# GEKKO

GEKKO is a Python package for machine learning and optimization, specializing in time series and differential algebraic equations (DAE) systems. GEKKO provides a user-friendly interface to the APMonitor optimization suite. It is coupled with large-scale solvers for linear, quadratic, nonlinear, and mixed integer programming (LP, QP, NLP, MILP, MINLP). Modes of operation include parameter regression, dynamic data reconciliation, real-time optimization, dynamic simulation, and nonlinear predictive control.

## Installation

A pip package is available:

```
pip install gekko
```

The most recent version is listed on PyPi. You can upgrade from the command line with the upgrade flag:

```
pip install --upgrade gekko
```

GEKKO runs in remote mode (solved on high performance server) or on a local CPU when option `remote=False` such as `m=GEKKO(remote=False)`. The pip package includes the Windows executable (apm.exe), a Linux executable (apm), a MacOS executable (apm_mac), and a Linux ARM processor executable such as for a Raspberry Pi (apm_arm). Gekko can be used with remote server access (default option) for all operating systems.

## What does GEKKO do?

GEKKO is optimization software for machine learning and optimization of mixed-integer and differential algebraic equations. It is coupled with large-scale solvers for linear, quadratic, nonlinear, and mixed integer programming (LP, QP, NLP, MILP, MINLP). Modes of operation include data reconciliation, real-time optimization, dynamic simulation, and nonlinear predictive control. The client or server is freely available with interfaces in MATLAB, Python, or from a web browser.

GEKKO is a high-level abstraction of mathematical optimization problems. Values in the models are defined by Constants, Parameters, and Variables. The values are related to each other by Intermediates or Equations. Objective functions are defined to maximize or minimize certain values. Objects are built-in collections of values (constants, parameters, and variables) and relationships (intermediates, equations, and objective functions). Objects can build upon other objects with object-oriented relationships.

The APMonitor executable on the back-end compiles a model to byte-code and performs model reduction based on analysis of the sparsity structure (incidence of variables in equations or objective function) of the model. For differential and algebraic equation systems, orthogonal collocation on finite elements is used to transcribe the problem into a purely algebraic system of equations. APMonitor has several modes of operation, adjustable with the imode parameter. The core of all modes is the nonlinear model. Each mode interacts with the nonlinear model to receive or provide information. The 9 modes of operation are:

1. Steady-state simulation (SS)
2. Model parameter update (MPU)
3. Real-time optimization (RTO)
4. Dynamic simulation (SIM)
5. Moving horizon estimation (EST)
6. Nonlinear control / dynamic optimization (CTL)
7. Sequential dynamic simulation (SQS)
8. Sequential dynamic estimation (SQE)
9. Sequential dynamic optimization (SQO)

Modes 1-3 are steady state modes with all derivatives set equal to zero. Modes 4-6 are dynamic modes where the differential equations define how the variables change with time. Modes 7-9 are the same as 4-6 except the solution is performed with a sequential versus a simultaneous approach. Each mode for simulation, estimation, and optimization has a steady state and dynamic option.

APMonitor provides the following to a Nonlinear Programming Solver (APOPT, BPOPT, IPOPT, MINOS, SNOPT) in sparse form:

- Variables with default values and constraints
- Objective function
- Equations
- Evaluation of equation residuals

- Sparsity structure
- Gradients (1st derivatives)
- Gradient of the equations
- Gradient of the objective function
- Hessian of the Lagrangian (2nd derivatives)
- 2nd Derivative of the equations
- 2nd Derivative of the objective function

Once the solution is complete, APMonitor writes the results in results.csv that is loaded back into the python variables by GEKKO

When the system of equations does not converge, APMonitor produces a convergence report in 'infeasibilities.txt'. There are other levels of debugging that help expose the steps that APMonitor is taking to analyze or solve the problem. Setting APM.diaglevel to higher levels (0-10) gives more output to the user. Setting APM.coldstart to 2 decomposes the problem into irreducible sets of variables and equations to identify infeasible equations or properly initialize a model.

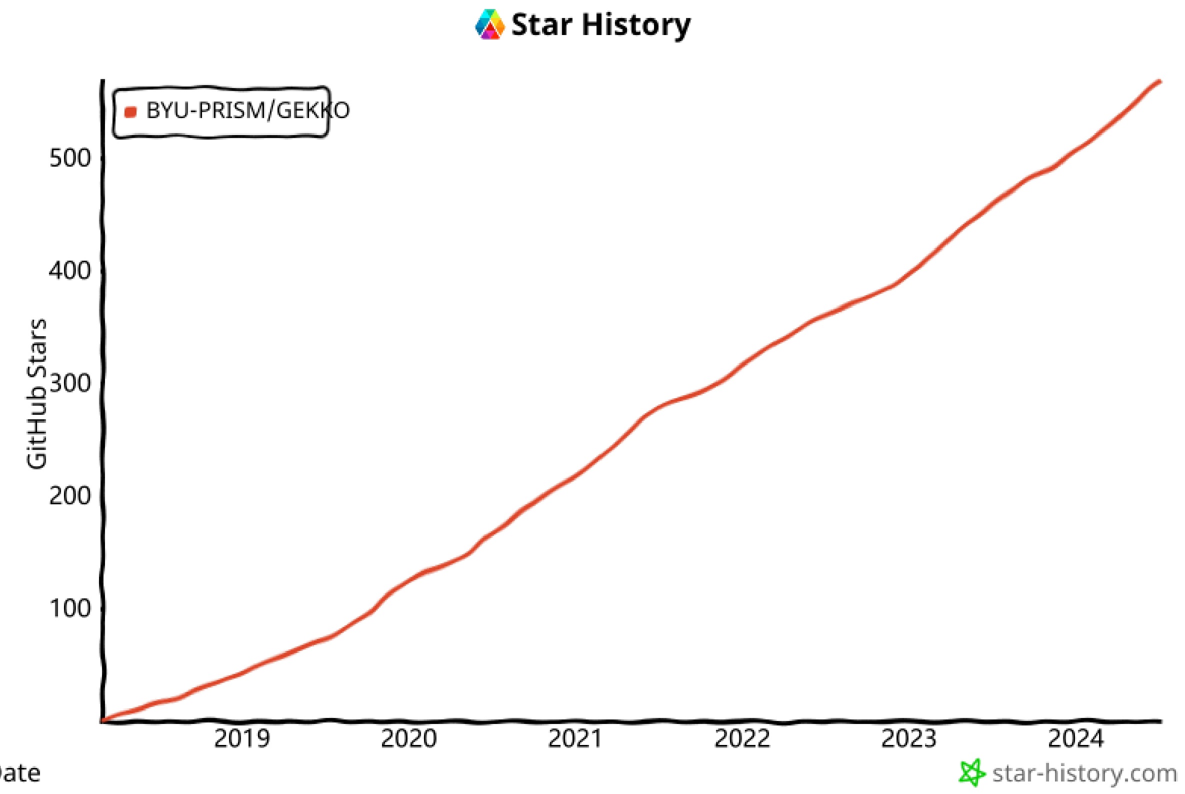
## Open-Source Modeling

Open-source modeling languages are essential tools for data analysis and scientific computing. Gekko is a mathematical programming language for optimization problems with physics-based and data-driven models. The importance of a collaborative development effort in open-source modeling languages cannot be overstated. Collaboration allows developers to pool their resources and knowledge, leading to faster and more robust development. Additionally, collaboration fosters a culture of sharing and support, helping to ensure the longevity and continued improvement of these essential tools. Below are select references to the Gekko Optimization Suite.

> Gunnell, L., Nicholson, B., Hedengren, J.D., Equation-based and data-driven modeling: Open-source software current state and future directions, Computers & Chemical Engineering, 2024, 108521, ISSN 0098-1354, DOI: 10.1016/j.compchemeng.2023.108521. Preprint

> Hedengren, J. D. and Asgharzadeh Shishavan, R., Powell, K.M., and Edgar, T.F., Nonlinear Modeling, Estimation and Predictive Control in APMonitor, Computers and Chemical Engineering, Volume 70, pg. 133–148, 2014, doi: 10.1016/j.compchemeng.2014.04.013. Preprint

> Beal, L.D.R., Hill, D., Martin, R.A., and Hedengren, J. D., GEKKO Optimization Suite, Processes, Volume 6, Number 8, 2018, doi: 10.3390/pr6080106. Article (Open Access)


Star History
BYU-PRISM/GEKKO
GitHub Stars
500
400
300
200
100
2019
2020
2021
2022
2023
2024
Date
star-history.com