Development of a software for motion optimization of robots - Application to the kick motion of the HRP-2 robot

Sylvain Miossec
JRL AIST
Tsukuba, Japan
sylvain.miossec@aist.go.jp

Kazuhito Yokoi
JRL AIST
Tsukuba, Japan
kazuhito.yokoi@aist.go.jp

Abderrahmane Kheddar
JRL CNRS
Tsukuba, Japan
kheddar@ieee.org

***Abstract*— This paper presents software that has been devised for the purpose of optimal motions computation for robots. For the time being, this study considered only open-loop and fully actuated structures. Generated motion satisfies a stability criterion while minimizing energy consumption. Motion optimization is solved with the IPOPT optimization package. In order to achieve fast and reliable optimization convergence, an efficient calculation of gradients is presented. Moreover, almost rarely considered in the literature, joint friction, which has a non-negligible effect on the optimized motion, is taken into account. Efficiency of the proposed software is demonstrated through the optimization of a kicking motion for the 30 degrees of freedom humanoid robot HRP-2.**

***Index Terms*— Optimal motion, humanoid, joint friction.**

I. Introduction

In the literature, considerable work has been devoted to optimal control of systems. This is due to the fact that in many fields, e.g. aerospace and robotics, it is beneficial to obtain energy or time optimal motions. There are mainly two numerical approaches to solve such problems: (i) indirect methods consist in using the Pontryagin's Maximum Principle, which results in solving a two-points boundary value problem, and (ii) direct methods that consist in solving the discretized problem by using parametric optimization techniques. Indirect methods are known to be more precise relatively to direct ones, but their domain of convergence is smaller. Since they are relatively more convenient, direct methods have been used extensively.

Direct methods can be classified in three approaches: (i) Collocation methods for which both discretized control variables and state variables are the motion parameters, and are constrained to satisfy the model of the system at collocation points, (ii) Multiple-shooting methods for which both discretized control variables and state variables are also the motion parameters, but equality constraints are imposed between a state at a discretization point and the state obtained from numerical integration from the previous discretization point, and (iii) Methods based on parameterization of the state variables by using functions like B-splines or polynomials, and where the control variables are obtained from inverse dynamic model. Collocation method and multiple-shooting method are based on the forward dynamic model. The results in [1] and [2] have proved that optimal motion based on state variables parameterization and inverse dynamic model are more efficient in the robotics' context.

Determining optimal motions for complex robots is still computationally costly. Improving computation speed is still an open problem in view of reactive control perspectives. Furthermore, joint friction, which has a non-negligible effect on the resulting motion, is rarely taken into account in optimal computations.

In [3] optimal motions for humans is studied. They simplified the human model to have six to twelve degrees of freedom (dof). In order to improve the efficiency of the optimization, they computed recursively the gradient of the dynamic model. Work in [4] presented an efficient motion optimization algorithm with a dimensionality reduction technique based on a motion capture database. They generated optimal motions for 60 dof human figures in only few minutes. However, it is by essence difficult to extrapolate the database for dissimilar systems. Even if some robots exhibit kinematics similarities with some living beings, there are differences in their dynamics and in their actuators' types and actuations' modes. Therefore, a living being optimal motion may reveal to be different from a similar-goal robot optimal motion.

For robotics, the authors in [5] considered a 4 dof planar biped robot and used a collocation method to optimize the walking motion. The authors in [6] generated optimal walking and running motions for a 4-dof planar robot using polynomials to define joints trajectory. In [7] a database of 3D optimal motions is generated for a 13-dof walking robot. But, none of these works considered joint's friction. Recently, an interesting result shows that the regularization of static friction allows to solve the problem with usual techniques, provided the discretization of the problem is sufficient [8].

In this paper we devised software to deal with such systems. For the time being, only non-cyclic robotic structures that are fully actuated are considered. We improved the efficiency of the gradient computation of the dynamic model, as proposed in [3] and [9], by considering the dependences

This research has been conducted at the AIST/CNRS Joint Japanese-French Robotics Laboratory (JRL) at the Intelligent Systems Research Institute, AIST Central 2, 1-1-1 Umezono, Tsukuba 305-8568, Japan, and was supported by a Post-doctoral Fellowship of the Japan Society for the Promotion of Science (JSPS)

in the recursive Newton-Euler dynamics computations. We also considered the joint friction in the motion optimization. For this, we used a regularization method proposed in [8]. Section II presents the general motion optimization problem for fully actuated robots, and how it is approximated to be solved numerically. Section III presents our software, the constraints formulation, and the gradients computation method. Section IV exemplifies our software for the case of a kicking motion implemented on the HRP-2 robot. However, the application to a real kick is not experienced, since one needs to take into account the impact with the kicked object. Finally, conclusions and perspectives are given in section V.

II. Problem statement

A. Exact problem

We consider the parameters of a fully actuated Lagrangian dynamic system as the vector q. The general dynamic equation is written as

$$u = f(q, \dot{q}, \ddot{q}) \tag{1}$$

where u is the control vector. The Lagrangian system may be subject to technological constraints that hold during the whole motion, that is

$$c_t(q, \dot{q}, \ddot{q}, u) \leq 0 \tag{2}$$

Constraints translating specifications of the system's desired motion are also considered. One distinguishes between constraints that hold during the whole duration of the motion,

$$\begin{cases} c_{meq}(q) = 0 \\ c_{mineq}(q) \leq 0 \end{cases} \tag{3}$$

and the constraints that holds only at discrete-time t_d,

$$c_{mt}(q(t_d)) = 0 \tag{4}$$

The constraints considered in this paper will be detailed in section III-D.

The criteria for which the motion will be optimized will be written as $\mathcal{C}(q, \dot{q}, \ddot{q}, u, t_f)$, where t_f is the duration of the motion.

The motion optimization problem to solve is the following

$$\begin{aligned} &\min_{q(t),u(t),t_f,t_d} \mathcal{C}(q(t), \dot{q}(t), u(t), t_f) \\ &\text{subject to (1), (2), (3), (4)} \end{aligned} \tag{5}$$

The goal is to find the optimal functions $q(t)$, $u(t)$, the final time t_f, and the times t_d at which constraints apply. For simple problems, the Pontryagin's Maximum Principle allows finding analytical solutions. In the general case, no method exists to find $q(t)$, $u(t)$, t_f and t_d analytically or exactly; numerical methods are used instead. In section II-B we present how we approximate this problem so that it can be solved with existing numerical methods.

B. Approximate discretized problem

In order to solve the problem (5) we approximate it with a discretized version. If the criterion is an integral over the motion, it is determined by numerical integration.

1) Constraints discretization: The problem (5) with constraints (2) and (3) that apply during the entire duration of the motion is a semi-infinite optimization problem. One can refer to [10] for a review of the methods solving such problems. We are rather considering the constraints at discrete instants during the motion. Subsequently, the number of constraints is reduced to a finite one. This way allows one to use available nonlinear optimization packages. Other methods consist in changing the discretization of the constraints during the optimization. Those methods are also based on classical nonlinear optimization packages, but need to restart an optimization after each new constraints discretization. For simplicity reasons and computational speed, we did not consider discretization adaptation and perform a single run optimization. This will certainly result in a less precise constraints satisfaction but a faster motion generation.

The constraints (2) and (3) are discretized at $N+1$ instants $t_k = \frac{kt_f}{N}$, $k \in [0, N]$, so far written

$$c_t(q(t_k), \dot{q}(t_k), u(t_k)) \leq 0 \tag{6}$$

$$\begin{cases} c_{meq}(q(t_k)) = 0 \\ c_{mineq}(q(t_k)) \leq 0 \end{cases} \tag{7}$$

In the following, the variables labeled with subscribe k are those considered at the time discretization of constraints.

2) Motion parameterization: The original problem (5) is a function optimization. We reduce this problem to parameter optimization by considering the joint angles as functions depending on given parameters p, that is

$$q(t) = q(p, t) \tag{8}$$

These parameterized functions can be polynomials, B-splines, or any other function basis.

3) Obtained problem: We make an additional modification of the original problem, by considering that the control variables u are not parameterized but obtained from the dynamic model (1). This is possible because we are considering fully actuated systems. Finally, the optimization problem writes

$$\begin{aligned} &\min_{p,t_f,t_d} \mathcal{C}(q(p,t), \dot{q}(p,t), u(t), t_f) \\ &\text{subject to (6), (7), (4)} \end{aligned} \tag{9}$$

In the case of under-actuated systems, a collocation approach could be envisaged so that non-actuated degrees of freedom satisfy the dynamic model. This problem has been addressed in [9] with a slightly different method. In the case of over-actuation, it is possible to parameterize some of the torques u or choose them so that they minimize the criteria, e.g. [11].

III. Presentation of the software

The software includes the definition of the motion characteristics, the dynamic computations, the joints computations as B-spline, the constraints definition and computation and the criteria computation. Gradients are also computed. The software includes an interface with the optimization program IPOPT (see [12] for more details). This interface computes the size of our problem, the criteria and constraints, and their gradient.

A. General systems considered

We consider kinematics chains composed of revolute joints each of which having viscous and dry friction. As stated previously, we only address systems with one fixed contact with the environment of surface-surface contact or fixed joint base.

B. Basis of function

Trajectory parameterization of a given joint angle (j) is made with B-spline functions, which is written

$$q_j(p,t) = \sum_{i=1}^{n_{pj}} B_i(t) c_{ij} \tag{10}$$

where n_{pj} is the number of basis functions, c_{ij} is the B-spline coefficient for the i^{th} basis function $B_i(t)$. The parameters of the motion are then $p = \{c_{ij} \mid i \in [1,N], j \in [1,n_{pj}]\}$. There is $N \times n_{pj}$ parameters. N is the total number of joints.

We choose B-spline with 3^{rd} degree polynomial basis function. Hence, the resulting motion obtained is continuous and C^2 differentiable (up to the acceleration). An important advantage of this B-spline is that at an instant, a joint depends only on 4 B-spline coefficients. Moreover, the computation time of the gradient is consequently reduced. Other parameterizations by polynomials induce joint value to depend on more coefficients at a given instant. For instance, polynomials of degree five that allow acceleration continuity and six boundary conditions interpolation (initial and final: position, velocity and acceleration), induces a given joint angle to depend on the six boundary conditions at a given time.

For the parameterization of a point-to-point motion (with zero initial and final velocity and acceleration), we implemented the computation of the three initial and the three final B-spline coefficients with respect to the initial configuration q_{init} and the final configuration q_{final}. The parameters of a motion are then $p = \{q_{\text{init}}, q_{\text{final}}, c_{ij} \mid i \in [1,N], j \in [4, n_{pj}-3]\}$. There is $N \times (n_{pj} - 4)$ parameters.

C. Dynamics computations

1) Dynamic model considered: The dynamic of the considered class of system can be modeled as follows,

$$u_m = A(q)\ddot{q} + H(q,\dot{q}) \tag{11}$$

$u_m \in \mathbb{R}^n$ is the joint torque applied to the mechanical system, $q \in \mathbb{R}^n$, $\dot{q} \in \mathbb{R}^n$, $\ddot{q} \in \mathbb{R}^n$ are respectively the joints' positions, velocities and accelerations, $A(q) \in \mathbb{R}^{n \times n}$ is the inertia matrix, and $H(q,\dot{q}) \in \mathbb{R}^n$ is the vector of Coriolis, centrifugal and gravity effects.

This dynamic model is computed with the recursive Newton-Euler algorithm. In addition to joint torques, it calculates interaction efforts between the environment and the contacting base, noted as f for the force and m for the moment.

The joint friction are included by considering the control input being u, the joint torque u_m with the effects of friction, that is

$$u_j = u_{m,j} + \frac{2u_{d,j}}{\pi} \arctan(\dot{q_j} c_r) + u_{v,j}\dot{q_j} \tag{12}$$

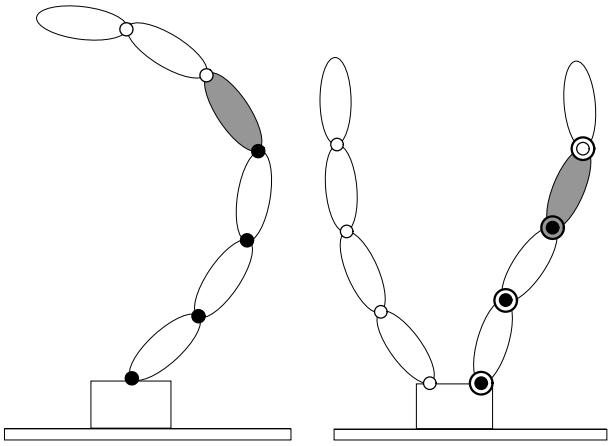
u_d is the torque of static friction, c_r is the regularization coefficient for the static friction, u_v is the coefficient for viscous friction. The bigger c_r is, the better the static friction approximation, but also the stiffer the system is.

2) Gradient of dynamic model: For the efficiency of the optimization process, it is recommended to compute analytically and efficiently the criterion and the constraints gradients with respect to the optimization parameters p. The main computation occurs with the gradient of joint torques $\frac{\partial u_k}{\partial p}$. $\frac{\partial u_k}{\partial p}$ will be computed from: the gradient of joint positions $\frac{\partial q_k}{\partial p}$, the gradient of joint velocities $\frac{\partial \dot{q}_k}{\partial p}$, and the gradient of joint accelerations $\frac{\partial \ddot{q}_k}{\partial p}$. Such computations can be found in references [3], [9], [11]. The improvement of our gradient computation comes from the consideration of the system structure which allows computing only the non-zero components of the gradient. For this, we analyzed the dependencies of the recursive Newton-Euler method:

- the first recursion of the Newton-Euler algorithm goes from the base to the extremity of branches computing the positions, orientations, velocities and accelerations of all bodies composing the robot. It is worth noticing that for a given body, the computation concerns only joints which link it to the base. Therefore the gradient of the position, orientation, velocity and acceleration of this body will not depend on the remaining joints (i.e. those not connecting the given body to the base).
- the second recursion of the Newton-Euler algorithm goes from extremities (leafs) back to the reference body; joint screws and torques are computed. Subsequently, for a given body, computation will depends on all the farther joints in the branch (i.e. its childes), and joints which link it to the base. The gradient of a joint torque will not depend on the remaining joints.

For instance, consider the serial and branched system illustrated in fig. 1. For the serial system, the gradient computations of the first recursion with dependencies will be about $1/2$ of the computation without taking into account dependencies. In this case however, the gradient computations for the second recursion will be the same with or without dependencies. For the branched system case, the gradient computations of the first recursion with dependencies will be about $1/4$ of the computation which does not take into account dependencies. In this case, the gradient computations during the second recursion will be $1/2$ of the computation without dependencies consideration. Therefore, the computation gains can be substantial for branched systems.

For gradient computation at a discrete instant k, we took into account the time decoupling of splines. That is to say, the joints positions, velocity and accelerations are defined as B-splines. Because of its chosen degree, and for a given instant, a B-spline depends only on four of its coefficients among the n_{pj} per joint. This dependency decreases the gradient



(a) Serial chain (b) Branched chain

Fig. 1. Representation of the computation dependencies for serial and branched chains. The square body is the contacting base with the environment. The computations of the first recursion for the gray body depend only on the black joints. The computations of the second recursion for the big gray joint depend only on the big joints (we did not represented those big joints for the serial chain, since the second recursion computations depend on all joints).

computation time by $\approx \frac{4}{n_{pj}}$ relatively to the time it would take without it.

D. Constraints

Several constraints can be added to the system. They can be divided into two types: (i) physical limitations of the system and (ii) motion constraints defining the characteristics of the desired motion.

The physical limitations are expressed as follows:

- joint limits are given by

$$q_{\min} \leq q \leq q_{\max} \tag{13}$$

  $q_{\min}$, $q_{\max}$ are respectively the minimal and maximal limits.
- actuators limits are approximated by linear inequalities on u and $\dot{q}$

$$au + b\dot{q} + c \leq 0 \tag{14}$$

- constraints of no sliding, no take off, no turn over the edges of the contact of the base are given by

$$\begin{cases} \sqrt{f_x^2 + f_y^2} - \mu f_z \leq 0 \\ -f_z x_{\text{neg}} \leq f_x z_{\text{surf}} + m_x \leq f_z x_{\text{pos}} \\ -f_z y_{\text{neg}} \leq f_y z_{\text{surf}} - m_y \leq f_z y_{\text{pos}} \end{cases} \tag{15}$$

  In the contact frame, $f_{x,y}$ and f_z are respectively the tangential and normal components of the external forces acting on the base, $m_{x,y}$ are the external moments acting on the base, x_{neg}, x_{pos}, y_{neg}, y_{pos} are the edges of the contact surface respectively in x and y directions, and in negative and positive directions, and z_{surf} is the distance along normal axis between the surface and the origin of base. We made the hypothesis that the contact surface is rectangular and aligned with frame axis of the base. We did not considered the friction constraint in rotation, as the model of this friction is still an open problem, and this constraint is rarely limiting the motion. However we plan to study the question and include such a constraint in future work.

The constraints on the motion that are considered:

- position constraints of body j, at some discrete instant t_d of the motion, given by

$$\text{pos}_j(q(t_d)) = \text{pos}_{j,d}^0 \tag{16}$$

  where $\text{pos}_j(q)$ is the position of the body's origin j, $\text{pos}_{j,d}^0$ is the desired position for the body j at instant t_d.
- inequality constraints on the position of body j of the system, for all discrete instants t_k, are given by

$$n.\text{pos}_j(q(t_k)) \leq n.\text{OP}_{\text{front},j}^0 \tag{17}$$

  where n is the normal to the plane delimiting the admissible zone for the origin of the body j, $\text{OP}_{\text{front},j}^0$ is a vector from the origin of absolute frame to a point belonging to this plane, and symbol '.' is the scalar product.
- orientation constraint of body j of the system, at discrete instant t_d is given by

$$\begin{cases} \theta_1(q(t_d)) = \arctan\left(\frac{u_{j,x_j z_j}.z_j}{u_{j,x_j z_j}.x_j}\right) = 0 \\ \theta_2(q(t_d)) = \arctan\left(\frac{u_{j,x_j y_j}.y_j}{u_{j,x_j y_j}.x_j}\right) = 0 \\ \theta_3(q(t_d)) = \arctan\left(\frac{v_{j,y_j z_j}.z_j}{v_{j,y_j z_j}.y_j}\right) = 0 \end{cases} \tag{18}$$

  where (u_j, v_j, w_j) is the frame of body j, (x_j, y_j, z_j) is the absolute frame to which (u_j, v_j, w_j) superposes, the symbol '.' is the scalar product, and $u_{j,x_j z_j}$ is the projection of u_j in the plane defined by vectors x_j and z_j. This way, the orientation is simple to compute; avoiding singularities and behaving well during the optimization process. It also has a unique solution.
- fixed joint angle j for all discrete instants t_k is given by

$$q_j(t_k) = q_j^0 \tag{19}$$

  where q_j^0 is the reference value for the joint angle q_j.

E. Criteria considered

For the time being, we implemented an energy criterion, but other useful criteria can be easily programmed, e.g. minimize the time to perform a motion or maximize a speed at a given instant of the motion, etc. The energy criterion we considered takes into account the mechanical loss due to friction, and the loss in the resistor of the motor. This criterion includes the fact that the energy restitution can be *stored* by the robot to be reused later. The resulting criterion is given by

$$\mathcal{C}(\dot{q}, u, t_f) = \int_{t_0}^{t_f} \sum_{j=1}^{N} \frac{R_j {u_j}^2}{K_{em,j}^2} + u_j \dot{q}_j \, dt \tag{20}$$

where R_j is the resistance and $K_{em,j}$ the electro-mechanical constant for the actuator of joint j.

IV. CASE STUDY

The proposed software has been used for the generation of an optimal trajectory kicking motion for the Humanoid robot HRP-2. The HRP-2 robot has 30 degrees-of-freedom (dof). It has 6 dof per leg, 7 dof per arm (including for the gripper), 2 dof between waist and chest, and 2 dof for the head. For more detail on the HRP-2 robot, readers may refer to [13]. For the optimal motion generation, we used a dynamic model of the robot including static friction. The dynamic parameters were obtained from the CAD Design of the robot. The joint static friction used was obtained from the characteristics of the motors and the reduction ratio. The static friction of the reduction system and the viscous friction could not be determined. An experimental identification is necessary to know more precisely these values.

In section IV-A we present how the desired motion is defined, and in section IV-B we present the optimal motion obtained and some characteristics of the process of motion generation. The energetic criteria (20) is used.

A. Motion considered

To specify a kicking motion, we imposed some of the constraints presented in section III. We considered joints limits, actuators limits obtained from the characteristics of the motors, no sliding, no take off, and no turn over of the supporting foot. To obtain a kicking motion, we specified initial and final equality constraints (16) on the position of the free foot. To deal with auto-collision we used inequality constraints (17) on the position of the hands and elbows. Those inequality constraints were tuned in order to obtain enough clearance without over-constraining the motion. In order to implement the motion on the real robot, we also considered a constraint on the knee of the supporting leg. Indeed, the stabilizer implemented on the robot, that correct the effects of flexibility in the feet and control the ZMP position, does not deal with stretched legs. Therefore, we imposed a bending the knees by at least 20 degrees simply by modifying the joint limit of the knee.

We used a point-to-point parameterization motion, as presented in III-B. Therefore, the initial and final configurations of the robot are also optimized to obtain an energy efficient motion.

B. Results

For the parameterization of the motion, we choose $n_{pj} = 9$ B-spline basis functions. We obtain a total of $n_p = 151$ optimization parameters. This B-spline parameterization gives six third-degree polynomials over the motion, distributed on six equal-length intervals. For the computation of the criterion we used 61 integration points and a trapezoidal integration scheme. For the constraints, we considered 13 discrete points.

The kicking motion obtained is presented in fig. 2. It also presents the experimental kicking motion. The motion was stable on the real robot. Detailed results will be presented in a further publication. We believe that taking into account the viscous component of joint friction will give a slower motion (high speed will be penalized). The table I presents the computation times, and criteria obtained, depending on some parameters used for the optimization software IPOPT. This table also compares the results between a motion without the bend knee constraint, and with the bend knee constraint, imposed in order to implement the motion on the robot. Fig. 2 presents the bend knee motion with high BFGS history. We can see that a very good optimal motion can be obtained very fast, in about 1min 20sec. A little gain of criteria improvement of about 5% can be obtained, but at a high computational cost (by improving the Hessian approximation and increasing the number of iterations). Furthermore, we can see that the control software limitation imposes to consider a motion with bend knee that consumes about 3 times more energy. Finally, we can see that lots of time is spent in the optimization package. We plan to improve this by implementing full knowledge of the sparsity pattern of gradient computations. Indeed, in the sparsity pattern definition, we did not take into account the dependencies of bodies positions and joint torques like we did in the gradient computation presented in section III-C.2.

TABLE I

RESULTS OF DIFFERENT MOTION OPTIMIZATION, WITH AND WITHOUT THE CONSTRAINT ON THE KNEE OF SUPPORTING LEG, AND DIFFERENT PRECISION OF THE SOLUTION.

Motion	with knee constraint		without knee constraint	
BFGS history	70	150	70	150
Number of iterations	300	1000	300	1000
Total time	1min 24s	12min 45s	1min 19s	13min 18s
IPOPT time	54s	11min 9s	50s	11min 41s
Constraints and criteria computation time	30s	1min 36s	29s	1min 37s
Optimized criteria	239.7 kJ	225.4 kJ	83.7 kJ	81.2 kJ
Constraints verification	$2.6\ 10^{-4}$	$1.0\ 10^{-7}$	$7.4\ 10^{-4}$	$3.3\ 10^{-3}$

For the moment, we did not succeed to make IPOPT package finish properly the optimization. The optimization converges but does not meet the optimality conditions. We suppose that this may come from three problems:

- the problem might be scaled badly. But there are no rules to perform a good scaling and it can be obtained only by trial and error;
- our computations are not precise enough: lots of computations are needed for the dynamics, and numerical errors can accumulate;
- the Hessian of the problem is approximated with limited-memory BFGS method. And we noticed that the quality of convergence depends on the history of this computation.

V. CONCLUSION

In this paper, we have devised dedicated software for the optimization of motions for robots with acyclic kinematics

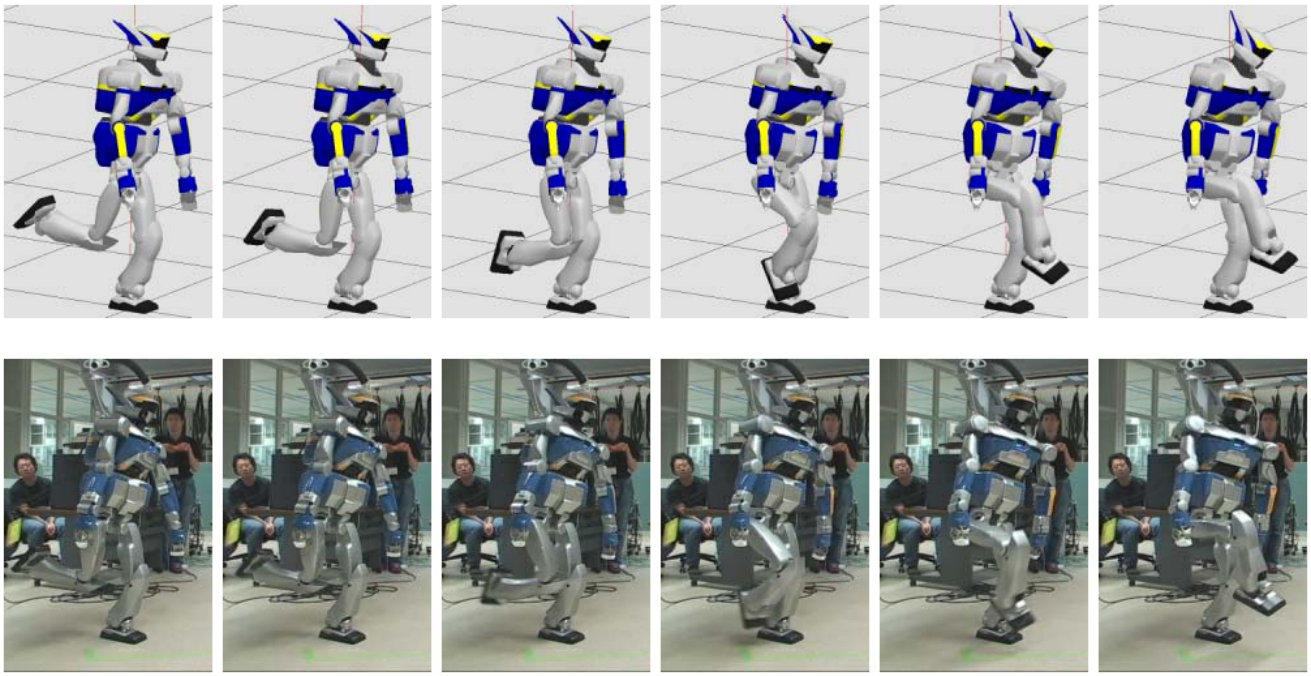


Fig. 2. Optimized kicking motion.

chains. We presented an efficient way to compute the gradient of the dynamics with respect to optimization parameters, by considering the dependencies of the dynamics computations. We have shown the efficiency of our algorithm for a kicking motion of the 30-dof HRP-2 robot. We have also taken into account the joint friction in order to obtain optimal motions closer to real optimal motions of robots. But we still have to identify the real joint frictions. We plan to include the auto-collision avoidance and the no-sliding constraint of a contact in rotation. We are implementing the motion optimization for the case of closed kinematics chains.

REFERENCES

[1] O. V. Stryk, "Optimal control of multibody systems in minimal coordinates," in *Proceedings of the Annual GAMM Conference*, 1997.

[2] M. C. Steinbach, "Optimal motion design using inverse dynamics," Tech. Rep., 1997.

[3] J. Lo, G. Huang, and D. Metaxas, "Human motion planning based on recursive dynamics and optimal control techniques," *Multiboldy System Dynamics*, vol. 8, pp. 433–458, 2002.

[4] A. Safonova, J. Hodgins, and N. Pollard, "Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces," *ACM Transactions on Graphics*, vol. 23, no. 3, pp. 514–521, August 2004.

[5] M. Hardt, K. Kreutz-Delgado, and J. W. Helton, "Optimal biped walking with a complete dynamical model," in *Proceedings of the 38th IEEE Conference on Decision and Control*, 1999.

[6] C. Chevallereau and Y. Aoustin, "Optimal reference trajectories for walking and running of a biped robot," *Robotica*, vol. 19, pp. 557–569, 2001.

[7] J. Denk and G. Schmidt, "Synthesis of a walking primitive database for a humanoid robot using optimal control techniques," in *Proceedings of IEEE-RAS International Conference on Humanoid Robots*, 2001, pp. 319–326.

[8] D. E. Stewart and M. Anitescu, "Optimal control of systems with discontinuous differential equations," *submitted*, 2006.

[9] G. Sohl, "Optimal dynamic motion planning for underactuated robots," Ph.D. dissertation, University of California, 2000.

[10] R. Hettich and K. Kortanek, "Semi-infinite programming: Theory, methods, and application," *SIAM review*, vol. 35, no. 3, pp. 380–429, 1993.

[11] S.-H. Lee, J. Kim, F. Park, M. Kim, and J. Bobrow, "Newton-type algorithms for dynamics-based robot movement optimization," *IEEE Transactions on Robotics*, vol. 21, no. 4, August 2005.

[12] A. Wachter and L. T. Biegler, "On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming," *Mathematical Programming*, vol. 106, no. 1, pp. 25–57, 2006.

[13] K. Kaneko, F. Kanehiro, S. Kajita, H. Hirukawa, T. Kawasaki, K. Hirata, M. Akachi, and T. Isozumi, "Humanoid robot hrp-2," in *Proceedings of the 2004 IEEE International Conference on Robotics and Automation*, 2004.