Skidy – a new tool for closed form solution of the equations of motion for open chain robots

Hannah Isermann[1]
[1]Robotics Innovation Center, DFKI GmbH, Bremen

Motivation

An efficient calculation of the kinematics and dynamics is essential in the control of robots. Skidy, as symbolic code generation tool, can calculate these equations in closed form to provide code for the control of open chain robots. The individual parameters stay thereby symbolic, so that they can be adapted in the finished code. This enables efficient implementations of adaptive control algorithms and parameter optimizations. By supporting a wide range of programming languages for code generation, Skidy also facilitates efficient work on a wide range of systems without the need to learn a new tool.

Main Features

- Calculate the forward position and velocity kinematics of open chain robots.
- Calculate the equation of motion and its first and second order timer derivatives.
- Code generation:

Interfaces

Skidy needs the joint screw coordinates, body transforms and inertia parameter as program input. They can be provided via:

- URDF (also convertible to YAML)
- YAML (see Example)
- directly in python

To increase the usability, templates for the YAML and python files can be generated automatically, where only the parameters have to be adapted. E.g.:

```
$ skidy --template -S RR robot.yaml
```

for a robot with 2 revolute joints.

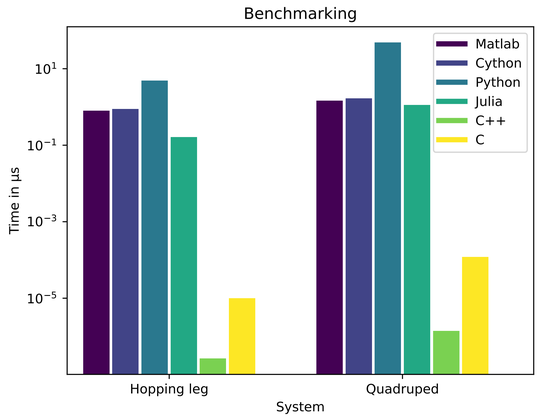

Figure: Execution time of generated inverse dynamics code for 3 DOF hopping leg and 12 DOF quadruped.

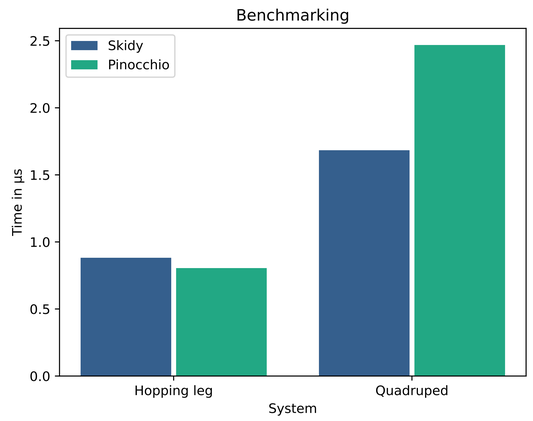

Figure: Execution time of inverse dynamics Skidy (Cython) vs Pinocchio [2] for 3 DOF hopping leg and 12 DOF quadruped.

Implementation

Skidy was implemented in Python using the symbolic mathematics library Sympy [1]. It can be used as a command line tool or as a library. Skidy is divided into three main packages:

- *kinematics_generator:* equation and code generation
- *matrices:* matrix operations and helper functions
- *parser:* interfaces

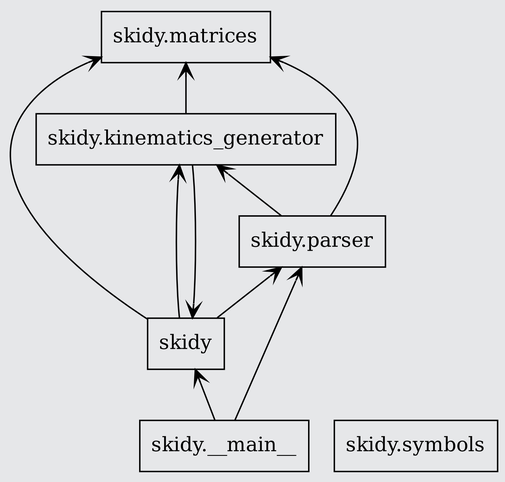

Figure: Packages in Skidy.

Equations of Motion

The implementation followed [3] using Lie group formulation for rigid body systems to calculate the kinematics and dynamics in compact equations. The equations of motion can thereby calculated as followed:
Mass and Coriolis matrix:

$$\boldsymbol{M}(\boldsymbol{q}) = J^T M J, \quad \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = J^T C J$$

where

$$M := diag(\boldsymbol{M_1}, ..., \boldsymbol{M_n})$$
$$C(\boldsymbol{q}, \dot{\boldsymbol{q}}, V(\dot{\boldsymbol{q}})) := -MAa - b^T M$$

and

$$b(V) := diag(\boldsymbol{ad_{V_1}}, ..., \boldsymbol{ad_{V_n}})$$

Gravity vector:

$$\boldsymbol{Q}_{grav}(\boldsymbol{q}) = J^T M U \boldsymbol{G}$$

with

$$\boldsymbol{G} := -\begin{pmatrix} \boldsymbol{0} \\ {}^0\boldsymbol{g} \end{pmatrix}, \quad U(\boldsymbol{q}) := \begin{pmatrix} \boldsymbol{Ad}_{C_1}^{-1} \\ \vdots \\ \boldsymbol{Ad}_{C_n}^{-1} \end{pmatrix}$$

External forces:

$$\boldsymbol{Q}_{ext}(\boldsymbol{q}, t) = J^T(\boldsymbol{q}) W_{EE}(t)$$

where $W_{EE}(t)$ is the external Wrench at the end-effector.
Inverse dynamics:

$$\tau = \boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{Q}_{grav}(\boldsymbol{q}) + \boldsymbol{Q}_{ext}(\boldsymbol{q}, t)$$

q: joint position, J: system Jacobian, M_i: 6x6 inertia matrix in body frame, V: twist, A, a: Block diagonal of the Adjoint and spacial cross product of the body frame, ${}^0g$ gravity vector, τ: joint torques.

Usage/Example

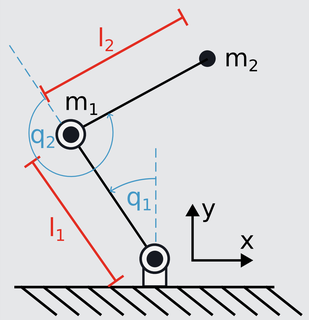

Figure: Robotic arm to illustrate how a robot can be represented in the YAML file to the right using symbolic values. The symbolic variables are used as arguments in the generated code.

```yaml
---
gravity: [0,-g,0]
representation: body_fixed
joint_screw_coord:
  - type: revolute
    axis: [0,0,1]
  - type: revolute
    axis: [0,0,1]
body_ref_config:
  - translation: [0,0,0]
  - translation: [0,l1,0]
ee:
    translation: [0,l2,0]
mass_inertia:
  - mass: m1
    inertia: m1*l2**2
    com: [0,l1,0]
  - mass: m2
    inertia: m2*l2**2
    com: [0,l2,0]
q: [q1,q2]
qd: [dq1,dq2]
q2d: [ddq1,ddq2]
```

Generate the kinematics and equation of motion for this robot and save them as LaTeX file using:

```
$ skidy -s --latex robot.yaml
```

Acknowledgment

This work was supported by the federal state of Bremen for setting up the Underactuated Robotics Lab under Grant 201-342-04-2/2021-4-1.

References

[1] Aaron Meurer et al. "SymPy: symbolic computing in Python". In: *PeerJ Computer Science* 3 (Jan. 2017), e103. ISSN: 2376-5992. DOI: 10.7717/peerj-cs.103. URL: https://doi.org/10.7717/peerj-cs.103.

[2] Justin Carpentier, Florian Valenza, Nicolas Mansard, et al. *Pinocchio: fast forward and inverse dynamics for poly-articulated systems.* https://stack-of-tasks.github.io/pinocchio. 2015–2021.

[3] Andreas Mueller and Shivesh Kumar. "Closed-form time derivatives of the equations of motion of rigid body systems". In: *Multibody System Dynamics* 53.3 (2021), pp. 257–273.