Contribution to the Optimization of Closed-Loop Multibody Systems : Application to Parallel Manipulators

J.-F. COLLARD[1], P. FISETTE[1] and P. DUYSINX[2]
[1] *Université Catholique de Louvain, Center for Research in Mechatronics, place du Levant, 2, B-1348 Louvain-la-Neuve, Belgium;*
[2] *Université de Liège, Département ProMéThé, chemin des Chevreuils, 1, Bât B52, B-4000 Liège, Belgium*

March 1, 2004

Abstract. This paper describes an original and robust method to optimize the design of closed-loop mechanisms, especially parallel manipulators. These mechanisms involve non linear assembling constraints. During optimization, the Newton-Raphson algorithm we use to solve these constraints may fail when the Jacobian matrix of the constraints is ill-conditioned and stops the redesign process. To circumvent the difficulty, the technique we propose takes advantage of numerical conditioning to penalize the objective function. Applications to an academic example and parallel robots demonstrate the capabilities of the methodology.

Keywords: design optimization, penalty, closed-loop multibody systems, parallel manipulators

1. Introduction

Most of present issues in the field of multibody system (MBS) dynamics involve other scientific disciplines to enlarge and enrich the results of the MBS analysis. Combining multibody analysis and optimization techniques has sometimes been exploited in the literature (e.g. : [1, 2, 3]), but the few existing results are still not able to cope completely with some limitations and/or conflicts. Several issues are addressed in recent researches: applicability of optimization methods for some families of MBS, problem formulation in terms of cost function, especially for constrained systems, computer efficiency and parallel computation algorithms, etc. However, current researches mainly produce guidelines rather than rigid rules regarding the coupling of multibody dynamics and optimization in a particular context, i.e. a family of applications and a given type of objective function. As examples, one can cite the optimal design of a vehicle transmission [1], of 3D manipulators [4, 5, 6, 7, 8], the optimal synthesis of mechanisms [2, 9, 10] and parallel robots [11, 12, 13] as well as the optimization of suspension and railway vehicle [3, 14, 15], or machine tools [16].

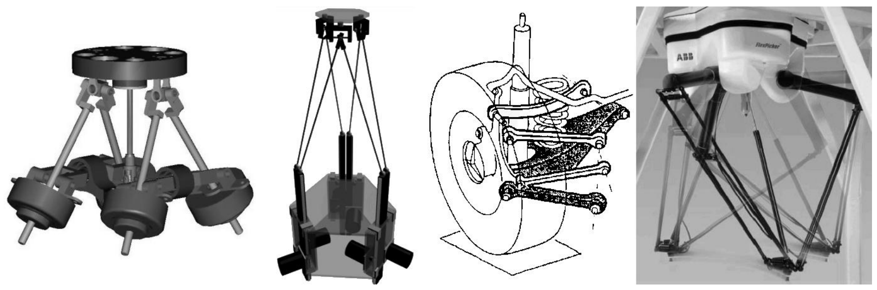


Figure 1. Examples of closed-loop mechanisms

The present research tackles the optimization problem of MBS containing three-dimensional kinematic loops as shown in Figure 1, involving :

- *geometrical* design parameters;
- a single scalar objective function whose nature may be *geometrical*, *kinematic* or *dynamical*.

In this case, the question that is addressed is the following: how to build a robust cost function for those systems such that a classical optimization method can iterate with good convergence properties and without troubleshooting in terms of loops closure (or "system assembling") ? For simple systems, one can obviously circumvent the problem by expressing *explicitly* some optimization constraints on the geometrical parameters (length of segments, amplitude of motion, etc.), but, as soon as complex multi-loop systems like 3D parallel manipulators are involved, this is not possible anymore. The original approach that is used here is based on a penalty technique in which unreachable configurations (i.e. configurations where it is not possible to assemble or to actuate the mechanism) and design constraint violations are avoided by a large cost of the objective function. The proposed method extends the parameter space and thus the cost function definition domain beyond the feasible and assembly domain, leading to a continuous and well-defined cost function. Hence, this allows to use efficient unconstrained optimization methods as BFGS method, Nelder-Mead Simplex, etc...

The approach will be illustrated at first on the basis of a quite academic example: the design optimization of a planar ejector to improve dynamical performances. Then the method will be applied on more realistic problems: the dexterity optimization of a parallel manipulator – the Hunt platform – to enhance kinematic performances. More precisely, the goal is to maximize the average robot isotropy over a workspace cube with respect to geometric design parameters.

2. Optimization and Assembling Constraints

Before expressing the limitations due to assembling constraints, we will briefly describe the multibody formalisms we use to compute the inverse and direct models of constrained MBS which are generally involved in the computation of the objective function. Indeed, it is important to know the origin of these assembling constraints in the case of closed-loop mechanisms. MBS dynamical formalisms will be described first. Main issues due to assembling constraints will be explained thereafter.

2.1. MBS dynamical formalisms

Most multibody applications contain loops of bodies (car suspension, parallel robots, mechanisms, etc...) which force the generalized coordinates q – relative joint coordinates in our formalism – to satisfy m geometrical constraints $h(q) = 0$ at any time. In order to fully describe the system, these assembling constraints and their first and second time derivatives must be added to the equations of motion, in which constraint forces are introduced via the Lagrange multipliers technique:

$$M(q)\,\ddot{q} + c(q, \dot{q}, f_{ext}, t_{ext}, g) = Q(q, \dot{q}) + J^t \lambda \tag{1}$$

$$h(q) = 0 \tag{2}$$

$$\dot{h}(q, \dot{q}) = J(q)\dot{q} = 0 \tag{3}$$

$$\ddot{h}(q, \dot{q}, \ddot{q}) = J(q)\ddot{q} + \dot{J}\dot{q}(q, \dot{q}) = 0 \tag{4}$$

where:

- M $[n * n]$ is the symmetric generalized mass matrix of the system,
- q $[n * 1]$ denotes the relative – or joint – generalized coordinates,
- $c\,[n * 1]$ is the non linear dynamical vector which contains the gyroscopic, centrifugal and gravity terms as well as the contribution of components of external resultant forces f_{ext} and torques t_{ext},
- Q $[n * 1]$ represents the generalized joint forces (torques).
- $J = \frac{\partial h}{\partial q^t}$ denotes the constraint Jacobian matrix (dimension: $[m * n]$),
- $\dot{J}\dot{q}(q, \dot{q})$ $[m * 1]$ is the quadratic term (expression in $\dot{q}^i\,\dot{q}^j$) of the constraints at acceleration level (dimension: $[m * 1]$),
- λ represents the Lagrange multipliers associated with the constraints (dimension: $[m * 1]$).

Various methods can be used to solve the system (1–4), i.e. to predict the motion of the system $(q(t), \dot{q}(t))$, starting from an initial configuration $(q(t=0), \dot{q}(t=0))$, by time-integrating the accelerations $\ddot{q}(t)$. Amongst these, one can opt for a full reduction of the system to a purely differential form, which can be obtained by means of the *Coordinate Partitioning* [17]. Assuming that the constraints $h(q) = 0$ are independent and after reordering the vector of generalized coordinates q (and the columns of the constraint Jacobian J accordingly), we can perform the following partition:

$$q = \begin{pmatrix} u \\ v \end{pmatrix} ; \; J = \begin{pmatrix} J_u & J_v \end{pmatrix} \tag{5}$$

where u denotes the subset of $(n-m)$ *independent* coordinates and v denotes the subset of m *dependent* coordinates. By correctly[1] choosing the subset v, the m by m matrix J_v will be regular.

Once the coordinate partitioning is established, the reduction method simply uses matrix permutations and operations to produce the equations of motion in ODE form. Let us first partition the generalized mass matrix M and the vector c according to the coordinate partitioning (5):

$$\begin{pmatrix} M_{uu} & M_{uv} \\ M_{vu} & M_{vv} \end{pmatrix} \begin{pmatrix} \ddot{u} \\ \ddot{v} \end{pmatrix} + \begin{pmatrix} c_u \\ c_v \end{pmatrix} = \begin{pmatrix} Q_u \\ Q_v \end{pmatrix} + \begin{pmatrix} {J_u}^t \\ {J_v}^t \end{pmatrix} \lambda \tag{6}$$

When J_v is regular, eliminating the unknowns λ using the lower part of system (6) produces:

$$\begin{pmatrix} M_{uu} & M_{uv} \end{pmatrix} \begin{pmatrix} \ddot{u} \\ \ddot{v} \end{pmatrix} + {B_{vu}}^t \begin{pmatrix} M_{vu} & M_{vv} \end{pmatrix} \begin{pmatrix} \ddot{u} \\ \ddot{v} \end{pmatrix} + c_u + {B_{vu}}^t c_v = Q_u + B_{vu}^t Q_v \tag{7}$$

where we define the so-called coupling matrix $B_{vu} \triangleq -(J_v)^{-1} J_u$.

Then using the first (eq. (3)) and second derivatives (eq. (4)) of the constraints, the generalized velocities and accelerations $\dot{v}$ and $\ddot{v}$ are respectively given by:

$$\dot{v} = B_{vu} \dot{u} \tag{8}$$

$$\ddot{v} = B_{vu} \ddot{u} + b \quad \text{with} \quad b \triangleq -J_v^{-1} (\dot{J} \, \dot{q}) \tag{9}$$

and can also be eliminated from the differential equations (7). This produces the final *reduced* ODE system, concisely written as:

$$\mathcal{M}(u,v)\ddot{u} + \mathcal{F}(\dot{u},u,v) = \mathcal{Q}(\dot{u},u,v) \tag{10}$$

[1] This constitutes one of the issues explained below in 2.2.

where $\mathcal{Q}$ denotes the joint generalized forces (torques) associated with the actuated[2] joints. From this system, direct ($\ddot{u} = \mathcal{M}^{-1}(\mathcal{Q} - \mathcal{F})$) and inverse[3] ($\mathcal{Q} = \mathcal{M}\ddot{u} + \mathcal{F}$) dynamic formulations may be easily extracted.

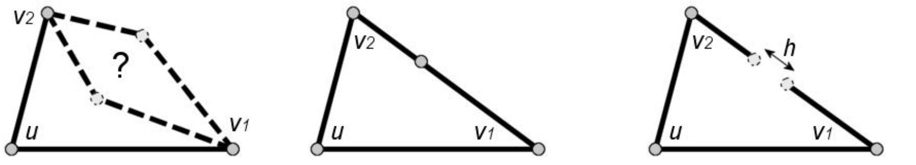
The algebraic constraints still have to be solved in order to eliminate the dependent variables v from (10). While analytical solutions can exist for specific cases, general algebraic constraints (2) require a numerical procedure to be solved: the Newton-Raphson iterative algorithm can be used for successive estimations of v:

$$v^{k+1} = v^k - (J_v)^{-1}\, h\,|_{v=v^k} \tag{11}$$

where the right hand side is evaluated for $v = v^k$ and the values of u corresponding to the instantaneous system configuration.
Thanks to this final numerical elimination, the set of purely differential equations (10) constitutes the equations of motion of the constrained system described in terms of the $n-m$ independent generalized coordinates u.

2.2. Main issues due to assembling constraints

In some cases, we may face convergence problems of the Newton-Raphson algorithm to solve the constraints (11) and such problems can be frequently encountered when performing a geometrical optimization process.



a. Multiple solutions b. Singularity: $\left|\frac{\partial h}{\partial v^t}\right| = 0$ c. Impossibility: $h(u,v) \neq 0, \forall v$

Figure 2. Troubleshooting cases when solving assembling constraints

- A first case may occur when the solution in term of v is not unique for a given vector u (see Figure 2.a). A way to prevent from that is to start the algorithm with initial values close to the expected solution.

[2] Which do not necessary coincide with the independent joint coordinates u.
[3] In practice, inverse dynamics does not require the explicit computation of the mass matrix ($O(N^2)$ complexity). An $O(N)$ recursive formalism is used to get $\mathcal{Q}$ more straightforwardly.

- A second problem may happen if the mechanism reaches a singular configuration, the constraint Jacobian matrix $J_v = \frac{\partial h}{\partial v^t}$ becoming singular[4] (see Figure 2.b). Practically, those singularities correspond to a loss of mobility of the robot by locking one or more actuators associated with joint variables u. They can be assimilated to unreachable points in the parameter space. Therefore, the cost function will be penalized at that point, even if it is possible to close the mechanism by another choice of the partitioning.

- Finally, it can be impossible to close the mechanism simply because a constraint h_i has no root: the Newton-Raphson algorithm cannot converge towards a solution (see Figure 2.c). In that case, the closed mechanism doesn't exist but, instead of rejecting it, we will keep it and penalize the cost function accordingly.

In the optimization process, the second and third cases will be treated in the same way, as explained in the next section. For the first case, we rely on the robustness of the Newton-Raphson process when iterating from a neighborhood of a well-known closed configuration. Other configurations are found by small increment around and from the latter: by experience, this technique is reliable and ensures a robust convergence in any case.

Let's point out that the penalization of the cost function which is caused by unsatisfied geometrical constraints may occur whatever is the nature (geometrical, kinematic or dynamical) of the cost function itself.

3. Penalty Optimization Method

3.1. How to Extend the Objective Function outside the Closed-Loop Domain ?

In order to perform a smooth penalization of the objective function $f(P)$, it is important to find the closed-loop border according to the u, v partitioning in the parameter space. The idea here is to observe the conditioning of the constraint Jacobian matrix J_v which indicates

[4] Two kinds of singularities are considered here: the mathematical singularities, arising from a wrong choice of the sequence of rotation variables, and those associated with ignorable variables such as the axial rotation of a connecting rod. They can be avoided by achieving a suitable model of the mechanism.

the proximity of that border where the determinant of J_v vanishes to zero[5].

Let's take an application with two parameters P_1,P_2 (see Figure 3). Let's suppose that the optimizer is calling the objective function – assumed to be scalar – outside the closed-loop border, at point X. Then a fixed point G is chosen inside the boundary[6] and the penalization is computed along the direction GX from a point B'. The latter is located close to the border, where the absolute value of the determinant of J_v reaches a threshold ϵ (strictly greater than zero), to avoid singular configurations and numerical round-off problems. The closed-loop border is for instance detected on the basis of the number *iter* of iterations of the Newton-Raphson algorithm which reaches the maximum $iter_M$[7].

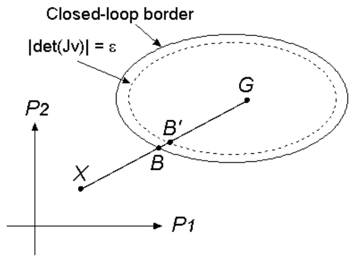


Figure 3. Penalization along direction GX

The search of B' is simply done by a dichotomic search along the segment $[G, X]$ (see Figure 3). Various cases are considered:

- The Newton-Raphson algorithm converges: $iter < iter_M \Rightarrow [B, G]$ segment:
    - $|det(J_v)| \geq \epsilon \Rightarrow [B', G]$ segment
    - $|det(J_v)| \leq \epsilon \Rightarrow [B', B]$ segment
- The Newton-Raphson algorithm doesn't converge: $iter \geq iter_M \Rightarrow [B, X]$ segment

Once point B' has been found and $f(B')$ has been evaluated, we suggest to compute and penalize $f(X)$, on the basis of $f(B')$. This extension may be continuous or not, the only condition is to make sure that

[5] The present approach is clearly partitioning-dependent. A way to circumvent this limitation should be to consider all the possible coherent partitioning to find the border proximity in the parameter space.

[6] Various possibilities can be envisaged to choose point G: for instance, the last update of the optimal parameter values.

[7] Typically, $iter_M = 20$.

$f(X) > f(B')$. In the proposed examples, different kinds of extension have been applied and a rather simple algorithm has been used: the Nelder-Mead simplex method, which is robust but slowly convergent.

3.2. The Objective Function Algorithm

The optimization procedure flowchart is given in Figure 4.

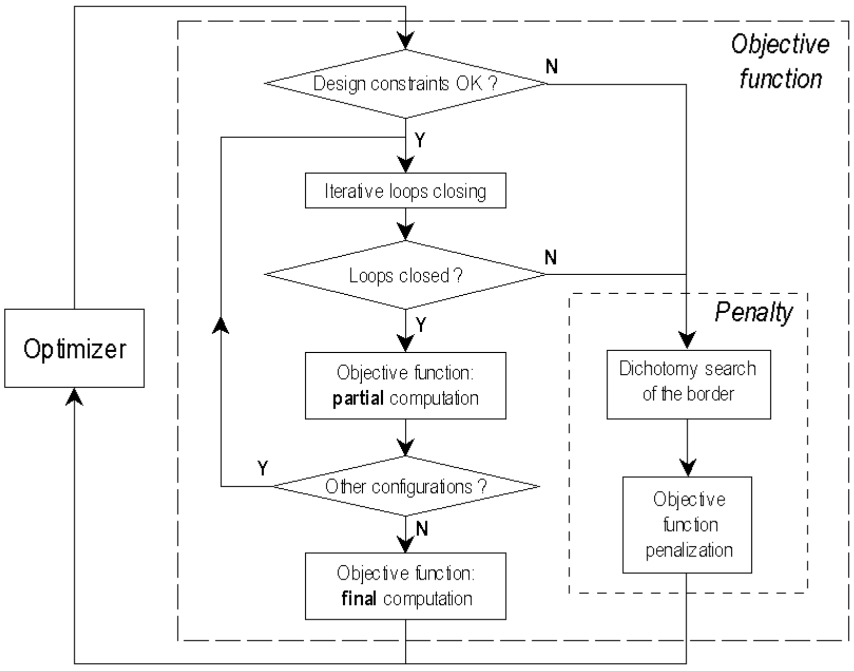

Objective function
Design constraints OK ?
N
Y
Iterative loops closing
Loops closed ?
N
Y
Penalty
Optimizer
Objective function: partial computation
Dichotomy search of the border
Y
Other configurations ?
N
Objective function penalization
Objective function: final computation

Figure 4. Objective function computation algorithm

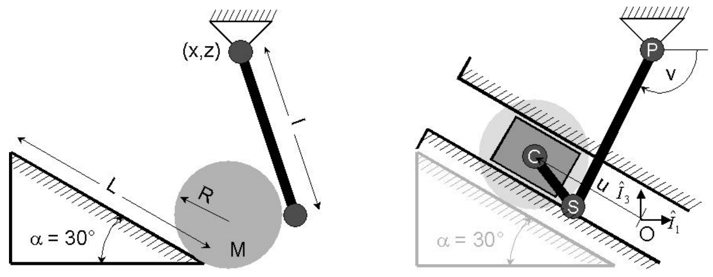
Each time the optimizer calls the objective function, design constraints[8] imposed by the designer are evaluated first. If they are satisfied, the mechanism can be assembled using the Newton-Raphson algorithm described before. If the latter converges, the objective function which involves that configuration (or a set of configurations, as in the application section 5.2) can be evaluated. Otherwise, if one of both tests is unsatisfied, the penalization process[9] begins: first, the border is found by the dichotomy method as explained before and then, the objective function is extended continuously and returns a penalized

[8] e.g. : upper or lower bound of a parameter.
[9] Up to now, we systematically deal with the design constraints in the same way as the closed-loop constraints, using the same penalty approach.

value to the optimizer. However, in some cases, if an upper bound of the objective function is known, it may be quickly extended discontinuously without searching the border[10].

4. First Example: Design of a Planar Ejector

The system (see Figure 5.a) consists of two bodies: a ball (radius: $R = 7\ cm$, mass: $M = 300\ g$) and a simple articulated arm (negligible mass) which has to push the ball with a torque Q_P over a distance $L = 20\ cm$, slope: $\alpha = 30°$. The problem was stated in the frame of a mobile robotic project for which such an ejector had to be designed. The contact point S is supposed to be fixed on the right arm tip thanks to a roller (whose axial rotation is disregarded). This first - quite academic - example can thus be modelled as a slider-crank mechanism (see Figure 5.b) whose optimization parameters are the position (x, z) of the joint P and the length of the crank $\|\overrightarrow{PS}\| = l$ ($\|\overrightarrow{CS}\| = R$). All the coordinates are expressed in the inertial frame $\{O, \hat{I}_1, \hat{I}_3\}$, where O is located at the origin of the ball movement.



a. Planar ball ejector... b. ...modelled by a slider-crank mechanism

Figure 5. Planar ejector model

Two different formulations of the optimal design problem are proposed:

- The first one consists in minimizing the maximum torque Q_P necessary to supply a given constant acceleration a to the ball (i.e. a velocity ramp from 0 m/s to 0.4 m/s):

$$\min_{x,z,l} \left(\max_{t} \ Q_P \right) \tag{12}$$

- The second formulation is to maximize the velocity v_{final} of ejection (when the position u of the ball reaches l) for a given constant

[10] This is not represented on the flowchart.

torque Q_P (i.e. 1.4 Nm):

$$\max_{x,z,l}\left(v|_{u=l}\right) \tag{13}$$

These two objectives involves the computation of the inverse and direct dynamical equations of a closed-loop MBS respectively (see section 2.1). Thus, applying the *Coordinate Partitioning* Method [17] described in section 2.1, the partitioning is performed (see eq. (5)) where the independent coordinate $u = \|\overrightarrow{OC}\|$ and the dependent one v is the angle rotation of the joint P. The closed-loop constraint h of eq. (2) becomes here:

$$h(q) = \left\|\overrightarrow{OC} - \overrightarrow{OS}\right\|^2 - R^2 \tag{14}$$

$$\Leftrightarrow h(q) = \left(u\frac{\sqrt{3}}{2} + x + l\cos v\right)^2 + \left(\frac{u}{2} - z + l\sin v\right)^2 - R^2 \tag{15}$$

In the same way, the elements of the constraint Jacobian matrix (see (5)) are:

$$J_u = 2u + \sqrt{3}x - z + l(\sqrt{3}\cos v + \sin v) \tag{16}$$

$$J_v = l\left[\left(-u\sqrt{3} - 2x\right)\sin v + (u - 2z)\cos v\right] \tag{17}$$

For this simple case, the closed-loop constraint could be solved analytically but, in general, we use the Newton-Raphson algorithm (11) to find v with respect to u. Now, as shown in section 2.1, the generalized mass matrix M, the vector c and the joint forces (torques) Q may be computed:

$$M = \begin{pmatrix} M & 0 \\ 0 & 0 \end{pmatrix},\ c = \begin{pmatrix} \frac{Mg}{2} \\ 0 \end{pmatrix},\ Q = \begin{pmatrix} 0 \\ Q_P \end{pmatrix} \tag{18}$$

Finally, introducing (16, 17, 18) into (7), and using the coupling matrix $B_{vu} \triangleq -\left(J_v\right)^{-1} J_u$, the reduced system of this example is:

$$Q_P = B_{vu}^{-1} M\left(\ddot{u} + \frac{g}{2}\right) \tag{19}$$

This constitutes the inverse dynamical model from which the direct dynamical model can be trivially derived ($\ddot{u} = \ldots$).

Coming back to our objective functions, the first one (eq. (12)) to minimize gives us:

$$\max_t \left[B_{vu}^{-1} M\left(\ddot{u} + \frac{g}{2}\right)\right] = \max_t B_{vu}^{-1}, \quad \text{since} \quad \ddot{u} = a \tag{20}$$

and the second one (eq. (12)) to maximize reduces to:

$$\int_0^{t^*} B_{vu}\frac{Q_P}{M} - \frac{g}{2}\, dt \quad \text{with} \quad t^* \quad \text{s.t.} \quad u(t^*) = L \tag{21}$$

One restriction of the optimization problems is thus the closed-loop constraint which is *safely* satisfied (see section 3.1) if:

$$iter < iter_M \quad \text{and} \quad |J_v(u(t))| \geq \epsilon \tag{22}$$

To ensure that the ball is pushed instead of being pulled[11], another restriction has to be added to ensures that the crank (the arm) must always be located *safely* ($\epsilon > 0$) behind the ball:

$$J_u(u(t)) \geq \epsilon \tag{23}$$

Remark that both constraints (22,23) have to be satisfied $\forall\ u \in [0; L]$ (1^{st} obj), or $\forall\ t \in [0; t^*]$ (2^{nd} obj).

Finally, systems of equations (20,22,23) and (21,22,23) constitute both optimization problems. Now, the extended objective functions with penalization are :

$$f_{ext}(x,z,l) = \begin{cases} f(x,z,l) & \text{if } iter < iter_M \\ & \quad \text{and } |J_v| \geq \epsilon \\ & \quad \text{and } J_u \geq \epsilon \\ f(x,z,l)|_{border} + s\ d & \text{otherwise} \end{cases} \tag{24}$$

where f is either (20) or (21), s is the slope[12] of the linear extension and d is the distance between the border and the point $X(x,z,l)$ along the direction GX (see section 3.1, figure 3). Note that the computation of $f(x,z,l)|_{border}$ depends on which border is firstly crossed starting from G and going to X.

The results for both objective functions are presented in table I. Starting with $x = -2$ cm, $z = 12$ cm and $l = 12$ cm, the optimal values are found with respectively 1108 and 390 evaluations of both objective functions (using the Nelder-Mead Simplex method and a continuous penalty extension as discussed in Section 3.1). The global computation takes less than 2 minutes with a standard PC in the Matlab® environment.

[11] Which is physically irrelevant for the envisaged ejector.
[12] Chosen value : 100.

Table I. optimization results of the planar ejector

	Min. max. torque ($a = 0.4$ m/s^2)		Max. final velocity ($Q_P = 1.4$ Nm)	
Final value	[Nm]	0.168	[m/s]	5.03
x	[cm]	-1.46	[cm]	-3.08
z	[cm]	7.04	[cm]	6.58
l	[cm]	9.57	[cm]	8.21

5. Application to Parallel Manipulators: Kinematic Conditioning Optimization

5.1. Dexterity of manipulators

Dexterity of a manipulator is a kinetostatic performance that can be measured from the condition number κ of its forward kinematics Jacobian J [18]. In other words, if this Jacobian J is defined by:

$$\dot{\mathrm{x}} = J\ \dot{\mathrm{q}} \tag{25}$$

where $\dot{\mathrm{q}}$ is the joint velocity vector and $\dot{\mathrm{x}}$ the velocity vector of the end-effector described by Cartesian coordinates (position and orientation), this dexterity index is the ratio of the largest singular value of J to its smallest one. This definition assumes that all entries of J have the same units. Otherwise, this dimensional inhomogeneity can be solved by introducing a normalizing *characteristic length* as suggested in [18]. The latter is used to divide the positioning rows of J, making it dimensionally homogeneous. As explained in [18], let us note that the value of the characteristic length itself comes from the minimization of the condition number over all the reachable configurations [18].

The goal is to optimize a global posture-independent performance index which is the mean of the inverses of the condition number κ over a volume V in the Cartesian space of the end-effector, also called Global Dexterity Index (GDI) [11] :

$$GDI = \frac{\int_V \frac{1}{\kappa}\, dV}{V} \tag{26}$$

In the case of positioning and orientating manipulators (for instance, the Hunt platform described below), the value of κ is obviously computed after normalizing the Jacobian matrix, as previously explained. By analogy with the optimization proposed in [18], we suggest that the above-mentioned characteristic length becomes an additional parameter of our optimization problem which initially only deals with design parameters.

5.2. Six-DoF Hunt Platform

The Hunt platform (see Figure 6) has 3 position and 3 orientation degrees of freedom which require to normalize J before computing κ, each time the parameters change, i.e. at each call of the objective function. As explained above, this involves an additional optimization parameter: the characteristic length L_C. The nine other parameters of this optimization problem (see Figures 6 and [7] for more details) are the legs lengths LI,LS, the characteristic radius of the platform RP and of the base RB, the gauge H between adjoining actuators on the base, the angle α (around a vertical axis), followed by angle β (around an horizontal axis), angle ψ, and finally the vertical distance zc between the base and the center of the desired workspace volume (small cube in Figure 6).

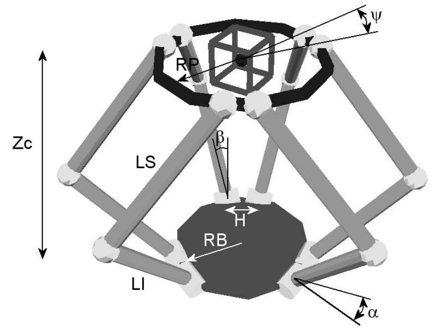


Figure 6. Hunt platform model

According to the research project specifications (for a surgical application), design parameters of this robot are limited by bounds (see Table II). The results obtained with the Nelder-Mead simplex method are presented in table II and initial and optimal design can be compared in Figure 7.

The “validation” of this result is made by using stochastic optimization algorithms for the same optimization problem: on the one hand, we apply a classical genetic algorithm and on the other hand, an evolutionary strategy is used.

Table II. Optimization bounds and results of the Hunt platform optimization

		minimum	maximum	Initial (G)	Optimal
Average Isotropy	[%]			26.45	*58.12*
zc	[mm]	100	500	300	263.6
LI	[mm]	50	300	200	220.4
LS	[mm]	50	300	200	300.0
RB	[mm]	50	200	100	50.0
RP	[mm]	50	200	100	163.5
H	[mm]	10	100	50	10.0
α	[°]	0	20	10	0.0
β	[°]	0	50	10	8.0
ψ	[°]	5	50	10	5.0
LC	[mm]	10^{-10}	1	10^{-2}	$6.1{\cdot}10^{-3}$

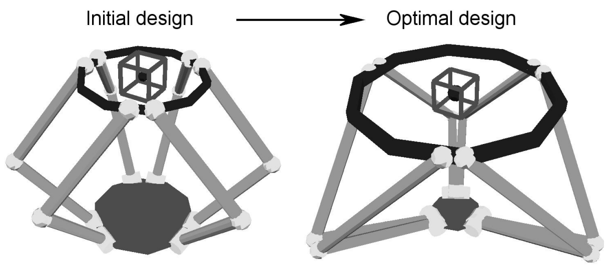


Figure 7. Initial and optimal designs of the Hunt platform

For the genetic algorithm[13], each of the 10 parameters is coded on 32 bits and the size of the population is 200. Three genetic operators are used to generate new populations. To begin, a selection is done by tournament taking – with a probability of 0.9 – the best of two individuals chosen randomly in the population. This generates half a population on which crossovers are performed, cutting chromosomes in two points. Finally, the mutation operator is applied on each bit with a fixed probability of 0.01. The evolution of the best fitness of each generation is plotted in Figure 8. The best fitness is found at the 407th generation and is equal to 58.07%.

In evolutionary strategies [3], a population of μ parents mutates in a population of λ offsprings by adding a Gaussian random variable to

[13] issued from the course of Prof. V. Magnin, Ecole polytechnique universitaire de Lille, France.

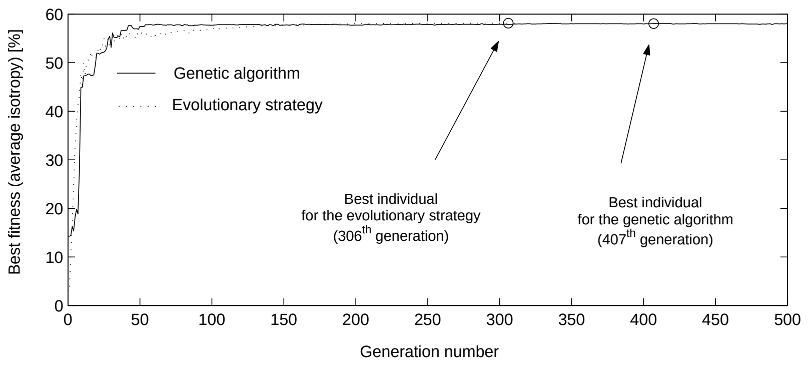


Figure 8. Optimization of the Hunt platform using stochastic methods

each optimization parameter. The standard deviation of that variable allows to control the speed of convergence [3]. It is also possible to enrich the algorithm by adding a step of recombination between parents before mutation. In our case, we choose $\mu = 20$ and $\lambda = 200$, and also a discrete recombination as described in [3]. In Figure 8, the evolution of the best fitness of each parent generation is plotted. The best one appeared at the 306^{th} generation and is equal to 58.12%.

If we now compare the results of three optimization methods (Simplex method, genetic algorithm and evolutionary strategy), we remark that they are very similar (see Table III). The slight differences comes probably from the different convergence criteria and the choice of the various parameters involved in the optimization process (e.g.: size of populations, selection and mutation probabilities,. . .). We may thus reasonably conclude that it should be a global optimum.

6. Conclusion and prospects

In this paper, a penalization method has been developed to optimize the design of closed-loop 3D mechanisms. The main issue lies in the assembling constraints and the way to solve them. So, we have shown how to exploit the conditioning of the constraints Jacobian matrix and/or the Newton-Raphson convergence to penalize the objective function. This enables to produce a nice optimization formulation that can be solved robustly, whatever the method used. Three applications are proposed: first, a simple planar ejector to illustrate the method, and then, a more 3D realistic 3D application dealing with parallel robot dexterity. Finally, a short comparison is made between optimization

Table III. Comparison between optimization methods

		Simplex	Genetic	Evolutionary
Average Isotropy	[%]	*58.12*	*58.07*	*58.12*
zc	[mm]	263.6	266.1	263.2
LI	[mm]	220.4	220.2	219.4
LS	[mm]	300.0	299.4	300.0
RB	[mm]	50.0	52.1	50.4
RP	[mm]	163.5	166.4	162.2
H	[mm]	10.0	10.3	10.0
α	[°]	0.0	0.1	0.0
β	[°]	8.0	8.4	7.4
ψ	[°]	5.0	5.1	5.0
LC	[μm]	6.1	6.0	6.2

results obtained with deterministic and stochastic methods, to assert the credibility of the method and of the solution.

In term of the prospects, we intend to develop further the proposed method:

- An interesting problem relates with the tuning of the parameters of the objective function computation algorithm detailed in section 3.1 (i.e. the choice of point G, the type of extension...)

- To make the method partitioning-independent, we will try to build a penalization criteria based on the global constraint Jacobian matrix conditioning instead of J_v (see section 3.1)

A last prospect concerns the optimization algorithm itself: other deterministic methods, more sophisticated than the Nelder-Mead Simplex, can be investigated and confronted.

Acknowledgements

This research has been sponsored by the Belgian Program on Interuniversity Attraction Poles initiated by the Belgian State — Prime Minister's Office — Science Policy Programming (IUAP V/6). The scientific responsibility is assumed by its authors.

References

1. Haj-Fraj, A. and F. Pfeiffer. Optimization of Automatic Gearshifting. In *Vehicle System Dynamics Supplement*, 35:207–222, 2001.
2. Jiminez, J. M., G. Alvarez, J. Cardenal, and J. Cuadrado. A simple and general method for kinematic synthesis of spatial mechanisms. In *Mechanism and Machine Theory*, 32:323–341, 1997.
3. Datoussaïd, S., O. Verlinden, and C. Conti. Application of Evolutionary Strategies to Optimal Design of Multibody Systems. In *Multibody Systems Dynamics*, 8(4):393-408, 2002.
4. Su, Y.X., B.Y. Duan, and C.H. Zheng. Genetic design of kinematically optimal fine tuning Stewart platform. In *Mechatronics*, 11:821–835, 2001.
5. Stocco, L.J., S.E. Salcudean, and F. Sassani. Optimal Kinematic Design of a Haptic Pen. In *IEEE/ASME Transactions on Mechatronics*, 6(3):210–220, 2001.
6. Fattah, A., and A.M. Hasan Ghasemi. *Isotropic Design of Spatial Parallel Manipulators.* In *The International Journal of Robotics Research*, 21(9):811–824, 2002.
7. Ryu, J. and J. Cha. Volumetric error analysis and architecture optimization for accuracy of HexaSlide type parallel manipulators. In *Mechanism and Machine Theory*, 38:227-240, 2003.
8. Ceccarelli , M., and C. Lanni. A multi-objective optimum design of general 3R manipulators for prescribed workspace limits. In *Mechanism and Machine Theory*, 39:119-132, 2004.
9. Vallejo, J, R. Aviles, A. Hernandez, and E. Amezua. Nonlinear optimization of planar linkages for kinematic syntheses. In *Mechanism and Machine Theory*, 30:501–518, 1995.
10. Cabrera, J.A., A. Simon, and M. Prado. Optimal synthesis of mechanisms with genetic algorithms. In *Mechanism and Machine Theory*, 37:1165–1177, 2002.
11. Gallant-Boudreau, M. and R. Boudreau. An Optimal Singularity-Free Planar Parallel Manipulator for a Prescribed Workspace Using a Genetic Algorithm. *Proc. of the IDMME'2000/Forum 2000 CSME Conference*, Montreal, 2000.
12. Gürsel A., and B. Shirinzadeh. Topology optimisation and singularity analysis of a 3-SPS parallel manipulator with a passive constraining spherical joint. In *Mechanism and Machine Theory*, 39:215-235, 2002.
13. Lemay, J., and L. Notash. Configuration engine for architecture planning of modular parallel robots. In *Mechanism and Machine Theory*, 39:101-117, 2004.
14. Datoussaïd, S. *Optimisatdion du comportement dynamique et cinématique de systèmes multicorps à structure cinématique complexe.* PhD thesis, Faculté Polytechnique de Mons, Belgium,1998.
15. Simionescu, P.A., and D. Beale. Synthesis and analysis of the five-link rear suspension system used in automobiles. In *Mechanism and Machine Theory*, 37:815-832, 2002.
16. Németh, I. *A CAD tool for the preliminary design of 3-axis machine tools: synthesis, analysis and optimisation.* PhD thesis, Katholieke Universiteit Leuven, Belgium,1998.
17. Wehage, R.-A., and E.-J. Haug. Generalized coordinate partitioning for dimension reduction in analysis of constrained dynamic systems. In *Journal of Mechanical Design*, 134;247–255, 1982.
18. Angeles, J. *Fundamentals of Robotic Mechanical Systems: theory, methods, and algoritnms.* Springer-Verlag, pp.174-190, 1997.