XVII International Symposium on Computer Simulation in Biomechanics
July 28th – 30th 2019, Alberta, Canada

A Benchmark Comparison of Optimal Control Software: GPOPS-II versus OpenSim Moco

Andrew Baines[1], Mohammad Shourijeh[1], Christopher Dembia[2], Nick Bianco[2], Scott Delp[2], Benjamin Fregly[1]

[1]Department of Mechanical Engineering, Rice University, US [2]Mechanical Engineering Department, Stanford University, United States
Email: andrew.baines@rice.edu, Web: rcnl.rice.edu

INTRODUCTION

As computational resources increase, so does the desire to solve more complex human motion prediction problems, and with it an innate need for software that is capable of accurately solving complex problems. This preliminary study reviews GPOPS-II, a commercially available optimal control software, and OpenSim Moco, which is currently under development by the Neuromuscular Biomechanics Lab (NMBL) at Stanford University. The intent of this study is not to determine which software is better, but rather to review the accuracy and performance of both software packages by solving an inverse kinematic analysis problem while maintaining dynamic consistency.

Optimal control is used to solve a range of movement prediction problems, such as inverse kinematics (IK) analyses, muscle force predictions, skeletal motion predictions, and joint contact force prediction. GPOPS-II and OpenSim Moco were the chosen optimal control software packages for this comparison study. GPOPS-II [1] is a commercially available MATLAB based software intended to solve general nonlinear optimal control problems. GPOPS-II implements variable-order Gaussian quadrature methods where a continuous-time optimal control problem is approximated as a sparse nonlinear programming problem (NLP). In GPOPS-II, the NLP is solved using either the IPOPT or SNOPT solver. OpenSim Moco is a framework for solving optimal control problems for musculoskeletal systems, developed by the NMBL at Stanford University. The framework is written in C++ and has MATLAB, Python, and XML interfaces. OpenSim Moco uses the direct collocation method to solve optimal control. The framework will eventually be freely available. Since OpenSim Moco is still under development, an alpha version of the software was used for this study.

METHODS

A simplified planar three-segment model of squatting was used for this study – this is a reproduction of a publicly available problem [2]. The model, shown in Figure 1, contains three reference frames (B, C and D), which represent the foot, femur and thigh respectively, and a point mass at the top of the thigh (hip joint) to represent the upper body collectively. The model contains five degrees of freedom (q_{1-5}), and 6 markers (M_{1-6}), which are also shown in Figure 1. Reference marker trajectory data, reference joint torques, and reference joint angles and velocities were provided for this problem. Ground reaction force data were also provided for this problem, which was used to apply provided ground reaction force at point T (toes) of the model. Using provided data the joint torques at point A (ankle joint) and point K (knee joint) are found.

The joint torques in the model are controlled directly to minimize the errors between the model and reference marker positions, while applying reference ground reaction forces at the toe of the model. The generalized coordinates (q_{1-5}) and generalized velocities (u_{1-5}) were defined as states for this problem, the joint torques (T_A, T_K) were defined as controls, and the cost function for this problem was to minimize the sum of squares of marker error.

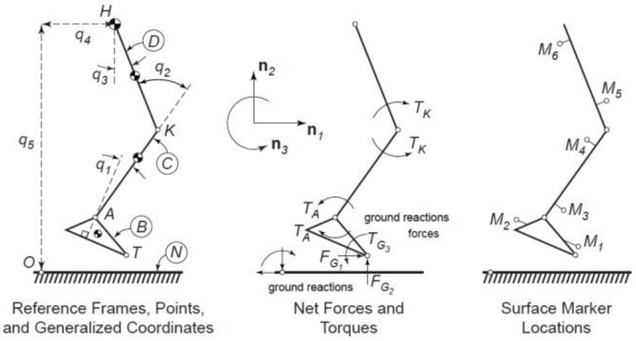

Fig 1: Planar squat model

Using MATLAB, this model was created in both GPOPS-II and OpenSim Moco.

RESULTS AND DISCUSSION

The marker positions and joint angles for both models were consistent with the provided reference data. This verifies that both software packages are able to accurately perform inverse kinematic analysis. However, Figure 2 shows that the GPOPS-II model joint velocities of q_1, q_2 and q_3 oscillate about the reference data between 0.3 and 0.5 seconds. Figure 3 shows that the ankle torque values from the GPOPS-II model are mostly consistent with the reference joint torques. However, the calculated knee torque values have considerable amounts of error between 0.25 and 0.4 seconds, and from 0.6 to 0.7 seconds. The total run-time for GPOPS-II was 182 seconds.

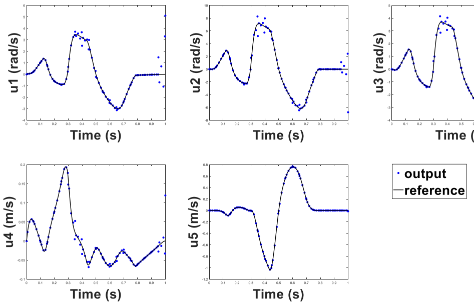

Fig 2: GPOPS-II joint velocity results for generalized coordinates u_1 (top left) – u_5 (bottom right)

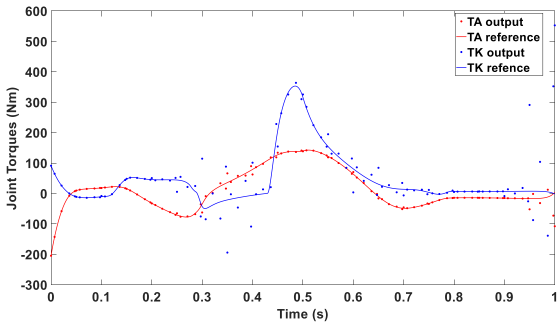

Fig 3: GPOPS-II ankle (red) and knee (blue) torques

The results of OpenSim Moco are shown in Figures 4 and 5. The calculated joint velocities shown in Figure 4 agree with the provided reference data, but there was a noticeable discrepancy in the ankle joint from 0 to 0.1 seconds. The OpenSim Moco run-time was approximately 500 seconds.

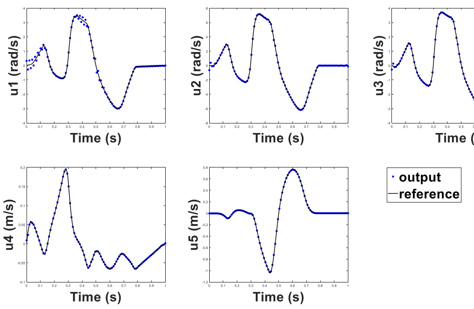

Fig 4: OpenSim Moco joint velocity results for generalized coordinates u_1 (top left) – u_5 (bottom right)

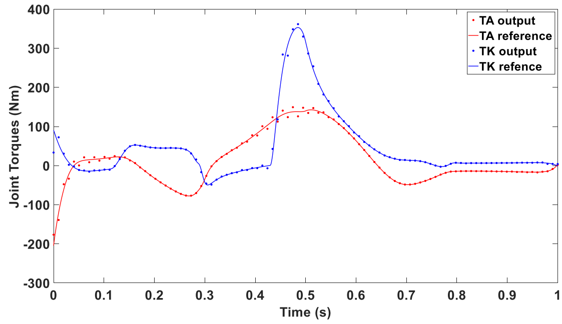

Fig 5: OpenSim Moco ankle (red) and knee (blue) torques

The same NLP solver (IPOPT) and number of collocation points (100) were used for both software programs. However, GPOPS provides additional functionality tools (e.g., mesh refinement), which improve the accuracy of the solution. These features are not currently implemented in OpenSim Moco. Figure 6 shows the calculated ankle and knee torque values of the GPOPS-II model when utilizing the mesh refinement capabilities.

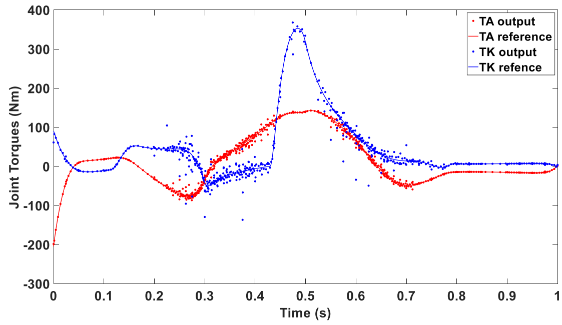

Fig 6: GPOPS-II ankle (red) and knee (blue) torques using mesh refinement

CONCLUSIONS

Both models converged to the optimal solution while maintaining dynamic consistency. Initial findings suggest that OpenSim Moco may provide more accurate results than GPOPS-II, but at the expense of longer computational run-times. From a usability standpoint, OpenSim Moco could be viewed as more user friendly than GPOPS-II because of how compact the MATLAB scripting is in comparison to GPOPS-II. Future work will further explore differences between both software, including sensitivity to solver type and effects of solver error tolerances on run-time.

REFERENCES

1. Patterson, M. A. and Rao, A. V., *ACM Transactions on Mathematical Software*, Vol. 41, No. 1., October 2014, pp. 1:1 - 1:37.
2. https://simtk.org/projects/ocworkshop

ACKNOWLEDGEMENTS

This research was funded by Cancer Prevention Research Institute of Texas (CPRIT) grant RR170026.