Handling control engineer preferences: getting the most of PI controllers

Gilberto Reynoso-Meza, Javier Sanchis, Xavier Blasco, Juan M. Herrero
Instituto Universitario de Automática e Informática Industrial
Universitat Politécnica de València
Valencia, España
gilreyme@posgrado.upv.es, {jsanchis,xblasco,juaherdu}@isa.upv.es

Abstract

By today, PI-PID controllers remain as reliable control solutions in a wide variety of industrial applications. Several tuning techniques have been designed and proposed successfully over the years. However the difficulty for tuning procedures increase as multiple requirements and criteria to evaluate the closed loop performance are demanded to be fulfilled. In this work, two approaches to incorporate control engineering preferences into an optimization statement for PI controller tuning are used. Their applicability is validated through simulation studies of well-known benchmark process.

Evolutionary Algorithms, PI controller tuning, Designer preferences, multiobjective optimization, global physical programming

1 Introduction

By today, PI and PID controllers remain as reliable digital control solutions, due to their simplicity and efficacy [3]. They are a common solution for industrial applications and therefore, there is still ongoing research on new techniques for robust PI and PID controller tuning.

New PI-PID controller tuning techniques mainly search for a trade-off solution between several control requirements as setpoint response, load rejection and robustness. The classical conflict between performance and robustness is well-known: an outstanding performance in the nominal model could lead to an unexpected performance in the real implementation.

Some approaches state the design problem as an optimization procedure [8, 23, 9, 2, 15]. From the control point of view it is usual to face a variety of specifications that do not only concern with some error measure, but with several requirements and specifications. Such problem, when multiple objectives to fulfill are required, is known as a multiobjective problem (MOP).

In a MOP, the designer (control engineer) has to deal with a list of requirements, and searches for a solution with a desired trade-off between objectives. As a Decision Maker (DM) has to design or translates such desired trade-off in the optimization problem at hand. A traditional approach to solve a MOP is to translate it into a single-objective problem using weighting factors, indicating the relative importance among objectives. These mechanisms for handling designer preferences in controller design strongly depends on which factors are used, and usually it is not a trivial task to select the right weighting vector to assure a quality solution with a reasonable trade-off among objectives [13].

More elaborated methods to incorporate the preference handling have been developed [10] such as lexicographic methods, goal programming and physical programming. However, new mechanisms and techniques are still required to handle preferences in a flexible and meaningful way. In this paper, two mechanisms for handling designer preferences will be presented, with their incorporation at PI controller tuning: global physical programming (GPP) and multiobjective optimization (MOO). The rest of this paper remains as follows: in section 2 an introduction to PI tuning by means of non-convex optimization is commented. In section 3 an overview on GPP and MOO is given. In section 4 the statement for PI controller tuning using both approaches is explained and validated in section 5. Finally, some concluding remarks are discussed.

2 PI tuning optimization statement

It will be used as guideline the non-convex optimization problem developed by [2] for PI controllers:

\[ G_c(s) = k_c \left( b + \frac{1}{T_i s} \right) R(s) - k_c \left( 1 + \frac{1}{T_i s} \right) Y(s) \] (1)

where \( k_c \) is the proportional gain, \( T_i \) the integral time and \( b \) the setpoint weighting. This optimization procedure is analytical and model oriented and it does not require time function computations. It is focused on getting a trade-off among load disturbance rejection, robustness and setpoint response. It defines as a parameter for design a given value of the maximum sensitivity function

\[ M_s = \max \left[ \frac{1}{1 + G_c(\omega) G_p(\omega)} \right] \in [1.2, 2.0], \omega \in [\omega_1, \omega_2] \]

and/or the maximum complementary sensitivity function
\[ M_p = \max \left| \frac{G_p(j\omega)}{1 + G_p(j\omega)} \right| \in [1.0, 1.5], \ \omega \in [\omega, \overline{\omega}], \]

where \( G_p(s) \) represents the process transfer function. The objective is to increase as much as possible the integral gain \( k_i = k_c/T_i \) subject to the pre-defined \( M_s \) and \( M_p \) values.

The original optimization statement to solve is:

\[
\max k_i = k_c/T_i \quad (2)
\]

subject to

\[
\begin{align*}
J(k_c, k_i, \omega) &\geq R^2 \quad (3) \\
\omega &\in [\omega, \overline{\omega}] \quad (4) \\
k_c &\in [k_{c_{1}}, k_{c_{2}}] \quad (5) \\
T_i &\in [T_{i_{1}}, T_{i_{2}}] \quad (6) \\
J(k_c, k_i, \omega) &= \left| C + \left( k_c - j_1 \frac{1}{\omega} \right) G(j\omega) \right|^2 \quad (7)
\end{align*}
\]

where \( C, R \) are the center and the radius respectively on the circle which serves as envelope for a desired value of \( M_s \) and \( M_p \) in the Nyquist diagram. As pointed in [2], to perform an optimization with desired values of \( M_s \) and \( M_p \) lead to a non-convex optimization, and thus the envelope using \( C, R \) is used. Parameter \( b \) is selected trying to keep \( M_{sp} = \max \left| \frac{k_{c_{1}} - j_1 \frac{1}{\omega}}{k_{c_{2}} - j_1 \frac{1}{\omega}} \right| \frac{G_p(j\omega)}{1 + G_p(j\omega)} \) as close to 1 as possible, in the resonance peak frequency. Stability margins are guarantee since \( M_s = 2.0 \) and \( M_p = 1.2 \) imply a gain margin \( g_m \geq 2.6 \) and a phase margin \( \varphi_m \geq 29^\circ, 50^\circ \) respectively [2].

3 Handling designer preferences

In a global framework, it can be stated that the DM has to look for a trade-off solution vector \( \theta \in \mathbb{R}^n \), where \( n \) is the number of decision variables, for \( M \) design objectives \( J_1(\theta), J_2(\theta), \ldots, J_M(\theta) \). Two methods for handling designer preferences can attain such problem: GPP and MOO techniques. The main differences between approaches relies in:

- GPP handles with designer preferences in a priori stage.
- MOO performs an analysis on preferences in a posteriori stage.

Next, a review on both techniques will be explained.

3.1 Global Physical Programming Review

GPP technique [13, 21] is based on Physical Programming [14], where the DM formulates his design objectives in an meaningful intuitive language structure. This is performed by means of class functions and range preferences to solve an aggregate function based on them. Whilst the original method makes an strong effort to built the proper class functions, with a set of requirements on convexity and continuity, GPP deals with the MOP in a more flexible way, by using evolutionary algorithms (EA's).

All settings of MOP and DM requirements become more transparent for the designer, who only needs to compute the objectives and to define the ranges of preference for each of them.

Let \( m \) the number of design objectives and let \( M \) the desired number of preference ranges that the designer wants to manage for the MOP. The images \( \alpha_k \) at the range boundaries \( J_k^\pm \) are calculated as:

\[
\begin{align*}
\alpha_0 &= 0 \\
\alpha_1 &= \alpha_{ini}, \ \alpha_{ini} \geq 0 \\
\alpha_k &= \alpha_{k-1} \cdot m \quad (1 < k \leq M)
\end{align*}
\]

All values are equal for all the objectives. Therefore, the intervals defined for each objective \( J_k(\theta) \) produce the same change image in their associated class function. Then, the following aggregate function is defined:

\[
J_{gpp}(\theta) = \sum_{q=1}^{m} \eta_q(\theta) \quad (11)
\]

where \( \eta_q(\theta) : \mathbb{R} \rightarrow \mathbb{R} \) is a function used to map \( J_q(\theta), q \in [1, \ldots, m] \) into its corresponding class function.

If constraints are considered, it is needed to penalize the unfeasible solutions. For this purpose, a modified cost index \( \bar{J}_{gpp}(\theta) \) is formed (according with [7]):

\[
\bar{J}_{gpp}(\theta) = \begin{cases} 
J_{gpp}(\theta) & \text{if } \sum_k \phi_k(\theta) = 0 \\
\text{offset} + \left( \sum_k \phi_k(\theta) \right) & \text{otherwise}
\end{cases}
\]

The constant offset is the minimum penalty for any unfeasible solution:

\[
\text{offset} = \begin{cases} 
\alpha_M, \ \exists \ k : \ \phi_k(\theta) \neq 0 \\
0, & \text{otherwise}
\end{cases}
\]

Note that with offset, all feasible solutions will have better \( \bar{J}_{gpp}(\theta) \) values than unfeasible ones; \( \sum_k \phi_k(\theta) \) represents the sum of all violation constraints incurred by the proposed solution vector \( \theta \).

3.2 Multiobjective optimization review

MOO can handle with MOP issues in a natural way, due to its simultaneous optimization approach. In MOO, all the objectives are significant from the designer point of view, and as consequence, each one is optimized, to get a set of optimal non-dominated solutions. The MOO approach offers to the designer a set of solutions, the Pareto set, where all the solutions are Pareto-optimal [10]. This set of solutions will offer to the DM a higher degree of
flexibility at the decision making stage. The role of the designer is to select the most preferable solution according to his needs and preferences for a particular situation.

A MOO problem, without loss of generality (since a maximization problem can be converted to a minimization problem), can be stated as follows:

$$\min_{\theta \in \mathbb{R}^m} J(\theta) = [J_1(\theta), \ldots, J_m(\theta)] \in \mathbb{R}^m$$ (12)

MOO techniques search for a discrete approximation $\Theta_\star_P$ of the Pareto set $\Theta_P$ capable of generate a good quality description of the Pareto front $J_P$. In this way, the DM has a set of solutions for a given problem and a higher degree of flexibility to choose a particular or desired solution.

Several algorithms used to calculate $J_P$ have been widely used, for instance the Normal Boundary Intersection method [6] or the Normal constraint method [1, 20, 12, 11], but lately multiobjective evolutionary algorithms (MOEA’s) have been used due to its flexibility to deal with non-convex and highly constrained functions [5].

The selection of a trade-off solution, according with the designer preferences, takes place in a posteriori analysis on the Pareto front $J_P$. In this work, to visualize the calculated front, the Level Diagram Tool (LDT) of [4]\(^1\) is used. The LDT helps the DM to perform an analysis on the obtained Pareto front $\Theta_P$, which is not a trivial task when the number of objectives is bigger than three. The LDT is a useful tool to analyze m-objective Pareto fronts [17], which is based on the classification of the $\Theta_P$ calculated. Each objective $J_q(\theta)$ is normalized with respect to its minimum and maximum values.

To each normalized objective vector $\hat{J}(\theta)$ a p-norm

$$||\hat{J}(\theta)||_p := \left(\sum_{q=1}^{m} ||\hat{J}(\theta)_q||^p\right)^{1/p}$$

is applied to evaluate the distance to an ideal solution $J^{ideal} = J^{min}$.

The LDT displays a graphical array for each objective $q \in [1, \ldots, m]$ and every decision variable $l \in [1, \ldots, n]$. The ordered pairs $\left(J_q(\theta), ||\hat{J}(\theta)||_p\right)$ in each objective sub-graphic and $\left(\theta_l, ||\hat{J}(\theta)||_p\right)$ in each decision variable sub-graphic are plotted. Therefore, a given solution will have the same $y$-value in all graphics (see figure 1). This correspondence will help to evaluate general tendencies along the Pareto front and to compare solutions according with the selected preference ranges. In all cases, the lower the norm, the closer to the ideal solution. For the remaining of this work the $\parallel \cdot \parallel_p$ norm will be used for the sake of simplicity. For a given Pareto front, the graphical array displayed by the LDT will be referred as level diagram (LD).

4 PI tuning optimization statement handling preferences

The MOP statement consists in to find a trade-off solution $\theta = [k_c, T_i, b]$ for the design objectives $J_1(\theta) = -k_c/T_i$, $J_2(\theta) = M_s$ and $J_3(\theta) = M_{sp}$.

For the GPP statement, it is required to establish the number of preference ranges $M$ and the $\alpha_{ini}$ value. For the remain of this work:

- $M = 5$ preference ranges.

\(^1\)Available at http://www.mathworks.com/matlabcentral/fileexchange/24042
• $\alpha_{ini} = 1/n^2 = 1/9$. With such selection, we have $\alpha_5 = 27$.
• As default parameters for $M_s$ and $M_p$ will be used: $\mathbf{J}_k = [1, 1.2, 1.4, 1.6, 1.8, 2.0, 10.0]$ and $\mathbf{J}_i = [1.0, 1.1, 1.3, 1.5, 2.0, 10.0]$, $k \in [1, 2, \ldots, 5]$.
• As default parameter for $k_i$ the preference ranges $\mathbf{J}_k = [-1/20, 1/20, 1/20, 1/20, 1/20, 0]$, $k \in [1, 2, \ldots, 5]$ are used; $\eta$ is the maximum value between 1 and the time delay process. This range selection guarantees to have an integral action effort lower than 1 before the system response is affected by the control action.

In each preference range, linear equations will be used. Therefore, GPP optimization statement takes the following form:

\[
\min_{\theta \in \mathbb{R}^n} J_{gpp}(\theta) = \begin{cases} 
\eta_1(\theta) + \eta_2(\theta) + \eta_3(\theta) & \text{if } \phi(\theta) = 0 \\
\text{offset} + \phi(\theta) & \text{otherwise}
\end{cases}
\]

subject to $k_c + \eta \cdot k_c/T_i \leq \gamma$, where

\[
\text{offset} = \begin{cases} 
27, & \exists \text{ if } \phi(\theta) \neq 0 \\
0, & \text{otherwise}
\end{cases}
\]

\[
\phi(\theta) = \max\{0, k_c + \eta \cdot k_c/T_i - \gamma\}
\]

For MOO the optimization statement takes the form:

\[
\min_{\theta \in \mathbb{R}^n} J(\theta) = \begin{cases} 
\mathbf{J}_k(\theta) & \text{if } \sum_{k=1}^{7} \phi_k(\theta) = 0 \\
\text{offset} + \left(\sum_{i=5}^{7} \phi_i(\theta)\right) \cdot R & \text{otherwise}
\end{cases}
\]

subject to $k_c + \eta \cdot k_c/T_i \leq \gamma$, $1.2 \leq M_s \leq 2.0$, $1.0 \leq M_{sp} \leq 1.5$, where

\[
\text{offset} = \begin{cases} 
\max(\mathbf{J}_k), & \exists \text{ if } \phi(\theta) \neq 0 \\
0, & \text{otherwise}
\end{cases}
\]

\[
\phi_1(\theta) = \max\{0, k_c + \eta \cdot k_c/T_i - \gamma\} \cdot [1, 1, 1]
\]

\[
\phi_2(\theta) = \max\{0, 1.2 - M_s\} \cdot [1, 1, 1]
\]

\[
\phi_3(\theta) = \max\{0, 1.0 - M_{sp}\} \cdot [1, 1, 1]
\]

\[
\phi_4(\theta) = \max\{0, M_s - 2.0\} \cdot [1, 1, 1]
\]

\[
\phi_5(\theta) = \max\{0, M_{sp} - 1.5\} \cdot [1, 1, 1]
\]

Constraint $k_c + \eta \cdot k_c/T_i \leq \gamma$ is used to bound the maximum allowed control action effort in both techniques. Constraints $1.2 \leq M_s \leq 2.0$ and $1.0 \leq M_{sp} \leq 1.5$ are used to get a Pareto front $J_p$ useful from the control point of view.

As evolutionary technique the Differential Evolution [22] algorithm is used. To obtain $J_p$, the sp-MODE algorithm is used, due to its performance in academic benchmarks for MOO algorithms [18] and its flexibility for control purposes [16, 19].

Table 1. Preferences for the example $G_1(s)$. Five preference ranges has been defined: highly desirable (HD), desirable (D), tolerable (T) undesirable (U) and highly undesirable (HU).

<table>
<thead>
<tr>
<th>Objective</th>
<th>Class</th>
<th>$g_1^0$</th>
<th>$g_1^1$</th>
<th>$g_1^2$</th>
<th>$g_1^3$</th>
<th>$g_1^4$</th>
<th>$g_1^5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_s$</td>
<td>1S</td>
<td>-1</td>
<td>-1/2</td>
<td>-1/3</td>
<td>-1/4</td>
<td>-1/5</td>
<td>0</td>
</tr>
<tr>
<td>$M_{sp}$</td>
<td>1S</td>
<td>1.2</td>
<td>1.4</td>
<td>1.6</td>
<td>1.8</td>
<td>2.0</td>
<td>10</td>
</tr>
</tbody>
</table>

Table 2. Solutions for the example $G_1(s)$.

<table>
<thead>
<tr>
<th>$k_c$</th>
<th>$T_i$</th>
<th>$b$</th>
<th>$k_i$</th>
<th>$M_s$</th>
<th>$M_{sp}$</th>
<th>$M_p$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.75</td>
<td>2.1473</td>
<td>0.0000</td>
<td>0.3376</td>
<td>1.4000</td>
<td>1.0000</td>
<td>1.0000</td>
</tr>
<tr>
<td>0.90</td>
<td>2.0550</td>
<td>0.0000</td>
<td>0.4691</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
</tr>
<tr>
<td>1.04</td>
<td>2.1600</td>
<td>0.0000</td>
<td>0.5800</td>
<td>1.6000</td>
<td>1.0000</td>
<td>1.2423</td>
</tr>
<tr>
<td>1.40</td>
<td>1.8900</td>
<td>0.0265</td>
<td>0.4250</td>
<td>1.5198</td>
<td>1.0000</td>
<td>1.0000</td>
</tr>
</tbody>
</table>

5 Examples

The first example under consideration is:

\[
G_1(s) = \frac{1}{(s + 1)^3}
\]

bound to:

\[
\omega \in [10^{-3}, 10^{3}]
\]

\[
k_c \in [0.01, 5]
\]

\[
T_i \in [\frac{1}{100}, 20]
\]

\[
b \in [0, 1]
\]

Three preference sets using GPP are used (table 1). In the first set, the preferences are stated with default parameters for $k_i$, $M_s$ and $M_{sp}$. The second and third preference set ask for better performance values of $k_i$. The obtained results are given at table 2.

The Pareto set $\Theta_p$ and Pareto front $J_p$ calculated by the MOO approach are shown in figure 2. As additional index, $I_1(\theta) = M_p$ is included in $J_p$ after the optimization stage, to have a bigger basis for comparison. With the graphical analysis on the LD it is possible to appreciate the level of trade-off achieved by each solution and to evaluate the tendencies along the Pareto front. For example,
Figure 2. Pareto set $\Theta^*_P$ and Pareto front $J^*_P$. Dark solutions match the requirement $J_1(\theta) = -k_i \leq -0.4$.

Figure 3. Pareto set $\Theta^*_P$ and Pareto front $J^*_P$. Dark solutions match the requirement $J_1(\theta) = -k_i \leq -0.04$. 
it is possible to see the classical conflict between performance and robustness: the lower \(-k_i\), the bigger \(M_s, M_p\) and \(M_{sp}\). Roughly speaking, in the range \(-k_i \leq -0.4\) it can be noticed that \(J_1(\theta), J_2(\theta)\) and \(I_1(\theta)\) have a rate \(\Delta J(\theta)\) of 0.3571, 0.4545 and 0.4545 respectively. That means that a unit change on the norm, makes an improvement on \(k_i\) around 0.35 units at exchange of getting a worst value on \(M_s\) and \(M_p\) around 0.45 units. The solutions obtained with the GPP approach are indicated to visualize their positions along the Pareto front.

In figure 5, the controller’s performances are shown. The results of the GPP approach are consistent with the class preference values previously defined. The solution with the lower \(\|J(\theta)\|_2\) value in \(\Theta^*_P\) is presented for comparison purposes.

![Figure 4](image1)

**Figure 4.** Performance attained by GPP and MOO on both examples.

The second example is a modification of \(G_1(s)\), but it incorporates a long time delay:

\[
G_2(s) = \frac{1}{(s + 1)^5}e^{-15s}
\]  

bound to

\[
\omega \in [10^{-3}, 10^3]\n\]

\[
k_c \in [0.01, 5]\n\]

\[
T_i \in [0.01, 20]\n\]

\[
b \in [0, 1]\n\]

Two preference sets using GPP are used (table 3). In the first set, the preferences are stated with default parameters. The second set is more exigent with the load disturbance rejection requisite. The obtained results are given at table 4 and their respective performance is shown in figure 5.

The Pareto set \(\Theta^*_P\) and Pareto front \(J^*_P\) obtained by the MOO technique are shown in figure 3. It is possible to appreciate the same tendencies as in the first example: the lower \(-k_i\), the bigger \(M_s\) and \(M_p\). The solutions obtained with the GPP approach are indicated to visualize their positions along the Pareto front.

Finally, at figure 5 and table 5 the GPP.B controller for both process are compared with the corresponding controller (denoted as APH) obtained by [2].

In general, it can be notice that,

- The solutions calculated by the GPP technique have a small improvement in overshoot and stabilizing time than the one computed by the original non-convex optimization statement.
- Whilst the GPP require a little more effort in the pre-optimization stage to design the preference class
functions, the MOO technique requires a further analysis over the Pareto front, to select a desired solution with a reasonable trade-off among objectives.

- One advantage with MOO is to have a set of solutions, which will be useful if the desired controller’s trade-off is continuously changing.
- GPP is more useful to retain an expert knowledge to automatize the selection procedure.

This last aspect brings GPP technique as a suitable solution for development of auto-tuning procedures.

6 Conclusions

Two preference handling schemes have been used for PI parameter tuning. One of the advantages of both techniques is the possibility to use any kind of optimization statement for parameter tuning.

GPP requires an a priori analysis, to determine the preference class functions which will determine the trade-off desired by the DM. The MOO techniques require an a posteriori analysis: the DM has to select the solution according with his needs at the end of the optimization stage. The MOO represents a solution when the DM requirements are constantly changing, whilst GPP is a suitable solution for auto-tuning aspects.

The use of the LDT allows to have an improved insight knowledge of the Pareto front, and in this particular case, of the closed loop response. With such graphical tool, it is possible to perform a global comparison among different controllers, by analyzing their relative positions in the Pareto front. Moreover, it is a flexible tool in the sense that it is possible to include any additional index (different from the ones used the optimization stage) to have a bigger basis for comparison and decision making.

Both approaches give consistent results in the performance evaluation. If time functions computations are possible, both scheme can use meaningful objectives from the DM point of view, such as the integral of the absolute error, the integral of the absolute change in control action, stabilization time, overshoot, among others.

Acknowledgment

This work was partially supported by the FPI-2010/19 grant from the Universitat Politècnica de València and the

<table>
<thead>
<tr>
<th>GPP ( G_1(s) )</th>
<th>Os</th>
<th>tss</th>
<th>( K_p )</th>
<th>( M_s )</th>
<th>( M_p )</th>
<th>( M_{sp} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPP.B ( G_1(s) )</td>
<td>0.13</td>
<td>12.8</td>
<td>0.4691</td>
<td>1.6000</td>
<td>1.0000</td>
<td>1.0460</td>
</tr>
<tr>
<td>APH ( G_1(s) )</td>
<td>12.44</td>
<td>12.0</td>
<td>0.4610</td>
<td>1.6000</td>
<td>1.0000</td>
<td>1.0500</td>
</tr>
<tr>
<td>GPP.B ( G_2(s) )</td>
<td>6.01</td>
<td>61</td>
<td>0.0426</td>
<td>1.6000</td>
<td>1.0000</td>
<td>1.0000</td>
</tr>
<tr>
<td>APH ( G_2(s) )</td>
<td>7.38</td>
<td>66</td>
<td>0.0425</td>
<td>1.6100</td>
<td>1.0000</td>
<td>1.0041</td>
</tr>
</tbody>
</table>

Table 5. Performance comparison between GPP and APH. Overshoot (Os) and stabilizing time (tss) for setpoint response are included as performance index.

Figure 5. Performance comparison between GPP.B and APH controllers.
 References


