

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

Aravind Sundararajan<sup>1</sup>✉, Varun Joshi<sup>2</sup>, Brian Umberger<sup>2</sup>, and Matthew O'Neill<sup>1</sup>

<sup>1</sup> Department of Anatomy, Midwestern University, Glendale Arizona, USA <sup>2</sup> School of Kinesiology, University of Michigan, Ann Arbor, Michigan, USA ✉ Corresponding author

DOI: [10.xxxxxx/draft](https://doi.org/10.xxxxxx/draft)

## Software

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

Editor: [Open Journals](#) ✉

## Reviewers:

- [@openjournals](#)

Submitted: 01 January 1970

Published: unpublished

## License

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

## Summary

optimal control problems using OpenSim Moco ([Dembia, 2021](#)) and MATLAB (The Mathworks, Inc., Natick, MA, USA). MEP features several templates for testing and prototyping novel MocoGoals in lieu of rebuilding OpenSim or generating an .omoco file from C++ to load the problem into MATLAB. Instead, users structure custom goals, build them, and call custom goals from 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.
- OpenSim versions lower than 4.5 require unique modifications to the 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.

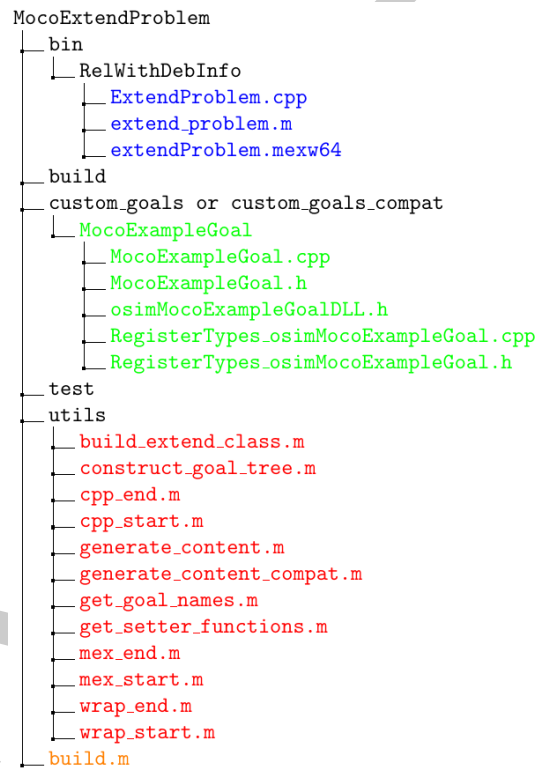
## Statement of need

OpenSim is an open-source software platform for modeling musculoskeletal structures and creating dynamic simulations of movement ([Seth, 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 ([Falisse et al., 2019](#)), the impact of injury or disease on movement ([Johnson, 2022](#)), and the effectiveness of rehabilitation exercises ([Spomer et al., 2023](#)).

OpenSim Moco ([Dembia, 2021](#)) employs an optimization paradigm called direct collocation to solve trajectory optimization problems that range from solving for muscle forces, to tracking experimental data, and fully predictive simulations. Direct collocation is a numerical optimal control method ([Kelly, 2017](#)) that is computationally efficient and is used extensively in computational approaches to understanding biological movement. While direct collocation is powerful, Moco only provides a fixed set of optimization goals. It can be daunting for many users to develop custom goals in C++. We developed MEP so Moco users without experience compiling C++ can still write and test custom goals. The OpenSim interfaces are created with SWIG, as opposed to MEX, which can be daunting for even experienced biomechanists.

40 MocoExtendProblem was developed using MATLAB versions 2022a. Running build.m will  
41 compile MocoGoals in the custom\_goals directory, or in the custom\_goals\_compat directory  
42 for OpenSim versions pre-4.5.

43 CMake and msbuild from Visual Studio 2019 or higher must be added to the system PATH.  
44 build.m will procedurally construct both extend\_problem.m and ExtendProblem.cpp by parsing  
45 the header files of the discovered goals within the custom\_goals directory. Both ExtendProb-  
46 lem.cpp and extend\_problem.m generate bindings to instantiate custom goals placed in the  
47 custom\_goals directory. Custom goals can be compiled with Visual Studio 2019 or higher  
48 and then MATLAB's MEX compiler is used to compile ExtendProblem. ExtendProblem.cpp  
49 leverages the C++ library mexplus (Yamaguchi, 2018) to gain access to MEX entry points  
50 through C++ macros.



**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.

51 To create a new goal with MEP:

- 52 1. OpenSim 4.5+ users should copy a goal in the custom\_goals directory while 4.2-4.4
- 53 users should copy a goal in custom\_goals\_compat.
- 54 2. Replace mentions of the original goal name to that of your new custom goal name in
- 55 each of the 5 files and file names, being careful to also modify the include guards in the
- 56 dll and register types header files.
- 57 3. Reimplement constructProperties(), initializeOnModelImpl(), calcIntegrandImpl(), cal-
- 58 cGoalImpl() such that they describe your custom goal.

59 To incorporate extend\_problem goals into an existing MATLAB script, a C-style pointer to the  
60 instantiated MocoProblem is passed as a constructor argument to the extend\_problem.m class  
61 that wraps the MEP MEX. Class methods of extend\_problem.m (Figure 1; blue) are then used

```

62 to add custom goals to the MocoProblem.
63 cptr = uint64(problem.getCPtr(problem));
64 ep = extend_problem(cptr);
65 ep.addMocoCustomGoal('custom_goal',weight,power,divide_by_distance);

```

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

## 70 Requirements

- 71 ■ Download and install OpenSim from [SimTK](#) and follow the documentation for setting  
72 up OpenSim's MATLAB scripting environment.
- 73 ■ Follow the instructions (OpenSim) to download necessary dependencies for both scripting  
74 in MATLAB and C++ development.
- 75 ■ In MATLAB, configure MEX with `mex -setup C++` to use the MS VisualStudio 2019+.

## 76 Showcases

77 To demonstrate the utility of this framework, we generated a two-dimensional (2-D) walking  
78 simulation using the MATLAB-OpenSim API ([Denton & Umberger, 2023](#)). The base code  
79 uses the built-in MocoControlEffortGoal and MocoAverageSpeedGoal to generate tracking and  
80 predictive simulations of minimum effort walking at an average speed of 1.3 m s<sup>-1</sup>. Additionally,  
81 each objective function includes implicit acceleration and auxiliary derivative terms that are  
82 minimized to ensure smooth trajectories.

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

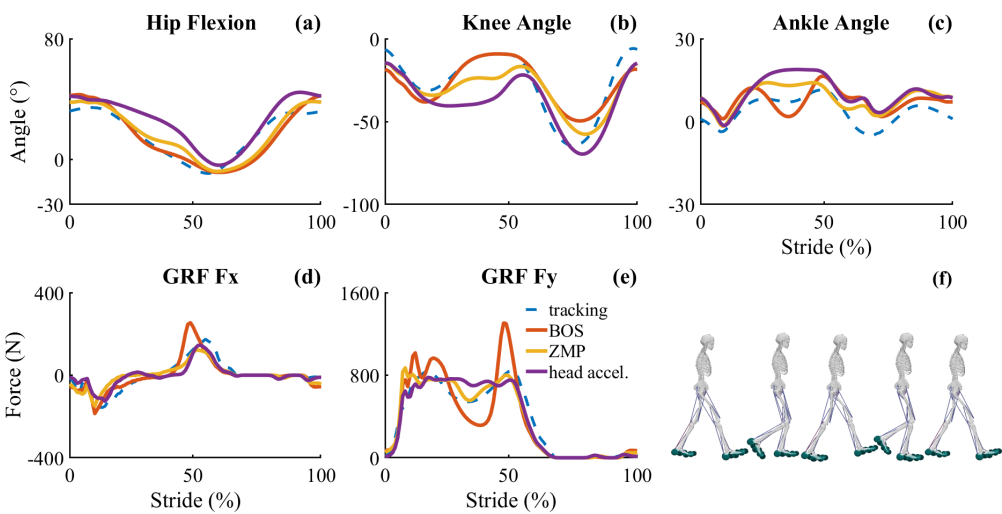
91 MEP's `build.m` was used to generate an `ExtendProblem` class that adds these new stability  
92 cost terms:

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

$$J_{zmp} = W_1 EFF^2 + W_2 ACC_{smoothing} + W_3 ZMP \quad (2)$$

$$J_{acc} = W_1 EFF^2 + W_2 ACC_{smoothing} + W_3 ACC_{marker} \quad (3)$$

93 The results of each multi-objective predictive simulation, in which the stability criterion was  
94 compiled using MEP, is shown against the results from a tracking simulation [[Figure 2](#); ??]  
95 that closely-matched experimental data ([Denton & Umberger, 2023](#)). As the purpose was  
96 to demonstrate the utility of MEP, we did not tune the stability term weights to match the  
97 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)

	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
$J_{accel}$	4.774932	3.797785	0.793123	0.174308

[Objective cost and term breakdown for three predictive simulations using MEP.]

While these examples used planar gait simulations, MEP is agnostic to model complexity or task, and is being used successfully in our ongoing research (Joshi, 2023; Sundararajan, 2023) of locomotor performance in humans and other animals. GNU Octave support would require minimal syntactical modification. An additional benefit of sequestering novel goals into ExtendProblem 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**

Dembia, N. A. A. F., C. L. AND Bianco. (2021). 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>

- 117 Falisse, A., Serrancolí, G., Dembia, C. L., Gillis, J., Jonkers, I., & De Groote, F. (2019). Rapid  
118 predictive simulations with complex musculoskeletal models suggest that diverse healthy  
119 and pathological human gaits can emerge from similar control strategies. *Journal of The*  
120 *Royal Society Interface*, 16(157), 20190402. <https://doi.org/10.1098/rsif.2019.0402>
- 121 Johnson, N. A. A. F., R. T. AND Bianco. (2022). Patterns of asymmetry and energy cost  
122 generated from predictive simulations of hemiparetic gait. *PLOS Computational Biology*,  
123 18(9), 1–26. <https://doi.org/10.1371/journal.pcbi.1010466>
- 124 Joshi, B., V. (2023). Optimal control gait simulations of older adults predict foot placement  
125 trends not captured by reflex-based models. In *North American Congress on Biomechanics*.  
126 North American Congress on Biomechanics.
- 127 Kelly, M. (2017). An introduction to trajectory optimization: How to do your own direct  
128 collocation. *SIAM Review*, 59(4), 849–904. <https://doi.org/10.1137/16M1062569>
- 129 Seth, J. L. A. U., A. AND Hicks. (2018). OpenSim: Simulating musculoskeletal dynamics  
130 and neuromuscular control to study human and animal movement. *PLOS Computational*  
131 *Biology*, 14(7), 1–20. <https://doi.org/10.1371/journal.pcbi.1006223>
- 132 Spomer, A., Conner, B., Schwartz, M., Lerner, Z., & Steele, K. (2023). Audiovisual biofeedback  
133 amplifies plantarflexor adaptation during walking among children with cerebral palsy. *Journal*  
134 *of NeuroEngineering and Rehabilitation*, 20. <https://doi.org/10.1186/s12984-023-01279-5>
- 135 Sundararajan, L., A. (2023). Optimal control simulations of 3-d walking in humans and bipedal  
136 chimpanzee. In *Proceedings of The American Society of Biomechanics*. Proceedings of  
137 The American Society of Biomechanics.
- 138 Yamaguchi, K. (2018). Mexplus. In *GitHub repository*. GitHub. [https://github.com/kyamagu/](https://github.com/kyamagu/mexplus)  
139 [mexplus](https://github.com/kyamagu/mexplus)