XVII International Symposium on Computer Simulation in Biomechanics
July 28th – 30th 2019, Alberta, Canada

Automatic, Computationally Efficient and Open-Source Optimal Control of Forward Dynamics Human Movement Simulations Using Python

Samuel G. Brockie and David J. Cole

Cambridge University Engineering Department, UK
Emails: sgb39@cam.ac.uk, djc13@cam.ac.uk

INTRODUCTION

Solving optimal control problems relating to human movements has been a focus of the biomechanics research community for the past 45 years [5]. Collocation methods have recently repeatedly been shown to offer superior speed and convergence compared to traditionally used shooting methods for both forward and inverse dynamics problems [8,9]. Most uses of collocation methods within the biomechanical simulation sphere, especially applications using OpenSim models, have been used to solve inverse dynamics problems [4,7]. The reason being that in comparison to shooting methods, collocation methods are reliant on gradient-based NLP solvers and therefore place onerous demands on the modeller to supply information on the derivatives of the system dynamics alongside their model for forward dynamics applications [9]. When forward dynamics have been optimised using collocation methods, the biomechanical models used and movement tasks simulated have been relatively simple [6,8].

There currently exists no easy-to-use framework that allows users to efficiently and reliably solve customisable forward dynamics optimal control problems using customisable neuromusculoskeletal models. The research described in this paper aims to provide tools that allow users easily and effectively to apply collocation methods to the optimal control of human movements.

METHODS

A suite of three Python packages has been developed to meet this research's main aim. Python was selected due to the wealth of open-source scientific packages available and its ease of prototyping. The three packages constituting the suite are:

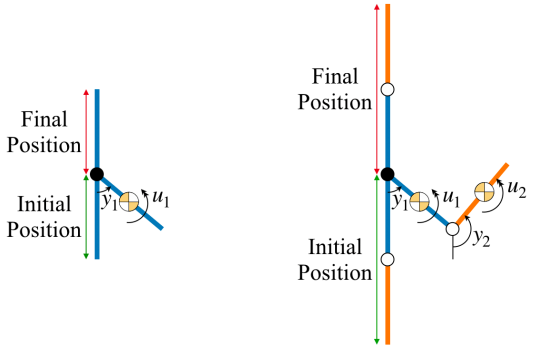
1. `pynamics`: a 3D multibody modelling package that allows creation of complex dynamic models via a simple Python API and auto-generates symbolic equations of motion.
2. `pyomechanics`: a biomechanical modelling extension to `pynamics` with an emphasis on the derivative continuity of its sub-models to ensure smooth search-space traversal during optimal control problem solving.
3. `opytimise`: a general purpose optimal control package that implements orthogonal collocation through Gaussian quadrature using Lagrange polynomials at the Radau points [2] and uses NLP solver IPOPT [11].

Efficiently generated symbolic equations of system dynamics allow rapid and accurate derivative calculation. Theano is used to compile rapidly-executing numerical functions for explicit evaluation of system dynamics and derivatives without the need for numerical differentiation to approximate the derivative vectors and matrices. Numba and Cython are used to enhance the computation speed of the developed packages' NumPy-dependent and pure-Python elements respectively.

RESULTS AND DISCUSSION

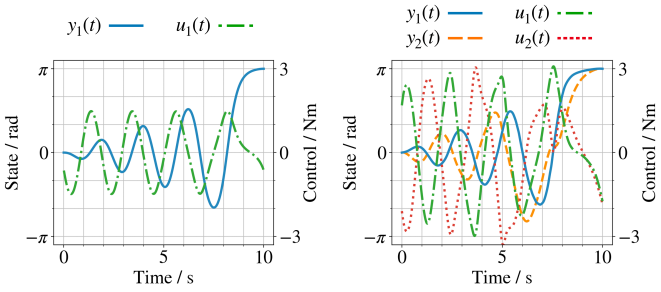
Initial validations of the developed suite have been conducted by recreating a previously presented simple pendulum swing-up optimal control task (task 1) [10] (Fig. 1(a)). An additional validation task using a double pendulum (task 2), a 2-dof extrapolation of task 1, has also been conducted (Fig. 1(b)). The problem objective was to swing all pendulum arms from vertical downwards to vertical upwards in 10.0 s while minimising the integral of the sum of all applied torques squared.

Both tasks were solved using two different approaches, firstly using the computational tools described in this abstract (auto-generated), and secondly as described by [10] (user-specified). In user-specified, the user was required manually to supply explicit functions for the objective function and its first derivative,



(a) Single pendulum (b) Double pendulum

Fig 1: Swing-up tasks.



(a) Single pendulum (b) Double pendulum

Fig 2: Optimal state and control for swing-up task.

and the constraint functions vector and its Jacobian matrix and sparsity structure. While this is possible for a simple problem like task 1 with two variables in its state and one in its control, the labour of providing this information manually increases with problem complexity and quickly becomes unfeasible. The auto-generated approach contains an additional initial step during the problem compilation where the constraints vector and all derivatives are automatically generated and compiled.

For both tasks, the global optimal state $y(t)$ and control $u(t)$ using auto-generated and user-specified were found to be the same within the numerical accuracy of the convergence tolerance (Fig. 2). Both tasks are multimodal; for both approaches convergence was achieved reliably from a wide range of random initial guesses throughout the solution space with the global optimum being found from 96% and 32% of initial guesses for tasks 1 and 2 respectively.

Benchmarks were taken as an average of 100 trial runs with 2000 even width collocation mesh sections. For task 1, auto-generated was 0.1s slower than user-specified (Table 1) due to the additional equation generation and compilation step. The ease of problem definition with auto-generated resulted in 15 times fewer lines of code for task 1; a line of code being either a single variable declaration or a single function call. The runtime trend reversed for task 2 with auto-generated running 0.2s faster than user-specified. This was due to auto-generated's more computationally efficient compiled numerical functions. The conciseness of problem definition with auto-generated also further increased for task 2 with 34 times fewer lines of code compared to user-specified. Auto-generated runtimes are expected to become proportionally faster relative to user-specified as problem complexity increases.

Although user-specified was initially published as a MATLAB script, the authors created a Python version to allow a fair comparison of the two approaches. All performance benchmarks were produced on a 2016 MacBook Pro 15-inch with 2.9 GHz Intel Core i7 running Python 3.6.

Table 1. Computational performance benchmarks of both tasks using auto-generated and user-specified.

Problem	Implementation	No. Lines Code	Runtime / s
Single pendulum	Auto-gen.	12	0.8
	User-spec.	177	0.7
Double pendulum	Auto-gen.	16	2.7
	User-spec.	542	2.9

There is ongoing work to add scaling algorithms and hp-adaptive mesh refinement to `opytimise`, and to validate the created software tools against three further predictive simulation tasks. These are: (A) a simple tug-of-war with a 1-dof, 2-muscle model [6]; (B) the maximum height of simulated maximal jumping [1]; and (C) simulated periodic maximal pedalling [3]. This further work will be presented at the ISCSB 2019. The open-source release of the developed packages is planned for later in 2019.

CONCLUSIONS

Reliable and rapid convergence to the same global optima using both approaches was found to occur given a sensible initial guess for the state and control in both tasks. These results show that the design decisions taken in this work meet the aim of the research and are intended to act as a blueprint for how the capability of collocation-based predictive simulation could in the future be built natively into biomechanical modelling software like OpenSim.

REFERENCES

1. Anderson FC et al. (1999). *Comput Methods Biomech Biomed Engin*, **2**: 201-231.
2. Betts JT (2010). *Siam*.
3. Bobbert MF et al. (2016). *Med Sci Sports Exerc*, **48**(5): 869-878.
4. de Groote F et al. (2016). *Ann Biomed Eng*, **44**(10): 2922-2936.
5. Hatze H (1976). *Math Biosci*, **28**(1), 99-135.
6. Lee L-F et al. (2016). *PeerJ*, **4**: e1638.
7. Meyer AJ et al. (2016). *Front Bioeng Biotechnol*, **4**(77): 1-26.
8. Porsa S et al. (2016). *Ann Biomed Eng*, **44**(8): 2542-2557.
9. van den Bogert AJ et al. (2011). *Procedia IUTAM,* **2**: 297-316.
10. van den Bogert AJ (2014). http://tinyurl.com/y2oek8ys
11. Wächter A et al. (2006). *Math Prog*, **106**(1): 25-57.

ACKNOWLEDGEMENTS

This work is supported by the EPSRC (UK Engineering and Physical Sciences Research Council) grant EP/N509620/1 PhD Studentship.