MASTER AUTOMATIQUE ET SYSTEME DE PRODUCTION

SPECIALITE ARTS

ANNÉE 2005 – 2006

rapport de Master ASP

Présenté par

Sébastien LENGAGNE
Le mardi 19 septembre 2006

TITRE

Optimisation de mouvement multi-contacts pour le robot HRP-2

JURY

Président :	Wisama KHALIL	Professeur
Examinateurs :	Abdelhamid CHRIETTE	Maître de conférences
	Yannick AOUSTIN	Maître de conférences
	Chrisitine CHEVALLEREAU	Chargé de recherche

Encadrants: Sylvain MIOSSEC (JRL, Japon) & Yannick AOUSTIN

Laboratoire : Joint Robotics Laboratory : AIST Tsukuba (Japon)

MASTER AUTOMATIQUE ET SYSTEME DE PRODUCTION

SPECIALITY ARTS

YEAR 2005 – 2006

Master ASP thesis

Presented by

Sébastien LENGAGNE

on Tuesday, september the 19 th 2006

TITLE

Multi-contact motion optimization for HRP-2 robot

Thesis supervisors: Sylvain MIOSSEC (JRL, Japan) & Yannick AOUSTIN

Laboratory : Joint Robotics Laboratory : AIST Tsukuba (Japan)

Contents

List of Figures

Acknowledgement

- This research is supported by grants from the ImmerSence EU CEC project, Contract No. 27141 (FET Presence) under the 6th Researh program.
- I would like to thank Yannick AOUSTIN who gave me the opportunity to do my training period about HRP-2 Robot in Tsukuba (Japan), though I was to do it at Nantes.
- Thanks Sylvain MIOSSEC, my supervisor, for offering me this training period and for all the talks we had, which allowed me to learn a lot about research methods and knowledge in robotics.
- I would like to thank also Mr. Abderrahmane KHEDDAR and Dr Kazuhito YOKOI for their cheerful greetings in the laboratory.
- Thanks all the trainees and the staff members of the laboratory.
- I would like to thank all the table tennis players in the AIST association for all the entertainment they gave me.

Introduction

For my master degree I did my training period in the Joint french-japanese Robotics Laboratory (JRL) , in the National Institute of Advanced Industrial Science and Technology (AIST) in Tsukuba (Japan). My work was about multi-contact motion optimization for the HRP-2 robot.

When I began my training period there was an optimization program which computes optimal one-contact motion. So it was possible to define some motions, as kicking motion. My work was to modify this program to optimize a multi-contact motion. With several contacts the robot will be able to do more different motions, as leaning on a table to catch objects or throwing small objects.

For a one-contact configuration, we can compute a unique value for the torques and for the forces applied by the robot to the environnement. But for a multi-contact configuration, the system has less degrees of freedom, and becomes over-actuated. Therefore the torques and contact forces are dependant, and one can express torques in function of contact forces. So the torques values are not unique. Therefore it is necessary to compute the best value of contact forces wich allow to minimize an objective function.

First we will present the AIST, the JRL and the HRP-2 robot. Then we will define the motion optimization problem, and how to compute the different values for a one-contact motion. Next we will present the different methods to get the contact forces, and to compute the torques values and gradient we used in the optimization. Finally we will show the computation time for those methods, and we will present how to optimize a throwing motion.

Chapter 1

AIST and HRP-2 presentation

1.1 AIST

Fifteen japanese research institutes were consolidated as of April 1, 2001 to found the National Institute of Advanced Industrial Science and Technology (AIST).

AIST has about 2,500 research scientists and conducts research and development that partly contribute to the japanese economic and industrial administration of the Ministry of Economy, Trade and Industry.

Though the scope of AIST includes the contributions to the administration, AIST is an independent research institute mainly funded by the government.

AIST consists of about 60 research institutes and centers, each of which is in charge of specific research missions.

1.2 JRL

The "Intelligent Systems Research Institute" (ISRI) of the AIST and "le Département des Sciences et Technologies de l'Information et de la Communication" (STIC) of "le Centre National de la Recherche Scientifique" (CNRS) have set up a joint research laboratory, named ISRI/AIST-STIC/CNRS Joint Japanese-French Robotics Laboratory (JRL). The AIST and the CNRS have concluded a comprehensive research collaboration agreement on November 22, 2001. The foundation of the JRL is a part of research efforts in specific areas under this agreement.

The JRL is operated in all aspects on the basis of bipolar management. The research base in Japan (JRL-Japan) is located in AIST Tsukuba Research Center, and the France side (JRL-France) is located in the Laboratory for Analysis and Architecture of Systems (LAAS)-CNRS in Toulouse.

Both research bases are operated by joint funding from the ISRI/AIST and the CNRS, and mixed Japanese-French teams work in cooperation at both bases.

In order to effectively integrate individual research efforts of Japanese and French scientists, a humanoid robot HRP-2, is used in both bases as a common experimental platform.

1.3 HRP-2 robot

HRP-2 is the final robotic platform for the Humanoid Robotics Project headed by the Manufacturing Science and Technology Center (MSTC), which are sponsored by the japanese Ministry of Economy, Trade and Industry (METI) through New Energy and Industrial Technology Development Organization (NEDO).

The total robotic system was designed and integrated by Kawada Industries, Inc. together with Humanoid Research Group of AIST.

HRP-2's height is 154 cm and weight is 58 kg including batteries. It has 30 degrees of freedom (DOF) including two DOF for its hip. The cantilevered crotch joint allows it to walk in a confined area. Its highly compact electrical system packaging allows it to forget the commonly used "backpack" used on other humanoid robots.

The external appearance of HRP-2 was designed by Mr. Yutaka Izubuchi, a mechanical animation designer famous for his robots that appear in Japanese comics. Mr. Izubuchi also named HRP-2 "Promet."

HRP-2 will be used for experiments to further develop robotic technologies in the areas of "walking on uneven surfaces," "tipping-over control," "getting up from a fallen position," and "human-interactive operations in open spaces." It will also be used for another 5-year "Key Technology Research Promotion Program," entitled "Key Technology Research and Development for Humanoid Robot Operating in Actual Environments." This project is also sponsored by METI and NEDO, spearheaded by Kawada Industries and supported by AIST and Kawasaki Heavy Industries, Inc.

Meanwhile, Kawada Industries will start renting HRP-2 as a humanoid robot R&D platform. Internal API for HRP-2 is expected to be available to the public and its users will be able to develop their own software. It is anticipated that HRP-2s will greatly enhance humanoid robot technology research activities.

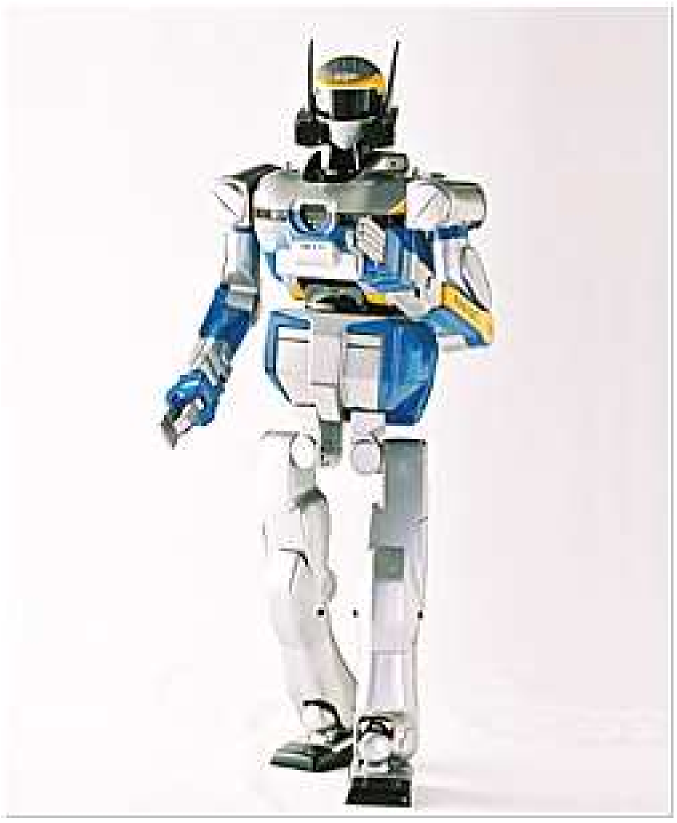


Figure 1.1: HRP-2 picture (from http://www.kawada.co.jp)

Chapter 2

Presentation of motion optimization problem

2.1 Introduction

The optimization problem is to minimize an objective function taking into account several constraints.

$$\min_{x \in \mathbb{R}^n} f_{(x)} \tag{2.1}$$

with :

$$g_{in(x)} \leq 0 \tag{2.2}$$

$$g_{eq(x)} = 0 \tag{2.3}$$

$$x_{min} \leq x \leq x_{max} \tag{2.4}$$

Where x is the input data vector. $f_{(x)}$ is the objective function and is defined according to the application. For example it can be the difference between a reference motion and the optimal robot motion, the time or the energy consumption (in our case). $g_{in(x)}, g_{eq(x)}$ are the inequality and equality constraints during the motion, they can inlcude the limits for x values.

2.2 Objective function

The objective function $f_{(x)}$ is to minimize the electrical energy consumption of the robot. This energy is expressed like this :

$$E = \int \sum P_{i(t)} dt \tag{2.5}$$

We discretize this energy :

$$E = \sum_{t=0}^{t_f} \sum_{i} P_{i(t)} \Delta t \tag{2.6}$$

Where $P_{i(t)}$ is the instantaneous power in joint i motor.

We will explain how to compute this electrical power. We consider the equivalent electric scheme as shown in figure (2.1).

From this scheme we can compute the electrical power in the motors and get the power assesment (2.2) :

$$P_{elec} = P_{mec} + P_f + P_R \tag{2.7}$$

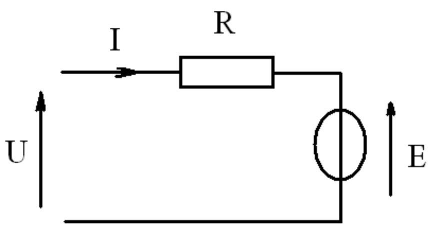


Figure 2.1: Electric scheme of motors

$$P_{elec} = \left(\Gamma + Frott_{(\Omega)}\right)\Omega + RI^2 \tag{2.8}$$

With $I = \frac{\Gamma_{em}}{K_{em}}$, $E = K_{em}\Omega$, K_{em} is the motor electromagnetic coefficient and $\Gamma_{em} = \Gamma + Frott_{(\Omega)}$ the electromagnetic torques.

$$P_{elec} = \Gamma_{em}\Omega + \frac{R\Gamma_{em}^2}{K_{em}^2} \tag{2.9}$$

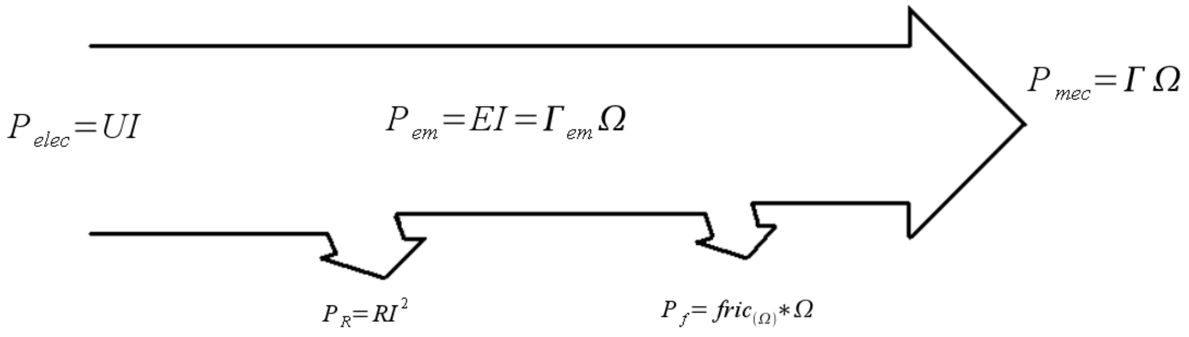


Figure 2.2: Power assesment

- P_{elec} : Electrical power (input of the motor).
- P_{em} : Eletro-magnetic power.
- P_{mec} : Mechanical power (output of the motor).
- P_R : Electrical power lost in Resistor by Joule effect.
- P_f : Mechanical power lost due to friction.

Due to the power supply system, the input electrical power P_{elec} cannot be negative. Therefore, we can only supply the motors and we cannot get power back. If one motor gets energy back, we cannot store it and it is lost. That is why we will implement this equation :

$$P = \max\left(P_{elec}, 0\right) = \max\left(\Gamma_{em}\Omega + \frac{R\Gamma_{em}^2}{K_e m^2}, 0\right) \tag{2.10}$$

2.3 Constraints

2.3.1 Joints limits

The joints limits are the minimal and maximal values for each angle value of the robot. These limits depend on the mechanical system of the robot.

$$q_i^L \leq q_i \leq q_i^U \tag{2.11}$$

2.3.2 Torques limits

The motors have several physical limits. These limits are determined by the supply system or the own caracteristic of the motor. The temperature in the motor is the main reason to put limits. The temperature is linked to the torques, we can choose to monitor the temperature, but, to make it easier, we can consider a constant limit for each torque.

$$\Gamma_i^L \leq \Gamma_i \leq \Gamma_i^U \tag{2.12}$$

In our case, we do not worry about torques limits in the optimization problem, we check the torques cannot damage the motors.

2.3.3 No sliding constraint

We want to control the robot during all the motion. Thus, we impose a no-sliding constraint for all the contact points. The no-sliding constraint is characterized by the friction cone :

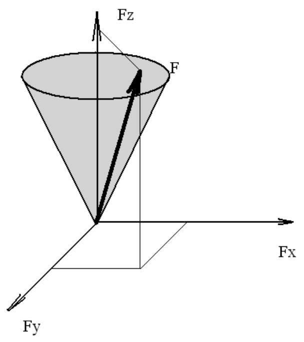


Figure 2.3: Friction cone.

Mathematicaly we can compute the friction cone constraint like this :

$$\sqrt{F_x^2 + F_y^2} \leq \sigma F_z \tag{2.13}$$

But the optimization program needs the derivate of this equation :

$$\left(\sqrt{u_{(x)}}\right)' = \frac{u'_{(x)}}{2\sqrt{u_{(x)}}} \tag{2.14}$$

To deal with numerical error when $u_{(x)} = F_x^2 + F_y^2 \approx 0$ we choose to transform this equation :

$$F_x^2 + F_y^2 \leq \sigma^2 F_z^2 \tag{2.15}$$

2.3.4 No turn over constraint

To ensure a good control of the robot we impose no take off of the contact surface and no turn over the edges of the surface contact. To ensure this, we compute the Zero Moment Point (ZMP) and the force F_R applied on ZMP that results of forces f and momenta m which are imposed at the origin of each contact surface. F_R must stay in the contact surface:

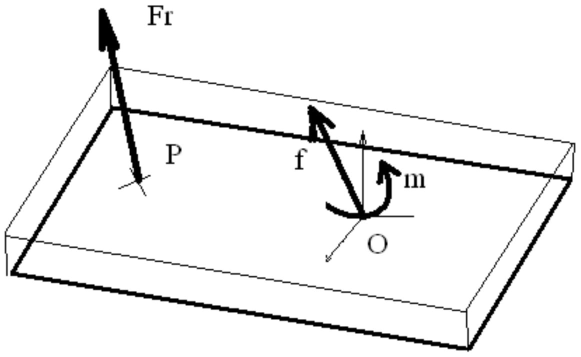


Figure 2.4: Zero Moment Point : ZMP.

We compute the position P_X, P_Y of F_R thanks to :

$$\vec{M_R} = \vec{m} - \vec{OP} \wedge \vec{f} = \vec{0} \tag{2.16}$$

with :

$$\vec{m} = \begin{bmatrix} M_x \\ M_y \\ M_z \end{bmatrix} ; \vec{OP} = \begin{bmatrix} P_x \\ P_y \\ 0 \end{bmatrix} ; \vec{f} = \begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} \tag{2.17}$$

Finally we get :

$$P_x = -\frac{M_y}{F_z} ; P_y = \frac{M_x}{F_z} \tag{2.18}$$

If we assume a rectangular contact surface, we have this inequality:

$$x_{min} \leq -\frac{M_y}{F_z} \leq x_{max} \tag{2.19}$$

$$y_{min} \leq \frac{M_y}{F_z} \leq y_{max} \tag{2.20}$$

But to have linear constraints we consider these equations, and we consider $F_z \geq 0$ to get an unilateral contact.

$$\begin{array}{c} x_{min}F_z + M_y \leq 0 \\ x_{max}F_z + M_y \geq 0 \\ y_{min}F_z - M_x \leq 0 \\ y_{max}F_z - M_x \geq 0 \end{array} \tag{2.21}$$

2.3.5 Bodies constraint

To optimize a motion, we impose some conditions on the position and orientation of several bodies. These conditions can be inequality or equality constraints for the whole motion or only one instant. These constraints are not the topic of the training period, to learn more about it one can refer to [1].

2.4 Problem parameters

The motion optimization is done by computing each joint value thanks to 9 splines parameters p_i. The robot has 30 joints, so we get an 270-parameters-optimization problem. We compute the joint value thanks to these splines parameters :

$$q_{(t)} = \sum_{i=1}^{9} p_i * b_{i(t)} \tag{2.22}$$

the spline functions b_i have the shape shown down :

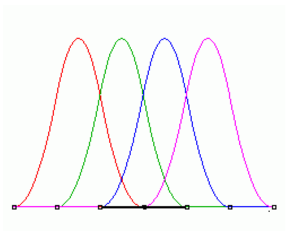


Figure 2.5: Bsplines.

Those functions have several caracteristics

- The initial and final value, velocity and acceleration of each spline are equal to zero (except the two first and the two last).
- At any moment joint values are expressed with only four splines parameters.
- The first spline starts with value, velocity and acceleration different from zero.
- The second one starts with velocity and acceleration different from zero, but the initial value is equal to zero.
- The third one starts with an acceleration different from zero but the initial value and velocity are null.
- Those conditions also are true for final values of the three last splines.

We consider that we start a motion with an initial and a final velocity and acceleration for all joints equal to zero. Thus, it produces a relation between the three first parameters and the initial value and between the three last parameters and the final value. So, we can reduce the number of optimization parameters, for each joint, from 9 to 5 (3 splines parameters and initial and final value). As such we get only 150 parameters for the joint values and if we choose to optimize the motion duration we can add the time as a parameter.

Chapter 3

One contact motion optimization

3.1 Introduction

We consider that the contact body can be modelized by plane contact. We define two contact kinds:

- A bilateral contact is defined as a mechanical link between two bodies which cannot be broken.
- A unilateral contact is defined as a link between two bodies which can be broken at any moment.

We suppose that the contact is a bilateral one, nevertheless we impose constraints on contact forces to get unilateral one.

We have to choose which value we will optimize: the joint or the torque values. In the optimization we will need those two values.

- If we choose to optimize the joint values we need to compute the inverse dynamics model to compute the torques.
- If we choose to optimize the torques values we need to compute the direct dynamics model to compute the joints.

[3] shows the inverse dynamics model is faster to compute than the direct dynamics model because we need only 2 recursions (3 for direct dynamics model).

Therefore we choose to do an optimization on joints value.

3.2 Dynamic equation

3.2.1 Dynamic model

We compute the dynamic model of the robot through the following equation:

$$M_{(q)}\ddot{X} + C_{(q,\dot{q})} + G_{(q)} = \begin{bmatrix} \Gamma_{0(nb\_j\times 1)} \\ 0_{(6\times 1)} \end{bmatrix} + \begin{bmatrix} 0_{(nb\_j\times 6)} \\ I_{(6\times 6)} \end{bmatrix} \begin{bmatrix} f_{ext} \\ t_{ext} \end{bmatrix} \tag{3.1}$$

Where $X = [q, x, y, z, \theta_x, \theta_y, \theta_z]^T$

To compute the torques Γ_0, the force f_{ext} and the momenta t_{ext}, we use the Newton-Euler algorithm in absolute frame. This algorithm requires two recursions [4].

- the first recursion computes the position, velocity and acceleration of each robot body.

- the second recursion computes the forces and torques for each joint.

We only present the equations we use in the next part of this report. To know more one can look at [4].

3.2.2 First recursion

The computation starts from the reference body of which we know the state (value, gradient and hessian for position and orientation) to compute the state of the waist(1) . And starting from the waist, we compute the state of all bodies (2,3,4,5) as shown in 3.1:

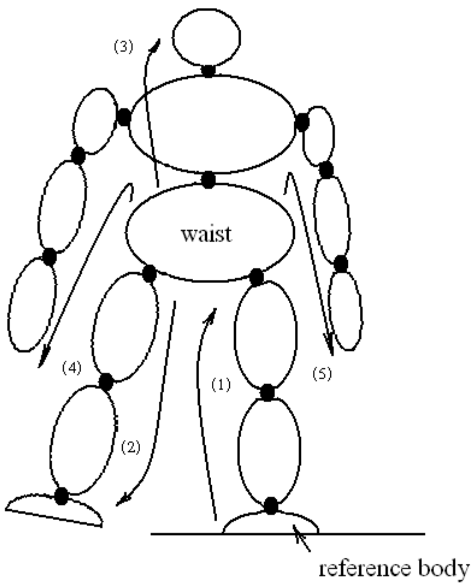


Figure 3.1: First recursion with left foot as reference body

We use equation of [5] to compute these values in an absolute frame. The mechanical system is known therefore we can express the orientation R_{0j} and the position P_{0j} of the link j, expressed in the frame $a_{(j)}$ through the position of the motor q_j. ($a_{(j)}$ is the link antecedent of j).

So we can compute the bodies position P_j and orientation R_j of joint j.

$$R_j = R_{a_{(j)}} R_{0j} \tag{3.2}$$

$$P_j = R_{a_{(j)}} P_{0j} + P_{a_{(j)}} \tag{3.3}$$

Then we compute sw_j and sv_j for each joint.

$$sw_j = R_{a_{(j)}} axe_j \tag{3.4}$$

$$sv_j = P_{(j)} \wedge sw_{(j)} \tag{3.5}$$

- sw_j is the axis orientation of joint j:
- sv_j is the direction that is perpendicular to axis orientation and to vector $\vec{OP_j}$ (O: origin of absolute frame ; P_j: position of joint j in absolute frame)

Then we compute the velocity and acceleration of the position and orientation. These values are not used later in this report, but one can look at [5].

3.2.3 Second recursion

The computation starts from the end of the limbs (except for the limb with the reference body). Taking into account all the torques and all the external forces on bodies, we spread the torques and forces applied by one link to the previous one until the waist(1,2,3,4). Then starting from the waist, we compute all the torques and the forces until the reference body (5). Therefore we get all the joints torque value and the force and momenta applied by the robot to the environnement (here the ground) as shown in 3.2.

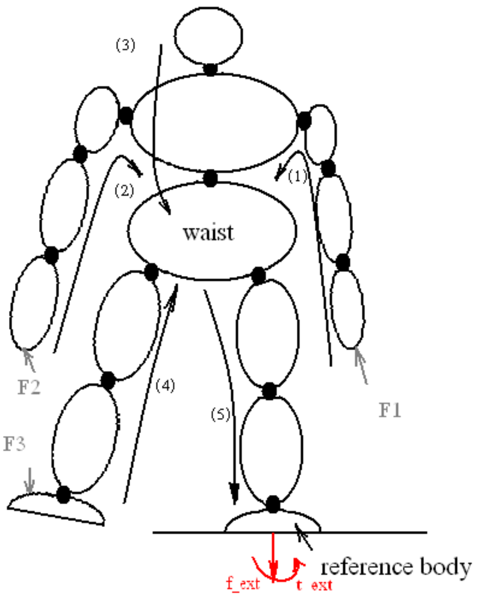


Figure 3.2: Second recursion with left foot as reference body

The computed joint torques take into account the friction, and we obtain in fact the electro-magnetical torques.

$$\Gamma_j = sv_{(j)}F_j + sw_{(j)}M_j - \Gamma_{d_j} + \mu_{v_j}\dot{q}_j \tag{3.6}$$

With :

- Γ_{d_j} dry friction torque of link j.
- μ_{v_j} coefficient of viscous friction of link j.
- F_j resulting force applied on link j.
- M_j resulting momenta applied on link j.

The resulting force and momenta depend on robot dynamic, gravity effect, external forces and momentas [6].

3.2.4 Computation of dry friction

The dry friction effect produces a constant torque wich is opposed to the velocity of the joint.

$$\Gamma_{d(\dot{q})} = -\Gamma_d * sign(\dot{q}) \tag{3.7}$$

This function is not smooth, therefore it can produce problem during the optimization. We approxime the dry friction effect with this equation:

$$\Gamma_{d(\dot{q})} = -\Gamma_d * \frac{2}{\Pi} \arctan(\nu\dot{q}) \tag{3.8}$$

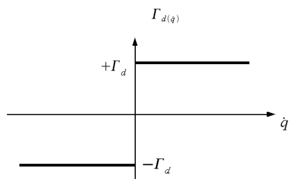


Figure 3.3: Exact dry friction

Where ν is a parameter. If ν is small the computed torque is closed to the real torque.

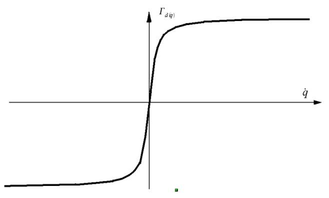


Figure 3.4: Computed dry friction

3.3 Gradient dynamic equation

The optimization needs the torques gradient. In this part we only present the equation we need later in this report. To know more one can refer [4]. The way to compute the gradient is the same as to compute the torques value. Therefore there are two recursions.

3.3.1 First recursion

To compute the gradient of position and orientation value, velocity and acceleration we use the same algorithm but we add several equations :

$$\frac{\partial R_j}{\partial p} = \frac{\partial R_{a_{(j)}}}{\partial p} R_{0j} + R_{a_{(j)}} \frac{\partial R_{0j}}{\partial p} \tag{3.9}$$

$$\frac{\partial P_j}{\partial p} = \frac{\partial R_{a_{(j)}}}{\partial p} P_{0j} + \frac{\partial P_{a_{(j)}}}{\partial p} \tag{3.10}$$

Then we compute the gradient of sw_j and sv_j of each joint :

$$\frac{\partial sw_j}{\partial p} = \frac{\partial R_{a_{(j)}}}{\partial p} axe_j \tag{3.11}$$

$$\frac{\partial sv_j}{\partial p} = \frac{\partial P_{(j)}}{\partial p} \wedge sw_{(j)} + P_{(j)} \wedge \frac{\partial sw_{(j)}}{\partial p} \tag{3.12}$$

3.3.2 Second recursion

To compute the electro-magnetic torques gradient we add those equations to the second recursion:

$$\frac{\partial \Gamma_j}{\partial p} = \frac{\partial sv_{(j)}}{\partial p} F_j + sv_{(j)} \frac{\partial F_j}{\partial p} + \frac{\partial sw_{(j)}}{\partial p} M_j + sw_{(j)} \frac{\partial M_j}{\partial p} + \mu_v \frac{\partial \dot q_j}{\partial p} \tag{3.13}$$

3.4 Optimization algorithm

3.4.1 Optimization functions

To optimize the motion we use an optimization software : IPOPT (Interior Point Optimizer) [2]. IPOPT is available in FORTRAN and C++ version (we use C++ version). It is necessary to define several functions to use IPOPT

- get_nlp_info : define the size of the problem (number variable, constraint)
- get_bounds_info : define the limits of variables and constraint.
- get_starting_point : define an initial value for variables or constraint multiplicator.
- eval_f : return the value of the criteria
- eval_grad_f : return the vector of the criteria gradient
- eval_g : return the values of the constraints
- eval_jac_g : return the structure and the value of the constraints'gradient.
- eval_h : return the value of the hessian (we can ask IPOPT to compute it by it-self)
- finalize_solution : get the value of the optimized variable with the value of constraints and Lagrangian multiplicator.

3.4.2 Algorithm

The figure (3.5) shows the algorithm to find the optimal solution. It is not realy IPOPT algorithm but it helps to understand the working of the computation.

3.5 Configuration optimization

3.5.1 Usefulness

To learn more about C++ and to understand the existing program, I started my training period by implementing these configuration optimizations. This optimization computes the optimal joints value in static case. Since the motion starts and finishes with joints velocity and acceleration equal to zero, we can initialize the motion optimization with these configurations to increase the motion optimization results quality.

3.5.2 Computation simplification

To compute the dynamic equation in static case, we do not worry about $\dot{q}$ and $\ddot{q}$ values. We consider that dry friction effect is opposed to the gravity effects. In static case, dry friction helps to minimize the energy consumption.

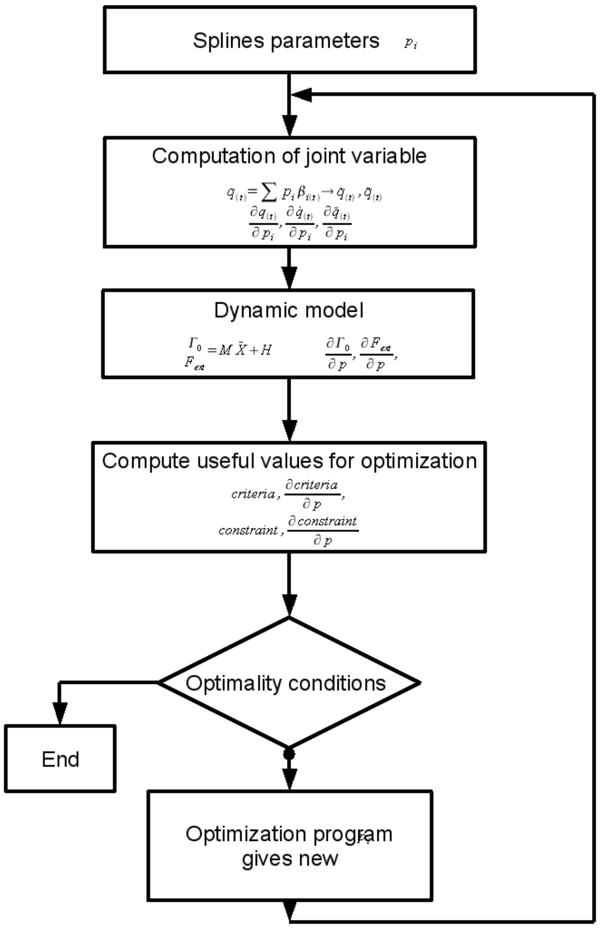

Splines parameters p_i
Computation of joint variable
$q_{(t)}=\sum p_i\beta_{i(t)}\rightarrow \dot{q}_{(t)}, \ddot{q}_{(t)}$
$\frac{\partial q_{(t)}}{\partial p_i}, \frac{\partial \dot{q}_{(t)}}{\partial p_i}, \frac{\partial \ddot{q}_{(t)}}{\partial p_i}$
Dynamic model
$\Gamma_0 = M\ddot{X}+H$
F_{ex}
$\frac{\partial \Gamma_0}{\partial p}, \frac{\partial F_{ex}}{\partial p},$
Compute useful values for optimization
$criteria, \frac{\partial criteria}{\partial p},$
$constraint, \frac{\partial constraint}{\partial p}$
Optimality conditions
End
Optimization program gives new p_i

Figure 3.5: Algorithm for one contact motion optimization

	final criteria	total computing time (sec)
with configuration optimization	$14,80$	169
without configuration optimization	$16,35$	126

Table 3.1: Optimization time with and without configuration optimization

3.5.3 Results

We present the computation time and final criteria value for the same motion optimization, with dry friction parameter $\nu = 1$

The motion optimization do 2000 iterations and returns the final criteria value. The configuration optimization stops before the 2000 iterations.

We can see that the final criteria is better with configuration optimization than without, even if the computing time is bigger. That means we get a better local minimum thanks to a better initialization of the motion optimization.

Chapter 4

Multi-contact motion optimization

4.1 Contact definition

To define a contact we need to do the equality between two frames:

- The frame on the body of the robot. The origin of this frame is in the contact surface.
- The frame on the environnement.

That is why we add position and orientation constraint for contact body frame.

4.2 Difference between one contact optimization

With several contact points, the contact forces are not unique, therefore there is not only one value for the torques. We have to compute the best forces for all the contact points to minimize the objective function .

4.2.1 Dynamic model

We remember the dynamic equation for one contact motion :

$$M_{(q)}\ddot{X} + C_{(q,\dot{q})} + G_{(q)} = \begin{bmatrix} \Gamma_{0(nb\_j,1)} \\ 0_{(6,1)} \end{bmatrix} + \begin{bmatrix} 0_{(nb\_j,6)} \\ I_{(6,6)} \end{bmatrix} \begin{bmatrix} f_{ext} \\ t_{ext} \end{bmatrix} \tag{4.1}$$

Where $X = [q, x, y, z, \theta_x, \theta_y, \theta_z]^T$

We can compute the torques through the dynamic equation of a *nb_c*-contacts motion :

$$M_{(q)}\ddot{X} + C_{(q,\dot{q})} + G_{(q)} = \begin{bmatrix} \Gamma_{(nb\_j,1)} \\ 0_{(6,1)} \end{bmatrix} + \begin{bmatrix} J_{1(nb\_j,6\times nb\_c)} \\ J_{2(6,6\times nb\_c)} \end{bmatrix} \begin{bmatrix} f_{c_1} \\ t_{c_1} \\ \dots \\ f_{c_{nb\_c}} \\ t_{c_{nb\_c}} \end{bmatrix} \tag{4.2}$$

note

Usually we use the jacobian transposed J^T, but to simplify the notation we consider an other convention, so we note J

4.2.2 Computation of J_1

The matrix J_1 allows to compute the effects of the contact forces on the torques. The equation (3.6) shows that this effect is due to sv_j and sw_j values. Therefore we just need to put these values for the joint which are between the contact body and the reference body. We take the example of a five joints robot :

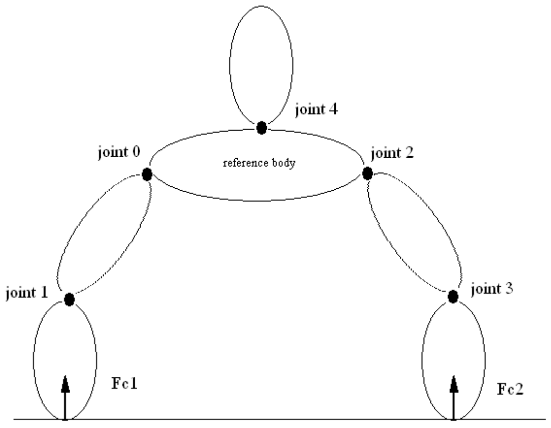


Figure 4.1: Example for J1 computation

$$J_1^T = \begin{bmatrix} sv_{0(3\times1)} & sv_{1(3\times1)} & 0 & 0 & 0 \\ sw_{0(3\times1)} & sw_{1(3\times1)} & 0 & 0 & 0 \\ 0 & 0 & sv_{2(3\times1)} & sv_{3(3\times1)} & 0 \\ 0 & 0 & sw_{2(3\times1)} & sw_{3(3\times1)} & 0 \end{bmatrix} \tag{4.3}$$

note

If sv_i and sw_i are computed when we start from the reference body to the waist, it is necessary to replace them by $-sv_i$ and $-sw_i$.

4.2.3 Computation of J_2

J_2 is used to compute the sum of all the contact forces in the absolute frame, therefore it contains only transformation matrix between contct forces frame and absolute frame.

$$J_2 = \begin{bmatrix} {}^{abs}T_{c_1} & {}^{abs}T_{c_2} & \dots & {}^{abs}T_{c_{nb\_c}} \end{bmatrix} \tag{4.4}$$

Where :

$$ {}^{abs}T_{c_i} = \begin{bmatrix} R_{c_i} & 0 \\ [P_{c_i}]R_{c_i} & R_{c_i} \end{bmatrix} \tag{4.5}$$

with P_{c_i} and R_{c_i} the position and orientation of the contact point in the absolute frame :

$$[P] = \begin{bmatrix} 0 & -P_z & P_y \\ P_z & 0 & -P_x \\ -P_y & P_x & 0 \end{bmatrix} \tag{4.6}$$

4.3 Contact forces optimization

4.3.1 Choices for optimization

To include contact forces we need to find the forces that minimize the criteria. To do it, we have the choice between

- Computing forces with splines parameters and add this parameters to the motion optimization.
- Optimizing the contact forces at each iteration of the motion optimization.

The first method supposes that the forces have spline shape, so the solution will be less optimal, but the computation time should be closed to one-contact motion optimization. The second one computes the exact value of optimal contact forces, but it should be slower because each time we need to know the torques we have to do a local optimization. Therefore we will compare this two methods

4.3.2 Including contact forces

So with those two methods, we have to compute the torques value and gradient for multi-contact motion.From equations (4.1) and (4.2) we get :

$$\Gamma = \Gamma_0 - J_1 F_c \tag{4.7}$$

With $F_c = \left[f_{c_1}, t_{c_1}, ..., f_{c_{nb\_c}}, t_{c_{nb\_c}}\right]^T$, f_{c_i}: force on contact i, t_{c_i}: momenta on contact i.
By derivate this equation, we find the torques gradient :

$$\frac{\partial \Gamma}{\partial p} = \frac{\partial \Gamma_0}{\partial p} - \frac{\partial F_1}{\partial p} F_c - J_1 \frac{\partial F_c}{\partial p} \tag{4.8}$$

4.3.3 Contact forces as splines

Computation of contact forces

With (4.1) and (4.2) we find the contact forces must verify this equation :

$$F_{ext} = J_2 F_c = \left[J_{2,1} ... J_{2,nb\_c}\right] \begin{bmatrix} F_{c_1} \\ \vdots \\ F_{c_{nb\_c}} \end{bmatrix} \tag{4.9}$$

To verify this equation we compute one contact force (6 values) thanks to F_{ext} and the other contact forces as :

$$F_{c_{nb\_c}} = (J_{2,nb\_c})^{-1} \left(F_{ext} - \sum_{i=0}^{nb\_c-1} J_{2,i} F_{c_i} \right) \tag{4.10}$$

For example, in a two-contacts motion (for example : 4.1), we consider only one contact force as splines and we compute the other.

In this case we compute one contact force F_c by using B-splines parameters with the same way to joint value (2.22)

$$F_{c_1(t)} = \sum_{i=1}^{9} p_i * b_{i(t)} \tag{4.11}$$

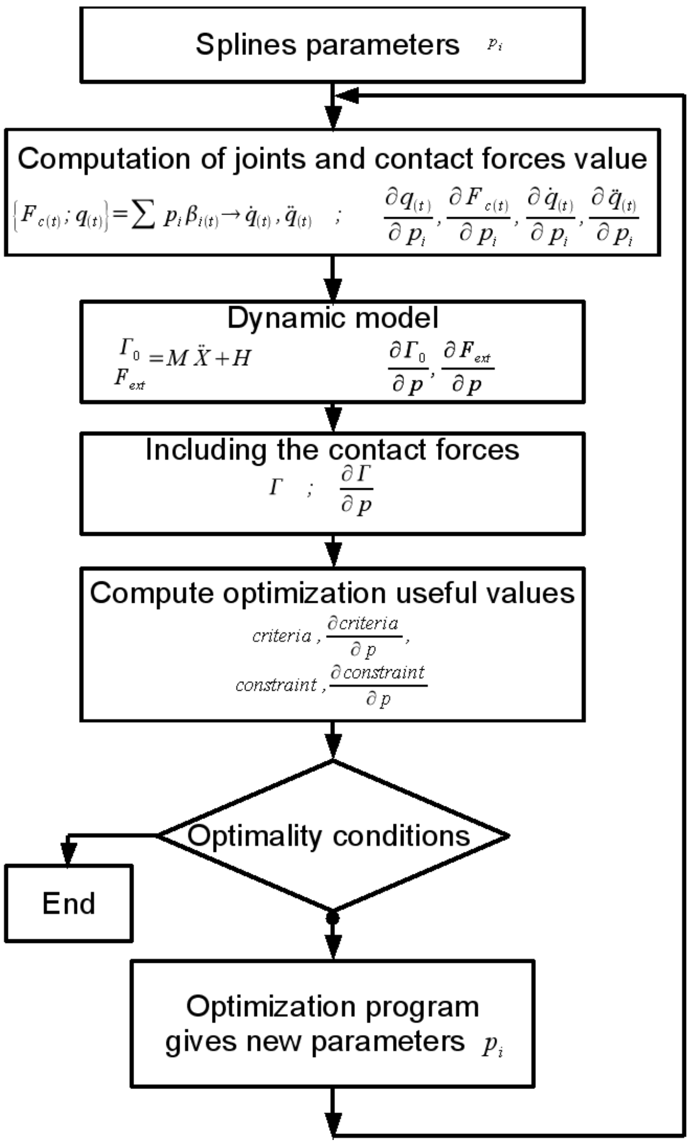
And to compute F_{c_2} we use equation (4.10)

Torques value and gradient

So, with splines parameters we compute directly the gradient of F_c : $\frac{\partial F_c}{\partial p} = \sum_{i=1}^{9} b_{i(t)}$ except for the last contact. To get the gradient of the last contact forces we derivate equation(4.10):

$$\frac{\partial F_{c_{nb\_c}}}{\partial p} = (J_{2,nb\_c})^{-1} \left(\frac{\partial F_{ext}}{\partial p} - \sum_{i=0}^{nb\_c-1} J_{2,i} \frac{\partial F_{c_i}}{\partial p} \right) \tag{4.12}$$

Algorithm



Splines parameters p_i
Computation of joints and contact forces value
$\{F_{c(t)}; q_{(t)}\} = \sum p_i \beta_{i(t)} \rightarrow \dot{q}_{(t)}, \ddot{q}_{(t)}$; $\frac{\partial q_{(t)}}{\partial p_i}, \frac{\partial F_{c(t)}}{\partial p_i}, \frac{\partial \dot{q}_{(t)}}{\partial p_i}, \frac{\partial \ddot{q}_{(t)}}{\partial p_i}$
Dynamic model
$\Gamma_0 = M\ddot{X} + H$
F_{ext}
$\frac{\partial \Gamma_0}{\partial p}, \frac{\partial F_{ext}}{\partial p}$
Including the contact forces
Γ ; $\frac{\partial \Gamma}{\partial p}$
Compute optimization useful values
$criteria, \frac{\partial criteria}{\partial p},$
$constraint, \frac{\partial constraint}{\partial p}$
Optimality conditions
End
Optimization program gives new parameters p_i

Figure 4.2: Motion optimization algorithm with contact forces compute with splines

4.3.4 Local optimization for contact forces

Now we will explain the program with an internal optimization for contact forces, for this local optimization we need to compute:

- the criteria.

- the gradient of the criteria with respect to the contact forces.
- the constraints.
- the gradient of the constraints with respect to the contact forces.

Criteria

The criteria is the same as in motion optimization, but we do not consider the friction because friction depends on $\dot{q}$ and not on the torques and so not on the contact forces. So friction adds constant value that cannot be minimized. To simplify the notations we suppose the motors parameters $\frac{R_i}{K_i^2} = 1$. Therefore the criteria is:

$$f_{(F_c)} = \Gamma^T\Gamma + \Gamma^T\dot{q} \tag{4.13}$$

$$f_{(F_c)} = f = (\Gamma_0 - J_1F_c)^T(\Gamma_0 - J_1F_c) + (\Gamma_0 - J_1F_c)^T\dot{q} \tag{4.14}$$

Gradient of criteria

We compute the gradient of the criteria ∇f :

$$\nabla f = \frac{\partial\left(\Gamma^T\Gamma\right)}{\partial F_c} + \frac{\partial\Gamma^T\dot{q}}{\partial F_c} \tag{4.15}$$

We impose that ∇f is a vertical vector with $6 \times nb\_c$ values, so :

$$\nabla f = 2\Gamma\frac{\partial\Gamma^T}{\partial F_c} + \frac{\partial\Gamma^T}{\partial F_c}\dot{q} \tag{4.16}$$

$$\nabla f = 2(\Gamma_0 - J_1F_c)\left(-J_1^T\right) - J_1^T\dot{q} \tag{4.17}$$

$$\nabla f = 2(-\Gamma_0J_1^T + J_1F_cJ_1^T) - J_1^T\dot{q} \tag{4.18}$$

Constraints

For this optimization we have only three kinds of constraint:

- from (4.1) and (4.2) we get a dynamics equation equality constraint : $F_{ext} = J_2F_c$
- friction inequality constraint : $f_x^2 + f_y^2 \leq \sigma^2 f_z^2$
- ZMP inequality constraint : $x_{min} \leq -\frac{t_y}{f_z} \leq x_{max}$ and $y_{min} \leq \frac{t_x}{f_z} \leq y_{max}$

To impose an unilateral contact we add a constraint : $F_z \geq 0$

The constraint vector contains only value of active constraints, but we show it when all constraints are active:

$$g_{(F_c)} = g = \begin{bmatrix} (J_2F_c - F_{ext} = 0)_{(6,1)} \\ \left(f_{x_i}{}^2 + f_{y_i}{}^2 - \sigma_i^2 f_{z_i}{}^2 \leq 0\right)_{(nb\_c,1)} \\ \begin{pmatrix} x_{min}f_z + t_y \leq 0 \\ x_{max}f_z + t_y \geq 0 \\ y_{min}f_z - t_x \leq 0 \\ y_{max}f_z - t_x \geq 0 \end{pmatrix}_{(4\times nb\_c,1)} \end{bmatrix} \tag{4.19}$$

Therefore the constraint vector size is $6 + 5 \times nb\_c$.

Gradients Constraints

We compute ∇g,the gradient of the active constraints as a matrix of which the size is $(6+5\times nb\_c, 6\times nb\_c)$ values.

$$\nabla g = \begin{bmatrix} (J_2)_{(6,6\times nb\_c)} \\ \begin{pmatrix} 2f_{x_i} & 2f_{y_i} & -2\sigma_i^2 f_{z_i} & 0 & 0 & 0 \end{pmatrix}_{(nb\_c,6\times nb\_c)} \\ \begin{pmatrix} 0 & 0 & x_{min} & 0 & 1 & 0 \\ 0 & 0 & x_{max} & 0 & 1 & 0 \\ 0 & 0 & y_{min} & -1 & 0 & 0 \\ 0 & 0 & y_{max} & -1 & 0 & 0 \end{pmatrix}_{(4\times nb\_c,6\times nb\_c)} \end{bmatrix} \tag{4.20}$$

Optimality conditions

If the local optimization converges, it returns contact forces F_c wich satisfy the KKT conditions [2]:

$$\nabla f_{(F_c)} + \nabla g_{(F_c)}\lambda - z = 0 \tag{4.21}$$

$$g_{(F_c)} = 0 \tag{4.22}$$

g and ∇g contain the value of the active constraints. In our case, there is no limit for the input values F_c, therefore the optimization program will return the input Lagrangian multiplicator $z \approx 0$, that is why we can simplify this equation:

$$\nabla f_{(F_c)} + \nabla g_{(F_c)}\lambda = 0 \tag{4.23}$$

$$g_{(F_c)} = 0 \tag{4.24}$$

Therefore the local optimization returns the contact forces F_c but we need to compute the gradient of contact forces with respect to the motion optimization parameters $\frac{\partial F_c}{\partial p}$.

Derivative of optimality condition

We derivate optimality condition equation to find :

$$\frac{\partial \nabla f}{\partial p} + \frac{\partial(\nabla g\lambda)}{\partial p} = 0 \tag{4.25}$$

$$\frac{\partial g}{\partial p} = 0 \tag{4.26}$$

Computation of $\frac{\partial \nabla f}{\partial p}$

$$\nabla f = 2(-\Gamma_0 J_1^T + J_1 F_c J_1^T) - J_1^T \dot{q} \tag{4.27}$$

$$\frac{\partial \nabla f}{\partial p} = 2\left(-\frac{\partial(\Gamma_0 J_1^T)}{\partial p} + \frac{\partial(J_1 F_c J_1^T)}{\partial p}\right) - \frac{\partial(\dot{qJ_1^T})}{\partial p} \tag{4.28}$$

We want $\frac{\partial \nabla f}{\partial p}$ to have a $(6\times nb\_c, nb\_p)$ size. To get the equation easier we will compute the gradient for only one parameter p_i.

$$\frac{\partial \nabla f}{\partial p_i} = -2\frac{\partial \Gamma_0}{\partial p_i}J_1^T - 2\Gamma_0\frac{\partial J_1^T}{\partial p_i} + 2\frac{\partial J_1}{\partial p_i}F_c J_1^T + 2J_1 F_c\frac{\partial J_1^T}{\partial p_i} + 2J_1\frac{\partial F_c}{\partial p_i}J_1^T - \frac{\partial J_1^T}{\partial p_i}\dot{q} - J_1^T\frac{\partial \dot{q}}{\partial p_i} \tag{4.29}$$

We know from the one contact dynamic model :

- Γ_0 : vector with size :(nb_joints) .
- $\frac{\partial \Gamma_0}{\partial p_i}$: matrix with size $(nb\_joints, 1)$.
- $\dot{q}$: vector with size (nb_joints).
- $\frac{\partial \dot{q}}{\partial p_i}$: matrix with size $(nb\_joints, 1)$.

We know from the contact optimization :

- J_1 : matrix with size: $(6 \times nb\_c, nb\_joints)$.
- F_c : vector with size: $(6 \times nb\_c)$.

We can easily compute $\frac{\partial J_1}{\partial p_i}$: matrix $(6 \times nb\_c, nb\_joints)$, by replacing sv_j and sw_j by $\frac{\partial sv_j}{\partial p_i}$ and $\frac{\partial sw_j}{\partial p_i}$. If we consider example of fig (4.1) we compute :

$$\frac{\partial J_1^T}{\partial p_i} = \begin{bmatrix} \frac{\partial sv_0}{\partial p_i} & \frac{\partial sv_1}{\partial p_i} & 0 & 0 & 0 \\ \frac{\partial sw_0}{\partial p_i} & \frac{\partial sw_1}{\partial p_i} & 0 & 0 & 0 \\ 0 & 0 & \frac{\partial sv_2}{\partial p_i} & \frac{\partial sv_3}{\partial p_i} & 0 \\ 0 & 0 & \frac{\partial sw_2}{\partial p_i} & \frac{\partial sw_3}{\partial p_i} & 0 \end{bmatrix} \tag{4.30}$$

There is only one unknown value $\left(\frac{\partial F_c}{\partial p}\right)$, therefore we can get an easier shape of equation(4.29):

$$\frac{\partial \nabla f}{\partial p_i} = A_{i\nabla f} + B_{i\nabla f} \frac{\partial F_c}{\partial p_i} \tag{4.31}$$

Computation of $\frac{\partial (\nabla g \lambda)}{\partial p}$

The program has to take into account the active constraints. Here we assume that all constraints are active and we show how to compute the different values :

$$\nabla g = \begin{bmatrix} (J_2)_{(6, 6\times nb\_c)} \\ \begin{pmatrix} 2f_{x_i} & 2f_{y_i} & -2\sigma_i^2 f_{z_i} & 0 & 0 & 0 \end{pmatrix}_{(nb\_c, 6\times nb\_c)} \\ \begin{pmatrix} 0 & 0 & x_{min} & 0 & 1 & 0 \\ 0 & 0 & x_{max} & 0 & 1 & 0 \\ 0 & 0 & y_{min} & -1 & 0 & 0 \\ 0 & 0 & y_{max} & -1 & 0 & 0 \end{pmatrix}_{(4\times nb\_c, 6\times nb\_c)} \end{bmatrix} \tag{4.32}$$

The equation is :

$$\frac{\partial (\nabla g \lambda)}{\partial p} = \frac{\partial (\nabla g)}{\partial p}\lambda + \nabla g \frac{\partial (\lambda)}{\partial p} \tag{4.33}$$

λ is given by the optimization program, and we want to know : $\frac{\partial (\lambda)}{\partial p}$. We consider only one derivative parameter p_i.

$$\frac{\partial (\nabla g)}{\partial p_i}\lambda = \begin{bmatrix} \left(\frac{\partial J_2}{\partial p_i}\right)_{(6, 6\times nb\_c)} \\ \begin{pmatrix} 2\frac{\partial f_{x_i}}{\partial p_i} & 2\frac{\partial f_{y_i}}{\partial p_i} & -2\sigma_i^2 \frac{\partial f_{z_i}}{\partial p_i} & 0 & 0 & 0 \end{pmatrix}_{(nb\_c, 6\times nb\_c)} \\ (0)_{(4\times nb\_c, 6\times nb\_c)} \end{bmatrix} [\lambda]_{(6+5\times nb\_c)} \tag{4.34}$$

We note that J_2 depends only on the contact position, wich one are constant, therefore $\frac{\partial J_2}{\partial p} = 0$

We can see $\frac{\partial(\nabla g)}{\partial p_i}\lambda$ is different from zero only for friction constraint, we modify this equation to obtain :

$$\left(\frac{\partial(\nabla g)}{\partial p_i}\lambda\right)_{fric} = B_{fric(nb\_c,nb\_p_i)}\frac{\partial F_c}{\partial p_i} \tag{4.35}$$

with :

$$B_{fric} = \begin{bmatrix} 2\lambda_{fric1} & 2\lambda_{fric1} & -2\sigma_1^2\lambda_{fric1} & \dots & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & \dots & 2\lambda_{fric2} & 2\lambda_{fric2} & -2\sigma_2^2\lambda_{fric2} & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \end{bmatrix} \tag{4.36}$$

Finally we get :

$$\frac{\partial(\nabla g\lambda)}{\partial p_i} = B_{i\nabla g}\frac{\partial F_c}{\partial p_i} + C_{i\nabla g}\frac{\partial\lambda}{\partial p_i} \tag{4.37}$$

with :$B_{i\nabla g} = \begin{bmatrix} 0_{(6,1)} \\ B_{fric(nb\_c,1)} \\ 0_{4\times nb\_c,1} \end{bmatrix}$ and $C_{i\nabla g} = \nabla g$.

Computation of $\frac{\partial g}{\partial p}$

$$g = \begin{bmatrix} (J_2F_c - F_{ext})_{(6,1)} \\ \left(f_{x_i}{}^2 + f_{y_i}{}^2 - \sigma_i^2 f_{z_i}{}^2\right)_{(nb\_c,1)} \\ \begin{pmatrix} x_{min}f_z + t_y \\ x_{max}f_z + t_y \\ y_{min}f_z - t_x \\ y_{max}f_z - t_x \end{pmatrix}_{(4\times nb\_c,1)} \end{bmatrix} \tag{4.38}$$

$$\frac{\partial g}{\partial p} = \begin{bmatrix} \left(J_2\frac{\partial F_c}{\partial p_i} - \frac{\partial F_{ext}}{\partial p_i}\right)_{(6,nb\_p)} \\ \left(2f_{x_i}\frac{\partial f_{x_i}}{\partial p}2f_{y_i}\frac{\partial f_{y_i}}{\partial p} - 2\sigma_i^2 f_{z_i}\frac{\partial f_{z_i}}{\partial p}\right)_{(nb\_c,nb\_p)} \\ \begin{pmatrix} x_{min}\frac{\partial f_z}{\partial p} + \frac{\partial t_y}{\partial p} \\ x_{max}\frac{\partial f_z}{\partial p} + \frac{\partial t_y}{\partial p} \\ y_{min}\frac{\partial f_z}{\partial p} - \frac{\partial t_x}{\partial p} \\ y_{max}\frac{\partial f_z}{\partial p} - \frac{\partial t_x}{\partial p} \end{pmatrix}_{(4\times nb\_c,nb\_p)} \end{bmatrix} = [0]_{(6+5\times nb\_c,nb\_p)} \tag{4.39}$$

We can transform it to get an equation for one derivate parameter p_i:

$$\frac{\partial g}{\partial p_i} = \begin{bmatrix} (J_2)_{(6,1)} \\ \begin{pmatrix} .. & 2f_{x_i} & 2f_{y_i} & -2\sigma_i^2 f_{z_i} & 0 & 0 & 0 & .. \end{pmatrix}_{(nb\_c,1)} \\ \begin{pmatrix} .. & 0 & 0 & x_{min} & 0 & 1 & 0 & .. \\ .. & 0 & 0 & x_{max} & 0 & 1 & 0 & .. \\ .. & 0 & 0 & y_{min} & -1 & 0 & 0 & .. \\ .. & 0 & 0 & y_{max} & -1 & 0 & 0 & .. \end{pmatrix}_{(4\times nb\_c,1)} \end{bmatrix}_{6+5\times nb\_c,1} \begin{bmatrix} \frac{\partial f_0}{\partial p_i} \\ \frac{\partial t_0}{\partial p_i} \\ \dots \\ \frac{\partial f_{nb\_c}}{\partial p_i} \\ \frac{\partial t_{nb\_c}}{\partial p_i} \end{bmatrix} - \begin{bmatrix} \left(\frac{\partial F_{ext}}{\partial p_i}\right)_{6,1} \\ (0)_{nb\_c,1} \\ (0)_{4\times nb\_c,1} \end{bmatrix} = 0 \tag{4.40}$$

We see this expression is of the shape:

$$\frac{\partial g}{\partial p_i} = B_{ig}\frac{\partial F_c}{\partial p_i} + A_{ig} = 0 \tag{4.41}$$

Computation of $\frac{\partial F_c}{\partial p}$ and $\frac{\partial \Gamma}{\partial p}$

We remember the derivate of optimiality condition :

$$\frac{\partial \nabla f}{\partial p} + \frac{\partial (\nabla g \lambda)}{\partial p} = 0 \tag{4.42}$$

$$\frac{\partial g}{\partial p} = 0 \tag{4.43}$$

We consider only one derivative parameter and we replace the elements with their values :

$$A_{i\nabla f} + B_{i\nabla f}\frac{\partial F_c}{\partial p_i} + B_{i\nabla g}\frac{\partial F_c}{\partial p_i} + C_{i\nabla g}\frac{\partial \lambda}{\partial p_i} = 0 \tag{4.44}$$

$$B_{ig}\frac{\partial F_c}{\partial p_i} + A_{ig} = 0 \tag{4.45}$$

We assemble these two equations and we get a equation system :

$$B_i\frac{\partial F_c}{\partial p_i} + C_i\frac{\partial \lambda}{\partial p_i} = A_i \tag{4.46}$$

with : $B_i = \begin{bmatrix} B_{i\nabla f} + B_{i\nabla g} \\ B_{ig} \end{bmatrix}$, $C_i = \begin{bmatrix} C_{i\nabla g} \\ 0 \end{bmatrix}$ and $A_i = \begin{bmatrix} -A_{i\nabla f} \\ -A_{ig} \end{bmatrix}$.

The B_i and C_i matrix are the same for all parameters p_i, that's why to decrease the computing time, we can gather them and reverse it one time to find the solution:

$$\begin{bmatrix} B_i & C_i \end{bmatrix} \begin{bmatrix} \frac{\partial F_c}{\partial p_i} \\ \frac{\partial \lambda}{\partial p_i} \end{bmatrix} = A_i \tag{4.47}$$

Therefore, after solving this equation with a classical algorithm for all parameters we obtain the value of : $\frac{\partial F_c}{\partial p}$, then we can compute :$\frac{\partial \Gamma}{\partial p}$ with eq(4.8).

4.3.5 Algorithm

The algorithm for a multi-contact motion optimization with a local optimization of contact forces is shown in fig (4.3)

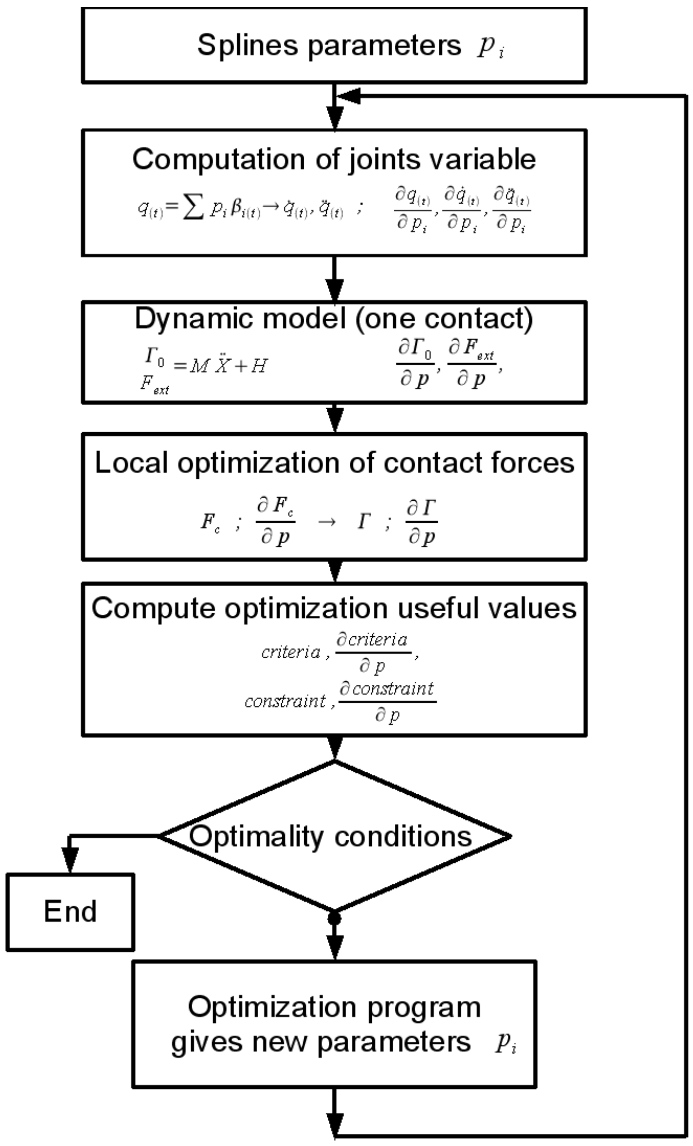

Splines parameters p_i
Computation of joints variable
$q_{(t)} = \sum p_i \beta_{i(t)} \rightarrow \dot{q}_{(t)}, \ddot{q}_{(t)}$; $\frac{\partial q_{(t)}}{\partial p_i}, \frac{\partial \dot{q}_{(t)}}{\partial p_i}, \frac{\partial \ddot{q}_{(t)}}{\partial p_i}$
Dynamic model (one contact)
$\begin{matrix} \Gamma_0 \\ F_{ext} \end{matrix} = M \ddot{X} + H$ $\frac{\partial \Gamma_0}{\partial p}, \frac{\partial F_{ext}}{\partial p},$
Local optimization of contact forces
F_c ; $\frac{\partial F_c}{\partial p}$ $\rightarrow$ Γ ; $\frac{\partial \Gamma}{\partial p}$
Compute optimization useful values
$criteria, \frac{\partial criteria}{\partial p},$
$constraint, \frac{\partial constraint}{\partial p}$
Optimality conditions
End
Optimization program gives new parameters p_i

Figure 4.3: Motion optimization algorithm with local contact forces optimization

Chapter 5

Results

5.1 Computation time

We will present the optimization computation time for a one-contact motion (5.1) and a multi-contact motion (5.2) with different choices.

	Dynamic model	Gradient dynamic model	optimization
One contact motion optimization	3,2 ms	73 ms	1 ~ 10 min

Figure 5.1: Computation time for one-contact motion optimization

	Dynamic model	Gradient dynamic model	Optimization (estimated)
Multi-contact optimization with contact forces optimization (IPOPT)	1380 ms	1830 ms	1 ~ 2 days
Multi-contact optimization with contact forces optimization (Knitro)	340 ms	800 ms	10 ~15 hours
Multi-contact optimization with contact forces optimization (FSQP)	303 ms	750 ms	10 ~ 15 hours
Multi-contact optimization with contact forces as splines	3,9 ms	280 ms	5 ~ 30 min

Figure 5.2: Computation time for multi-contact motion optimization

We can see the dynamics model is very slow with a local optimization, whereas when we compute the contact forces as splines the dynamic model is as fast as in one-contact motion.

When we do a local optimization we need to use an appropriate program, we can see IPOPT is very slow for this optimization whereas Knitro and FSQP are a bit faster. We plan to test anu other program, such as SOCP, to decrease the computation time .

The optimization computation time for multi-contact motion is only evaluated, because we get some troubles to test it. With the local optimization, we didn't test it because the computation time would have been too long. And in the case of contact forces as splines we didn't have enough time to finish the implementation.

5.2 Throwing motion

We present an example for motion optimization. We choose a throwing motion. To optimize this motion we need to add these constraints :

- Position and orientation constraint on contact bodies.
- Velocity constraint on the hand at one instant , with position constraint.

For the moment we have programming trouble, so we cannot optimize this motion yet, but we hope we can do it for the presentation.

Conclusion

We have shown that, for a multi-contact motion, we need to take into account the contact forces to compute the dynamic model. The contacts create redundant closed chain, so we have to optimize the contact forces to minimize the torques and so the energy consumption of the robot. To do this optimization, we can add optimization parameters to compute those contact forces, or do a local optimization. The local optimization is very slow to compute but return an optimal value for all the instants, on the contrary computing the contact forces thanks to parameters is faster. Nevertheless we assume we know the shape of this contact forces but we are not sure this shape is the real one. So we have to choose between the computation time and the optimality of the motion.

With only one contact the robot cannot execute a lot of motion. Therefore this multi-contact motion optimization allows to increase the range of the feasible optimal motion. Some programming trouble prevented us to present an optimal throwing motion yet.

Bibliography

[1] S. MIOSSEC, K. YOKOI, and A. KHEDDAR, "Developpment of software or motion optimization of robots - application to the kick motion of the hrp-2 robot," *Submitted to Robio*, 2006.

[2] *Introduction to IPOPT : a tutorial for downloading, installing and using IPOPT*.

[3] O. von Stryk, "Optimal control of multibody systems in minimal coordinates," *Zeitschrift für Angewandte Mathematik 78*, 1998.

[4] S.-H. Lee, J. Kim, F. Park, M. Kim, and J. E. Bobrow, "Newtom-type algorithms for dynamics-based robot movement optimization," *IEEE Transactions on robotics*, 2005.

[5] R. Featherstone, *Robot Dynamics Algorithms*. Kluwer Academic Publishers, 2003.

[6] W. Khalil and E. Dombre, *Modélisation, identification et commande des robots*. Hermès, 1999.

Summary

English summary

During this training period I had to modify a one-contact motion optimization program to optimize a multi-contact motion that will allow to increase the range of the possible optimal motions.

First, we studied the working of a one contact motion optimization, thus, next we highlighted what we have to modify to do a multi-contact optimization. As there are several contacts, a close chain appears in the dynamic model. Moreover, humanoid robots are highly redundant, so that the close chain is over-actuated. So, we had to find out the best torques possible in this chain. For that operation, we optimized the contact forces of the robot with its environment. We faced two possibilities for that optimisation :

- Either, approximate those contact forces with a parameterized function (those parameters will be added to the initial parameters of the motion optimisation)
- Or, operate an internal optimization which will return the optimal value of the contact forces each time we need to know the torques.

In that report, we compare the proprieties of those two methods and mostly the computation time which can cover a period ranging from only a few minutes to some days. This laps of time depends not only on the method used but also on the relevance of the optimization program choice. We have planned to test this optimization program by having the robot throw an object but some programming problems prevented us to present that motion in this report.

Résumé (French summary)

Le but de ce stage était de modifer un programme d'optimisation de mouvement dans lequel le robot avait seulement un contact avec son environnement, afin d'effectuer une optimisation de mouvement avec plusieurs contacts. Ce qui nous permettra d'étendre la gamme des mouvements optimaux possibles.

Nous avons d'abord étudié le fonctionnement de l'optimisation pour un seul contact, pour ensuite, mettre en évidence les modifications à effectuer pour une optimisation multi-contacts.

La présence de plusiseurs contacts amène l'apparition de boucles fermées dans la modélisation. De plus les robots de type humanoïde sont hautement redondants ce qui nous donne des boucles fermées redondantes. Il nous a fallu alors déterminé la meilleure répartition des couples possibles dans ces chaînes. Pour cela nous avons optimisé les forces de contact du robot avec son environnement. Pour cette optimisation nous avions le choix entre:

- Soit, approximer ces forces de contact par une courbe paramétrée, dont les paramètres vont être ajoutés aux paramètres initiaux de l'optimisation de mouvement.
- Ou alors, effectuer une optimisation interne qui retournera la valeur optimale des forces de contact à chaque fois que nous aurons besoin de connaitre les couples.

Dans ce rapport nous comparons les propriétés de ces deux méthodes et notamment le temps de calcul qui peut s'étendre de, seulement, plusieurs minutes à quelques jours. Ce temps dépend de la méthode choisie mais également de la pertinence du choix des programmes d'optimisation.

Nous avons prévu de tester ce programme d'optimisation sur un mouvement de lancet d'objet mais des problèmes de programmation ne nous permettent pas de présenter ce mouvement dans ce rapport.