use “reckless” genetic operators (here “reckless” refers to operators that do not guarantee the feasibility of the children), check the feasibility of each generated child, and eliminate children that are not feasible; (2) repairing, i.e., use “reckless” genetic operators, check the feasibility of each generated child, modify candidates that are not feasible and make them feasible; (3) preserving, where specific genetic operators are used which produce feasible children from feasible parents.

Filtering is very inefficient because many infeasible children are produced. Preserving is mostly problem specific and needs newly designed genetic operators. Repairing is used for this problem by adding a simple repair mechanism to the algorithm. Although it requires some extra operations,
the overall performance of the GA is improved. The repair operation for unfalsified PI control is based on the continuity of a controller's performance. Due to this continuity, a point in the immediate neighborhood of a feasible point (i.e., a controller that satisfies the constraint (??)) has a high chance of also being feasible, although feasibility is not guaranteed. Hence, if a genetic operator generated an infeasible point, the repairing operation replaced this infeasible point with a feasible point found in a neighborhood of a currently known, but randomly chosen feasible point. (Later we will mention that the currently known feasible points are stored in a table along with the corresponding costs.) The known feasible point was chosen randomly to reduce the possibility of overlapping chromosomes.

A binary representation is used in the GA for unfalsification. Hence, it is better to keep the domain with a cardinality of $2^m$, where $m$ is a positive integer. In this way each value has a unique binary representation. Otherwise, the algorithm will produce redundant or meaningless solutions.

4.2. Genetic Algorithm Flowchart

The flowchart of the algorithm developed for unfalsification is shown in Figure ???. The steps in this flowchart are elaborated below:

- **Initialize key algorithm parameters.** The maximum number of generations $g$, the population size, the crossover rate, and the mutation rate are initialized.

- **Generate the initial population.** The initial population is generated randomly. Only chromosomes whose fitness functions satisfy the constraint (??) are maintained in the initial population.

- **Perform genetic selection.** A genetic selection method is used to choose parents. In this research the roulette wheel selection method? was used.

- **Perform genetic reproduction.** Both crossover and mutation are used to generate children.

- **Evaluate and store the values of the fitness function.** The fitness function is evaluated and stored in a fitness table for each new child, i.e., a child that has not previously been generated. If the child was previously generated, the value of the fitness function is determined from the fitness table to save computational time.

- **Repair the children and store the values of the fitness function.** For those children who do not satisfy the constraint (??), the children are repaired so that the constraint is satisfied. (See the discussion in subsection 4.1.) All the children generated by the repair operation are also stored in the fitness table.
Initialize key algorithm parameters (e.g., maximum number of generations $g$).

- Generate the initial population. Set $i=1$.

- Perform genetic selection.

- Perform genetic reproduction.

- Evaluate and store the values of the fitness function.

- Repair the children and store the values of the fitness function.

- $i < g$? $i=i+1$

Stop.

**Figure 4:** Flow Chart of Genetic Algorithm for Unfalsification

4.3. **Several Factors Affecting the Efficiency**

Several factors influence the efficiency of a GA. The most important of these are the initial population, the crossover rate, and the mutation rate.

The initial population of a GA is usually generated randomly. For the constrained optimization considered here the initial population is chosen such that each member satisfies the constraint. This was seen to greatly increase the overall efficiency of the method even though the time it takes to perform the initialization increases.

Crossover and mutation are the two commonly used genetic operators in GAs. The role of crossover is to preserve, to survive, and to construct building blocks. Mutation provides a small amount of random search which means mutation is expected to introduce diversity. Adding diversity to the random search reduces the possibility that the chromosomes will crowd over a small region.

Another problem with a GA is that in different generations, some of the newly generated chromosomes overlap with the old ones. To reduce the computational burden of repeatedly computing the cost functions of these overlapping chromosomes, a table was used for storing the cost function values. Hence, for overlapping chromosomes, the cost was determined by a table lookup instead of
re-computation. This greatly reduced the computational time of the GA.

5. Implementation and Results

Due to the nonlinearities in the dynamics of the feeder, specifically the friction and actuator saturation, one fixed controller does not perform well for each setpoint. Hence, different PI controllers need to be designed for different setpoints. Gain scheduling is a very effective way of controlling systems whose dynamics change with the operating conditions.

For PI control design for the weigh belt feeder controllers were designed for setpoints of 1 volt, 2 volts, ..., 5 volts, where 1 volt corresponds to a belt speed of $5.08 \times 10^{-3} \text{m/sec}$ (1 ft/min). As mentioned above, 5 volts is the maximum possible value of the reference command. The initial set of candidate proportional and integral gains were both chosen within the range of $[0.1 \ 3.2]$ with the grid corresponding to 0.1.

A closed-loop data set, generated from the actual hardware, was used for the unfalsified control design process for each of the 5 setpoints. In particular, each data set was obtained by inserting an arbitrary PI controller for this setpoint, in this paper, the arbitrary PI controller used was $K_P = 1.0, K_I = 0.6$ for setpoint 1, 3, 4, 5, and $K_P = 1.4, K_I = 1.2$ for setpoint 2. The GA was used to perform the unfalsification. The parameter $\rho$ in (??) was chosen as $\rho = r_c/5$ where $r_c$ denotes the setpoint. This means that the constraint (??) is relaxed for a lower setpoint and tightened for a higher setpoint. The parameters $\rho_1$ and $\rho_2$ in the cost function (??) were chosen as $\rho_1 = 0.01$ and $\rho_2 = 0.03 + 0.005(r_c - 1)^2$.

A binary representation was used in the GA. The corresponding genetic operators were binary mutation operators and simple crossover operators. The stopping criterion was the user supplied maximum number of generations. A crossover rate of 0.7 and a mutation rate of 0.38 were used in the GA and each optimization process was implemented for 15 generations. The size of populations for the setpoints 1, 2, 3, 4, and 5 (volts) were respectively 36, 30, 18, 10 and 7. The reason for choosing different populations is that with increasing setpoint value, the subset of the initial set of controllers which satisfy the constraint (??) shrinks. Hence, to increase the efficiency of the GA, it was necessary to also shrink the population size.

The P and I gains obtained from the GA for “best” unfalsified controllers (i.e., the ones with the minimum costs (??)) are listed in Table ??.

Figures ?? to ?? are the corresponding step responses of the plant output and the control signal. Clearly for each of the 5 setpoints the control signals are within the range of saturation and the system outputs have desirable performance (i.e., no
Figure 5: Step Response for Setpoint=1 with Best PI Gain

Figure 6: Step Response for Setpoint=2 with Best PI Gain

Figure 7: Step Response for Setpoint=3 with Best PI Gain

Figure 8: Step Response for Setpoint=4 with Best PI Gain

Figure 9: Step Response for Setpoint=5 with Best PI Gain
<table>
<thead>
<tr>
<th>Setpoint</th>
<th>P gain</th>
<th>I gain</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.5</td>
<td>1.2</td>
</tr>
<tr>
<td>2</td>
<td>2.6</td>
<td>0.7</td>
</tr>
<tr>
<td>3</td>
<td>1.8</td>
<td>0.6</td>
</tr>
<tr>
<td>4</td>
<td>1.4</td>
<td>0.5</td>
</tr>
<tr>
<td>5</td>
<td>1.0</td>
<td>0.5</td>
</tr>
</tbody>
</table>

Table 1: The Best Unfalsified PI Controller for Different Setpoints

overshoot and fast rise times).

To show the necessity of gain scheduling for the controller design, the step response of the closed loop system with the best unfalsified controller obtained for a given setpoint is compared with the step response of the best unfalsified controller for another setpoint. Figures ?? to ?? are some examples of these comparisons, using experimental data. It is seen that when the best controller for a higher setpoint is implemented for a lower setpoint, the system has a slower response. Also, when the best controller obtained for a lower setpoint is implemented for a higher setpoint, the system has a significant overshoot. This phenomena is due to the nonlinearities in the weigh belt feeder dynamics.

![Figure 10: Performance Comparison, Setpoint=1](image1)

![Figure 11: Performance Comparison, Setpoint=2](image2)

6. Effectiveness of the Genetic Algorithm

As previously mentioned, a GA is chosen for the unfalsification because of its random search nature. In particular, it does not check the performance of each controller in the candidate set. To illustrate the effectiveness of the GA, we compared the above GA implementation with that of an exhaustive grid search method in which each candidate controller is checked before the best ones are obtained.
Figure 12: Performance Comparison, Setpoint=3  Figure 13: Performance Comparison, Setpoint=5

The above un falsification process was repeated by using the grid search method. Table ?? compares the computation time, the number of feasible points and the best controller obtained for the GA algorithm and exhaustive search method. In Table ?? an unfalsified point is a controller which satisfies the hard constraint. It is seen that although the GA computed a smaller number of unfalsified points, it yielded the same best unfalsified controller with less computation time for the setpoints from 1 to 4 (volts). For the 5 volt setpoint, the GA took slightly longer.

<table>
<thead>
<tr>
<th>Setpoint</th>
<th>GA Time(s)</th>
<th>No. Unfalsified Pts.</th>
<th>Best Point</th>
<th>Grid Search Time(s)</th>
<th>No. Unfalsified Pts.</th>
<th>Best Point</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>110.910</td>
<td>338</td>
<td>(1.5 1.2)</td>
<td>128.765</td>
<td>605</td>
<td>(1.5 1.2)</td>
</tr>
<tr>
<td>2</td>
<td>105.932</td>
<td>286</td>
<td>(2.6 0.7)</td>
<td>118.090</td>
<td>512</td>
<td>(2.6 0.7)</td>
</tr>
<tr>
<td>3</td>
<td>87.746</td>
<td>110</td>
<td>(1.8 0.6)</td>
<td>93.405</td>
<td>176</td>
<td>(1.8 0.6)</td>
</tr>
<tr>
<td>4</td>
<td>62.540</td>
<td>60</td>
<td>(1.4 0.5)</td>
<td>76.971</td>
<td>94</td>
<td>(1.4 0.5)</td>
</tr>
<tr>
<td>5</td>
<td>76.390</td>
<td>18</td>
<td>(1.0 0.5)</td>
<td>72.765</td>
<td>25</td>
<td>(1.0 0.5)</td>
</tr>
</tbody>
</table>

Table 2: Comparison of GA and Grid Search with Different Setpoint

Figures ?? to ?? are the comparison of regions of the controllers unfalsified by hard constraint and the best controllers obtained for 5 different setpoints by the GA and an exhaustive grid search method. It is clearly seen that due to the random search nature of GA, it does not check the performance of each candidate controller. Thus GA implementation has the ability to obtain the same best controller at each setpoint with less computation cost.

<table>
<thead>
<tr>
<th>P and I Region</th>
<th>GA Computation Time (sec)</th>
<th>Grid Search Computation Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0.1 3.2]</td>
<td>110.910</td>
<td>128.765</td>
</tr>
<tr>
<td>[0.1 6.4]</td>
<td>230.732</td>
<td>323.896</td>
</tr>
<tr>
<td>[0.1 12.8]</td>
<td>314.312</td>
<td>1194.700</td>
</tr>
</tbody>
</table>

Table 3: Comparison of GA and Grid Search with Different Initial Candidate Sizes
Figure 14: Unfalsified Regions for Setpoint=1: GA (left) and Grid Search (right)

Figure 15: Unfalsified Regions for Setpoint=2: GA (left) and Grid Search (right)

Figure 16: Unfalsified Regions for Setpoint=3: GA (left) and Grid Search (right)
The GA is expected to show its greatest advantage for controller designs with large initial candidate sets. To demonstrate this the GA was compared with the exhaustive search method for PI controller designs with candidate sets of different sizes. The setpoint was chosen as 1 volt. We chose the candidate set with proportional gain and integral gain within the range of [0.1 3.2], [0.1 6.4], [0.1 12.8] respectively assuming that the axes were gridded using increments of 0.1 for each axis. In all three cases the GA was implemented for 15 generations and with a population of 36. As shown in Table ??, as the initial candidate set size increased, the computational performance of the GA relative to the exhaustive search method increased. A large reason for this is that as the initial candidate size increases the size of the infeasible set increases and the GA does not test nearly as many infeasible points as the exhaustive search, which tests all of these points.

7. Conclusions

This paper developed an unfalsified control approach to PI tuning for a weigh belt feeder. A key was the construction of cost functions that reflect the engineering performance specifications.
It was seen that the unfalsified controller design problem may be considered a constrained optimization problem. To avoid searching over the entire candidate sets a genetic algorithm was used to implement the unfalsification procedure. The results clearly demonstrate the ability of the unfalsification procedure to yield high performing PI controllers. The results also demonstrated the ability of the GA to reduce the computational requirements of an exhaustive search, especially as the size of the initial candidate set increased.
References


