Bioptim, a Python interface for Musculoskeletal Optimal Control in Biomechanics

Benjamin Michaud[a,*,†], François Bailly[a,†], Amedeo Ceglia[a], Eve Charbonneau[a], Léa Sanchez[a] and Mickael Begon[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. motion duration, energy expenditure, matching experimental data, etc.). Up to very recently, there was no off-the-shelf software available to the community to quickly formulate and solve such musculoskeletal OCPs. Consequently, researchers had to develop their—often simplified—own solutions, with little or no dissemination to the community, limiting knowledge and expertise sharing.

As a result, many approaches coexist to formulate and solve OCPs in the biomechanical 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

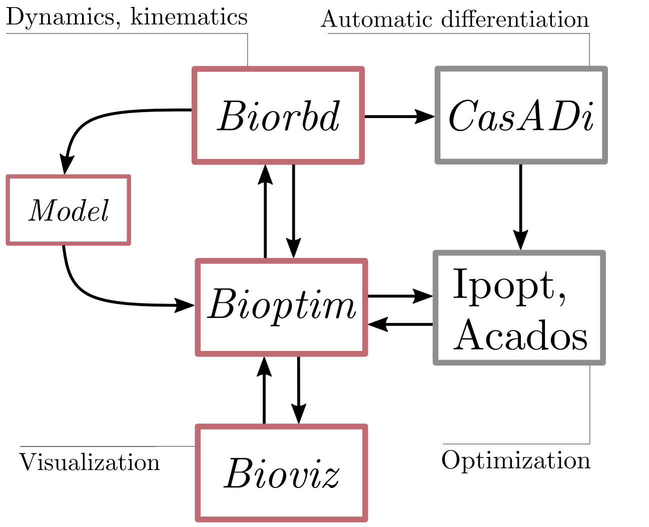

Fig. 1: *Bioptim* dependencies flowchart. The red-boxed software are developed by the S2M team. The *Bioptim* part is further detailed in Fig. 2.

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, other 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, Mombaur, 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].

Concerning the non-linear solver, a variety of software exist and have been used to solve transcribed musculoskeletal NLPs.

[†] These authors have contributed equally to this work and share first authorship.

[a] Laboratoire de Simulation et Modélisation du Mouvement, Faculté de Médecine, Université de Montréal, Laval, QC, Canada
[*] BENJAM@umontreal.ca

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 finite differences (inaccurate thus comprising convergence) or computed automatically using 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 arbitrarily precise ODE solvers for the integration, which is fully parallelized for more efficiency. *Bioptim's* core is fully written in *Casadi* symbolics 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 *Bioptim* 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.

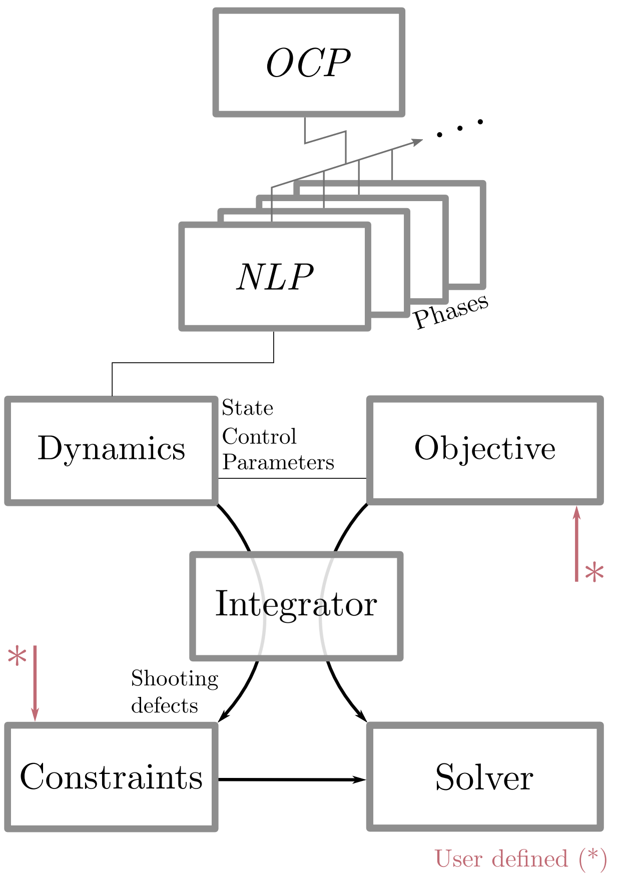

Fig. 2: *Bioptim* design flowchart.

II. Implementation and Design

A. Implementation and dependencies

Bioptim is the top layer of a succession of dependencies (*Biorbd*: dynamics and MSK modeling; *Casadi*: automatic differentiation; *Ipopt/Acados*: optimization; *Bioviz*: visualization). Within this software suite, *Bioptim*'s main role is to shape the problem to allow its dependencies to communicate efficiently, while providing an intuitive and flexible interface to the user (Fig. 2). Therefore, it was written in Python for its flexibility and its widespread use among researchers. However, all intensive calculations behind the interface are performed in C/C++, keeping *Bioptim* both fast and easy to customize.

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 markers, the actuators of the model (muscles and joint torques possibly with 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 or joint torques, the states can be muscle activations and/or joint kinematics. They can include contact points, external forces, etc. Even if these dynamics types exhaustively span the current usages in biomechanics, a custom dynamics type is also pre-implemented to easily customize problems.

2) Objective functions: In line with 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 states / controls / markers / contact forces / problem duration, etc. Should one go missing, a custom objective type is also possible to define.

When declaring the desired list of objective functions for a given NLP, each objective function type is associated with a weight, and the user can 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: Classically, constraints are hard penalties of the optimization problem, i.e., a solution will not be considered optimal, unless every constraint (equality or inequality) is met. In *Bioptim*, there exist two classes of constraints: the `Bounds` and the `Constraint`. Essentially, the `Bounds` are constraints directly related to the states and the controls. They are useful for defining model-related constraints such as kinematic, torque or muscle excitation / activation limits. The `Constraint` class contains a variety of already implemented constraints. Among them, the user will find the constraint equivalents of every objective function in addition to other ones, commonly useful in biomechanical problems (e.g. non-slipping contact point, non-linear bounds on torque depending on the state, etc.). Should one go missing, a custom constraint type is also possible to define.

[1]github link

III. Examples

In this section, six 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. I.

A. Muscle activation driven pointing task

In this first example, 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 were added to compensate for the model weaknesses. The first three Lagrange terms of the objective function (Eq. 2) were added for control regularization (muscle activation a and joint torques τ) and for state (x) regularization. The last Mayer term corresponds to the pointing tasks at the final node, to superimpose two markers, the first one, m_u, fixed in the ulna system of coordinates and the second one, m_s^*, fixed in the scene. :

$$\begin{aligned} \mathcal{C} = \int_{t=0}^{T} \underbrace{\|a\|^2}_{\texttt{MIN\_ACTIVATION}} + \underbrace{\|\tau\|^2}_{\texttt{MIN\_TORQUE}} + \underbrace{\|x\|^2}_{\texttt{MIN\_STATE}} \; dt, \\ + \omega_1 \underbrace{\|m_u - m_s^*\|^2}_{\texttt{TRACK\_MARKERS}} \end{aligned} \tag{2}$$

where T is the duration of the motion, and $\omega_1 = 1e5$. The movement lasted for 2 seconds and was discretized using 50 shooting nodes. The problem was solved using *Ipopt* and *Acados* resulting in two significantly different solutions. *Acados* provided a 16 times smaller optimized objective (Tab. I), 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. 3, top) makes good use of gravity to minimize the control inputs, while the *Ipopt*-based solution (Fig. 3, 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

In this example of acrobatic sports biomechanics, the goal was to maximize the twist rotation (ϕ) of a 8-DoF model in a backward somersault and to illustrate *bioptim*'s ability to handle quaternionic representations of rotations. The model was composed of a 6-DoF root segment and two 1-DoF torque actuated shoulder joints. Two different numerical descriptions of the root segment rotations were used (Euler angles and quaternions). The objective function was as follows:

$$\mathcal{J} = \int_0^T \underbrace{\omega_1 \dot{\phi}}_{\texttt{MIN\_TWIST}} + \underbrace{\omega_2 \|\boldsymbol{\tau}\|^2}_{\texttt{MIN\_TORQUE}} \; dt, \tag{3}$$

with $\omega_1 = -1$ (resulting in the maximization of the first term) and $\omega_2 = 1e-6$, T is the duration of the movement and $\boldsymbol{\tau}$ is the torque control vector. The first term of the objective

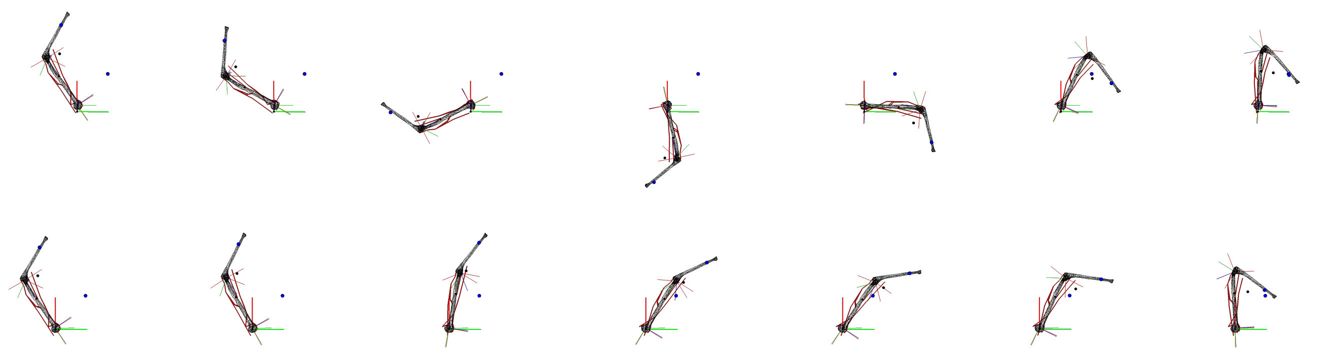

Fig. 3: Snapshots of an optimized muscle activation driven pointing task. Top: using ACADOS. Bottom: using IPOPT.

function (Eq. 3) corresponds to maximizing the change in twist rotation and the second term is for control regularization.

The movement lasted for approximately 1 second and was discretized with 80 shooting nodes. The optimal kinematics were different for the two types of models (Fig. 4) because of the presence of local minima. However, both models take profits of a common biomechanical strategy (i.e. tilting the body to bring closer together the twist axis and the angular momentum vector) 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 (leading to numerical instabilities). The use of a quaternion-based representation tackles this numerical stability issue when a joint is free to rotate on a three-dimensional range of motion. Quaternion's integration must be handled with care [**?**], indeed, when representing orientations, quaternions must be unitary and thus belong to a constrained manifold (namely, the unit 3-sphere S^3). However, classical numerical integration schemes such as Runge–Kutta methods treat unit quaternions as if they were arbitrarily defined in $\mathbb{R}^4$. This was taken care of in *bioptim* by performing a normalization step after each Runge–Kutta iteration to project non-unitary quaternions onto S^3.

C. *Pendulum on a spring*

This spring-mass-pendulum-based example is presented to introduce *bioptim*'s ability to use of external forces. The goal was to maintain the position of a $1\,\text{kg}$ mass hanging on a linear spring attached to the ground. A 0.2 m-long pendulum weighting $10\,\text{kg}$ was attached to the mass and free to rotate in one dimension (Fig. 5). In addition to the spring force, the mass was actuated by a vertical external force (e.g., something pulling on it) while the pendulum rotation was passive. The system therefore comprised two DoFs, the mass position (q_m) and the pendulum angle (q_p) and one control input, the vertical external force pulling on the mass ($\mathcal{F}$). The spring force $\mathcal{F}_s$ was:

$$\mathcal{F}_s = -k * q_m, \tag{4}$$

with k the spring stiffness constant.
The OCP was composed of two phases each lasting for $5\,\text{s}$, with 50 shooting nodes. In the first phase, no objective function was minimized and $\mathcal{F}$ was constrained to be 0, letting the mass oscillating freely. Then, in the second phase, a cost function (Eq. 5) was minimized, to enforce a reference position q_m^* of the mass. This objective function, exclusively composed of Lagrange terms, was formulated as follows:

$$\mathcal{J} = \int_{T/2}^{T} \underbrace{(q_m - q_m^*)^2}_{\texttt{TRACK\_STATE}} + \omega_1 \underbrace{\mathcal{F}^2\ dt}_{\texttt{MIN\_TORQUE}} \quad dt, \tag{5}$$

with $q_m^* = -0.5m$ and $\omega_1 = 1e-6$ and T is the duration of the movement. The first term of the objective function (Eq. 5) acts as a position controller for the mass. The second was added for control regularization.

During the first phase, the mass is passively oscillating around its stationary position due to the spring force (Fig. 6). At the beginning of the second phase, when an additional external force acts on the mass, it stabilizes around the targeted position. This example highlights the possibility of using optimal control to find activation patterns compensating for external passive forces (e.g., ortheses flexibility, contact surface deformation, interaction between two models, etc.).

D. *Multiphase torque driven walking cycle*

This example is presented to introduce *bioptim*'s ability to deal with a multiphase locomotion estimation problem with muscle actuation and contact forces. The goal was to estimate muscles activation by tracking markers trajectories and ground reaction forces and moments. The model was a 3D leg with 12 DoFs (6-DoFs pelvis, 3-DoFs hip, 1-DoF knee and 2-DoFs ankle), driven by 17 muscle activations and residual joint torques to compensate for potential muscle actuation weaknesses. The gait cycle was defined from the first heel strike to the end of the swing phase discretized into 90 shooting intervals. To follow the natural rolling of the foot, the stance was divided into three phases (heel, flatfoot and forefoot contacts) of fixed duration deduced from experimental force platform data and markers position (0.05, 0.36 and 0.16 s). The swing phase lasted 0.38 s. The interaction between the ground and the foot was modeled using a 4-contact points model located at the heel and the forefoot (first, fifth metatarsi and hallux). The optimization problem consisted in minimizing the errors between predicted $\mathbf{m}$ and reference $\mathbf{m}^*$ markers trajectories, predicted $\boldsymbol{\mathcal{F}}$, $\boldsymbol{\mathcal{M}}$ and reference $\boldsymbol{\mathcal{F}}^*$, $\boldsymbol{\mathcal{M}}^*$, respectively ground reaction forces and moments at all

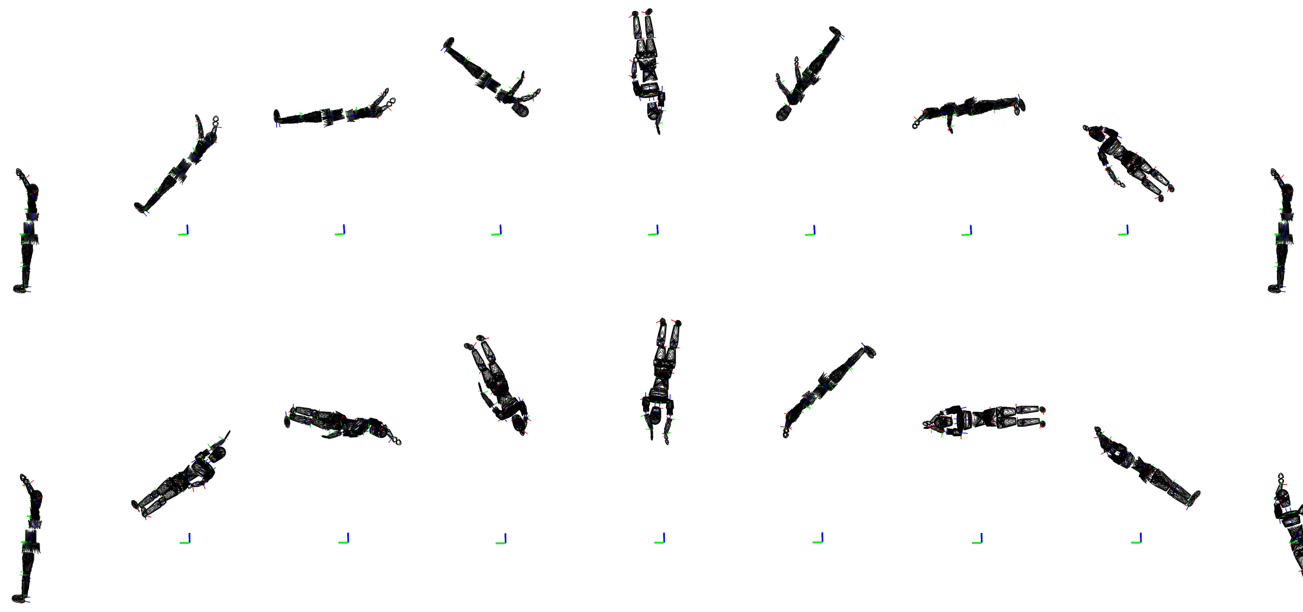
Fig. 4: Snapshots of maximally twisting somersaults driven by shoulder torque actuators and a free base whose rotation is either expressed by Euler angles (top) or by quaternions (bottom).

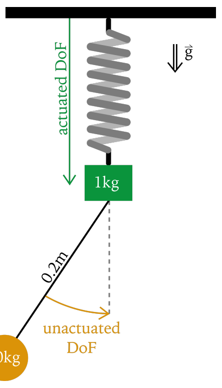

Fig. 5: Definition of the spring-mass-pendulum model.

contact points. * stands for reference (i.e., measured) data. A regularization term on muscle activations, $\mathbf{a}$, was also added (least-activations) as well as a penalization term on the residual torques $\boldsymbol{\tau}$:

$$\mathcal{J} = \int_{t=0}^{T} \underbrace{\omega_1(\|\boldsymbol{m} - \boldsymbol{m}^*\|^2)}_{\texttt{TRACK\_MARKERS}} + \underbrace{\omega_2(\|\boldsymbol{\mathcal{F}} - \boldsymbol{\mathcal{F}}^*\|^2)}_{\texttt{TRACK\_FORCES}} + \underbrace{\omega_3(\|\boldsymbol{\mathcal{M}} - \boldsymbol{\mathcal{M}}^*\|^2)}_{\texttt{TRACK\_MOMENTS}} + \underbrace{\omega_4\|\mathbf{a}\|^2}_{\texttt{MIN\_ACTIVATION}} + \underbrace{\|\boldsymbol{\tau}\|^2}_{\texttt{MIN\_TORQUE}} \; dt, \tag{6}$$

where ω_1=1e5, ω_2=0.1, ω_3=0.1, ω_4=10 are weighting factors and T is the the duration of the current phase.

Non-slipping (NON_SLIPPING) and unilateral contact force (CONTACT_FORCE) constraints were added to prevent the foot from slipping and pulling from the ground. In between phases, the use of the IMPACT state transition allowed to represent the gain or loss of contact(s) in the dynamics (e.g., swing phase to heel strike [thesis Felis - articles?])

Fig. 7 shows the leg motion during the walking cycle. The root mean square tracking error on markers trajectories was 0.027 m (mean error on pelvic and foot markers was 0.0075 m and 0.0147 m, respectively). Concerning ground reaction forces tracking, the root mean square error was 4.85 N. Fig. 8 shows muscle activity patterns during the gait cycle. Gluteal muscles were only activated during the stance phase. The Gluteus Maximus was mainly activated during the early flatfoot phase with a maximal activation of 0.2 allowing to prepare the stance phase. For the thigh muscles, the semimembranous, semitendinous and biceps femoris were activated during the early stance phase and terminal swing (maximal activation of 0.3, 0.4 and 0.4, respectively) which is coherent with the activity patterns recorded from Winter (1991) [Winter DA (1991). The Biomechanics and Motor Control of Human Gait: Normal, Elderly and Pathological.]. The knee extensors, vastus lateralis, medialis and intermedius followed the same pattern and were mainly activated during the flatfoot phase (ie loading response). Chang et al. (2007) observed a significant pre-swing and initial swing activity for those muscles which was found a bit earlier in this simulation (terminal stance). However, while they observed highest activity of the rectus femoris at late swing and loading response (early flatfoot phase), the maximal activation appeared at forefoot phase (terminal stance) and pre swing. [Chang et al (2007). https://doi.org/10.1016/j.jelekin.2006.02.003]["Five basic muscle activation patterns account for muscle activity during human locomotion" Ivanko et al. (2004)] Leg muscles were highly activated (saturation of gastocnemius lateralis and medialis) at the terminal stance and early swing. The tibialis anterior is the sole muscle for dorsiflexion and the

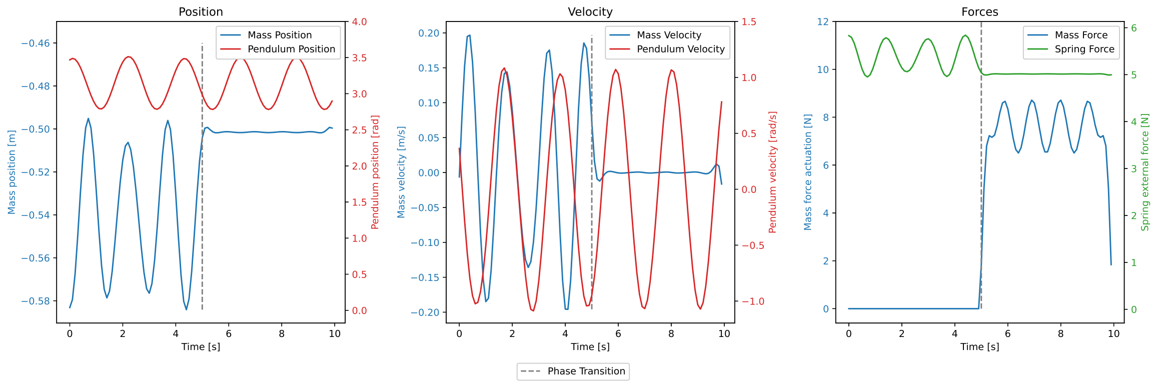

Fig. 6: Two-phases kinematics of the mass-pendulum-spring system. Gray dashed lines show the phase transition, blue lines are related to the mass (position velocity and external force acting on it), red lines are related to the pendulum (position and velocity) and the green line depicts the spring force.

only antagonist of the triceps surae in this model which could explain its high activity at the same time.

E. Moving Horizon Estimation of Shoulder Elevation

This example is presented to introduce *bioptim*'s ability to provide real-time estimation of biomechanical variables. The goal was to perform a real-time estimation of dynamically consistent joint kinematics and muscle forces, using a moving horizon estimation (MHE). A shoulder elevation motion was performed with a 4-DoFs (q) arm actuated by 19 Hill-type muscle elements. The control inputs of the model were the muscle activations (a). The MHE implementation consists in splitting the OCP into a succession of smaller one for processing fixed-size subsets of the tracking data moving forward in time. Each time one subproblem is solved, a new measurement is acquired, the oldest one is discarded and a new subproblem is defined. Due to their similarities, the solution of the previous OCP is a good initial guess to the new one. The dynamical consistency of the final solution is enforced by continuity constraints on the initial state. Each objective function (Eq. 7) was written as the sum of three terms: tracking reference joint angles (q^*), states and muscle activations regularizations (i.e., least-square criteria):

$$\mathcal{J} = \int_t^{t+t_{mhe}} \underbrace{\omega_1(\|q - q^*\|^2)}_{\texttt{TRACK\_STATE}} + \underbrace{\omega_2\|q\|^2}_{\texttt{MIN\_STATE}} + \underbrace{\omega_3\|a\|^2}_{\texttt{MIN\_ACTIVATION}} \; dt, \qquad (7)$$

where $\omega_1 = 1e5$, $\omega_2 = 100$, $\omega_3 = 1e4$ and t_{mhe} is duration of each sub-problem.

In this example, reference data of an 8 s series of four arm elevations were generated at 100 Hz, by computer simulation. Centered Gaussian noise (mean = 0, std = $0.005q^*(t)$) was added to q^*, to simulate experimental-like joints angle measurements. Using a windows size of 7 nodes, the estimator ran at about 33 Hz (one in three reference data frame was sent to the estimator to simulate experimental-like conditions), i.e., two and half times faster than standard biofeedback (13 Hz, [**?**]). The MHE was able to forecast the movement kinematics with a root mean square error of 1.33 ± 0.52 ° (Fig. 9) and to provide a realistic estimation of muscle forces close to the ground truth with a root mean square error of 10.6 ± 13.6 N (Fig. 10, only the muscles with significative action (peaks force $>$ 15 N) are represented).

F. Multiphase jumper

The main goal of the jumper is (without much of a surprise) to jump as high as possible. This example was designed to introduce the *bioptim*'s ability to reduce the number of degrees-of-freedom (DoF) of a model via the mappings, to include nonlinear boundaries on the controls, and to solve complex multiphase programs that include impacts and free time phases.

The model used is a full body model consisting of 3 DoF at the pelvis (the forward and upward translation, and the tranverse rotation), 2 DoF at each upper limb (shoulder flexion and abduction), and 3 DoF at each lower limb (hip, knee and ankle flexion) for a total of 13 DoF. For this optimization, the *BidirectionalMapping* is used to ignore the shoulder abduction—by setting the mapping to *None*—and to symmetrize the left-hand and right-hand sides, effectively creating a 7 DoF pseudo-2D model. Since this is a full body model, the root segment (i.e., the pelvis) is left uncontrolled, reducing the number of control variables to 4, namely the shoulder, hip, knee and ankle flexions.

A total of five phases are used to describe the dynamics of the jump and landing. The first two consists in a phase with two (i.e., heel and toe) and one (i.e., toe) contacts on the ground, respectively. A contact is described as a point where forces can be applied to nullify the accelerations. The aerial phase is free-fall dynamics, that is a dynamics without contact points. The last two phases (i.e., the reception phase) are the opposite of the push off phase, i.e., a phase with one contact point to the ground (i.e., toe) and then a phase with two contact points (i.e., heel and toe). The transitions between the phases with less contact points toward more contact points are approximated by the build-in inelastic impact *PhaseTransition*, which recomputes the segment velocities post impact.

The main objective function is a Mayer objective computed

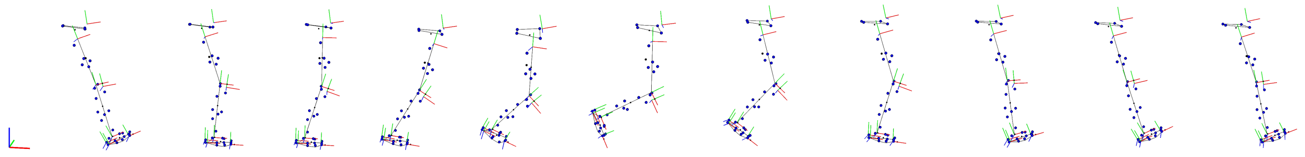
Fig. 7: Snapshots of a walking gait cycle driven by muscles activation.

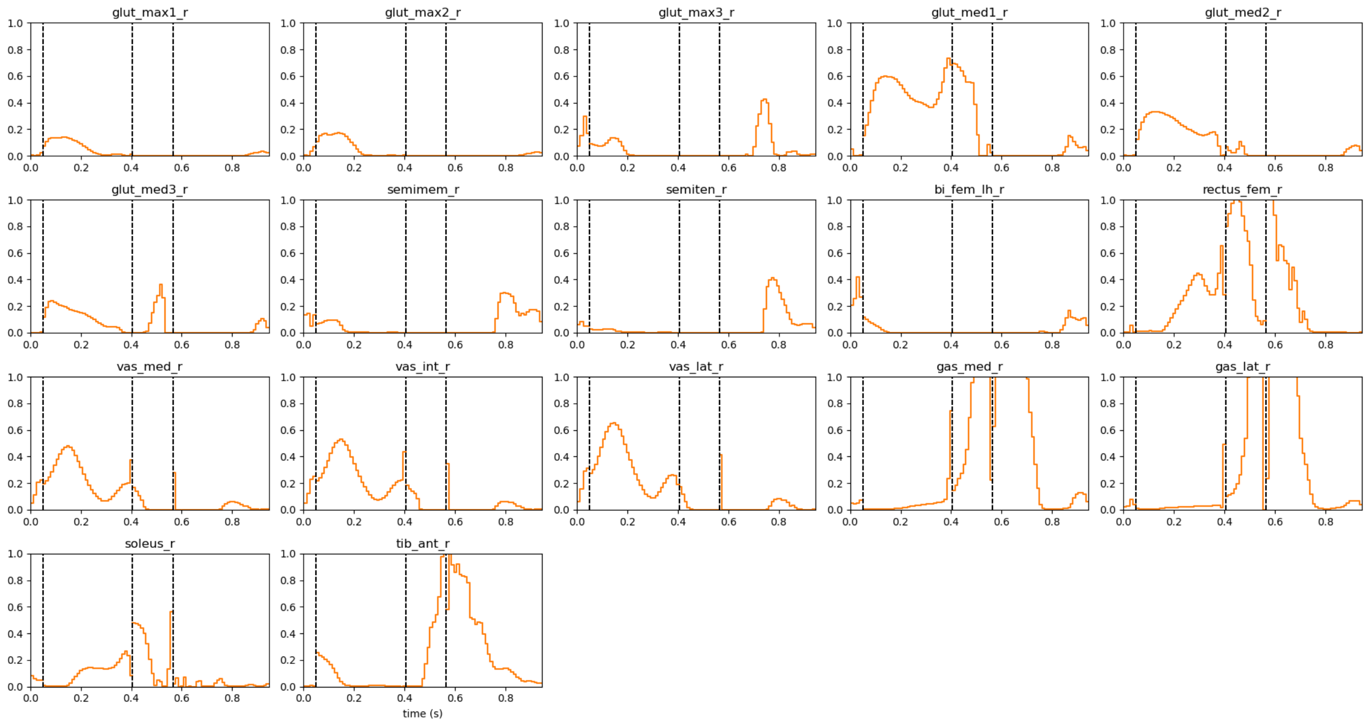

Fig. 8: Muscle activity patterns during walking cycle.

at the end of the pushoff phase consisting in maximizing the jumps height (h) such that:

$$h = \frac{\dot{CoM}_z^2}{-2g} + CoM_z$$

where CoM_z and $\dot{CoM}_z$ are the center of mass position and velocity, respectively, and g is the gravitional acceleration constant ($-9.81\,\mathrm{m/s^2}$). Since 'bioptim' minimizes objective, instead of maximizing them, a negative weight is applied to the function. Two additional objective are added to facilitate the convergence of the solver. The first is a minimization of the derivative of the velocities, preventing from large movements during the aerial and reception phases. The second is to minimize the each phase time. Both of these objective have a small weight—four order of magnitude smaller than jump height—so they are not prioritized. Finally, the end (i.e., the last node of the last phase) pose and velocities of the model are a Mayer objective function so that the model is more or less static, standing knee flexed with its arm horizontally raised.

$\mathcal{J} = XXX.$

Several constraints are necessary to describe a realistic jump. The generalized coordinates—i.e., the maximum flexibility expected—are bounded to human-like limits. The generalized velocities are arbitrarily bounded to $[-10\pi; 10pi]$. The genelized forces are bounded by the torque/position/velocity relashionship measured on a high level athlete using an isokinetic dynamometer (Fig. 11). For the contact forces from the contact points, a directional constraint is applied such that the force is alway pointing upward, meaning that the model is not allowed to pull on the floor. The lateral force norm is constrained to be below the half of the upward force, i.e., the model remain in a cone of friction more or less corresponding in a normal surface contacting shoes. The time of the phases were constraints as such Finally, some constraint were added to prevent the gradient descending solve from exploring non interesting ideas. A first one is that during the push off and reception phases, the heel must remain over the floor. A second is that the center of mass velocity must point upward when leaving the floor, and finally, at the same instant, the arms must be frontward.

To accelerate solving the problem using *ipopt*, the problem was first approximately solved using the BFGS hessian approximation for 200 iterations maximum. Then, the solution was re-optimized, from that solution, with an exact-hessian computations for up to 1000 iterations.

The optimized solution of a $1.35\,\mathrm{m}$ jump height was

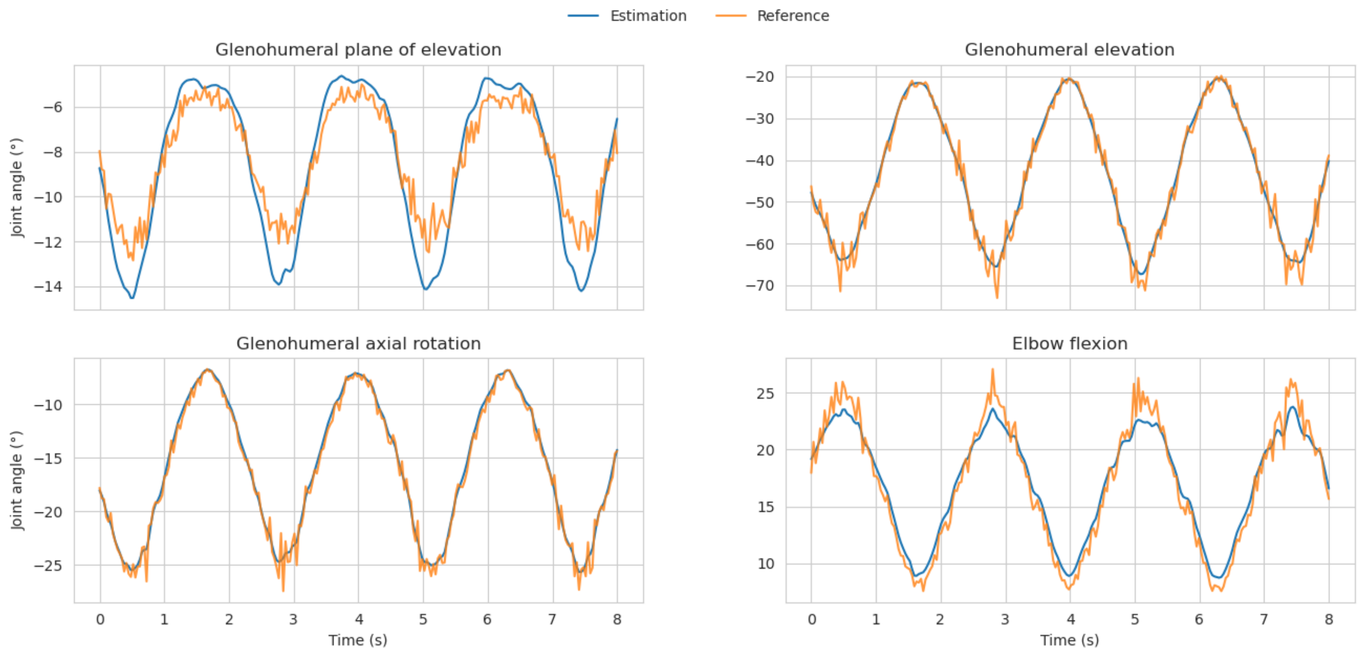

Fig. 9: Time series of estimated joint angles (blue) and noisy reference joint angles (orange).

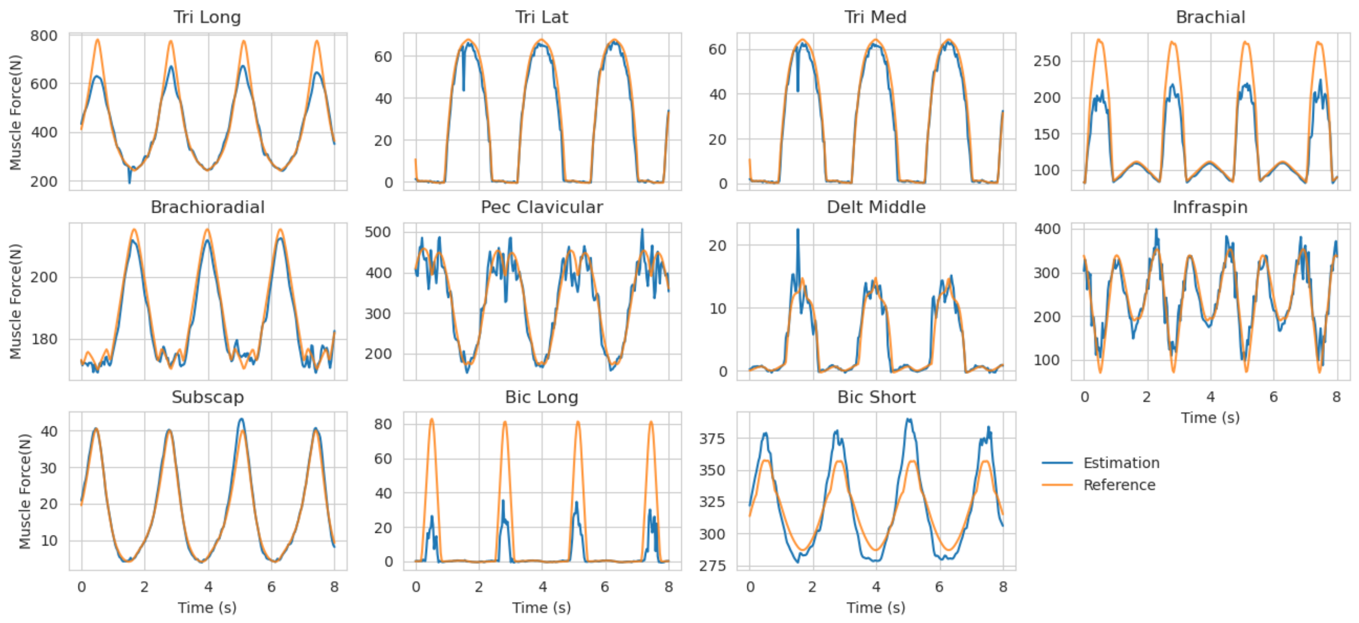

Fig. 10: Time series of estimated muscle forces (blue) and ground truth muscle forces (orange). Only the muscles with significative action (peaks force > 15 N) are represented. Muscle abbreviations stand for (from left to right and top to bottom): Triceps Long head, Lateral and Medial, Brachial, Brachioradialis, Deltoid Anterior and Middle, Infraspinatus, Subscapularis, Biceps Brachial Long and Short head.

obtained in $0\,$s. The solution reproduced a human proximo-distal strategy, that is activating large segments first (for instance the torso) and adding sequentially the more distal segments, the most distal—and the consequently the last to activate—being the ankle.

IV. Discussion

The purpose of *bioptim* is to solve a variety of biomechanical OCPs with both minimal user effort and high performances in terms of computational time. The main

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 $\boldsymbol{x}(t)$	4		–	–	–	–
	# control $\boldsymbol{u}(t)$	6		–	–	–	–
	# 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	–	–	–	–

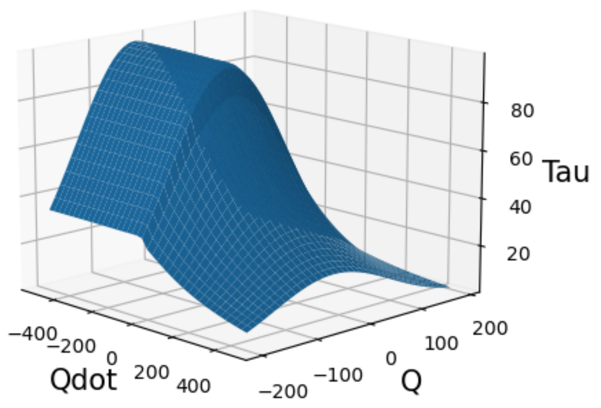

Fig. 11: Surface representing the nonlinear constraint for torque/position/velocity relationship

features illustrated by the six provided examples are (Tab. X):

- the possibility to use torque-, activation- or excitation-driven models (and their combinations)
- a variety of ready-to-use cost functions, constraints and dynamics (with and without contacts)...
- ... easily customizable in Python when required by the user;
- the possibility to solve advanced OCPs (possibly multi-phased) in a few seconds or minutes, that were previously taking hours;
- the interface with two different NLP solvers

In the following, several aspects of *bioptim* are discussed.

A. Direct multiple shooting-based

While the debate remains about the performances of direct collocations versus DMS [?][donner d'autres favorables a DC], the development of *bioptim* was oriented toward DMS, because: *i)* it allows to select effortlessly an arbitrary accuracy for the integration (e.g., order and numbers of RK steps); *ii)* it allows to use DMS-based fast NLP solvers such as *Acados*. Concerning the integration, either internally or via *Acados*, several schemes are implemented in *bioptim* (RK4, RK8, IRK). While IRK showed better convergence in our experience with hard problems in *Acados*, in most of the cases, RK4 showed to be a good speed/robustness trade-off. In contrast to what is claimed in [REF], DMS is not a limitation to the performances (cost value and time to convergence), since, in our experience, the performances of *bioptim* often outperform state-of-the-art results.

B. Automatic differentiation

One of the reasons explaining the performances of *Bioptim* is the rewriting of the core software, *RBDL* and *biorbd* implementing the dynamics, into *casadi* symbolics to automatically provide the exact Jacobians and Hessians of the resulting NLP. This feature is somewhat more computationally expensive than finite differences, but the gain in accuracy for the calculation of derivatives often leads to shorter convergence times (due to much less iterations) and to optimal solutions reached with lower tolerances. This last aspect must be emphasized for complex motions (fast, highly dynamics ones), because talking about *Ipopt* for instance, an optimal solution obtained with a convergence criterion of $1e{-}2$ is very unlikely to be dynamically consistent; it would diverge when forward integrating the controls in a single-shooting manner. By experience, a lower tolerance ($1e{-}6$. $1e{-}8$) only reachable with exact derivatives could lead to better forward dynamics results.

C. Python based but fast!

Bioptim was thought as an interface, and was therefore written in Python to allow the user to easily combine existing cost functions or constraints and self-implemented ones, to switch from one solver to another, etc. We believe this feature to be of importance given that the biomechanics community is known to be mainly made of software users rather than developers. Therefore, providing a custom interface in Python rather than in C++ [MOCO], was a driving objective of our work to facilitate a rapid appropriation by the community. Since flexibility and ease-of-use should not compromise the performances, all the inside computations are expressed as C++ CasAdi graphs, interfaced with C++ NLP solvers. These graphs can either be built in `casadi.MX()` or `casadi.SX()` which requires more RAM for building the problem but is faster to solve. While both may be used with *Ipopt*, *Acados* is only compatible with `casadi.SX()`.
Based on the *Bioviz* Python library, *Bioptim* proposes a series of online-generated figures, inspired by the real-time graphics from *Muscod-II* [ref], to analyze the iterations of the solvers.

This is made with minimal computational cost thanks to a XXX protocole. Our implementation leverages the *Python pickle* library for easily saving and loading OCPs for, e.g., post-processing analysis.

D. *Fast vs robust NLP solvers*

Fast solvers, such as *Acados*, offer the opportunity to use multi-start approaches 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 harder problems. On the other hand, robust solvers, such as *Ipopt*, are convenient when the user lacks information about the sought solutions and thus cannot guide the solver through a good initial guess. For biomechanics applications, the complementary characteristics of the interfaced solvers is a really useful tool.

E. *Multiphase*

Biomechanics studies often face changing dynamics or objective functions due to the loss or gain of contacts or time-varying biomechanical tasks. When tracking such a motion or trying to predict it, these changes translate into multi-phase OCP. This is currently one of the drawbacks of *Moco*, which does not provide this feature. *Bioptim*, however, is able to handle multi-phase OCPs, although they can currently only be solved with ipopt (see Ex. XXX and XXX).

F. *Cost vs constraints . . . relâcher le problème simplement*

???

For a more in-depth analysis of the real-time estimation capabilities of *Bioptim*, see [**?**].

G. *Limitations*

Bioptim is already a mature solution for solving biomechanical OCP. However some limitations should be raised. First, it is based on *Biorbd* which is not as advanced as *Opensim* or *Anybody* in terms of biomechanical features and audience. Nevertheless, *Biorbd* is actively maintained, fast and *Casadi*-compatible for automatic differentiation. The variety of proposed examples highlighted simple to advanced models. Even if defining a new model was made straightforward thanks to the `.bioMod` file format, *biorbd* does not include a GUI for building models. Some Opensim models can be translated into `.bioMod` [LINK] but our library does not yet support multiple wrapping objects, non-orthogonal DoFs between bodies, compliant contact force models (e.g. [SmoothSphereHalfSpaceForce [52]]) or muscle-tendon equilibrium. As seen in *Moco*, wrapping objects are rare due to computational cost and required optimization when a line of action is in contact with more than one object, which compromises automatic differentiation. 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.

H. *Future directions*

Realtime estimation using MHE

MOOCP .. using front pareto

Prediction of adaptations due to muscular fatigue using NMPC

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].

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 [REF] 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 biorbd 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). Bioptim acts as a catalyst in our group and several students contribute to this library; thank you to Paul, Théophile, Ariane, André, Kilperic.

References

[1] M. Diehl, H. G. Bock, H. Diedam, and P.-B. Wieber, “Fast direct multiple shooting algorithms for optimal robot control,” in *Fast motions in biomechanics and robotics*. Springer, 2006, pp. 65–93.

[2] M. R. Yeadon and M. J. Hiley, “The mechanics of the backward giant circle on the high bar,” *Human Movement Science*, vol. 19, no. 2, pp. 153–173, 2000.

[3] F. Leboeuf, G. Bessonnet, P. Seguin, and P. Lacouture, “Energetic versus sthenic optimality criteria for gymnastic movement synthesis,” *Multibody System Dynamics*, vol. 16, no. 3, pp. 213–236, 2006.

[4] F. Bailly, E. Charbonneau, L. Danès, and M. Begon, “Optimal 3d arm strategies for maximizing twist rotation during somersault of a rigid-body model,” *Multibody System Dynamics*, pp. 1–17, 2020.

[5] O. A. Kannape and O. Blanke, “Self in motion: sensorimotor and cognitive mechanisms in gait agency,” *Journal of Neurophysiology*, vol. 110, no. 8, pp. 1837–1847, 2013.

[6] A. Huchez, D. Haering, P. Holvoët, F. Barbier, and M. Begon, “Local versus global optimal sports techniques in a group of athletes,” *Computer methods in biomechanics and biomedical engineering*, vol. 18, no. 8, pp. 829–838, 2015.

[7] F. Bailly, A. Ceglia, B. Michaud, D. M. Rouleau, and M. Begon, “Real-time and dynamically consistent estimation of muscle forces using a moving horizon emg-marker tracking algorithm,” *Under review in Frontiers in Bioengineering and Biotechnology*, 2021.

Appendix

The appendix