Heuristic Approaches for Non-convex Problems. Application to the Design of Structured Controllers and Spiral Inductors

Rosario Toscano, Ioan Alexandru Ivan

Université de Lyon, France

ABSTRACT

This paper aims at solving difficult optimization problems arising in many engineering areas. To this end, two recently developed optimization method will be introduced: the heuristic Kalman algorithms (HKA) and the quasi geometric programming (QGP) problems. The principle of HKA is to consider the optimization problem as a measurement process intended to give an estimate of the optimum. A specific procedure, based on the Kalman estimator, is developed to improve the quality of the estimate obtained through a measurement process. A significant advantage of HKA against other stochastic methods lies mainly in the small number of parameters which have to be set by the user. In this paper we also introduce an extension of standard geometric programming (GP) problems which we call quasi geometric programming (QGP) problems. The consideration of this particular kind of nonlinear and possibly non smooth optimization problem is motivated by the fact that many engineering problems can be formulated as a QGP. To solve this kind of problems (QGP), an algorithm is proposed which is based on the resolution of a succession of standard GP. An interesting feature of the proposed approach is that it does not need to develop specific program solver and works well with any existing solver able to solve conventional GP. In the last part of the paper, it is to shown that HKA and QGP can be efficiently used to solve difficult non-convex optimization problems. In particular, we have addressed the problem of robust structured control and on-ship spiral inductor design. Numerical experiments exemplify the resolution of this kind of problems.

Keywords: *Non-convex optimization problems, stochastic optimization methods, heuristic Kalman algorithm (HKA),quasi geometric programming (QGP), fixed structure controller, mixed sensitivity control problem, loop-shaping design, on-ship spiral inductor.*

INTRODUCTION

In all areas of engineering, physical and social sciences, one encounters problems involving the optimization of some objective function. Usually, the problem to solve can be formulated precisely but is often, difficult or impossible to solve either analytically or through conventional numerical procedures. This is the case when the problem is non-convex and so inherently nonlinear and multimodal. In fact it is now well established that the frontier between the efficiently solvable optimization problems and the others rely on its convexity (Rockafellar, 1993). This is confirmed by the fact that very efficient algorithms for solving convex problems exist (Boyd & Vandenberghe, 2004), whereas the problem of non-convex optimization remains largely open despite an enormous amount of effort devoted to its resolution.

In this context, several heuristic methods, also called metaheuristics, have been developed in the last two decades, which have demonstrated a strong ability to solve problems that were previously difficult or impossible to solve (Fogel 2006, Kirkpatrick & Gelatt 1983, Toscano 2013). These metaheuristics include simulated annealing (SA), genetic algorithm (GA), particle swarm (PS), to cite only the most used in the framework of continuous optimization problems.

Simulated annealing (SA) is a random-search method introduced by S. Kirkpatrick in 1983 and by V. Cerný in 1985 (Kirkpatrick & Gelatt 1983, Cerny 1985). The name comes from a technique used in metallurgy, called annealing, which consists in heating and slowly cooling a metal to obtain a "well ordered" solid state of minimal energy (Dréo & al 2006). An interesting property of SA is its ability to avoid getting stuck in a local minima. This is obtained by using a random procedure which not only accepts changes that decrease the cost function J (assuming a minimization problem), but also some changes that increase it. The latter are accepted with a probability $exp(-\Delta J/T)$, where ΔJ is the increase in J and T is a control parameter, which by analogy with the physical annealing is known as the system temperature. The main advantage of the SA is that it achieves a good quality solution, i.e. the absolute error to the global minimum is generally lower than that obtained via other metaheuristics. Moreover, it is versatile and easy to implement. The main drawbacks of SA lie mainly in the choice of the various parameters involved by this algorithm. The results obtained are indeed very sensitive to the parameter settings. Consequently, the problem of the selection of the "good parameters" (for a given cost function) is a crucial issue, which is however not yet entirely solved. Another weakness of the method, linked to the problem of parameter setting, is its excessive computing time in most applications. More detailed developments on SA, both practical and theoretical, can be found in (Spall 2003, Dréo & al 2006).

Genetic algorithm (GA), is a population-based stochastic search technique introduced by J. H. Holland in 1962 and popularized by D. E. Goldberg in 1989 (see also Coello, Lamont & Veldhuizen 2007, Lobo, Lima, & Michalewicz, 2007, Goldberg 2013). This approach uses a population of points containing several potential solutions, each of which is evaluated and a new population is created from the best of them via randomized operators, such as selection, crossover and mutation, inspired by the natural reproduction and evolution of living creatures. The process is continued through a number of generations with the aim that the population evolves toward an acceptable solution. The main advantage of GA (and its many versions) is its robustness as well as its intuitiveness, ease of implementation, and the ability to deal successfully with a wide range of difficult problems. By robustness it must be understood that, within fairly wide margins, the problem of adjusting the parameters is not very critical. This insensitivity makes it possible to find acceptable solutions without excessive effort. A main drawback with GA is that some well adapted individuals (compared to the other members of the population, but faraway from the optimum point), dominate the population, causing it to converge on a local minimum. In these

conditions, the probability of finding better solutions is very small because crossover between similar individuals, produces little changes. Only mutation remains to seek the best individuals, but this is generally not sufficient for a fast convergence toward the best solution. The latter requires thus an excessive computational time.

Particle swarm optimization (PSO) is a relatively recent stochastic optimization technique developed by J. Kennedy and R. Eberhart in 1995 (see also Clerc 2010). GA and PSO are similar in the sense that these two approaches are population-based random search methods but with different strategies of evolution. PSO draws its inspiration from the collective behavior of living beings, including the notion of collective intelligence of a population of individuals. It is a population based search algorithm where each individual is called a particle and represents a candidate solution. Each particle *i* evolves within the search space and is characterized by its position x_i and its change in position *vi*, called velocity. The key point lies in the manner in which the velocity is modified at each iteration. By analogy with the observations made about social behaviors[1], the velocity of a particle is modified according to its own previous best solution and its group's previous best solution, with the aim to get an improvement (i.e. in the sense of a decrease of the cost function). The main advantage of the PSO is its ease of implementation as well as its ability to find good solutions much faster than other metaheuristics (less function evaluations). However, it cannot improve the quality of the solutions as the number of iterations is increased (Angeline 1998). Similar to the GA, an important drawback with PSO, is that the swarm may prematurely converge. This is mainly because particles converge to a point which is on the line between the global best point and the personal best positions. However this point is not guaranteed to be even a local optimum. Another drawback, similar to the SA, is the great sensitivity of PSO to parameter settings: a small change in parameters may result in a proportionally large effect (Lovberg and Krink 2002).

Although a large number of approaches have been proposed in the literature to improve these metaheuristics, non-convex optimization is still a challenging subject, mainly because of very large variability concerning the topological properties of the underlying objective function. For this reason, it is always useful to explore new principles allowing the resolution of a wide range of non-convex optimization problems. In this spirit, one of the objectives of this paper is to introduce a new alternative optimization method (developed by the author), which we call **Heuristic Kalman Algorithm** (HKA) (Toscano & Lyonnet, 2009a, 2009b, 2010). Another objective of this chapter is to introduce an extension of standard **geometric programming** (GP) problems which we call **quasi geometric programming** (QGP) problems (Toscano & Amouri 2012). The consideration of this particular kind of nonlinear and possibly non smooth optimization problem is motivated by the fact that many engineering problems can be formulated as a QGP. To solve this kind of problems (QGP), an algorithm is proposed which is based on the resolution of a succession of standard GP. An interesting feature of the proposed approach is that it does not need to develop specific program solver and works well with any existing solver able to solve conventional geometric programs. Some considerations on the robustness issue are also presented.

[1] According to the observation of Boyd and Richardson 1985, human beings utilize two important kinds of information in the decision process. The first one is their own experience, i.e. they have tried the choice and know which state has been better so far and also how good it was. The second one is the experience of others, i.e. the knowledge about how the other agents around them have performed (Chakrabarti & al. 2006).

In the last part, the ability of HKA and QGP in solving difficult non-convex problems is illustrated through two domains of application, namely: robust structured control and on-ship spiral inductor design. Numerical experiments exemplify the resolution of this kind of problems.

BACKGROUND: THE OPTIMIZATION PROBLEM

Optimization is the way of obtaining the best possible outcome given the degrees of freedom and the constraints. To make our discussion more precise, consider the general system presented in Figure 1, which produces an output in response to a given input. In addition, this system has some tuning parameters allowing the modification of its behaviour. By behavior we mean the relationship existing between the inputs and outputs.

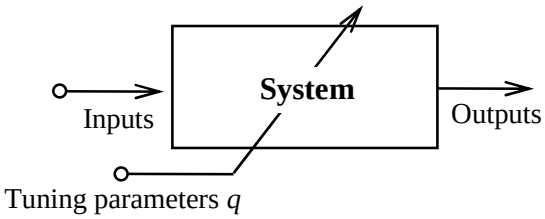

Figure 1: An optimization problem.

The problem is then how to tune these parameters so that the system behaves well. Usually, the desired behavior can be formulated via an *objective function* (or *cost function*) depending on the tuning parameters $f(q)$, which needs to be maximized or minimized with respect to q. More formally, the problem to solve can be formulated as follows: find the optimal tuning parameters q_{opt}, solution of the following problem:

$$\begin{cases} q_{opt} = \arg\min_{q \in \mathcal{F}} f(q) \\ \mathcal{F} = \{ q \in \mathcal{D} : g_i(q) \le 0, i = 1, \ldots, N_c \} \\ \mathcal{D} = \left\{ q \in \mathbf{R}^{n_q} : \underline{q} \le_e q \le_e \overline{q} \right\} \end{cases} \tag{1}$$

where $f : \mathbf{R}^{n_q} \to \mathbf{R}$ is a function for which the minimum[2] ensures that the system behaves as we want, $\mathcal{F}$ is the **feasible domain** i.e. the set of vector $q \in \mathcal{D}$ satisfying the N_c constraints g_i, and $\mathcal{D}$ is the search domain[3] i.e. the set under which the minimization is performed. Generally $q = [q_1 \cdots q_{n_q}]^T$ is called the *design* (or *decision*) vector, and its n_q components the decision or design variables. The vectors $\underline{q} = [\underline{q}_1 \cdots \underline{q}_{n_q}]^T$ and $\overline{q} = [\overline{q}_1 \cdots \overline{q}_{n_q}]^T$ are the bounds of the search domain and the symbol $\le_e$ means a componentwise inequality. The functional constraints g_i can be handled by introducing a new objective function including penalty functions:

$$J(q) = f(q) + \sum_{i=1}^{N_c} w_i \max(g_i(q), 0) \tag{2}$$

---

[2] Note that any maximisation problem can be converted into a minimization problem, indeed: $q_{opt} = \arg\max_{q \in \mathcal{F}} J(q) = \arg\min_{q \in \mathcal{F}} - J(q)$

[3] $\mathcal{D}$ is a hyberbox and so it is also called the hyperbox search domain.

Where N_c is the number of constraints and the w_i's are weighting factors. There exists a vast literature dealing with the problem of the updating rule of the weighting factor (see for instance Coello, 2000, Coello, 2002). However, in most practical applications, the choice of constant weighting factors leads to a satisfying solution (possibly sub-optimal). In this case, the setting of the w_i's must be done to penalize more or less strongly the violation constraints. Note that if q satisfies the constraints then $J(q) = f(q)$. In these conditions solving problem (1) is the same as solving the following optimization problem:

$$\begin{cases} q_{opt} = \arg\min_{q \in \mathcal{D}} J(q) \\ \mathcal{D} = \left\{ q \in \mathbf{R}^{n_q} : \underline{q} \leq_e q \leq_e \overline{q} \right\} \end{cases} \tag{3}$$

Thus posed, the objective is then to find the optimum q_{opt} i.e. the n_q-dimensional decision vector $q \in \mathcal{D}$ which minimizes the cost function J.

Unfortunately, there are several obstacles for solving this kind of problem. The main obstacle is that most of the optimization problems are NP-hard (Garey & Johnson, 1979). Therefore the known theoretical methods cannot be applied except possibly for some small size problems. Other difficulties are that the cost function may be not differentiable and/or multimodal. Therefore the set of methods requiring the derivatives of the cost function cannot be used. Another obstacle is when the cost function cannot be expressed in an analytic form, in this case, the cost function can be only evaluated through simulations.

In these situations, heuristic approaches seem to be the only way for solving optimization problems. By heuristic approach, we mean a computational method employing experimentations, evaluations and trial-and-errors procedures in order to obtain an approximate solution for computationally difficult problems. In the next section, we will present a recently developed optimization methods called **Heuristic Kalman Algorithm** (HKA) which seems to be, in some cases, an interesting alternative to the conventional approaches.

Heuristic Kalman Algorithm (HKA)

In this section, we introduce a recently developed optimization method called **Heuristic Kalman Algorithm** (HKA) (Toscano & Lyonnet, 2009, Toscano & Lyonnet, 2010, Toscano 2013). As GA and PSO, HKA falls into the category of the so called "population based stochastic optimization technique". However, its principle is entirely different to other known stochastic algorithms. Indeed, HKA considers the optimization problem as kind of learning process intended to give an estimate of the optimum. It utilizes a Gaussian probability density function (GPDF), a measurement process (MP) and a Kalman estimator (KE) allowing to improve the quality of the estimate obtained through the MP. The GPDF evolves in the search space seeking the optimal solution of the optimization problem. A GPDF is characterized by its mean vector m and its variance matrix Σ. For seeking the optimal solution, the parameters of the GPDF are updated by taking into account sample points obtained through a measurement process; this is done using a Kalman estimator. Indeed, a Kalman estimator can be seen as a mechanism able to update our knowledge about unknown quantities of interest, by taking into account new gained information. The "movement" of the GPDF is then adjusted according to its current mean value and the new information obtained via the measurement process. The repetition of this procedure leads the GPDF toward a domain of the search space containing, hopefully, high-quality solutions.

Principle of the algorithm

The principle of the algorithm is shown Figure 2. The proposed procedure is iterative, and we denote by *k*, the k^{th} iteration of the algorithm. We have a random generator of probability density function (pdf) $g(q)$, which produces, at each iteration a collection of *N* vectors that are distributed about a given mean vector $m(k)$ with a given variance-covariance matrix $\Sigma(k)$. This collection can be written as follows:

$$\mathbf{q}(k) = \left\{ q^1(k), q^2(k), \cdots, q^N(k) \right\} \tag{4}$$

where $q^i(k)$ is the i^{th} vector generated at the iteration number *k*: $q^i(k) = [q_1^i(k), \cdots, q_{n_q}^i(k)]^T$, and $q_l^i(k)$ is the l^{th} component of $q^i(k)$ ($l = 1, \ldots, n_q$).

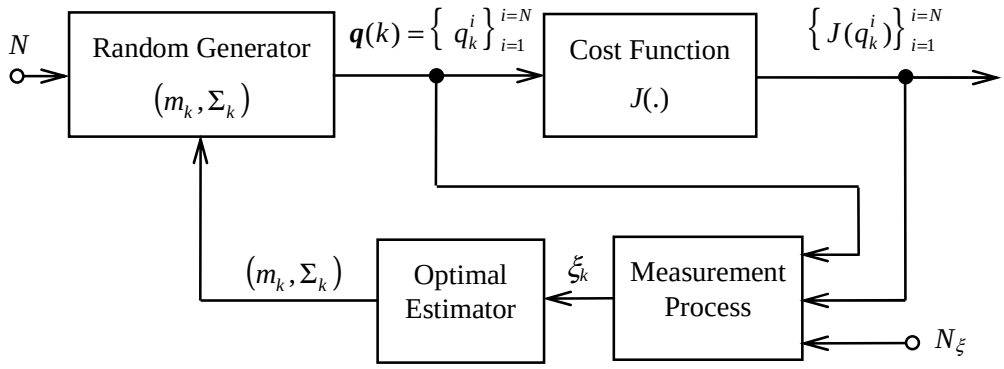

Figure 2: Principle of the algorithm.

This random generator is applied to the cost function *J*. Without loss of generality, we assume that the vectors are ordered by their increasing cost function i.e.:

$$J(q^1(k)) < J(q^2(k)) < \cdots < J(q^N(k)) \tag{5}$$

The principle of the algorithm is to modify the mean vector and the variance matrix of the random generator until a high quality solution is reached. More precisely, let N_ξ be the number of considered best samples, that is such that $J(q^{N_\xi}(k)) < J(q^i(k))$ for all $i > N_\xi$. Note that the best samples are those of the sequence (4) which have the smallest cost function. The objective is then to generate, from the best samples, a new random distribution that approaches the minimum of the cost function *J*. The problem is how to modify the parameters of the random generator to achieve a reliable estimate of the optimum.

To solve this problem, we introduce a measurement procedure followed by an optimal estimator of the parameters of the random generator. The measurement process consists in computing the average of the candidates that are the more representative of the optimum. For the iteration *k*, the measurement, denoted $\xi(k)$, is then defined as follows:

$$\xi(k) = \frac{1}{N_\xi} \sum_{i=1}^{N_\xi} q^i(k) \tag{6}$$

where N_ξ is the number of considered candidates. We can consider that this measure gives a perturbed knowledge about the optimum, i.e.

$$\xi(k) = q_{opt} + v(k) \tag{7}$$

where $v(k)$ is an unknown disturbance, which is centered on q_{opt}, and acting on the measurement process. Note that $v(k)$ is the random vector between the measure $\xi(k)$ and the unknown optimum q_{opt}. In other words, $v(k)$ is a kind of measure of our ignorance about q_{opt}. Of course, this uncertainty cannot be measured but only estimated by taking into account all available knowledge. In our case, the uncertainty of the measure is closely related to the dispersion of the best samples $q^i(k)$ ($i = 1, \ldots, N_\xi$).

Our ignorance about the optimum can thus be taken into account by using the variance vector associated to these best samples:

$$V(k) = \frac{1}{N_\xi}\left[\sum_{i=1}^{N_\xi}\left(q_1^i(k) - \xi_1(k)\right), \cdots, \sum_{i=1}^{N_\xi}\left(q_{n_q}^i(k) - \xi_{n_q}(k)\right)\right]^T \tag{8}$$

In these conditions, the Kalman estimator can then be used to make an estimate, so-called "*a posteriori*", of the optimum, i.e. taking into account the measure as well as the confidence we place in it. As seen, this confidence can be quantified by the variance vector (8).

Updating rules of the Gaussian generator. Our objective is to design an optimal estimator that combines a prior estimation of q_{opt} and the measurement $\xi(k)$, so that the resulting posterior estimate is better in the sense of a diminution of the cost function (minimization problem). Based on the Kalman equations, the updating rule of the Gaussian generator are as follows (see Toscano & Lyonnet, (2010) for a detailed derivation):

$$\begin{cases} m(k+1) = m(k) + L(k)(\xi(k) - m(k)) \\ S(k+1) = S(k) + a(k)(W(k) - S(k)) \end{cases} \tag{9}$$

With:

$$\begin{cases} L(k) = \Sigma(k)(\Sigma(k) + \mathrm{diag}(V(k)))^{-1} \\ W(k) = [\mathrm{vec}^{\mathrm{d}}[(I - L(k))\Sigma(k)]]^{1/2} \end{cases}, \text{ and: } a(k) = \frac{\alpha \min\left(1, \left(\frac{1}{n_q}\sum_{i=1}^{n_q}\sqrt{v_i(k)}\right)^2\right)}{\min\left(1, \left(\frac{1}{n_q}\sum_{i=1}^{n_q}\sqrt{v_i(k)}\right)^2\right) + \max_{1 \le i \le n_q}(w_i(k))} \tag{10}$$

where $m(k)$ is the mean value of the Gaussian distribution, $S(k)$ is the standard deviation vector of the Gaussian generator $S(k) = (\mathrm{vec}^{\mathrm{d}}(\Sigma(k)))^{1/2}$, $\mathrm{vec}^{\mathrm{d}}(.)$ is the diagonal vector of the matrix (.), $\mathrm{diag}(V(k))$ is a diagonal matrix having in its diagonal the variance vector $V(k)$, $v_i(k)$ represents the i^{th} component of the variance vector $V(k)$ defined in (8), $w_i(k)$ is the i^{th} component of the vector $W(k)$, and $\alpha \in (0,1]$ is given by the user (usually α is set about 0.4 to 0.7). The coefficient $a(k)$ is used to control the decrease over time of the variance matrix $\Sigma(k)$. This decrease ensures a progressive transition from global search to local search.

Note that all the matrices used in this formulation (i.e. $L(k)$, $\Sigma(k)$) are diagonals. Consequently, to save computation time we have to use a vectorial form for computing the various quantities of interest. The vectorial form of (9) and, (10) are given by:

$$\begin{aligned} m(k+1) &= m(k) + \mathrm{vec}^{\mathrm{d}}(L(k)) \otimes (\xi(k) - m(k)) \\ S(k+1) &= S(k) - a(k)(W(k) - S(k)) \\ \mathrm{vec}^{\mathrm{d}}(L(k)) &= \mathrm{vec}^{\mathrm{d}}(\Sigma(k)) \,//\, (\mathrm{vec}^{\mathrm{d}}(\Sigma(k)) + V(k)) \end{aligned} \tag{11}$$

where the symbol $\otimes$ stand for a element-by-element product and, similarly, // means a element-by-element divide. The variance matrix $\Sigma(k+1)$ of the Gaussian generator is then updated as follows: $\Sigma(k+1) = (\mathrm{diag}(S(k+1)))^2$.

Algorithm

According to the principles discussed above, the minimization of the objective function $J(q)$ (see relation (3)) can be done according to the following algorithm.

1. **(Initialization).** Choose N, N_ξ and α. Set $k = 0$, $m(k) = m_0$, $\Sigma(k) = \Sigma_0$.

2. **(Gaussian generator).** Generate a sequence of N vectors $q^1(k), q^2(k), \cdots, q^N(k)$, according to a Gaussian distribution parametrized by $m(k)$ and $\Sigma(k)$.

3. **(Measurement process).** Using relations (6) and (8) compute $\xi(k)$ and $V(k)$.
4. **(Updating rules of the Gaussian generator).** Using relations (11) update the parameter of the Gaussian generator.

5. **(Stopping rule).** If the stopping rule is not satisfied go to step 2 otherwise stop.

The practical implementation of this algorithm requires: an appropriate initialization of the the Gaussian distribution i.e. m_0 and Σ_0; the selection of the user defined parameters namely i.e.: N, N_ξ and α.; the introduction of a stopping rule. These various aspects are considered hereafter.

Initialization and parameter settings. The initial parameters of the Gaussian generator are selected to cover the entire search space. To this end, the following rule can be used:

$$m_0 = \begin{bmatrix} \mu_1 \\ \vdots \\ \mu_{n_q} \end{bmatrix}, \quad \Sigma_0 = \begin{bmatrix} \sigma_1^2 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \sigma_{n_q}^2 \end{bmatrix}, \quad \text{with : } \begin{cases} \mu_i = \dfrac{\overline{q}_i + \underline{q}_i}{2} \\ \sigma_i = \dfrac{\overline{q}_i - \underline{q}_i}{6} \end{cases}, \quad i = 1, \cdots, n_q \tag{12}$$

where $\overline{q}_i$ (respectively $\underline{q}_i$) is the i^{th} upper bound (respectively lower bound) of the hyperbox search domain. With this rule, 99% of the samples are generated in the intervals: $\mu_i \pm 3\sigma_i$, $i = 1, \cdots, n_q$.

We have to set the three following parameters: the number of points N, the number of best candidates N_ξ and the coefficient α. To facilitate this task, table 1 summarizes the standard parameter setting of HKA.

Table 1. *Standard parameter setting of HKA.*

Number of sample points (N)	$20 \le N \le 150$
Number of best candidates	$2 \le N_\xi < N$
Coefficient α	0.4 to 0.9

Stopping rule. The algorithm stops when a given number of iterations `MaxIter` is reached (`MaxIter` = 300 in all our experiments) or a given accuracy indicator is obtained. The latest take into account the dispersion of the N_ξ best points. To this end, we consider that no significant improvement can be done when the N_ξ best points are in a ball of a given radius ρ_{HKA} (e.g. $\rho_{HKA} = 0.005$). More precisely, the algorithm stops when:

$$\max_{2 \le i \le N_\xi} \left\| q^1 - q^i \right\|_2 \le \rho_{HKA} \tag{13}$$

where $\left\| \cdot \right\|_2$ represents the Euclidean norm of its argument, and $q^1, \cdots, q^{N_\xi}$ are the N_ξ best candidate solutions.

In conclusion, the search procedure HKA is articulated around three main components, the Gaussian pdf function $g_k(q)$ (parametrized by $m(k)$ and $\Sigma(k)$, the measurement process and the Kalman estimator. Sampling from the pdf $g_k(q)$ at iteration k, creates a collection of vectors $\mathbf{q}(k)$. This collection is then used by the measurement process to give an information about the optimum. Via the Kalman estimator, this information is then combined with the pdf $g_k(q)$ in order to produce a new pdf $g_{k+1}(q)$ which will be used in the next iteration. After a sufficient number of iterations, the sequence of estimates (i.e. the $m(k)$) thus produced leads to a near optimal solution.

Advantages and disadvantages of HKA

HKA shares with some other stochastic algorithms the same interesting features such as: ease of implementation, low memory and CPU speed requirements, search procedure based only on the values of the objective function, no need of strong assumptions such as linearity, differentiability, convexity etc, to solve the optimization problem. In fact it could be used even when the objective function cannot be expressed in an analytic form; in this case, the objective function is evaluated through simulations. However, the main drawback is that HKA may prematurely converge to a local solution, notably when the coefficient α is too high (say about 0.9). The trick is to use low values of this parameter but this lead to a slow convergence of the algorithm. In fact this parameter allows to adjust the trade off between global and local search.

Quasi Geometric Programming

Geometric programming (GP) has proved to be a very efficient tool for solving various kinds of engineering problems. This efficiency comes from the fact that geometric programs can be transformed to convex optimization problems for which powerful global optimization methods have been developed. As a result, globally optimal solution can be computed with great efficiency, even for problems with hundreds of variables and thousands of constraints, using recently developed interior-point algorithms. A detailed tutorial of GP and comprehensive survey of its recent applications to various engineering problems can be found in the paper by Boyd, Kim, Vandenberghe & Hassibi 2007.

In this section we introduce a particular type of nonlinear program which we call quasi geometric programming (QGP) problems. The idea behind QGP is very simple, it means that a problem become GP when some variables are kept constants. To solve this kind of problems (QGP), an algorithm is proposed which is based on the resolution of a succession of standard GP. The interesting thing is that the proposed approaches doesn't need to develop specific program solver and works well with any existing solver able to solve conventional geometric programs (for instance cvx, see Grant & Boyd 2010). From a practical point of view this is very interesting because the engineers often have not so much time to develop specific algorithm for solving particular problems.

Geometric Programming (GP)

GP is a special type of nonlinear, non-convex optimisation problems. A useful property of GP is that it can be turned into a convex optimization problem and thus a local optimum is also a global one, which can be computed very efficiently. Since QGP is based on the resolution of GP, this section gives a short presentation of GP both in standard and convex form.

Standard formulation. Monomials are the basic elements for formulating a geometric programming problem. A monomial is a function $\mathbf{R}_{++}^{n} \to \mathbf{R}$ defined by[4]:

$$f(q) = cq_1^{a^1} q_2^{a^2} \cdots q_n^{a^n} \tag{14}$$

where $q_1 \cdots q_n$ are n positive variables, c is a positive multiplicative constant and the exponentials $a^i,\ i = 1 \cdots n$ are real numbers. We will denote by q the vector $(q_1 \cdots q_n)$. A sum of monomial is called a posynomial:

$$f(q) = \sum_{k=1}^{K} c_k q_1^{a_k^1} q_2^{a_k^2} \cdots q_n^{a_k^n} \tag{15}$$

Minimizing a posynomial subject to posynomial upper bound inequality constraints and monomial equality constraints is called GP in standard form:

$$\begin{array}{ll} \text{minimize} & f_0(q) \\ \text{subject to} & f_i(q) \le 1, \quad i = 1, \ldots, m \\ & g_i(q) = 1, \quad i = 1, \cdots, p \end{array} \tag{16}$$

where $f_i,\ i = 1 \cdots m$, are posynomials and $g_i,\ i = 1 \cdots p$, are monomials.

Convex formulation. GP in standard form is not a convex optimisation problem[5], but it can be transformed to a convex problem by an appropriate change of variables and a log

[4] In our notations, $\mathbf{R}_{++}$ represents the set of positive real numbers.

[5] A convex optimization problem consists in minimizing a convex function subject to convex inequality constraints and linear equality constraints.

transformation of the objective and constraint functions. Indeed, if we introduce the change of variables $y_i = \log q_i$ (and so $q_i = e^{y_i}$), the posynomial function (15) becomes:

$$f(y) = \sum_{k=1}^{K} c_k \exp\left(\sum_{i=1}^{n} a_k^i y_i \right) = \sum_{k=1}^{K} \exp\left(a_k^T y + b_k\right) \tag{17}$$

where $b_k = \log c_k$, taking the log we obtain $\bar{f}(y) = \log(\sum_{k=1}^{K} \exp(a_k^T y + b_k))$, which is a convex function of the new variable *y*. Applying this change of variable and the log transformation to the problem (16) gives the following equivalent optimization problem:

$$\begin{aligned} \text{minimize} \quad & \bar{f}_0(y) = \log\left(\sum_{k=1}^{K_0} \exp(a_{0k}^T y + b_{0k})\right) \\ \text{subject to} \quad & \bar{f}_i(y) = \log\left(\sum_{k=1}^{K_i} \exp(a_{ik}^T y + b_{ik})\right) \le 0, \quad i = 1, \ldots, m \\ & \bar{g}_j(y) = a_j^T y + b_j, \quad j = 1, \cdots, p \end{aligned} \tag{18}$$

Since the functions $\bar{f}_i$ are convex, and $\bar{g}_j$ are affine, this problem is a convex optimization problem, called geometric program in convex form. However, in some practical situations, it is not possible to formulate the problem in standard geometric form, the problem is then not convex. In this case the problem is generally difficult to solve even approximately. In these situations, it seems very useful to introduce simple approaches able to give if not the optimum, at least a good near-optimum. In this spirit, we are now ready to introduce the concept of quasi geometric programming.

Formulation of a Quasi Geometric Programming Problem (QGP)

Consider the nonlinear program defined by

$$\begin{aligned} \text{minimize} \quad & f_0(z) \\ \text{subject to} \quad & f_i(z) \le 0, \quad i = 1, \ldots, m \\ & g_j(z) = 1, \quad j = 1, \cdots, p \end{aligned} \tag{19}$$

where the vector $z \in \mathbf{R}_{++}^n$ include all the optimization variables, $f_0 : \mathbf{R}_{++}^n \to \mathbf{R}$ is the objective function or cost function, $f_i : \mathbf{R}_{++}^n \to \mathbf{R}$ are the inequality constraint functions and $g_j : \mathbf{R}_{++}^n \to \mathbf{R}$ are the equality constraint functions. This nonlinear optimization problem is called a quasi geometric programming problem if it can be formulated into the following form (Toscano & Amouri, 2012):

$$\begin{aligned} \text{minimize} \quad & \varphi_0(x, \xi) - Q_0(\xi) \\ \text{subject to} \quad & \varphi_i(x, \xi) \le Q_i(\xi), \quad i = 1, \ldots, m \\ & h_j(x, \xi) = Q'_j(\xi), \quad j = 1, \cdots, p \end{aligned} \tag{20}$$

where $x \in \mathbf{R}_{++}^{n_x}$ and $\xi \in \mathbf{R}_{++}^{n_\xi}$, with $n_x + n_\xi = n$, are sub-vector of the optimization variable $z \in \mathbf{R}_{++}^{n}$. The functions $\varphi_i(x,\xi),\ i = 0,\ldots,m$ are posynomials and $h_j(x,\xi),\ j = 1,\ldots,p$ are monomials. The only particular assumption made about the functions $Q_0(\xi)$, $Q_i(\xi)$ and $Q'_j(\xi)$, is that they are positives. Except for their positivity, no other particular assumption is made; these functions can be even non-smooth.

It is important to insist on the fact that the problem (20) cannot be converted into a GP in the standard form (16) and thus the problem is not convex. As a consequence, no approach exists for finding quickly even a sub optimal solution by using available GP solvers. Although specific algorithms can be designed to find out a sub optimal solution to problem (20), we think that it could be very interesting solving these problems by using standard GP solvers. Indeed, this should be interesting for at least two reasons. Firstly the ability of solving problem (20) using available GP solvers allows time saving; the development of a specific algorithm is always a long process and in an industrial context of great concurrency there is often no time to do that. Secondly, the available GP solvers like for instance cvx are very easy to use and, which is most important, are very very efficient. Problems involving tens of variables and hundreds of constraints can be solved on a small current workstation in less than one second.

All these reasons justify the approach presented here after. Indeed, this method does not require the development of particular algorithms and are based on the use of available zero order algorithm (ZOA) and GP solvers.

Resolution of a QGP

The QGP (20) is not at all easy to solve when $Q_0(\xi)$, $Q_i(\xi)$ and $Q'_j(\xi)$ have no particular form. In this case indeed, the problem is intrinsically non-convex, and thus, in general, there is no obvious transformation allowing to solve (20) via the resolution of a sequence of standard GP. To solve this kind of problem, we can see the QGP (20) like a function of ξ that we want to minimize:

$$\begin{aligned} &\text{minimize} \quad F(\xi) = J(\xi) - Q_0(\xi) \\ &\text{subject to} \quad \underline{\xi} \le \xi \le \overline{\xi} \end{aligned} \tag{21}$$

where $\underline{\xi}$ and $\overline{\xi}$ are simple bound constraints on the decision variable ξ, and the function $J(\xi)$ is defined as follows:

$$\begin{aligned} J(\xi) = \min_x \quad & \varphi_0(x,\xi) \\ \text{s.t.} \quad & \varphi_i(x,\xi) \le Q_i(\xi), \quad i = 1,\ldots,m \\ & h_j(x,\xi) = Q'_j(\xi), \quad j = 1,\cdots,p \end{aligned} \tag{22}$$

Problem (21) is a non-convex unconstrained optimization problem[6] and can be solved using well known zero order algorithms[7] (ZOA) such as: Nelder-Mead simplex method (NMSM) (Kelley, 1999), simulated annealing, genetic algorithm, particle swarm optimization or Heuristic Kalman Algorithm. The code associated to these various algorithms are easily available and thus don't need to be programmed.

When ξ is kept constant, problem (22) is a standard GP which can be solved very efficiently using available GP solvers. This suggests that we can solve the QGP problem (20) with a two levels procedure. At the first level, the chosen ZOA search algorithm is used to select a value of ξ within the bounds. For the selected value of ξ, the standard GP (22) is solved using available solvers. This procedure is continued until some stopping rule is satisfied. The suggested procedure is formalized more precisely in the following algorithm.

1. Set $F_{best} := \texttt{inf}$.
2. Using a zero order algorithm (ZOA), generate ξ^* such that $\underline{\xi} \le \xi \le \overline{\xi}$.
3. For the value ξ^* solve the standard GP problem (22). This gives, w.r.t ξ^*, the optimal solution denoted x^*.
4. If problem (22) is not feasible, then set $F := \inf$ and goto 2. Else set $F := \varphi_0(x^*, \xi^*) - Q_0(\xi^*)$.
5. If $F \ge F_{best}$ then goto 2. Else set $F_{best} := F$, $x_{best} := x^*$, $\xi_{best} := \xi^*$ and goto 2.
6. At the end of the ZOA, the optimal solution is given by (x_{best}, ξ_{best}).

In this algorithm, `inf` represents the IEEE arithmetic representation for positive infinity, and F_{best} is a variable containing the current best objective function. Note that the use of ``global optimization methods" like SA, GA, PSO or HKA, increases the probability of finding a global optimum but this is not guaranteed, except perhaps if the search space of problem (21) is explored very finely, but this is cannot be done in a reasonable time.

APPLICATION TO ROBUST STRUCTURED CONTROL AND SPIRAL INDUCTORS DESIGN

Robust structured control

The problem of designing a robust controller with a given fixed structure (e.g. a MIMO PID) remains an open issue (Toscano & Lyonnet, 2009b, Toscano, 2013). This is mainly due to the fact that the set of all fixed-order/structure stabilizing controllers is non-convex and disconnected in the space of controller parameters. This is a major source of computational intractability and conservatism. Nevertheless, due to their practical importance, some approaches for **structured**

[6] We have only simple bound constraints on the decision variable ξ.
[7] Zero order algorithms does not require the knowledge of the derivatives of the objective function. Thus smoothness is not required.

control have been proposed in the literature. Most of them are based on the resolution of Linear Matrix Inequalities LMIs. However, a major drawback with this kind of approaches is the use of Lyapunov variables, whose number grows quadratically with the system size. For instance, if we consider a system of order 70, this requires, at least, the introduction of 2485 unknown variables whereas we are looking for the parameters of a fixed-order/structure controller which contains a comparatively very small number of unknowns. It is then necessary to introduce new techniques capable of dealing with the non-convexity of certain problem arising in automatic control without introducing extra unknown variables. We will show that **stochastic methods** can be used to this end.

Formulation of the optimization problem.

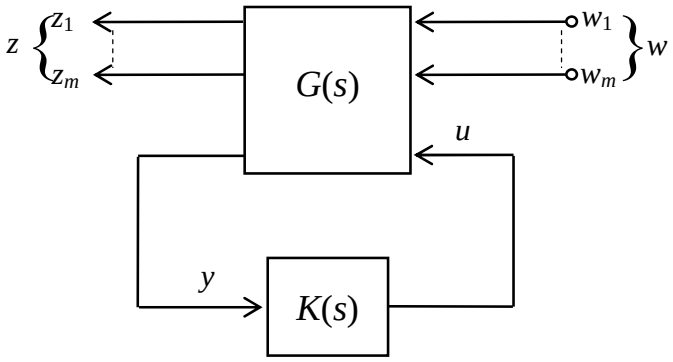

Figure 3: Block diagram of the feedback control system.

Consider the general feedback setup shown in Figure 3, in which $G(s)$ represents the transfer matrix of the process to be controlled:

$$\begin{bmatrix} z \\ y \end{bmatrix} = G(s) \begin{bmatrix} w \\ u \end{bmatrix}, \quad \text{with :} \quad G(s) = \left[\begin{array}{c|cc} A & B_1 & B_2 \\ \hline C_1 & D_{11} & D_{12} \\ C_2 & D_{21} & D_{22} \end{array}\right] \tag{23}$$

and $K(s)$ is, for instance, the transfer matrix of a PID controller[8]

$$K(s) = K_p + K_i \frac{1}{s} + K_d \frac{s}{1+\tau s} = \left[\begin{array}{c|c} A_K & B_K \\ \hline C_K & D_K \end{array}\right] = \left[\begin{array}{cc|c} 0 & 0 & K_i \\ 0 & -\frac{1}{\tau} I & -\frac{1}{\tau^2} K_d \\ \hline I & I & K_p + \frac{1}{\tau} K_d \end{array}\right] \tag{24}$$

where K_p is the proportional gain, K_i and K_d are the integral and derivative gains respectively, and τ is the time constant of the filter applied to the derivative action. This low-pass first-order filter ensures the properness of the PID controller and thus its physical realizability. In addition, since $G(s)$ is strictly proper (i.e. it is assumed that $D_{22} = 0$), the properness of the controller ensures the well-posedness of feedback loop.

[8] Due to its large diffusion, we consider a PID controller, but the described approach applies for any other fixed structure controller.

As depicted Figure 4, the closed-loop system has m external input vectors $w_1 \in \mathbf{R}^{n_{w1}}, \cdots, w_m \in \mathbf{R}^{n_{wm}}$ and m output vectors $z_1 \in \mathbf{R}^{n_{z1}}, \cdots, z_m \in \mathbf{R}^{n_{zm}}$. Roughly speaking, the global input vector $w = [w_1 \ \ldots \ w_m]^T$ captures the effects of the environment on the feedback system; for instance noise, disturbances and references. The global output vector $z = [z_1 \ \ldots \ z_m]^T$ contains all characteristics of the closed-loop system that are to be controlled. To this end, the controller $K(s)$ utilizes the measured output vector $y \in \mathbf{R}^{n_y}$, to elaborate the control action vector $u \in \mathbf{R}^{n_u}$ which modify the natural behavior of the process $G(s)$.

The objective is then to determine the PID parameters (K_p, K_i, K_d, τ) allowing to satisfy some performance specifications such as: a good set point tracking, a satisfactory load disturbance rejection, a good robustness to model uncertainties and so on. A powerful way to enforce these kinds of requirements is first to formulate the performance specifications as an optimization problem and then to solve it by an appropriate method. In the $\mathcal{H}_\infty$ framework, the optimization problem can take the following forms:

$$
\begin{aligned}
\text{Minimize} \quad & J_\infty(q) = \left\| T_{w_1 z_1}(s,q) \right\|_\infty, \quad q = [\mathrm{vec}(K_p) \ \ \mathrm{vec}(K_i) \ \ \mathrm{vec}(K_d) \ \ \tau]^T \\
\text{Subject to:} \quad & g_1(q) = \arg\max_{\lambda_i(q)} \{ \mathrm{Re}(\lambda_i(q)), \forall i \} - \lambda_{min} \le 0 \\
& g_2(q) = \left\| T_{w_2 z_2}(s,q) \right\|_\infty - \gamma_2 \le 0 \\
& \vdots \\
& g_m(q) = \left\| T_{w_m z_m}(s,q) \right\|_\infty - \gamma_m \le 0
\end{aligned}
\qquad (25)
$$

where $T_{w_i z_i}(s,q)$ denotes the closed-loop transfer matrix from w_i to z_i, $q \in \mathbf{R}^{n_q}$ is the vector of decision variables regrouping the entries of the matrices K_p, K_i, K_d, and the time constant τ, $\lambda_i(q)$ denotes the i^{th} pole of the closed-loop system and s is the Laplace variable. In the formulation (21) the constraint $g_1(q)$ is required to ensure the stability of the closed-loop system. To do so, the parameter λ_{min} must be set to a negative value.

Note that this formulation is quite general and can be used to specify many control objectives. For instance, the formulation (25) includes the PID loop-shaping design problem as well as the single or mixed sensitivity PID control problem. In the numerical experiments we will see some applications belonging to theses two kind of control problems.

The constrained optimization problem (25) can be transformed into an unconstrained one, by introducing a new objective function which includes penalty functions (see relation (2) and (3)).

Remark concerning the feasibility issue. In many engineering problems the bounds of a **feasible search domain** are often known a priori because they are linked to purely material, physical considerations. This is not so clear in control problem for which we have to impose a priori an hyperbox search domain containing stabilizing controllers (i.e. potential solutions of the optimal $\mathcal{H}_\infty$ problem). Finding a priori such a hyperbox is not trivial at all. However, for a given hyperbox search domain it is possible to say whether or not the problem is feasible. More precisely, the feasibility problem can be stated as follows. Given the hyperbox search domain $\mathcal{D} = \{ q_i \in R : \underline{q}_i \le q_i \le \overline{q}_i, \quad i = 1, \cdots, n_q \}$ is there a stabilizing controller? This important issue can be treated via HKA by solving the following optimization problem:

$$\begin{aligned} &\text{Minimize} && J_\lambda(q) = \arg\max_{\lambda_i(q)} \{\mathrm{Re}(\lambda_i(q)), \forall i\} \\ &\text{Subject to:} && \underline{q}_i \le q_i \le \overline{q}_i, \quad i = 1, \cdots, n_q \end{aligned} \tag{26}$$

where $\lambda_i(q)$ represents the i^{th} pole f the closed-loop system. Let q^* the solution found to the problem (26). If $J_\lambda(q^*) < 0$, then the problem is feasible within $\mathcal{D}$.

Numerical experiments

In this section, we show the ability of **stochastic methods** to solve problems (25). Although this can be done using SA, GA or PSO, we utilize the HKA mainly because of its novelty. Concerning the control design by means of other **stochastic methods** see for instance Jamshidi *et al.* (2002), Motoda, Stengel & Miyazawa (2002), Maruta, Kim, & Sugie (2008).

Mixed sensitivity approach. In this example we consider the mixed sensitivity control problem shown in figure 4, where $G(s)$ represents the transfer matrix of the process to be controlled (see Saeki, 2006):

$$G(s) = \begin{bmatrix} \dfrac{1}{s+1} & \dfrac{0.2}{s+3} \\ \dfrac{0.1}{s+2} & \dfrac{1}{s+1} \end{bmatrix} \tag{27}$$

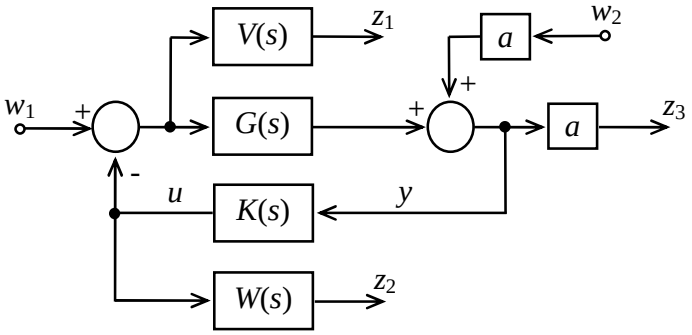

Figure 4: Mixed sensitivity control problem.

The weighting functions are $V(s)=v_1(s)I_2$, $W(s)=v_2(s)I_2$, $v_1(s)=(s+3)/(3s+0.3)$, $v_2(s)=(10s+2)/(s+40)$, and $a=0.01$. The objective is to develop a decentralized PID controller

$$\begin{aligned} &K(s,q) = \begin{bmatrix} K_{p1} & 0 \\ 0 & K_{p2} \end{bmatrix} + \begin{bmatrix} K_{i1} & 0 \\ 0 & K_{i2} \end{bmatrix} \frac{1}{s} + \begin{bmatrix} K_{d1} & 0 \\ 0 & K_{d2} \end{bmatrix} \frac{s}{1+0.01s} \\ &q = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \end{bmatrix}^T, \text{with}: x_1 = \log_{10}(K_{p1}),\ x_2 = \log_{10}(K_{i1}), \\ &x_3 = \log_{10}(K_{d1}),\ x_4 = \log_{10}(K_{p2}),\ x_5 = \log_{10}(K_{i2}),\ x_6 = \log_{10}(K_{d2}) \end{aligned} \tag{28}$$

that minimize $J(x) = \left\| T_{wz}(s,x) \right\|_\infty$, where $T_{wz}(s,x)$ is the closed-loop transfer matrix from w to z:

$$T_{wz}(s,x)=\begin{bmatrix} V(s)(I-(I+L(s,x))^{-1}L(s,x)) & -aV(s)(I+L(s,x))^{-1}K(s,x) \\ W(s)(I+L(s,x))^{-1}L(s,x) & aW(s)(I+L(s,x))^{-1}K(s,x) \\ aG(s)(I-(I+L(s,x))^{-1}L(s,x)) & a^2(I-G(s)(I+L(s,x))^{-1}K(s,x)) \end{bmatrix} \quad (29)$$

where $L(s,\underline{x})$ is defined as $L(s,\underline{x})=K(s,\underline{x})G(s)$. Note that this problem is of the form (25). The search space is $-3 \le x_i \le 3$, $i=1,\ldots,6$. In this test, we performed the minimization 30 times and we compared our results with those obtained via ALPSO (Augmented Lagrangian Particle Swarm Optimization see Kim, Maruta & Sugie, 2008). The design parameters are: N=25, N_ξ=5 and α=0.4.

The best solutions obtained via ALPSO and HKA are listed in Table 2, and the statistical results are shown in Table 3.

Table 2. *Comparison of the best solutions found via ALPSO and HKA.*

	K_{p1}	K_{i1}	K_{d1}	K_{p2}	K_{i2}	K_{d2}	$J(x)$
ALPSO	1.8015	1.9477	2.683e-2	1.8252	1.8135	1.188e-2	0.5842
HKA	1.8109	1.8422	1.833e-2	1.8188	1.9259	2.235e-2	0.5836

Table 3. *Statistical results (comparison between ALPSO and HKA).*

	Best	Mean	Worst	Std Dev	CPU time	Average number of function evaluations
ALPSO	0.5842	-	-	-	50 s	-
HKA	0.5836	0.5850	0.5884	0.001	35 s	3175

From Table 2, it can be seen that the best solution found by HKA is similar to the one found via the ALPSO method. From Table 3, we can observe that HKA is faster than ALPSO, in addition, the dispersion of the solutions is very low.

Note that the PID controller gains have been found directly by solving the optimization problem via the HKA method. The proposed strategy is straightforward, and thus can be easily applied to a wide range of engineering control problems. For instance, in this third example, the mixed sensitivity control problem has been solved without using LMI transformation as it is required in the approach of Saeki (2006).

PID loop-shaping design. $\mathscr{H}_\infty$ loop-shaping design procedure proposed by Mc Farlane and Glover (1992) is an efficient method to design robust controllers and has been successfully applied to a variety of practical problems. In this framework, the plant $G(s)$ is first shaped with a pre-compensator $W_1(s)$ and a post-compensator $W_2(s)$. The ponderation $W_1(s)$ and $W_2(s)$ are chosen so that the weighted plant $W_1(s)G(s)W_2(s)$ has a desired loop shape, typically a large gain at low frequencies for performance and a small gain at high frequencies for noise attenuation.

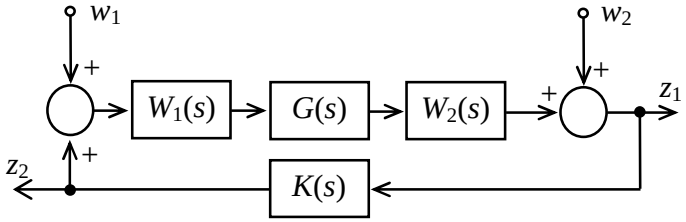

Figure 5: Loop-shaping $\mathscr{H}_\infty$ design.

Once the desired loop shape is achieved, $\mathcal{H}_\infty$ norm of the transfer function matrix from disturbances w_1 and w_2 to the outputs z_1 and z_2 (see figure 5) is minimized over all stabilizing controllers[9]:

$$\begin{cases} K_{opt} = \arg\min_K \left\| T_{zw}(K) \right\|_\infty \\ \qquad = \arg\min_K \left\| \begin{bmatrix} K(I - W_2GW_1K)^{-1}W_2GW_1 & K(I - W_2GW_1K)^{-1} \\ (I - W_2GW_1K)^{-1}W_2GW_1 & (I - W_2GW_1K)^{-1} \end{bmatrix} \right\|_\infty \\ \text{Subject to:} \quad \max_{\lambda_i(K)} \{\text{Re}(\lambda_i(K)), \forall i\} < 0 \end{cases} \tag{30}$$

The final controller is then implemented as $W_1(s)K(s)W_2(s)$ and has no specific structure. The quantity $\varepsilon = 1/\left\| T_{zw}(K_{opt}) \right\|_\infty$ is known as the robust stability margin; usually value of $\varepsilon > 0.2$ or 0.3 is considered as very satisfactory in the sense that the controller K_{opt} does not significantly alter the desired open-loop frequency response. Moreover, this ensures robustness of the closed-loop system to coprime factor uncertainties (Mc Farlane and Glover, 1992).

For loop-shaping design with PID, we adopt the strategy introduced in Genc (2000) and Apkarian, Bompart and Noll (2007). In this approach, the controller K is structured as $K = W_1^{-1}K_{PID}$, where K_{PID} is a PID controller. Thus the optimization problem (30) becomes:

$$\begin{cases} K_{PID}^* = \arg\min_{K_{PID}} \left\| T_{zw}(K_{PID}) \right\|_\infty \\ \qquad = \arg\min_{K_{PID}} \left\| \begin{bmatrix} W_1^{-1}K_{PID}(I - W_2GK_{PID})^{-1}W_2GW_1 & W_1^{-1}K_{PID}(I - W_2GK_{PID})^{-1} \\ (I - W_2GK_{PID})^{-1}W_2GW_1 & (I - W_2GK_{PID})^{-1} \end{bmatrix} \right\|_\infty \\ \text{Subject to:} \quad \max_{\lambda_i(K_{PID})} \{\text{Re}(\lambda_i(K_{PID})), \forall i\} < 0 \\ K_{PID}(s) = K_p + K_i\frac{1}{s} + K_d\frac{s}{1+\tau s}, \quad K_p, K_i, K_d \in \mathbf{R}^{n_u \times n_y} \end{cases} \tag{31}$$

and the final controller is implemented as $K_{PID}^* W_2$. Since W_2 is usually chosen as a low pass filter, the resulting controller has better noise attenuation in the high frequency range than an usual PID controller.

Application to a separating tower. We consider now the application of the HKA to the control design for a chemical process described in Genc (2000) and Apkarian, Bompart and Noll (2007). It consists of a 24-tray tower for separating methanol and water. The transfer matrix model for controlling the temperature on the 4th and 17th trays is given as:

$$\begin{bmatrix} t_{17} \\ t_4 \end{bmatrix} = \begin{bmatrix} \dfrac{-2.2e^{-s}}{7s+1} & \dfrac{1.3e^{-0.3s}}{7s+1} \\ \dfrac{-2.8e^{-1.8s}}{9.5s+1} & \dfrac{-4.3e^{-0.35s}}{9.2s+1} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \tag{32}$$

[9] Note that this optimisation problem is of the form (25).

The transfer matrix (32) is approximated by a rational model using 2nd-order Padé approximation of the delays. This lead to a 12th-order model. The weighting matrix W_1 and W_2 are taken from Genc (2000) and Apkarian, Bompart and Noll (2007):

$$W_1(s)=\begin{bmatrix}\dfrac{5s+2}{s+0.001} & 0\\ 0 & \dfrac{5s+2}{s+0.001}\end{bmatrix},\quad W_2(s)=\begin{bmatrix}\dfrac{10}{s+10} & 0\\ 0 & \dfrac{10}{s+10}\end{bmatrix} \tag{33}$$

The complete system incorporating the compensators is therefore of 18th-order. Our objective is to find the PID parameters $x=[x_1,\cdots,x_{13}]^T$ $(-3\le x_i\le 3,\ i=1,\ldots,13)$ defined as follows:

$$K_{PID}(s,x)=\underbrace{\begin{bmatrix}x_1 & x_2\\ x_3 & x_4\end{bmatrix}}_{K_p}+\underbrace{\begin{bmatrix}x_5 & x_6\\ x_7 & x_8\end{bmatrix}}_{K_i}\frac{1}{s}+\underbrace{\begin{bmatrix}x_9 & x_{10}\\ x_{11} & x_{12}\end{bmatrix}}_{K_d}\frac{s}{1+x_{13}s} \tag{34}$$

to obtain the best possible robustness margin. In Genc (2000) a state-space BMI (bilinear matrix inequality) formulation has been used to characterize PID solutions of the $\mathscr{H}_\infty$ optimization problem (31). The algorithm used to solve this problem is a D-K iteration scheme. The author reported 38 minutes of cputime to obtain the following solution:

$$K_p=\begin{bmatrix}2.4719 & -1.2098\\ -1.1667 & -2.4766\end{bmatrix},\quad K_i=\begin{bmatrix}0.4657 & -0.31\\ -0.2329 & -0.487\end{bmatrix},\quad K_d=\begin{bmatrix}0.0534 & -0.0072\\ -0.015 & -0.0434\end{bmatrix},\quad \tau=0.06$$

The corresponding robustness margin is $\varepsilon=1/4.02=0.249$. In Apkarian, Bompart and Noll (2007) the same problem was solved using a non-smooth optimization technique. The algorithm was initialized with the above solution and the following PID was found in about 1 minute:

$$K_p=\begin{bmatrix}2.6047 & -0.6543\\ -1.1253 & -2.3226\end{bmatrix},\quad K_i=\begin{bmatrix}0.8527 & -0.2591\\ 0.0701 & -0.9362\end{bmatrix},\quad K_d=\begin{bmatrix}0.7414 & -0.2551\\ -1.5610 & -0.0331\end{bmatrix},\quad \tau=0.1527$$

The corresponding robustness margin is $\varepsilon=1/2.91=0.343$. This is an impressive improvement in term of cputime and robustness margin compared to the result reported by Genc (2000). In our case, we solved the optimization problem 15 times (N=50, N_ξ=3 and α=0.5). The best solution found via HKA is as follows:

$$K_p=\begin{bmatrix}2.6091 & -0.6952\\ -1.1068 & -2.4394\end{bmatrix},\quad K_i=\begin{bmatrix}0.8025 & -0.2047\\ 0.0390 & -0.8408\end{bmatrix},\quad K_d=\begin{bmatrix}0.7442 & -0.2852\\ -1.5979 & -0.0300\end{bmatrix},\quad \tau=0.1538$$

The corresponding robustness margin is $\varepsilon=1/2.93=0.341$. Step responses are shown in figures 6. The best solution was found in about 66 seconds on 1.2 Ghz Celeron personal computer. Note that this PID controller is very close to the result obtained by Apkarian, Bompart and Noll (2007). However it must be noticed that the proposed approach is very easy to use and does not require any complicated mathematical derivation. Compared to D-K iteration or non-

smooth optimization, HKA seems to be a good alternative in term of simplicity, near optimallity of the solutions and computation time.

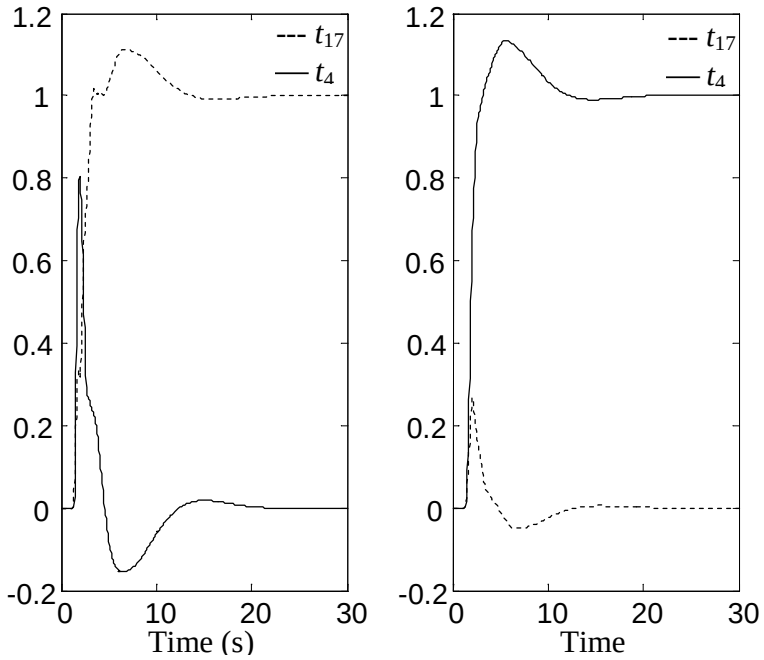

Figure 6: Step responses obtained with the HKA PID controller.

Design of spiral inductors on silicon

On-chip spiral inductors is an essential part of any radio frequency integrated circuit such as voltage controlled oscillators, low-noise amplifiers etc. Consequently, the optimal design of this kind of component is of great practical importance (Toscano & Lyonnet, 2012).

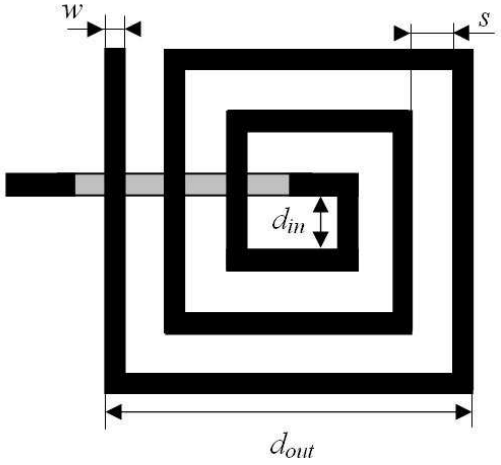

Figure 5: Square inductor layout.

Figure 5 shows the layout for square inductors, some other shapes can be used such as hexagonal, octagonal, or circular. For a given shape, an inductor is completely specified by the number of turns n, the turn width w, the turn spacing s, the inner diameter d_{in} and the outer diameter d_{out} (see

figure 5). These parameters are typically the design variables of the inductor. Indeed, the inductance depends upon the geometry of the inductor, and so, for a desired inductance we have to determine the values of the layout parameters. But this is not sufficient, because at high frequencies (i.e. in the Ghz range), some complicated losses mechanisms must be taken into account to make a realistic design.

In the sequel, we first introduce a well-accepted inductor model able to take into account the losses via parasitic resistances and capacitances. On the basis of this model, the optimal design of an on chip inductor is realized by using quasi geometric programming (QGP).

Inductor model

Figure 6(a) illustrates the basic structure of a planar spiral inductor on silicon. It consists of a metal trace manufactured by low-resistivity metals such as aluminium, copper, gold or silver. The metal spiral is mounted on silicon dioxide layer which acts as insulation between the metal trace and the silicon substrate. Figure 6(a) also highlights the parasitic resistances and capacitances which are introduced to model the losses.

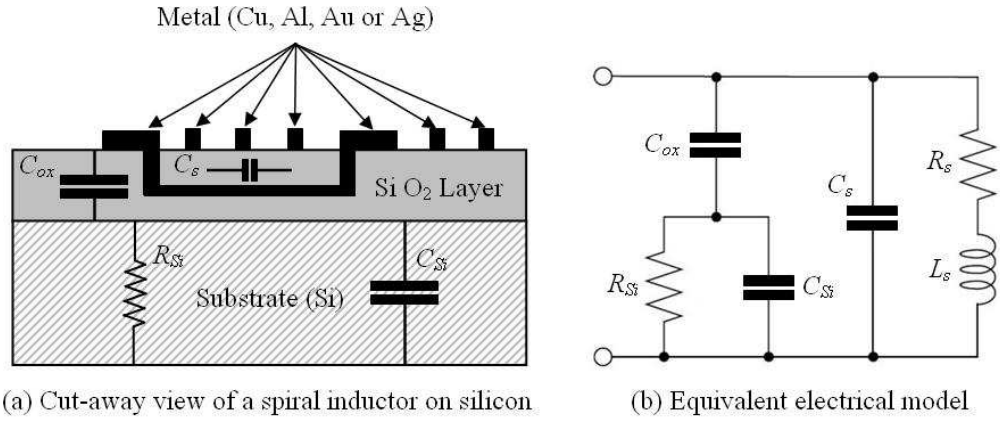

Figure 6: Structure of an inductor on silicon and equivalent electrical model.

The corresponding electrical model of the spiral inductor on silicon is presented in figure 6(b), see the paper by Yue, Ryu, Lau, Lee & Wong 1996, for a detailed derivation. This model takes into account the parasitic resistances and capacitances responsible of the losses in the structure. The inductance *Ls*, and the resistances and capacitances *Rs*, *Cs*, *Rp*, *Cp* are defined as follows:

$$\begin{aligned} &L_s = k_1 n^2 z(d_{in}, d_{out}), \quad R_s = k_2 n(d_{in} + d_{out})/w, \quad C_s = k_3 n w^2 \\ &R_p = 2k_7/(nw(d_{in} + d_{out})), \quad C_p = (k_8 + k_9) nw(d_{in} + d_{out})/2 \end{aligned} \tag{35}$$

The function $z(d_{in}, d_{out})$ and the constants k_1, k_2, k_3, k_7, k_8 and k_9 are given by:

$$\begin{aligned} &z(d_{in}, d_{out}) = c_1(\ln(c_2/r) + c_3 r + c_4 r^2), \quad r = (d_{out} - d_{in})/(d_{in} + d_{out})/w \\ &k_1 = 2\pi 10^{-7}, \quad k_2 = \eta\rho/(d(1 - e^{-t/\delta})), \quad \eta = c_5 \tan(\pi/c_5), \quad \delta = \sqrt{5\times 10^6 \rho/(\pi\omega)} \\ &k_3 = \varepsilon_{ox}/t_{ox,M_1-M_2}, \quad k_4 = \eta\varepsilon_{ox}/(2t_{ox}), \quad k_5 = \eta C_{sub}/2, \quad k_6 = 2/(\eta G_{sub}) \\ &k_7 = 1/(\omega^2 k_4^2 k_6) + k_6(k_4 + k_5)^2/k_4^2, \quad k_8 = k_4/(1 + \omega^2(k_4 + k_5)^2 k_6^2) \\ &k_9 = k_4\omega^2(k_4 + k_5)k_5 k_6^2/(1 + \omega^2(k_4 + k_5)^2 k_6^2) \end{aligned} \tag{36}$$

where the parameters c_1, c_2, c_3, c_4, c_5 depend upon the shape of the inductor (square, hexagonal, octagonal or circular); the parameters ρ, r, ε_{ox}, t_{ox}, $t_{ox,M1\text{-}M2}$, C_{sub}, G_{sub} are technology dependent, and ω is the working frequency of the inductor.

The performance of an inductor is measured by its quality factor Q, which is limited by the parasitics. This quantity is defined as the ratio of peak magnetic energy minus peak electric energy to energy dissipated in the inductor see Yue, Ryu, Lau, Lee & Wong 1996:

$$Q = \frac{\omega L_s}{R_s} \frac{R_p \left(1 - \frac{R_s^2 (C_s + C_p)}{L_s} - \omega^2 L_s (C_s + C_p)\right)}{R_p + \left(\left(\frac{\omega L_s}{R_s}\right)^2 + 1\right) R_s} \tag{37}$$

An inductor is at self-resonance when the peak magnetic and electric energies are equal. Therefore, Q vanishes to zero at the self-resonance frequency ω_{sr} i.e.:

$$Q = \frac{R_s^2 (C_s + C_p)}{L_s} + \omega_{sr}^2 L_s (C_s + C_p) = 1 \tag{38}$$

Above the self-resonance frequency, no net magnetic energy is available and thus it is generally required that $\omega_{sr} \geq \omega_{sr,min}$, where $\omega_{sr,min}$ is the desired minimal self-resonance frequency.

Formulation of the optimization problem

For a required value L_{req} of the inductance, the optimization consists in determining the values of the layout parameters (i.e n, w, s, d_{out} and d_{in}) which maximizes the quality factor while ensuring the desired minimal self-resonance frequency $\omega_{sr,min}$. In addition some geometry constraints must be added such as: a minimum turn width w_{min}, a minimum spacing s_{min}, a minimum inner diameter $d_{in,min}$ and a maximum outer diameter $d_{out,max}$ which limit the inductor area. The design variables d_{in} and d_{out} are not independents and are related to the other design variables by the expression $d_{in} + 2(n-1)s + 2nw = d_{out}$. Since s is typically small compared to d_{in}, d_{out} and w, we can recast this equality constraint as the inequality constraint: $d_{in} + 2n(w+s) \leq d_{out}$. The optimal design problem of the inductor can then be formulated as:

$$\begin{aligned}
&\text{minimize} && Q \\
&\text{subject to} && L_s = L_{req} \\
& && \omega_{sr} \geq \omega_{sr,min} \\
& && d_{in} + 2n(w+s) \leq d_{out} \\
& && s \geq s_{min}, \; w \geq w_{min} \\
& && d_{in} \geq d_{in,min}, \; d_{out} \leq d_{out,max}
\end{aligned} \tag{39}$$

This optimization problem can be solved using, for instance, a genetic algorithm (see section 2.2). However, problem (39) can be also formulated as a QGP problem.

Indeed, after some basic manipulations we get the following equivalent problem:

$$\begin{aligned}
&\text{minimize} && \alpha \\
&\text{subject to} && \frac{\alpha R_s}{\omega L_s R_p}\left(\frac{\omega^2 L_s^2}{R_s}+R_s+R_p\right)+(C_s+C_p)\left(\frac{R_s^2}{L_s}+\omega^2 L_s\right)\le 1 \\
& && \omega_{smin}^2 L_s(C_s+C_p)+R_s^2(C_s+C_p)/L_s \le 1 \\
& && L_{smin} \ge L_{req} \\
& && d_{in}+2n(w+s)\le d_{out} \\
& && s \ge s_{min},\ w \ge w_{min} \\
& && d_{in} \ge d_{in,min},\ d_{out} \le d_{out,max}
\end{aligned} \tag{40}$$

where α is an additional variable, L_s, R_s, C_s, R_p and C_p are given by (35). Thus formulated, the problem (40) is QGP in the design variables d_{in} and d_{out} and so can be efficiently solved using the approach described in section 3.

Numerical experiments

Problem (40) has been solved using the Nelder-Mead simplex method[10] based QGP (NMSM-QGP), the results thus obtained were then compared to those obtained using a standard genetic algorithm (GA). In our experiments, the following parameters have been used:

$$\begin{aligned}
&c_1=1.27,\quad c_2=2.07,\quad c_3=0.18,\quad c_4=0.13,\quad c_5=4,\quad \rho=2\times10^{-8}\,\Omega\text{m} \\
&t=10^{-6}\,\text{m},\quad \omega=3\pi10^9\,\text{rad/s},\quad \varepsilon_{ox}=3.45\times10^{-6}\,\text{F/m},\quad t_{ox}=4.5\times10^{-6}\,\text{m} \\
&t_{ox,M_1-M_2}=1.3\times10^{-6}\,\text{m},\quad C_{sub}=1.6\times10^{-6}\,\text{F},\quad G_{sub}=4\times10^4\,\text{S/m}^2 \\
&s_{min}=w_{min}=1.9\times10^{-6}\,\text{m},\quad d_{in,min}=10^{-4}\,\text{m},\quad d_{out,min}=4\times10^{-4}\,\text{m} \\
&\omega_{smin}=8\pi\times10^9\,\text{rad/s},\quad L_{req}=30\times10^{-9}\,\text{H}.
\end{aligned} \tag{41}$$

The solutions found via NMSM-QGP and GA are presented in Table 9. As we can see, the result obtained using NMSM-QGP is significantly better than the solution found by GA. However, despite a small number of function evaluations (which is in fact a number of GP-solver call), the computation time is large compared to GA. This is because the time cost for a GP-solver call is generally higher than the time cost of the objective function.

Table 9. *Comparison of the solutions found via GA (with N=200, N_G=150, p_c=0.7, p_m=0.07) and NMSM-QGP (with the starting point (d_{in}=200, d_{out}=300).*

Method	n	w	s	d_{in}	d_{out}	L_s	Q	NbEval	CPU Time
Standard GA	9.440	4.491	3.73	147.60	309.07	29.99	2.821	30000	5 s
NMSM-QGP	10.862	3.683	1.90	111.5	232.82	30.00	3.233	47	64 s

CONCLUSIONS

It is a matter of fact that Nature has been, and is always, a major source of inspiration for scientific and technical developments. Optimization does not escape to this rule and many

[10] The Nelder-Mead simplex method is available in MatLab through the function `fminsearch`. In this example, the following parameters have been used for the stopping rule: `TolFun`=10^{-5} and `TolX`=0.1, where `TolFun` is the termination tolerance on the function value and `TolX` is the termination tolerance.

heuristic searches draw their foundations from physical or biological principles such as the main approaches reviewed in this chapter. Although they are pale imitations of the reality, these approaches have proven their efficiency in solving difficult optimizations problems. One of the main purposes with this chapter was to provide the essential ideas behind each presented optimization method as well as the algorithm and the usually adopted parameter setting. This could help the reader in the practical use of these methods. In addition to the standard stochastic algorithm, we have presented a recently developed optimization method called HKA as well as an extension of standard geometric programming, called QGP.

The ability of **HKA** and **QGP** in solving difficult non-convex problem has been shown on many practical examples. In particular, we have addressed the problems of robust structured control and on-chip spiral inductor design. These topics lead indeed to non-convex constrained optimization problems which are known to be difficult to deal with using conventional methods. We have shown that stochastic methods in general and HKA/QGP in particular can be used to find out, in a straightforward manner, if not the optimal solution but at least a suboptimal one, which is very useful for the practitioner.

REFERENCES

Angeline, P. J. (1998). Using selection improve particle swarm optimization. In *Proceedings of the IEEE International Conference on Evolutionary Computation* (pp. 84-89). Piscataway, NJ: IEEE Press.

Apkarian, P., Bompart, V., & Noll, D. (2007). Nonsmooth structured control design with application to PID loop-shaping of a process. Int. J. Robust Nonlinear Control, vol. 17, pp. 1320-1342.

Boyd, S., & Vandenberghe, L. (2004). *Convex optimization*. New York: Cambridge University Press.

Boyd, S., Kim, S.-J., Vandenberghe, L., & Hassibi, A. (2007). A Tutorial on Geometric Programming. *Optimization and Engineering*, vol. 8(1), pp. 67-127.

Boyd, R., & Richardson, P. (1985). *Culture and the evolutionary process*. Chicago: University of Chicago Press.

Cerny, V. (1985). A thermodynamical approach to the travelling salesman problem: an efficient simulation algorithm. *Journal of Optimization Theory and Applications, 43*(1), 41-51.

Chakrabarti, R., Chattopadhyay, P. K., Basu, M. and Panigrahi, C. K.. (2006). Particle swarm optimization technique for dynamic economic dispatch. IE (I) Journal-EL, 87(3):48–54.

Coello, C. A. C. (2000). Use of a self-adaptive penalty approach for engineering optimization problems. *Computers in Industry, 41*(2), 113-127.

Coello, C. A. C. (2002). Theoretical and numerical constraint handling techniques used with evolutionary algorithms: a survey of the state of the art. *Computer Methods in Applied Mechanics and Engineering, 191*(11-12), 1245-1287.

Coello, C. A. C., Lamont G. B. & Van Veldhuizen D. A. (2007). *Evolutionary Algorithms for Solving Multi-Objective Problems*. Springer.

Clerc, M. (2010). *Particle Swarm Optimization for hard optimization*. John Wiley & Sons.

Dréo, J., Pétrowski, A., Siarry, P., & Taillard, E. (2006). *Metaheuristic for hard optimization*. Berlin: Springer-Verlag.

Fogel, D. B. (2006). *Evolutionary computation: towards a new philosophy of machine intelligence*. New York: Wiley-IEEE Press.

Garey, M. R., & Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco: Freeman

Grant, M. & Boyd, S. (2010). *CVX: Matlab Software for Disciplined Convex Programming, version 1.21*. Available at: http://cvxr.com/cvx.

Genc, A. U. (2000). A state-space algorithm for designing H∞ loopshaping PID controllers. Tech. report, Cambridge University, Cambridge, UK.

Goldberg, D. E. (2013). *Genetic Algorithms*. Pearson Education.

Holland, J. H. (1962). Outline for logical theory of adaptive systems. *Journal of the ACM, 9*(3), 297-394.

Jamshidi, M., Krohling, R., Dos Santos Coelho L., & Fleming, P. (2002). *Robust control system with genetic algorithms*. Boca Raton, FL: CRC Press (Taylor and Francis Group).

Kelley, C. T. (1999). *Iterative Methods for Optimization*. SIAM Frontiers in Applied Mathematics, N° 18.

Kennedy, J., & Eberhart, R. C. (1995). Particle swarm optimization. In *Proceedings of IEEE International Conference on Neural Networks* (pp. 1942-1948). Piscataway, NJ: IEEE Press.

Kim, T. H., Maruta, I., & Sugie, T. (2008). Robust PID controller tuning based on the constrained particle swarm optimization. *Automatica, 44*(4), 1104-1110.

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by Simulated Annealing. *Science, 220*(4598), 671-680.

Lobo, F., Lima, C. F., & Michalewicz, Z. (Eds.). (2007). *Parameter Setting in Evolutionary Algorithms. Studies in Computational Intelligence*. Berlin:Springer Verlag.

Lovberg, M., & Krink, T. (2002). Extending particle swarm optimizers with self-organized criticality. In *Proceedings of the fourth congress on evolutionary computation, Vol. 2* (pp. 1588-1593).

Maruta, L., Kim, T. H., & Sugie, T. (2008). Synthesis of fixed-structure Hinf, controllers via Constrained Particle Swarm Optimization. In *Proceedings of the 17th IFAC World Congress*, Seoul, Korea.

McFarlane, D. & Glover K. (1992). A loop shaping design procedure using H∞ synthesis. IEEE Transactions on Automatic Control, vol. 37, pp. 759-769.

Motoda, T., Stengel, R. F., Miyazawa, Y. (2002). Robust Control System Design Using Simulated Annealing. *Journal of Guidance, Control, and Dynamics 25*(2), 267-274.

Rockafellar, R. T. (1993). Lagrange multipliers and optimality. *SIAM Review*, *35*(2), 183-238.

Saeki, M. (2006). Fixed structure PID controller design for standard H∞ control problem. *Automatica*, vol. 42, pp. 93-100, 2006.

Spall, J. C. (2003). *Introduction to stochastic search and optimization*. New York: Wiley-Interscience, John Wiley & Sons.

Toscano, R., & Lyonnet, P. (2009a). Heuristic Kalman Algorithm for solving optimization problems. *IEEE Transaction on Systems, Man, and Cybernetics, Part B.* Vol. 35(5), pp. 1231-1244.

Toscano, R., & Lyonnet, P. (2009b). Robust PID controller tuning based on the Heuristic Kalman Algorithm. *Automatica*, Vol. 45(9), pp. 2099-2106.

Toscano, R., & Lyonnet, P. (2010). A new heuristic approach for non-convex optimization. *Information Sciences.* Vol. 180(10), pp. 1955-1966.

Toscano, R. (2013). *Structured Controllers for Uncertain Systems. A Stochastic Optimization Approach*. Springer-Verlag.

Toscano, R., & Lyonnet, P. (2012). A Kalman optimization approach for solving some industrial electronics problems. *IEEE Transactions on Industrial Electronic*, Vol. 59, N° 11, pp. 4456-4464, 2012.

Toscano, R., & Amouri, S. B. (2012). Some heuristic approaches for solving extended geometric programming problems. *Engineering Optimization*, Vol. 44, N° 12, pp. 1425-1446.

Yue, C. P., Ryu, C., Lau, J., Lee, T. H. & Wong, S. S. (1996). A physical model for planar spiral inductors on silicon. In *Proceedings of the IEEE International conference on Electronic Devices*. San Francisco, CA.