The Journal of Open Source Software

Automatic Computation for Robot Design (ACRoD): A Python package for numerical calculation of Jacobian of a robot at a given configuration around a specified end-effector point

Akkarapakam Suneesh Jacob [1*¶] **and Rituparna Datta** [2*]

1 Indian Institute of Technology Kanpur, Kanpur, India. **2** Capgemini Technological Services India Limited, Bengaluru, India. ¶ Corresponding author * These authors contributed equally.

DOI: 10.21105/joss.05927

Software

- Review
- Repository
- Archive

Editor: Kevin M. Moerman

Reviewers:

- @gfadini
- @CameronDevine
- @JHartzer

Submitted: 24 May 2023
Published: 21 March 2024

Summary

The Jacobian of a robot refers to the matrix that linearly maps the velocity components of the end-effector and the velocities at the actuated joints. The Jacobian is extensively used in dimensional synthesis for Jacobian-based optimal performances of robotic manipulators, in which the optimal dimensional parameters of robots are computed. Determination of accurate mobility (Yang et al., 2008) of planar and spatial mechanisms can also be performed by using Jacobian in cases where Chebychev–Grübler–Kutzbach criterion cannot accurately determine the mobility (Gogu, 2005). As a result, Jacobian is a significant part for both kinematic analysis, dimensional synthesis and mobility determination of a mechanism. Hence, the formulation of Jacobian has its key importance in the literature and in the application of performance optimisation along with mobility computation. Formulation of Jacobian for serial manipulators can be computed easily, however, it is increasingly complicated to formulate Jacobian for parallel manipulators due to the existence of passive joint velocities and the nature in which these are related to active joint velocities. Several studies (Altuzarra et al., 2006; Dutre et al., 1997; D. Kim et al., 2000; S.-G. Kim & Ryu, 2003) exist for formulation of parallel manipulators but all these studies are dependent on human inspection at some level. Several open source software packages are also available for Jacobian formulation (Baumgärtner & Miller, 2022; Lee et al., 2018; Nadeau, 2019; Owan et al., 2018), but either their application is limited to serial manipulators, they require human intervention, or they are part of computationally expensive simulations. For example, TriP (Baumgärtner & Miller, 2022) does facilitate Jacobian computation with closed-loop chains in the manipulator structure, however, as obvious from the tripedal robot example in the documentary, human inspection is apparently required to appropriately join the legs of the robot. To alleviate the drawback of the requirement of human inspection, the present research aims to formulate Jacobian that is required for dimensional synthesis for optimal performance around a single point that can be used for any non-redundant manipulator without any dependency on human inspection. Jacob and Dasgupta (Jacob & Dasgupta, 2022) used a systematic method as a tool to formulate Jacobian matrices for several manipulators in bulk for performance optimisation around a given task point. However, several steps in that algorithm are not totally computerised but rather human-intervention-dependent. Moreover, it can be applicable only with four types of joints. The current paper extends their method to present a fully computerisable Jacobian formulation algorithm that is applicable for general non-redundant planar and spatial manipulators of any topological structure of seven types of joints, namely revolute, prismatic, cylindrical, spherical, universal, helical and plane joints. Based on this extended method, the Python package `Automatic Computation for Robot Design (ACRoD)` is developed by the authors. `ACRoD` provides a Python-based package for generating functions required to compute the Jacobian at a given configuration for a

given end-effector point, merely from the simple topological information of the robot in a fully automated manner. This can be directly used in optimisation process to derive optimal dimensions of the robot for optimal performance around a given end-effector point, thereby avoiding many tedious steps in manual formulation, especially when a comparison study is performed on multiple manipulators in bulk (Jacob & Dasgupta, 2022). `ACRoD` uses NumPy (Harris et al., 2020) and SymPy (Meurer et al., 2017) packages to generate the functions for Jacobian, which can be directly used in optimisation process to find the optimal dimensional parameters of the robot.

Statement of need

For a manipulator of a given topology, designing the dimensions based on optimising Jacobian-based performance parameters (such as manipulability index and condition number (Patel & Sobh, 2015)) around a given end-effector point would require only the topological information for the formulation of Jacobian, as every other step can be automated. Formulation of Jacobian for parallel manipulators and serial-parallel hybrid manipulators are non-trivial, although all the steps of Jacobian formulation even in those cases would have to stem from the mere information of topology of the robot. ACRoD automates the non-trivial formulation of Jacobian systematically. It uses a matrix-based representation of the topology of the robotic manipulator (referred to here as the robot-topology matrix, of which more information is provided here) which is a modified version of the graph adjacency matrix representation (Jacob et al., 2022) of robotic manipulators. This Jacobian formulation can be used to generate numerical Jacobian matrices with a few random configurations, from which the singular values can be calculated which can confirm the Degree of Freedom (DoF). In other words, the DoF of a given robot topology for a given base link and a given end-effector link can be verified by using this Jacobian function even in cases where Chebychev–Grübler–Kutzbach criterion fails to verify. This can be useful in mechanism synthesis to accurately verify the mobility of a given manipulator directly from its robot-topology matrix by using the method shown in Yang et al.'s paper (Yang et al., 2008). Even though the repository does not include scripts for computing optimisation or performance metrics for dimensional synthesis or for computing the validity of DOF, it does include scripts for computation of Jacobian which is an essential part for these analyses.

Method

The topology of a valid robot (with a single base-link and a single end-effector link and without non-contributing chains) is to be specified using robot-topology matrix in NumPy matrix format. The Jacobian class object takes this robot-topology matrix as input argument and generates functions that are required to compute Jacobian. As byproducts, the Jacobian function generation produces the symbolic matrices, the set of independent paths, etc., the sets of active joint velocities and passive joint velocities, etc., which can be accessed from the attributes of the Jacobian class object. More technical details on formulation of Jacobian (along with appropriate algorithms) can be found here, and the notations and the nomenclature are explained here in detail. The robot-topology matrix representation is explained here in detail. Jacobian formulation for three robot examples, namely the 3R planar serial robot, a 4R-4P planar serial-parallel hybrid robot and an RSSR-SSR spatial parallel robot, are explained in detail in the corresponding hyperlinks.

Comparison with other Jacobian-computation software packages

TriP (Baumgärtner & Miller, 2022) is a software that is developed to address kinematics of hybrid linkages. However, it is evident from the tripedal example that human inspection is required to develop the model for each leg of tripedal robot. DART (Lee et al., 2018) uses biped robot (which can be seen as involving a closed-loop mechanism) in the examples, however it is imported from .sktl file rather than modelling a customised closed-loop robot from scratch.

Even though C++ based software packages such as DART (Lee et al., 2018), CoreRobotics (Owan et al., 2018) and pinocchio (Carpentier et al., 2019) may facilitate closed-loop linkages, apparently no documentation is provided for modelling closed-loop linkages. Furthermore, all these software (TriP, DART, CoreRobotics, pinocchio) apparently require human intervention to build a robot of a given topology. Even though some of them support importing models from URDF files, URDF has the limitation of "inability to model parallel linkages and closed-chain systems" (Tola & Corke, 2023), and furthermore preparing a URDF file (or similar file) of a robot from its mere topological information also requires human intervention. Pybotics provides automatic modelling of robot from the mere information of DH parameters, however it is apparently limited to serial manipulators. ACRoD addresses this issue of human intervention by automatically generating functions required to compute Jacobian for a given end-effector point. A comparison of ACRoD with other software packages is shown in the table below.

Software	Base Language	Closed-loop Linkages	Automation Level From Mere Topology	Primary focus
TriP (2022)	Python	yes	H*	Kinematics of Hybrid Linkages
Pybotics (2019)	Python	no	A	Kinematics, Dynamics, Trajectory Generations and Calibration of Serial Robots
DART (2018)	C++	X	H*	Kinematic and Dynamic Applications of Robotics
CoreRobotics (2018)	C++	X	H	Computational Algorithms for Real Time Robot Control
pinocchio (2019)	C++	X	H*	Analytical Derivatives for Kinematics and Dynamics, Features for Control, Planning and Simulation
ACRoD (2023)	Python	yes	yes	Dimensional Synthesis

- X = possible but neither documentation is provided nor an example is provided.
- * = Accepts importing models from URDF files, etc.
- H = Human intervention required to build the robot.
- A = Automatically buildable from its DH parameters.

Acknowledgements

This paper is an extended version of the Jacobian formulation presented in previous work (Jacob & Dasgupta, 2022) co-authored by Dr. Bhaskar Dasgupta, whom the authors would like to acknowledge for whatever credits applicable. However, the authors would like to clarify that the technical extension of theory and the whole software are developed solely by the authors of the current paper.

References

Altuzarra, O., Salgado, O., Petuya, V., & Hernández, A. (2006). Point-based jacobian formulation for computational kinematics of manipulators. *Mechanism and Machine Theory*, *41*(12), 1407–1423. https://doi.org/10.1016/j.mechmachtheory.2006.01.011

Baumgärtner, J., & Miller, T. (2022). TriP: A python package for the kinematic modeling of serial-parallel hybrid robots. *Journal of Open Source Software*, *7*(71), 3967. https://doi.org/10.21105/joss.03967

Carpentier, J., Saurel, G., Buondonno, G., Mirabel, J., Lamiraux, F., Stasse, O., & Mansard, N. (2019). The pinocchio c++ library : A fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives. *2019 IEEE/SICE International Symposium on System Integration (SII)*, 614–619. https://doi.org/10.1109/SII.2019.8700380

Dutre, S., Bruyninckx, H., & De Schutter, J. (1997). The analytical jacobian and its derivative for a parallel manipulator. *Proceedings of International Conference on Robotics and Automation*, *4*, 2961–2966 vol.4. https://doi.org/10.1109/robot.1997.606737

Gogu, G. (2005). Chebychev–Grübler–Kutzbach's criterion for mobility calculation of multi-loop mechanisms revisited via theory of linear transformations. *European Journal of Mechanics - A/Solids*, *24*(3), 427–441. https://doi.org/10.1016/j.euromechsol.2004.12.003

Harris, C. R., Millman, K. J., Walt, S. J. van der, Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., Kerkwijk, M. H. van, Brett, M., Haldane, A., Río, J. F. del, Wiebe, M., Peterson, P., … Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

Jacob, A. S., & Dasgupta, B. (2022). *Dimensional synthesis of spatial manipulators for velocity and force transmission for operation around a specified task point*. https://doi.org/10.48550/arXiv.2210.04446

Jacob, A. S., Dasgupta, B., & Datta, R. (2022). *Enumeration of spatial manipulators by using the concept of adjacency matrix*. arXiv. https://doi.org/10.48550/arXiv.2210.03327

Kim, D., Chung, W., & Youm, Y. (2000). Analytic jacobian of in-parallel manipulators. *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, *3*, 2376–2381 vol.3. https://doi.org/10.1109/robot.2000.846382

Kim, S.-G., & Ryu, J. (2003). New dimensionally homogeneous jacobian matrix formulation by three end-effector points for optimal design of parallel manipulators. *IEEE Transactions on Robotics and Automation*, *19*(4), 731–736. https://doi.org/10.1109/tra.2003.814496

Lee, J., Grey, M. X., Ha, S., Kunz, T., Jain, S., Ye, Y., Srinivasa, S. S., Stilman, M., & Liu, C. K. (2018). DART: Dynamic animation and robotics toolkit. *Journal of Open Source Software*, *3*(22), 500. https://doi.org/10.21105/joss.00500

Meurer, A., Smith, C. P., Paprocki, M., Čertík, O., Kirpichev, S. B., Rocklin, M., Kumar, A., Ivanov, S., Moore, J. K., Singh, S., Rathnayake, T., Vig, S., Granger, B. E., Muller, R. P., Bonazzi, F., Gupta, H., Vats, S., Johansson, F., Pedregosa, F., … Scopatz, A. (2017). SymPy: Symbolic computing in python. *PeerJ Computer Science*, *3*, e103. https://doi.org/10.7717/peerj-cs.103

Nadeau, N. (2019). Pybotics: Python toolbox for robotics. *Journal of Open Source Software*, *4*(41), 1738. https://doi.org/10.21105/joss.01738

Owan, P., Devine, C., & Piaskowy, W. T. (2018). CoreRobotics: An object-oriented c++ library with cross-language wrappers for cross-platform robot control. *Journal of Open Source Software*, *3*(22), 489. https://doi.org/10.21105/joss.00489

Patel, S., & Sobh, T. (2015). Manipulator performance measures-a comprehensive literature survey. *Journal of Intelligent & Robotic Systems*, *77*(3), 547–570. https://doi.org/10.1007/s10846-014-0024-y

Tola, D., & Corke, P. (2023). Understanding URDF: A survey based on user experience. *2023 IEEE 19th International Conference on Automation Science and Engineering (CASE)*, 1–7. https://doi.org/10.1109/CASE56687.2023.10260660

Yang, D.-C., Xiong, J., & Yang, X.-D. (2008). A simple method to calculate mobility with jacobian. *Mechanism and Machine Theory*, *43*(9), 1175–1185. https://doi.org/10.1016/j.mechmachtheory.2007.08.001