Bioptim, a Python interface for Musculoskeletal Optimal Control in Biomechanics

author#1[a,*], author#2[a], author#3[b] and author#4[a]

***Abstract*— The abstract**

Keywords – TODO

I. Introduction

Biomechanics researchers rely on numerical simulations of motion to gain understanding on a variety of scientific topics such as the physiological causes of movement disorders and their consequences on health [ref], the estimation of non-measurable physiological quantities [ref] and the optimality of human movement [ref]. The musculoskeletal models used in these simulations generally have a large number of degrees of freedom and they are governed by several ordinary differential equations (ODEs) which mainly describe multibody and muscle activation dynamics. The complexity of these systems has led scientists to formulate their simulations as optimal control problems (OCP), relying on efficient non-linear optimization software to find trajectories that fulfil a desired task while enforcing the system dynamics and minimizing a cost (e.g. time of execution, energy expenditure, matching experimental data, etc.).test Up to very recently, there was no off-the-shelf software available to the community to quickly formulate and solve such musculoskeletal OCPs, leading researchers to develop their—often simplified—own solutions.

As a result, many approaches coexist to formulate and solve OCPs in the biomechanics literature. The formulation, also called transcription, consists in turning a continuous trajectory optimization problem into a generic discrete non-linear program (NLP) that is solved using a dedicated algorithm. The main family of so-called *direct* transcription methods comes from numerical optimal control. They consist in straightforwardly choosing the state and/or the control as optimization variables at a given number of points along the trajectory and they rely on the integration of the system dynamics between these points. For instance, the direct collocation method has shown its efficiency in several studies investigating human motion [ref, à prendre dans papier MOCO]. It consists in approximating the integration of the system dynamics using polynomials that describe the state and control trajectories. Its main advantages are that it leads to very sparse NLPs, that knowledge about the state trajectory can be used in the initialization, and that it handles unstable systems well [**?**]. Its major disadvantage is that adaptive integration error control implies regridding the whole problem and thus changes the NLP dimensions, discarding its use for such application. Direct multiple shooting is another direct method that was also applied with success in a lot of biomechanics [ref] and robotics [ref] studies. Its advantages are mostly the same as for direct collocation in addition to combine integration error control with fixed NLP dimensions, as it relies on possibly adaptive ODE solvers to integrate the system dynamics. Besides direct methods, simpler choices can be made, as in [**?**] [+ ref Begon], where the optimization variables are instants at which a switch in the motor strategy occurs, using polynomials function (4th, 5th order) in-between, or in [**?**] [+ ref Huchez, Mowmbaur, McPhee, Opensim]], where the optimization variables are the coefficients of fourth order polynomial approximations of the states, with linking conditions to enforce the continuity of the controls. These last approaches are less generic than the direct methods as they either require a knowledge about the state and control trajectories that one often does not have when investigating complex biomechanics issues or [discuter les méthodes type leboeuf].

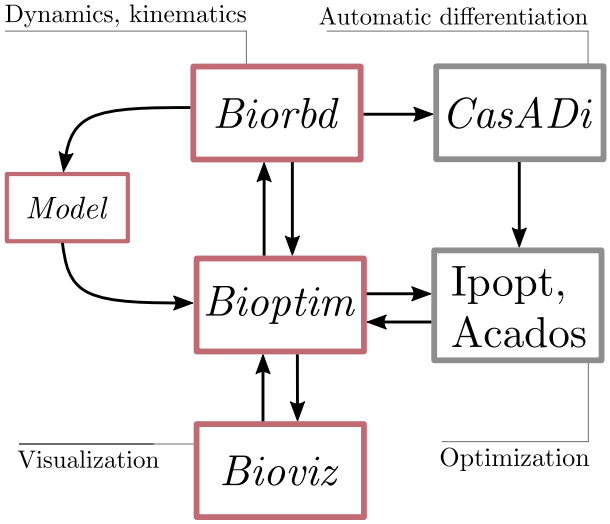

Fig. 1: *Bioptim* dependencies flowchart. The red-boxed software are developed by the S2M team. The *Bioptim* part is further detailed in Fig. **??**.

Concerning the non-linear solver, a variety of software exist and have been used to solve transcribed musculoskeletal NLPs. They can use different heuristics: interior point methods (*ipopt*, [ref]) or sequential quadratic programming (*snopt*, *acados* [ref]), but they are all gradient based. Therefore, derivatives of the NLP cost function and constraints are required to perform optimization. These derivatives can be obtained by

[a] Laboratoire de Simulation et Modélisation du Mouvement, Faculté de Médecine, Université de Montréal, Laval, QC, Canada
[b] other, elsewhere
[*] author#1@umontreal.ca

finite differences (inaccurate thus comprising convergence) or they can be computed automatically thanks to algorithmic differentiation [casadi].

In order to promote the use of musculoskeletal optimal control among biomechanics researcher, we identified a strong need for a dedicated tool, as shown by the recently launched *moco* [ref, Opensim]. The biomechanics community being mainly composed of software users [ref], such a tool should request a flexible user interface written in a widely used high-level language (e.g. Python) with a low-level core (e.g. C++) for efficiency. To develop such a software, four interrelated components are required: *i)* a musculoskeletal modeling software, with a visualization module (multibody kinematics and dynamics, muscle dynamics, etc.), *ii)* a method for automatic differentiation, *iii)* a discretization approach, and *iv)* a nonlinear programming (NLP) solver. General-purpose optimal control software (e.g. GPOPS-II [ref], Muscod-II [ref], Acado [ref]) address *ii)* to *iv)* but they need to be interfaced with a musculoskeletal modeling module and they do not provide any built-in biomechanics features (physiological cost functions, kinematic constraints, etc.). In that sense, the aforementioned *moco*, is a welcome initiative that draws its strength from its integration with the widely used *opensim*. However, it faces the following limitations: it uses finite differences to avoid the complexity of adapting the *openSim* codebase to support algorithmic differentiation, it uses direct collocation as transcription method, preventing the use of adaptive ODE solvers and it is not as flexible as required by the community, since it requires the user to develop in C++.

The objective of the present paper is to introduce *bioptim*, an open-source optimal control software dedicated to musculoskeletal biomechanics. *Bioptim* is based on C++ code for computational efficiency but the user interface is written in Python for flexibility and ease-of-use. The OCP transcription uses direct multiple shooting to preserve the possibility of using adaptive ODE solvers in the integration, which is fully parallelized for more efficiency. *Bioptim's* core is fully written in *casadi* symbolics in order to benefit from algorithmic differentiation and to exploit *casadi's* interface with several non-linear solvers (*ipopt, snopt*). Also, *bioptim* is interfaced with the cutting-edge solver *acados*, a recent NLP solver dedicated to direct multiple shooting, intended for real-time applications. The purpose of *biotpim* is to allow fast and flexible musculoskeletal OCP formulation and solving by providing a framework with a lot of typical biomechanics problem already implemented and customizable at will. The paper is organized as follows: first, the design and implementation of *bioptim* are described. Next, the versatility and performances of *bioptim* are shown through a variety of examples available online.

II. Implementation and Design

A. Implementation and dependencies

Bioptim is the top layer of a succession of software on which it depends to perform various calculations (*Biorbd*: dynamics and MSK modeling; *CasADi*: automatic differentiation; *IPOpt, Acados*: optimization; *Bioviz*: visualization). Within this software bundle, *Bioptim*'s main role is to shape the problem in order to allow its dependencies to communicate efficiently, while providing an intuitive and flexible interface to the user (Fig. **??**). Therefore, it was chosen to be written in Python for its flexibility and its widespread use among researchers. However, all intensive calculations behind the interface are performed in C or C++, keeping *Bioptim* both fast and easy to customize.

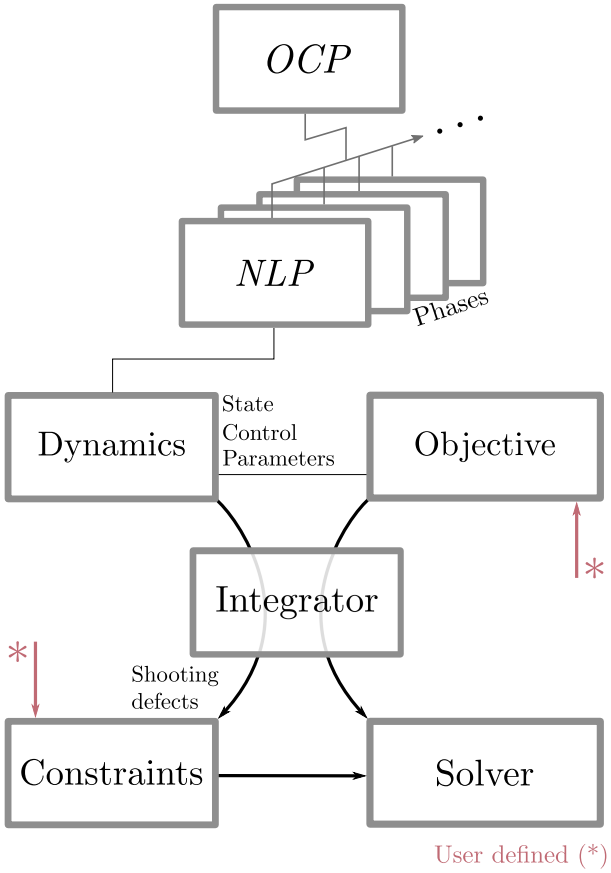

Fig. 2: *Bioptim* design flowchart.

B. Design

Bioptim shapes and solves optimal control problems whose two required entries are a model (*.bioMod* file) and an OCP. The model file contains the geometrical characteristics, the segment inertias, the geometrical markers, the actuators of the model (muscles and joint torques accounting for angle/angular velocity/torque relationships) as well as bounds on joint kinematics and torques. It also allows the user to design or import meshes for visualization purposes. The OCP is implemented as a combination of nonlinear problems (NLPs) for allowing the formulation of multi-staged OCPs. Each NLP has the following attributes: a dynamics type, an objective function, constraints, a number of shooting points, the duration of the problem and initial guesses. Based on these inputs, *Bioptim* properly sets up the multiple shooting transcription of the OCP, with appropriate continuity constraints in the case of multiple NLPs, and shapes it up to feed the chosen non-linear solver (Ipopt or Acados).

1) Dynamics types: The dynamics type defines which variables are states ($\mathbf{x}$), which ones are controls ($\mathbf{u}$) and which ones are parameters ($\mathbf{p}$). Then, it implements the ordinary differential equation governing the state transition:

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}, \mathbf{p}). \tag{1}$$

More than 10 dynamics are implemented in *Bioptim*[1], among which the controls can be muscle excitations/muscle activations/joint torques, the states can be joint kinematics/muscle activations, including/excluding contacts, etc. Even if these dynamics types exhaustively span the current usages in biomechanics, a custom dynamics type is also pre-implemented to allow easy problem customization.

2) Objective functions: Accordingly to the optimal control formalism, there are two main types of objective functions, namely Lagrange and Mayer. Lagrange types are running objectives, integrated over the NLP duration. Mayer types are time-specific objectives. Classically, they correspond to a terminal objective, but to be more versatile, they can be defined at any instant in *Bioptim*.

These objective functions can depend on any of the optimization variable, *i.e.* the controls, the states, the parameters and the duration of the problem. A lot of objective function types are already implemented in *Bioptim* (> 20), among which tracking/minimizing, on the states/controls/markers/contact forces/problem duration, etc. Should one go missing, a custom objective type is also pre-implemented.

When declaring the desired list of objective function for a given NLP, each objective function type is associated with a weight, and the user can flexibly choose on which components of the vector variables the objective must apply. If applicable (for tracking objective functions mainly), the user must also specify the numerical target of the objective.

3) Constraints:

III. Examples

In the following, seven applications are presented to illustrate the versatility of *Bioptim* and give a practical overview on how to use its main features. The performances and the Github links of each OCP are summarized in Tab. **??**.

A. Muscle activation driven pointing task

The goal was to achieve a muscle activation driven pointing task using a 2-DoF, 6-muscle arm model. In addition to muscle-induced torques, pure joint torques could compensate for the model weaknesses.

The first term of the objective function (Eq. **??**) corresponds to the pointing tasks described by a Mayer term (heaviest weight), to superimpose two markers, the first one fixed in the ulna system of coordinates and the second one fixed in the scene. The second and third were added for control regularization (muscle activation and torques) and the fourth one for state regularization:

$$\begin{aligned}\mathcal{C} =& 1e6 \text{ align\_markers} \\ &+ \int 10 \text{ (min\_muscle\_ctrl} + \text{min\_torque} + \text{min\_state)},\end{aligned} \tag{2}$$

where the names of the objective terms correspond to the nomenclature used in *Bioptim*. The movement lasted for 2 seconds and was discretized using 51 shooting nodes. The problem was solved using IPOPT and ACADOS resulting in two significantly different solutions. ACADOS provided a 16 times smaller optimized cost (Tab. **??**), which illustrate the pitfalls of local minima as well as the benefits of having straightforward access to different solvers. Indeed, the ACADOS-based solution (Fig. **??**, top) makes good use of gravity to minimize the control inputs, while the IPOPT-based solution (Fig. **??**, bottom) moved the arm in the opposite direction and was stuck in a local minimum (still achieving the task though). It is worth mentioning that for the purpose of this illustration, no constraint was given about the shoulder range of motion to ensure physiological muscle trajectories.

B. Quaternion base twisting somersault

The goal was to maximize the twist rotation (ϕ) in a backward somersault. The model is composed of a 6-DoF root segment and two 1-DoF torque actuated arms. The OCP was solved for two models. First, rotations of the root segment were expressed as Euler angles. They were expressed as a quaternion for the second model. The objective functions were written as follow:

$$\mathcal{J} = - \underbrace{\int_0^T \dot{\phi} \, dt}_{MINIMIZE\_TWIST} + \omega_1 \underbrace{\int_0^T \sum_{i=1}^{2} \tau_i^2 \, dt}_{MINIMIZE\_TORQUE}, \tag{3}$$

with $\omega_1 = 1 \times 10^{-6}$, T the duration of the movement and τ_i the torque control of the i^{th} arm DoF. The first term of the objective function (Eq. **??**) corresponds to maximizing the twist velocity and the second term is for control regularization.

The movement lasted for approximately 1 second and was discretized in 100 shooting nodes. The solutions for both models were similar (Fig. **??**) highlighting the equivalence of the two rotation representations. Euler angles have the advantage to be easily interpretable, but they suffer from the loss of a DoF at the gimbal lock. The use of quaternion representation is advantageous for numerical stability when a joint is free to rotate on a wide three-dimensional range for motion.

C. Multiphase torque driven walking cycle

The goal was to estimate joint torques which are dynamically consistent during one gait cycle from the first heel strike to the end of the swing phase using a 3D one-leg torque driven model with 12 DoFs.

The experimental joint angles q_m (obtained from markers using an extended Kalman filter), ground reaction forces F_m and moments M_m were tracked:

[1] github link

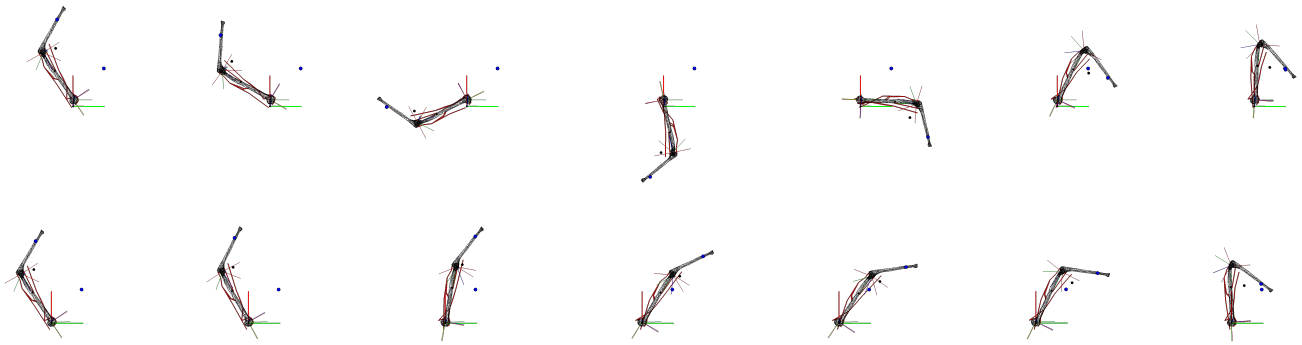

Fig. 3: Snapshots of an optimized muscle activation driven pointing task. Top: using ACADOS. Bottom: using IPOPT.

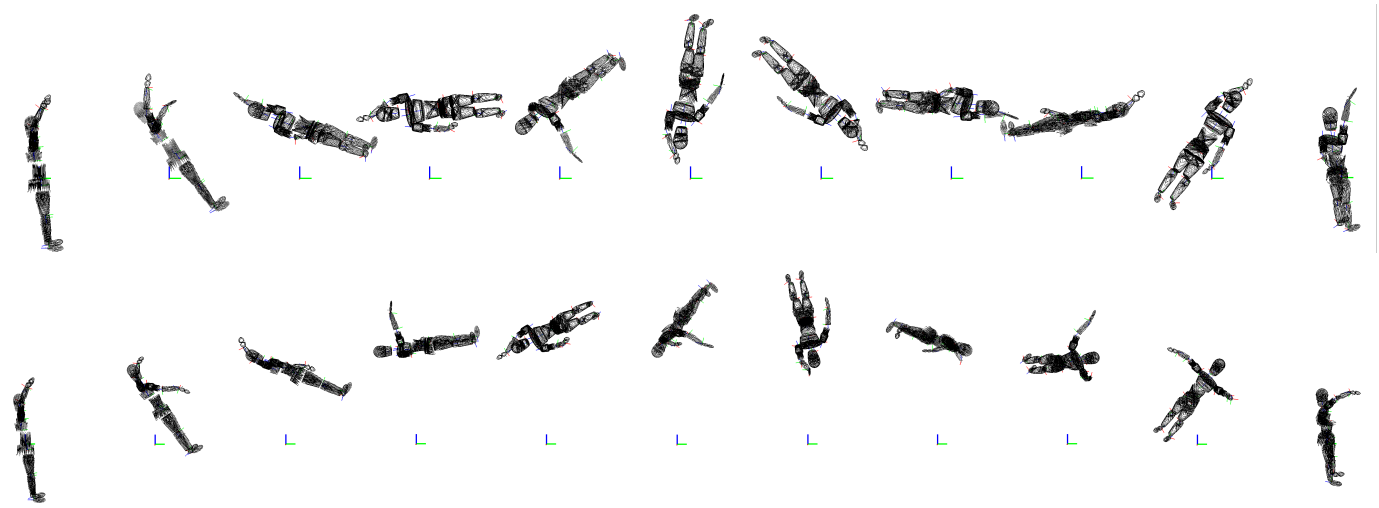

Fig. 4: Snapshots of a maximally twisting somersault driven by shoulder torque actuators and a free base expressed by Euler angles (top) or quaternions (bottom).

$$\mathcal{J} = \sum_{i=1}^{N_i} \Bigg(\underbrace{\omega_1(\|q_p - q_m\|^2)}_{TRACK\_STATE} \tag{4}$$

$$+ \underbrace{\omega_2(\| \sum_{c=1}^{N_c} F_c - F_m\|^2)}_{TRACK\_FORCES} \tag{5}$$

$$+ \underbrace{\omega_3(\| \sum_{c=1}^{N_c} OC \times F_c - M_{m,O}\|^2)}_{TRACK\_MOMENTS} \Bigg) \tag{6}$$

where N_i and N_c are the number of time frames and contact points, respectively.

The interaction between the ground and the foot was modelled using a 4-contact point model located at the heel and the forefoot (first and fifth metatarsi and toes - digit of the second toe). The stance phase was divided in three to follow the natural rolling movement of the foot from heel strike to toe off: heel, flatfoot and forefoot contacts. A constraint of non-slipping (NON_SLIPPING) and unilateral contact force (CONTACT_FORCE) were added for each stance phase. The use of the IMPACT state transition allowed to represent the gain of contact from a system without any contact (swing phase) to a system with contacts (heel strike) [ref thesis Felis - articles?].

Based on force plateform data and markers position, each phase had a definite time inducing a complete simulation time of 0.93 s and was discretized in 94 intervals. The solution was able to reproduce a complete gait cycle [value RMSE - better with 3 contact points?] (Fig. **??**).

D. Moving Horizon Estimation of Shoulder Elevation

The goal was to estimate in real-time joint kinematic and muscle activation using a moving horizon estimation (MHE). Therefore, muscle driven optimal control problem using a 4DoFs and 18 muscles arm model was built and split into a succession of smaller one. Thanks to the high similarity between successive problems, a warm-start strategy using previous solutions were implemented. Each OCP problems consisted to track articular angles (Eq. 3), minimize muscles controls, for controls regularization, and minimize states, for states regularization. At the end, only the first node of each

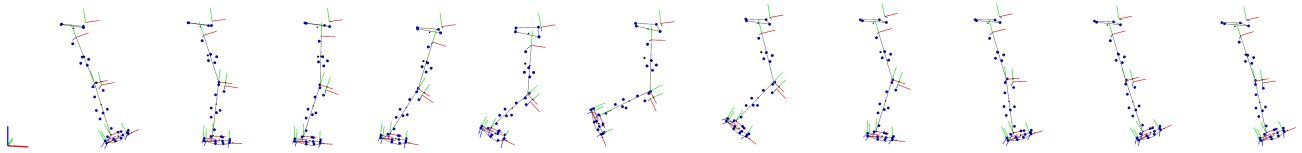

Fig. 5: Snapshots of a walking gait cycle driven by torque actuators.

estimation was kept to build the entirely solution.
A shoulder elevation movement of 8s on 800 frames was generated with co-contraction on the triceps/biceps muscles groups to test estimator. A windows size of 7 nodes which allows the estimator to run around 50Hz, four times faster than standard biofeedback (13Hz) was chosen. Whereas experimental data were generated at 100Hz, only one on two frames was sent to the estimator to correspond with experimental conditions. The estimator was able to forecast the movement kinematics (Fig. xx) with a consistent dynamic. As expected, the estimated muscles activations are lower than reference motion activation but with similar pattern (Fig. XX).

IV. Discussion

The objective of *Bioptim* is to solve a variety of biomechanical OCPs with minimal programming with high performance in terms of computational time and cost value. The main observation of the summary Table X of the six examples are: i) the ability to use torque-drive to excitation-driven models (and their combination) and dynamics with/without contact ii) a combination of cost functions and constraints iii) solve advanced OCPs in a few seconds or minutes, that were previously known to take hours. iv) easily switch from one NLP solver to another.

Through various examples, we highlighted key features of bioptim including

A. Utiliser des solveurs spécifiques et open source à DMS (Acados)

Bioptim takes advantage of several open source libraries to achieve a robust and/or fast convergence, especially CasADi combined with NLP solver namely ipopt or Acados, respectively. Whereas the choice of ipopt or Acados requires limited effort for the user, basic knowledge of both NLP solvers may help to determine appropriate weightings and best options. By comparison to previous studies, OCPs were solved much faster (e.g. XXXX s in [Colombe] vs XX s here;)

B. Différenciation algorithmique (ADOL-C dans Moco pour les collocations) … MX vs SX

C. DMS (privilégié dans biorbdoptim) vs direct collocation (link)

While the debate remains about the performance of direct collocations versus DMS, the development of bioptim was mainly oriented toward DMS, especially for the parallel computing. Nevertheless, existing collocations (Legendre) available in CasADi can be used and Acados proposes both explicit and implicit Runge-Kutta, the latter being a collocation approach. While implicit RK showed better convergence according to our own experience with Acados, explicit RK4 shows faster convergence in most of our implemented examples (those of the present paper and those available with biorbdoptim to present most of the features). In contrast to several papers [REFS], DMS was not a limitation to the performance (cost value and time to convergence) in our OCPs and is used in the most advanced real-time XXXX (Acados)

D. "Python based but fast" … biomécanique communauté d'utilisateurs

Since bioptim is written 100

Its performance is not affected by our Python architecture since the components of the OCP (i.e. continuity constraints which rely on the forward and muscle activation dynamics, paths constraints, Mayer and Lagrange costs) are all expressed as CasAdi trees for algorithmic differentiation and evaluation in C++ by the CasADi virtual machine.

Inspired by the real-time graphics from MUSCOD-II, biordbdoptim proposes a series of figures to analyze the solution at each iteration with minimal computational cost thanks to a XXX protocole. Other save and load options are valuable for post-processing analysis.

E. Custom en python et pas en C++

F. Fast resolution = multistart

Fast solvers offer the opportunity to use multistart on complex problems in order to circumvent the obstacle of local minima [**?**], [**?**] or to get meaningful initial solutions from simpler problems, for guiding the resolution of the sought problem.

G. Multiphase

H. Cost vs constraints … relâcher le problème simplement

I. Limitations

Bioptim is already a mature solution for solving OCP in biomechanics, however some limitations should be raised. First, it is based on the musculoskeletal package developed in our laboratory, namely, biorbd. While the strength of biorbd is to be fast, XXX, and CasADi-friendly for the algorithmic differentiation of most of the kinematic and dynamic functions [REF], it is not as advanced as OpenSim or Anybody in terms of biomechanical features. The few examples (section X) highlighted simple to advanced models. Currently, biorbd does not include a model builder with a GUI. Some Opensim models can be translated to biorbd's models using Python's functions [LINK] but our MSK library does not support multiple wrapping objects, non-orthogonal DoFs between two bodies, compliant contact force models

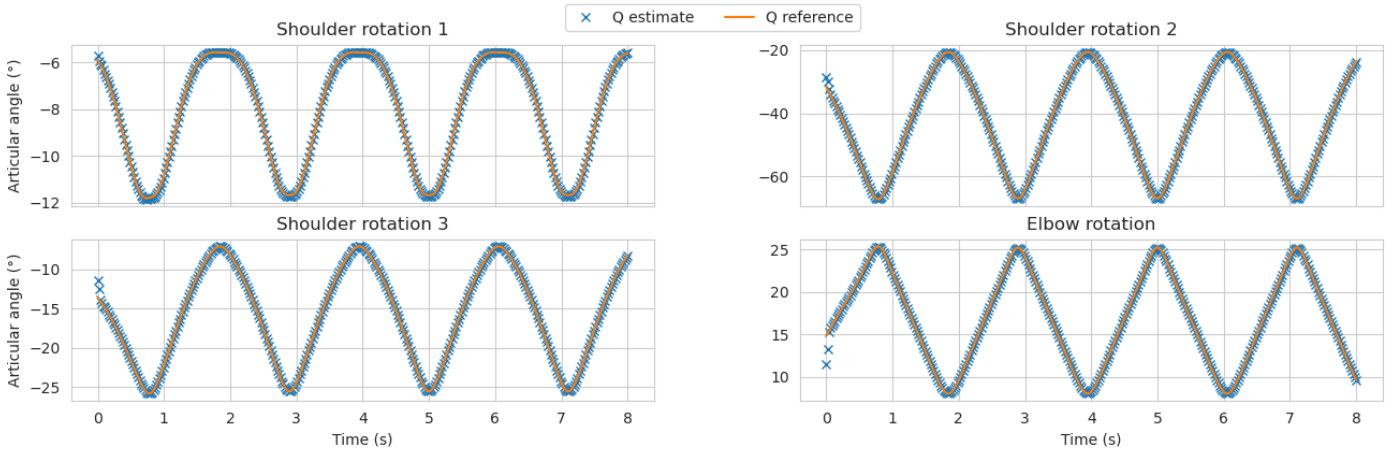

Fig. 6: Representation of estimate articlation angles (blue cross) and reference articulation angles (orange line).

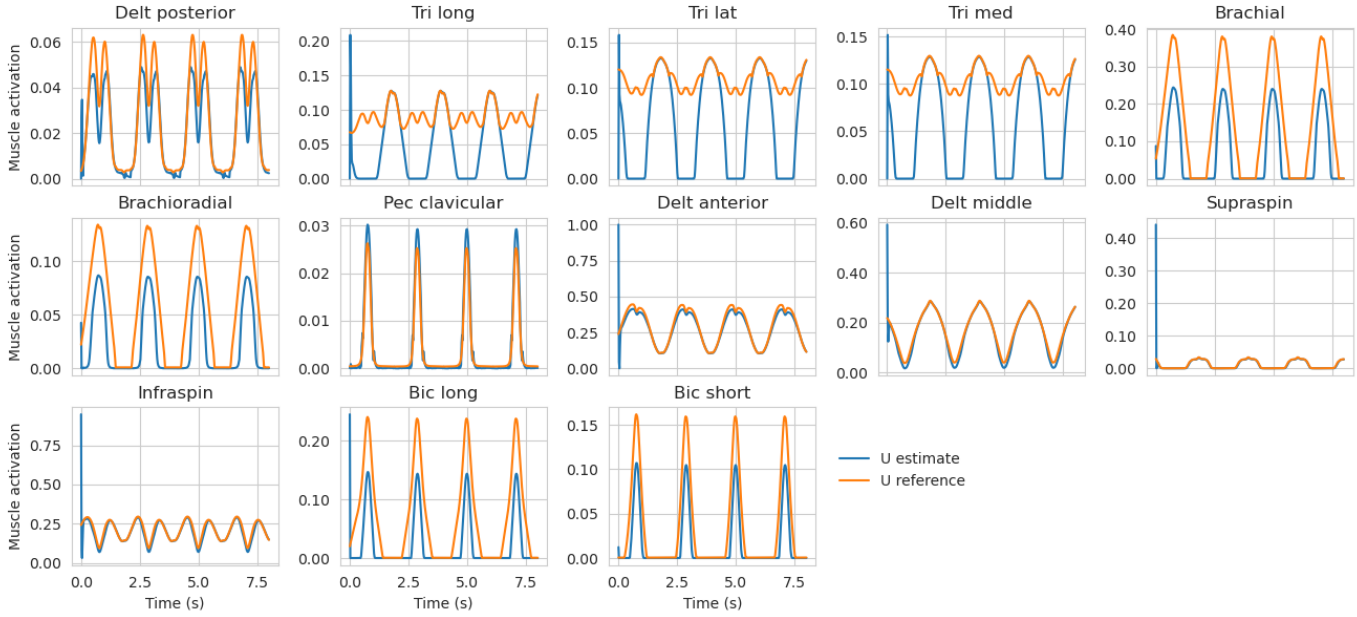

Fig. 7: Representation of estimate muscles activations (blue) and co-contracted muscles activations(orange) with significative action on motion (activation $>$ 1e-3).

TABLE I: Overview of computational results for the different OCPs cases and links to detailed implementations. * stands for free time OCP, otherwise it is fixed.

		Activation-driven pointing		Ex# 2		Ex# 3	
Setup	# states $\boldsymbol{x}(t)$	2		–	–	–	–
	# control $\boldsymbol{u}(t)$	8		–	–	–	–
	# shooting nodes	51		–	–	–	–
	OCP duration (s)	2		–	–	–	–
		IPOpt	ACADOS	IPOpt	ACADOS	IPOpt	ACADOS
Solve	# NLP iterations	27	21	–	–	–	–
	Optimized cost	6959.3	427.5	–	–	–	–
	Time to convergence (s)	9.9	0.19	–	–	–	–

(e.g. [SmoothSphereHalfSpaceForce [52]]) or muscle-tendon equilibrium yet. As seen in Mocco and other MSK models for OCPs, wrapping objects are rare due to computational cost and required optimization when a line of action is in contact with more than one object. Via points [REF] and pre-processed moment arms (to be expressed as polynomial

functions of crossed DoFs) are often preferred. In example X, the model differences (Opensim vs Biorbd), especially at the knee may explain the XXXX. Muscle-tendon equilibrium and model builder are already planned and the former will either required an additional optimisation procedure to achieve the equilibrium as in CEINMS [REF] or adding the length of muscle part with explicit constraints of XXXX in the OCP like in [REF]. Algorithms available in RBDL (core of biorbd for XXX) for ellipsoid foot model XXXXX (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6693511/).

A second limitation, which is to be adressed, is the fact that bioptim does not support multithreading in all conditions, e.g.: multiphase or multi-trial OCPs, Acados (despite its nfold faster convergence compared to Ipopt in multithread). Based on our experience about multithreading based on the architecture of bioptim (only the integration of the different shooting intervals are parallelized), the best advantage was found when the Hessian is calculated by algorithmic differentiation. Being the most costly part of the OCP, multithreading gives nearly linear reduction of the convergence time up to X threads. Such a gain is not found when the Hessian is updated using the BFGS quasi-Newton algorithm (i.e. the 'limited-memory' option in ipopt).

J. Future directions

Realtime estimation using MHE

MOOCP .. using front pareto

Prediction of adaptations due to muscular fatigue using NMPC

In line with the current studies of our research group, the future developments will first include nonlinear model predictive control and MHE (see example X) to predict optimal performances in repetitive tasks that generate muscular fatigue and real time estimation of joint torques and muscle forces, respectively. As shown in our previous studies [REFS] the analysis of a series near-optimal solutions is relevant in sport (but also in rehabilitation and ergonomics) and further efforts about multiobjective OCP of human performances are anticipated.

ACKNOWLEDGMENT

This study and the biorbdp library development was partly funded by scholarships of the Vanier program (BM) and the TransMedTech Institute XXX Apogée Canada (FB) and MB NSERC Discovey Programme (XXXX). Biorbdoptim acts as a catalyst in our group and several students contribute to this library; thank you to Amedeo, Ariane, André, Kilperic.

APPENDIX

The appendix