The Journal of Open Source Software

C4DYNAMICS: The Python framework for state-space modeling and algorithm development

Ziv Meri [1]

1 Independent Researcher, Israel

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Sophie Beck

Reviewers:

- @hweifluids
- @borgesaugusto

Submitted: 23 December 2024
Published: unpublished

Summary

Dynamic systems play a critical role across various fields such as robotics, aerospace, and guidance, navigation, and control (GNC). The state-space representation is the most widely used modeling approach for dynamic systems in the time domain. While Python offers robust mathematical tools, it lacks a dedicated framework tailored for state-space modeling. C4DYNAMICS bridges this gap by introducing a Python-based platform designed for state-space modeling and analysis (Kalman, 1959; Luenberger, 1979). The framework's modular architecture, with "state objects" at its core, simplifies the development of algorithms for sensors, filters, and detectors. This allows researchers, engineers, and students to effectively design, simulate, and analyze dynamic systems. By integrating state objects with a scientific library, **C4DYNAMICS** offers a scalable and efficient solution for dynamic systems modeling.

Statement of Need

Modeling and simulation of dynamical systems are essential across robotics, aerospace, and control engineering. While Python provides powerful numerical libraries (e.g., NumPy, SciPy) and several domain-specific frameworks, none directly support low-level, state-space–based algorithm development.

C4DYNAMICS is designed for engineers who prefer code-based modeling and want a framework to explicitly define the variables encapsulated in the system's state vector, and perform streamlined mathematical operations such as scalar multiplication, dot products, and vector addition/subtraction, and data operations such as storing states, retrieving histories, and plotting variables.

Comparison with Existing Software

Existing tools generally fall into two categories: 1) Block-diagram frameworks (e.g., SimuPy (Margolis, 2017), BdSim (Nevay et al., 2020)) mimic Simulink and simplify model building through graphical interfaces, but they abstract away the state vector and limit direct mathematical manipulation. 2) High-level simulators (e.g., IR-Sim, RobotDART (Chatzilygeroudis et al., 2024)) allow algorithm testing in predefined environments, but lack flexibility for core system modeling and algorithm design.

This leaves a gap for engineers and researchers who wish to work explicitly at the state-space level—defining variables, performing mathematical operations on them, and integrating with modern data-driven pipelines.

C4DYNAMICS fills this gap with a modular, Python-native framework for state-space modeling of dynamical systems. Built around explicit state representations and complemented by a

scientific library of filters and sensor models, it enables reproducible modeling, testing, and optimization of dynamic systems within the scientific Python ecosystem.

Example

A simple pendulum with a point mass on a rigid massless rod of length = `1[m]`, swiniging under gravity of `9.8[m/s²]`. with initial angle `θ = 50[deg]` and angle rate `q = 0[deg/s]`, integrated with `solve_ivp` with a time step of `0.01[s]` for `5[s]`.

Import required packages:

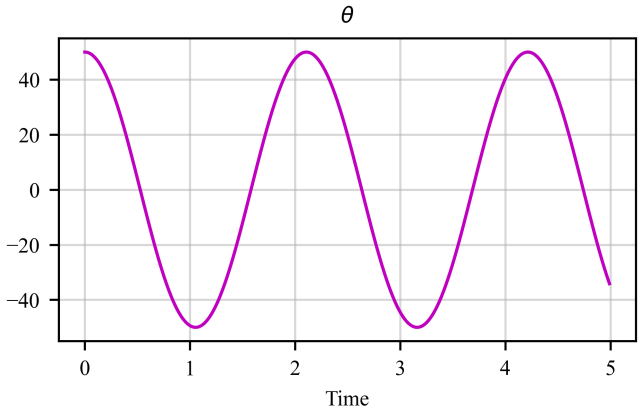
```
from scipy.integrate import solve_ivp
from matplotlib import pyplot as plt
import c4dynamics as c4d
import numpy as np
```

Define a state object (`pend`) and run the main-loop:

```
dt = 0.01
pend  = c4d.state(theta = 50 * c4d.d2r, q = 0)

for ti in np.arange(0, 5, dt):
  pend.store(ti)
  pend.X = solve_ivp(lambda t, y: [y[1], -9.8 * c4d.sin(y[0])], [ti, ti + dt], pend.X).y
1]

pend.plot('theta', scale = c4d.r2d, darkmode = False)
plt.show()
```



References

Chatzilygeroudis, K., Totsila, D., & Mouret, J.-B. (2024). RobotDART: A versatile robot simulator for robotics and machine learning researchers. *Journal of Open Source Software*, *9*(102), 6771. https://doi.org/10.21105/joss.06771

Kalman, R. (1959). On the general theory of control systems. *IRE Transactions on Automatic Control*, *4*(3), 110–110. https://doi.org/10.1109/TAC.1959.1104873

Luenberger, D. G. (1979). *Introduction to dynamic systems, theory, models, and applications*. 446–446. https://doi.org/10.1109/TAC.1959.1104873

Margolis, B. W. (2017). SimuPy: A Python framework for modeling and simulating dynamical systems. *J. Open Source Software*, *2*(17), 396. https://doi.org/10.21105/joss.00396

Nevay, L. J., Boogert, S. T., Snuverink, J., Abramov, A., Deacon, L. C., Garcia-Morales, H., Lefebvre, H., Gibson, S. M., Kwee-Hinzmann, R., Shields, W., & Walker, S. D. (2020). BDSIM: An accelerator tracking code with particle–matter interactions. *Computer Physics Communications*, *252*, 107200. https://doi.org/https://doi.org/10.1016/j.cpc.2020.107200