The Journal of Open Source Software

biorbd: A C++, Python and MATLAB library to analyze and simulate the human body biomechanics

Benjamin Michaud[1] **and Mickaël Begon**[1]

1 École de Kinésiologie et de Sciences de l'Activité Physique, Université de Montréal

DOI: 10.21105/joss.02562

Software

- Review
- Repository
- Archive

Editor: Tania Allard

Reviewers:

- @trallard
- @abhishektha

Submitted: 16 June 2020
Published: 10 August 2020

Summary

Biomechanics is at the interface of several fields of science, such as mechanics, human physiology and robotics. Although this transdisciplinarity encourages the emergence of new ideas, the variety of data to analyze simultaneously can be overwhelming. Commonly biomechanical datasets are composed of skin markers trajectories (termed as markers), contact forces, electromyography (EMG) signal, inertial measurement units (IMU) kinematics, etc., which by nature are not straightforward to combine. It is at their meeting point—the body movement—that `biorbd` steps in; `bio` standing for biomechanics and `rbd` for `rigid body dynamics`. `biorbd` is a *feature-based development* library that targets the manipulation of biomechanical data in a comprehensive and accessible manner. For a given musculoskeletal model, it provides functions for inverse flow—i.e., from markers to EMG—and direct flow—i.e., from EMG to markers.

Since biomechanics often requires computationally expensive or real-time computations, the core of `biorbd` is written in C++. Although this language provides fast computations, it lacks the flexibility of higher-level languages. To meet the needs of the biomechanics community, Python and MATLAB binders are provided with `biorbd`. As a result, `biorbd` can elegantly be implemented to common workflows of researchers without compromising the required speed.

Finally, biomechanical data are often multidimensional and almost always time-dependent which can be challenging to visualize. To help with that, `biorbd-viz` (Michaud & Begon, 2018), a Python visualizer, was purposely designed. This visualizer allows animating the model, record videos, and, for models that include muscles, plot muscular outputs against various features of the movement.

A `biorbd` overview, the inverse and direct flow

Biomechanical analyses are usually based on one (or a mixture) of the inverse or direct flow (Kainz et al., 2016). Briefly, the former uses measurements from a movement (e.g., markers) and infers its cause, while the latter assumes control (e.g., EMG) and outputs the resulting kinematics.

Inverse flow

Inverse kinematics: Estimates the generalized coordinates (q)—i.e., the body kinematics—from body sensor measurements (e.g., markers, IMU, etc.). The main algorithm implemented is the Extended Kalman Filter (Fohanno, Colloud, Begon, & Lacouture, 2010) which by design facilitates the merging of multiple data sources and takes care of missing data.

Inverse dynamics: Estimates the generalized forces (τ) producing a given generalized acceleration ($\ddot{q}$) (the second time derivative of q):

$$\tau = M(q)\ddot{q} + N(q, \dot{q})$$

where $\dot{q}$ is the generalized velocities, $M(q)$ is the mass matrix and $N(q, \dot{q})$ are the nonlinear effects.

Static optimization: Estimates the muscle activations (α) producing a given τ (Anderson & Pandy, 2001). It minimizes the muscle activation *p*-norm (p usually being 2) that matches a given τ using nonlinear optimization (Ipopt (Wächter & Biegler, 2006)).

$$\begin{aligned} \underset{\alpha \in \mathbb{R}^m}{\text{minimize}} \quad & \|\alpha\|_p \\ \text{subject to} \quad & \tau_{mus_i}(\alpha, q, \dot{q}) - \tau_{kin_i}(q, \dot{q}, \ddot{q}) = 0, \qquad i = 1, \dots, n \\ & 0 \leq \alpha_{t_j} \leq 1, \qquad\qquad\qquad\qquad\quad j = 1, \dots, m \end{aligned}$$

where $\tau_{mus_i}(\alpha, q, \dot{q})$ and $\tau_{kin_i}(q, \dot{q}, \ddot{q})$ are τ computed from muscle forces ($F_{mus}(\alpha, q, \dot{q})$) and inverse dynamics, respectively.

Direct flow

Muscle activation dynamics: Estimates the muscle activation derivative ($\dot{\alpha}$) from the muscle excitation—that is the calcium release in the muscle that triggers the muscle contraction. Multiple activation/excitation dynamics are implemented (e.g., Thelen (2003) and Manal & Buchanan (2003)).

Muscular joint torque: Estimates the τ_{mus} from muscle forces ($F_{mus}(q, \dot{q}, \alpha)$) (Sherman, Seth, & Delp, 2010), estimated from α using a muscle model (e.g., Hill (1938), Thelen (2003)):

$$\tau_{mus} = J_{mus}(q)^T F_{mus}(q, \dot{q}, \alpha)$$

where $J_{mus}(q)$ is the muscle lengths Jacobian.

Forward dynamics: Estimates the $\ddot{q}$ from a given τ:

$$\ddot{q} = M(q)^{-1}\tau - N(q, \dot{q})$$

All the forward dynamics implemented in `RBDL` (Felis, 2017) are available.

Forward kinematics: Estimates the model kinematics outputs (e.g., markers, IMU) from a given q, after integrating twice $\ddot{q}$.

The dependencies

`biorbd` takes advantage of efficient back ends, especially the `RBDL` and `CasADi` libraries. `RBDL`, written by Martin Feliz (Felis, 2017), implements Featherstone equations of spatial geometry (Featherstone & Orin, 2000), successfully used in the field of robotics (Diehl, Bock, Diedam, & Wieber, 2006; Kurfess, 2018; Macchietto, Zordan, & Shelton, 2009). `RBDL` provides the computational core for body dynamics. `biorbd` extends `RBDL` by giving commonly used biomechanics nomenclature, and by adding biomechanical modules, amongst others. `RBDL` is based on the highly efficient C++ linear algebra library `Eigen` (Guennebaud, Jacob, & others, 2010). Although `Eigen` is flexible and fast enough for most of the common usage, it cannot automatically provide derivatives of functions. Therefore, `RBDL` was also augmented with the algorithmic differentiation library `CasADi` (Andersson, Gillis, Horn, Rawlings, & Diehl, n.d.). `CasADi` allows computing at low cost the derivatives of almost all the functions in `RBDL` and `biorbd`. This is particularly useful when using `biorbd` in a gradient-based optimization setting.

The need for `biorbd`

`OpenSim` (Seth et al., 2018) and `Anybody` (Damsgaard, Rasmussen, Christensen, Surma, & de Zee, 2006) are state-of-the-art biomechanics software that provides similar analysis flows with advanced user interface. `Anybody` being a closed and proprietary software, the reason to create another library for the open-source community is self-explanatory. Conversely, `OpenSim` is open-source and well established in the biomechanics community.

Nevertheless, in line with the idea that simulation software in biomechanics should be validated in multiple ways (Hicks, Uchida, Seth, Rajagopal, & Delp, 2015), providing similar tools but different in their approach allows the community to cross-validate the different implementation of the algorithms. For instance, two papers (Kim, Jung, Choi, Lee, & Koo (2018); Trinler, Schwameder, Baker, & Alexander (2019)) recently compared the outputs of `Anybody` and `OpenSim` and came to different results. Although the authors provided plausible explanations for these differences, due to the closed-source nature of `Anybody`, they had to assume that the implementation of the algorithms are flawless in both software. However, since a direct comparison between the actual codes is impossible, this is not verifiable. Having multiple open source software that produces similar ends by different means is a quality assurance for the end users: "Do not put all your eggs in one basket." To the best of our knowledge, there is no other open-source software that provides a complete direct and inverse flow in biomechanics. Therefore, in our opinion, `biorbd` and `OpenSim` are complementary.

Previous usage of `biorbd`

`biorbd` was used in most of the project of the *Laboratoire de Simulation et Modélisation du Mouvement* (S2M); particularly in analysis settings (jacksonImprovementsMeasuring-Shoulder2012a; Desmyttere, Hajizadeh, Bleau, & Begon, 2019; Verdugo, Pelletier, Michaud, Traube, & Begon, 2020) and simulation settings (Bélaise, Dal Maso, Michaud, Mombaur, & Begon, 2018, p. @moissenetOptimizationMethodTracking2019a) for a wide variety of movements (walking, piano playing, upper limb maximal exertions, etc.) More recently, an optimal control framework for biomechanics (`biorbd-optim` (Michaud & Begon, 2020)) based on Ipopt (Wächter & Biegler, 2006) and ACADOS (Verschueren et al., 2019) was developed around `biorbd`.

Acknowledgements

A huge thanks to Ariane Dang for her patience and contribution to writing the tests for `biorbd`!

References

Anderson, F. C., & Pandy, M. G. (2001). Static and dynamic optimization solutions for gait are practically equivalent. *Journal of Biomechanics*, *34*(2), 153–161. doi:10.1016/S0021-9290(00)00155-X

Andersson, J. A. E., Gillis, J., Horn, G., Rawlings, J. B., & Diehl, M. (n.d.). CasADi A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*.

Bélaise, C., Dal Maso, F., Michaud, B., Mombaur, K., & Begon, M. (2018). An EMG-marker tracking optimisation method for estimating muscle forces. *Multibody System Dynamics*, *42*(2), 119–143. doi:10.1007/s11044-017-9587-2

Damsgaard, M., Rasmussen, J., Christensen, S. T., Surma, E., & de Zee, M. (2006). Analysis of musculoskeletal systems in the AnyBody Modeling System. *Simulation Modelling Practice and Theory*, SIMS 2004, *14*(8), 1100–1111. doi:10.1016/j.simpat.2006.09.001

Desmyttere, G., Hajizadeh, M., Bleau, J., & Begon, M. (2019). Foot orthosis with add-on rearfoot posting can alter foot kinematics. In.

Diehl, M., Bock, H. G., Diedam, H., & Wieber, P.-B. (2006). Fast Direct Multiple Shooting Algorithms for Optimal Robot Control. In M. Diehl & K. Mombaur (Eds.), *Fast Motions in Biomechanics and Robotics: Optimization and Feedback Control*, Lecture Notes in Control and Information Sciences (pp. 65–93). Berlin, Heidelberg: Springer. doi:10.1007/978-3-540-36119-0_4

Featherstone, R., & Orin, D. (2000). Robot dynamics: Equations and algorithms. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)* (Vol. 1, pp. 826–834 vol.1). doi:10.1109/ROBOT.2000.844153

Felis, M. L. (2017). RBDL: An efficient rigid-body dynamics library using recursive algorithms. *Autonomous Robots*, *41*(2), 495–511. doi:10.1007/s10514-016-9574-0

Fohanno, V., Colloud, F., Begon, M., & Lacouture, P. (2010). Estimation of the 3D kinematics in kayak using an extended Kalman filter algorithm: A pilot study. *Computer Methods in Biomechanics and Biomedical Engineering*, *13*(S1), 55–56.

Guennebaud, G., Jacob, B., & others. (2010). Eigen v3.

Hicks, J. L., Uchida, T. K., Seth, A., Rajagopal, A., & Delp, S. L. (2015). Is My Model Good Enough? Best Practices for Verification and Validation of Musculoskeletal Models and Simulations of Movement. *Journal of Biomechanical Engineering*, *137*(2). doi:10.1115/1.4029304

Hill, A. V. (1938). The heat of shortening and the dynamic constants of muscle | Proceedings of the Royal Society of London. Series B - Biological Sciences. *Biological Science*, *126*(843), 136–195.

Kainz, H., Modenese, L., Lloyd, D. G., Maine, S., Walsh, H. P. J., & Carty, C. P. (2016). Joint kinematic calculation based on clinical direct kinematic versus inverse kinematic gait models. *Journal of Biomechanics*, *49*(9), 1658–1669. doi:10.1016/j.jbiomech.2016.03.052

Kim, Y., Jung, Y., Choi, W., Lee, K., & Koo, S. (2018). Similarities and differences between musculoskeletal simulations of OpenSim and AnyBody modeling system. *Journal of Mechanical Science and Technology*, *32*(12), 6037–6044. doi:10.1007/s12206-018-1154-0

Kurfess, T. (2018). *Robotics and automation handbook* (CRC press.).

Macchietto, A., Zordan, V., & Shelton, C. R. (2009). Momentum control for balance. In *ACM SIGGRAPH 2009 papers*, SIGGRAPH '09 (pp. 1–8). New Orleans, Louisiana: Association for Computing Machinery. doi:10.1145/1576246.1531386

Manal, K., & Buchanan, T. S. (2003). A one-parameter neural activation to muscle activation model: Estimating isometric joint moments from electromyograms. *Journal of Biomechanics*, *36*(8), 1197–1202. doi:10.1016/S0021-9290(03)00152-0

Michaud, B., & Begon, M. (2018). Biorbd-viz: A vizualization python toolbox for biorbd.

Michaud, B., & Begon, M. (2020). Biorbd-optim: An optimal control framework for biomechanical analyses using biorbd.

Moissenet, F., Bélaise, C., Piche, E., Michaud, B., & Begon, M. (2019). An Optimization Method Tracking EMG, Ground Reactions Forces, and Marker Trajectories for Musculo-Tendon Forces Estimation in Equinus Gait. *Frontiers in Neurorobotics*, *13*. doi:10.3389/fnbot.2019.00048

Seth, A., Hicks, J. L., Uchida, T. K., Habib, A., Dembia, C. L., Dunne, J. J., Ong, C. F., et al. (2018). OpenSim: Simulating musculoskeletal dynamics and neuromuscular control to study human and animal movement. *PLoS Computational Biology*, *14*(7). doi:10.1371/journal.pcbi.1006223

Sherman, M., Seth, A., & Delp, S. L. (2010). How to compute muscle moment arm using generalized coordinates. *Stanford University*.

Thelen, D. G. (2003). Adjustment of Muscle Mechanics Model Parameters to Simulate Dynamic Contractions in Older Adults. *Journal of Biomechanical Engineering*, *125*(1), 70–77. doi:10.1115/1.1531112

Trinler, U., Schwameder, H., Baker, R., & Alexander, N. (2019). Muscle force estimation in clinical gait analysis using AnyBody and OpenSim. *Journal of Biomechanics*, *86*, 55–63. doi:10.1016/j.jbiomech.2019.01.045

Verdugo, F., Pelletier, J., Michaud, B., Traube, C., & Begon, M. (2020). Effects of Trunk Motion, Touch, and Articulation on Upper-Limb Velocities and on Joint Contribution to Endpoint Velocities During the Production of Loud Piano Tones. *Frontiers in Psychology*, *11*. doi:10.3389/fpsyg.2020.01159

Verschueren, R., Frison, G., Kouzoupis, D., van Duijkeren, N., Zanelli, A., Novoselnik, B., Frey, J., et al. (2019). Acados: A modular open-source framework for fast embedded optimal control. *arXiv preprint*. Retrieved from http://arxiv.org/abs/1910.13753

Wächter, A., & Biegler, L. T. (2006). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, *106*(1), 25–57. doi:10.1007/s10107-004-0559-y