The Journal of Open Source Software

MocoExtendProblem: Interface Between OpenSim and MATLAB for Rapidly Developing Direct Collocation Goals in Moco

Aravind Sundararajan [1¶], **Varun Joshi** [2], **Brian R. Umberger** [2], and **Matthew C. O'Neill** [1]

1 Department of Anatomy, Midwestern University, Glendale Arizona, United States of America **2** School of Kinesiology, University of Michigan, Ann Arbor, Michigan, United States of America ¶ Corresponding author

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Open Journals

Reviewers:

- @openjournals

Submitted: 01 January 1970
Published: unpublished

Summary

MocoExtendProblem (`MEP`) is a framework to rapidly develop novel goals for biomechanical optimal control problems using OpenSim Moco (Dembia et al., 2020) and MATLAB (The MathWorks, Inc., Natick, MA, USA). `MEP` features several templates for testing and prototyping novel MocoGoals as well as a build tool to create a MEX function using OpenSim's API for MATLAB in lieu of rebuilding OpenSim from source or building a plugin and generating an .omoco file from C++ to load the problem into MATLAB. Instead, users structure and design custom goals in C++, build them with the provided tool, and call custom goals from within MATLAB scripts.

This repository features:

- A `build.m` script that compiles goals in the `custom_goals` directory and procedurally constructs the C++/MATLAB class implementations and compiles the MEX interface.
- Compatibility tested with OpenSim 4.2-4.5.
  - Support for OpenSim versions 4.2-4.4 require unique considerations to custom goal development and build pipeline since Booleans for division by duration, distance and mass were migrated to the abstract MocoGoal.
- The ability to include `MEP` as a submodule, build, and use valid custom goals.
- Three example custom goals in the `custom_goals` and `custom_goals_compat` directories.

Statement of need

OpenSim is an open-source software platform for modeling musculoskeletal structures and creating dynamic simulations of movement Seth et al. (2018). OpenSim enables researchers and clinicians to investigate how biological and non-biological structures respond to different loads, postures and activities in both static and dynamic situations. OpenSim has been used to study a wide range of biomechanical problems, such as the mechanics of walking and running (e.g. Falisse et al., 2019), the impact of injury or disease on movement (e.g. Johnson et al., 2022), and the effectiveness of rehabilitation exercises (e.g. Spomer et al., 2023).

While OpenSim originally featured several single-shooting methods like Computed Muscle Control and static optimization to solve for kinematics, kinetics and controls, OpenSim now includes Moco (Dembia et al., 2020) which employs an optimization paradigm called direct collocation for solving curve-fitting problems that range from solving for muscle forces, to tracking experimental data, and fully predictive simulations that solve the optimal control problem (OCP) subject to some predefined goals and constraints. Direct collocation is a

The Journal of Open Source Software

numerical optimal control method (Kelly, 2017) that is computationally efficient compared to single-shooting algorithms and is used extensively in computational approaches to understanding biological movement. While direct collocation is powerful, OpenSim Moco only provides a small predefined set of optimization goals which can be modified easily using OpenSim's MATLAB API; However, more sophisticated goals such as the 3 stability criteria explored in the showcases require understanding the C++ API and ability to build a custom plugin or building OpenSim from source. It can be daunting for many users to develop custom goals without experience in building software written in C++. We developed `MEP` so Moco users without experience in compiling C++ can still write and test custom goals.

`MEP` was developed using MATLAB (v. 2022a), which is a multimodal software platform that is commonly used by biomechanics researchers. Typically, OpenSim interfaces are generated automatically with SWIG (Simplified Wrapper and Interface Generator), as opposed to developing an interface with MATLAB classes and MEX (MATLAB Executable), which can be challenging for even experienced biomechanists because of the complexity of developing the MATLAB-OpenSim API plugin and the need to develop a C++ interface for this plugin. `MEP` only requires that CMake and msbuild from Visual Studio (VS) 2019 or higher as well as the C++ desktop development workload for VS to use MATLAB's MEX compiler with VS.

With `MEP`, OpenSim 4.5 users can simply run `build.m` to compile MocoGoals placed in the custom_goals directory, or in the custom_goals_compat directory for OpenSim versions 4.2-4.4. `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 can be compiled with Visual Studio 2019 or higher and then MATLAB's MEX compiler is used to compile ExtendProblem. `ExtendProblem.cpp` leverages the C++ library mexplus (Yamaguchi, 2018) to gain access to MEX entry points through C++ macros.

```
MocoExtendProblem
├── bin
│   └── RelWithDebInfo
│       ├── ExtendProblem.cpp
│       ├── extend_problem.m
│       └── extendProblem.mexw64
├── build
├── custom_goals or custom_goals_compat
│   └── MocoExampleGoal
│       ├── MocoExampleGoal.cpp
│       ├── MocoExampleGoal.h
│       ├── osimMocoExampleGoalDLL.h
│       ├── RegisterTypes_osimMocoExampleGoal.cpp
│       └── RegisterTypes_osimMocoExampleGoal.h
├── test
├── utils
│   ├── build_extend_class.m
│   ├── construct_goal_tree.m
│   ├── cpp_end.m
│   ├── cpp_start.m
│   ├── generate_content.m
│   ├── generate_content_compat.m
│   ├── get_goal_names.m
│   ├── get_setter_functions.m
│   ├── mex_end.m
│   ├── mex_start.m
│   ├── wrap_end.m
│   └── wrap_start.m
└── build.m
```

Figure 1: MEP framework. The researcher runs the `build.m` script (orange) that subsequently calls methods in the utils folder (red) which are tasked with reading the `custom_goals` folder (green) and procedurally construct the mex and the interface class that calls the mex (blue). Each custom goal (green) is handled as its own compiled plugin.

To create a new goal with MEP:

1. OpenSim 4.5+ users should copy a goal folder in the `custom_goals` directory while 4.2-4.4 users should copy a goal folder in `custom_goals_compat` to serve as a template.
2. Replace mentions of the original goal name to that of your new custom goal name in each of the 5 files and file names, being careful to also modify the include guards in the dll and register types header files.
3. Reimplement constructProperties(), initializeOnModelImpl(), calcIntegrandImpl(), calcGoalImpl() such that they describe your custom goal.

To incorporate extend_problem goals into an existing MATLAB script, a C-style pointer to the instantiated MocoProblem is passed as a constructor argument to the `extend_problem.m` class that wraps the MEP MEX. Class methods of `extend_problem.m` (Figure 1; blue) are then used to add custom goals to the MocoProblem broadly using the following syntax:

```
cptr = uint64(problem.getCPtr(problem)); % c-style pointer to instantiated MocoProblem
ep = extend_problem(cptr);               % instantiate procedurally-generated ExtendProb
ep.addMocoCustomGoal('custom_goal',weight,power,divide_by_distance); %add custom goal to
```

This paradigm has implications for OpenSim and MATLAB developers beyond the scope of just incorporating novel MocoGoals; these same strategies can be used to extend other OpenSim classes and easily incorporate them into existing MATLAB-OpenSim scripts. We have posted all tools, instructions and simulation results related to this project on GitHub and SimTK.org.

Requirements

- install cmake (tested with 3.23.3) and Visual Studio 2019+ with the C++ desktop development workload.
- install MATLAB (tested with 2022a/b), and configure MEX one time with mex -setup C++ to use VS.
- Download and install OpenSim from SimTK and follow the documentation for setting up OpenSim's MATLAB scripting environment.

Showcases

To demonstrate the utility of this framework, we generated a two-dimensional (2-D) walking simulation using the MATLAB-OpenSim API (Denton & Umberger, 2023). The base code uses the built-in MocoControlEffortGoal and MocoAverageSpeedGoal to generate tracking and predictive simulations of minimum effort walking at an average speed of 1.3 ms^{-1}. Additionally, each objective function includes implicit acceleration which minimizes the integral of squared continuous joint acceleration variables, and an auxiliary derivative term that minimizes the integral of squared derivative continuous variables such as fiber velocity to ensure smooth trajectories ($ACC_{smoothing}$).

Since Moco lacks built-in gait stability goals, we developed three stability goals using `MEP build.m` to create an ExtendProblem class that adds these to an existing MocoProblem (Figure 1; blue). The first is a base of support (Equation 1 BOS) criterion in which the whole-body center of mass (COM) is optimized to lay between the two hindfeet COMs projected to the ground reference frame in the transverse plane. The second is a zero-moment-point goal (Equation 2 ZMP) where the whole-body COM tracks the computed zero-tilting moment location in the transverse plane. The third is a marker acceleration minimization goal (Equation 3 ACC_{marker}) that minimizes the explicit accelerations of a marker placed on the head (marker location is arbitrary and can be set by the user).

`MEP`'s `build.m` was used to generate an ExtendProblem class that adds these new stability cost terms:

$$J_{BOS} = W_1 EFF^2 + W_2 ACC_{smoothing} + W_3 BOS \tag{1}$$

$$J_{ZMP} = W_1 EFF^2 + W_2 ACC_{smoothing} + W_3 ZMP \tag{2}$$

$$J_{ACC} = W_1 EFF^2 + W_2 ACC_{smoothing} + W_3 ACC_{marker} \tag{3}$$

The results of each multi-objective predictive simulation, in which the stability criterion was compiled using `MEP`, is shown against the results from a tracking simulation (Figure 2; Table 1) that closely-matched experimental data (Denton & Umberger, 2023). As the purpose was to demonstrate the utility of `MEP`, we did not tune the stability term weights to match the tracking result as closely as possible.

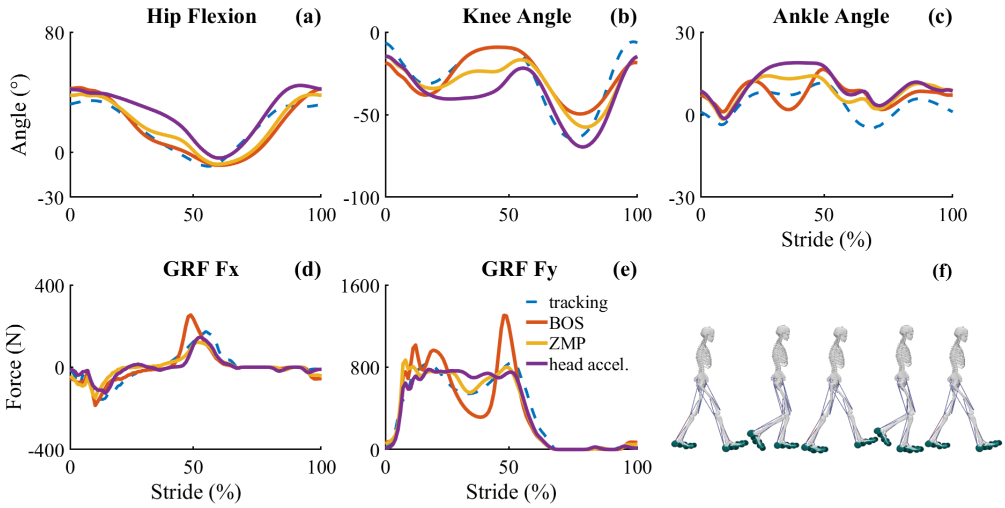

Figure 2: Sagittal plane hip, knee and ankle angles (a-c), vertical and A-P ground reaction forces (d-e), the 11 degree-of-freedom, 18 muscle sagittal plane human walking model used for tracking and predictive simulations (f)

Table 1: Objective cost and term breakdown for three predictive simulations using `MEP`.

	Objective cost	Effort cost	Smoothing cost	Stability cost
J_{BOS}	3.759046	2.270912	0.683608	0.794155
J_{ZMP}	4.184254	2.751212	0.725837	0.686290
J_{ACC}	4.774932	3.797785	0.793123	0.174308

While these examples used planar gait simulations, `MEP` is agnostic to model complexity or task, and is being used successfully in our ongoing research (e.g. Joshi et al., 2022; Sundararajan et al., 2023) of locomotor performance in humans and other animals. An additional benefit of sequestering novel goals into `MEP` is being able to back-port goals from a newer OpenSim version to an older version (i.e. taking a goal from OpenSim 4.4 and bringing that functionality to 4.2). Ultimately, `MEP` offers a modular framework to rapidly develop, test and compare novel MocoGoals for features beyond OpenSim Moco's current scope.

Funding

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

References

Delp, S., Anderson, F., Arnold, A., Loan, P., Habib, A., John, C., Guendelman, E., & Thelen, D. (2007). OpenSim: Open-source software to create and analyze dynamic simulations of movement. *Biomedical Engineering, IEEE Transactions on*, *54*, 1940–1950. https://doi.org/10.1109/TBME.2007.901024

Dembia, C. L., Bianco, N. A., Falisse, A., Hicks, J. L., & Delp, S. L. (2020). OpenSim Moco: Musculoskeletal optimal control. *PLoS Computational Biology*, *16*(12), 1–21. https://doi.org/10.1371/journal.pcbi.1008493

Denton, A. N., & Umberger, B. R. (2023). Computational performance of musculoskeletal simulation in OpenSim Moco using parallel computing. *International Journal for Numerical Methods in Biomedical Engineering*, *39*(12), e3777. https://doi.org/10.1002/cnm.3777

Falisse, A., Serrancolí, G., Dembia, C. L., Gillis, J., Jonkers, I., & De Groote, F. (2019). Rapid predictive simulations with complex musculoskeletal models suggest that diverse healthy and pathological human gaits can emerge from similar control strategies. *Journal of The Royal Society Interface*, *16*(157), 20190402. https://doi.org/10.1098/rsif.2019.0402

Johnson, R. T., Bianco, N. A., & Finley, J. M. (2022). Patterns of asymmetry and energy cost generated from predictive simulations of hemiparetic gait. *PLoS Computational Biology*, *18*(9), 1–26. https://doi.org/10.1371/journal.pcbi.1010466

Joshi, V., Boyer, K., & Umberger, B. R. (2022). Optimal control gait simulations of older adults predict foot placement trends not captured by reflex-based models. In *the Proceedings of the North American Congress on Biomechanics*. North American Congress on Biomechanics.

Kelly, M. (2017). An Introduction to Trajectory Optimization: How to Do Your Own Direct Collocation. *SIAM Review*, *59*(4), 849–904. https://doi.org/10.1137/16M1062569

Seth, A., Hicks, J. L., Uchida, T. K., Habib, A., Dembia, C. L., Dunne, J. J., Ong, C. F., DeMers, M. S., Rajagopal, A., Millard, M., Hamner, S. R., Arnold, E. M., Yong, J. R., Lakshmikanth, S. K., Sherman, M. A., Ku, J. P., & Delp, S. L. (2018). OpenSim: Simulating musculoskeletal dynamics and neuromuscular control to study human and animal movement. *PLoS Computational Biology*, *14*(7), 1–20. https://doi.org/10.1371/journal.pcbi.1006223

Spomer, A., Conner, B., Schwartz, M., Lerner, Z., & Steele, K. (2023). Audiovisual biofeedback amplifies plantarflexor adaptation during walking among children with cerebral palsy. *Journal of NeuroEngineering and Rehabilitation*, *20*. https://doi.org/10.1186/s12984-023-01279-5

Sundararajan, A., Larson, S. G., Umberger, B. R., & O'Neill, M. C. (2023). Optimal Control Simulations of 3-D Walking in Humans and Bipedal Chimpanzee. In *the Proceedings of The American Society of Biomechanics*. American Society of Biomechanics.

Yamaguchi, K. (2018). mexplus. In *GitHub repository*. GitHub. https://github.com/kyamagu/mexplus