




opty: Software for trajectory optimization and parameter identification using direct collocation

Jason K. Moore¹ and Antonie van den Bogert²

¹ University of California, Davis ² Cleveland State University

DOI: [10.21105/joss.00300](https://doi.org/10.21105/joss.00300)

Software

- [Review](#) 
- [Repository](#) 
- [Archive](#) 

Submitted: 04 June 2017

Published: 10 January 2018

Licence

Authors of JOSS papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC-BY](#)).

Summary

opty is a tool for describing and solving trajectory optimization and parameter identification problems based on symbolic descriptions of ordinary differential equations that describe a dynamical system. The motivation for its development resides in the need to solve optimal control problems of biomechanical systems. The target audience is engineers and scientists interested in solving nonlinear optimal control and parameter identification problems with minimal computational overhead.

A user of opty is responsible for specifying the system's dynamics (the ODEs), the cost or fitness function, bounds on the solution, and the initial guess for the solution. opty uses this problem specification to derive the constraints needed to solve the optimization problem using the direct collocation method (Betts 2010). This method maps the problem to a non-linear programming problem and the result is then solved numerically with an interior point optimizer, IPOPT (Wächter and Biegler 2006) which is wrapped by cyipopt (cyipopt Developers 2017) for use in Python. The software allows the user to describe the dynamical system of interest at a high level in symbolic form without needing to concern themselves with the numerical computation details. This is made possible by utilizing SymPy (Meurer et al. 2017) and Cython (Behnel et al. 2011) for code generation and just-in-time compilation to obtain wrap optimized C functions that are accessible in Python.

Direct collocation methods have been especially successful in the field of human movement (Ackermann and Bogert 2010, A. J. van den Bogert et al. (2012)) because those systems are highly nonlinear, dynamically stiff, and unstable with open loop control. Typically, closed-source tools were used for multibody dynamics (e.g. SD/Fast, Autolev), for optimization (SNOPT), and for the programming environment (Matlab). Recently, promising work has been done with the Opensim/Simbody dynamics engine (Lee and Umberger 2016, Lin and Pandy (2017)), but this requires that Jacobian matrices are approximated by finite differences. In contrast, opty provides symbolic differentiation which makes the code faster and prevents poor convergence when the severe nonlinearity causes finite differences to be inaccurate. Furthermore, opty allows the system to be formulated using an implicit differential equation, which often results in far simpler equations and better numerical conditioning. The first application of opty was in the identification of feedback control parameters for human standing (Moore and Bogert 2015). It should be noted that opty can use any dynamic system model and is not limited to human movement.

Presently, opty only implements first order (backward Euler) and second order (midpoint Euler) approximations of the dynamics. Higher accuracy and/or larger time steps can be achieved with higher order polynomials (Patterson and Rao 2014), and opty could be extended towards such capabilities. In our experience, however, the low order discretizations provide more robust convergence to the globally optimal trajectory in a nonlinear system when a good initial guess is not available (Zarei 2016).

There are existing software packages that can solve optimal control problems and have have similarities to opty. Below, is a feature comparison:

Name
Citation
Language
License
Derivatives
Discretization
Implicit Dynamics
Solvers
Project Website
Casadi [1]
[@Andersson2013]
C++, Python, Octave,
LGPL
Automatic differentiation
None
Yes
IPOPT, WORHP, SNOPT, KNITRO
Casadi Website
DIDO
[@Ross2002]
Matlab
Commercial
Analytic
Pseudospectral
Yes
built-in
DIDO Website
DIRCOL
[@vonStryk1993]
Fortran
Non-commercial
Finite differences
Piecewise linear/cubic
Yes
NPSOL, SNOPT

DIRCOL Website
 DYNOPT
 [@Cizniar2005]
 Matlab
 Custom Open Source, Non-commercial
 Must be supplied by user
 Pseudospectral
 Mass matrix
 fmincon
 DYNOPT Code and Documentation
 FROST
 [@Hereid2017]
 Matlab, Mathematica
 BSD 3-Clause
 Analytic
 ?
 ?
 IPOPT, fmincon
 FROST Documentation
 GESOP
 [@Gath2001]
 Matlab, C, Fortan, Ada
 Commercial
 ?
 Pseudospectral
 No
 SLLSQP, SNOPT, SOCS
 Astos Solutions Gmbh
 GPOPS
 [@PattersonRao2014]
 Matlab
 Commercial
 Automatic differentiation
 Pseudospectral
 No
 SNOPT, IPOPT
 GPOPS Website

opty
 NA
 Python
 BSD 2-Clause
 Analytic
 Euler, Midpoint
 Yes
 IPOPT
 opty Documentation
 OTIS
 [@Hargraves1987]
 Fortran
 US Export Controlled
 ?
 Gauss-Labatto, Pseudospectral
 Yes
 SNOPT
 OTIS Website
 PROPT
 [@Rutquist2010]
 Matlab
 Commercial
 Analytic
 Pseudospectral
 Yes
 SNOPT, KNITRO
 TOMDYN Website
 PSOPT
 [@Becerra2010]
 C++
 GPL
 Automatic differentiation, Sparse finite differences
 Pseudospectral, RK
 Yes
 IPOPT, SNOPT
 PSOPT Website
 SOCS

[@Betts2010]

Fortran

Commercial

Finite differences

Euler, RK, & others

Yes

built-in

SOCS Documentation

[1]

Casadi does not have a built in direct collocation transcription but includes examples which show how to do so for specific problems.

Each of these software packages offer a different combination of attributes and features that make it useful for different problems. `opty` is the only package that has a liberal open source license, following precedent set by other core scientific Python packages. This allows anyone to use and modify the code without having to share the source of their application. `opty` also is the only package, open source or not, that allows (in fact forces) the user to describe their problem via a high level symbolic mathematical description using the API of a widely used computer algebra system. This relieves the user from having to translate the much simpler continuous problem definition into a discretize NLP problem. `opty` leverages the popular Scientific Python core tools like NumPy, SymPy, and matplotlib allowing users to include `opty` code into Python programs. set of tools. Lastly, the numeric code generated by `opty` to evaluate the NLP constraints is highly optimized and provides extremely efficient [parallel] evaluation of the constraints. This becomes very valuable for high dimensional dynamics. `opty` currently does not offer a wide range of discretization methods nor support for solvers other than IPOPT, but those could relatively easily be added based on user need.

References

- Ackermann, Marko, and Antonie J. van den Bogert. 2010. “Optimality Principles for Model-Based Prediction of Human Gait.” *Journal of Biomechanics* 43 (6):1055–60. <https://doi.org/https://doi.org/10.1016/j.jbiomech.2009.12.012>.
- Behnel, S., R. Bradshaw, C. Citro, L. Dalcin, D.S. Seljebotn, and K. Smith. 2011. “Cython: The Best of Both Worlds.” *Computing in Science Engineering* 13 (2):31–39. <https://doi.org/10.1109/MCSE.2010.118>.
- Betts, J. 2010. *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*. Advances in Design and Control. Society for Industrial; Applied Mathematics. <https://doi.org/10.1137/1.9780898718577>.
- Bogert, Antonie J van den, Maarten Hupperets, Heiko Schlarb, and Berthold Krabbe. 2012. “Predictive Musculoskeletal Simulation Using Optimal Control: Effects of Added Limb Mass on Energy Cost and Kinematics of Walking and Running.” *Proceedings of the Institution of Mechanical Engineers, Part P: Journal of Sports Engineering and Technology* 226 (2):123–33. <https://doi.org/10.1177/1754337112440644>.
- cyipopt Developers. 2017. “Cyipopt: Cython Interface for the Interior Point Optimzer IPOPT.”

Lee, Leng-Feng, and Brian R. Umberger. 2016. “Generating Optimal Control Simulations of Musculoskeletal Movement Using OpenSim and MATLAB.” *PeerJ* 4 (January):e1638. <https://doi.org/10.7717/peerj.1638>.

Lin, Yi-Chung, and Marcus G. Pandy. 2017. “Three-Dimensional Data-Tracking Dynamic Optimization Simulations of Human Locomotion Generated by Direct Collocation.” *Journal of Biomechanics* 59 (Supplement C):1–8. <https://doi.org/10.1016/j.jbiomech.2017.04.038>.

Meurer, Aaron, Christopher P. Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B. Kirpichev, Matthew Rocklin, AMiT Kumar, et al. 2017. “SymPy: symbolic Computing in Python.” *PeerJ Computer Science* 3 (January):e103. <https://doi.org/10.7717/peerj-cs.103>.

Moore, Jason K., and Antonie J. van den Bogert. 2015. “Quiet Standing Control Parameter Identification with Direct Collocation.” In *15th International Symposium on Computer Simulation in Biomechanics*, 21–22. Glasgow, U.K.

Patterson, Michael A., and Anil V. Rao. 2014. “GPOPS-II: A MATLAB Software for Solving Multiple-Phase Optimal Control Problems Using Hp-Adaptive Gaussian Quadrature Collocation Methods and Sparse Nonlinear Programming.” *ACM Trans. Math. Softw.* 41 (1). New York, NY, USA: ACM:1:1–1:37. <https://doi.org/10.1145/2558904>.

Wächter, Andreas, and Lorenz T. Biegler. 2006. “On the Implementation of an Interior-Point Filter Line-Search Algorithm for Large-Scale Nonlinear Programming.” *Mathematical Programming* 106 (1):25–57. <https://doi.org/10.1007/s10107-004-0559-y>.

Zarei, Milad. 2016. “Predictive Simulation of Rowing Exercise.” Master’s thesis, Cleveland State University; Cleveland State University. http://rave.ohiolink.edu/etdc/view?acc_num=csu1472557492.