BiorbdOptim, a New Software 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

Many approaches coexist to discretize and solve OCPs. In sports biomechanics, joint angle-driven algorithms are XXXX in which the joint angle time histories are xxx by series of quintic polynomial functions [REF Yeadon, Begon] or quartic splines [REF Leboeuf, Huchez]. Whereas evolutionary algorithms

The number of papers about optimal control or dynamic optimisation is growing in robotics and biomechanics (Fig. X), probably because of increasing computer power but also the emergence of advanced open source (and proprietary) librairies for algorithmic differentiation and NLP solvers. Tools dedicated to OCP in biomechanics are rare. As shown by the recently launched MOCCO, four interrelated components are required: i) musculoskeletal modeling software (multibody kinematics and dynamics, muscle dynamics, etc.), ii) a method for automatic differentiation, iii) a discretization approach, and iv) a nonlinear programming (NLP) solver. Generic optimal control software (e.g. GPOPS-II [ref], Muscod-II [ref], Acado [ref]) provides solutions for component ii to iv but xxx. Since biomechanics is a community of software users [REFS], we believe that dedicated optimal control software will request a graphic user interface or at least a complete interface with a open source high-level language (e.g. Python) with a low-level core (C++) for efficiency.

When developing such software, we should consider that human movements are often multiphase (i.e. with different dynamics due to change in contact forces), [trouver d'autres], and models should be personalized, which may require several trials and some parameters' identification. Moreover, in contrast to the inverse flow which relies on measures, solving the ordinary differential equations (ODEs) of motion may result in unanticipated behaviours , ranging from non-physiological states to singularities. Either convenient constraints' definition and XXXXX,

Lifting and relaxing OCPs, DMS et DC Constraints vs cost

While CasADi is used in MOCCO mainly for its interface to ipopt (ADOL-C being used for automatic differentiation), this tool was first and foremost designed for algorithmic differentiation and is consequently widely used for NLP to reduce the cost and increase the accuracy of gradient and Hessian compared to finite-difference method. Acados, a recent NLP solver dedicated to DMS, was recently launched by the same research group as CasADi taking advantage of the algorithmic differentiation for real-time applications. Some applications, such as the real-time estimation of muscle forces, presently solved by inverse approach [REF] (from inverse dynamics to static optimization) or hybrid approach like the EMG-assisted algorithm in CEINMS [REF], become possible [citer article François-Amedeo]

The objective of the present paper is to introduce an innovative optimal control software for OCPs in biomechanics with the following features: Written in Python with Real-time capability

The paper is organized as follow:

II. Design and Implementation

The method

III. Examples

[a] Laboratoire de Simulation et Modélisation du Mouvement, Faculté de Médecine, Université de Montréal, Laval, QC, Canada

[b] other, elsewhere

[*] author#1@umontreal.ca

In this section, some applications are presented to illustrate the versatility of BiorbdOptim and to give a practical overview on how to use its various features. The performances and the Github links of each OCP are listed in Tab. II.

A. *Muscle activation driven pointing task*

The goal of this example was to achieve a muscle activation driven pointing task using a 2-DoF, 6-muscle arm model. In addition to muscle-induced torques, pure torques could compensate for the model weaknesses. The movement lasted for 2 seconds and was discretized using 51 shooting nodes.

Term #1 of the objective function (Tab. I) 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. Terms #2 and #3 were added for control regularization (muscle activation and torques) and #4 for state regularization.

TABLE I: Objective terms of the activation-driven pointing task

	Type	Function	Weight
#1	Mayer	ALIGN_ MARKERS	$1e6$
#2	Lagrange	MINIMIZE_ MUSCLE_ CONTROL	$1e1$
#3	Lagrange	MINIMIZE_ TORQUE	$1e1$
#4	Lagrange	MINIMIZE_ STATE	$1e1$

The problem was solved with IPOPT and ACADOS resulting in two significantly different solutions with ACADOS proving a 16 times smaller optimized cost (Tab. II), which illustrate the pitfalls of local minima as well as the benefits of having access to different solvers with minimal effort. Indeed, the ACADOS-based solution (Fig. 1, top) makes good use of gravity to minimize the control inputs, while the IPOPT-based solution (Fig. 1, bottom) moved the arm in the opposite direction and was stuck in a local minimum (still achieving the task though). It is worth noting that 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:

$$\mathcal{J} = - \underbrace{\int_0^T \dot{\phi} dt}_{MINIMIZE\_TWIST} + 1 \times 10^{-6} \underbrace{\sum_{i=1}^{2} \int_0^T \tau_i^2 dt}_{MINIMIZE\_TORQUE} , \quad (1)$$

where T in the duration of the movement and τ_i is the torque control of the i^{th} arm DoF. The first term of the objective function (Eq. 1) 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.

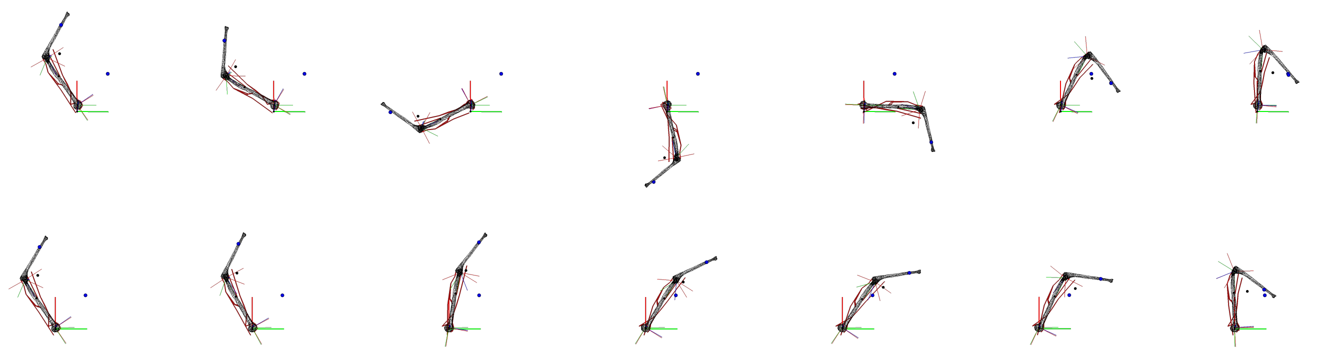

Fig. 1: Snapshots of an optimized muscle activation driven pointing task. Top: using ACADOS. Bottom: using IPOPT.

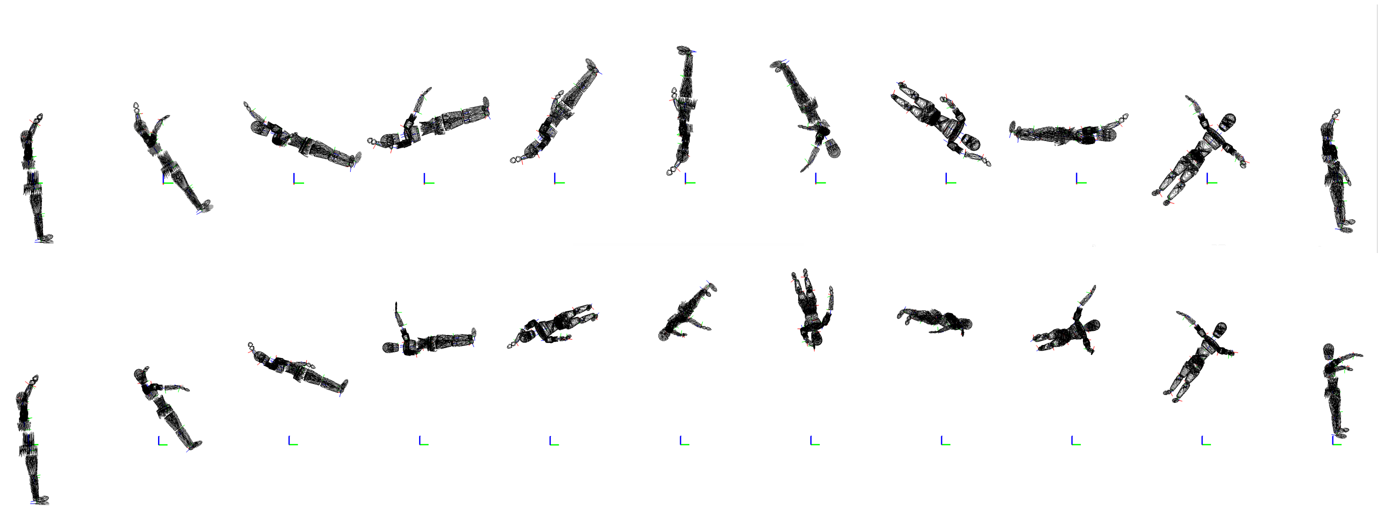

Fig. 2: Snapshots of a maximally twisting somersault driven by torque actuation. Top: Euler angles. Bottom: quaternion.

TABLE II: Overview of computational results for the different OCPs cases and links to detailed implementations. When running with IPOpt, 6 threads were systematically used. * 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	–	–	–	–

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. Multiphase

G. Cost vs constraints ... relâcher le problème simplement

H. 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 (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).

I. *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