1-30-2018

opty: Software for Trajectory Optimization and Parameter Identification Using Direct Collocation

Jason K. Moore
University of California

Antonie J. van den Bogert
Cleveland State University, a.vandenbogert@csuohio.edu

Follow this and additional works at: https://engagedscholarship.csuohio.edu/enme_facpub

Part of the Biomechanical Engineering Commons

How does access to this work benefit you? Let us know!

Recommended Citation

https://engagedscholarship.csuohio.edu/enme_facpub/370

This Article is brought to you for free and open access by the Mechanical Engineering Department at EngagedScholarship@CSU. It has been accepted for inclusion in Mechanical Engineering Faculty Publications by an authorized administrator of EngagedScholarship@CSU. For more information, please contact library.es@csuohio.edu.
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

Summary

opty is a tool for describing and solving trajectory optimization and parameter identification problems based on symbolic descriptions of ordinary differential equations and differential algebraic 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 ODE/DAEs), 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 similarities to opty. Below, is a feature comparison of those we are aware of:

<table>
<thead>
<tr>
<th>Name</th>
<th>Citation</th>
<th>Language</th>
<th>License</th>
<th>Derivatives</th>
<th>Discretization</th>
<th>Implicit</th>
<th>Solvers</th>
<th>Project</th>
<th>Website</th>
</tr>
</thead>
<tbody>
<tr>
<td>Casadi¹</td>
<td>(Andersson 2013)</td>
<td>C++, Python, Octave,</td>
<td>LGPL</td>
<td>Automatic differentiation</td>
<td>None</td>
<td>Yes</td>
<td>IPOPT, WORHP, SNOPT, KNITRO, built-in</td>
<td>Casadi</td>
<td>Website</td>
</tr>
<tr>
<td>DIDO</td>
<td>(Roo and Fahren 2002)</td>
<td>Matlab</td>
<td>Commercial</td>
<td>Analytic</td>
<td>Pseudospectral</td>
<td>Yes</td>
<td>SNOPT</td>
<td>DIDO</td>
<td>Website</td>
</tr>
<tr>
<td>DIRCOL</td>
<td>(Strock and Bulirsch 1992)</td>
<td>Fortran</td>
<td>Non-commercial</td>
<td>Finite differences</td>
<td>Piecewise linear/robic</td>
<td>Yes</td>
<td>NPSOL, SNOPT</td>
<td>DIRCOL</td>
<td>Website</td>
</tr>
<tr>
<td>GESOP</td>
<td>(Gauth and Wall 2001)</td>
<td>Matlab, C, Fortran, Ada</td>
<td>Commercial</td>
<td>?</td>
<td>Pseudospectral</td>
<td>No</td>
<td>SNOPT, SNOPT, SCS</td>
<td>GESOP</td>
<td>Website</td>
</tr>
<tr>
<td>GPOPS</td>
<td>(Patterson and Rau 2014)</td>
<td>Matlab</td>
<td>Commercial</td>
<td>Automatic differentiation</td>
<td>Pseudospectral</td>
<td>No</td>
<td>SNOPT, IPOPT</td>
<td>GPOPS</td>
<td>Website</td>
</tr>
<tr>
<td>IPOPT</td>
<td>NA</td>
<td>Python</td>
<td>BSD</td>
<td>Analytic</td>
<td>Euler, Midpoint</td>
<td>Yes</td>
<td>IPOPT</td>
<td>opty Documentation</td>
<td>IPOPT</td>
</tr>
<tr>
<td>OTIS</td>
<td>(Hargraves and Parks 1997)</td>
<td>Fortran</td>
<td>US</td>
<td>?</td>
<td>Gauss, Labatto, Pseudospectral</td>
<td>Yes</td>
<td>SNOPT, IPOPT</td>
<td>OTIS</td>
<td>Website</td>
</tr>
<tr>
<td>PROPT</td>
<td>(Rutquist and Edwall 2010)</td>
<td>Matlab</td>
<td>Controlled</td>
<td>Commercial</td>
<td>Analytic</td>
<td>Pseudospectral</td>
<td>SNOPT, KNITRO</td>
<td>TOMDYN</td>
<td>Website</td>
</tr>
<tr>
<td>PSOPT</td>
<td>(Solving Complex Optimal Control Problems at No Cost with PSOPT 2010)</td>
<td>Matlab</td>
<td>C++</td>
<td>GPL</td>
<td>Automatic differentiation, Sparse finite differences</td>
<td>Pseudospectral, RK</td>
<td>IPOPT, SNOPT</td>
<td>PSOPT</td>
<td>Website</td>
</tr>
<tr>
<td>SOCSD</td>
<td>(Bette 2010)</td>
<td>Fortran</td>
<td>Commercial</td>
<td>Finite differences</td>
<td>Euler, RK, &amp; others</td>
<td>Yes</td>
<td>built-in</td>
<td>SOCSD Documentation</td>
<td>SOCSD Website</td>
</tr>
</tbody>
</table>

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 for itself and its dependencies, 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 instead of a domain specific language. 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, Cython, and matplotlib allowing users to include opty code into Python programs. Lastly, the numeric code generated by opty to evaluate the NLP constraints is optimized providing extremely efficient parallel evaluation of the contraints. 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


¹Casadi does not have a built in direct collocation transcription but includes examples which show how to do so for specific problems.


