XX International Symposium on Computer Simulation in Biomechanics
July 23rd – 25th 2025, Uppsala

FAST SYMBOLIC METHODS FOR MUSCLE-DRIVEN OPTIMAL CONTROL

Jason K. Moore[1] and Samuel G. Brockie[2] and Timótheüs J. Stienstra[3] and Antonie van den Bogert[4]

[1]Department of BioMechanical Engineering, Delft University of Technology, The Netherlands
[2]Department of Engineering, University of Cambridge, United Kingdom
[3]Aerospace Engineering, Delft University of Technology, The Netherlands
[4]Mechanical Engineering, Cleveland State University, USA
Email: j.k.moore@tudelft.nl

INTRODUCTION

Direct collocation is now widely used for solving biomechanical optimal control problems. For example, OpenSim now includes Moco [1] which allows users to find optimal control solutions with lower overhead. One advantage of direct collocation is the speed at which solutions can be found, which can be orders of magnitude faster than methods which require sequentially integrating the differential equations. Direct collocation methods can still result in overall slower speed when searching for global minima and choices in the software design can reduce computational efficiency. For example, Moco relies on finite differences for gradients which may reduce performance and accuracy.

Solving an optimal control problem resulting from direct collocation requires evaluating the system's equations of motion, its Jacobian, and possibly its Hessian thousands to millions of times, but this evaluation is parallelizable over multiple compute cores and can leverage the massive sparsity of the equations. Maximizing the compute and memory performance of the functions that evaluate the constraints and their derivatives allow the subsequent optimization algorithms to solve very large problems, both in simulation duration and number of degrees of freedom.

Early dynamics simulation software used computer-aided algebra techniques, which allowed for analytical derivatives when forming equations of motion, but this has faded from popularity with the growth of numerical physics engines. Code generated from computer-aided algebra can be optimized for evaluation performance using both compiler pre-optimizations and built-in compiler optimizations that can be harder to apply to numeric physics engines.

We will demonstrate software that leverages symbolic formulations to generate computationally efficient parallelized functions based on implicit dynamics to evaluate the non-linear programming problem's constraints and its Jacobian based on the ideas of [4]. We demonstrate solving a 3D arm-muscle driven bicycle-rider model that has three degrees of freedom and seven algebraic constraints resulting in millions of floating point operations in the implicit equations of motion.

METHODS

The non-linear equations of motion for musculoskeletal-vehicle systems can be described by differential algebraic equations that are functions of the states $\mathbf{y} \in \mathbb{R}^n$, controls $\mathbf{r}$, and constant parameters $\mathbf{p}$, in general. The implicit equations take this form:

$$\mathbf{f}(\dot{\mathbf{y}}(t), \mathbf{y}(t), \mathbf{r}(t), \mathbf{p}) = \mathbf{0} \in \mathbb{R}^n \tag{1}$$

These equations can contain elements such as musculotendon force-velocity relationships, kinematic loops, friction and collision forces, etc. We write these equations as analytically differentiable functions. To map these to non-linear programming constraints we discretize the equations with backward Euler differentiation over a constant time step and code generate functions in C using opty [2] that evaluate the constraints and Jacobian over all N nodes while exploiting the high sparsity of the Jacobian. The symbolic Jacobian is calculated in forward mode applying the chain rule through all common sub-expressions which efficiently handles differentiating equations of motion with many mathematical operations. This results in cacheable computationally efficient numerical functions that evaluate the non-linear programming problem constraints and its Jacobian. Evaluation of the objective is generally a low computational cost, so this is only done in Python, but also with a symbolic-to-numeric conversion.

Table 1: Mean performance of seven bicycle-rider $N = 200$ optimal control problem (OCP) solutions on a Macbook Pro with 2.4 GHz 8-Core processor, running Python 3.12.9, SymPy 1.13.3, and opty 1.4.0 using Ipopt 3.14.17 with Mumps 5.7.3.

Symbolic EoM [s]	Symbolic Jacobian [s]	Constraint Clang [s]	Jacobian Clang [s]	Symbolic OCP [s]	OCP Solve [s]	NLP iterations	Objective evaluations	Gradient evaluations	Constraint evaluations	Jacobian evaluations
14.9	35.4	58.3	66.7	169.8	73.0	286	1098	286	1098	292

To demonstrate our methods, we developed a bicycle-rider model that is the non-linear Carvallo-Whipple model of the vehicle with four additional rigid bodies representing the upper and lower arm body segments using SymBRiM [3]. We include four lumped Hill-type muscle models representing the extensor and flexor groups for the elbow. The model has seven holonomic constraints and four nonholonomic constraints resulting in a 10^{th} order model with seven additional algebraic equations. This results in 17 equations with 2.8M floating point operations in the implicit symbolic equations of motion. After efficient Jacobian formulation and compiler pre-optimizations, this reduces to 10K and 69K operations for evaluating the constraints and Jacobian, respectively. This numerical evaluation time depends on the number of nodes, so proportional to $N\times$79K. For $N = 10$K we time the evaluation of these two functions with and without parallelization using OpenMP 4.5 seeing a 4.6$\times$ and 3.1$\times$ speed up over 23 ms and 190 ms, respectively. Lastly, we define a minimal muscle excitation lane change tracking task, starting from zero speed up to a mean speed of 3 m/s over the fixed duration of 6 seconds, as an optimal control problem.

RESULTS AND DISCUSSION

Table 1 shows timings for solving a 200 node lane change problem. Building the model and forming the OCP takes 185 seconds with most time spent compiling the generated functions with Clang. This is a fixed cost regardless of the size of N and can be cached to disk and need not be built again unless the dynamics model changes. Ipopt solves the problem in 73 seconds. Fig. 1 shows an example optimal solution.

CONCLUSIONS

Our methods focus on maximizing the performance of evaluating the constraint and Jacobian numerical functions. The performance of these functions scales linearly with the number of collocation nodes and can be sped up multiplicatively by the number of compute cores available in the best-case scenario. When Ipopt needs to evaluate these functions thousands or millions of times, for example in a long-duration problem, the overall compute time can be greatly reduced.

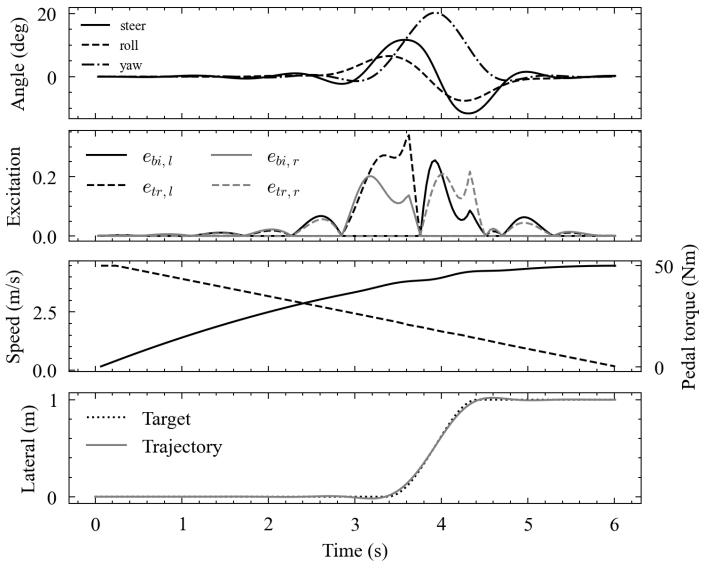

Fig 1: An optimal solution: 1) bicycle motion, 2) left/right arm biceps/triceps muscle excitations, 3) bicycle speed (solid) and propulsion torque (dashed), and 4) lateral displacement.

REFERENCES

[1] C. L. Dembia, N. A. Bianco, A. Falisse, J. L. Hicks, and S. L. Delp. OpenSim Moco: Musculoskeletal optimal control. *PLOS Computational Biology*, 16(12):e1008493, Dec. 2020.

[2] J. K. Moore and A. van den Bogert. Opty: Software for trajectory optimization and parameter identification using direct collocation. *JOSS*, 3(21):300, Jan. 2018.

[3] T. J. Stienstra, S. G. Brockie, and J. K. Moore. BRiM: A modular bicycle-rider modeling framework. In *Bicycle and Motorcycle Dynamics 2023*, Delft, The Netherlands, Oct. 2023. TU Delft OPEN Publishing.

[4] A. J. van den Bogert, D. Blana, and D. Heinrich. Implicit methods for efficient musculoskeletal simulation and optimal control. *Procedia IUTAM*, 2:297–316, Jan. 2011.

ACKNOWLEDGMENTS

P. Stahlecker contributed to opty; funds were from CZI.