

# Snowdrop: Python Package for DSGE Modeling

Alexei Goumilevski<sup>1</sup> and James Otterson<sup>1</sup>

<sup>1</sup> International Monetary Fund

DOI: [10.xxxxxx/draft](https://doi.org/10.xxxxxx/draft)

## Software

- [Review](#)
- [Repository](#)
- [Archive](#)

Editor: [✉](#)

Submitted: 29 January 2025

Published: unpublished

## License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](https://creativecommons.org/licenses/by/4.0/)).

## Summary

At its core, Snowdrop is a robust and versatile Python package designed for the analysis of macroeconomic *Dynamic Stochastic General Equilibrium* (DSGE) models. In its entirety, this package offers an extensive framework for the study of 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 a critical need for many DSGE models.

## Statement of need

DSGE models are a mainstay class of models employed by Central Banks around the world, informing key country 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, necessitating the combined 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). Currently, the two primary DSGE modeling toolboxes, DYNARE and IRIS are comprehensive toolsets that offer an user-friendly infrastructure with support to all stages of model development. These, and similar, applications, however, are either commercial, or rely on commercial software to run, and hence require expensive licensing costs. There is no integrated software package to our knowledge that is both flexible to handle a wide class of models with all required software to run the models available for free under the GNU General Public Licensing agreements. This Framework, built entirely on Python, is intended to fill that void.

## 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 M., 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.

- 41     ▪ Several desirable computational techniques for *DSGE* models are implemented in
- 42       Snowdrop, including:
- 43         – Non-linear models can be run with time dependents parameters
- 44         – Goodness of fit of model data can be checked via the *Bayesian* approach to the
- 45           maximization of likelihood functions.
- 46         – Model parameters can be sampled via the *Markov Chain Monte Carlo* affine invariant
- 47           ensemble sampler algorithm of Jonathan Goodman and an adaptive Metropolis-
- 48           Hasting's algorithms of Paul Miles. The former algorithm is useful for sampling
- 49           badly scaled distributions of parameters. The later algorithm employs adaptive
- 50           Metropolis methods that incorporate delayed rejection to stimulate samples' states
- 51           mixing.
- 52     ▪ Finally, Snowdrop streamlines the model production process by aiding users with the
- 53       plotting and model reporting and storage process.

## 54   Examples of model files and python code

55   The simplest way to write a Snowdrop model, is by specifying it via an *YAML* file in a manner

56   that is familiar to *DYNARE* and *IRIS* users. Overall, the quickest way to run a model involves

57   the following steps:

- 58       1. Create or modify existing *YAML model file* in models folder.
- 59       2. Open *src/tests/test\_toy\_models.py* file and set *fname* to the name of this model file.
- 60       3. Run the python script to get the desired simulations.

61   For example, the following specify a simple growth model with lagged variables.

## 62   Monetary policy model file

```

name: Monetary policy model example
symbols:
  variables: [PD0T,RR,RS,Y]
  exogenous: [ers]
  shocks: [ey]
  parameters: [g,p1,p2,p3,p4,p5,p6,p7]
  equations:
    - PD0T=p1*PD0T(+1)+(1-p1)*PD0T(-1)+p2*(g^2/(g-Y)-g)+p3*(g^2/(g-Y(-1))-g)
    - RR=RS-p1*PD0T(+1)-(1-p1)*PD0T(-1)
    - RS=p4*PD0T+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]
```

## 63   Imposing shocks

```

# Create model object
from snowdrop.src import driver
model = driver.importModel(model_file_path)
```

```
# Set shocks
model.options["periods"] = [1]
model.options["shock_values"] = [0.02]
# Define list of variables for which decomposition plots are produced
decomp = ['PDOT','RR','RS','Y']
# Run forecast
y, dates = driver.run(model=model, decomp_variables=decomp, Plot=True)
```

#### 64 Anticipated, unanticipated shocks, and judgmental adjustments

```
from snowdrop.src.driver import run
# Set shock for gap of output to 1% at period 3
d = {"SHK_L_GDP_GAP": [(3,1)]}
model.setShocks(d)
model.anticipate = True
# Impose judgments
date_range = pandas.date_range(start, end, freq="QS")
m = {"L_GDP_GAP": pandas.Series([-1.0, -1.0, -1.0], date_range)}
shocks_names = ["SHK_L_GDP_GAP"]
# Endogenize shock and exogenize output gap endogenous variable
model.swap(m, shocks_names)
# Run simulations
y, dates = driver.run(model)
```

#### 65 Status

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

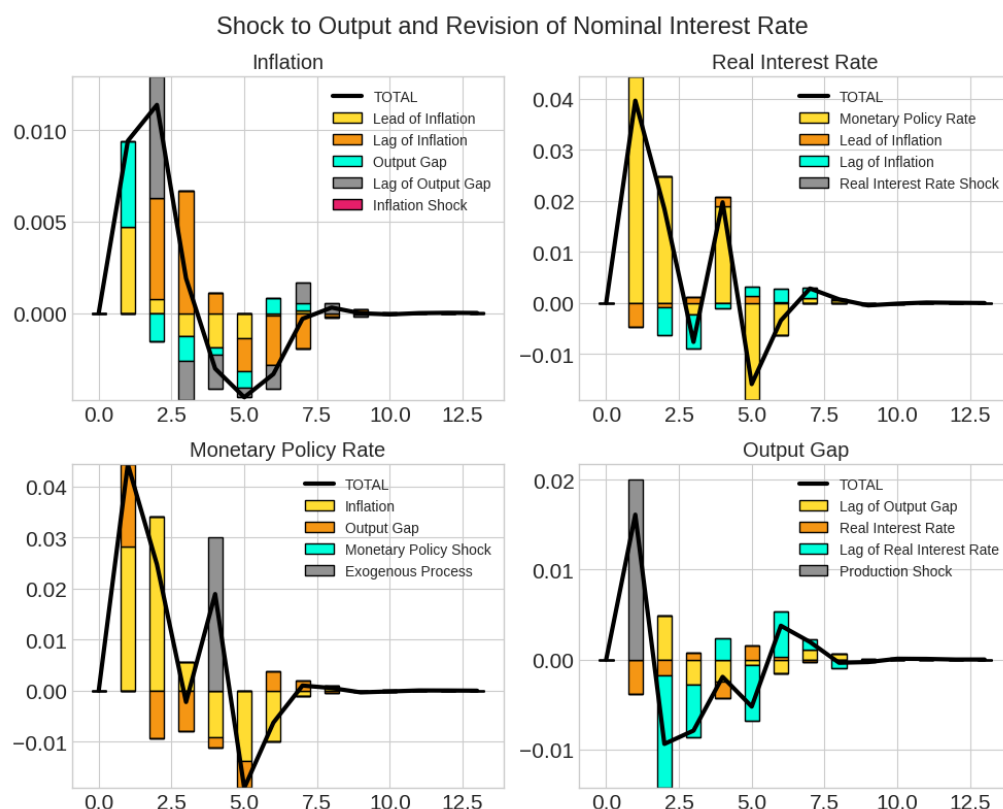


Figure 1: Monetary Policy Example

Another example illustrates economic effects of pandemic. We used Eichenbaum-Rebelo-Trabandt (ERT) model (Eichenbaum M., 2020) which embeds epidemiological concepts into *New Keynesian* modelling framework. We assumed that there two strains of pathogens and employed Suspected-Infected-Recovered (SIR) epidemiological model. These epidemiological equations were plugged in into ERT model consisting of sixty-four equations of macroeconomic variables of sticky and flexible price economies. The macroeconomic variables of these two economies were linked thru Taylor rule equation for policy interest rate. Model is highly non-linear and is solved by using a homotopy method where parameters are adjusted step-by-step. We assumed that the government containment measures were more lenient during the second strain of virus compared to the first one. Figures 2 and 3 illustrate forecast of virus transmission and deviations of macroeconomic variables from their steady state.

## Virus

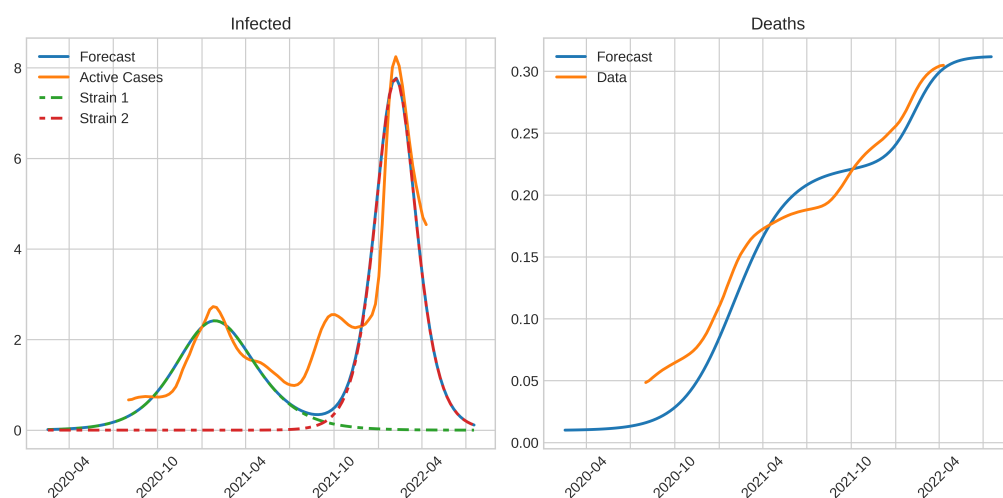
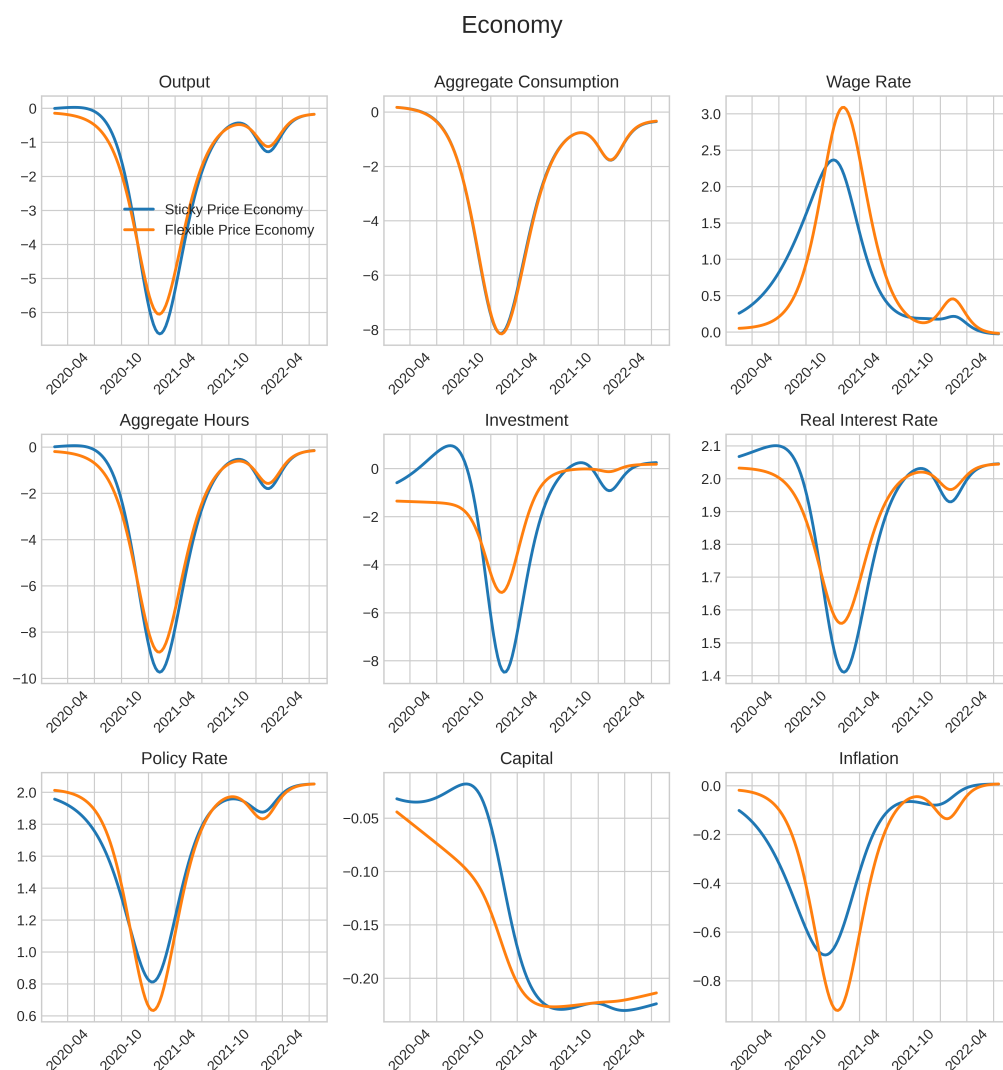


Figure 2: Epidemic Forecast



**Figure 3: Forecast of Macroeconomic Variables**

## Acknowledgements

Authors would like to thank Doug 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*.
- 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.
- Eichenbaum M., T. M., Rebelo S. (2020). The macroeconomics of epidemics. *ER Working Paper*, No. 26882.

- 96 Fernández-Villaverde, J., & Guerrón-Quintana, P. A. (2021). Estimating DSGE models: Recent  
97 advances and future challenges. *Annual Review of Economics*, 13(1), 229–252.
- 98 Goumilevski A., T. K., Farias A. (2021). A new macroeconomic modeling framework applied  
99 to assess effects of COVID-19. *Proceedings of the 27th International Conference on*  
100 *Computing in Economics and Finance*.
- 101 Juillard M., M. P., Laxton D. (1998). An algorithm competition: First-order iterations  
102 versus newton-based technique. *Journal of Economic Dynamics and Control*, No. 22,  
103 Pp.1291—1318.
- 104 P., I. (2004). Technology shocks in the new keynesian model. *NBER Working Paper*, No.  
105 10309.
- 106 Smets, F., Christoffel, K., Coenen, G., Motto, R., & Rostagno, M. (2010). DSGE models and  
107 their use at the ECB. *SERIEs*, 1, 51–65.
- 108 Yagihashi, T. (2020). DSGE models used by policymakers: A survey. *Policy Research Institute*,  
109 *Ministry of Finance, Japan, PRI Discussion Paper Series*.

DRAFT