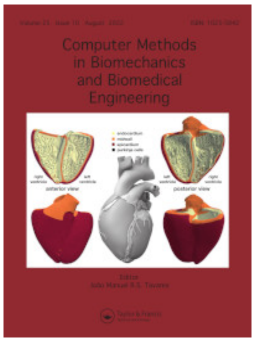

Computer Methods in Biomechanics and Biomedical Engineering

ISSN: (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/gcmb20

OpenColab project: OpenSim in Google colaboratory to explore biomechanics on the web

Hossein Mokhtarzadeh, Fangwei Jiang, Shengzhe Zhao & Fatemeh Malekipour

https://doi.org/10.1080/10255842.2022.2104607

OpenColab project: OpenSim in Google colaboratory to explore biomechanics on the web

Hossein Mokhtarzadeh[a], Fangwei Jiang[b], Shengzhe Zhao[b] and Fatemeh Malekipour[a]

[a]Department of Biomedical Engineering, The University of Melbourne, Melbourne, Australia; [b]Faculty of Engineering and Information Technology, The University of Melbourne, Melbourne, Australia

ABSTRACT
OpenSim is an open-source biomechanical package with a variety of applications. It is available for many users with bindings in MATLAB, Python, and Java *via* its application programming interfaces (APIs). Although the developers described well the OpenSim installation on different operating systems (Windows, Mac, and Linux), it is time-consuming and complex since each operating system requires a different configuration. This project aims to demystify the development of neuro-musculoskeletal modeling in OpenSim with zero configuration on any operating system for installation (thus cross-platform), easy to share models while accessing free graphical processing units (GPUs) on a web-based platform of Google Colab. To achieve this, OpenColab was developed where OpenSim source code was used to build a Conda package that can be installed on the Google Colab with only one block of code in less than 7 min. To use OpenColab, one requires a connection to the internet and a Gmail account. Moreover, OpenColab accesses vast libraries of machine learning methods available within free Google products, e.g. TensorFlow. Next, we performed an inverse problem in biomechanics and compared OpenColab results with OpenSim graphical user interface (GUI) for validation. The outcomes of OpenColab and GUI matched well ($r \geq 0.82$). OpenColab takes advantage of the zero-configuration of cloud-based platforms, accesses GPUs, and enables users to share and reproduce modeling approaches for further validation, innovative online training, and research applications. Step-by-step installation processes and examples are available at: https://simtk.org/projects/opencolab.

ARTICLE HISTORY
Received 23 October 2021
Accepted 18 July 2022

KEYWORDS
OpenSim; Python; MATLAB; GitHub; Explorer; collaboration; Anaconda; Pandas; TensorFlow; machine learning; Biomechanics; computational modeling; neuro-musculoskeletal modeling; cloud computing; simulation

Introduction

Using computational tools to understand human and animal neuro-musculoskeletal systems (NMSKs) is vital since we cannot quantify these functions through only experimental measurements. NMSK models are powerful tools that use numerous computer-based, mathematical, and physics-based concepts to study the function of different muscles, joints, ligaments during daily activities (Blemker et al. 2007; Delp et al. 2007; Mokhtarzadeh et al. 2013; Bruno et al. 2017; Halilaj et al. 2018; Mokhtarzadeh, Anderson, et al. 2021; Mokhtarzadeh, Forte, et al. 2021). To date, using these tools requires the installation of software packages on a local computer plus some computer-specific configurations that should be replicated if the simulation is to run elsewhere (Delp et al. 2007; de Zee et al. 2007). Application programming interfaces (APIs) are developed to help a wider community to utilize the functionalities of computational models. However, they still require the installation of software packages on a local computer and changes in configurations to set up for further analyses particularly when several people work on one project on different computers. Such challenges may create resistance among clinicians and end-users who may not be experts in programming and sometimes do not access the relevant software packages to run a biomechanical simulation. Moreover, it is of great importance to address clinical needs (Fregly 2021; Peng et al. 2021; Wang et al. 2021; Zaman et al. 2021). These may include, but are not limited to, the reproducibility of results and easily sharing the models (Erdemir et al. 2016, 2018, 2019; Perkel 2019; Fregly 2021) with minimal setups on a PC. Moreover, clinicians and technical NMSK developers can communicate better *via* a user-friendly platform to address limitations above.

CONTACT Hossein Mokhtarzadeh Mokhtarzadeh.hossein@gmail.com
Supplemental data for this article is available online at https://doi.org/10.1080/10255842.2022.2104607

Several biomechanical tools have been developed in the last few decades from the open-source or in-house packages including OpenSim (Delp et al. 2007), Anybody (Damsgaard et al. 2006), Toyota THUMS (Iwamoto et al. 2015), and Finite Element Models (Vychytil et al. 2014). However, their installation still should be done on a local computer. Though such an approach makes it challenging for training, reproducibility, validation, verification, and even sharing models in a wider community (Erdemir et al. 2016, 2018, 2019; Fregly 2021) if all the configuration needs to be replicated on every PC. Industrial engineering firms have been tackling these issues with their cloud-based solutions, though they are not always free and open source (e.g. Ansys Cloud, Abaqus, and MATLAB Cloud). To this end, open-source software packages installed on the web (e.g. Google Cloud) play a major role so that everyone can access them *via* the internet with minimal setups on their computer. One of these web-based platforms is Google Colaboratory (Colab), a Jupyter-based project, where Python programs can be written and executed on the web with the ability to use hypertext markup language (HTML). Thus, an online dynamic report can be accessible *via* Colab. Cloud-based computational platforms enable users to avoid the high cost of hardware and software while providing scalability for their products and models. Currently, there are no comprehensive cloud-based biomechanical tools available freely to all.

Therefore, this article aims to provide a set of freely available computational tools to address the above shortcomings regarding cross-platform installation, collaboration, validation, and reproducibility of biomechanical models in OpenSim. The aim of this article is twofold: a) to develop OpenColab, a project in which the OpenSim software was installed on the cloud, and b) to validate the outcomes of OpenSim analyses on the cloud with OpenSim's graphical user interface (GUI).

Methods

Overview

A Conda package was built to install OpenSim on Google Colab (i.e. a Linux-based platform) with only one block of code accessible *via* https://simtk.org/projects/opencolab/. Google Colab is suitable for online collaboration with team members by invitation. OpenColab is the online platform of OpenSim, just like how Google Docs is to Microsoft Word.

OpenSim libraries and Google cloud plus Anaconda cloud were used to solve several NMSK examples. Next, the OpenColab results were compared with OpenSim GUI results. These steps were performed without the installation of any software on a user's personal computer. We imported the prerequisites and dependencies *via* a web browser. Below, the details of how these procedures work are briefly explained. Many examples from validated OpenSim resources are presented. A step-by-step tutorial regarding the details of the installation process is in Supplementary materials 1 and 2. Following running the accompanying Jupyter notebook in Google Colab, the results of this article can be reproduced (Supplementary materials 1–4) or refer to https://simtk.org/projects/opencolab.

Developing OpenColab, i.e. OpenSim in Google colaboratory

This article focused on how to generate the latest Conda package for OpenSim and installed it in Google Colab. There are three common types of Conda packages (Windows, Linux-64, and Osx-64). However, to make it compatible with Colab, an Ubuntu desktop image (18.04) was required to generate a Linux-64-based Conda package (Supplementary material 1). Again, the end-users do not need to do any of these procedures to install OpenSim on the web. They can only use a block of code shared in this article to access all the OpenSim functionalities with no configurations on their PC.

To build the OpenSim in Conda, the following steps were performed. First, a virtual machine environment for Linux-64 was set up. Second, all the OpenSim dependencies were downloaded. A series of scripts was written to generate a Conda package. It was uploaded to the Anaconda cloud and then successfully tested it in the Colab. To build the Conda package, the following were needed: ∼50 GB storage and ∼10 GB memories. The whole process of Conda package development may take over 4–5 h. Again, this package has already been created, thus end-users need only to run a few lines of code to install OpenSim (nearly 1 GB) on the web in about 5–7 min. Finally, several examples were run, and the outcomes compared with OpenSim GUI. One can find the examples in this study in the notebook freely available at https://simtk.org/projects/opencolab/ (Figure 1 and 2).

Validation of OpenColab: OpenSim workflow in Google Colab

Step 1: Install OpenSim Package on Google Colab

The following script installed OpenSim on the cloud and printed the version installed (Figure 3).

Step 2: Upload OpenSim models and setup files

All relevant files and models were copied on the cloud from GitHub for further analyses (Figure 1). It should be noted that GitHub was used as an example for models and data storage. One can upload these files from their PC or any other hosted storage into the OpenColab (i.e. Google Cloud platform). It is also highly recommended to practice caution when dealing with sensitive/clinical data. Since these platforms presented in this proposal are based on other companies' platforms (e.g. Google and GitHub), the users need to read their security processes before uploading their data.

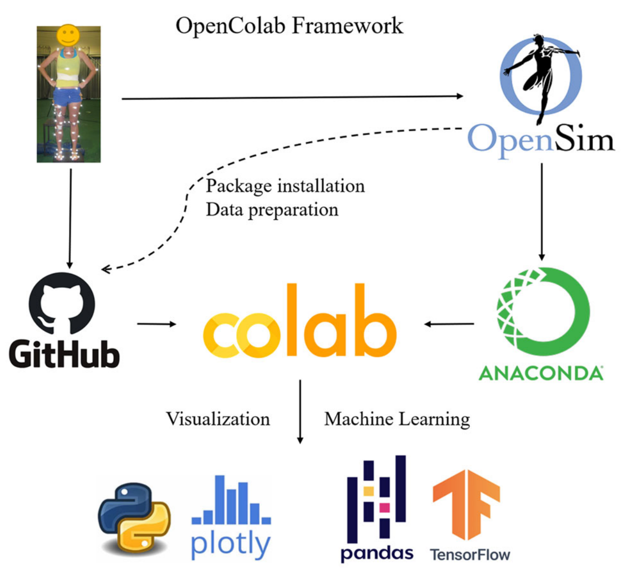

Figure 1. OpenColab Framework. Motion data is collected in the lab uploaded to the cloud, e.g. GitHub. This data can come from different sources such as IMUs, marker data, or RGB data, e.g. Kinect, etc. OpenSim libraries are installed in Google Colab *via* Anaconda Cloud. Using OpenSim libraries and motion data, one can perform a variety of analyses including scaling, inverse dynamic (ID), IK, SO, and CMC easily on the explorer and finally share their results and scripts with others. The visualization and future machine learning (e.g. TensorFlow) parts also can be done in Colab *via*, e.g. Plotly, Pandas package. The dashed line from OpenSim to GitHub shows how we imported GUI results into GitHub and later cloned them in Google Colab for validation with OpenColab outcomes.

Step 3: Explore an OpenSim model details, e.g. joints, bodies, and muscles (optional)

Following the installation of OpenSim, one first could check some details of a generic model using the following scripts (Figure 4). They included joints, bodies of the model, and muscles. This step was optional, however, it was recommended to make sure the right model was being used.

Step 4: Perform OpenSim Pipeline (Inverse Problem) in OpenColab

Finally, the following scripts were run (Figure 5) to perform the OpenSim workflow: scaling, inverse kinematics (IK), inverse dynamics (ID), residual reduction algorithm (RRA), static optimization (SO) and computed muscle control (CMC) to estimate muscle function (Seth et al. 2018).

The gait data is from OpenSim resources. The subject's mass and height were 72.6 kg, and 1.80 m, respectively. The gait2354 model was used for further analyses. Gait data and relevant OpenSim setup files were imported from GitHub. Data analyses and visualizations were performed using Plotly and Pandas packages (Figure 1 and Figure 2).

To scale a generic model, ScaleTool was imported from OpenSim libraries and ran it in Colab (Figure 2). The scaled model was stored in a predefined results folder based on the scaling setup file. Following scaling, IK in OpenSim provides optimized joint motions by minimizing a cost function based on anatomical markers in an experiment (Delp et al. 2007). The ID tool provides generalized joint

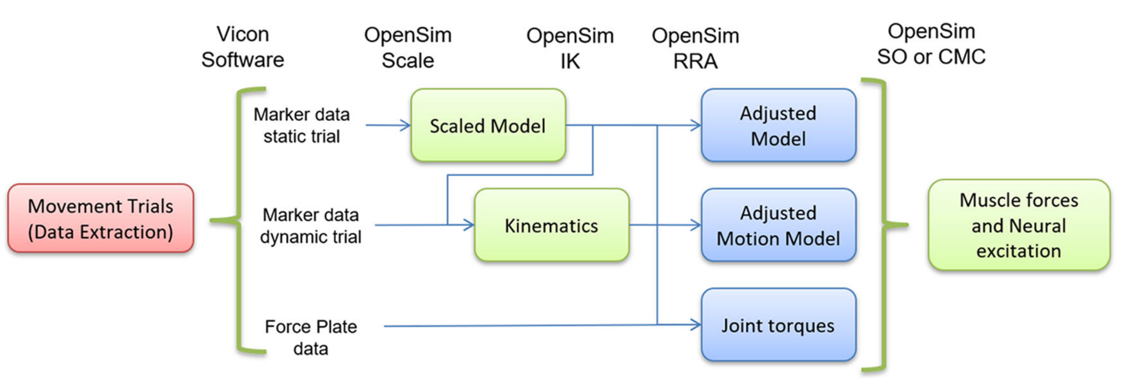

Figure 2. OpenSim inverse problem workflow. Data collection normally occurs in a lab. Then a generic model (e.g. gait2354) is adjusted, i.e. scaled to create a subject-specific model. Using scaled model and motion data, IK provides joint motions. RRA like ID calculated joint moments and provides a dynamically adjusted model. SO or CMC are used to estimate muscle forces or neural activities (Delp et al. 2007; Mokhtarzadeh 2013).

```
[ ] #Step 1: Install OpenSim package from link above & other dependency packages
    !wget -c https://repo.anaconda.com/miniconda/Miniconda3-py37_4.8.3-Linux-x86_64.sh
    !chmod +x Miniconda3-py37_4.8.3-Linux-x86_64.sh
    !bash ./Miniconda3-py37_4.8.3-Linux-x86_64.sh -b -f -p /usr/local
    import sys
    sys.path.append('/usr/local/lib/python3.7/site-packages')
    !conda install -y --prefix /usr/local -c ember123 opencolab
    import opensim as osim
    print('OpenSim Version Installed is version:',osim.__version__)
```

```
# Step 2: Upload OpenSim models, setup files
!git clone https://github.com/opensim-org/opensim-models.git
```

Figure 3. OpenSim installation on Google Colab (Step 1) and importation of OpenSim models and setup files from GitHub (Step 2).

```
#OpenSim model details e.g. joints, bodies, muscles
from opensim import Model
a = Model("/content/opensim-models/Pipelines/Gait2354_Simbody/gait2354_simbody.osim")
print("bodyset:")
for d in a.getBodySet():
    print("  " + d.getName())
print()
print("Jointset:")
for d in a.getJointSet():
    print("  " + d.getName())
print()
print("Forceset:")
for d in a.get_ForceSet():
    print("  " + d.getName())
```

Figure 4. Exploring the OpenSim's generic model and their elements. All the outputs can be confirmed and reproduced in the Supplementary material 1 notebook.

```
#Scaling
from opensim import ScaleTool
ScaleTool("./Gait2354_Simbody/subject01_Setup_Scale.xml").run()
```

```
[ ] #Inverse Kinematics (IK)
    from opensim import InverseKinematicsTool
    InverseKinematicsTool("./Gait2354_Simbody/subject01_Setup_IK.xml").run()
```

```
[ ] #Inverse Dynamics (ID)
    from opensim import InverseDynamicsTool
    InverseDynamicsTool("./Gait2354_Simbody/subject01_Setup_InverseDynamics.xml").run()
```

```
[ ] #Residual Reduction Algorithm (RRA)
    from opensim import RRATool
    RRATool("./Gait2354_Simbody/subject01_Setup_RRA.xml").run()
```

```
[ ] #Computed Muscle Control (CMC)
    from opensim import CMCTool
    CMCTool("./Gait2354_Simbody/subject01_Setup_CMC.xml").run()
```

Figure 5. OpenColab Python Scripts: Implementation of OpenSim workflow in Colab including performing scaling a generic model, IK, ID, RRA, and CMC using OpenSim tools called in OpenColab.

moments during a movement based on GRFs and joint motions from IK. Similarly, RRA adjusts the model for dynamic inconsistencies and provided joint angles (Figure 2). SO and CMC use optimization methods to estimate muscle activations using the RRA-adjusted model. These methods are consistent with those done in previous OpenSim studies (Delp et al. 2007; Seth et al. 2018). Briefly, OpenSim was installed online to solve an inverse problem in OpenSim *via* a web browser. OpenColab and GUI outcomes were compared (Figure 1) in Colab with Python version 3.7.13 (Python Software Foundation, https://www.python.org/) using Pearson Correlation analyses. Correlation coefficients greater than 0.8 were considered as very strong correlation based on the user's guide to correlation coefficients (Akoglu 2018). Finally, the results were presented in one gait cycle based on right leg foot strikes.

OpenColab: user's guide and tutorials

Several video tutorials were created on how to use OpenColab at this link: http://www.tinyurl.com/OpenColab. The reader could perform the following section first. To run this section and the IPython notebook, one requires only a Gmail account and internet connection to complete the next section.

One-minute OpenColab setup on the web

This process can be set up in less than one minute. Imagine a collaborator emails a Python file or IPython notebook (e.g. OpenColab.ipynb) to re-run OpenSim simulations and check the results. The reader may not access any software packages on their PCs. By following the below steps, the reader can start running OpenSim simulations in less than 1 min.

First download OpenColab.ipynb from https://simtk.org/projects/opencolab. Then, go to this website and complete the steps below. https://colab.research.google.com/:

a) Upload the following file from Supplementary material 3: '*OpenColab.ipynb*'
   a. The file can be downloaded from https://simtk.org/projects/opencolab too.
b) Wait till the file is loaded.
c) Press Ctrl + F9 or Runtime → Run all (setup finished in less than 1 min).
d) No action is needed by the user: OpenSim will be installed (5–7 min).
e) The simulations will generate the results of this article.

This process means zero configuration on a PC. This short video also illustrates the process: https://youtu.be/bDK2hxOHb4g.

Results

Table 1 shows the outcomes of scaling a generic model in OpenSim GUI and Colab. The results matched well where the total mean squared error was 0.04 and the difference between GUI and Colab was 1e–6. As shown in Figure 6, lower limb joint angles, and moments following IK, ID, and RRA matched quite perfectly in both approaches ($r \geq 0.98$, Table 2) according to the guide to correlation coefficients (Akoglu 2018).

Table 1. Differences between GUI and OpenColab following Scaling the generic model.

	Total squared error	RMS error	Max (Top. Head)
GUI	0.04	0.03	0.09
OpenColab	0.04	0.03	0.09
Absolute difference	1e–6	4e–7	4e–6

The results show quite perfect match.

Moreover, GUI and Colab outcomes were very strongly correlated for the lower limb muscles (Figure 7, Table 3) following CMC and SO analyses. Only Tibialis Anterior (TA) muscles showed $r = 0.83$ and rest of the muscles presented $r \geq 0.90$. The substantially high correlation between GUI and OpenSim results indicated validation of OpenColab. The reader can replicate the outcomes using the notebook provided with this article (Supplementary material 2).

Discussion

OpenColab, a simple framework to use OpenSim on the cloud *via* a web browser, has been developed with a minimal package installation and zero configuration on a PC. The framework introduced in this article successfully installed the latest official OpenSim package on Google Colab and presented several NMSK modeling examples by running OpenSim tools, which will benefit researchers and clinicians in a collaborative manner. The examples can be modified and run on the web. Though there have been some attempts to install OpenSim on Conda, none has been published or implemented on Colab. The results indicated that OpenSim GUI and OpenColab' results are highly correlated when OpenSim tools were performed including scaling, IK, ID, RRA, SO, and CMC. Correlation coefficients between GUI and OpenColab ranged from very strong to perfect and a minor difference in TA muscle force estimation ($r > 0.83$) could be explained by computational procedures toward optimized solutions following SO in GUI (C++-based) *versus* OpenColab (Python-based). The root cause may stem from a slight difference after rounding the results, e.g. in scaling, IK, and ID analyses which are the inputs of SO and CMC. The validated results of this study indicate the usability of OpenColab in the Biomechanics community. OpenColab can be easily used by end-users to share

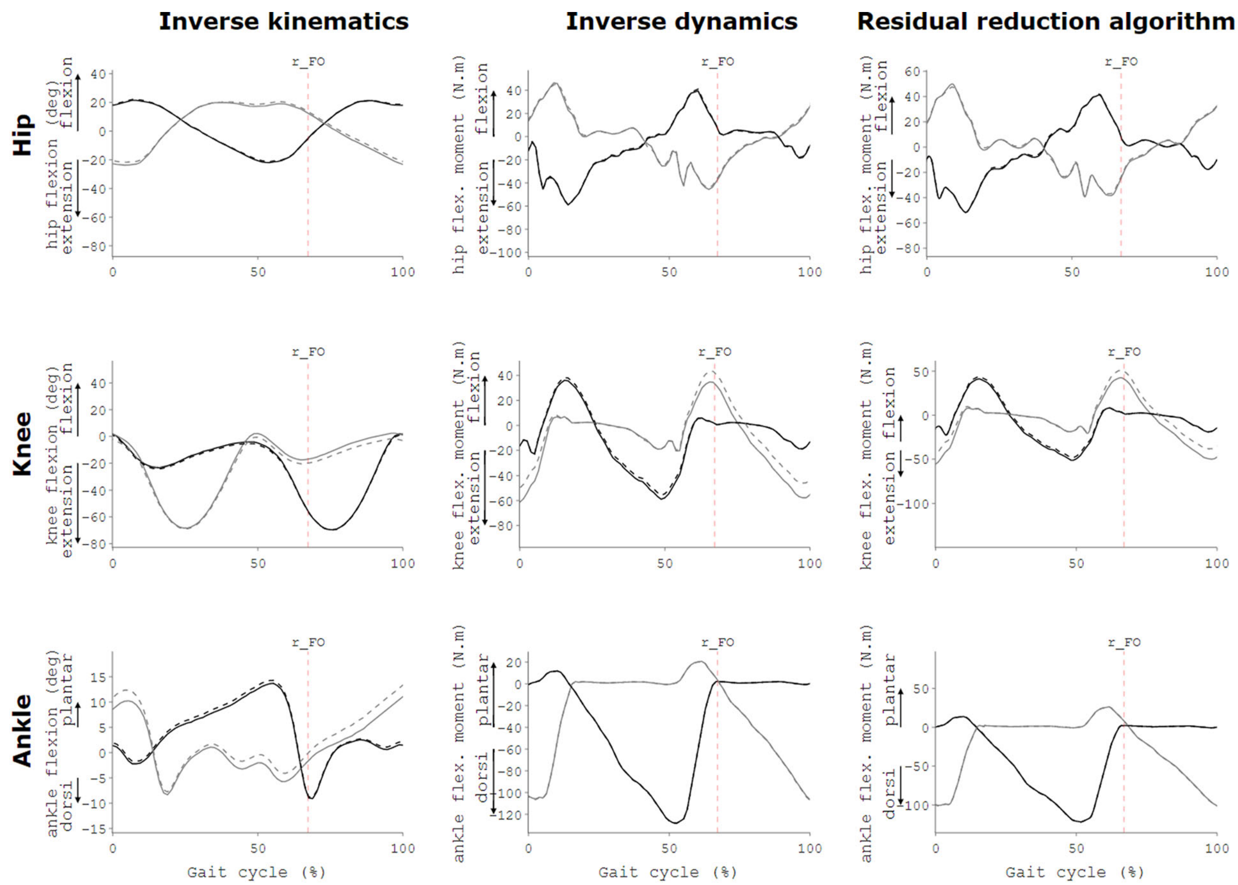

Figure 6. Major lower limb joint kinematics and kinetics calculations using OpenColab and OpenSim GUI. r_colab (solid, black): right side in colab. r_GUI(dashed, black): right side in OpenSim GUI. l_colab (solid, grey), l_GUI (dashed, grey).

Table 2. Correlation coefficients between GUI and Colab results for IK, ID, and RRA analyses for hip, knee, and ankle joints.

GUI/Colab	Sides	IK	ID	RRA
Hip	R	0.99	0.99	0.99
	L	0.99	0.99	0.99
Knee	R	0.99	0.99	0.99
	L	0.99	0.99	0.98
Ankle	R	0.99	1.00	0.99
	L	0.99	0.99	0.99

L: left side; R: right side

and collaborate on NMSK models on the cloud with minimal setup and is accessible *via* https://simtk.org/projects/opencolab/.

What is the added value of OpenColab?

OpenSim has great interfaces with MATLAB and Python with thousands of users. OpenColab, which is OpenSim in Google Colab, is not a replacement or superior, but a complementary interface that only requires a Gmail account and internet access and zero configuration on the user's PC. The Conda package built in this study allows the users to install OpenSim on Google Colab with just one block of code in less than 7 min. The initial setup can take less than a minute. OpenColab takes advantage of the Google Colab platform where machine learning (or artificial intelligent) studies can be performed too (Perkel 2019; Kidziński et al. 2020; Boswell et al. 2021; Ballit and Dao 2022; Ileşan et al. 2022; Low et al. 2022; Vallejo et al. 2022). The Google Colab enables users to collaborate with their team members by invitation. As mentioned above, one can compare it with online platform of Google Docs *versus* Microsoft Word on a PC. Correspondingly, OpenColab, an online platform, was built *versus* other APIs, on a local PC. Moreover, different platform users can install OpenColab on their web browsers. A qualitative comparison between different OpenSim APIs was provided in Supplementary material 4.

OpenColab provides the first comprehensive framework where one can easily implement NMSK model without the need to install any software on a PC. It can accept motion capture data and eventually can use IMUs, and other OpenSim modules thus addressing numerous clinical needs. Additionally, since Colab accepts both HTML input and scripting

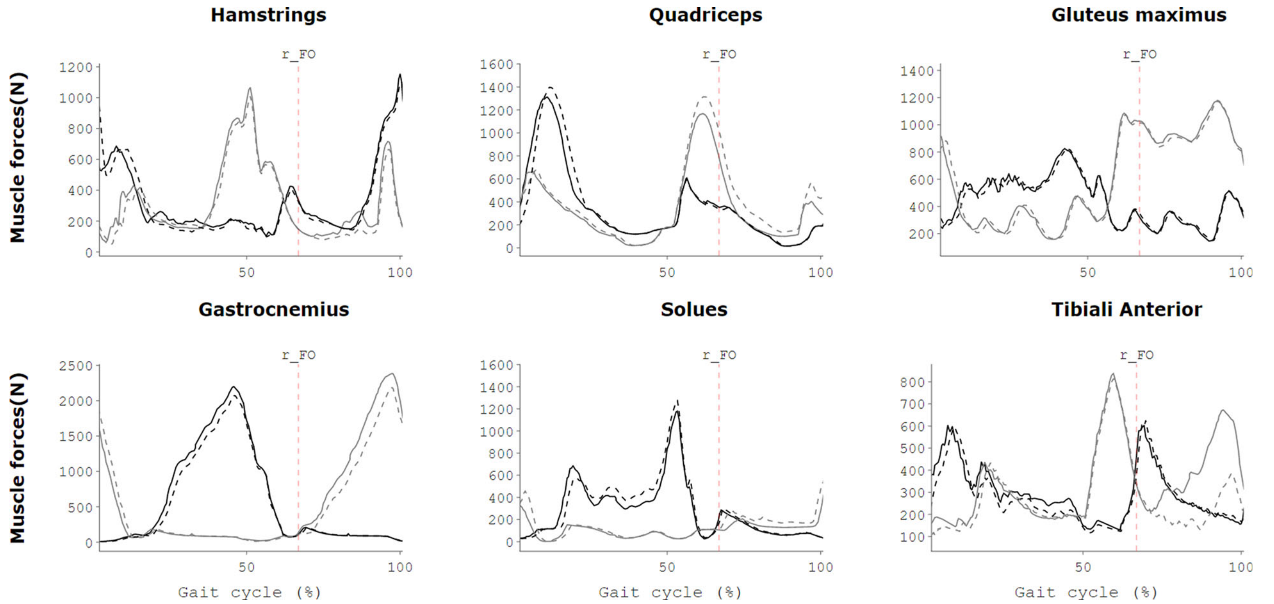

Figure 7. Major lower limb muscle force estimations using OpenColab and OpenSim GUI. This is a gait cycle for the right leg (0–100%). r_FO stands for right foot strike event. r_colab (solid, black): right side in colab. r_GUI (dashed, black): right side in OpenSim GUI. l_colab (solid, grey), l_GUI (dashed, grey).

Table 3. Correlation coefficients between GUI and Colab results for SO, and CMC analyses for major lower limb muscles.

GUI/Colab	Sides	SO	CMC
Hamstring	R	0.99	0.95
	L	0.99	0.92
Quadriceps	R	0.99	0.98
	L	0.99	0.97
Glute	R	0.99	0.97
	L	0.99	0.97
Gastrocnemius	R	0.99	0.99
	L	0.99	0.98
Soleus	R	0.99	0.96
	L	0.90	0.91
Tibialis anterior	R	0.99	0.92
	L	0.88	0.83

L: left side; R: right side

in Python online, researchers and clinicians can access online 'dynamic reports' where the results of analyses can be easily reproduced, and even new findings can be explored in a collaborative manner.

OpenColab can be applied in machine learning (ML) studies, education (Nelson and Hoover 2020; Canesche et al. 2021; Vallejo et al. 2022), control and AI (Freeman et al. 2021; Vittorio et al. 2022), and even motion analyses (Mathis et al. 2018; Nath et al. 2019). Though OpenColab is the first framework to run NMSK models on the web, other researchers implemented their methods on Colab in various areas. For instance, Google has shared its TensorFlow-based ML process on Colab where data, codes, and access to GUPs are available *via* Colab (e.g. https://www.tensorflow.org/resources/learn-ml). OpenColab has similar potential so that users can share their findings easily *via* Colab and their results can be reproducible in the browser. Furthermore, this article is the first of its kind to compare OpenSim GUI with Colab results and validated the new framework of OpenColab. However, the new era of cloud computing using, e.g. rigid body simulation on the web is only in its infancy (Freeman et al. 2021; Vittorio et al. 2022) and huge opportunities for using such frameworks lie ahead of us as researchers, and clinicians.

Limitations

There are several limitations in developing the OpenColab framework. First, Google Colab is improving with many users across the globe. But its 3D visual performance is not yet up to the speed of a PC. Thus, 3D visualization of a simulation can be the future direction of research. However, the users might soon have access to a vast library of 3D visualization with the advancement of visual TensorFlow. Second, the Google Colab allows 12 h of maximum lifetimes for a connection. But the premium version of Google Colab permits longer runtimes. Using Google Colab still needs some programming skills. Nevertheless, the authors aim to build user-friendly GUIs within Google Colab so that beginners can benefit from using the OpenColab framework. Finally, debugging a code in Colab is still in its early stages, but more features are being built within the Colab by the community and developers.

Future research directions

The next steps involve improving OpenColab with several new features. These include the design of a Windows version package, visualization of 3D models and outcomes, automated OpenSim workflows (e.g. from data collection to data analyses), optimized use of Google Colab's vast machine learning libraries, such as TensorFlow, and importation of Imaging data (e.g. MRI, CT scans) into OpenSim models (Bruno et al. 2017; Anderson et al. 2020; Mokhtarzadeh, Anderson, et al. 2021; Mokhtarzadeh, Forte, et al. 2021). Moreover, OpenColab can be used for training and educational purposes as an interactive NMSK computing platform for several disciplines interested in human movement science including biomechanics researchers, clinicians, sports scientists, occupational therapists, and robotics researchers. Finally, using OpenColab, the reviewers of neuro-musculoskeletal modeling can access all the codes and analyses to reproduce the results (Kidziński et al. 2020; Boswell et al. 2021).

Conclusion

Neuro-musculoskeletal modeling on the cloud is possible with the current technologies. This article presented how to install OpenSim on the cloud and develop/run biomechanical analyses on the web with minimal setups (i.e. a Gmail account and internet access only) with zero configuration on a personal computer. Such a platform can open new avenues in training, developing, sharing, and reproducing biomechanical models. These abilities are crucial among end-users. They may include clinicians, sports scientists, occupational therapists, and robotics researchers.

Acknowledgments

We would like to thank our previous interns and especially Ms. Song for their contributions and the wonderful and fruitful discussions with Dr. Ayman Habib (Lead Software Architect, OpenSim team at Stanford University).

Disclosure statement

No potential conflict of interest was reported by the author(s).

Funding

The author(s) reported there is no funding associated with the work featured in this article.

ORCID

Hossein Mokhtarzadeh http://orcid.org/0000-0002-6519-7735
Fangwei Jiang http://orcid.org/0000-0003-2336-3261

References

Akoglu H. 2018. User's guide to correlation coefficients. Turk J Emerg Med. 18(3):91–93.

Anderson D, Mokhtarzadeh H, Allaire B, Burkhart K, Bouxsein M. 2020. Subject-specific spine models for 250 individuals from the Framingham Heart Study [Internet]. V1. https://dataverse.harvard.edu/dataverse/SpineModeling.

Ballit A, Dao TT. 2022. Recurrent neural network to predict hyperelastic constitutive behaviors of the skeletal muscle. Med Biol Eng Comput. 60(4):1177–1185.

Blemker SS, Asakawa DS, Gold GE, Delp SL. 2007. Image-based musculoskeletal modeling: applications, advances, and future opportunities. J Magn Reson Imaging. 25(2): 441–451.

Boswell MA, Uhlrich SD, Kidziński Ł, Thomas K, Kolesar JA, Gold GE, Beaupre GS, Delp SL. 2021. A neural network to predict the knee adduction moment in patients with osteoarthritis using anatomical landmarks obtainable from 2D video analysis. Osteoarthritis Cartilage. 29(3):346–356.

Bruno AG, Mokhtarzadeh H, Allaire BT, Velie KR, Kaluza MCDP, Anderson DE, Bouxsein ML. 2017. Incorporation of CT-based measurements of trunk anatomy into subject-specific musculoskeletal models of the spine influences vertebral loading predictions. J Orthop Res. 35(10): 2164–2173.

Canesche M, Bragança L, Neto OPV, Nacif JA, Ferreira R. 2021. Google colab cad4u: hands-on cloud laboratories for digital design. IEEE International Symposium on Circuits and Systems. Piscataway (NJ): IEEE; p. 1–5.

Damsgaard M, Rasmussen J, Christensen ST, Surma E, De Zee M. 2006. Analysis of musculoskeletal systems in the anybody modeling system. Simul Model Pract Theory. 14(8):1100–1111.

Delp SL, Anderson FC, Arnold AS, Loan P, Habib A, John CT, Guendelman E, Thelen DG. 2007. OpenSim: open-source software to create and analyze dynamic simulations of movement. IEEE Trans Biomed Eng. 54(11): 1940–1950.

Erdemir A, Besier TF, Halloran JP, Imhauser CW, Laz PJ, Morrison TM, Shelburne KB. 2019. Deciphering the "art" in modeling and simulation of the knee joint: overall strategy. J Biomech Eng. 141(7):0710021–07100210.

Erdemir A, Guess TM, Halloran JP, Modenese L, Reinbolt JA, Thelen DG, Umberger BR, Erdemir A, Guess TM, Halloran JP, et al. 2016. Commentary on the integration of model sharing and reproducibility analysis to scholarly publishing workflow in computational biomechanics. IEEE Trans Biomed Eng. 63(10):2080–2085.

Erdemir A, Hunter PJ, Holzapfel GA, Loew LM, Middleton J, Jacobs CR, Nithiarasu P, Löhner R, Wei G, Winkelstein BA. 2018. Perspectives on sharing models and related resources in computational biomechanics research. J Biomech Eng. 140(2):0247011–02470111.

Freeman CD, Frey E, Raichuk A, Girgin S, Mordatch I, Bachem O. 2021. Brax–A differentiable physics engine for large scale rigid body simulation. arXiv preprint arXiv2106.13281.

Fregly BJ. 2021. A conceptual blueprint for making neuromusculoskeletal models clinically useful. Appl Sci. 11(5): 2037.

Halilaj E, Rajagopal A, Fiterau M, Hicks JL, Trevor J, Delp SL. 2018. Machine learning in human movement biomechanics: best practices, common pitfalls, and new opportunities. J Biomech. 81:1–11.

Ileșan RR, Cordoș CG, Mihăilă LI, Fleșar R, Popescu A-S, Perju-Dumbravă L, Faragó P. 2022. Proof of concept in artificial-intelligence-based wearable gait monitoring for Parkinson's disease management optimization. Biosensors. 12(4):189.

Iwamoto M, Nakahira Y, Kimpara H. 2015. Development and validation of the total human model for safety (THUMS) toward further understanding of occupant injury mechanisms in precrash and during crash. Traffic Inj Prev. 16(1):S36–S48.

Kidziński Ł, Yang B, Hicks JL, Rajagopal A, Delp SL, Schwartz MH. 2020. Deep neural networks enable quantitative movement analysis using single-camera videos. Nat Commun. 11(1):1–10.

Low WS, Goh KY, Goh SK, Yeow CH, Lai KW, Goh SL, Chuah JH, Chan CK. 2022. Lower extremity kinematics walking speed classification using long short-term memory neural frameworks. Multimed Tools Appl. 1–16.

Mathis A, Mamidanna P, Cury KM, Abe T, Murthy VN, Mathis MW, Bethge M. 2018. DeepLabCut: markerless pose estimation of user-defined body parts with deep learning. Nat Neurosci. 21(9):1281–1289.

Mokhtarzadeh H, Anderson DE, Allaire BT, Bouxsein ML. 2021. Patterns of load-to-strength ratios along the spine in a population-based cohort to evaluate the contribution of spinal loading to vertebral fractures. J Bone Miner Res. 36(4):704–711.

Mokhtarzadeh H, Forte JD, Lee PVS. 2021. Biomechanical and cognitive interactions during visuo motor targeting task. Gait & Posture, 86:287–291.

Mokhtarzadeh H, Yeow CH, Hong Goh JC, Oetomo D, Malekipour F, Lee PVS. 2013. Contributions of the Soleus and Gastrocnemius muscles to the anterior cruciate ligament loading during single-leg landing. J Biomech. 46(11):1913–1920.

Nath T, Mathis A, Chen AC, Patel A, Bethge M, Mathis MW. 2019. Using DeepLabCut for 3D markerless pose estimation across species and behaviors. Nat Protoc. 14(7):2152–2176.

Nelson MJ, Hoover AK. 2020. Notes on using google colaboratory in AI education. In Proceedings of 2020 ACM Conference on Innovation and Technology in Computer Science Education. p. 533–534.

Peng MJ, Ju X, Ma L, Hu Y, Li X. 2021. Dynamics analysis for flexion and extension of elbow joint motion based on musculoskeletal model of anybody. Int J Med Robot Comput Assist Surg. 17(6):e2321.

Perkel JM. 2019. Make code accessible with these cloud services. Nature. 575(7781):247–249.

Seth A, Hicks JL, Uchida TK, Habib A, Dembia CL, Dunne JJ, Ong CF, DeMers MS, Rajagopal A, Millard M, et al. 2018. OpenSim: simulating musculoskeletal dynamics and neuromuscular control to study human and animal movement. PLoS Comput Biol. 14(7):e1006223.

Vallejo W, Díaz-Uribe C, Fajardo C. 2022. Google colab and virtual simulations: practical e-learning tools to support the teaching of thermodynamics and to introduce coding to students. ACS Omega. 7(8):7421–7429.

Vittorio C, Huawei W, Guillaume D, Massimo S, Vikash K. 2022. Fast and physiologically realistic MuJoCo models for musculoskeletal and exoskeletal studies. In 2022 International Conference on Robotics and Automation (ICRA). IEEE. p. 8104–8111.

Vychytil J, Manas J, Cechova H, Spirk S, Hyncik L, Kovar L. 2014. Scalable multi-purpose virtual human model for future safety assessment. SAE Technical Paper 2014-01-0534.

Wang M, Li S, Teo EC, Fekete G, Gu Y. 2021. The influence of heel height on strain variation of plantar fascia during high heel shoes walking-combined musculoskeletal modeling and finite element analysis. Front Bioeng Biotechnol. 9:791238.

Zaman R, Xiang Y, Rakshit R, Yang J. 2021. Hybrid predictive model for lifting by integrating skeletal motion prediction with an OpenSim musculoskeletal model. IEEE Trans Biomed Eng. 69:1111–1122.

de Zee M, Hansen L, Wong C, Rasmussen J, Simonsen EB. 2007. A generic detailed rigid-body lumbar spine model. J Biomech. 40(6):1219–1227.