

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

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Published: unpublished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

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 (Seth, 2018). The platform enables researchers and healthcare professionals to investigate how biological and non-biological structures respond to different loads, postures and activities. It has been used to study a wide range of biomechanical problems, such as the mechanics of walking and running (Falisce et al., 2019), the impact of injury or disease on movement (Johnson, 2022), and the effectiveness of rehabilitation exercises (Spomer et al., 2023).

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 the software toolkit Moco (Dembia, 2021), which employs direct collocation with the interior-point optimizer IPOPT (Wächter & Biegler, 2006) in order to solve trajectory optimization problems that could range from tracking experimental motion capture data for

40 solving generalized coordinates, actuator controls, and kinetics to fully predictive simulations.  
41 Moco employs CasADi (Andersson et al., 2018) to transform MocoProblems consisting of  
42 control goals and constraints into sets of matrices for nonlinear optimization. While direct  
43 collocation is powerful and OpenSim can be used to generate a broad range of dynamically-  
44 consistent simulations, it can be daunting for some users to modify and rebuild novel direct  
45 collocation goals.

46 We developed MEP so researchers, clinicians, and students without experience compiling C++  
47 can still write and test custom goals. By contrast, OpenSim's interfaces for MATLAB  
48 are developed using SWIG, as opposed to MEX, which can be daunting for even seasoned  
49 biomechanists. Running build.m will compile custom goals developed and placed in the  
50 custom\_goals directory, or if using OpenSim 4.5, MEP will search the custom\_goals45. This  
51 distinction for pre- and post- 4.5 MEP is to handle where scaling arguments are moved to the  
52 abstract MocoGoal.

53 No further modifications to CMakeLists.txt are required; however cmake and msbuild.exe from  
54 Visual Studio 2019 or higher needs to be added to the system PATH. MocoExtendProblem was  
55 designed for use in matlab versions 2022a or igher. Presumably, the framework can be finessed  
56 to run on GNU Octave instead of matlab; however, this has not been tested. build.m will  
57 procedurally construct both extend\_problem.m and ExtendProblem.cpp by parsing the header  
58 files of the discovered goals within the custom\_goals directory. Both ExtendProblem.cpp and  
59 extend\_problem.m generate bindings to instantiate custom goals placed in the custom\_goals  
60 directory. Custom Goals will be compiled with VS2019+ and then MATLAB's MEX compiler  
61 is used to compile the MEX function. ExtendProblem.cpp leverages the C++ library mexplus  
62 (Yamaguchi, 2018) to gain access to MEX entry points entry and exit points through C++  
63 macros.

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



**Figure 1:** MEP Framework organization. The end user runs the build.m script (orange) that subsequently calls methods in the utils folder (red) which are tasked with reading the custom\_goals and custom\_goals45 folder (green) and procedurally construct the mex and the interface class that calls the mex (blue). Each custom goal (green) is essentially handled as its own compiled plugin.

68 To create a new goal with MEP:

- 69 1. copy one of the goals in the custom\_goals folder.

- 70 2. rename all files to that of your custom goal's name.
- 71 3. regex replace (or by hand) mentions of the original copied goal name to that of your
- 72 goal's name in each of the 5 files, being careful to also modify the include guards in the
- 73 dll and register types header files.
- 74 4. Reimplement `initializeOnModelImpl`, `calcIntegrandImpl`, `calcGoalImpl` such that they
- 75 describe your custom goal.

76 In order to run a new custom goal, obtain the C-style pointer from OpenSim's existing SWIG  
77 interface and pass this as a constructor argument to `extend_problem`.

```
cptr = uint64(problem.getCPtr(problem));
ep = extend_problem(cptr);
ep.addMocoCustomGoal('custom_goal',weight,power,divide_by_distance);
```

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

## 83 Showcases

84 To demonstrate the utility of this framework, we utilized a two-dimensional (2-D) full-body  
85 human musculoskeletal model operated through the MATLAB-OpenSim API to simulate  
86 a half walking gait cycle (Denton & Umberger, 2023). The base code uses Moco's built-  
87 in `MocoControlEffortGoal` and `MocoAverageSpeedGoal` to generate tracking and predictive  
88 simulations of a dynamically-consistent walking step that minimizes the sum of the squared  
89 control effort, at an average speed of 1.3 m s<sup>-1</sup>. Additionally each objective function has an  
90 implicit acceleration minimization and auxiliary derivative cost term which help to smooth  
91 model kinetics and remove transient oscillations in ground reaction forces.

92 Since Moco lacks any built-in gait stability goals, we developed three custom stability goals  
93 using MEP to prototype and compile into a new `extend_problem` class that adds these to an  
94 existing `MocoProblem`. The first is a base of support (BOS) criterion in which the center  
95 of mass is optimized to lay between the average of the two mass centers for the calcaneus  
96 projected to the ground reference frame, the second is a zero-moment-point goal (ZMP)  
97 criterion that assumes the model is an inverted pendulum and the center of mass tracks the  
98 computed zero-tilting moment location, and the third is a marker acceleration minimization  
99 goal (`ACCmarker`) that minimizes the explicit accelerations of the marker's station location.

100 MEP's `build.m` was used to generate an `extendproblem.cpp` and `extend_problem.m` class which  
101 wraps the custom goal to create a new, multi-objective function based on the sum of squared  
102 control effort + stability criteria, such that:

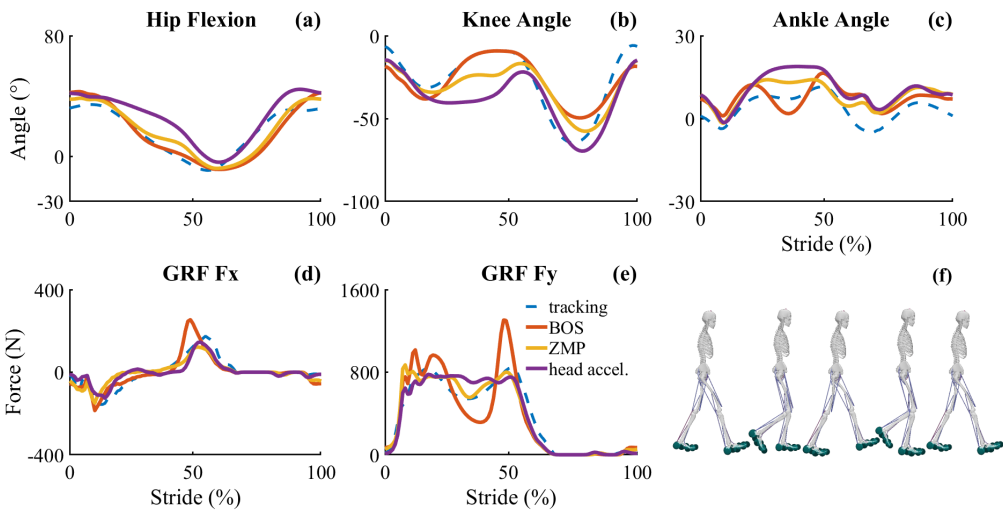
$$J_{BOS} = W_1 EFF^2 + W_2 ACC_{smoothing} + W_3 BOS \quad (1)$$

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

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

103 The results of each multi-objective predictive simulation, in which the stability criterion was  
104 compiled using MEP, is shown against the results from a tracking simulation Figure 2. The  
105 tracking simulation objective cost was a weighted sum of the tracking error (i.e. squared sum

106 of simulation from experimental kinematic and ground reaction force data) and sum of the  
107 squared control efforts.



**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
$J_{accel}$	4.774932	2.308934	0.672749	6.250044

108 MEP is being used in ongoing research (Aravind Sundararajan, 2023; Varun Joshi, 2023) of  
109 locomotor performance in humans and other animals. For validating MEP's results, after solving  
110 each optimal control problem, a test is done to verify if the output MocoSolution numerically  
111 equals an output reference of the same problem and weights to within a tolerance based on  
112 the model's assembly error tolerance.

113 #Funding

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

115 **References**

116 Andersson, J. A. E., Gillis, J., Horn, G., Rawlings, J. B., & Diehl, M. (2018). CasADi: A software  
117 framework for nonlinear optimization and optimal control. *Mathematical Programming  
118 Computation*, 11, 1–36. <https://api.semanticscholar.org/CorpusID:53559074>

119 Aravind Sundararajan, U. B., Larson SG. (2023). Optimal control simulations of 3-d walking in  
120 humans and bipedal chimpanzee. In *Proceedings of The American Society of Biomechanics*.  
121 Proceedings of The American Society of Biomechanics.

122 Dembia, N. A. A. F., Christopher L. AND Bianco. (2021). OpenSim moco: Musculoskeletal  
123 optimal control. *PLOS Computational Biology*, 16(12), 1–21. [https://doi.org/10.1371/  
124 journal.pcbi.1008493](https://doi.org/10.1371/journal.pcbi.1008493)

- 125 Denton, A. N., & Umberger, B. R. (2023). Computational performance of musculoskeletal  
126 simulation in OpenSim moco using parallel computing. *International Journal for Numerical*  
127 *Methods in Biomedical Engineering*, 39(12), e3777. <https://doi.org/10.1002/cnm.3777>
- 129 Falisse, A., Serrancolí, G., Dembia, C. L., Gillis, J., Jonkers, I., & De Groote, F. (2019). Rapid  
130 predictive simulations with complex musculoskeletal models suggest that diverse healthy  
131 and pathological human gaits can emerge from similar control strategies. *Journal of The*  
132 *Royal Society Interface*, 16(157), 20190402. <https://doi.org/10.1098/rsif.2019.0402>
- 133 Johnson, N. A. A. F., Russell T. AND Bianco. (2022). Patterns of asymmetry and energy cost  
134 generated from predictive simulations of hemiparetic gait. *PLOS Computational Biology*,  
135 18(9), 1–26. <https://doi.org/10.1371/journal.pcbi.1010466>
- 136 Seth, J. L. A. U., Ajay AND Hicks. (2018). OpenSim: Simulating musculoskeletal dynamics  
137 and neuromuscular control to study human and animal movement. *PLOS Computational*  
138 *Biology*, 14(7), 1–20. <https://doi.org/10.1371/journal.pcbi.1006223>
- 139 Spomer, A., Conner, B., Schwartz, M., Lerner, Z., & Steele, K. (2023). Audiovisual biofeedback  
140 amplifies plantarflexor adaptation during walking among children with cerebral palsy. *Journal*  
141 *of NeuroEngineering and Rehabilitation*, 20. <https://doi.org/10.1186/s12984-023-01279-5>
- 142 Varun Joshi, B. U., Kathryn Boyer. (2023). Optimal control gait simulations of older adults  
143 predict foot placement trends not captured by reflex-based models. In *North American*  
144 *Congress on Biomechanics*. North American Congress on Biomechanics.
- 145 Wächter, A., & Biegler, L. T. (2006). On the implementation of an interior-point filter  
146 line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*,  
147 106, 25–57. <https://api.semanticscholar.org/CorpusID:14183894>
- 148 Yamaguchi, K. (2018). Mexplus. In *GitHub repository*. GitHub. <https://github.com/kyamagu/mexplus>
- 149