Classical Mechanics: Extending the Construction of Systems From Bodies and Joints

About me

Personal information

Name: Timo Stienstra
University: Delft University of Technology
E-mail: timostienstra00@gmail.com & T.J.Stienstra@student.tudelft.nl
GitHub: https://github.com/TJStienstra & https://github.com/Tstienstra
Timezone: CEST (UTC+2)

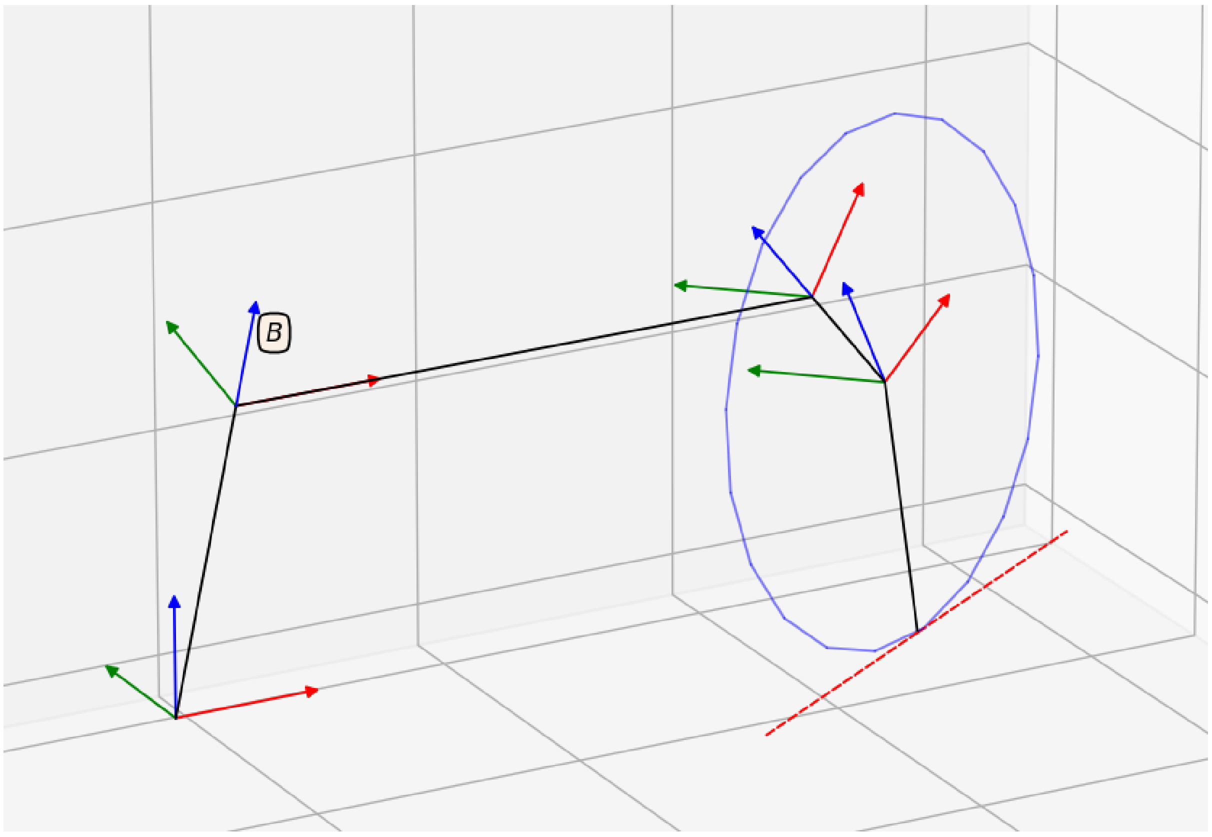
Personal Background

After completing my BSc. Mechanical Engineering with a minor Computer Science, I am now in my first year MSc. Mechanical Engineering student at Delft University of Technology. Here I follow the track BioMechanical Design, where I also specialize myself in robotics. My favorite subject is mechanics, dynamics to be more specific, and on close second comes programming. This results in that I like designing printer parts for others, modifying my 3D printer, creating puzzles and repairing other broken things in my spare time. But besides these more technical hobbies, I do also love studying the Bible and spend time with family and friends.

Programming Background

While my major was mainly focused on mechanics, I also spend quite a lot of time programming. One of my projects during the bacher was for example optimizing the melting temperature and thickness of phase change materials. This involved running various simulations in Python, where we also had to solve higher order equations. Here I encountered SymPy for the first time and immediately loved it. Just put in your equations and get the solution. At this time I was using Spyder in Anaconda. However after hearing about PyCharm during my minor, I made the switch to PyCharm. This is the IDE I'm still using today, since it just has a lot of useful features which are organized in a good layout. To give an example the code coverage of your testing is displayed in your file structure with percentages and in your code file with a colored bar at the side.
As already quite visible from what I just mentioned is Python my goto language. Since it was easy to learn and it has loads of libraries. Another big plus is the fact that you do not need to compile it and you can overwrite the same variable with different types, however at the same time I do also like C++ for the fact that typing, use of pointers keeps it really clear what you are currently working with. So do have their pros and cons, but when you want to quickly compute something or automate a simple task, then Python is just the perfect language.

A project I'm currently working on in my spare time is a plotter for mechanical objects from SymPy [1], the image below shows an example plot. And one of the things I like in the current setup used is the fact how arguments are parsed to other functions, which are in the end if not solved parsed to Matplotlibs function. This way quite a lot of features are easily integrated without a lot of programming leading to something that seems quite advanced, but is in fact rather easy.



Contributions

My current contributions to the SymPy library are:

- #12322 opened a draft PR, but this is still work in progress.
- #22956 proposed PR 23392
- #23358 (closed) fixed with PR 23362 (merged)
- #23382 raised issue
- #23393 raised issue

The Project

The aim of this project is to further develop the construction of systems from bodies and joints [2]. This involves fixing existing bugs, which will also be discovered by creating more examples (and their accompanying documentation). The next step will be implementing a quaternion definition for the child_axis to solve an ambiguity. After fixing the current implementation and documentation the focus will be on adding joints, starting with a planar and cylindrical joint. Next up will be the spherical joint, which can either be defined with euler angles or a quaternion.

There are multiple things that really excite me about this project. First of that it is both dynamics and programming in Python, two things I find amazing and yes the combination sounds even better. Designing and creating a program that helps you solve your dynamics problems more easily is a great and fun challenge. Besides this I also get to contribute to an open-source library I love to use and with that help also a lot of others.

As mentioned in my background there are several courses which qualify me for this project:

- BSc Mechanical Engineering courses:
  - Introductory Dynamics
  - Rigid-body dynamics
  - Several mathematics courses
- Minor Computer Science courses:
  - Introduction to Python Programming
  - Algorithms and Data Structures
  - Software Engineering Methods
  - Data Analytics: analyzing data in Python
  - Visual Data Processing: processing images in Python with NumPy
  - Minor CS project
- MSc Mechanical Engineering courses:
  - Robot Software Practicals
  - Multi-body Dynamics B (currently following)

Status

As mentioned in the GSoC idea:

- Sahil Shekewat worked on implementing a joint based descriptor for systems: https://github.com/sympy/sympy/pulls/sahilshekhawat
- Sudeep Sidhu completed Sahil's work and merged a functioning joint based system than can solve open chain problems. See his report: https://github.com/sympy/sympy/wiki/GSoC-2021-Report-Sudeep-Sidhu-:-Implement-JointsMethod

Phases

To make a planning of this project it will be divided in 3 phases with a clear deliverable at the end of each of them.

Phase 1

The first phase will be focused on getting more familiar with the current program. This means that I will start with fixing the existing bugs. At the same time I will probably also start creating more examples, which further check the program for bugs. Parts of these examples will also be used to modify, expand and add tests. So the end goal of this phase will be to make the joints well documented, fix existing bugs and make it ready for expansion.
Currently open issues to fix:

- The bugs I am currently working on
- #21705 Documentation explaining the joints

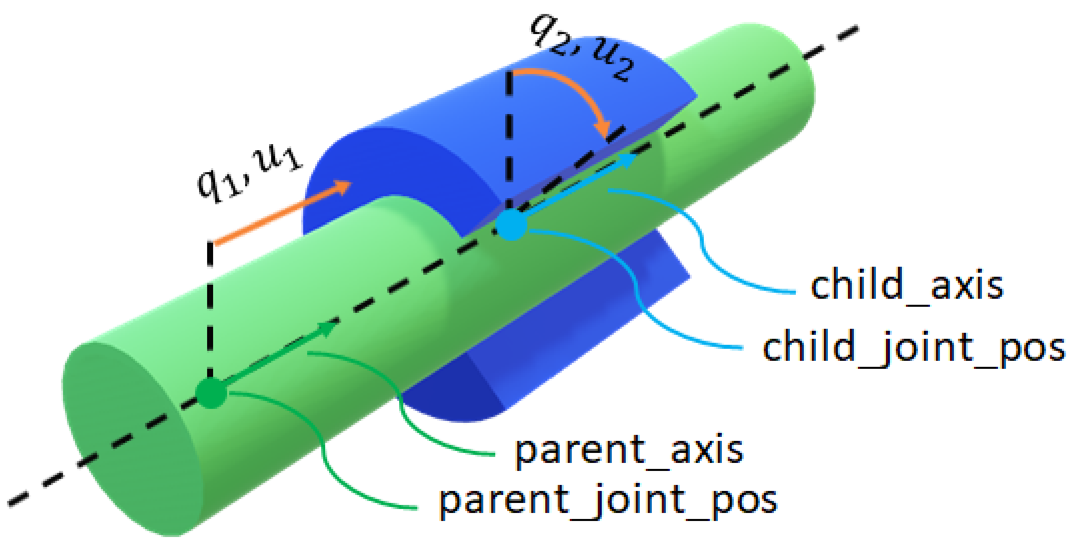
Phase 2

One frustration I encountered while preparing this proposal is that the joint direction is currently defined with a parent_axis and a child_axis. Which are aligned, however in this process the angle around this axis is currently automatically set. To solve this ambiguity I would like to implement an option to pass a quaternion, which fixes this alignment.
This is something I just ran into a day before the deadline, so I will have to think it out a bit more.

Phase 3

In this second phase the CylindricalJoint and PlanarJoint as mentioned in #21519 will be implemented. These two joints are also explained in the proposal of Sudeep Sidhu. But a nice overview of joint is also found here [3]

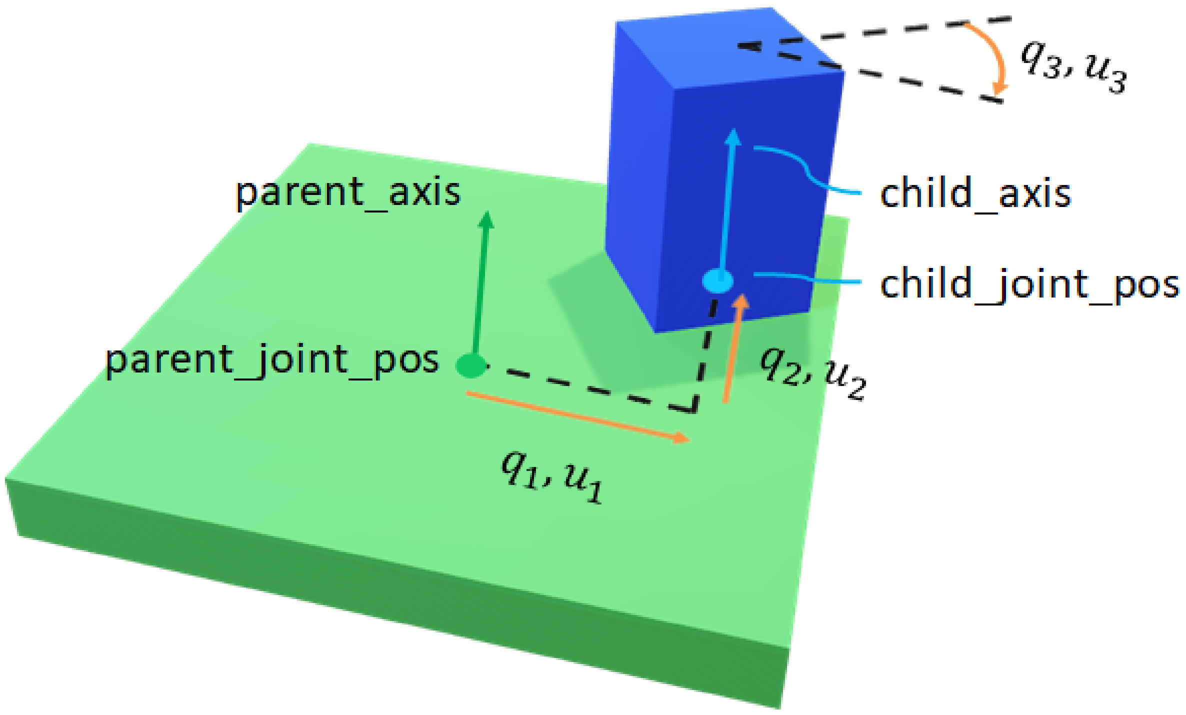
Cylindrical Joint



The cylindrical joint is a widely used joint with 2 DoF as shown in the image above. Here is some example code:

```
>>> q1, q2, u1, u2 = dynamicsymbols('q1, q2, u1, u2')
>>> parent = Body('parent')
>>> child = Body('child')
>>> joint = CylindricalJoint('joint', parent, child,
                             coordinates=[q1, q2], speeds=[u1, u2],
                             parent_axis=parent.x, child_axis=child.x)
```

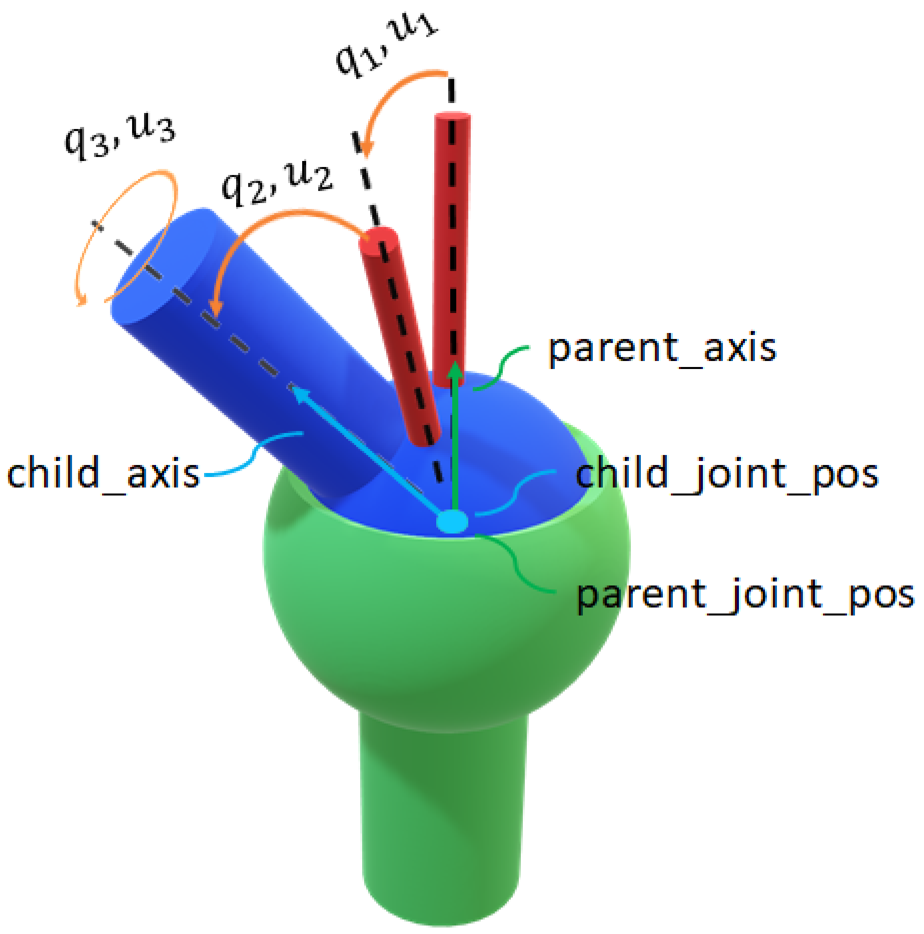
Planar Joint



Shown above is a planar joint with 3 DoF, two translations and one rotation.

```
>>> q1, q2, q3, u1, u2, u3 = dynamicsymbols('q1, q2, q3, u1, u2, u3')
>>> parent = Body('parent')
>>> child = Body('child')
>>> joint = PlanarJoint('joint', parent, child,
                        coordinates=[q1, q2, q3], speeds=[u1, u2, u3],
                        parent_axis=parent.z, child_axis=child.z)
```

Phase 4



In this phase the spherical joint will be added.

```
>>> q1, q2, q3, u1, u2, u3 = dynamicsymbols('q1, q2, q3, u1, u2, u3')
>>> parent = Body('parent')
>>> child = Body('child')
>>> joint = SphericalJoint('joint', parent, child,
                           coordinates=[q1, q2, q3], speeds=[u1, u2, u3],
                           parent_axis=parent.z, child_axis=child.z)
```

Timeline

My aim is to spend around 350 hours on this project with ~30-40h/week.

Week 1-3 (27 June - 13 July) ~ 80 hours

During these first two and a half weeks I expect to complete phase 1 of the project.

Holiday (15 July - 6 August) ~ 0 hours

During these weeks I will be on vacation.

Week 4-6 (8 - 26 August) ~ 100 hours

During these weeks the focus will be on implementing phase 2.

Week 7-9 (29 August - 16 September) ~ 90 hours

During these weeks the Cylindrical- and PlanarJoint will be implemented (phase 3).

Week 10-11 (19 - 30 September) ~ 60 hours

Implementation SphericalJoint (phase 4).

Week 12 (3 - 7 October) ~40 hours

Finishing the project.

References

[1] https://github.com/TJStienstra/SympyMechanicsPlotter (note that this is still work in progress)
[2]
https://github.com/sympy/sympy/wiki/GSoC-Ideas#classical-mechanics-constructing-systems-from-bodies-and-joints
[3] https://community.3dcs.com/help_manual/mechanicaljoints.htm
[4] https://en.wikipedia.org/wiki/Cylindrical_joint