Contribution to the optimization of closed-loop multibody systems : application to parallel manipulators

J.-F. Collard [1], P. Fisette [1], P. Duysinx [2]

[1] Université Catholique de Louvain, Center for Research in Mechatronics
Place du Levant 2, 1348 Louvain-la-Neuve
[2] Université de Liège, Département ProMéThé
Rue Ernest SOLVAY 21, 4000 Liège
e-mail : collard@prm.ucl.ac.be, fisette@prm.ucl.ac.be, P.Duysinx@ulg.ac.be

1. Abstract
This paper describes an original and robust method to optimize the design of closed-loop mechanisms, especially parallel manipulators. In other words, these mechanisms include assembling constraints we solved using a Newton-Raphson algorithm which may fail when the Jacobian matrix of the constraints is ill-conditioned. Therefore, the technique we propose takes advantage of that conditioning to penalize properly the objective function. Applications are shown: on the one hand, a simple example about the design of a planar ejector and, on the other hand, more realistic examples about the kinematical properties of parallel robots, in particular Delta-type and HexaSlide-type manipulators.

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

3. Introduction
Most of present issues in the field of multibody system (MBS) dynamics involve other scientific disciplines in order to enlarge and enrich the results of the system analysis. Combining multibody analysis and optimization techniques has sometimes been exploited in the literature, but the few existing results are still not able to cope with some limitations and/or conflicts when applied to multibody systems (MBS). 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 resort to parallel computation, etc. With this respect, current researches mainly produce guidelines rather than rigid rules regarding the marriage of these two disciplines (multibody dynamics and optimization) in a given context, i.e. a family of applications and a given type of objective function (ex. : [1],[2],[3]).

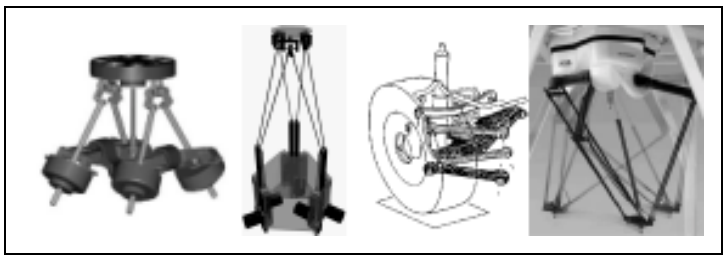

Fig. 1. Examples of closed-loop mechanisms

The present research tackles the problem of geometrical optimization of MBS containing three-dimensional closed loops as shown in Fig. 1. In this case, the question is : how to build a robust cost function for those systems in such a way that a classical optimization method can iterate with a good convergence and without troubleshooting in terms of loops closure (or "system assembling") ? For simple systems, we can obviously circumvent the problem by expressing explicitly some optimization constraints on the geometrical parameters (length of segments, amplitude of motion, etc.). However, as soon as complex multi-loops systems (ex. 3D parallel manipulators) are involved, such a method is not realistic or too restrictive. Thus, we propose an approach that uses an original cost penalty technique in which unreachable configurations (i.e. configuration where it is not possible to assemble or to control the mechanism) and other design constraints are properly pushed away in the cost function itself. The proposed method extends the cost function definition on the basis of elements like the conditioning of the constraints Jacobian matrix of the MBS, leading to a continuous and rather well-defined cost function. Hence, this allows the use of unconstrained optimization methods (BFGS method, Nelder-Mead Simplex…) which make the solution progress in an artificially extended parameter space.
The approach will be illustrated at first on the basis of a quite academic example (the torque minimization of a planar ejector) and then on more realistic applications : the dexterity optimization of Delta-type and HexaSlide-type parallel manipulators, whose goal is to maximize the mean robot isotropy over a cube with respect to design parameters (lengths of segments, radii of base and mobile platforms of the robot, distance between the cube and the base, etc.).
Once optimization of closed-loop mechanisms will be robustly achieved, the present research aims at tackling topological optimization of mechanisms. Emphasize will be shortly made on the advantages in using a dedicated symbolic multibody software to build cost and constraints functions for MBS, and to "procreate" new models straightforwardly, within a few milliseconds, which represents a real asset in the context of topological optimization.

4. Optimization and Assembling Constraints

The main issue with closed-loop MBS concerns assembling constraints. Generally speaking, the latter are highly non-linear and can be expressed by a set of implicit algebraic equations :

$$h(q) = 0 \tag{1}$$

where q is the vector of joint variables. Thus, assembling the mechanism means solving the set of constraints (1). And this has to be done each time the configuration of the mechanism changes. The technique we used to solve them is the "coordinates partitioning" method proposed in [4]. Joint variables q are first partitioned into independent variables u and dependent ones v :

$$q = [u \quad v] \tag{2}$$

Then, we compute v by means of the Newton-Raphson iterative algorithm :

$$v_{i+1} = v_i - J_v^{-1} h(q_i) \tag{3}$$

where

$$J_v = \left[\frac{\partial h}{\partial v^T} \right] \tag{4}$$

In some cases, we may meet problems of convergence of the assembling constraints during the optimization process.

- A first case may occur when the solution is not unique for a given vector u. A way to prevent from that is to start the algorithm with initial values close to the expected solution.
- A second problem may also happen if the mechanism reaches a singular configuration, the constraints Jacobian J_v becoming singular. Mathematically speaking, this can be avoided by modifying the chosen partitioning, which requires to reformulate the model. In practice, some of those singularities correspond to a loss of mobility of the actuator and are considered as unreachable points in the parameter space.
- Finally, it is also impossible to close the mechanism simply when a constraint h_i has no root : the Newton-Raphson algorithm stops when the maximum iteration number is reached. In that case, the mechanism doesn't exist and any objective function to optimize it has to be penalized as explained in the next section.

The second and third cases will be thus treated in the same way.

5. Penalty Optimization Method

5.1. How to extend the objective function outside the closed-loop border ?

In order to perform a smooth penalization of the objective function, it is important to find the closed-loop border in the parameter space. The idea here is to observe the conditioning of the constraints Jacobian J_v which indicates the proximity of that border where the determinant of J_v vanishes to zero.

Let's take a theoretical example with two parameters *P1*,*P2* (see Fig.2). Let suppose that the optimizer is calling the objective function outside the closed-loop border, at point *X*. Then a fixed point *G* is chosen inside the boundaries and the penalization is computed along the direction *GX* from a point *F*. The latter is located close to the border, where the absolute value of the determinant of J_v reaches a threshold (different from zero), to avoid singular configurations.

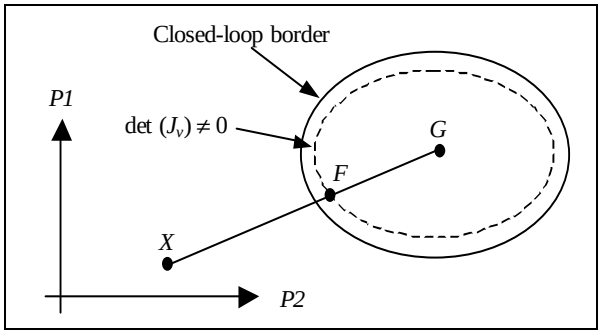

Fig. 2. Penalization along direction GX

The search of *F* is simply done by dichotomy on the segment [*G*,*X*]. Finally, the value of the objective function in *X* is an extrapolation of its value in *F*. This extension is continuous, monotonous (linear for instance), with a derivative which is not necessarily continuous in *F*. All those properties depend on the needs of the optimization algorithm. As proposed in [5], we apply here a continuous linear extension from *F* to *X* with a derivative discontinuity in *F*, and the algorithm used is the simplex (Nelder-Mead) which is robust but slowly converging. Later on, we intend to try other types of extension to be compatible with more sophisticated algorithms.

5.2. The objective function algorithm
Globally, the previous development takes place during the optimization process represented on the next diagram (Fig.3).

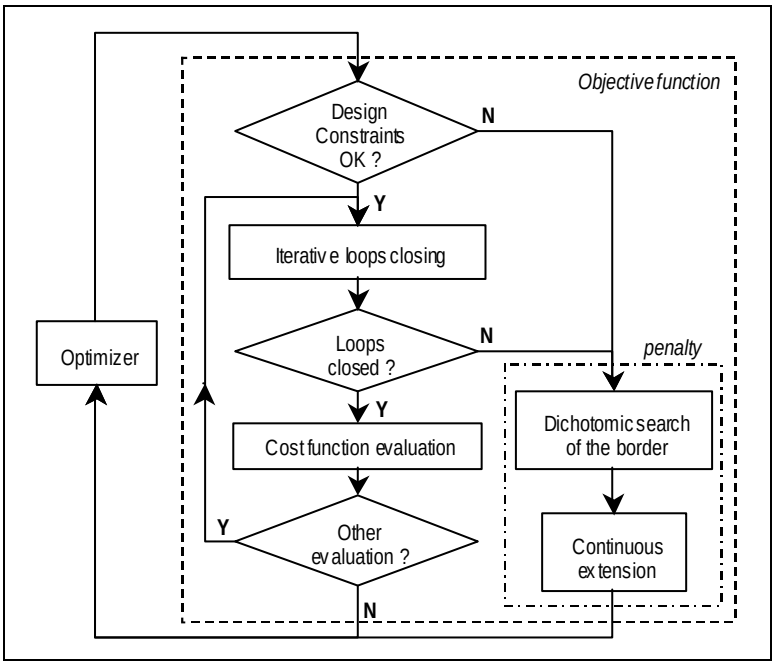

Fig. 3. Objective function calling algorithm

Each time the optimizer calls the objective function, design constraints, clearly identified in advance (or even imposed by the designer), are first evaluated. If they are satisfied, the mechanism is assembled by the Newton-Raphson algorithm described before. If it converges, the objective can be computed for that configuration and possibly more times for other ones. Otherwise, if one of both tests is unsatisfied, the penalization process begins. First, the border is found by dichotomy and then, the objective function is continuously extended to return an artificial value to the optimizer.

6. First example : design of a planar ejector
The system (drawn in gray, Fig. 4) consists of two bodies : a ball (radius : 7 cm, mass : 300g) and a simple articulated arm (mass : 100g) which has to push the ball over a distance *L* (20 cm, slope : 30°). The contact point is supposed fixed on the right arm tip thanks to a roller (whose axial rotation is disregarded). This first - quite academic - example can thus be modeled as a slider-crank mechanism whose optimization parameters are the position (*x*,*y*) of the rotation axis and the length of the crank *larm* (see Fig. 4).

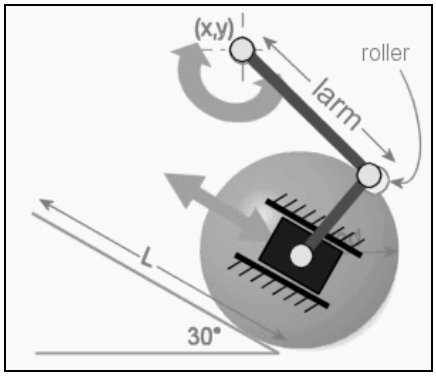

Fig. 4. Planar ejector model

Two different cost functions are tested.
- The first one to minimize is the maximum torque obtained for a given constant acceleration from a null velocity (i.e. a velocity ramp from 0 m/s to 0.4 m/s).
- The second objective to maximize is the velocity of ejection for a given constant torque (i.e. 1.4 Nm).

The restrictions of the optimization problem are the assembling constraints and a so-called “pushed-ball” constraint. The latter ensures that the ball is pushed instead of being pulled, meaning that the crank (the arm) must always be located behind the ball.
The results for both objective functions are shown in the next table.

Table 1. Optimization results of the planar ejector problem

	Max. torque (acc. = 0.4 m/s^2)	Velocity (T = 1.4 Nm)
Final value	0.168 Nm	5.03 m/s
x	-1.46 cm	-3.08 cm
y	7.04 cm	6.58 cm
larm	9.57 cm	8.21 cm

Starting with x = -2 cm, y = 12 cm and *larm* = 12 cm, the optimal values are found with 1108 and 390 evaluations of both objective functions respectively. This corresponds to computation times respectively of 94 sec. and 113 sec. on a processor AMD Duron 750 MHz, using the Matlab Optimization Toolbox (the objective function is written in Matlab language).

6. Application To Parallel Manipulators : Kinematic Conditioning Optimization

Dexterity of a manipulator is a kinetostatic performance that can be measured from the condition number κ of its forward-kinematics Jacobian *J* [6]. In other words, if this Jacobian *J* is defined by :

$$\dot{x} = J\,\dot{q} \tag{5}$$

where $\dot{q}$ is the joint velocity vector and $\dot{x}$ the velocity vector of the end-effector Cartesian coordinates (position and orientation), this dexterity index is the ratio of the largest singular value of *J* to the smallest one.

It assumes that all entries of *J* have the same units. Otherwise, this dimensional inhomogeneity is resolved by introducing a normalizing *characteristic length*, as suggested in [6]. The latter is used to divide the positioning rows of *J*, making it dimensionnaly homogeneous. Let us note that its value itself comes from the minimization of the condition number over all the reachable configurations.

Coming back to our application, the goal is to optimize a global posture-independent performance index which is the mean of the inverses of κ over a volume *V* in the Cartesian space of the end-effector, also called Global Dexterity Index (GDI) [7] :

$$GDI = \frac{\int_V \frac{1}{\kappa}\,dV}{V} \tag{6}$$

In the case of positioning and orientating manipulators (for instance, the HexaSlide robot), the value of κ is obviously computed after normalizing the Jacobian matrix, as previously explained. By analogy with the optimization proposed in [6], the characteristic length becomes thus an additional parameter of our optimization problem.

6.1. Three dof Delta-type robot

Since the Delta robot (Fig. 5) is a positioning manipulator, there is no need of normalizing *J* to compute κ. The optimization parameters are the lengths (*la*, *lb*) of the legs (identical for the three legs), the characteristic radius of the base *Rb*, the characteristic radius of the mobile platform *Rp*, and finally the distance *z* between the base and the center of the desired workspace volume (here, a 2cm-sided cube).

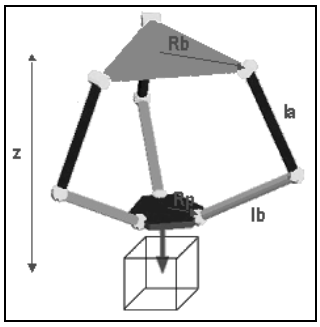

Fig. 5. Delta robot model

The restrictions of this optimization problem are obviously the 3D assembling constraints. Moreover, the legs lengths are limited to a maximum value of 20 cm (design constraints). The results of the optimization process are shown in the table below. Initial and optimum designs are compared in Fig. 6.

Table 2. Optimization results of the Delta manipulator

	Initial	Optimum
Average dexterity	36 %	95 %
la	10 cm	13.6 cm
lb	10 cm	20 cm
z	10 cm	13.3 cm
Rb	5 cm	6.26 cm
Rp	2 cm	3.29 cm

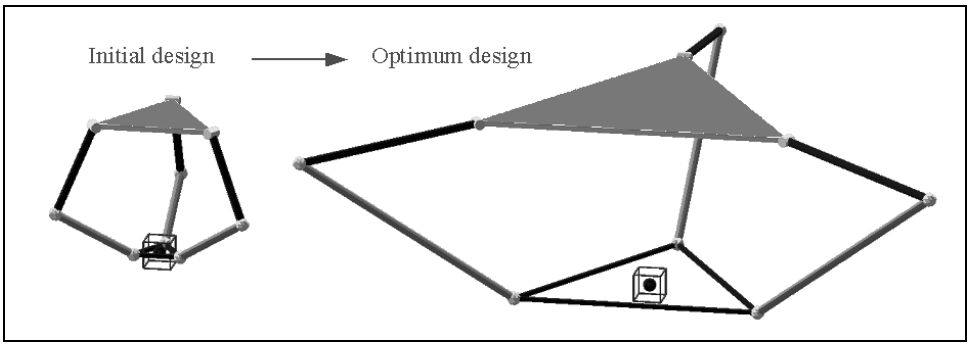

Fig. 6. Initial and optimum designs of the Delta robot optimization

That optimization result is obtained with 395 evaluations of the objective function which takes 40 sec. on a processor AMD Duron 750 MHz. Note that in this case, the function is written in C-mex langage and compiled using the "mex" Matlab compiler. In order to validate this result, other optimization processes are run, starting with random initial conditions : over 100 runs, this optimal result is found 77 times, which may reasonably convince us that this optimum is global.

6.2. Six dof HexaSlide robot

The HexaSlide manipulator (Fig. 7) has 3 positioning and 3 orientating degrees of freedom which requires to normalize J before computing κ each time the parameters change, i.e. at each call of the objective function. As described above, this involves an additional optimization parameter : the characteristic length.

The other parameters of this optimization problem (cfr Fig. 7) are the legs length l, the characteristic radius of the platform RB, the gap H, the horizontal and vertical slant angles α and β of the prismatic actuator rail axes, the separation angle y between ball-and-socket joints on the platform, and finally the distance Gz between the base and the center of the desired workspace volume (see [8] for more details). The actuator range $lmax$ and radius RA of the base are predetermined to sufficiently large values.

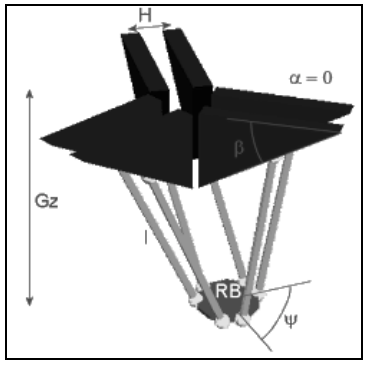

Fig. 7. Hexaslide robot model

An interest of that application is to compare the results of those obtained in a similar paper talking about architecture optimization for accuracy or the HexaSlide [8]. In that paper, the objective function is based on a global volumetric error analysis which is more detailed than here. But the search of the optimum seems too restrictive from our point of view since the workspace volume is fixed at one third, from the bottom, of the distance between the upper and the lower limits of the whole translational workspace. Here, that distance is part of the optimization parameters.

The results are shown in table 3 below and you can also compare initial and optimum design in Fig. 8.

Table 3. Optimization results of the HexaSlide manipulator

	Initial	Optimum
Average dexterity	0.1 %	50 %
Gz	100 cm	45.6 cm
l	0.9928 m	1.676 m
RB	16.5 cm	52.1 cm
α	0°	48°
β	30°	47°
H	22 cm	27.6 cm
ψ	83.6°	104.4°
L_c	1 mm	0.002 mm

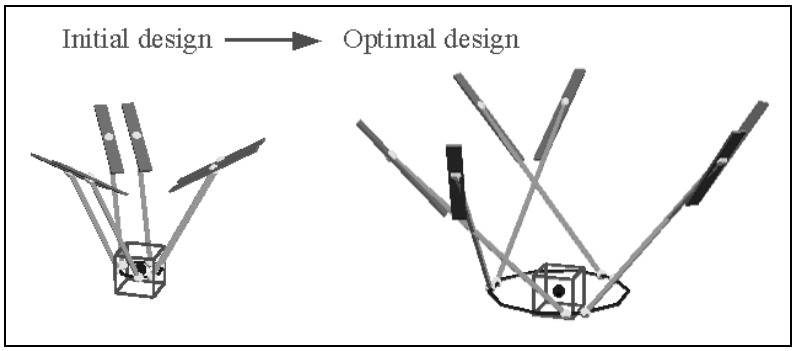

Fig. 8. Initial and optimum designs of the HexaSlide robot optimization

Let's remark that the optimization process doesn't take into account the problem of collisions since the legs of the optimum design are crossed. Actually, we haven't set any such constraint to let the maximum freedom to the optimizer.

7. Conclusion

In this paper, a penalization method has been developed to optimize the design of closed-loop mechanism. The issue of such problem lies in the assembling constraints and the way to solve them. So, we have shown how to exploit the conditioning of the constraints Jacobian to penalize the objective function. Finally, applications are proposed : first, an academic example of planar ejector to illustrate the method, and finally, 3D more realistic applications dealing with parallel robot dexterity.

8. Acknowledgments

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.

9. References

1. Haj-Fraj A. and Pfeiffer F. Optimization of Automatic Gearshifting. Vehicle System Dynamics Supplement, 2001, 35:207-222
2. Su Y.X., Duan B.Y., Zheng C.H. Genetic design of kinematically optimal fine tuning Stewart platform. Mechatronics, 2001, 11:821-835
3. Cabrera J.A., Simon A., Prado M. Optimal synthesis of mechanisms with genetic algorithms. Mechanism and Machine Theory, 2002, 37:1165-1177
4. Wehage R.-A. and Haug E.-J. Generalized coordinate partitioning for dimension reduction in analysis of constrained dynamic systems, Journal of Mechanical Design, 1982, 134:247-255
5. Collard J.-F., Fisette P., Duysinx P. Contribution to the Optimization of Closed-Loop Multibody Systems : Application to Parallel Manipulators. Euromech 442, 2003
6. Angeles J., Fundamentals of Robotic Mechanical Systems. Springer-Verlag, 1997
7. Gallant-Boudreau M., Boudreau R. 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
8. Ryu J., Cha J. Volumetric error analysis and architecture optimization for accuracy of HexaSlide type parallel manipulators. Mechanism and Machine Theory, 2003, 38:227-240