The Journal of Open Source Software

MocoExtendProblem: Interface Between OpenSim and MATLAB for Rapidly Prototyping Direct Collocation Goals

Aravind Sundararajan [1,2*¶], **Varun Joshi**[2*], **Brian Umberger**[3¶], **and Ludwig van Beethoven**[3]

1 Lyman Spitzer, Jr. Fellow, Princeton University, USA **2** Institution Name, Country **3** Independent Researcher, Country ¶ Corresponding author * These authors contributed equally.

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Open Journals

Reviewers:

- @openjournals

Submitted: 01 January 1970
Published: unpublished

Summary

MocoExtendProblem (MEP) is a more convenient MATLAB framework for prototyping and developing direct collocation goals for OpenSim Moco. MEP features several tools for testing and prototyping and using novel MocoGoals without resorting to rebuilding all of opensim or generating an .omoco file from C++ and loading the problem into MATLAB. Instead, users can structure their custom goals, build these with the visual studio and the MEX compiler and add them to existing MATLAB scripts.

This repository features :

- A set of C++ and MATLAB scripts and models for prototyping and testing custom goals
- a build.m script that compiles goals in the custom_goals or custom_goals45 and procedurally constructs the c++/MATLAB and compiles the MEX interface.
- Compatibility with OpenSIm 4.2-4.4 and 4.5
- The ability to include MEP as a submodule, build, and use valid custom goals
- Custom goals previously developed in our labs are in the custom_goals directory

Statement of need

OpenSim is an open-source software platform for biomechanical modeling and simulation. It is used to create and analyze computational models of anatomical structure and movement, with a focus on biomechanics, ergonomics, and physical rehabilitation. The platform enables researchers and healthcare professionals to investigate how this anatomical structure responds to different loads, postures, and activities. OpenSim can be used to study a wide range of biomechanical problems, such as the mechanics of walking and running, the impact of injury or disease on movement, and the effectiveness of rehabilitation exercises.

Direct collocation is a numerical optimization method used in dynamic systems and control engineering. It involves representing the system dynamics as a set of algebraic equations, which are then discretized over time, and solved as a nonlinear optimization problem to obtain the optimal control inputs. The method aims to find a numerical solution that satisfies the system constraints and optimizes a performance measure. Optimization paradigms like direct collocation have begun to play a critical role in expanding our understanding of biological locomotion through the in-silico testing of novel therapies and predictive capabilities.

Within OpenSim is a software toolkit called Moco (Dembia 2020) which employs direct collocation with IPOPT in order to solve trajectory optimization problems that could range from tracking experimental motion capture data for solving generalized coordinates, actuator

controls, and kinetics to fully predictive simulations of live or extinct taxa. While direct collocation is powerful and OpenSim can be used to generate a broad range of dynamically-consistent simulations it can be daunting for some users to modify and rebuild novel direct collocation goals.

We developed a set of build tools so researchers, clinicians, and students without experience compiling C++ can still write and test custom goals. Running build.m will compile custom goals developed and placed in the custom_goals directory. No further modifications to CMakeLists.txt are required; however cmake and Visual Studio 2019's msbuild.exe needs to be added to the system PATH . build.m will procedurally construct both extend_problem.m and ExtendProblem.cpp by parsing the header files of the discovered goals within the custom_goals directory. Both ExtendProblem.cpp and extend_problem.m generate bindings to instantiate custom goals placed in the custom_goals directory. Custom Goals will be compiled with VS2019+ and then MATLAB's MEX compiler is used to compile the MEX function. ExtendProblem.cpp leverages the C++ library mexplus (2014 Kota Yamaguchi) to gain access to MEX entry points entry and exit points through C++ macros.

To incorporate extend_problem goals into an existing script, a C-style pointer to the instantiated MocoProblem is passed as a constructor argument to the extend_problem.m class. Class methods of extend_problem.m are then used to add custom goals to the MocoProblem. This paradigm has implications for OpenSim and MATLAB developers beyond the scope of just incorporating MocoProblems; these same tools can be used to develop other tools or expand other classes and easily incorporate them into existing MATLAB problems. We have posted all tools, instructions and simulation results related to this project on GitHub.

Showcases

To demonstrate the utility of this framework, we developed a MATLAB script using OpenSim's API for developing predictive simulations of human walking that features Moco's built-in MocoControlEffortGoal and MocoAverageSpeedGoal. Since Moco lacks any built-in gait stability goals, we developed these as custom goals and built them into an extend problem class to compare their performance against a developed kinematic tracking simulation. To this aim, we developed 2 goals that are not currently available, 1 being the mass center should follow the center of pressure or the zero-moment point. And the other being that the center of mass should lay between the two feet frames in the ground reference frame, also known as the base of support, a zonotope that encompasses the two frames for calcaneus projected to the ground.

Data from 3 subjects walking at nondimensional walking speed v=0.56 was used to develop a generic model and averaged kinematic and force plate data of a 1.7 m tall 75 kg human walking overground at 1.6 m/s. The model features 54 muscles as well as 19 degrees of freedom. OpenSim Moco was used to generate a dynamically consistent simulation that tracked generalized coordinates, ground contact forces, while minimizing control.

`MEP` has been used in several previous and ongoing scientific works (ASB abstracts) for investigating both human and animal locomotion.

Citations

Citations to entries in paper.bib should be in rMarkdown format.

If you want to cite a software repository URL (e.g. something on GitHub without a preferred citation) then you can do it with the example BibTeX entry below for Smith et al. (2020).

For a quick reference, the following citation commands can be used: - `@author:2001` -> "Author et al. (2001)" - `[@author:2001]` -> "(Author et al., 2001)" - `[@author1:2001;`

@author2:2001] -> “(Author1 et al., 2001; Author2 et al., 2002)”

Figures

Figures can be included like this: Caption for example figure. and referenced from text using section .

Figure sizes can be customized by adding an optional second parameter: Caption for example figure.

Author Contributions

#Funding

This work was supported by the National Science Foundation (BCS 2018436 and BCS 2018523)

References

Smith, A. M., Thaney, K., & Hahnel, M. (2020). Fidgit: An ungodly union of GitHub and figshare. In *GitHub repository*. GitHub. https://github.com/arfon/fidgit