Evolutionary auto-tuning algorithm for PID controllers ⋆

Gilberto Reynoso-Meza, Javier Sanchis, Juan M. Herrero, César Ramos

Instituto Universitario de Automática e Informática Industrial
Universidad Politécnica de Valencia, Valencia, España
(e-mail: gilreyme@posgrado.upv.es,
{jsanchis,juaherdu,cramos}@isa.upv.es).

Abstract: In this work, an auto-tuning procedure for PID controllers is presented. This auto-tuning procedure identifies a FOPDT model for a given process and uses an evolutionary algorithm to solve a constrained non-convex optimization problem to adjust the parameters of a PID controller with derivative filter. The auto-tuning procedure is validated with a set of process with different characteristics. Presented results validate the auto-tuning algorithm as a practical and viable application for auto-tuning procedures.

Keywords: PID auto-tuning, PID controllers, evolutionary algorithms.

1. INTRODUCTION

PI-PID controller remains as a reliable and practical control solution for several industrial processes. Owing to this, research for new tuning techniques is an ongoing research topic (see for example Ge et al. (2002); Toscano (2005); Goncalves et al. (2008); Astrom et al. (1998); Panagopoulos et al. (2002); Alfaro (2007)). Current research points to guarantee reasonable stability margins as well as a good overall performance for a wide variety of process. One of the main advantages of PI-PID controllers is their ease of implementation as well as their tuning, giving a good trade-off between simplicity and cost to implement (Tan et al. (2004); Stewart and Samad (2011)).

Auto-tuning routines for PI-PID commercial controllers are a highly desirable characteristic for the industry. With such applications, it is possible to reduce the amount of time required to adjust the controller parameters for a given process, as well as the expertise required for such task. Because of this fact, several commercial products include auto-tuning procedures. Nevertheless, in several circumstances those auto-tuning procedures give as result a poor performance (Greg Baker (2009)) or guarantee a good performance under particular circumstances. For this reason, new auto-tuning techniques to face a wide variety of process are focus of attention for researchers.

In this work, we shown and implement an on-line auto-tuning algorithm for PID controllers based on evolutionary algorithms (EvoTune in the following). The EvoTune algorithm solve a constrained non-convex optimization statement for an identified FOPDT process to adjust the parameters of the PID controller. The remain of this work is as follows: in section 2, a brief problem description is given. In section 3, the optimization problem is stated whilst in section 4 the EvoTune algorithm is explained. In section 5, the EvoTune algorithm is evaluated over a set of benchmark processes. Finally, some concluding remarks are given.

⋆ This work was partially supported by the projects ENE2011-25900 and TIN2011-28082 from the Ministerio de Ciencia e Innovación, Gobierno de España and the Universitat Politécnica de València by the project PAID-06-11 and the FPI-UPV 2010/19 grant.

2. PROBLEM DESCRIPTION

Here, we deal with the auto-tuning procedure of a PID controller with derivative filter:

$$G_c(s) = k_c \left(1 + \frac{1}{T_i s} + \frac{T_d s}{\frac{T_d}{N} s + 1}\right) E(s) \qquad (1)$$

where k_c is the proportional gain, T_i the integral time (sec.), T_d the derivative time (sec.), N the derivative filter and $E(s)$ the error signal.

The desired characteristics for the auto-tuning procedure are:

- The algorithm must be executed in each sampling time.
- The algorithm must include an experiment phase to get information from the process.
- The algorithm gets two measurements from the process in each sampling time: time (since the beginning of the experiment phase) and the measurement from the process.

To test the auto-tuning procedure, the Simulink© platform developed by Romero and Sanchis (2010, 2011) is used. It is a Simulink© platform which allows to simulate, the auto-tuning procedure for a given process, with noise in the measurement. It uses a sampling time of 0.1 secs.

3. PROBLEM STATEMENT

As guideline, it is used the non-convex optimization statement developed in Astrom et al. (1998) and Panagopoulos et al. (2002). This optimization problem has been solved using evolutionary algorithms (Reynoso-Meza et al. (2011a,b)). Nevertheless, it has been not implemented yet for auto-tuning procedure for on-line applications.

Given a process $G_p(s)$, we look for a set of PID parameters $\boldsymbol{x} = \{k_c, T_i, T_d, N\}$ which maximizes the integral gain (or its equivalent):

$$\min_{\boldsymbol{x}\in\Re^4} J(\boldsymbol{x}) = -\frac{k_c}{T_i} \tag{2}$$

subject to a maximum value of the sensitivity function. As noticed in Astrom et al. (1998), this is a non-convex optimization statement. Evolutionary algorithms are reliable tools when controller tuning procedure is stated as an optimization problem (Fleming and Purshouse (2002)). Evolutionary algorithms are global optimizers, capable of deal with non-lineal, non-convex and highly constrained problems. They have been used with success for PI-PID tuning (Herreros et al. (2002); Tavakoli et al. (2007); Kim et al. (2008); Nobakhti and Wang (2008); Iruthayarajan and Baskar (2009); Xue et al. (2010); Iruthayarajan and Baskar (2010); Zhao et al. (2011); Reynoso-Meza et al. (2012b,a)). Owing to this, they are used in this auto-tuning procedure.

3.1 Constraints

Three main constraints are included to guarantee a good overall performance:

Constraint 1 To guarantee an acceptable stability margin, it is used the maximum value of the sensitivity function $M_s = \max|S(\mathrm{j}\omega)| = \max\left|\frac{1}{1+G_p(\mathrm{j}\omega)G_c(\mathrm{j}\omega)}\right|$. As a rule of thumb, accepted values for M_s are in the interval $M_s \in [1.2, 2.0]$ (Astrom et al. (1998); Panagopoulos et al. (2002)). The auto-tuning procedure uses a maximum value for M_s, M_s^{target} defined by the user. That is,

$$M_s \leq M_S^{target} \tag{3}$$

Computation of M_s could require a high computational load. If we considerer a FOPDT process:

$$G_p(s) = \frac{k_p}{Ts+1}\mathfrak{e}^{-Ls} \tag{4}$$

the following simplifications arises:

$$\begin{aligned} S(\mathrm{j}\omega) &= \frac{1}{1+G_c(\mathrm{j}\omega)G_p(\mathrm{j}\omega)} \\ &\approx \frac{(a\omega^4+b\omega^2)+(c\omega^3+d\omega)\mathrm{j}}{(e\omega^4+f\omega^2+g\omega^0)+(h\omega^3+i\omega)\mathrm{j}} \end{aligned} \tag{5}$$

where

$$a = \frac{T_i\cdot T_d\cdot L}{2N}$$

$$b = -T_i\left(T+\frac{T_d}{N}+\frac{L}{2}\right)$$

$$c = -\frac{T_i\cdot T_d\cdot T}{N} - T_i\cdot L\left(T+\frac{T_d}{N}\right)$$

$$d = T_i$$

$$e = \frac{T_i\cdot T_d\cdot T\cdot L}{2N}$$

$$\begin{aligned} f = &-T_i\left(T+\frac{T_d}{N}+\frac{L}{2}\right) \\ &-k_c\cdot k_p\cdot T_i\cdot T_d\left(\frac{1}{N}+1\right) \\ &+\frac{k_c\cdot k_p\cdot L}{2}\left(T_i+\frac{T_d}{N}\right) \end{aligned}$$

$$g = k_c\cdot k_p$$

$$\begin{aligned} h = &-T_i\left(T+\frac{T_d}{N}+\frac{L}{2}\right) \\ &+\frac{k_c\cdot k_p\cdot L\cdot T_i\cdot T_d}{2}\left(\frac{1}{N}+1\right) \end{aligned}$$

$$\begin{aligned} i = &T_i + k_c\cdot k_p\left(T_i+\frac{T_d}{N}\right) \\ &-\frac{k_c\cdot k_p\cdot L}{2} \end{aligned}$$

where a Padé approximation of first order $\mathfrak{e}^{-Ls} \approx \frac{1-\frac{L}{2}s}{1+\frac{L}{2}s}$ is used.

Constraint 2 Constraint number two is used to bound the maximum allowable control action of the PID controller in each sampling time:

$$k_c + \max(T_s, L)\cdot k_c/T_i + k_c\cdot T_d \leq K_u \tag{6}$$

where K_u is the estimated ultimate gain of the process $G_p(s)$ and T_s the sampling time.

Constraint 3 Constraint number 3 is used to consider the measurement noise into the optimization statement and its effect in the control action. A useful indicator is:

$$M_u = \max\left|\frac{G_c(\mathrm{j}\omega)}{1+G_c(\mathrm{j}\omega)G_p(\mathrm{j}\omega)}\right| \tag{7}$$

To avoid M_u computation, the following approximation is used (Åström and Hägglund (2005)):

$$M_u \approx k_c\cdot N \tag{8}$$

To bound such value, it is used the constraint $M_u \approx k_c\cdot N \leq K_u$.

3.2 Optimization statement

The constrained optimization statement proposed is:

$$\min_{\boldsymbol{x}\in\Re^4}\mathfrak{J}(\boldsymbol{x}) = \begin{cases} J(\boldsymbol{x}) & \textbf{\textit{if}} \sum_{k=1}^{3}\phi_k(\boldsymbol{x}) = 0 \\ \sum_{k=1}^{3}\phi_k(\boldsymbol{x}) & \textbf{\textit{otherwise}} \end{cases} \tag{9}$$

where:

$$\phi_1(\boldsymbol{x}) = \frac{\max\{0, M_s - M_s^{target}\}}{M_s^{target}} \tag{10}$$

$$\phi_2(\boldsymbol{x}) = \frac{\max\{0, k_c + \max(T_s, L)\cdot\frac{k_c}{T_i} + k_c\cdot T_d - K_u\}}{K_u} \tag{11}$$

$$\phi_3(\boldsymbol{x}) = \frac{\max\{0, k_c\cdot N - K_u\}}{K_u} \tag{12}$$

Next, the auto-tuning algorithm is explained.

4. AUTO-TUNING ALGORITHM

Given the last optimization statement, the EvoTune (Algorithm I) is defined. It has five main steps: noise estimation, experiment phase, identification step, optimization initialization and optimization process.

Algorithm I: *EvoTune*

```
Read input variable time.
Read input variable measurement.
IF time == 0
  Initialize variables.
END IF
IF Noise estimation process == TRUE
  Execute algorithm Noise.Estimation.
END IF
IF Experimentation process == TRUE
  Execute algorithm Biased.Feedback.Rele.
END IF
IF Identification process == TRUE
  Identify FOPDT process.
END IF
IF Optimization initialization process == TRUE
  Generate initial population P|_0 with known tuning rules and random individuals.
END IF
IF Optimization process == TRUE
  Read generation counter G.
  Execute algorithm DE.Build.Offspring to generate the offspring O|_G with P|_G.
  Evaluate offspring O|_G.
  Execute algorithm Population.Update to update population P|_G.
  Look for best solution found so far x^best|_G from P|_G
  G = G + 1.
END IF
IF Optimization process == END
  Algorithm terminates. x^best|_G is the vector with the tuning parameters.
END IF
```

4.1 Noise estimation

Through a pre-defined time interval (with the stable system), it will be recorded the measurement signal from the process to estimate the noise amplitude. This is necessary for the adequate selection of the histeresis range and the amplitude of the biased feedback rele test.

4.2 Experiment phase

It is used a biased feedback rele test (Hang et al. (2002); Wang et al. (1997)) to get information from the process for the model identification step.

4.3 Model identification

With the obtained data in the experiment stage, a FOPDT process is identified according to the biased rele feedback test (Hang et al. (2002); Wang et al. (1997)).

4.4 Optimization process initialization

The evolutionary optimization process requires an initial vector population. To this purpose, a random set of solutions is generated. To improve convergence properties, in the initial population are included a set of well-known tuning rules based on a FOPDT model: IMC, Lambda and AMIGO.

4.5 Constrained non-convex optimization

Through the optimization process, two main task are performed: the *offspring* generation and the selection of the best solution found so far.

Offspring generation As evolutionary technique it is used the Differential Evolution algorithm (DE) of Storn and Price (1997); Storn (2008); Das and Suganthan (2010). DE algorithm is used mainly due to its simplicity and compactness. This algorithm has two main operators to generate the offspring: mutation and crossover. They are several primal versions of the algorithm but in this work, we use the version known as DE/*current-to-rand*/2 (Price (1999)) which uses a non-invariant rotationally lineal recombination strategy in the crossover operator.

Mutation For each decision vector $\boldsymbol{x}^i|_k$ (parent vector), it is calculated a mutant vector $\boldsymbol{v}^i|_k$ in each generation k according to the equation (13):

$$\boldsymbol{v}^i|_k = \boldsymbol{x}^{r_1}|_k + F(\boldsymbol{x}^{r_2}|_k - \boldsymbol{x}^{r_3}|_k) \tag{13}$$

where $r_1 \neq r_2 \neq r_3 \neq i$ and F is known as the scaling factor.

Crossover (Lineal recombination) For each parent vector in the decision space $\boldsymbol{x}^i|_k$ and its mutant vector $\boldsymbol{v}^i|_k$, it is calculated a trial vector (child vector) $\boldsymbol{u}^i|_k = [u_1^i|_k, u_2^i|_k, \ldots, u_n^i|_k]$:

$$\boldsymbol{u}^i|_k = \boldsymbol{x}^i|_k + F_i(\boldsymbol{v}^i|_k - \boldsymbol{x}^i|_k) \tag{14}$$

where F_i is the scaling factor for the lineal recombination.

Selection As selection mechanism, a pair comparison between parent and child vector is performed. The best individual is selected to move into the next generation:

$$\boldsymbol{x}^i|_{k+1} = \begin{cases} \boldsymbol{u}^i|_k \ if \ f(\boldsymbol{u}^i|_k) \leq f(\boldsymbol{x}^i|_k) \\ \boldsymbol{x}^i|_k \quad otherwise \end{cases} \tag{15}$$

In each execution of the algorithm, during the optimization stage, the best controller found so far is selected. Such controller is used to calculate the control action to the system. When the convergence criteria is reached, the algorithm will terminate and the final PID controller will be given. Next, the EvoTune is validated over a set of different processes, to shown its viability.

5. TEST

A set of well-known processes is used to validate the auto-tuning procedure:

$$G_1(s) = \frac{1}{(s+1)^3}, G_2(s) = \frac{\mathfrak{e}^{-5s}}{(s+1)^3}$$

$$G_3(s) = \frac{1}{(1+10s)(1+2s)(1+\frac{4}{10}s)(1+\frac{8}{100}s)}$$

$$G_{4,5,6,7} = \frac{1}{(s+1)^\alpha}, \alpha = 4, 5, 6, 7$$

$$G_8(s) = \frac{-2(s-2)}{(s+1)^3}$$

The parameters used by the auto-tuning procedure are:

- Scaling factor $F = 0.9$. With this value, according to Storn (2008), exploration in the decision space is promoted.
- Scaling factor for the lineal recombination $F_i = 0.95$. According to Price (1999), a good initial choice for F_i is $F_i = 0.5 \cdot (1 + F)$.
- Population size: $|P| = 10 \cdot 4 = 40$ (Storn and Price (1997)). As a rule of thumb, it is recommended to use a population ten times the number of decision variables. To guarantee the on-line execution, in each sampling time five individuals are evaluated.

- Upper bound: $[\overline{k_c}, \overline{T_i}, \overline{T_d}, \overline{N}] = [K_u, 2 \cdot T, 2 \cdot L, 20]$.
- Lower bound: $[\underline{k_c}, \underline{T_i}, \underline{T_d}, \underline{N}] = [0, 0.1 \cdot T, 0, 3]$.
- Frequency range (with a little of notation abuse) $[\omega = 1/100 : 1/25 : 8]$.
- Histéresis $= max\{0.01, 2 \cdot NoiseAmplitude\}$
- $M_s^{target} = 1.4, 1.6, 1.8$.

5.1 Results and discussions

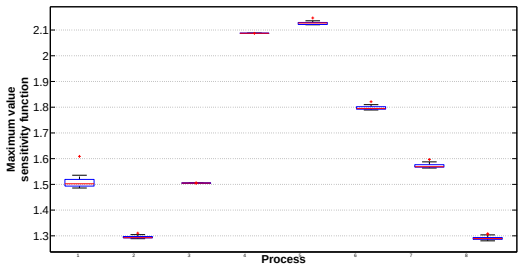
Given the stochastic nature of the auto-tuning procedure, a statistical analysis is proposed to shown its viability and robustness. A total of 25 independent trials for each process with $M_s^{target} = [1.4, 1.6, 1.8]$ are performed. In tables 1 to 5 are presented the median value of the integral of the absolute value of the control action (IADU), the integral of the absolute error (IAE), the maximum deviation from reference, the M_s with the simulated model and the time required to adjust the parameters, respectively. Also, a Ziegler-Nichols (ZN) tuning (based on ultimate gain identification) and the AMIGO tuning rules (using an estimated FOPDT as described in section 4.3) with derivative filter of $N = 30$ and $N = 3$ are presented for comparison purposes. In table 1, it is possible to notice how the EvoTune has a better performance than the ZN and AMIGO procedures with $N = 30$. Compared with AMIGO at $N = 3$ has almost the same performance. Nevertheless, the inclusion of the derivative filter leads to a change in M_s (see table 4); as notice in Isaksson and Graebe (2002), derivative filter must be considered an integral part on the PID design. In the case of the EvoTune procedure, it has a $M_s > 2.0$ for models $G_4(s)$ and $G_5(s)$ in all cases. Due to space limitation, boxplot graphs for M_s are presented in figure 1 to show its dispersion. The corresponding trade-off among approaches can be appreciated in table 2: the less the stability margins and the most the control action, the better the IAE performance. In the case of the maximum deviation, all controllers have almost the same performance. Finally, in table 5, the time required to adjust the parameters in each case are shown. Due to space limitation, just the graphical performance comparison for processes $G_1(s)$, $G_2(s)$, $G_3(s)$ and $G_8(s)$ with ZN procedure are shown in figures 2-5.

6. CONCLUSIONS

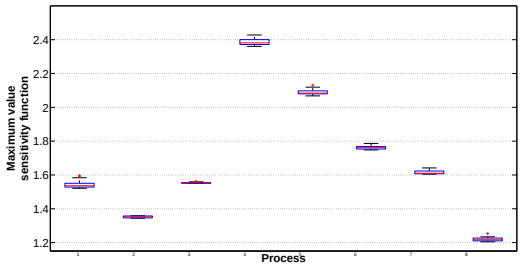
In this work, an auto-tuning procedure using evolutionary algorithms (EvoTune) has been presented. The EvoTune states a constrained non-convex optimization problem to adjust the proportional gain, the integral time, the derivative time and the derivative filter of a PID controller. The EvoTune includes an identification phase, to approximate a FOPDT model with a biased rele feedback test.
The EvoTune mechanism is robust, and is capable of adjust different kind of process with characteristics as multiple poles, long time delay, non-minimum phase, etc. When compared with other tuning techniques (ZN and AMIGO), is more robust, since it considers the derivative filter in the tuning procedure. It allows to control a bigger subset of processes with better stability margins. Ongoing work points to real-time testing as well as preference articulation for the user.

REFERENCES

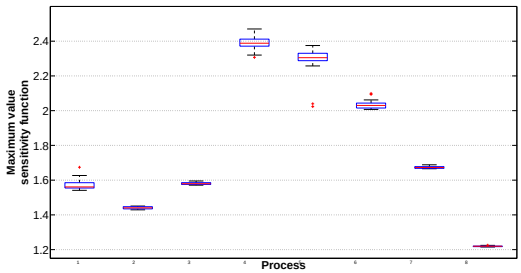
Alfaro, V.M. (2007). Pid controller's fragility. *ISA Transactions*, 46(4), 555 – 559. doi:10.1016/j.isatra.2007.03.006.



(a) Ms=1.4



(b) Ms=1.6



(c) Ms=1.8

Fig. 1. Maximum value sensitivity function dispersion.

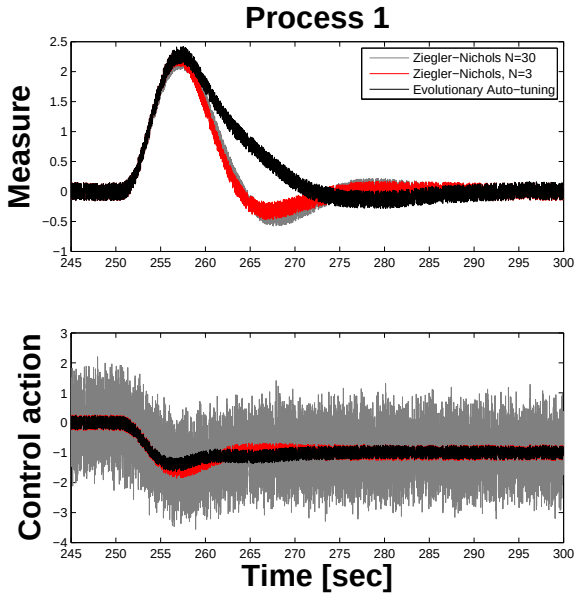


Fig. 2. Performance comparison for process 1.

Table 1. IADU performance ($1e3$).

Process	ZN		AMIGO		M_s^{target}		
	N=30	N=3	N=30	N=3	1.4	1.6	1.8
1	24.30	3.43	10.93	2.13	2.16	2.15	2.22
2	26.03	3.59	7.39	1.46	1.45	1.34	1.15
3	05.27	0.73	8.49	1.70	1.87	1.95	1.95
4	17.99	2.65	13.25	2.67	2.94	3.01	2.86
5	19.12	2.74	12.70	2.52	2.72	2.69	2.64
6	22.43	3.15	13.17	2.59	2.69	2.64	2.72
7	23.63	3.28	11.62	2.27	2.37	2.38	2.35
8	23.36	3.40	6.52	1.29	1.18	0.98	0.83

Table 2. IAE performance ($1e3$).

Process	ZN		AMIGO		M_s^{target}		
	N=30	N=3	N=30	N=3	1.4	1.6	1.8
1	3.20	3.09	8.13	8.14	7.66	6.17	5.41
2	1.99	1.93	2.79	2.79	2.67	2.37	2.20
3	0.33	0.33	0.22	0.22	0.15	0.13	0.12
4	0.55	0.56	0.84	0.84	0.66	0.58	0.55
5	0.75	0.75	1.09	1.09	0.93	0.85	0.78
6	0.99	1.00	1.42	1.42	1.34	1.20	1.09
7	1.21	1.21	1.77	1.77	1.78	1.55	1.42
8	2.34	1.95	3.50	3.51	3.27	2.93	2.74

Table 3. Maximum deviation performance

Process	ZN		AMIGO		M_s^{target}		
	N=30	N=3	N=30	N=3	1.4	1.6	1.8
1	2.32	2.39	2.92	2.84	3.46	3.33	3.28
2	1.03	1.04	1.04	1.05	1.05	1.05	1.06
3	0.21	0.21	0.12	0.12	0.11	0.10	0.10
4	0.45	0.46	0.43	0.44	0.43	0.41	0.41
5	0.54	0.55	0.51	0.54	0.55	0.55	0.54
6	0.62	0.63	0.62	0.64	0.67	0.67	0.66
7	0.68	0.70	0.70	0.73	0.76	0.76	0.76
8	1.02	1.14	0.93	0.97	0.99	1.03	1.06

Table 4. Maximum value sensitivity function performance

Process	ZN		AMIGO		M_s^{target}		
	N=30	N=3	N=30	N=3	1.4	1.6	1.8
1	1.83	2.15	1.53	2.06	1.50	1.54	1.56
2	2.77	3.64	1.39	1.47	1.29	1.35	1.44
3	1.10	1.24	1.24	1.52	1.51	1.55	1.58
4	1.52	1.78	1.57	2.12	2.09	2.38	2.39
5	1.68	2.03	1.82	2.41	2.13	2.08	2.30
6	1.81	2.17	1.88	2.43	1.79	1.76	2.03
7	1.89	2.31	1.80	2.19	1.57	1.61	1.67
8	2.14	2.68	1.49	1.53	1.29	1.22	1.22

Table 5. Time performance (secs) for experiment phase and optimization process.

Process	ZN		AMIGO		M_s^{target}		
	N=30	N=3	N=30	N=3	1.4	1.6	1.8
1	21.95	21.95	23.95	23.95	34.00	34.34	34.27
2	22.12	22.12	31.31	31.31	42.19	42.06	41.05
3	20.32	20.32	17.61	17.61	39.02	35.88	35.12
4	9.81	9.81	15.15	15.15	33.48	30.53	29.86
5	12.68	12.68	18.59	18.59	33.66	32.29	32.95
6	15.54	15.54	22.76	22.76	36.05	35.45	36.14
7	18.39	18.39	26.59	26.59	38.85	38.56	37.94
8	11.56	11.56	24.62	24.62	35.48	34.38	33.87

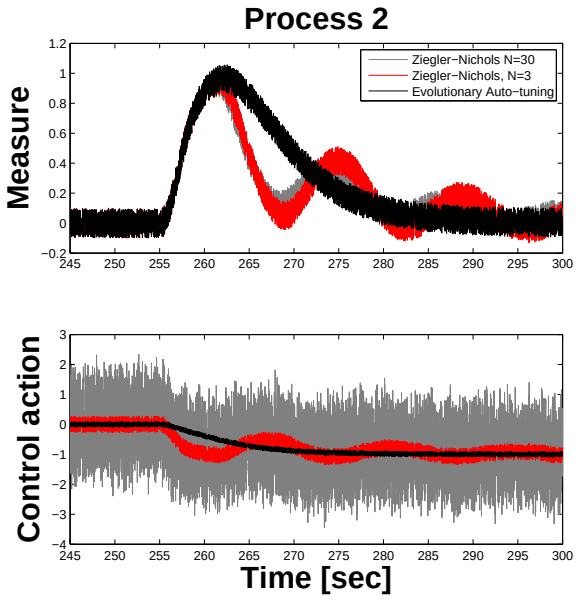


Fig. 3. Performance comparison for process 2.

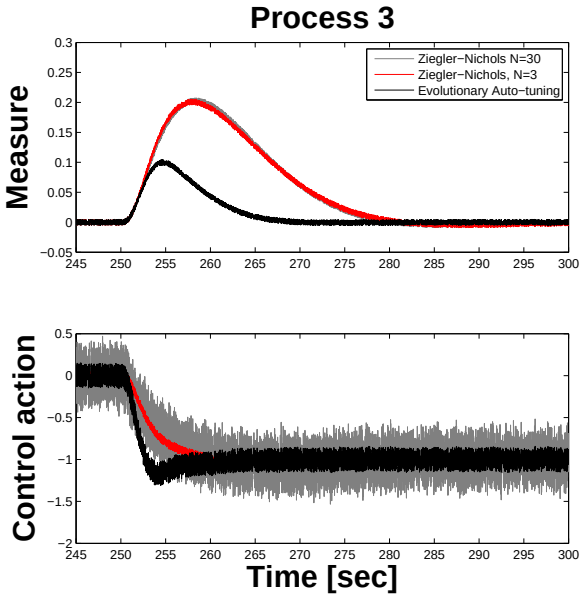


Fig. 4. Performance comparison for process 3.

Åström, K.J. and Hägglund, T. (2005). *Advanced PID Control*. ISA - The Instrumentation, Systems, and Automation Society, Research Triangle Park, NC 27709.

Astrom, K., Panagopoulos, H., and Hagglund, T. (1998). Design of pi controllers based on non-convex optimization. *Automatica*, 34(5), 585 – 601. doi:DOI:10.1016/S0005-1098(98)00011-9.

Das, S. and Suganthan, P.N. (2010). Differential evolution: A survey of the state-of-the-art. *Evolutionary Computation, IEEE Transactions on*.

Fleming, P. and Purshouse, R. (2002). Evolutionary algorithms in control systems engineering: a survey. *Control Engineering Practice*, (10), 1223 – 1241.

Ge, M., Chiu, M.S., and Wang, Q.G. (2002). Robust pid controller design via lmi approach. *Journal of process control*, (12), 3 – 13.

Goncalves, E.N., Palhares, R.M., and Takahashi, R.H. (2008). A novel approach for h_2/h_∞ robust pid synthesis for uncertain systems. *Journal of process control*, (18), 19 – 26.

Greg Baker, W. (2009). Is automated pid tuning dependable? Avilable on-line at `http://www.`

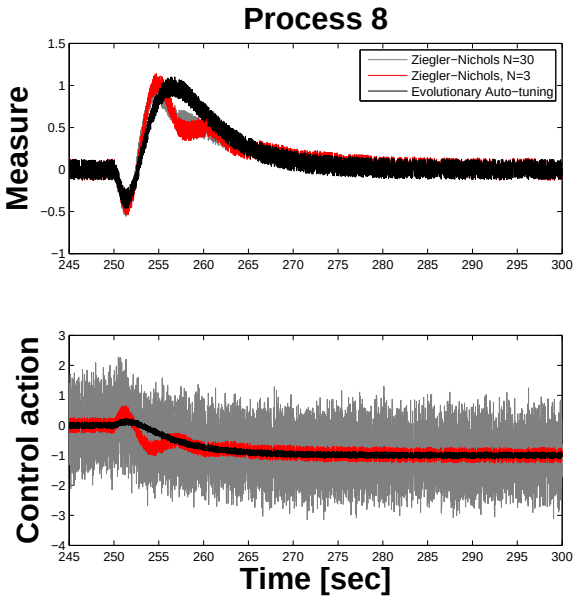


Fig. 5. Performance comparison for process 8.

controleng.com/index.php?id=483&cHash=081010&tx_ttnews[tt_news]=12682.

Hang, C.C., Astrom, K.J., and Wang, Q.G. (2002). Relay feedback auto-tuning of process controllers – a tutorial review. *Journal of Process Control*, 12(1), 143 – 162. doi:DOI:10.1016/S0959-1524(01)00025-7.

Herreros, A., Baeyens, E., and Perán, J.R. (2002). Design of pid-type controllers using multiobjective genetic algorithms. *ISA Transactions*, 41(4), 457 – 472. doi:DOI:10.1016/S0019-0578(07)60102-5.

Iruthayarajan, M.W. and Baskar, S. (2009). Evolutionary algorithms based design of multivariable pid controller. *Expert Systems with applications*, 3(36), 9159 – 9167.

Iruthayarajan, M.W. and Baskar, S. (2010). Covariance matrix adaptation evolution strategy based design of centralized pid controller. *Expert Systems with Applications*, 37(8), 5775 – 5781. doi:DOI:10.1016/j.eswa.2010.02.031.

Isaksson, A. and Graebe, S. (2002). Derivative filter is an integral part of pid design. *Control Theory and Applications, IEE Proceedings* -, 149(1), 41 –45. doi:10.1049/ip-cta:2002011.

Kim, T.H., Maruta, I., and Sugie, T. (2008). Robust pid controller tuning based on the constrained particle swarm optimization. *Automatica*, 44(4), 1104 – 1110. doi:DOI:10.1016/j.automatica.2007.08.017.

Nobakhti, A. and Wang, H. (2008). A simple self-adaptive differential evolution algorithm with application on the alstom gasifier. *Applied soft computing*, 8, 350 – 370.

Panagopoulos, H., Astrom, K., and Hagglund, T. (2002). Design of pid controllers based on constrained optimisation. *Control Theory and Applications, IEE Proceedings* -, 149(1), 32 – 40. doi:10.1049/ip-cta:20020102.

Price, K.V. (1999). *An introduction to differential evolution*, 79 – 108. McGraw-Hill Ltd., UK, Maidenhead, UK, England.

Reynoso-Meza, G., Sanchis, J., Blasco, X., and Herrero, J. (2011a). Handling control engineer preferences: Getting the most of pi controllers. In *Emerging Technologies Factory Automation (ETFA), 2011 IEEE 16th Conference on*, 1 –8. doi:10.1109/ETFA.2011.6059074.

Reynoso-Meza, G., Sanchis, J., Blasco, X., and Martinez, M. (2011b). An empirical study on parameter selection for multiobjective optimization algorithms using differential evolution. In *Differential Evolution (SDE), 2011 IEEE Symposium on*, 1 –7. doi:10.1109/SDE.2011.5952055.

Reynoso-Meza, G., García-Nieto, S., Sanchis, J., and Blasco, X. (2012a). Controller tuning using multiobjective optimization algorithms: a global tuning framework. *IEEE Transactions on Control Systems*, Article in press. doi:10.1109/TCST.2012.2185698.

Reynoso-Meza, G., Sanchis, J., Blasco, X., and Juan M., H. (2012b). Multiobjective evolutionary algortihms for multivariable pi controller tuning. *Expert Systems with Applications*, Article in press. doi:10.1016/j.eswa.2012.01.111.

Romero, J.A. and Sanchis, R. (2010). Benchmark 2010-2011 grupo temático de ingeniería de control de cea: Evaluación de auto-ajuste de controladores pid. Avalaible at `http://www.ceautomatica.es/og/ingenieria-de-control/benchmark-2010-11`.

Romero, J.A. and Sanchis, R. (2011). Benchmark para la evaluación de algoritmos de auto-ajuste de controladores pid. *Revista Iberoamericana de Automática e Informática Industrial*, 8(1), 112 – 117.

Stewart, G. and Samad, T. (2011). Cross-application perspectives: Application and market requirements. In T. Samad and A. Annaswamy (eds.), *The Impact of Control Technology*, 95 – 100. IEEE Control SystemsSociety.

Storn, R. (2008). Sci: Differential evolution research: Trends and open questions. volume LNCS 143, 1 – 31. Springer, Heidelberg.

Storn, R. and Price, K. (1997). Differential evolution: A simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11, 341 – 359.

Tan, W., Liu, J., Fang, F., and Chen, Y. (2004). Tuning of pid controllers for boiler-turbine units. *ISA Transactions*, 43(4), 571 – 583. doi:10.1016/S0019-0578(07)60169-4.

Tavakoli, S., Griffin, I., and Fleming, P.J. (2007). Multi-objective optimization approach to the pi tuning problem. In *Proceedings of the IEEE congress on evolutionary computation (CEC2007)*, 3165 – 3171.

Toscano, R. (2005). A simple robust pi/pid controller design via numerical optimization approach. *Journal of process control*, (15), 81 – 88.

Wang, Q.G., Hang, C.C., and Zou, B. (1997). Low-order modeling from relay feedback. *Industrial & Engineering Chemistry Research*, 36(2), 375–381. doi:10.1021/ie960412+.

Xue, Y., Li, D., and Gao, F. (2010). Multi-objective optimization and selection for the pi control of alstom gasifier problem. *Control Engineering Practice*, 18(1), 67 – 76. doi:DOI:10.1016/j.conengprac.2009.09.004.

Zhao, S.Z., Iruthayarajan, M.W., Baskar, S., and Suganthan, P. (2011). Multi-objective robust pid controller tuning using two lbests multi-objective particle swarm optimization. *Information Sciences*, 181(16), 3323 – 3335. doi:DOI:10.1016/j.ins.2011.04.003.