The Journal of Open Source Software

Pycapacity: a real-time task-space capacity calculation package for robotics and biomechanics

Antun Skuric [1¶], **Vincent Padois**[1], **and David Daney**[1]

1 INRIA, Bordeaux, France ¶ Corresponding author

DOI: 10.21105/joss.05670

Software

- Review
- Repository
- Archive

Editor: Dana Solav

Reviewers:

- @sea-bass
- @ShravanTata
- @JHartzer

Submitted: 23 May 2023
Published: 11 September 2023

Summary

This paper presents a Python package called `pycapacity`, which provides a set of tools for evaluating task-space physical ability metrics for humans and robots, based on polytopes and ellipsoids. The aim of `pycapacity` is to provide a set of efficient tools for their evaluation in an easy to use framework that can be easily integrated with standard robotics and biomechanics libraries. The package implements several state of the art algorithms for polytope evaluation that bring many of the polytope metrics to the few milliseconds evaluation time, making it possible to use them in online and interactive applications.

The package can be easily interfaced with standard libraries for robotic manipulator rigid body simulation such as `robotictoolbox` (Corke & Haviland, 2021) or `pinocchio` (Carpentier et al., 2019), as well as human musculoskeletal model biomechanics software `opensim` (Delp et al., 2007) and `biorbd` (Michaud & Begon, 2021). The package can also be used with the Robot Operating System (`ROS`) (Quigley et al., 2009).

The package additionally implements a set of visualization tools for polytopes and ellipsoids based on the Python package `matplotlib` intended for fast prototyping and quick and interactive visualization.

Statement of need

There is a rising interest in collaborative robotics and physical human robot interaction, where the robots are often required to adapt to certain needs of the human in real-time. This adaptation raises a fundamental challenge: the ability to evaluate the need of assistance of the operator. One way to quantify the need of assistance is by evaluating the operator's physical abilities in real-time and comparing them to the physical abilities required to execute the task. Having this real-time information enables creating collaborative robot control strategies that assist the operators by compensating for their lacking physical ability to accomplish the tasks.

Beyond the characterization of human physical capabilities, as today's collaborative robotic manipulators are designed for safety, their performance characteristics are relatively limited with respect to the more standard industrial robots. Therefore it is becoming increasingly important to exploit their full (physical) abilities when executing their task.

There are many different metrics available in the literature that might be used to characterize physical abilities: force capacity, velocity capacity, acceleration capacity, accuracy, stiffness etc. Most of these metrics can be represented by two families of geometric shapes: ellipsoids (Yoshikawa, 1985) and polytopes (Chiacchio et al., 1996). These metrics are traditionally important tools for off-line analysis purposes (workspace design, human motion and ergonomics analysis) and recently, they have shown a great potential to be used for interactive online applications, to be integrated in robot control strategies or as a visual feedback to the operator.

Ellipsoid metrics are often used for evaluating the manipulability of the robot's end-effector. The manipulability ellipsoid is a geometric shape that represents the robot's ability to move within the task-space. Due to their computational efficiency and intuitive visualization, they have been used in many different applications, such as robot control, workspace design, robot design, etc. Therefore, there are several open-source packages that implement the manipulability ellipsoid evaluation and visualization, such as `MMC` (Haviland & Corke, 2020), `manipulability_metrics` (Prada, 2023), `Manipulability` (N. Jaquier et al., 2021; Noémie Jaquier, 2023). However, most of these packages are limited to the evaluation of the manipulability ellipsoid, representing the velocity capacity, and they do not provide tools for evaluating other ellipsoid metrics, such as force capacity, acceleration capacity, etc. Additionally these software packages are often developed for the use with a specific robotics library, such as `robotic-toolbox` (Corke & Haviland, 2021) or `ROS` (Quigley et al., 2009), and they are not trivial to integrate with other libraries.

Even though different efficient tools for evaluating ellipsoids are widely available in the literature and open-source community, the tools for evaluating polytopes are still relatively scarce. The main reason for this is that the polytopes are in general more complex to evaluate and manipulate than ellipsoids. However, the polytopes are much more accurate representation of the true limits. Additionally, polytopes are easy to visualize, as they are essentially triangulated meshes, and they can be easily integrated in the robot control strategies, as they can be expressed as a set of linear constraints.

The evaluation of polytopes is often a computationally expensive task, as their resolution requires using different vertex and facet enumeration algorithms (Fukuda, 2004). Therefore, their computation time is often the limiting factor for the use of polytopes in real world applications, especially when it comes to their online use. Furthermore, even though there are several open-source projects that implement polytope evaluation algorithms, such as `pypoman` (Caron, 2023), Multi-Parametric Toolbox 3 (Herceg et al., 2013) or `cddlib` (Fukuda, 1997; Zurich, 2023), they are often very generic and not easy to use with standard physical ability polytopes. On the other hand, more specific polytope resolution software solutions, such as Constrained Manipulability package (Philip Long, 2023; P. Long & Padir, 2018) or `pygradientpolytope` (Sagar, 2023), are often very specific to their applications, they lack the documentation and flexibility to be extended to new metrics and integrated with other libraries.

Therefore, this paper presents a Python `pycapacity` package in an effort to provide a set of tools specifically tailored for evaluating task-space physical ability metrics for humans and robots, based on polytopes and ellipsoids. This package groups a set of efficient algorithms for their evaluation in an easy to use framework that can be easily integrated with standard robotics and biomechanics libraries. Furthermore, the package implements several state of the art algorithms for polytope evaluation that bring many of the polytope metrics to the few milliseconds evaluation time, making it possible to use them in online and interactive applications.

`pycapacity` has been used in several scientific papers, for real-time control of collaborative carrying using two Franka Emika Panda robots (Skuric et al., 2021), for developing an assist-as-needed control strategy for collaborative carrying task of the human operator and the Franka robot (Skuric et al., 2022). The package has also been used to calculate the approximation of the robot's reachable space using convex polytope (Skuric et al., 2023). On the other hand, the package has been used for the biomechanical calibration of the human musculoskeletal models (Laisné et al., 2023).

Ellipsoids and polytopes as physical ability metrics

In robotics, task-space physical ability metrics establish the relationship between different limits of robot's actuators (joint positions, velocities, torques, etc.), their kinematics and dynamics, and the achievable sets of different task related physical quantities, such as achievable

positions, velocities, forces and similar. Similar metrics can be established for humans as well, by leveraging their musculoskeletal models. Where the humans in addition to the joint limits (joint positions and velocities) have additional limits due to using their muscles as actuators (contraction forces and velocities).

When it comes to characterizing these achievable sets, the two most common approaches are using ellipsoids and polytopes. Ellipsoids are often used to represent the robot's velocity capacity, so called manipulability, while polytopes are mostly used to represent the robot's force capacity. However, both ellipsoids and polytopes can be used to represent any of the task-space physical ability.

To compare the ellipsoid and polytope metrics, the example of the manipulability ellipsoid and manipulability polytope can be used.

The manipulability ellipsoid E, proposed by (Yoshikawa, 1985), is defined as the set of all achievable task-space velocities $\dot{x}$ for a given robot configuration q and joint velocity limits $-1 \leq \dot{q} \leq 1$, and it can be expressed as:

$$E = \{\dot{x} \mid \dot{x} = J(q)\dot{q}, \quad ||\dot{q}||_2 \leq 1\} \tag{1}$$

The equivalent polytope P representation of the manipulability ellipsoid is the manipulability polytope, which is defined as the set of all achievable task-space velocities $\dot{x}$ for a given robot configuration q and joint velocity limits $-1 \leq \dot{q} \leq 1$, and it can be expressed as:

$$P = \{\dot{x} \mid \dot{x} = J(q)\dot{q}, \quad -1 \leq \dot{q} \leq 1\} \tag{2}$$

Figure 1. illustrates the difference between the manipulability ellipsoid and polytope for a planar robot with two joints. The manipulability ellipsoid is an underestimation of the true robot's capacity, as it considers that the robot's velocity limits have the shape of a sphere, while in reality the robot's velocity limits $-1 \leq \dot{q} \leq 1$ define a cube. The manipulability polytope is a more accurate representation of the robot's capacity, as it considers the true shape of the robot's velocity limits.

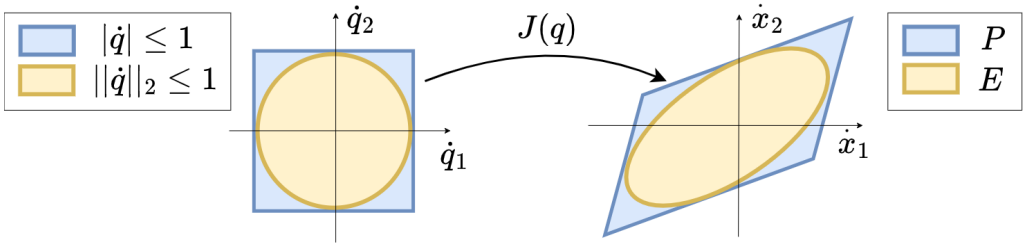

Figure 1: An example manipulability polytope and ellipsoid geometry for a planar $m = 2$ robot with $n = 2$. The difference between the joint space limits for ellipsoid described with $||\dot{q}||_2 \leq 1$ (orange) and the range limits $-1 \leq \dot{q} \leq 1$ (blue) is shown on the right. The difference in obtained achievable task-space velocity $\dot{x}$ polytope P (blue) and ellipsoid E (orange) is shown on the right plot. The plots show that both in joint and task-space the ellipsoid metric is an underestimation of the true robot's capacity.

More generally, polytope based representations of different physical abilities present the exact solution both for robots and for human musculoskeletal models, while ellipsoids present an approximation. Figure 2. shows the difference between the force ellipsoid and polytope (Chiacchio et al., 1996) for one configuration of the Franka Emika Panda robot.

Ellipsoids, however, are much more present in the literature, as their computation is much faster than the computation of polytopes (Finotello et al., 1998).

Evaluating ellipsoids

Evaluating ellipsoids is a computationally efficient task, as it can be done using the singular value decomposition (SVD) (Yoshikawa, 1985). Ellipsoids can be fully defined using their principal axis and principal axis lengths. Once they are known, the ellipsoid can be easily visualized and used for further analysis.

`pycapacity` provides tools for evaluating several common ellipsoid metrics for robots and humans, such as velocity (manipulability), force and acceleration, and it provides a set of tools for their easy visualization.

Evaluating polytopes

Evaluating polytopes consists in finding either the minimal set of their vertices, $\mathcal{V}$-representation, or the minimal set of the half-planes defining their faces, $\mathcal{H}$-representation. The $\mathcal{V}$-representation is often used for visualization purposes, while the $\mathcal{H}$-representation is often integrated in different optimization problems, as it can be represented as a set of linear inequalities.

However, finding the $\mathcal{V}$-representation or the $\mathcal{H}$-representation of a polytope is a computationally expensive task, relying on different vertex and facet enumeration algorithms (Fukuda, 2004). The computational complexity of these algorithms depends on the polytope formulation, the dimensionality of the input (number of robot's joints or human muscles) and output spaces (1D, 2D, 3D or 6D Cartesian space) and the complexity of the polytope geometry (number of vertices and faces).

Therefore, polytope evaluation is often a bottleneck in the computation of different physical ability metrics, especially for human musculoskeletal models, which have a large number of degrees of freedom and a large number of muscles. Furthermore, due to the inherent complexity of the polytope evaluation algorithms, finding the appropriate algorithm for a given polytope formulation and dimensionality of the input and output spaces is not a trivial task.

This package aims to provide a selection of algorithms for polytope evaluation, capable of evaluating common physical ability polytopes in an easy to use and efficient way. These algorithms are implemented in Python and can be used as standalone tools as well. Additionally, the package provides tools for easy visualization the 2D and 3D polytopes.

Implemented physical capacity metrics

The package implements different physical ability metrics for robotic manipulators and humans based on musculoskeletal models.

Robotic manipulators metrics

For robotic manipulators the package integrates several velocity, force and acceleration capacity calculation functions based on ellipsoids and polytopes. A visual comparison of the force polytope and ellipsoid, calculated using this package, is shown on Figure 2.

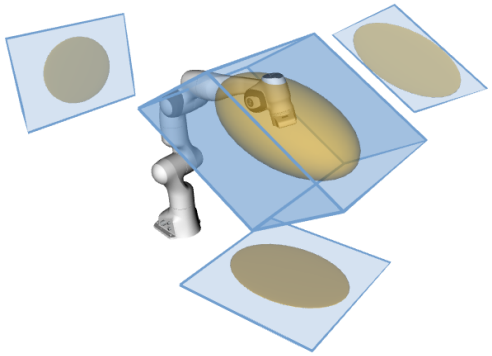

Figure 2: 2D and 3D force polytopes and their ellipsoid counterparts for a 7 degrees of freedom (DOF) *Franka Emika Panda* robot. Both polytopes and ellipsoids are calculated separately for the 3D and for each of the 2D reduced task-space cases. Both polytopes and ellipsoids take in consideration the true joint torque limits provided by the manufacturer. The underestimation of the true force capabilities of the robot by ellipsoids appears clearly.

Ellipsoids

- Velocity (manipulability) ellipsoid E_v

$$E_v = \{\dot{x} \,|\dot{x} = J\dot{q},\ ||W^{-1}\dot{q}|| \leq 1\}, \qquad W = diag(\dot{q}_{max}) \tag{3}$$

- Acceleration (dynamic manipulability) ellipsoid E_a

$$E_a = \{\ddot{x} \,|\ddot{x} = JM^{-1}\tau,\ ||W^{-1}\tau|| \leq 1\}, \qquad W = diag(\tau_{max}) \tag{4}$$

- Force ellipsoid E_f

$$E_f = \{f \,|J^T f = \tau,\ ||W^{-1}\tau|| \leq 1\}, \qquad W = diag(\tau_{max}) \tag{5}$$

In the above definitions, J is the robot Jacobian matrix, M is the inertia matrix, f is the vector of Cartesian forces, $\dot{x}$ and $\ddot{x}$ are vectors fo Cartesian velocities and accelerations, q is the vector of joint positions, $\dot{q}$ is the vector of the joint velocities and τ is the vector of joint torques. Matrix W is a scaling matrix that normalizes the joint space limits.

Polytopes

- Velocity polytope P_v

$$P_v = \{\dot{x} \,|\dot{x} = J\dot{q},\ \dot{q}_{min} \leq \dot{q} \leq \dot{q}_{max}\} \tag{6}$$

- Acceleration polytope P_a

$$P_a = \{\ddot{x} \,|\ddot{x} = JM^{-1}\tau,\ \tau_{min} \leq \tau \leq \tau_{max}\} \tag{7}$$

- Force polytope P_f

$$P_f = \{f \,|J^T f = \tau,\ \tau_{min} \leq \tau \leq \tau_{max}\} \tag{8}$$

- Force polytopes *Minkowski sum* $P_\oplus$ and *intersection* $P_\cap$

$$P_\cap = P_{f1} \cap P_{f1} \qquad P_\oplus = P_{f1} \oplus P_{f1} \tag{9}$$

- Robot's reachable space approximation in the desired horizon of interest Δt_h using the convex polytope formulation P_x, described in the paper by (Skuric et al., 2023)

$$
\begin{aligned}
P_x = \{\Delta x \mid \Delta x = JM^{-1}\tau\frac{\Delta t_h^2}{2}, \\
\tau_{min} \leq \tau \leq \tau_{max}, \\
\dot{q}_{min} \leq M^{-1}\tau\Delta t_h \leq \dot{q}_{max}, \\
q_{min} \leq M^{-1}\tau\frac{\Delta t_h^2}{2} \leq q_{max}\}
\end{aligned}
\tag{10}
$$

In the above definitions, J is the robot Jacobian matrix, M is the inertia matrix, f is the vector of Cartesian forces, $\dot{x}$ and $\ddot{x}$ are vectors fo Cartesian velocities and accelerations, q is the vector of joint positions, $\dot{q}$ is the vector of the joint velocities and τ is the vector of joint torques.

Human musculoskeletal model metrics

For the human musculoskeletal models this package implements the polytope and ellipsoid evaluation functions for the following metrics. A visual representation of the force polytope of a musculoskeletal model, calculated using this package, is shown on Figure 3.

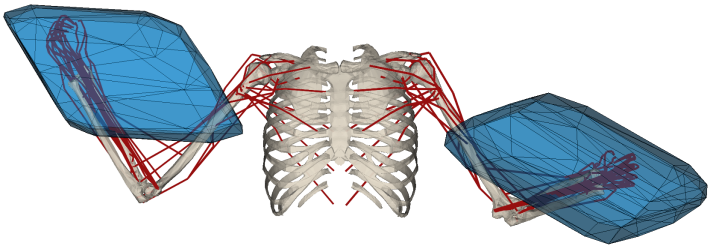

Figure 3: Cartesian force polytope of a musculoskeletal model of both human upper limbs with 7DOf and 50 muscles each, visualized with `biorbd`. The polytopes are scaled with a ratio 1m : 1000N.

Ellipsoids

- Velocity (manipulability) ellipsoid E_v

$$
E_v = \{\dot{x} \mid J\dot{q} = \dot{x},\ L\dot{q} = \dot{l} \quad ||W^{-1}\dot{l}|| \leq 1\}, \qquad W = diag(\dot{l}_{max}) \tag{11}
$$

- Acceleration (dynamic manipulability) ellipsoid E_a

$$
E_a = \{\ddot{x} \mid \ddot{x} = JM^{-1}NF, \quad ||W^{-1}F|| \leq 1\}, \qquad W = diag(F_{max}) \tag{12}
$$

- Force ellipsoid E_f

$$
E_f = \{f \mid NF = J^T f, \quad ||W^{-1}F|| \leq 1\}, \qquad W = diag(F_{max}) \tag{13}
$$

In the above definitions, J is the robot Jacobian matrix, M is the inertia matrix, L si the muscle length Jacobian matrix and $N = -L^T$ is the moment arm matrix. f is the vector of Cartesian forces, $\dot{x}$ and $\ddot{x}$ are vectors fo Cartesian velocities and accelerations, q is the vector of joint positions, $\dot{q}$ is the vector of the joint velocities and τ is the vector of joint torques, $\dot{l}$ is the vector of the muscle stretching velocities and F is the vector of muscular forces. Matrix W is a scaling matrix that normalizes the joint space limits.

Polytopes

- Velocity polytope

$$P_v = \{\dot{x} \mid \dot{l} = L\dot{q},\ \dot{x} = J\dot{q},\ \dot{q}_{min},\ \leq \dot{q} \leq \dot{q}_{max},\ \dot{l}_{min} \leq \dot{l} \leq \dot{l}_{max}\} \tag{14}$$

- Acceleration polytope

$$P_a = \{\ddot{x} \mid \ddot{x} = JM^{-1}NF,\ F_{min} \leq F \leq F_{max}\} \tag{15}$$

- Force polytope

$$P_f = \{f \mid J^T f = NF,\ F_{min} \leq F \leq F_{max}\} \tag{16}$$

In the above definitions, J is the robot Jacobian matrix, M is the inertia matrix, L si the muscle length Jacobian matrix and $N = -L^T$ is the moment arm matrix. f is the vector of Cartesian forces, $\dot{x}$ and $\ddot{x}$ are vectors fo Cartesian velocities and accelerations, q is the vector of joint positions, $\dot{q}$ is the vector of the joint velocities and τ is the vector of joint torques, $\dot{l}$ is the vector of the muscle stretching velocities and F is the vector of muscular forces.

Implemented polytope evaluation algorithms

This package implements several algorithms for polytope evaluation

- Hyper-Plane Shifting Method (HPSM)
- Vertex Enumeration Algorithm (VEPOLI^2)
- Iterative Convex Hull Method (ICHM)

These algorithms are all implemented in Python and used to evaluate different polytope based physical ability metrics. Additionally, the algorithms are available to the users to be used standalone as well.

Hyper-plane shifting method (HPSM)

This is an algorithm based on the paper by (Gouttefarde & Krut, 2010) which presents an efficient way of determining the minimal half-space $\mathcal{H}$ representation of the polytope described by the equation

$$P = \{x \mid x = By, \quad y_{min} \leq y \leq y_{max}\} \tag{17}$$

Vertex enumeration algorithm (VEPOLI^2)

This is an algorithm based on the paper by (Skuric et al., 2021) which describes an efficient method for finding vertex $\mathcal{V}$ representation of the polytope described by the equation

$$P = \{x \mid Ax = y, \quad y_{min} \leq y \leq y_{max}\} \tag{18}$$

Iterative convex-hull method (ICHM)

This is an algorithm described in the paper by (Skuric et al., 2022) which implements an efficient method which iteratively approximates the polytope

$$P = \{x \mid Ax = By, \quad y_{min} \leq y \leq y_{max}\} \tag{19}$$

The method finds both vertex $\mathcal{V}$ and half-plane $\mathcal{H}$ representation of the polytope at the same time.

It can be additionally extended to the case where there is an additional projection matrix P making a class of problems:

$$P = \{x \mid x = Pz, Az = By, \quad y_{min} \leq y \leq y_{max}\} \tag{20}$$

Performance evaluation of polytope metrics

The applicable methods to evaluate different polytope based metrics depend on the family of problems they correspond to. Therefore this section brings the information about which algorithm is used for which polytope metric and provides a brief performance evaluation of their execution times.

Additionally, to give brief information about the efficiency of the proposed methods, the section provides the execution times of the methods for the example problems. However, as these execution times can vary significantly depending on the complexity of the model used and the hardware it is run on, the users are encouraged to run the benchmark scripts themselves to get the most accurate results. This package provides several benchmarking scripts in the `examples` folder.

Robotic manipulators

In case of robotic manipulators the methods used are given in the following table.

Polytope Metric	Algorithm	Problem type	Execution time [ms] mean $\pm$ std. (max)
Velocity	HPSM	$x = By,\ y \in [y_{min}, y_{max}]$	3.6 $\pm$ 0.21 (5.7)
Acceleration	HPSM	$x = By,\ y \in [y_{min}, y_{max}]$	6.6 $\pm$ 1.4 (14.2)
Force	VEPOLI^2	$Ax = b,\ b \in [b_{min}, b_{max}]$	6.8 $\pm$ 0.88 (16.4)
Force intersection	VEPOLI^2	$Ax = b,\ b \in [b_{min}, b_{max}]$	98.2 $\pm$ 29.33 (165.8)
Force sum	VEPOLI^2	$Ax = b,\ b \in [b_{min}, b_{max}]$	17.1 $\pm$ 3.4 (44.9)
Reachable space	ICHM	$x = By,\ y \in P_y$	30.5 $\pm$ 6.6 (76.7)

The average execution time is calculated for 1000 random configuration of a 7 DOF Franka Emika panda robot, the model was used with `pinocchio` software. All the experiments are run on a computer equipped with a 1.90GHz Intel i7-8650U processor. The results are obtained using the benchmarking script provided in the by the repository in the `examples` folder.

Musculoskeletal models

In case of human musculoskeletal models the methods used are given in the table below.

Polytope Metric	Algorithm	Problem type	Execution time [ms] mean $\pm$ std. (max)
Force	ICHM	$Ax = By,\ y \in [y_{min}, y_{max}]$	186.8 $\pm$ 45.6 (281.6)
Acceleration	HPSM or ICHM	$x = By,\ y \in [y_{min}, y_{max}]$	378.8 $\pm$ 62.3 (643.7)
Velocity	ICHM	$x = By,\ y \in P_y$	223.1 $\pm$ 60.4 (389.1)

The average execution time is calculated for 1000 random configuration of a 50 muscle 7 DOF musculoskeletal model introduced by (Holzbaur et al., 2005), the model was used with `biorbd` biomechanics software. The experiments are run on a computer equipped with a 1.90GHz Intel

i7-8650U processor. The results are obtained using the benchmarking script provided in the by the repository in the `examples` folder.

Conclusion

This paper introduces the `pycapacity` Python package, a toolkit designed to evaluate task-space physical ability metrics for both humans and robots based on polytopes and ellipsoids. The aim of this package is to provide efficient tools for evaluating these metrics within an easily accessible framework, which can seamlessly integrate with standard robotics and biomechanics libraries. By implementing state-of-the-art algorithms for polytope evaluation, `pycapacity` enables the evaluation of these metrics in an efficient manner, making them applicable for interactive online applications.

Acknowledgements

This work has been funded by the BPI France Lichie project.

References

Caron, S. (2023). Pypoman: Polyhedron manipulation in python. In *GitHub repository*. https://github.com/stephane-caron/pypoman; GitHub.

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

Chiacchio, P., Bouffard-Vercelli, Y., & Pierrot, F. (1996). Evaluation of force capabilities for redundant manipulators. *Proceedings of IEEE International Conference on Robotics and Automation*, *4*, 3520–3525 vol.4. https://doi.org/10.1109/ROBOT.1996.509249

Corke, P., & Haviland, J. (2021). Not your grandmother's toolbox – the robotics toolbox reinvented for python. *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 11357–11363. https://doi.org/10.1109/ICRA48506.2021.9561366

Delp, S. L., Anderson, F. C., Arnold, A. S., Loan, P., Habib, A., John, C. T., Guendelman, E., & Thelen, D. G. (2007). OpenSim: Open-source software to create and analyze dynamic simulations of movement. *IEEE Transactions on Biomedical Engineering*, *54*(11), 1940–1950. https://doi.org/10.1109/TBME.2007.901024

Finotello, R., Grasso, T., Rossi, G., & Terribile, A. (1998). Computation of kinetostatic performances of robot manipulators with polytopes. *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No.98CH36146)*, *4*, 3241–3246 vol.4. https://doi.org/10.1109/ROBOT.1998.680928

Fukuda, K. (1997). Cdd/cdd+ reference manual. *Institute for Operations Research, ETH-Zentrum*, 91–111.

Fukuda, K. (2004). Frequently asked questions in polyhedral computation. *ETH, Zurich, Switzerland*, *85*.

Gouttefarde, M., & Krut, S. (2010). Characterization of parallel manipulator available wrench set facets. In J. Lenarcic & M. M. Stanisic (Eds.), *Advances in robot kinematics: Motion in man and machine* (pp. 475–482). Springer Netherlands. https://doi.org/10.1007/978-90-481-9262-5_51

Haviland, J., & Corke, P. (2020). Maximising manipulability during resolved-rate motion control. *CoRR*, *abs/2002.11901*. https://arxiv.org/abs/2002.11901

Herceg, M., Kvasnica, M., Jones, C. N., & Morari, M. (2013). Multi-parametric toolbox 3.0. *2013 European Control Conference (ECC)*, 502–510. https://doi.org/10.23919/ECC.2013.6669862

Holzbaur, K. R., Murray, W. M., & Delp, S. L. (2005). A model of the upper extremity for simulating musculoskeletal surgery and analyzing neuromuscular control. *Annals of Biomedical Engineering*, *33*(6), 829–840. https://doi.org/10.1007/s10439-005-3320-7

Jaquier, Noémie. (2023). Manipulability: Repository containing the codes for manipulability learning, tracking and transfer. In *GitHub repository*. https://github.com/NoemieJaquier/Manipulability; GitHub.

Jaquier, N., Rozo, L., Caldwell, D. G., & Calinon, S. (2021). Geometry-aware manipulability learning, tracking and transfer. *Intl. Journal of Robotics Research"*, *20*(2-3), 624–650.

Laisné, G., Salotti, J.-M., & Rezzoug, N. (2023, October). Genetic algorithms for force polytopes prediction. *48ème Congrès de la Société de Biomécanique*. https://inria.hal.science/hal-04151258

Long, Philip. (2023). Constrained manipulability: A library used to compute and visualize a robot's capacities in constrained environments. In *GitHub repository*. https://github.com/philip-long/constrained_manipulability; GitHub.

Long, P., & Padir, T. (2018). Evaluating robot manipulability in constrained environments by velocity polytope reduction. *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*, 1–9. https://doi.org/10.1109/HUMANOIDS.2018.8624962

Michaud, B., & Begon, M. (2021). Biorbd: A C++, Python and MATLAB library to analyze and simulate the human body biomechanics. *Journal of Open Source Software*, *6*(57), 2562. https://doi.org/10.21105/joss.02562

Prada, M. (2023). Manipulability metrics: ROS-based library for computing manipulability metrics. In *GitHub repository*. https://github.com/tecnalia-medical-robotics/manipulability_metrics; GitHub.

Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A. Y., & others. (2009). ROS: An open-source robot operating system. *ICRA Workshop on Open Source Software*, *3*, 5.

Sagar, K. (2023). Pygradientpolytope: A set of ROS nodes for generating and visualizing cartesian velocity, force and desired polytopes. In *GitLab repository*. https://gitlab.com/KeerthiSagarSN/rospygradientpolytope; GitLab.

Skuric, A., Padois, V., & Daney, D. (2023). Approximating robot reachable space using convex polytopes. In P. Borja, C. Della Santina, L. Peternel, & E. Torta (Eds.), *Human-friendly robotics 2022* (pp. 45–60). Springer International Publishing. https://doi.org/10.1007/978-3-031-22731-8_4

Skuric, A., Padois, V., & Daney, D. (2021). On-line force capability evaluation based on efficient polytope vertex search. *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 1700–1706. https://doi.org/10.1109/ICRA48506.2021.9562050

Skuric, A., Padois, V., Rezzoug, N., & Daney, D. (2022). On-line feasible wrench polytope evaluation based on human musculoskeletal models: An iterative convex hull method. *IEEE Robotics and Automation Letters*, *7*(2), 5206–5213. https://doi.org/10.1109/LRA.2022.3155374

Yoshikawa, T. (1985). Manipulability of robotic mechanisms. *The International Journal of Robotics Research*, *4*(2), 3–9. https://doi.org/10.1177/027836498500400201

Zurich, E. (2023). Cddlib: An efficient implementation of the double description method. In *GitHub repository*. https://github.com/cddlib; GitHub.