The Journal of Open Source Software

Snowdrop: Python Package for DSGE Modeling

Alexei Goumilevski [1] and **James Otterson** [1]

1 International Monetary Fund

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Nikoleta Glynatsi

Reviewers:

- @dazhwu
- @phumthep

Submitted: 29 January 2025
Published: unpublished

Summary

At its core, `Snowdrop` is a robust and versatile Python package designed for the analysis of macroeconomic *Dynamic Stochastic General Equilibrium* (*DSGE*) models. This package provides an extensive framework for studying various related economic models, including *New Keynesian* models, *Real Business Cycle* models, *Gap* models, and *Overlapping Generations* models. `Snowdrop` equips researchers with essential tools to address the fundamental requirements of these models, encompassing estimation, simulation, and forecasting processes. In particular, the package employs robust and efficient solution techniques to solve both linear and nonlinear perfect foresight models based on the rational expectations hypothesis, which is critical for many DSGE models.

Statement of need

DSGE models are a fundamental class of models utilized by central banks worldwide, informing key monetary policy decisions (Botman et al., 2007), (Smets et al., 2010), (Del Negro et al., 2013), (Yagihashi, 2020). These models capture the dynamic evolution of economic variables influenced by agents who respond to anticipated future outcomes in the present. This necessitates the use of specialized techniques that are not readily available even in the extensive list of Python's scientific modeling packages (Fernández-Villaverde & Guerrón-Quintana, 2021). The three primary DSGE modeling toolboxes currently include DYNARE, IRIS and TROLL. While *Dynare* and *IRIS* are free and open-source software, they were primarily developed to run on the *MATLAB* platform, which is commercial. In contrast, *TROLL*, on the other hand, is a commerial application and requires subscription.

Each of these applications has its own advantage. The ease of use through a user-friendly interface, combined with the capability to handle a variety of models, has led to the immense popularity of *DYNARE* among general equilibrium modelers. However, *DYNARE* can only handle stationary *DSGE* models and requires users to write models in a stationary format by introducing variable deflators. The *IRIS* macroeconomic toolbox is another excellent tool that has gained popularity among economists for analyzing non-stationary *DSGE* models. *TROLL*, on the other hand, specializes in efficiently solving and simulating large systems of equations. All these applications, however, are either commercial or rely on commercial software that requires expensive licensing costs. To our knowledge, there is no integrated software package that is flexible enough to handle a wide range of models and available for free under the GNU General Public License agreements. This framework, built entirely on Python, aims to fill that void.

Benchmarking the Python Framework against the *DYNARE* and *IRIS* toolboxes for small to medium-sized models with several hundred equations demonstrates comparable CPU execution times but a smaller memory footprint, due to the significant memory requirements of *MATLAB*.

Highlights

- `Snowdrop` is a Python package that only uses open source libraries listed in the pypi repository.
- This package is platform neutral and can be run on Windows, Linux, Unix, and Mac machines.
- `Snowdrop` models can be written in user-friendly *YAML* format, pure Python scripts, or in a combination of both.
- Non-linear equations are solved iteratively via Newton's method. `Snowdrop` implements the *ABLR* stacked matrices and *LBJ* (Juillard et al., 1998) forward-backward substitution method to solve such systems. Linear models are solved with *Binder Pesaran's* method, *Anderson and More's* method and two generalized *Schur's* method that reproduce calculations employed in *Dynare* and *Iris*.
- Several desirable computational techniques for *DSGE* models are implemented in `Snowdrop`, including:
- Non-linear models can be run with time dependents parameters
- Goodness of fit of model data can be checked via the *Bayesian* approach to the maximization of likelihood functions.
- Model parameters can be sampled via the *Markov Chain Monte Carlo* affine invariant ensemble sampler algorithm of Jonathan Goodman and an adaptive Metropolis-Hasting's algorithms of Paul Miles. The former algorithm is useful for sampling badly scaled distributions of parameters. The later algorithm employs adaptive Metropolis methods that incorporate delayed rejection to stimulate samples' states mixing.
- Finally, `Snowdrop` streamlines the model production process by aiding users with the plotting and model reporting and storage process

Examples of model files and python code

The simplest way to write a `Snowdrop` model, is by specifing it via an *YAML* file in a manner that is familiar to *DYNARE* and *IRIS* users. Overall, the quickest way to run a model involves the following steps:

1. Create or modify an existing *YAML model file* in the models folder.
2. Open the *tests/test_toy_models.py* file and set *fname* to the name of your model file.
3. Run the Python script to obtain the desired simulations.

For example, the following specifies a simple monetary policy model with lagged variables.

Monetary policy model file

```yaml
name:  Monetary policy model example
symbols:
  variables: [PDOT,RR,RS,Y]
  exogenous: [ers]
  shocks: [ey]
  parameters: [g,p1,p2,p3,p4,p5,p6,p7]
  equations:
   - PDOT=p1*PDOT(+1)+(1-p1)*PDOT(-1)+p2*(g^2/(g-Y)-g)+p3*(g^2/(g-Y(-1))-g)
   - RR=RS-p1*PDOT(+1)-(1-p1)*PDOT(-1)
   - RS=p4*PDOT+Y+ers
   - Y=p5*Y(-1)-p6*RR-p7*RR(-1)+ey
  calibration:
   #Parameters
   g: 0.049
   #Set time varying parameters; the last value will be used for the rest of this ar
   p1: 0.414 #[0.4,0.5,0.6]
```

```
    std: 0.02
options:
    T: 14
    periods: [1]
    shock_values: [std]
```

Status

This toolkit provides users with an integrated Framework to input their models, import data, perform desired computational tasks (solve, simulate, calibrate, or estimate), and obtain well-formatted post-process output in the form of tables, graphs, etc. (Goumilevski et al., 2021). It has been applied in several cases, including studying the macroeconomic effects of monetary policy, estimating Peter's Ireland model (Ireland, 2004), and forecasting the economic effects of the COVID-19 virus, to name a few. The figure below illustrates the forecast of inflation, nominal and real interest rates, and the output gap in response to an output shock of 2% imposed at period 1 and a revision of the monetary policy rate of 3% imposed at period 4.

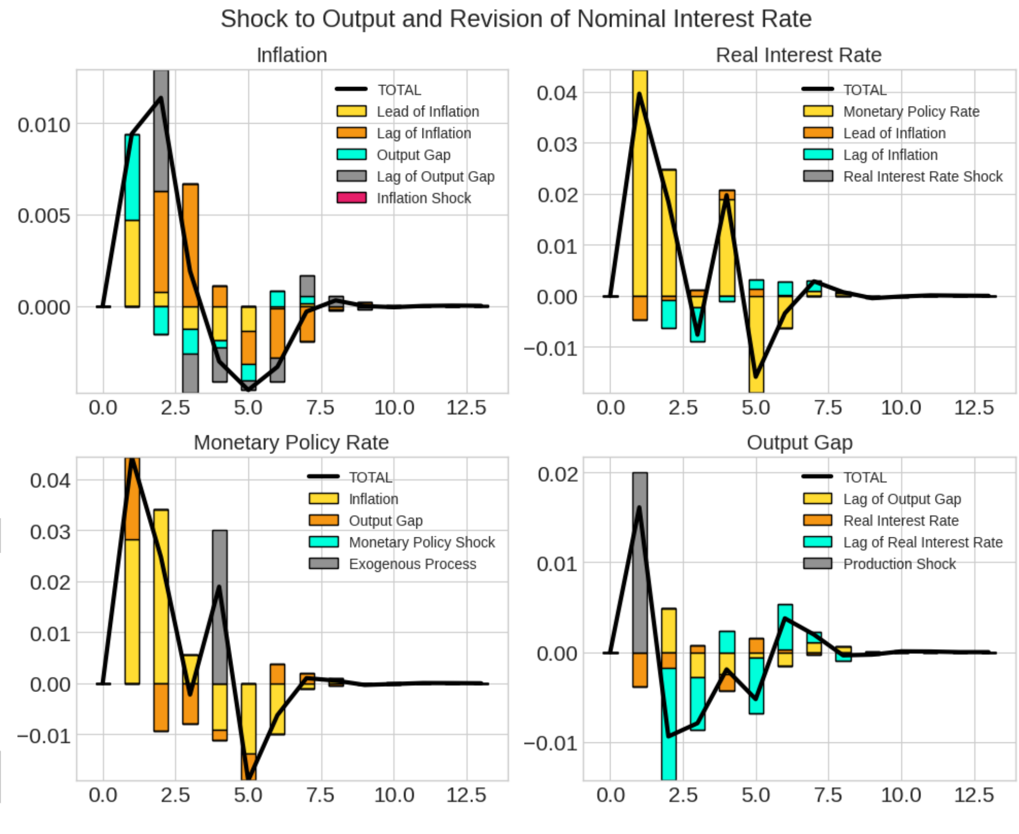

Figure 1: Monetary Policy Example

Another example illustrates the economic effects of the pandemic. We used the Eichenbaum-Rebelo-Trabandt (ERT) model (Eichenbaum et al., 2020), which embeds epidemiological concepts into a New Keynesian modeling framework. We assumed that there are two strains of pathogens and employed a Suspected-Infected-Recovered (SIR) epidemiological model. These epidemiological equations were incorporated into the ERT model, consisting of sixty-four equations of macroeconomic variables from sticky and flexible price economies. The macroeconomic variables of these two economies were linked through a Taylor rule equation for the policy interest rate. The model is highly non-linear and is solved using a homotopy method,

The Journal of Open Source Software

where parameters are adjusted step-by-step. We assumed that government containment measures were more lenient during the second strain of the virus compared to the first one. Figures 2 and 3 illustrate the forecast of virus transmission and deviations of macroeconomic variables from their steady state.

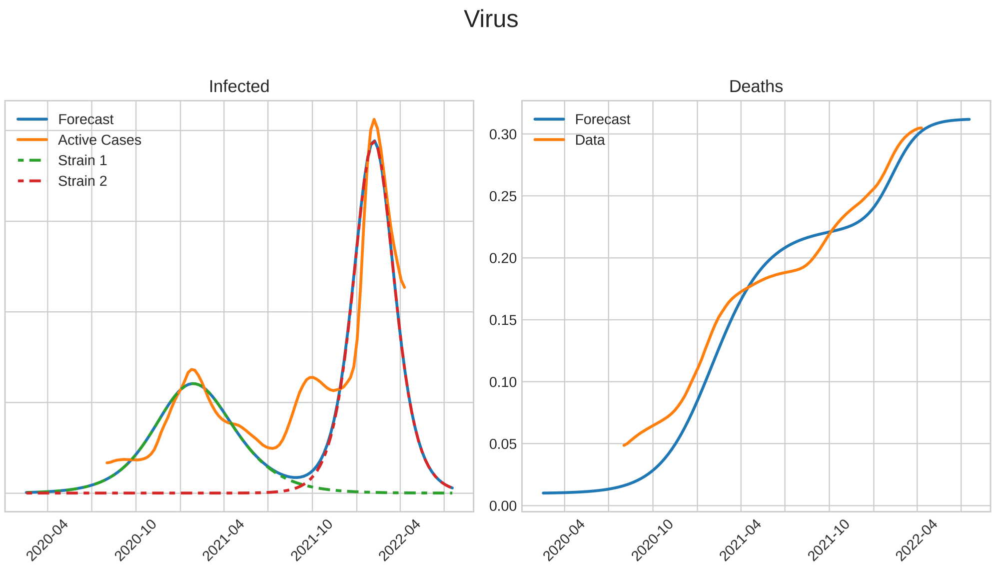

Figure 2: Epidemic Forecast

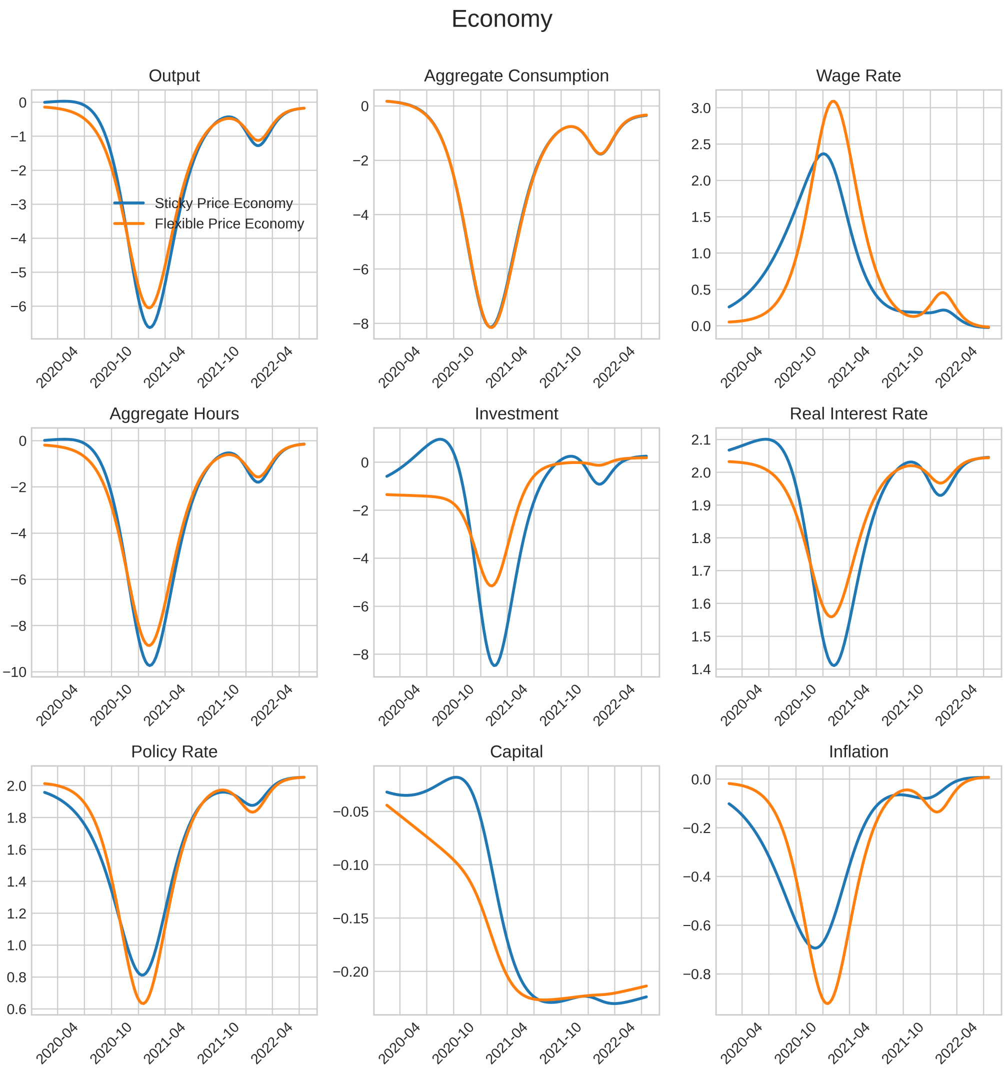

Figure 3: Forecast of Macroeconomic Variables

Acknowledgements

The authors would like to thank Douglas Laxton for initiating this project, Farias Aquiles for his guidance and support, and Kadir Tanyeri for his valuable comments.

References

Botman, D. P., Karam, P. D., Laxton, D., & Rose, D. (2007). *DSGE modeling at the fund: Applications and further developments*. https://doi.org/10.5089/9781451867640.001

Del Negro, M., Eusepi, S., Giannoni, M. P., Sbordone, A. M., Tambalotti, A., Cocci, M., Hasegawa, R., & Linder, M. (2013). The FRBNY DSGE model. *FRB of New York Staff Report*, *647*. https://doi.org/10.2139/ssrn.2344312

Eichenbaum, M., Rebelo, S., & Trabandt, M. (2020). The macroeconomics of epidemics. *NBER Working Paper, No. 26882*. https://doi.org/10.3386/w26882

Fernández-Villaverde, J., & Guerrón-Quintana, P. A. (2021). Estimating DSGE models: Recent advances and future challenges. *Annual Review of Economics*, *13*(1), 229–252. https://doi.org/10.3386/w27715

Goumilevski, A., Farias, A., & Tanyeri, K. (2021). A new macroeconomic modeling framework applied to assess effects of COVID-19. *Proceedings of the 27th International Conference on Computing in Economics and Finance*. https://www.academia.edu/117687479/A_New_Macroeconomic_Modeling_Platform_Applied_to_Assess_Effects_of_COVID_19

Ireland, P. (2004). Technology shocks in the new keynesian model. *NBER Working Paper, No. 10309*. https://doi.org/10.3386/w10309

Juillard, M., Laxton, D., McAdam, P., & Pioro, H. (1998). An algorithm competition: First-order iterations versus newton-based technique. *Journal of Economic Dynamics and Control, No. 22, Pp.1291—1318*. https://doi.org/10.1016/S0165-1889(98)00013-X

Smets, F., Christoffel, K., Coenen, G., Motto, R., & Rostagno, M. (2010). DSGE models and their use at the ECB. *SERIEs*, *1*, 51–65. https://doi.org/10.1007/s13209-010-0020-9

Yagihashi, T. (2020). DSGE models used by policymakers: A survey. *Policy Research Institute, Ministry of Finance, Japan, PRI Discussion Paper Series*. https://doi.org/10.3390/econometrics4010001