The Journal of Open Source Software

gym-electric-motor (GEM): A Python toolbox for the simulation of electric drive systems

Praneeth Balakrishna[1], **Gerrit Book**[1], **Wilhelm Kirchgässner**[1], **Maximilian Schenke**[1], **Arne Traue**[1], **and Oliver Wallscheid**[1]

1 Department of Power Electronics and Electrical Drives, Paderborn University, Germany

DOI: 10.21105/joss.02498

Software

- Review
- Repository
- Archive

Editor: Kevin M. Moerman

Reviewers:

- @moorepants
- @dineshresearch

Submitted: 29 May 2020
Published: 06 February 2021

Summary

The `gym-electric-motor` (`GEM`) library provides simulation environments for electrical drive systems and, therefore, allows to easily design and analyze drive control solutions in Python. Since `GEM` is strongly inspired by OpenAI's `gym` (Brockman et al., 2016), it is particularly well-equipped for (but not limited to) applications in the field of reinforcement-learning-based control algorithms. In addition, the interface allows to plug in any expert-driven control approach, such as model predictive control, to be tested and to perform benchmark comparisons. The `GEM` package includes a wide variety of motors, power electronic converters and mechanical load models that can be flexibly selected and parameterized via the API. A modular structure allows additional system components to be included in the simulation framework.

Statement of Need

Electric drive systems and their control are an important topic in both academic and industrial research due to their worldwide usage and deployment. Control algorithms for these systems have usually been designed, parameterized and tested within `MATLAB - Simulink` (The MathWorks, Inc., 2020), which is developed and promoted specifically for such engineering tasks. In the more recent past, however, commercial software like `MATLAB` has difficulties to stay on par with state-of-the-art concepts for scientific modeling and the flexibility offered by open-source libraries that are available for more accessible programming languages like Python. Consequently, a Python-based drive simulation framework like `GEM` is an evident step in order to accelerate corresponding control research and development. Specifically, the latest efforts concerning industrial application of reinforcement-learning control algorithms heavily depend on Python packages like `Keras` (Chollet & others, 2015), `Tensorflow` (Abadi et al., 2015) or `PyTorch`(Paszke et al., 2019). Hence, the built-in OpenAI `gym` interface allows to easily couple `GEM` to other open-source reinforcement learning toolboxes such as `Stable Baselines3` (Raffin et al., 2019), `TF-Agents` (Guadarrama et al., 2018) or `keras-rl` (Plappert, 2016).

Providing easy access to the non-commercial, open-source `GEM` library allows users from any engineering domain to include accurate drive models into their simulations, also beyond the topic of control applications. Considering the prevalence of commercial software like `MATLAB` for educational purposes, a free-of-charge simulation alternative that does not force students or institutions to pay for licenses, has great potential to support and encourage training of new talents in the field of electrical drives and neighbouring domains (e.g. power electronics or energy systems). `GEM` has already been used in graduate courses on reinforcement learning (Kirchgässner et al., 2020).

Related software

Due to the strong dependence of downstream industrial development on simulated environments there is a comprehensive variety of commercial software that enables numerical analysis of every facet of electric drives. To name just a few, `MATLAB - Simulink` is probably the most popular software environment for numerical analysis in engineering. Herein, `MATLAB` is providing for a scientific calculation framework and `Simulink` for a model-driven graphical interface with a very large field of applications. Examples that are designed for real-time capability (e.g., for hardware-in-the-loop prototyping) can be found in `VEOS` (dSPACE GmbH, 2019) or `HYPERSIM` (OPAL-RT TECHNOLOGIES, Inc., 2020). Non-commercial simulation libraries exist, but they rarely come with predefined system models. An exemplary package from this category is `SimuPy` (Margolis, 2017), which provides lots of flexibility for the synthesis of generic simulation models, but also requires the user to possess the necessary expert knowledge in order to implement a desired system model. Likewise, general purpose component-oriented simulation frameworks like `OpenModelica` (Open Source Modelica Consortium, 2020) or `XCos` (Scilab, 2020) can be used for setting up electrical drive models, too, but this requires expert domain knowledge and out-of-the-box Python interfaces (e.g., for reinforcement learning) are not available.

In the domain of motor construction it is furthermore interesting to observe the behavior of magnetic and electric fields within a motor (simulation of partial differential equations). Corresponding commercial simulation environments, like `ANSYS Maxwell` (ANSYS, Inc., 2020), `Motor-CAD` (Motor Design Ltd., 2020) or `MotorWizard` (ElectroMagneticWorks, Inc., 2020) and the exemplary non-commercial alternative `FEMM` (Meeker, 2020) are very resource and time consuming because they depend on the finite element method, which is a spatial discretization and numerical integration procedure. Hence, these software packages are usually not considered in control development, and complement `GEM` at most. This particularly applies in the early control design phase when researching new, innovative control approaches (rapid control prototyping) or when students want to receive quasi-instantaneous simulation feedbacks.

Package Architecture

The `GEM` library models an electric drive system by its four main components: voltage supply, power converter, electric motor and mechanical load. The general structure of such a system is depicted in Figure 1.

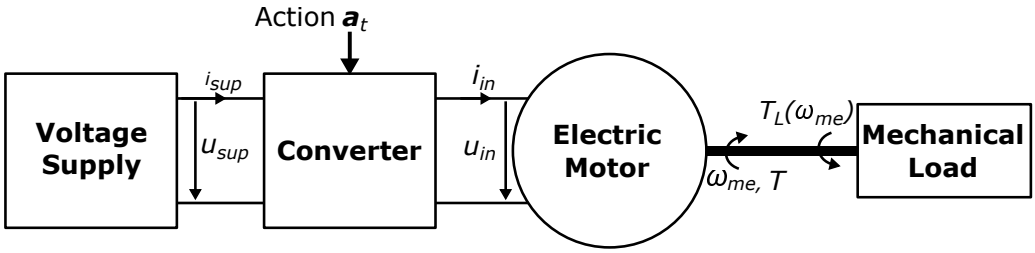

Figure 1: Simplified structure diagram of an electric drive system

The **voltage supply** provides the necessary power that is used by the motor. It is modeled by a fixed supply voltage u_{sup}, which allows to monitor the supply current into the converter. A **power electronic converter** is needed to supply the motor with electric power of proper frequency and magnitude, which commonly includes the conversion of the supply's direct current to alternating current. Typical drive converters exhibit switching behavior: there is a

finite set of different voltages that can be applied to the motor, depending on which switches are open and which are closed. Besides this physically accurate view, a popular modeling approach for switched mode converters is based on dynamic averaging of the applied voltage u_{in}, rendering the voltage a continuous variable. Both of these modeling approaches are implemented and can be chosen freely, allowing usage of control algorithms that operate on a finite set of switching states or on continuous input voltages. The **electric motor** is the centerpiece of every drive system. It is described by a system of ordinary differential equations (ODEs), which represents the motor's electrical behavior. In particular, the domain of three-phase drives makes use of coordinate transformations to view these ODEs in the more interpretable frame of field-oriented coordinates. In `GEM`, both, the physically accurate three-phase system (abc-coordinates) and the simplified, two-dimensional, field-oriented system (dq-coordinates) are available to be used as the frame of input and output variables, allowing for easy and quick controller analysis and diagnose within the most convenient coordinate system. Finally, the torque T resulting from the motor is applied to the **mechanical load**. The load is characterized by a moment of inertia and by a load torque T_{L} that is directed against the motor torque. Load torque behavior can be parameterized with respect to the angular velocity ω_{me} in the form of constant, linear and quadratic dependency (and arbitrary combinations thereof). Speed changes that result from the difference between motor and load torque are modeled with another ODE which completely covers the mechanical system behavior. Alternatively, the motor speed can be set to a fixed value, which can be useful for the investigation of control algorithms concerning generator operation, or it can be set to follow a specified trajectory, which is convenient when inspecting scenarios with defined speed demands like in traction applications.

Features

A large number of different motor systems is already implemented. These include DC drives as well as synchronous and induction three-phase drives. A complete list can be viewed in the `GEM` documentation (Balakrishna et al., 2020). The corresponding power converters allow to control the motor either directly via applied voltage (continuous-control-set) or by defining the converter's switching state (finite-control-set). Specifically for the use within reinforcement-learning applications and for testing state-of-the-art expert-driven control designs, the toolbox comes with a built-in reference generator, which can be used to create arbitrary reference trajectories (e.g., for the motor current, velocity or torque). These generated references are furthermore used to calculate a reward. In the domain of reinforcement learning, reward is the optimization variable that is to be maximized. For the control system scenario, reward is usually defined by the negative distance between the momentary and desired operation point, such that expedient controller behavior can be monitored easily. The reward mechanism also allows to take physical limitations of the drive system into account, e.g., in the way of a notably low reward if limit values are surpassed. Optionally, the environment can be setup such that a reset of the system is induced in case of a limit violation. In addition, built-in visualization and plotting routines allow to monitor the training process of reinforcement learning agents or the performance of expert-driven control approaches.

Examples

A minimal example of `GEM`'s simulation capability is presented in Figure 2. The plot shows the start-up behavior of a squirrel cage induction motor connected to an idealized three-phase electric grid depicting the angular velocity ω_{me}, the torque T, the voltage $u_{a,b,c}$ and the current $i_{d,q}$. Here, the voltage is depicted within the physical abc-frame while the current is viewed within the simplified dq-frame.

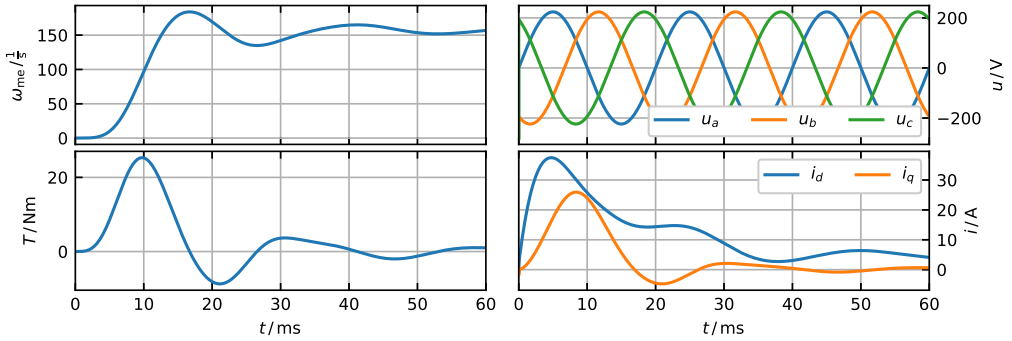

Figure 2: Simulation of a squirrel cage induction motor connected to a rigid network at $50\,\mathrm{Hz}$

Exemplary code snippets that demonstrate the usage of `GEM` within both, the classical control and the reinforcement learning context are included within the project's examples folder. Featured examples:

- `GEM_cookbook.ipynb`: a basic tutorial-style notebook that presents the basic interface and usage of GEM
- `scim_ideal_grid_simulation.py`: a simple motor simulation showcase of the squirrel cage induction motor that was used to create Figure 2

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., … Zheng, X. (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. https://www.tensorflow.org/

ANSYS, Inc. (2020). *ANSYS Maxwell 2020 R1*. https://www.ansys.com/products/electronics/ansys-maxwell

Balakrishna, P., Book, G., Kirchgässner, W., Schenke, M., Traue, A., & Wallscheid, O. (2020). *Gym-electric-motor documentation*. https://upb-lea.github.io/gym-electric-motor/

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). *OpenAI Gym*. https://github.com/openai/gym

Chollet, F., & others. (2015). *Keras*. https://keras.io

dSPACE GmbH. (2019). *VEOS 4.5*. https://www.dspace.com/en/inc/home/products/sw/simulation_software/veos.cfm

ElectroMagneticWorks, Inc. (2020). *MotorWizard*. https://www.emworks.com/product/MotorWizard

Guadarrama, S., Korattikara, A., Ramirez, O., Castro, P., Holly, E., Fishman, S., Wang, K., Gonina, E., Wu, N., Kokiopoulou, E., Sbaiz, L., Smith, J., Bartók, G., Berent, J., Harris, C., Vanhoucke, V., & Brevdo, E. (2018). TF-Agents: A library for reinforcement learning in TensorFlow. In *GitHub repository*. https://github.com/tensorflow/agents. https://github.com/tensorflow/agents

Kirchgässner, W., Schenke, M., Wallscheid, O., & Weber, D. (2020). *Paderborn university: Reinforcement learning course materials*. https://github.com/upb-lea/reinforcement_learning_course_materials

Margolis, B. W. L. (2017). SimuPy 1.0.0. In *GitHub repository*. https://github.com/simupy/simupy

Meeker, D. (2020). *FEMM 4.2*. http://www.femm.info/wiki/HomePage

Motor Design Ltd. (2020). *Motor-CAD v12*. https://www.motor-design.com/motor-cad-software/

OPAL-RT TECHNOLOGIES, Inc. (2020). *HYPERSIM 2020.1*. https://www.opal-rt.com/systems-hypersim/

Open Source Modelica Consortium. (2020). *OpenModelica*. https://openmodelica.org/. https://openmodelica.org/

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., … Chintala, S. (2019). PyTorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in neural information processing systems 32* (pp. 8024–8035). Curran Associates, Inc. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

Plappert, M. (2016). Keras-rl. In *GitHub repository*. https://github.com/keras-rl/keras-rl; GitHub. https://github.com/keras-rl/keras-rl

Raffin, A., Hill, A., Ernestus, M., Gleave, A., Kanervisto, A., & Dormann, N. (2019). Stable Baselines3. In *GitHub repository*. https://github.com/DLR-RM/stable-baselines3; GitHub.

Scilab. (2020). *Xcos (scilab 6.1.0)*. https://www.scilab.org/software/xcos. https://www.scilab.org/software/xcos

The MathWorks, Inc. (2020). *MATLAB - simulink R2020a*. https://www.mathworks.com/products/matlab.html