`covbayesvar`: A Python Package for Large Bayesian VAR Model with COVID Volatility

Sudiksha Joshi[1]
University of Washington

Abstract

This paper introduces `covbayesvar`, a Python package for estimating large Bayesian Vector Autoregression (BVAR) models, that account for COVID-induced volatility. The package enables us to estimate the model with hierarchical prior selection when the parameters proliferate, accounting for structural shifts in macroeconomic and financial data during the pandemic. Incorporating various priors, it is versatile enough to answer wide-ranging policy-related questions. With detailed programming examples, I explain how to apply the functions on monthly and quarterly macro and financial data to construct unconditional and conditional forecasts, scenario analysis, assess joint predictive densities of variables, examine structural breaks during and after COVID, and how we can employ entropic tilting to modify the forecast distribution and construct forecasts conditional on long-term targets set by the Federal Reserve Bank. Accessible via PyPI, the package includes extensive documentation and code examples. `covbayesvar` advances the state-of-the-art in open-source econometric and statistical software, offering researchers a robust tool for analyzing large-scale systems under unprecedented uncertainty.

Keywords: Bayesian statistics, vector autoregressions, time series, macroeconometrics, hierarchical model, forecasts, Minnesota prior, Inverse-Wishart prior, scenario analysis, entropic tilting, FRED-MD, FRED-QD, conditional probability, joint predictive density, structural breaks

1. Introduction

Macroeconometric forecasting often entails smaller time series of historical data, particularly, if we forecast using frequentist vector autoregressions (VARs) at the monthly frequency. Central banks and financial institutions employ vector autoregressions to forecast future economic conditions that assess the impact of monetary policy transmission mechanisms to understand how policy instruments such as interest rates affect the dynamic relationship between

[1]

- Please email me at `joshi27s@uw.edu` if you have questions or comments. I am grateful to Domenico Giannone for his invaluable advice and suggestions.
- This package, accompanying research paper, codes and supplementary materials are not affiliated with, endorsed from, or sponsored by Amazon.com, its subsidiaries, or its employees. I independently wrote the functions of the package for research purposes prior to joining Amazon.com. Prior to my internship at Amazon, I had mapped the python functions one-to-one with the publicly-available MATLAB functions. I am solely responsible for any omissions or errors.

inflation, output, and unemployment. Among these are examining how fiscal stimulus or tax changes affect debt burden, can the US economy lands softly, and how high should the mortgage rates be to slow down the real estate market. This is critical in making decisions that have far-reaching consequences for the economy. Regulators and banks use these models to simulate adverse economic scenarios and analyze how banks will perform under stress such as severe recessions, rising defaults on loans, and diminished cash reserves.

Albeit these flexible time series models capture convoluted dynamic relationships among macro and financial variables, they become densely parametrized, yielding imprecise out-of-sample forecasts. Creating a problem of the "curse of dimensionality" where the number of observations exceeds the number of parameters to estimate, statisticians and economists have introduced Bayesian methods to address the problems of unstable and burgeoning parameters. Amongst those are BVARs with informative priors that shrink the overly parameterized models towards parsimonious models with sparse coefficients, developed by Giannone Lenza and Primiceri (2015). However, this methodology isn't necessarily most suitable to estimate a model on the pandemic-driven extreme observations, when key macro variables such as unemployment rate, oil prices, industrial production, and retail sales vary substantially. For instance, WTI oil prices nosedived to -53 percent in March 2020 compared to one year ago, then skyrocketed to 272 percent in April 2021 when measured in year-over-year growth rates. Such extreme outliers warrant changing the estimation strategy to account for wider uncertainty when acutely large shocks propagate through the economy and skew the distribution of the time series data.

Owing to the contaminated data from the COVID outbreak, I forecast and estimate the responses to shocks when extreme observations such as those seen during the pandemic, starting from March 20, 2020, are present. Recognizing that the COVID-19 pandemic altered the statistical properties of a plethora of economic and financial variables, such as the moments, and autocorrelation, I employ a Bayesian inferential approach to combine standard prior beliefs with the information present in the data. Currently, there is no equivalent to the MATLAB estimation of BVAR models in Python that incorporates covid volatility in big data. Therefore, in Python, I forecast and estimate the responses to shocks when extreme observations such as those seen during the pandemic, starting from March 20, 2020, are present. Incorporating the Bayesian VAR developed by Primeceri and Lenza (2021), I first fit a Bayesian VAR model with covid volatility on medium-sized macro and financial variables data with the model specifications from Primiceri and Lenza (2022). Unlike their model with seven variables, I added numerous financial variables in a model typically used to analyze macro variables, showcasing use-cases with the monthly data to 28 variables, and a quarterly data of 29 variables. Macroeconomists often program their models in MATLAB. So, I first translated the MATLAB codes to Python, and built a Python package called `covbayesvar`, saving the functions in `large_bvar` module of the package. Users can install this package as it is available in the Python Package Index (PyPI), and can view the source code of the functions in the Github repository. It implements the modeling approach to prior selection and conditional forecasting as done in Giannone et. al (2015), Banbura, Giannone and Lenza (2015), Lenza and Primiceri (2022), and Crump et. al (2021). This package encompasses all the functions necessary to transform the data, estimate the model for inferences, calculate reduced-form impulse responses, construct beautiful unconditional (baseline) forecasts based on the historical correlations between variables, forecasts conditional on shocks on certain variables for given time horizons, plot scenario analysis of those shocks, plot joint distribution of forecasts, examine structural breaks, and employ entropic tilting to make projections conditional on long run targets set by the Federal Reserve Bank. To illustrate the usage of these functions, I've created and shared Google Colab Notebooks that showcase how to construct the model using macro and financial data and make forecasts. Furthermore, I published a separate documentation of all the functions along with examples in Github.

Several packages in R exist to estimate BVAR using varying estimation strategies. For instance, in the `bsvars` R package, Wozniak (2024) implemented the techniques in Lutkepohl, Shang and Wozniak (2024) to create an efficient

procedure to estimate Structural BVAR model with homoskedastic, heteroskedastic, and non-normal specifications. Kruegar (2015) developed an R package called `bvarsv` to estimate Primiceri (2005)'s Time Varying Parameter VAR and construct impulse responses and posterior predictive distribution. The `bvartools` R package contains algorithms for Bayesian inference of VAR and vector error correction models, leveraging functions from the introductory texts of Chan, Koop, Poirier and Tobias (2019) and Koop and Korobilis (2010) to simulate the posterior distributions, forecast, and create forecast error variance decomposition. Similarly, Kuschnig and Vashold (2021) developed the `BVAR` package to estimate the model with hierarchical prior selection, featuring structural analysis of forecasts and impulse responses using conjugate priors. The `BayVAR_R` package by Quilis (2022) models both classical and Bayesian versions, and the `mfbvar` package implements Bayesian mixed frequency VAR models. Complementing the structural VARs is the `bsvarSIGNs` package that constructs the structural BVAR model identified by zero, sign and narrative restrictions. Notwithstanding the plethora of R packages, none exists in python. `covbayesvar` is the first open-source python package to estimate the model with COVID-volatility as explained in Lenza and Primiceri (2022), and apply the method to a large number of variables as shown in Crump et. al (2021). Whilst researchers may drop the data seen in COVID months for estimation, disregarding the recent data is inappropriate to forecast the future trends of the economy as it underestimates uncertainty.

Next, I briefly introduce the BVAR model mathematically in Section 2 and elucidate the important estimation and forecasting functions in Section 3. Section 4 presents a simple Monte Carlo simulation of the constant volatility BVAR model to test its performance, where I evaluate how the simulated posterior estimates of the parameters fare relative to the true hypothesized values. Subsequently, in section 5, I discuss the monthly economic and financial data that the model estimates, present the forecasting results, a scenario analysis exercise, and a routine to condition the forecasts on Fed's long-run targets using entropic tilting. Section 6 demonstrates a diverse array of applications of the model using quarterly data. Section 7 replicates three figures from Lenza and Primiceri (2022) to juxtapose how the forecasts and impulse responses from the COVID-volatility model fare relative to those from the constant volatility model. Finally, section 8 concludes.

2. Model

Lenza and Primiceri's (2022) method explicitly models the changing volatilities in shocks and incorporates significant financial and macroeconomic innovations during the pandemic. Unlike other models that explain volatilities such as TVP-VAR, we know when the shock hit the economy, i.e., the variance of the innovations is known. So, I estimate the model and measure changes in the volatility, keeping in mind that the volatility can persist for several periods in the future. As the first extreme observation was in March 2020, I rescale the standard deviation of the shocks in March, April, and May to unknown parameters η_1, η_2, η_3, and estimate these parameters using Bayesian methods explained in Giannone et al. (2015).

Then, I estimate the variance of the residuals after March 2020, assuming that the volatility decays at a constant rate every month henceforth. The main model is a VAR(12):

$$y_t = \alpha + A_1 y_{t-1} + \cdots + A_{12} y_{t-12} + \eta_t \epsilon_t, \quad \epsilon_t \sim N(0, \Sigma)$$

where $\eta_t = \eta_1$ before the pandemic began, and scales up the residual covariance matrix during the pandemic at time t^*.

$$\eta_t = \begin{cases} \eta_1 & t = t^* \\ \eta_2 & t = t^* + 1 \\ \eta_3 & t = t^* + 2 \\ \eta_{t+j} = 1 + (\eta_3 - 1) n_4^{-j/2} & \text{otherwise.} \end{cases}$$

The scaling factors take distinct values in the initial three periods after the pandemic began, before decaying at the rate η_4 henceforth. Since more lags proliferate the number of parameters in the model, we cannot feasibly estimate the model by OLS, especially when the "curse of dimensionality" precludes estimation in a richly parameterized model. Thus, I shrink the autoregressive parameter coefficients using the Minnesota prior. This prior presumes that the variables follow a random walk with drift, and shrink the coefficients of faraway lags. I also impose the natural conjugate prior, wherein I draw the elements of the variance-covariance matrix Σ from the inverse-Wishart distribution. Then, I obtain the posterior estimates of the parameters, fit generalized impulse responses, and forecast. More specifically, to estimate equation (1), if we assume that the scaling factor η_t is known, then we can rewrite equation (1) where I stack coefficients as:

$$y_t = X_t \Phi + \eta_t \epsilon_t,$$

where $X_t = I_k \otimes [1, y'_{t-1}, y'_{t-2}, \dots, y'_{t-12}]$ and $\Phi = \text{vec}([\alpha, A_1, A_2, \dots, A_{12}])$.

Normalizing the coefficients to account for the scaled idiosyncratic error, I divide equation (2) by f_t as:

$$\frac{y_t}{\eta_t} = \frac{X_t}{\eta_t}\Phi + \epsilon_t \quad \Rightarrow \quad \tilde{y_t} = \tilde{X}_t \Phi + \epsilon_t.$$

Using the transformed data to estimate the parameters $\tilde{y_t}$ and $\tilde{X}_t$, firstly, the prior distribution for the VAR error variance-covariance matrix Σ is the normal inverse-Wishart with the scale S and degrees of freedom $n+2$. In other words, $\Sigma \sim IW(S, n+2)$.

Secondly, the conditional distribution of the coefficient matrix Φ given the estimates of the variance-covariance matrix Σ is:

$$\Phi | \Sigma \sim N(\psi, \Sigma \otimes \Omega).$$

Thirdly, I characterize extreme values via the Pareto distribution, where the prior belief on all the scaling factors is the Pareto distribution with the unit values as the scale and shape parameters. To corroborate, the shape parameter connotes the speed at which the tail falls off, and the scale parameter regulates the threshold above which the distribution becomes conspicuously relevant. The unit values indicate that the Pareto distribution has a very long and flat tail, attributing that most of the probability mass is concentrated in lower values and the higher values occasionally occur. This is compatible with the belief that the variance of the errors is large during the pandemic:

$$\eta_t \sim Pareto(1, 1), \quad t = 0, 1, 2.$$

Fourthly, I impose a single unit root prior to reflect a belief with Bernoulli probability p that a series might be non-stationary or random walk. The single unit root prior reflects this persistence without strictly enforcing stationarity,

unlike the Minnesota prior, which assumes stationarity. This Bernoulli probability p that a given series in unit root follows a Beta prior with the shape parameters (α, β) to determine how strongly we believe a series is likely non-stationary. Using a Beta distribution with a large mode of 0.8 and a standard deviation of 0.2 reflects a belief that we expect that series is highly persistent with some degree of uncertainty.

Finally, I employ another shrinkage prior, known as the sum of coefficients prior, to shrink the coefficients towards zero to preclude an overfit model. Conjecturing that a no-change forecast is a good forecast when the sample begins, it tapers the importance of deterministic components in a VAR estimated condition on initial observations. The deterministic element is:

$$E[y_t \,|\, Y_1, Y_2, ..., Y_{12}, \Phi].$$

Formulated using the Theil Mixed Estimation, it forms conjugate priors by constructing artificial data and appending them when the actual sample ends. Centering the prior belief that all the coefficients on their own lags sum to 1 in each equation:

$$(A_1 + A_2 + \cdots + A_{12}) \,|\Sigma \sim N\left(I_k, V(\omega, \tilde{y}_0, \Sigma)\right),$$

where ω is the hyperparameter that regulates how dispersed the prior beliefs are. As $\omega \to \infty$, the prior is flat, and as $\omega \to 0$, the model has a unit root in each equation, ruling out cointegrated variables. It creates a sparse model as it restrains higher probability mass on smaller values of the coefficients, curtailing the number of parameters in the model. Whereas the Minnesota prior allows us to identify the important variables in the model, the sum of the coefficients prior identifies few important coefficients. Therefore, utilizing both the sum of coefficients and Minnesota priors in the same model additionally regularizes the model and allows the different types of priors to capture different features of the data. To estimate the posterior density of the parameters, I adopt a hierarchical technique of sampling using the Metropolis-Hastings algorithm defined by Primiceri and Lenza (2022). Briefly, first, I initialize the hyperparameters at their posterior mode; and draw the candidate values of these hyperparameters from the proposal distribution. Pre-specifying the acceptance rate at roughly 25 percent, I accept and reject the proposed draws of hyperparameters; then draw the coefficients Φ and variance-covariance matrix Σ from the normal-inverse Wishart density function.

3. Python Implementation: Main Functions

This section sheds light on four main functions in the `covbayesvar` package that are instrumental in defining the prior distributions, estimating the model to update the posterior distributions, and estimating and plotting the unconditional and conditional forecasts.

Priors and Hyperparameters

We assume external beliefs via additional inputs in the `set_priors_covid` function. Here, the priors enable us to regularize or shrink the parameter space that prevents overfitting when data is insufficient to estimate the parameters reliably, and the economy experiences sudden structural shifts.

The function returns a tuple with several dictionaries and lists, where each argument represents:

1. `r` **(Priors)**:
   A dictionary containing the priors for the BVAR model. This includes settings like `hyperpriors`, `Vc`, `pos`, and the hyperparameters for the Minnesota prior `MNalpha`, the time horizon for forecasts `hz`, and MCMC-related parameters such as the total number of draws in the MCMC simulation `Ndraws` and the initial number of draws burnt-in `Ndrawsdiscard`.

2. `mode` **(Hyperpriors' Mode Values)**:
   A dictionary containing the mode values for hyperpriors, such as:

   - `lambda`: The tightness of the prior for coefficients on own lags.
   - `miu`: A prior for the sum of coefficients.
   - `theta`: Controls the overall shrinkage.
   - `eta`: Hyperparameters related to volatility changes due to the onset of COVID-19.

3. `sd` **(Standard Deviations for Hyperpriors)**:
   This dictionary contains the standard deviations for the hyperpriors `lambda`, `miu`, `theta`, `eta`, reflecting the uncertainty around their respective mode values.

4. `priorcoef` **(Priors' Coefficients)**:
   A dictionary of coefficients for the hyperpriors, calculated using functions like `gamma_coef()` for gamma priors and `beta_coef()` for beta priors. These coefficients are necessary for estimating the posterior distribution in the BVAR model.

5. `MIN` **and** `MAX` **(Bounds for Optimization)**:
   These dictionaries set lower and upper bounds for variables in the maximization process. For example, `MIN['lambda] = 0.0001` ensures that the prior's tightness cannot be smaller than 0.0001, and `MAX['lambda] = 5` prevents it from exceeding 5.

6. `albet` **and** `mosd` **(Beta Distribution Parameters)**:
   These are parameters related to the Beta distribution that affect volatility `eta`. The `alpha` and `beta` parameters describe the distribution of hyperparameters that govern the dynamics during COVID-19.

Estimating the BVAR

The function `bvarGLP_covid` estimates the Bayesian VAR model after considering changes in volatility due to the COVID-19 pandemic. Originally founded upon the model derived by Giannone, Lenza and Primiceri (2015) that utilizes Markov chain Monte Carlo (MCMC) methods, this function additionally captures volatile movements starting from March 2020 to estimate the posterior distribution of the model parameters.

Input Parameters

1. `y` : The matrix of economic time series data, where rows represent time periods and columns represent different variables.

2. `lags` : The number of lags to include in the VAR model.

3. `kwargs`: Additional arguments to specify the BVAR model settings, including:

- `mcmc`: Indicator for running MCMC.
- `MCMCconst`: MCMC constant.
- `MNpsi` : Minnesota prior hyperparameter.
- `sur`: Indicator for seemingly unrelated regressions in the model.
- `noc` : Indicator for no constant in the model.
- `Ndraws` : Number of MCMC draws.
- `hyperpriors` : Indicator for using hyperpriors.
- `Tcovid` : Time index that determines the onset of volatility due to COVID-19.

The function sets the priors using the `set_priors_covid` function which prepares various hyperparameters needed to estimate the model. Next, it prepares the input data matrix `y` by constructing lagged values as regressors. This involves creating a matrix `x` that includes a constant term and the lagged values of `y`. With the prepared inputs, it constructs the Minnesota prior mean vector `b` and defines the initial values for the key hyperparameters `lambda`, `theta`, `mu`, `alpha` that control the prior distributions. Later, it adjusts for volatility, particularly emphasizing changes post-COVID-19. To establish scaling factors, it compares the volatility before and after COVID-19. Fitting an AR(1) model to each variable, it estimates its residual variance and numerically optimizes over hyperparameters using the `csminwel` function to maximize the log marginal likelihood of the VAR model.

Unconditional and Conditional Forecast

Crump et. al (2019) define unconditional predictive density as:

$$f\left(y_{T+1}, \cdots, y_{T+h} \mid y_{1:T}\right) = \int f\left(y_{T+1}, \cdots, y_{T+h} \mid y_{1:T}, \theta\right) f\left(\theta \mid y_{1:T}\right) d\theta$$

where, θ has all the coefficients of the model. Computing the predictive density requires iterating in two steps. First, we draw the coefficients from their posterior distribution. For instance, at the m-th draw, the posterior estimate is $\theta^{(m)}$. Then, given the posterior estimate, we sample from:

$$f\left(y_{T+1}, \cdots, y_{T+h} \mid y_{1:T}, \theta^{(m)}\right)$$

by drawing from the forecast errors $\epsilon_{T+1}, \epsilon_{T+2}, \ldots, \epsilon_{T+h}$ and iterating the model forward using the simulation smoother.

Using the VAR model estimates, the function `VARcf_DKcks` calculates the conditional forecasts by applying the Kalman filter and smoother. After reformulating the model in state-space form, we can either generate point forecasts using the Kalman filter or generate forecasts at various quantiles using the Durban-Koopman simulation smoother. Reflecting an uncertain future, the forecasts at various quantiles are forecast intervals that widen as the forecast horizon increases.

Input Parameters

1. `X`: A matrix of observable variables containing historical data of dimension $(T \times N)$, wherein T is the number of time series observations, and N is the number of variables in the model.
2. `p`: The number of lags in the VAR model.

3. `beta`: The VAR model's coefficient matrix of dimension $(Np + 1) \times N$, where the first Np rows are lagged coefficients and the last row represents the constant.
4. `Su`: The model's variance-covariance matrix of dimension $(N \times N)$.
5. `nDraws`: The number of draws to generate using the Durban-Koopman smoother. If `nDraws = 0`, the function performs a simple Kalman smoothing; otherwise, it performs simulation smoothing with `nDraws` samples.

Quantile Plots

Designing line charts with quantile bands to visualize forecast intervals and the historical data, the `quantile_plot` function plots the forecasts, sandwiched between the forecast bands. These bands represent different percentiles, where the inner band is the 25th to 75th percentile, and the outer band is the 5th to 95th band.

Input Parameters:

1. `time`: It denotes a sequence of dates on the x-axis.
2. `quantiles`: This is a matrix of quantile values to be plotted. It can either have 5 or 7 columns:
   - For a 5-column matrix, the columns represent the outer lower quantile, inner lower quantile, center (median or mean), inner upper quantile, and outer upper quantile.
   - For a 7-column matrix, the additional two columns represent middle-lower and middle-upper quantiles.
3. `base_color`: An RGB tuple that defines the base color for the quantile bands. If not provided, a default blue color is used.
4. `run_scenario_analysis`: If `True`, the plot will use a red color scheme to indicate a scenario analysis, which typically highlights a special case or alternative scenario. Otherwise, it uses a blue color scheme.
5. `show_plot`: If `True`, the function will display the plot immediately after creating it.

4. Monte Carlo Simulations

Through a simple simulation exercise in *MCMC Simulations* file, I evaluate and test the performance of the BVAR with a constant volatility model to assess how accurately the `bvarGLP` function estimates the parameters. So, we compare the true parameters of the simulated model with the estimated parameters drawn from the distribution and examine how far apart they are. For this, I randomly draw 500-time series of observations from a VAR(1) process:

$$Y_t = c + AY_{t-1} + \epsilon_t, \quad \epsilon_t \sim N(0, \Sigma)$$

Simulating randomly generated data from the aforementioned model, I run the BVAR model and plot the posterior distribution of each of the estimated parameters in Figure 1: β_{ij}, $\forall i = 1, 2, 3$; $j = 1, 2$, which are the coefficients of the matrix A and the constant vector of the model; σ_{ij}, $\forall i = 1, 2$, which are the variances and covariance terms of the errors in Σ; and λ, governing how the coefficients shrink in estimation.

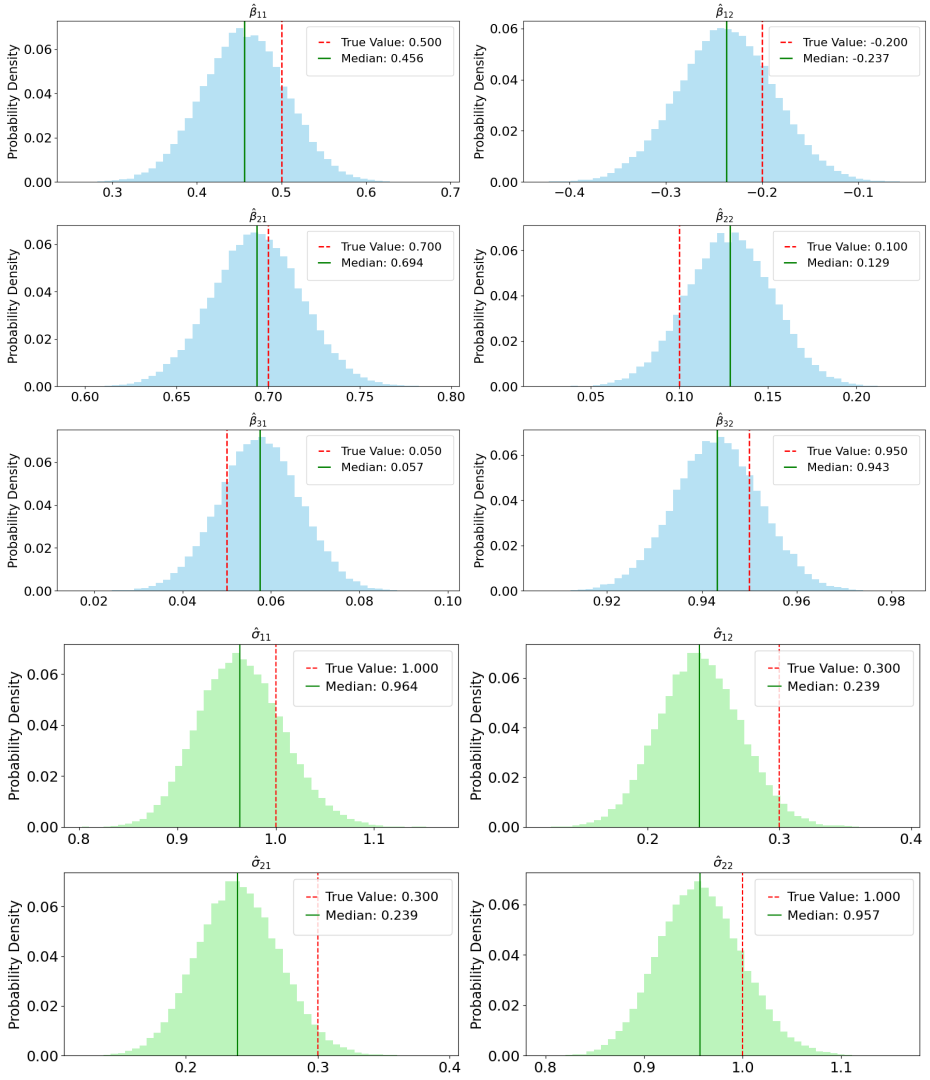

Figure 1: Posterior distribution of the coefficients of matrix A, and the variance and covariance terms of Σ from MCMC simulation with 100,000 draws where half of the draws were burnt-in, along with the true parameters.

Furthermore, Table 1 displays the posterior mean or the average of the 50,000 saved posterior samples, the 95 percent credible interval of the parameter, and the mean square error (MSE) for each parameter, which measures how close the estimates are to the true values. The true values for all the parameters lie within the 95 percent credible interval, suggesting that the posterior distribution is capturing the true value well, albeit the posterior mean may be slightly off. Likewise, in nearly all estimates, the MSE is very low, indicating that the posterior samples are generally very close to the true values.

Parameter	True Value	Posterior Mean	95% Credible Interval	MSE
β_{11}	0.5	0.4564	(0.3582, 0.5553)	0.0044
β_{12}	-0.2	-0.2372	(-0.336, -0.1387)	0.0039
β_{21}	0.7	0.6938	(0.6455, 0.7419)	0.0006
β_{22}	0.1	0.1285	(0.0802, 0.1762)	0.0014
β_{31}	0.05	0.0574	(0.0394, 0.0753)	0.0001
β_{32}	0.95	0.9433	(0.9252, 0.9613)	0.0001
σ_{11}	1	0.9652	(0.8849, 1.0538)	0.0031
σ_{12}	0.3	0.2399	(0.1799, 0.303)	0.0046
σ_{21}	0.3	0.2399	(0.1799, 0.303)	0.0046
σ_{22}	1	0.958	(0.8774, 1.0458)	0.0036
λ	0.2	0.1971	(0.1971, 0.1971)	0

Table 1: Summary of Posterior distributions of all parameters, 95 percent credible intervals, and mean square error as a function of each MCMC sample of the parameter and the true value of the parameter.

Finally, I create trace plots to visualize if the MCMC chains have converged to the true posterior distribution. The trace plots of all the parameters depict stationary patterns as the values fluctuate around the mean without trending over time, implying that the chain has converged. In the next three sectionss, I explain how we can apply this model using time series data of monthly and quarterly macro and financial variables, referencing from the Google Colab notebooks.

5. Large BVAR Model with Covid-Volatility: Applied Examples using Monthly Data Data

First, I downloaded the monthly data of 28 variables with four macro and financial variables each from January 1962 to October 2024 from FRED-MD. The macro variables are the measures of the labor market such as the unemployment rate, output such as industrial production, housing starts, and prices measured via CPI. Financial variables such as the 10-year Treasury yields, yield, spreads, and the S&P 500 index. Keeping the rate, such as the unemployment rate and borrowing rates, in levels or percentage points, I transformed the remaining variables, such as CPI, and VIX into logged values, specifically, $100 \times$ log of the levels, and we interpret the transformed variables as year-over-year growth rates.

The Google Colab notebook named as *Descriptives.ipynb* procures this data, and transforms every column identified as "log" to $100 \times$ log using the `transform_data` function to interpret them as percentage growth. Taking logs also helps in stabilizing the mean and variance of these non-stationary time series by compressing the spread of the data. This is particularly crucial when variables such as asset prices, volatility indexes, exhibit exponential growth rates, Without transforming them first will yield spurious correlations in VAR, and explode the forecasts. Lastly, I plot the time series of the original variable, and their transformation to detect the evolution of the variables and discern patterns or breaks (if any) in the data. The complete list of variables is in table A1 of the Appendix.

Applications of the Model

This section presents an example from *main* script, illustrating snippets of codes from the file. Open that file to review and run the code, and supplement it with guidelines from this section. To run the model from the *covbayesvar* package, install the package using the `pip` command and import the functions from the package's module known as `large_bvar` as:

```
!pip install covbayesvar
import covbayesvar.large_bvar as bvar
```

If we estimate the model and produce the forecasts using updated data for the first time, we set all the boolean flags to `True` so that the script runs the model and unconditional and conditional forecasts, and saves the estimates from the simulated draws in pickle files. Once we have already run the model first, if we would like to produce the conditional forecasts based on a different set of shocks, we can set each of the boolean flags of estimating the BVAR `estimateBVAR` and unconditional forecasts `runUNC` to `False`.

```
# Configuration settings
vis = True  # Set to False to hide figures
estimateBVAR = True
runUNC = True
plot_uncond_forecasts = True
plot_joint_uncond_forecasts = True
plot_scenario_analyses = True
runCF = True
lags = 12  # Number of lags in VAR, if monthly data
Ndraws = 40000  # Number of draws in MCMC simulation
discard = 20000  # Number of initial draws to discard (burn-in period)
vint = datetime(2024, 11, 1)  # The vintage date (forecasting start date)
```

Incorporating the COVID-related specifications, we determine the index when COVID began in March 2020, and the data from that date as a structural break, to ensure that the model accounts for the unprecedented volatility.

```
# COVID specific settings
if covid:
    # Index of start date of estimation
    T0 = 0
    # Index of end date of estimation (February 2020)
    TFeb2020 = np.where((dates.dt.year == 2020) & (dates.dt.month == 2))[0][0]
    # First time period of COVID (March 2020)
    Tcovid = TFeb2020 - T0 + 1
```

I establish the prior values in `prior_params` dictionary, which are the hyperparameters or the starting points of the parameters based on prior beliefs. These are the most likely value of each hyperparameter, measures of uncertainty

or degree of spread around each parameter's mode, and the lower and upper bounds within which the estimated parameters must lie during Gibbs sampling.

- **`lambda_mode` (default: 0.2)**: Mode of the "tightness" parameter of the Minnesota prior, which controls the scale of all the variances and covariances. Lower values tighten the prior, reducing the influence of the data on the model parameters to prevent overfitting.
- **`miu_mode` (default: 1)**: Mode for the persistence (mean reversion) hyperparameter. It reflects the prior beliefs on how long the shocks persist or how slowly the shocks dissipate over time.
- **`theta_mode` (default: 1)**: Mode for the cross-variable shrinkage parameter. This parameter controls how strongly we expect the variables to interact in the model, i.e., if a shock to one variable significantly impacts others in the model. Collectively, `lambda_mode`, `miu_mode`, and `theta_mode` influence the spread of the inverse-Wishart prior Σ. Specifically, these hyperparameters configure the variance structure of Σ, determining how much weight is placed on the data (likelihood), and prior beliefs of the parameters to drive the mean of the posterior distribution.
- **`eta_mode` (default: [0, 0, 0, 0.8])**: Mode for the COVID-19 scaling factor's decay parameter, η_4, specifically applied to the first three months after the disease outbreak. Since the scaling factors $\eta_{t+j}, i = 0, 1, 2$ take fixed values in the first three periods and decay at a constant rate from the fourth month, the vector `[0, 0, 0, 0.8]` controls the scaling factor for the residual covariance matrix for shocks in April, May, June 2020, and afterward. $\eta_4 \sim Beta(0.2, 0.8)$.
- **`eta_sd` (default: [0, 0, 0, 0.2])**: Standard deviation for the COVID-19 scaling factor. A low standard deviation (0.2) around the scaling factor indicates that we expect relatively small deviations from the central value of the scaling factor, reflecting a strong belief in the priors for the initial COVID-19 shock scaling.
- **`eta_min` (default: [1, 1, 1, 0.005])**: Minimum bounds for each element of the scaling factor during COVID. It ensures a minimum level of impact, particularly for the fourth element (0.005), which controls the decay of the impact of shocks after the initial COVID months.
- **`eta_max` (default: [500, 500, 500, 0.995])**: Maximum bound for each element of the COVID-19 scaling factor. High bounds on the first three elements (500) allow substantial impact from shocks in the first three COVID months. The fourth element (0.995) restricts the long-term effect slightly below 1, allowing for decay in the impact of shocks over time.

```
priors_params = {
   'lambda_mode': 0.2, #  "tightness" of the Minnesota prior: controls the scale of
      variances and covariances
   'miu_mode': 1, # mean reversion hyperparameter
   'theta_mode': 1, # mode of  cross-variable shrinkage
   'lambda_sd': 0.4, # standard deviation of the Minnesota tightness prior
   'miu_sd': 1, # standard deviation of the persistence prior
   'theta_sd': 1,
   'eta_mode': [0, 0, 0, 0.8], # mode of COVID-19 scaling factor, applied to first 3 months
      of COVID-19 period
   'eta_sd': [0, 0, 0, 0.2], # standard deviation of the covid-19 scaling factor
   'lambda_min': 0.0001,
   'alpha_min': 0.1,
```

```
    'theta_min': 0.0001,
     'miu_min': 0.0001,
    'eta_min': [1, 1, 1, 0.005],
   'lambda_max': 5,
    'alpha_max': 5,
    'theta_max': 50,
    'miu_max': 50,
    'eta_max': [500, 500, 500, 0.995]

}
```

The code estimates the BVAR on the completed transformed dataset without any missing values with 12 lags (as the data is monthly) and initial parameters.

```
# Estimate on complete panel
# Find the last time period that does not have NaN values
Testim = np.nanmax(np.where(~np.isnan(data_transformed.sum(axis=1)))[0]) + 1
bvar_results = bvar.bvarGLP_covid(data_transformed[:Testim, :], lags=lags,
    priors_params=priors_params, mcmc=1,MCMCconst=1, MNpsi=1, sur=0, noc=0, Ndraws=Ndraws,
    Ndrawsdiscard=discard,hyperpriors=1, Tcovid=Tcovid)
```

The arguments of the function `bvarGLP_covid` are:

- **`lags`**: The number of lags (12, in this case, since the data is monthly).
- **`mcmc=1`**: This specifies that the BVAR estimation will use the Markov Chain Monte Carlo (MCMC) method.
- **`MCMCconst=1`**: This parameter controls a constant factor used in the MCMC algorithm.
- **`MNpsi=0`**: A hyperparameter for the Minnesota prior, often set to 0 to strongly shrink the coefficients towards 0.
- **`sur=0`**: This parameter specifies whether to use seemingly unrelated regressions (SUR). If set to 1, the model uses SUR, allowing for cross-equation error correlation.
- **`noc=0`**: A flag for including no constant in the model. If set to 0, the model includes a constant term.
- **`Ndraws and Ndrawsdiscard`**: These specify the number of total MCMC draws and the number of discarded draws (burn-in period), respectively.
- **`hyperpriors=1`**: Enables hyperpriors for regularization.

After estimating the model, I plot the distribution of the parameters across all 20,000 saved draws. Figure 2 illustrates the distribution of the standard deviation of the March shocks or the factors that scale the shocks in the BVAR model during and after COVID-19 and the volatility decay parameter η_4. The posterior distribution of the standard deviation of the Minnesota prior. Since the standard deviation of the Minnesota prior is tightly distributed, the coefficients in Φ are shrunk towards 0, yielding more precise estimates. However, if the distribution appears wider, the uncertainty about the true parameters in Φ rises.

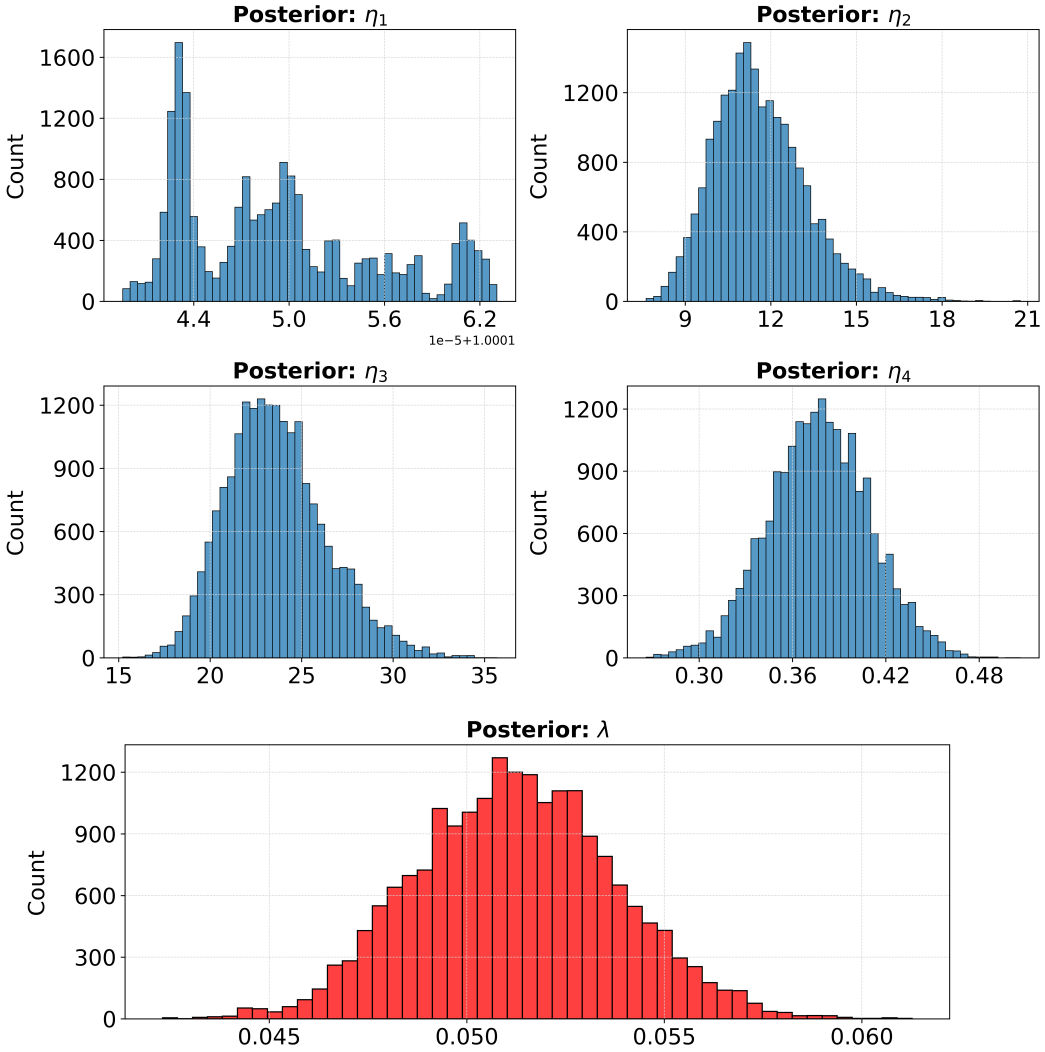

Figure 2. Posterior distribution of the Minnesota prior's standard deviation and the volatility scaling factors and the volatility decay parameter.

After estimating the parameters, I construct the unconditional forecasts for a forecast horizon of 36 months of all twenty-eight variables starting from October 2024. To accomplish that, I extract the saved parameter estimates from each of the $j = 1, \ldots, 20000$ draws from their posterior distribution. Then, I call the `VARcf_DKcks` function that generates the unconditional forecasts for j-th draw, which accumulates the forecast results for each draw across variables and months ahead in a 3D matrix called `PredY_unc`. In other words, the 3D array stores the forecasts at the h periods for n variables across all J draws.

```
for j in range(ndraws):  # Loop through the number of draws
    if (j % 10) == 0 or j == ndraws - 1:
        print(f"Generating unconditional forecasts: {j} of {ndraws} draws...")
        sys.stdout.flush()

    # Extract the j-th draw for beta and sigma
    beta_j = bvar_results['mcmc']['beta'][:, :, j]
    Gamma_j = np.vstack((beta_j[1:, :], beta_j[0, :]))
```

```
    Su_j = bvar_results['mcmc']['sigma'][:, :, j]

    PredY_unc[:, :, j] = bvar.VARcf_DKcks(YFore, bvar_results['lags']['lags'], Gamma_j,
        Su_j, 1)
```

I plot the unconditional forecasts in various quantiles: $[0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8]$ along with the historical data. For instance, figure 3 portrays the historical data from January 2020 to October 2024, followed by the model-based "baseline" probabilistic forecasts in varying shades of blue. The dark blue line denotes the most likely "median" forecast or the forecast in the middle of the distribution. Half of the forecasted values lie above the forecast denoted by dark blue, and the other half lies below this point. Likewise, the forecasts in the quantiles 0.2 and 0.8 connote that 20 percent of the forecasts fall below this point, and 80 percent are above it, implying that these forecasts are closer to the tails of the distribution. Whilst figure 3 shows the forecasts only for four variables, the code generates similar graphs for all the variables and saves them individually as PNG files.

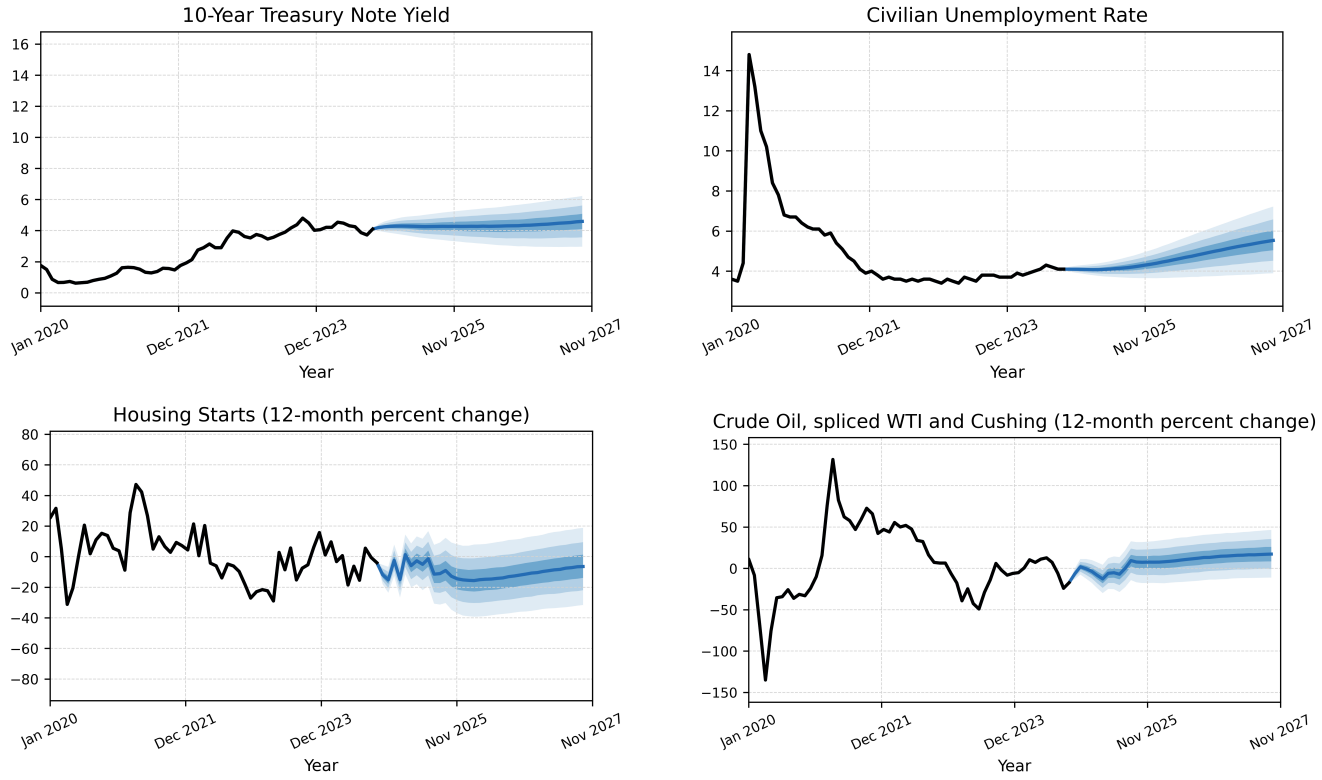

Figure 3. Actual values of the industrial production growth, unemployment rate, housing starts and oil prices (in black) till October 2024. Median forecasts for three years (in dark blue) and shaded regions represent forecast bands at 60, 70, and 80 percent coverage intervals.

Scenario Analysis

Further exploiting the versatile capabilities of the large BVAR model, we can employ the model to answer more policy-based questions, such as the impact of supply and demand shocks on predicted paths of key economic variables. I model a "forward-looking" scenario where tighter immigration rules such as mass deportations of undocumented

workers and border controls shrink the labor force, constricting the supply of workers in the economy. Fewer workers can reduce the production of goods in the short run, especially those employed in construction, hospitality, and food services, weakening demand. This supply shock puts an upward pressure on wages as employees may compete more aggressively for a smaller pool of workers in the short run. To model this simple scenario, I assume that the total employees on non-farm payroll reduce by 5 percent 5-10 months into the future and the average hourly earnings of production and non-supervisory workers rise by 2 percent after labor supply shock hits the economy 7-10 months ahead.

Coupled with the supply shocks, the PCE inflation declines by 1 percent point in the short run: 3-5 months ahead. Leaving other variables unconstrained, figure 3 depicts the scenario analyses for different segments of the economy, with posterior median response surrounded by the 68 and 90 percent coverage intervals. Here, the conditional forecasts assume that the fixed paths of all employees in the nonfarm payroll and average hourly earnings may possess information about how other variables could react. These shocks in non-farm payroll employees and average hourly earnings define a scenario:

$$S_t = \{\alpha_{L_s}\epsilon_{l,T+s} + \alpha_{m_s}\epsilon_{m,T+s}, s = 1, 2, ...\},$$

where α_{L_s} and α_{m_s} are the coefficients associated with non-farm payroll employees, and average hourly earnings, respectively.

Similar to the unconditional predictive density, the conditional predictive densities additionally depend on a conditioning set C_t that contains scenarios or realized future paths of a set of variables:

$$f(y_{T+1}, ..., y_{T+h}|y_{1:T}, C_t) = \int f(y_{T+1}, ..., y_{T+h}|y_{1:T}, \theta, C_T) f(\theta|Y_{1:T}) d\theta$$

Comparable to simulating from the unconditional predictive density, here we also iterate in two steps. First, we draw the coefficients $\theta^{(m)}$ at the m-th draw from their posterior distribution conditional on the scenario. Then, given the posterior estimate, we sample from $f(y_{T+1}, ..., y_{T+h}|y_{1:T}, \theta^{(m)}, C_T)$ using Kalman filter. Crump et. al (2021) apply the Kalman filter and simulation smoother derived in Banbura, Giannone and Lenza (2015) to make the model computationally feasible with a large dimension system.

In terms of code, the python function `VARcf_DKcks` generates and stores the conditional forecasts in a 3D matrix, just as the same function geneated the unconditional forecasts earlier. First, we define the indices of the variables on which we impose the shocks, then initialize a matrix of `NaN` values which will store the conditional forecasts. Next, we set the shocks for specified periods on the variables. Looping through the MCMC draws from the posterior distribution, we add the magnitude of the shock values to the unconditional forecasts and use the posterior estimate of the parameter from each draw to generate the conditional forecasts at each draw.

```
# Find indices of specific variables: All employees, total non farm
idxCV1 = Spec.index[Spec['SeriesID'] == 'PAYEMS'].tolist()[0]
# Average Hourly Earnings of Employees
idxCV2 = Spec.index[Spec['SeriesID'] == 'CES0600000008'].tolist()[0]
# Create a matrix of NaNs to store shocks
# n is the number of variables and T is the length of the initial data
Ycond = np.nan * np.ones((len(DateAll), n))

```

```
# Fill the initial part of Ycond with transformed data
Ycond[:T, :] = data_transformed
# Create a matrix for shocks
Shock = np.nan * np.ones((h_fore.sum(), n))  # h_fore is a boolean array indicating
    forecasts
Shock[5:10, idxCV1] = -5 # Apply shocks to % change in All Employees
Shock[7:10, idxCV2] = 2 # Apply shocks to % change in Average Hourly Earnings

# Initialize PredY_con for conditional density forecasts
PredY_con = np.nan * np.ones((len(DateAll), data_transformed.shape[1], ndraws))
for j in range(ndraws):  # Loop through the number of draws

    if (j % 10) == 0:
        print(f"Generating conditional forecasts: {j} of {ndraws} draws...")
        sys.stdout.flush()
    # Extract the j-th draw for beta and sigma
    Ycond[h_fore_1d, :] = PredY_unc[h_fore_1d, :, j] + Shock
    beta_j = bvar_results['mcmc']['beta'][:, :, j]
    Gamma = np.vstack((beta_j[1:, :], beta_j[0, :]))
    Su = bvar_results['mcmc']['sigma'][:, :, j]
    PredY_con[:, :, j] = bvar.VARcf_DKcks(Ycond, bvar_results['lags']['lags'], Gamma, Su, 0)


# Compute the difference between conditional and unconditional forecasts
dY = PredY_con - PredY_unc
```

The top graphs in Figure 4 a) and b) show the trajectory of all employees in the non-farm payroll and industrial production index after the former slumps by 5 percent points, respectively, from its benchmark forecasted paths. In contrast, the bottom graphs showcase the scenario, which is the difference between the forecasts conditional on the shocks on both variables, and the unconditional or baseline forecasts, which are devoid of any shocks. Mathematically,

$$D[y_{T+s}|y_{1:T}, C_T, \theta] = E[y_{T+s}|y_{1:T}, C_T, \theta] - E[y_{T+s}|Y_{1:T}, \theta]$$

$$\implies \int y_{T+s} f\left(y_{T+s}|y_{1:T}, \theta, C_T\right) dy_{T+s} - \int y_{T+s} f\left(y_{T+s}|y_{1:T}, \theta\right) dy_{T+s}$$

These forecasts are devoid of any structural identification schemes for the errors and are entirely based on past historical correlations among the variables in the system. Because the scenarios are derived from the reduced-form models, without identifying the structural shocks, I do not interpret them. In the Google Colab notebook, I use the `quantile_plot` function to generate these (2×1) subplots for all 28 variables and save each of them as PNG files.

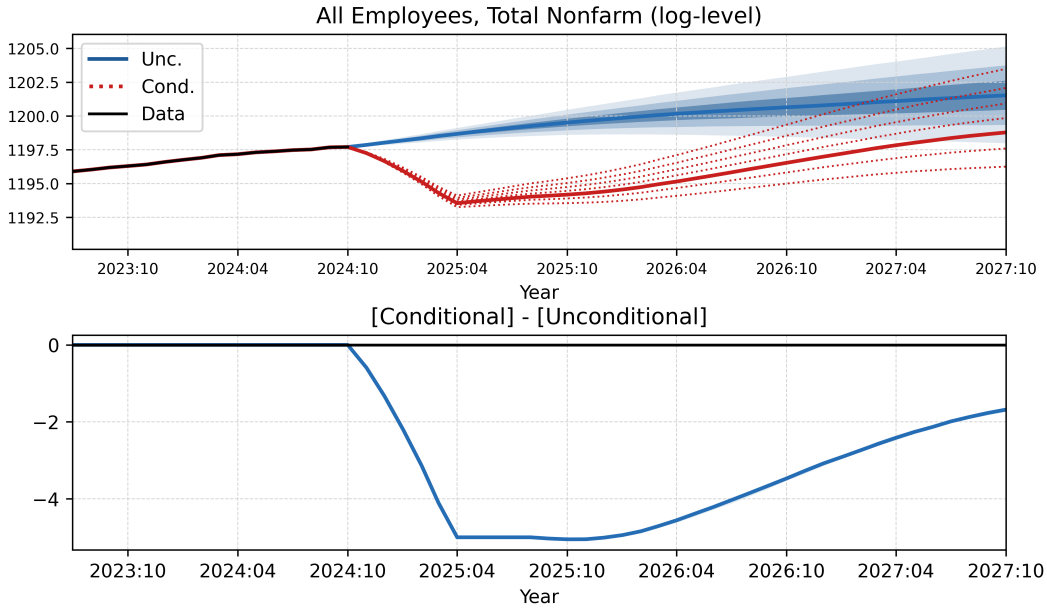

Figure 4a. Conditional forecasts (in red) and unconditional forecasts (in blue) of log-transformed data on all employees in non-farm payroll where the shaded areas are the 60, 70 and 80 percent forecast intervals. Lighter-shaded regions are forecasted paths that are less likely to occur. The forecast bands or credible intervals denoted by red dots result from the conditions that the number of employees in non-farm payroll falls by 5 percent, and wages rise by 2 percent.

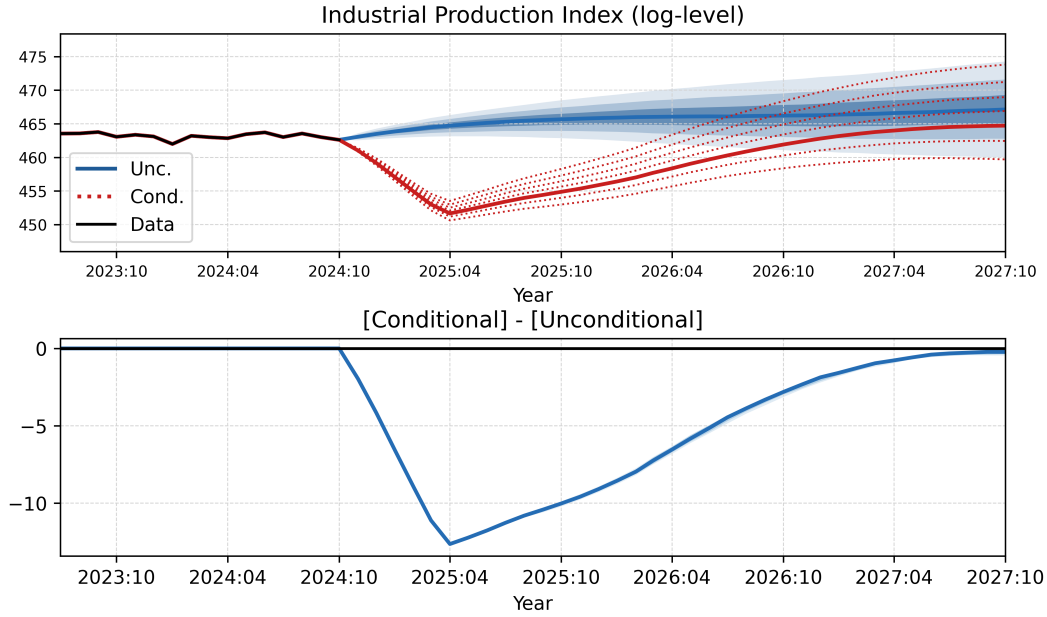

Figure 4b. Conditional forecasts (in red) and unconditional forecasts (in blue) of the log-transformed Industrial Production Index.

In the same spirit, the difference between conditional and unconditional forecasts is equivalent to the generalized impulse response functions to shocks in non-farm payroll employees and average hourly earnings. Formally, the impulse response function is the difference between the counterfactual and the baseline as:

$$\frac{\partial y_{i,T+h}}{\partial v_{j,T}} = \mathbb{E}\left[y_{i,T+h}|y_{1:T-1}, v_{j,T} = 1, \gamma\right] - \mathbb{E}\left[y_{i,T+h}|y_{1:T-1}, \gamma\right]$$

where, $v_{j,T}$ is the structural shock which is a linear combination of forecast errors, $v_{j,T} = \alpha' e_T$. In the above example, it is the linear combination of the forecast errors on non-farm payroll employees and average hourly earnings. Figure 5 displays the elasticities or responsiveness of all the variables to the persistent negative shocks on employees in the non-farm payroll and increasing hourly earnings.

Notably, we see that the measures of economic activity - capacity utilization, real manufacturing, and trade industrial sales decline by 10 percent from their baseline paths. Because of the relatively sudden shocks, firms cannot instantly replace or automate away the lost labor, slowing down output in the housing, construction, and manufacturing sectors, and spilling over into overall industrial production. The contracting economy dwindles the supply of goods and services, which also reduces the jobs available as the demand falls.

This spikes the unemployment rate as witnessed in the graph in the short run, immediately after the shock. A rise in the unemployment rate by 2.5 percent owing to lower output dips inflation if the negative output effect is larger than the cost-push effect from higher wages. But in the long run, if employers increasingly automate, offshore, or reorganize production, wage and price pressures dampen. In other words, they revert to their long-run means as evident in the scenario analysis in Figure 5.

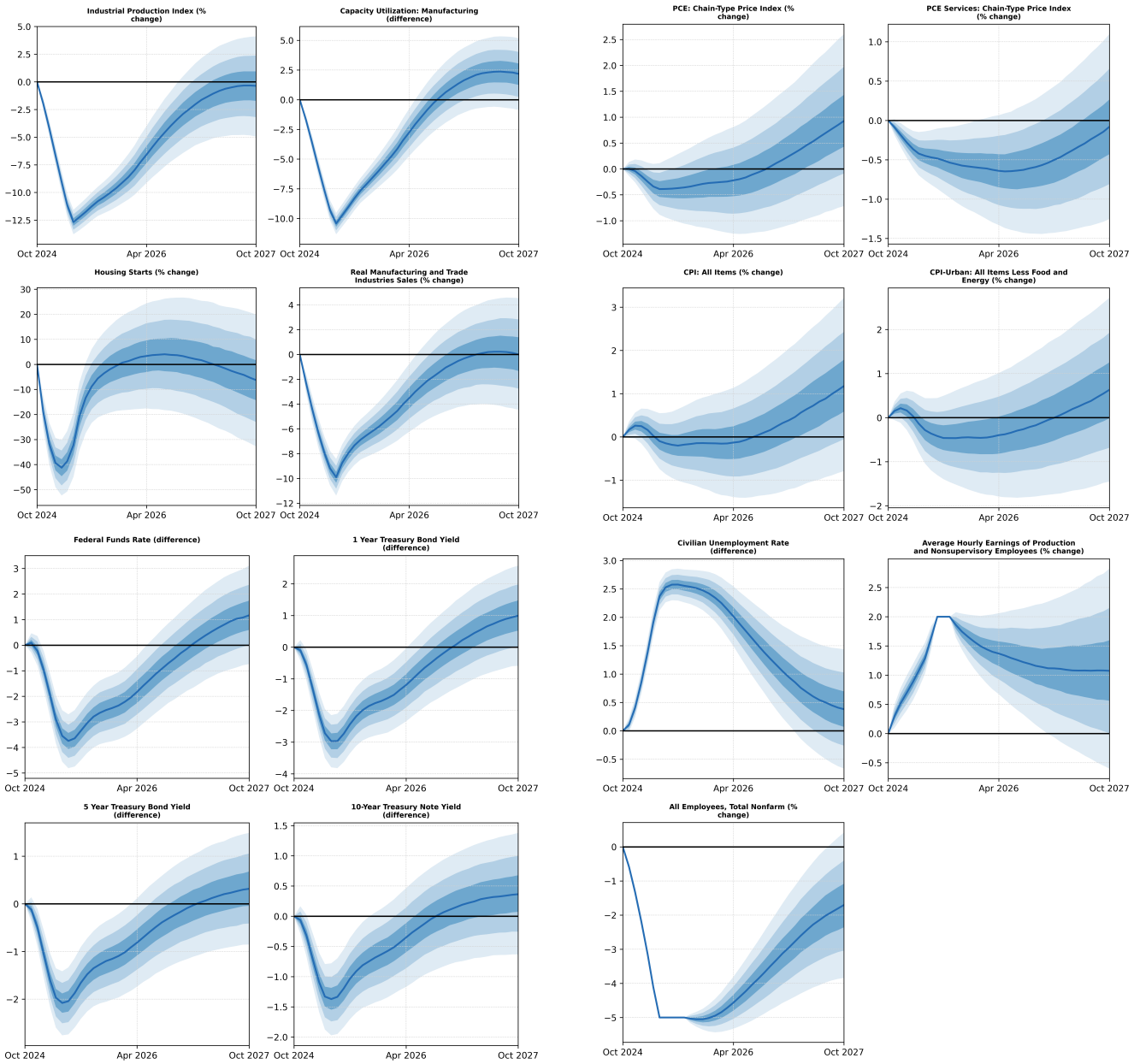

Figure 5. Response of all the variables when non-farm payroll employees persistently contract by 5 percent and average hourly earnings rise by 2 percent. The blue line connotes the median response, and the shaded regions indicate the 60, 70, and 80 percentile coverage intervals.

Joint Forecast Density: Unveiling Macro Risks, and Relationships

Until now, I had emphasized estimating the baseline and conditional forecasts of individual variables and computing the univariate densities of the forecasts. While this is useful to understand the expected path of the macro variables, we can also depict the entire plausible set of outcomes and demonstrate how two variables might co-move. For instance, with what probability does a high unemployment rate trigger a lower federal funds rate, or can both be high simultaneously? How does the predicted unemployment vary with monetary regime? How will the joint forecasts alter if the Fed targets lower inflation or unemployment? Are there potential trade-offs or correlations in

the forecast period, and how uncertain are the forecasts?

To answer these kinds of questions, we can construct joint forecast density plots of any forecast horizon between two variables as exemplified in figure 6. The vertical dimension tells us the estimated joint probability of observing two macro variables one year ahead in October 2025, using the historical data till October 2024. Every point in the 2D plain is a possible combination, and the height denotes the likelihood of the combination from the predictive density. The grey dots, representing the cloud of points, are the posterior draws from the BVAR's predictive distribution. The side panes denote the marginal density plots, indicating the marginal distribution of forecasts. Showing a likelihood of various combinations, it helps to form internally consistent narratives under different scenarios. The tallest part of the surface is around the center of the dot cloud, implying the highest joint density or the "most typical" scenario occurs where the Fed Funds rate is around 4–5 percent and unemployment around 3.5–4 percent. A single peak in the graph connotes that the joint forecast distribution is unimodal. Predominantly, this occurs when the federal funds rate is approximately 5 percent. The joint density contours are concentrated in specific areas, suggesting a slight negative correlation between unemployment rate and the federal funds rate. Namely, lower unemployment (3–5%) is likely to coincide with either moderate (~5%) or significantly elevated (~12%) federal funds rate.

As dots heavily cluster when the unemployment rate is around 3–5 percent, the model expects a resilient labor market even in scenarios with aggressive monetary transmission (of ~12%). The latter can occur when inflation is out of control, raising the federal funds rate in response, but the economy is strong and lands softly. However, the right tail indicates that there are risks of elevated unemployment (~6–8%) in adverse economic conditions, albeit these are less likely. Another less likely scenario is stagflation, which marks periods of high unemployment and inflation mired with stagnant growth in output, triggering the Fed to reign in on spiralling prices by hiking the policy rates. This is because high unemployment clusters with lower federal funds rate in the left graph. The spread of the density contours signal uncertainty in forecasts where broader spread along the federal funds rate axis connotes higher uncertainty about the future path of the interest rates relative to those of unemployment rate.

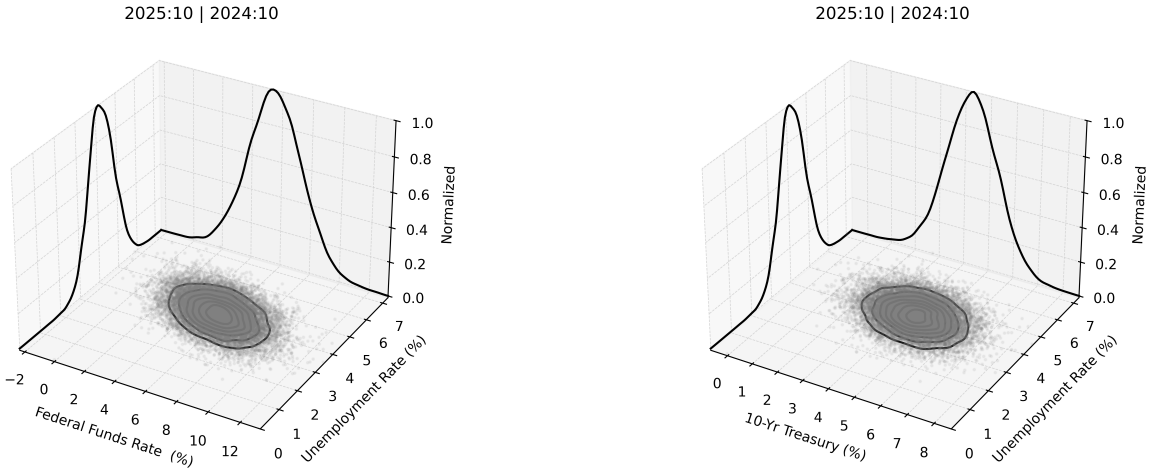

Figure 6. Joint forecasts of the variables for October 2025 using the historical data till October 2024. The grey cloud of the points in the center and the lightly visible contour line are the posterior predictive densities. The marginal densities of the forecasts are in the side panels.

Entropic Tilting: Softly Anchoring Forecasts to Long-Run Targets

Previously, I created baseline forecasts of all the variables without considering any targets, or long-run steady-state rates of variables set by the Federal Reserve. The BVAR model generates forecasts using MCMC simulations, assigning equal weights to each draw in the forecast distribution. Sometimes, central banks explicitly target variables such as inflation and unemployment rates, assuming that the inflation rate will converge to 2 percent in the long run. For instance, table 2 presents the central tendency of long-run projections of the FOMC released in the Summary of Economic Projections (SEP) in December 2024, capturing the middle-ground belief of policymakers regarding the long-run path of these variables. How can we adjust the forecasts to reflect these beliefs? The goal is to tilt the baseline distribution to reflect these beliefs while remaining as close to the original distribution. In this exercise, I employ the method of entropic tilting elucidated in Tallman and Zaman (2020) to modify the baseline distribution and refer to the *Entropic_Tilting* file in Google Colab and walk through the implementation procedure mathematically and programmatically. Unlike the scenario analyses above where I fixed the future paths of certain variables, also known as hard conditioning; I now relax that by allowing the conditioned future values of certain variables to lie within a certain range, also known as soft conditioning. Because we generally don't know the realized paths of endogenous variables in the long horizon, constraining the conditioned path of the variable within a certain range or an interval instead of an exact path is simpler. Furthermore, soft constraints recognize the uncertainty around the paths observed in the future.

Variable	**Projection for 2026 (%)**	**Projection for 2027 (%)**	**Long run projection (%)**
Unemployment rate	4.1 – 4.4	4 – 4.4	3.9 – 4.3
Real GDP	1.9 – 2.1	1.8 – 2	1.7 – 2
PCE inflation rate	2 – 2.2	2	2
Core PCE inflation rate	2 – 2.3	2	2
Federal funds rate	3.1 – 3.6	2.9 – 3.6	2.8 – 3.6

Table 2: Summary of Economic Projections released on December 18, 2024. These projections are central tendencies for change in the unemployment rate, real GDP, core PCE, PCE inflation rate, and the projected path of the federal funds rate. The central tendencies exclude the outlier projections (the three highest and lowest forecasts) and reflect FOMC's consensus view.

Let $y \in \mathbb{R}^n$ represent a vector of forecasts for n variables. If we run J draws from the MCMC simulations, then $y_j \; \forall \; j = 1, 2, \ldots, J$ is a forecast at the j^{th} draw. Initially, we assign equal weight to each draw, $w_j = \frac{1}{J}$. Now, we re-weight the baseline distribution $f(y)$ to a tilted distribution $\hat{f}(y)$ such that two conditions are met. Firstly, the reweighted distribution must satisfy specific moment conditions. Secondly, the tilted distribution must be as close to the original distribution.

The moment conditions are

$$E_{\hat{f}}[g(y)] = \bar{g},$$

where $g(y)$ is a set of m moment conditions, and $\bar{g} \in \mathbb{R}^m$ contains the target moments, which are the central tendency forecasts.

To determine the tilted distribution, we minimize the Kullback Leibler (KL) divergence between $f(y)$ and $\hat{f}(y)$:

$$\min_{w_j^*} KL(\hat{f}, f) = \sum_{j=1}^{J} w_j^* \log\left(J w_j^*\right) \text{ s.t. } \sum_{j=1}^{J} w_j^* = 1, \; \frac{1}{J}\sum_{j=1}^{J} w_j^* g(y_j) = \bar{g}.$$

The solution to find the optimal weights is

$$w_j^* = \frac{\exp(\gamma^\top g(y_j))}{\sum_{j=1}^{J} \exp(\gamma^\top g(y_j))}.$$

Here, γ is the Lagrange multiplier associated with the moment condition, and $g(y_j)$ is the moment function evaluated at j^{th} draw. We normalize so that the weights across all draws sum to 1. Before we can find the optimal weights, we optimize the Lagrange multiplier as

$$\gamma = \arg\min_{\gamma} \sum_{j=1}^{J} \exp(\gamma^\top g(y_j) - \gamma^\top \bar{g}).$$

Finally, the re-weighted (tilted) moments should match the target values $\bar{g}$ within a small tolerance:

$$E_{\hat{f}}[g(y)] = \frac{1}{J} \sum_{j=1}^{J} w_j^* g(y_j)$$

The code in *Entropic_Tilting* file implements "soft conditioning" by adjusting the baseline forecasts of key variables to reflect long-run projections from FOMC's SEP.

Below is the average of the range of the central tendency values.

```
# Conditioning assumptions: SEP released on Dec 2024 SEP for 2027
# Center of the central tendency: midpoint of the range
# find the index of the PCE inflation rate, Federal Funds Rate, and Unemployment Rate in
    the Spec DataFrame
idxCVPCE = Spec[Spec['SeriesName'] == 'PCE: Chain-Type Price Index'].index[0]
valCVPCE = (2 + 2) / 2

# Federal Funds Rate
idxCVFFR = Spec[Spec['SeriesName'] == 'Federal Funds Rate'].index[0]
valCVFFR = (2.9 + 3.6) / 2

# Unemployment Rate
idxCVLR = Spec[Spec['SeriesName'] == 'Civilian Unemployment Rate'].index[0]
valCVLR = (4 + 4.4) / 2
```

Using monthly data, I don't consider the average of the central tendency values of real GDP as that is available only in quarterly frequency, but I present another exercise using quarterly data in Appendix A.3 where I condition on real GDP, unemployment rate, PCE and core PCE inflation. Next, we procure the BVAR estimates `bvar_results` and distribution of unconditional forecasts `PredY_unc` computed and saved in pickle files in the *main* file. Then, we tilt the original forecast distribution to anchor the mean of the distribution to the midpoint of the central tendencies of SEP projections.

```
############################ Entropic Tilting: Shifting the mean
    ############################
```

```

n = PredY_unc.shape[1]  # Number of series (second dimension)
# 12-month change
dPredY_unc = np.vstack((
    np.full((12, n, ndraws), np.nan),  # Add NaNs at the beginning
    PredY_unc[12:, :, :] - PredY_unc[:-12, :, :]
))

# MCMC Draws for tilting variables (PCE inflation, unemployment rate, and federal funds rate) at forecast horizon
YYh = np.vstack((
    dPredY_unc[-1, idxCVPCE , :].T,  # Extract indices of federal funds rate and PCE inflation rate
    PredY_unc[-1, [idxCVLR, idxCVFFR], :]                  # Extract unemployment rate
)).T
# Target values: FOMC's centeral tendency projections for the targeted variables
target = np.array([valCVPCE, valCVLR, valCVFFR])

# Objective function to find the optimal lambda (Lagrange multiplier) values
def fun(gamma, YY, g0):
    return np.sum(np.exp((YY - g0) @ gamma))

# Optimization setup
opts = {'tol': 1e-20, 'options': {'maxiter': 1000}}
objective = lambda x: fun(x, YYh, target) # Objective function for optimization
gamma_init = np.ones(len(target)) # Initial guess for gamma
# Uses a quasi-Newton method for constrained optimization:
# minimizes the divergence between the original and tilted distribution while ensuring the mean of the tilted
# distribution satisfies the target constraints
res = minimize(objective, gamma_init, method='L-BFGS-B', **opts)
# optimal gamma values that adjust the weights of the forecast draws to align the distribution with the targets
gammaStarMean = res.x

# minimized objective function value (Kullback-Leibler divergence between the original and tilted distributions)
fStar = res.fun
# re-calculate weights for the forecast draws using the optimal gamma values
# normalize the weights to sum to 1
wStarMean = np.exp(YYh @ gammaStarMean) / np.sum(np.exp(YYh @ gammaStarMean))

# Verify moment condition
# check the mean of the re-weighted draws to ensure it aligns with the target values
print("Moment:")
# weighted average of the forecasted draws that must equalize the target values
print(np.mean(wStarMean[:, None] * YYh * 10000, axis=0))  # Scale by 10000
print("Target:")
print(target)
```

A disadvantage of tilting towards the mean is that the tilted forecasted distribution will be sensitive to outliers or extreme values, distorting results if outliers are unreliable. Skewing the forecasting distribution, the mean may not accurately represent the central tendency. On the other hand, extreme values don't affect the median, and central banks often focus on median forecasts while generating counterfactual forecasts. Therefore, I present an example where I tilt the forecast distribution so that the median of the forecast distribution matches the target values, ensuring that the tilted distribution stays as close as possible to the original distribution. I achieve this by minimizing the KL divergence:

$$\min_{w_j^*} KL(\hat{f}, f) = \sum_{j=1}^{J} w_j^* \log(J w_j^*) \text{ s.t. } \sum_{j=1}^{J} w_j^* = 1, \ \frac{1}{J}\sum_{j=1}^{J} w_j^* g(y_j) = 0.5$$

Now, the moment condition changes to:

$$E_{\hat{f}}[g(y)] = 0.5, \text{ where } g(y_j) = 1\{y_j \leq \text{Target}\}$$

Therefore, 50 percent of the tilted distribution's mass should lie below the target $\bar{g}$.

```
############################## Entropic Tilting: Shifting the median ##############################
# Create the indicator matrix to check if every value in the forecast matrix is less than or equal to the target value
YYhTemp = YYh <= target
YYhTemp = YYhTemp.astype(int)  # Convert the binary (True/False) to integer (1 and 0)

# Define the optimization objective
# find the optimal gamma that minimizes the Kullback-Leibler divergence between the original and tilted distributions
objective = lambda gamma: np.sum(np.exp((YYhTemp - 0.5) @ gamma))

# Perform optimization
gamma_init =  np.ones(len(target))  # Initial guess
res = minimize(objective, gamma_init, method='L-BFGS-B', options=opts)
gammaStarMedian = res.x
fStar = res.fun

# Calculate weights for each forecast draw using the optimal gamma values
# Reweight the forecast draws (rows of YYh) using the optimized gamma values.
# Normalize the weights so they sum to 1.
wStarMedian = np.exp(YYhTemp @ gammaStarMedian) / np.sum(np.exp(YYhTemp @ gammaStarMedian))

# Check that conditioning assumptions are satisfied
# Sort the forecast values (YYh) for each variable to compute the weighted median.
temp_s = np.sort(YYh, axis=0)  # Sorted values
idx = np.argsort(YYh, axis=0)  # Indices for sorting
cumsum_w = np.cumsum(wStarMedian[idx[:, 0]])  # Cumulative weights for the first column
j = np.argmin(np.abs(cumsum_w - 0.5))  # Find the index closest to 0.5
```

```
print("Value at median:", temp_s[j])  # Value corresponding to median
print("Target:")
print(target)

print("Median:")
median_values = np.array([
    bvar.wquantile(YYh[:, 0].reshape(1,-1), 0.5, wStarMedian),  # Weighted median for
        column 1
    bvar.wquantile(YYh[:, 1].reshape(1,-1), 0.5, wStarMedian),
    bvar.wquantile(YYh[:, 2].reshape(1,-1), 0.5, wStarMedian)
])
print(median_values)
```

Figure 7 illustrates the unconditional (baseline) forecast distribution from the BVAR model without imposing any external constraints or target (in shades of blue) and the conditional forecast distribution that considers long-run targets for PCE inflation rate, unemployment rate, and projected paths of the federal funds rate (in shades of red). Thereby, I introduce judgment by anchoring the median of the BVAR forecasts to the midpoint of the central tendency forecasts for the 2027 reference period. Upon tilting to the median, the red bands shift the forecast distribution such that 50 percent of the probability mass lies below their respective targets. Additionally, the median forecasts of PCE inflation, unemployment, and federal funds rate (denoted by dark red in the center of the conditional probabilistic distribution) converge to the stated targets of 2, 4.2, and 2.25 percent, respectively. This makes the forecasts more consistent with the consensus views of the FOMC, particularly in the medium to long run, where uncertainty might be higher, widening the forecast intervals.

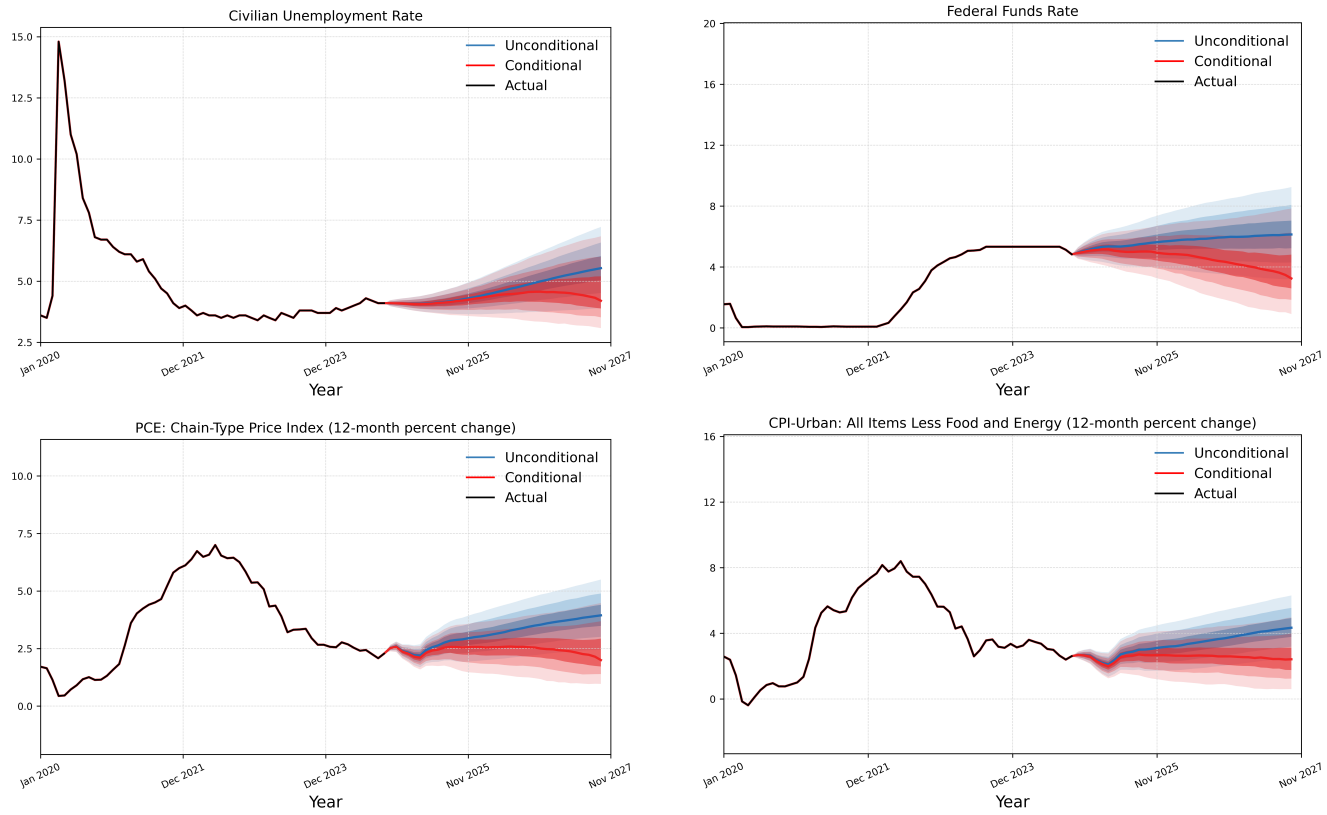

Figure 7. Unconditional forecasts and forecasts conditional on SEP projections of PCE inflation rate to 2 percent, unemployment rate to 4.2 percent, and federal funds rate to 3.25 percent 3 years into the future. These conditional forecasts are produced after tilting the forecast distribution such that the median of the forecast distribution anchors to the target SEP projections.

Figure 8 exhibits joint predictive densities of the unemployment and PCE inflation rates for 2 and 3 years into the future. The black lines are the unconditional joint predictive densities analogous to those in Figure 6. Alternatively, the red counterparts are the tilted distributions, matching the central tendencies of SEP projections for 2026 and 2027, respectively. The blue dots in the center of the red contours are the midpoint values of the central tendencies of the forecasts of a given year. These are the averages of the lower and upper range of the values in Table 2, yielding the midpoint forecast of 4.25 and 4.2 percent for the unemployment rate in 2026, and 2027, respectively. The unconditional and conditional marginal forecasts for densities for two years ahead (left) are comparable and unimodal, while those for three years ahead (right) are multi-modal.

Multiple peaks in the distribution can occur due to a few reasons. Firstly, as the forecast horizon extends further into the future, forecasts become more uncertain, amplifying the spread of the forecast distribution. Secondly, the Federal Reserve's stated long-run targets (for example, 2 percent inflation) anchor for expectations. However, the path to reaching these targets may vary significantly depending on how the Fed reacts to economic developments. To elaborate, the joint predictive forecast density for November 2027 has three peaks. One occurs when the PCE inflation rate is approximately 2 percent, and the unemployment rate is approximately 4.2 percent. This peak corresponds to the Fed's long-run target where inflation stabilizes at 2 percent and the unemployment rate at 4.2 percent. The red-tilted density heavily weights this outcome, reflecting alignment with the SEP projections. In a stable and ideal economic scenario, the variables smoothly converge to the long-run equilibrium. Another peak occurs when the PCE inflation rate is approximately 3 percent, and the unemployment rate is approximately 5.5 percent. Signifying a stagflation-like scenario where inflation remains above target and unemployment is higher, a

potential reason might be the wage-price spiral. Finally, a smaller peak occurs when inflation and unemployment rates are at approximately 1 and 3.5 percent, respectively. Whilst less likely, this optimistic outcome connotes a scenario where aggressive tightening of monetary policy can risk deflation and drive unemployment below long-run expectations.

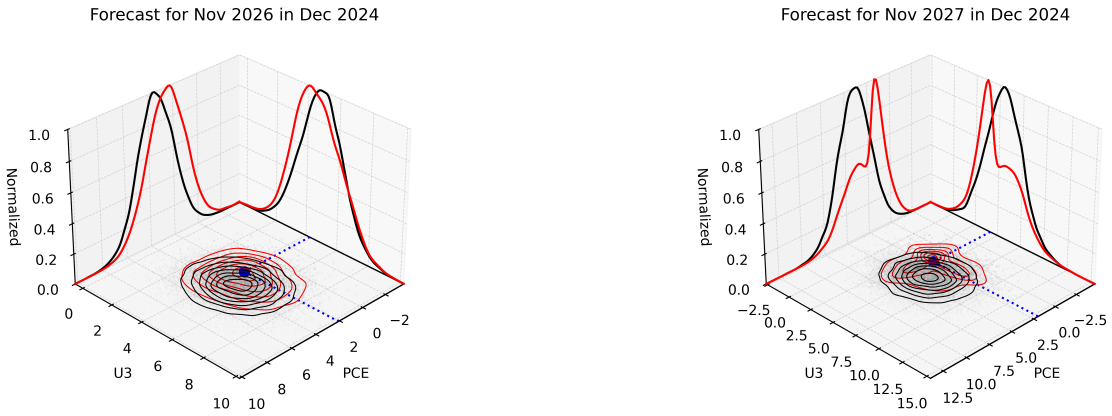

Figure 8. Joint forecast density functions of the forecast of the unemployment rate and PCE inflation rate for November 2026 (left), and November 2027 (right).

Thirdly, since entropic tilting modifies the posterior distribution by reweighting the MCMC draws to satisfy moment conditions, reweighting magnifies certain regions of the joint density (combination of unemployment and inflation rate) that aligns with the targets, and down-weights other regions. To elucidate, if the original posterior distribution already had higher density near the targets, tilting amplifies these regions, making the modes more pronounced. Moreover, if more draws from the less-dense regions are favored to satisfy the moments, tilting creates new modes. Relatedly, the scatterplot in Figure 9 visualizes the joint draws of inflation and unemployment rates for November 2027. Here, each point in the graph represents one MCMC draw from the posterior predictive distribution color-coded by the magnitude of the weights. For instance, MCMC draws (or points with darker colors) are weighted more than those with lighter colors. So, the black-colored draws contribute most to the tilted distribution as they strongly align with the moment conditions ($U3 = 4.2\%$, $PCE = 2\%$). On the flip side, the yellow-colored draws are down-weighted as they are far from the moment conditions, contributing minimally to the tilted distribution.

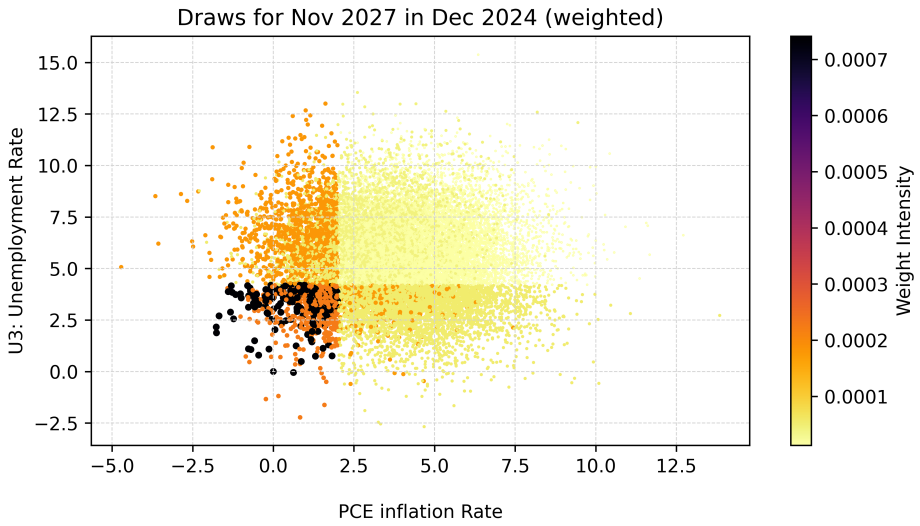

Figure 9: Joint MCMC draws of PCE inflation and unemployment rate for November 2027 in December 2024 categorized by weights. The size of each point proportionally changes with the weights optimized by entropic tilting so that the median of the tilted distribution anchors with the midpoint of the SEP's central tendencies.

6. BVAR with Covid Volatility on Quarterly Data

The BVAR model works with data of quarterly frequency as well. This is crucial because the components of the National Income Product Accounts such as Real Gross Domestic Product are measured only in quarterly frequency, enabling us to select a broader range of macroeconomic variables that organizations such as the Fed and Congressional Budget Office heavily incorporate in making forecasts. The baseline quarterly dataset contains 29 variables from FRED-QD to capture the macroeconomic and financial conditions employed in Crump et. al (2021) from Q1-1986 to Q3-2024. These include real indicators such as the real GDP, real federal government consumption; price indicators such as headline and core CPI, and PCE indices, GDP deflator; indicators on labor market, economic activity, Treasury yields and yield spreads and asset prices. The federal funds rate was near 0 during COVID from March 2020-2022 and during previous downturns, creating zero lower bound (ZLB), wherein interest rates cannot decline below in nominal terms. Since the Federal Reserve resorts to unconventional tools to stimulate the economy, the BVAR's linear equations cannot easily capture the effective "floor" at the zero. To avoid accounting for the non-linear and no variation when ZLB binds, I omit the federal funds rate in the model. Table A2 in the Appendix lists the quarterly variables and the transformations applied before estimating the model. Thereafter, I explore applications of how we can fit the model to quarterly data, construct scenario analyses, conditional and unconditional forecasts, gauge for structural breaks during the pandemic and build forecasts conditioned on long-term targets. The *Quarterly* folder saves all the scripts that plot the time series of the historical time series data in the *Descriptives* script and runs the model in other scripts.

Changes in Estimation of the Model with Quarterly Data

The codes in *main_quarterly* script show the minor adjustments made to account for the quarterly frequency of the data. Aggregating from monthly to quarterly reduces the time series by three-fold, making the model

more susceptible to overfitting as it ingests fewer data points. Therefore, in the `prior_params` dictionary, I place more weight on the prior beliefs by changing `lambda_mode = 0.2` at a monthly frequency to `lambda_mode = 0.6`, tightening the prior. Now, the posterior estimates rely more heavily on the prior. Another change is in how the scaling factors evolve in each pandemic quarter. In the monthly setup, `eta_mode` and `eta_sd` capture how the model scales the variance of the shocks over the first three months of COVID (March, April, May 2020), and then gradually decays from June 2020. Beyond the initial shock months, `eta_mode = 0.8` is the starting guess for the scale factor, while `eta_sd` array indicates the degree of uncertainty allowed around each mode. In the quarterly setup of `prior_params`:

```
'eta_mode': [0, 0.8, 0.7, 0.6], # mode of COVID-19 scaling factor, applied to first 3 quarters of COVID-19 period
'eta_sd': [0, 0.2, 0.15, 0.1], # standard deviation of the covid-19 scaling factor
'eta_min': [0.0, 0.5, 0.5, 0.3]


'lambda_mode': 0.6, #  "tightness" of the Minnesota prior: controls the scale of variances and covariances
'lambda_sd': 0.3, # standard deviation of the Minnesota tightness prior
```

This implies that the COVID shock factor in Q1-2020 is entirely data-driven with uninformative prior. Subsequently, for Q2-Q4 2020, the declining prior values suggest subsiding volatility, capturing the time-varying nature of the pandemic shock. Apart from this, I run the model as usual on the quarterly data. As the model runs on fewer observations, it is computationally efficient reducing the time taken to estimate and generate forecasts.

Beyond the Boom: Increase in Real GDP

I analyze six scenarios using conditional forecasting beginning with one-time 1 percent increase in real GDP one quarter ahead relative to the unconditional forecasts. Analogous to the residual impulse response function from the traditional reduced form VAR with unorthogonalized innovations, I don't structurally identify the shocks.

```
# Find indices of specific variables
idxCV1 = Spec.index[Spec['SeriesID'] == 'GDPC1'].tolist()[0]  # real GDP
# Create a matrix of NaNs to store shocks
# n is the number of variables and T is the length of the initial data
Shock = np.nan * np.ones((h_fore.sum(), n))  # h_fore is a boolean array indicating forecasts
Shock[0, idxCV1] = 1 # Apply shocks to % change in real GDP
```

Figure 10 demonstrates the responses of metrics pertaining to the labor market, real economic activity, prices, asset prices and interest rates. All the real macro variables such as personal consumption expenditure, investment, and income rise as GDP rises. Over time, these cascading effects taper as the the model's dynamics move the system back toward equilibrium. Measures of prices, Treasury yields, real exports and imports rise.

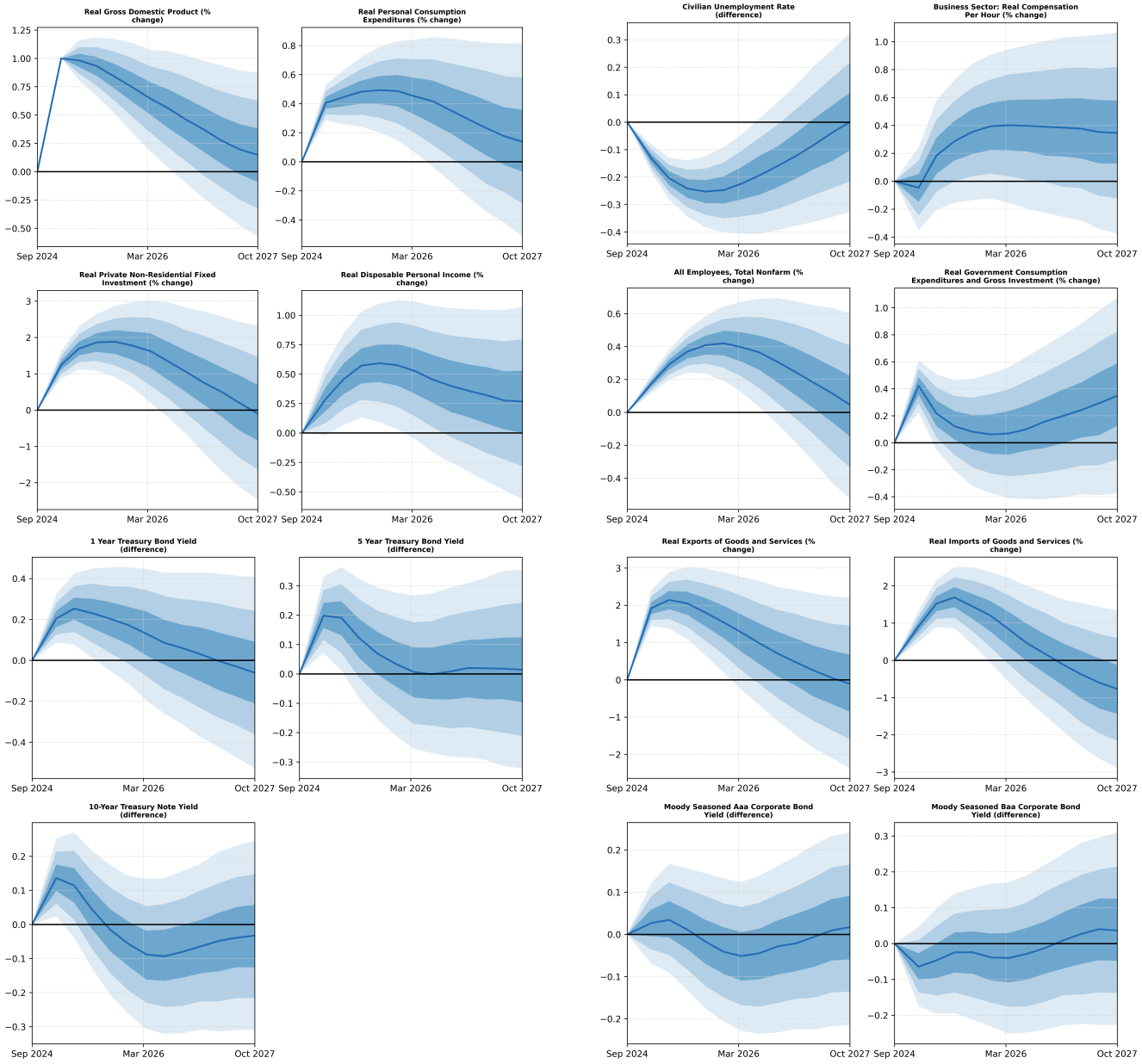

Figure 10. Response of 1 percent increase in real GDP one quarter ahead over a period of 3 years where the dark blue lines are the posterior median values.

In contrast to the short-term one time shock, figure 11 illustrates the responses of the variables when GDP increases 1 percent 8–12 quarters ahead, without affecting other variables. I model this scenario in *increaseGDP_longRun* script, by adjusting the index of shocks in the `Shock` numpy array as follows:

```
idxCV1 = Spec.index[Spec['SeriesID'] == 'GDPC1'].tolist()[0]  # real GDP
Shock = np.nan * np.ones((h_fore.sum(), n))
Shock[7:12, idxCV1] = 1 # Apply shocks to % change in real GDP
```

As Crump et. al (2021) state, this "medium-run" scenario conditions on a combination of reduced-form shocks that lifts real GDP by 1 percent in two years. In the reduced-form model, the historical relationships among variables determine the combination of shocks that produce the desired path where real GDP is elevated by 1 percent from

the baseline path. Unlike structural impulse responses that utilize Cholesky or sign restrictions to isolate each shock, I construct generalized impulse response functions (where the shocks are not orthogonal) as the difference between the conditional and unconditional forecasts presented in figure 12.

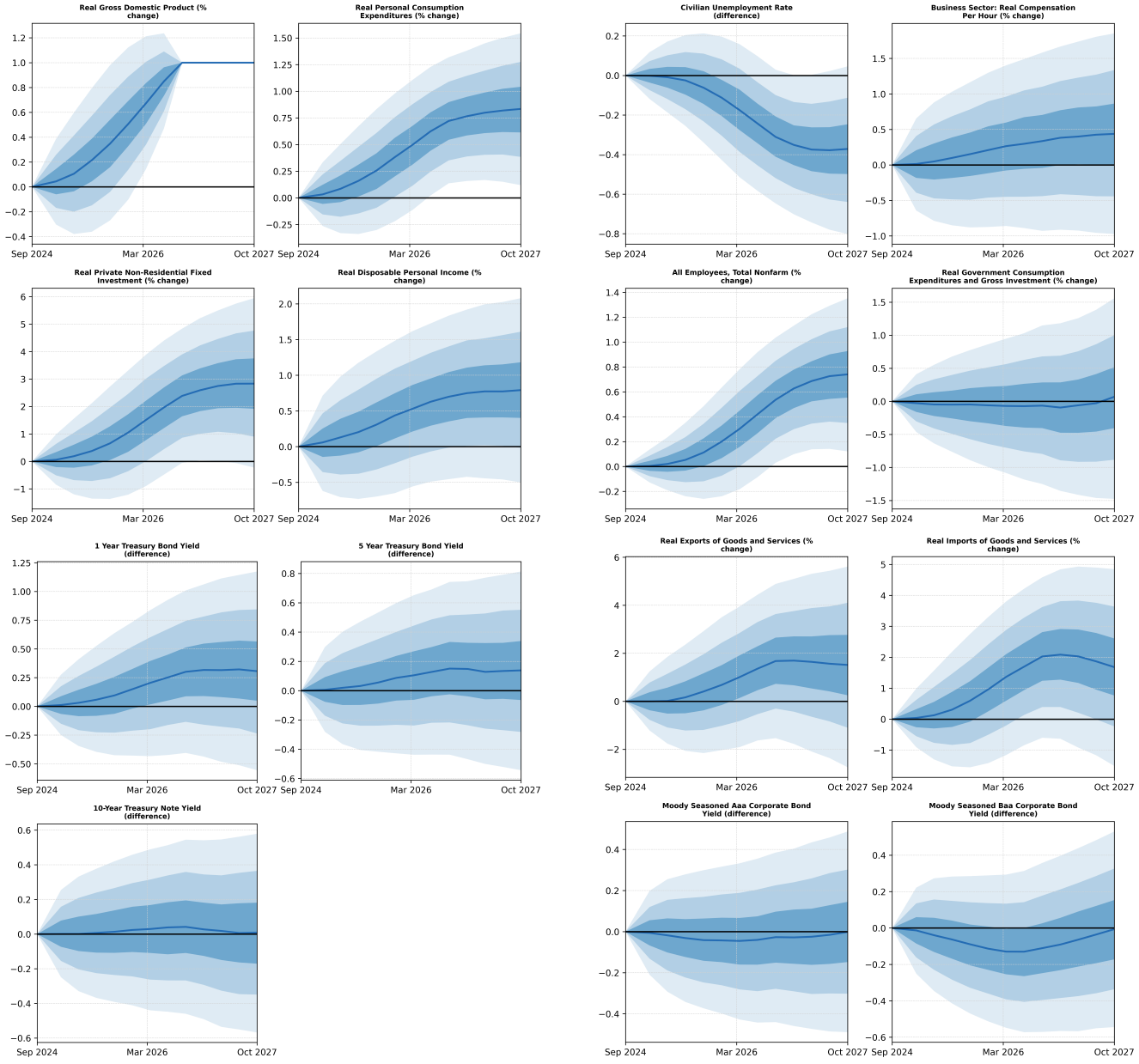

Figure 11. Response of 1 percent increase in real GDP 8-12 quarters ahead over a period of 3 years where the dark blue lines are the posterior median values.

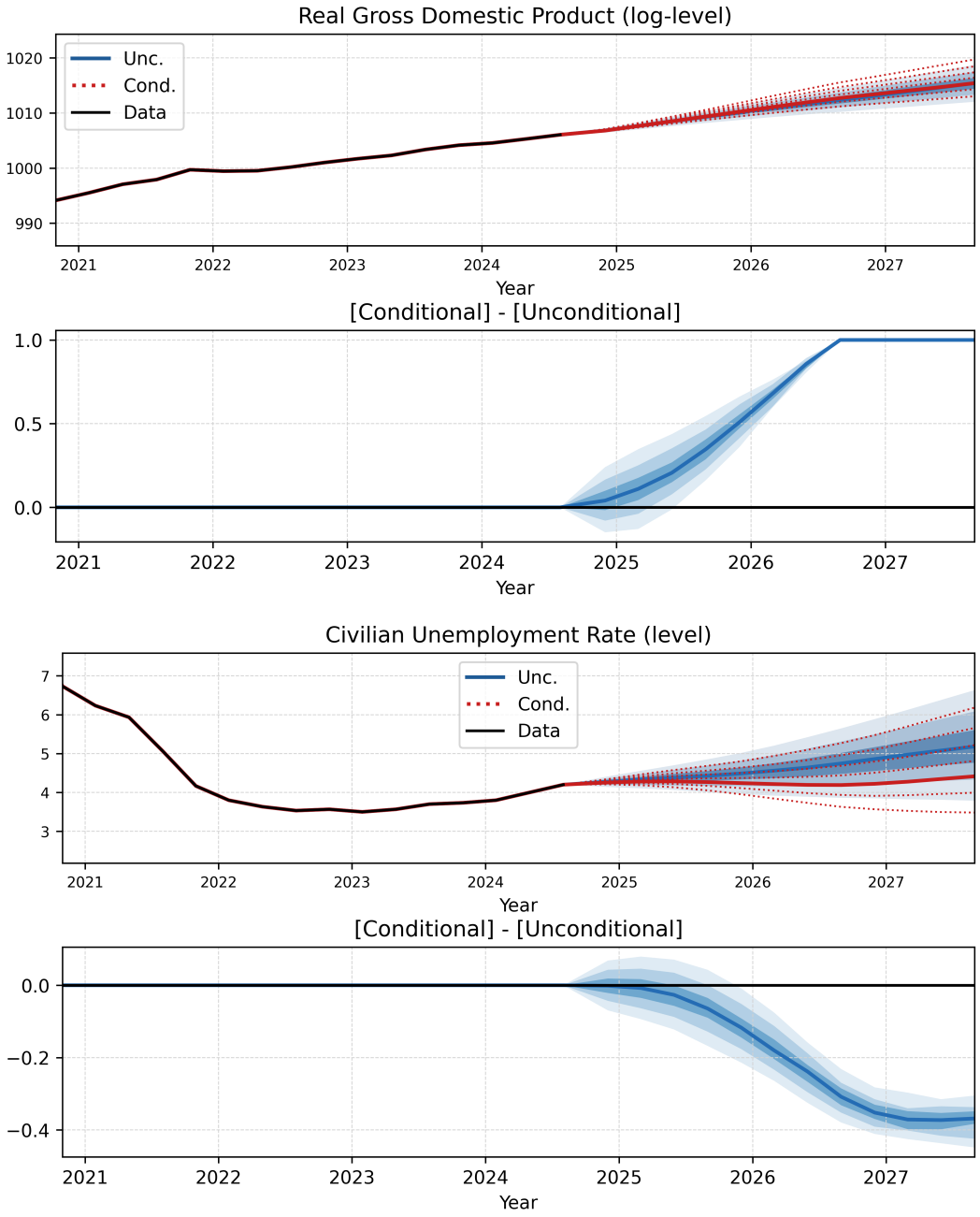

Figure 12a. Baseline (unconditional) forecasts (blue bands) and forecasts conditional on 1 percent increase in GDP 8-12 quarters ahead. The shaded bands are 60, 70, and 80 percent coverage intervals. The bottom row presents the plot of the log-level difference between the unconditional and conditional forecasts, which is equivalent to a generalized impulse response function to GDP.

Riding The Yield Curve: Increase in 1-year Treasury rate

Short and medium-term Treasury yields influence borrowing costs for households and business, impacting interest rates on car loans, student loans, corporate financing, etc. Changes in these yields ripple quickly through financial asset prices, namely equity valuations, risk premiums and exchange rates. Moreover, 1-year Treasury yield is very sensitive to changes in the federal funds rate or investors' expectations about the how the Fed will alter monetary

policy in the short-run. Given its importance, I examine two scenarios. First, in *increase1Y_unrestricted* script, I construct unorthogonalized impulse responses and scenario analyses to 75 basis points increase in 1-year Treasury yield one quarter ahead, leaving other variables unconstrained. Effectively, this imposes a scenario where the short-term rates jump above model's baseline predictions. Viewed as sudden tightening of financial conditions, this can happen for a number of reasons. Namely, when markets expect the policy rate to hike; the US Department of Treasury issues shorter-maturity Treasury bills to fulfill the escalating debt burden which drops prices and pushes yields upwards; tapered demand from large buyers of Treasuries; and perceived risk that inflation may run hotter than expected causing the buyers to demand higher yield to compensate for eroding purchasing power. Because the residuals in the reduced-form model are correlated, a shock to the 1-year rate implicitly moves other variables in the system based on historical correlations.

```
idxCV = Spec.index[Spec['SeriesID'] == 'GS1'].tolist()[0]  # 1-Year Treasury Yield
Shock = np.nan * np.ones((h_fore.sum(), n))
Shock[0, idxCV] = 0.75  # 75 bps increase in 1-Year Treasury Yield
```

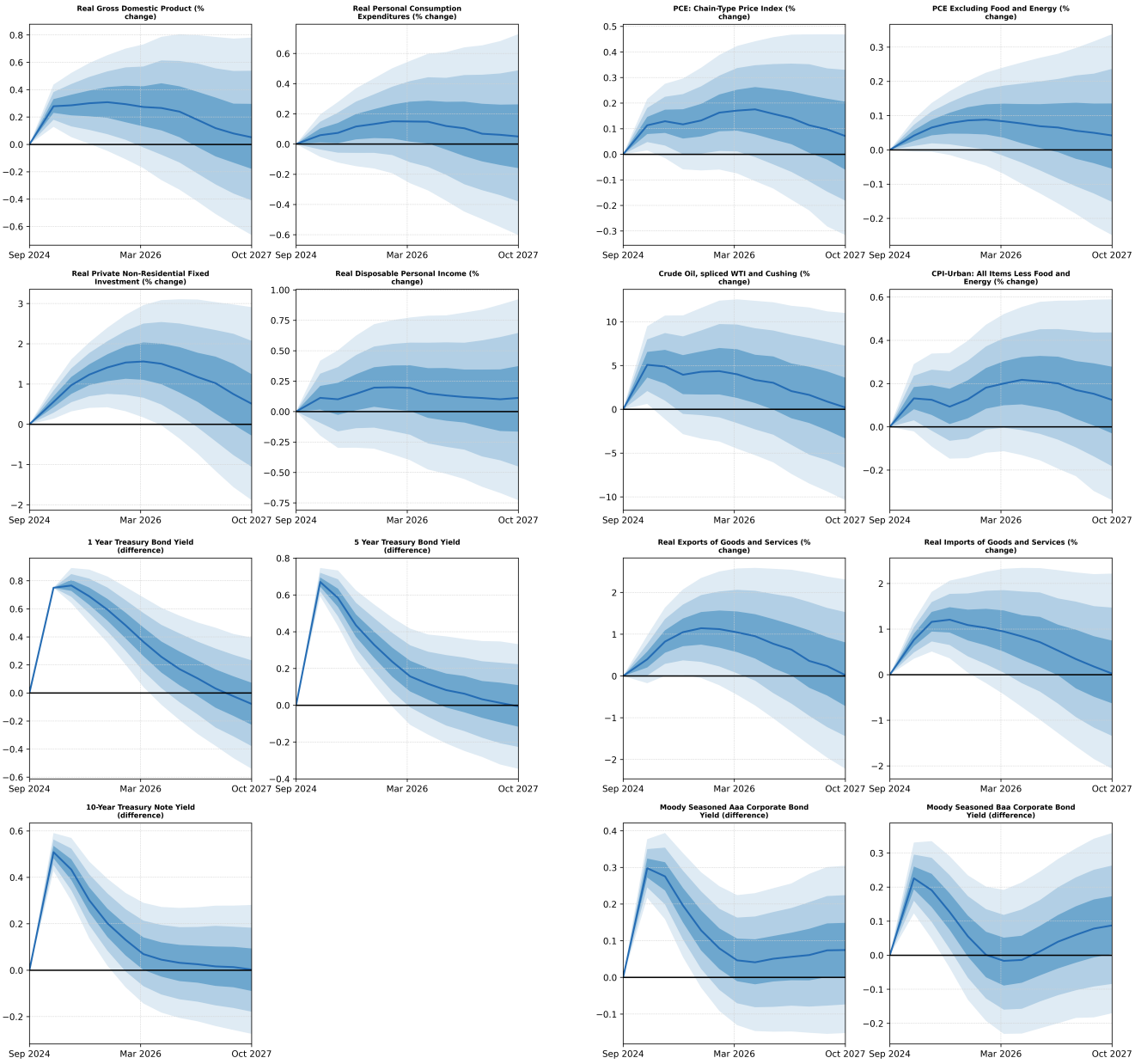

Figure 13: Response of a 75 bps increase in 1-year Treasury yield one quarter ahead over a period of 3 years where the dark blue lines are the posterior median values. All variables respond to the unrestricted shock endogenously.

Second, in *increase1Y_Cholesky* script, I impose a restricted scenario where I not only raise the 1-year Treasury rate by 75 basis points one quarter ahead, but also fix the near-term forecasts for real macroeconomic indicators (such as GDP, unemployment, prices, etc) to remain at their unconditional baseline for that same quarter.

```
idxGS1 = Spec.index[Spec['SeriesID'] == 'GS1'].tolist()[0]  # 1-Year Treasury Yield
# List of indices for macro variables
idxMacro = Spec.index[(Spec['isFinancial'] == 0)].tolist()
# Create a boolean array indicating variables used for plotting
idxCV = np.isin(range(n), idxMacro + [idxGS1])
Shock = np.nan * np.ones((h_fore.sum(), n))
Shock[0, idxMacro] = 0  # Set macro variables to unconditional forecast (0 shock)
```

```
Shock[0, idxGS1] = 0.75  # 75 bps (basis points) increase from the baseline forecast
```

In other words, albeit the Treasury rate hikes, the macro variables aren't immediately affected by the shock. This mimics a recursive or Cholesky identification scheme where "fast-moving" financial variables react contemporaneously to changes in the real-economy, but "slow-moving" macro variables adjust with a lag, analogous to tightening of monetary policy. For instance, a surprise hike of the policy rates may lift the short end of the yield curve instantaneously, but will not move real GDP in the same quarter.

Figure 13 and 14 depict the elasticities in the unconstrained and constrained scenarios, respectively. Juxtaposing the results, we can attribute the movement of the elasticities in the unconstrained scenario to all historical reasons for heightened Treasury yield, including those endogenously related to stronger economic conditions. By restricting the response of the macro variables immediately when Treasuries hike, the second restricted scenario filters out those "endogenous" components and isolates a pure (exogenous) monetary policy shock. Yet, the responses don't exactly align with the theoretically expected results where rate hikes decelerates real GDP shortly after the initial impact, and unemployment rises. Rather, real GDP, investment and government spending boost initially, unemployment falls as employees in non-farm payroll rises and wages decline *initially*, before reversing directions in the medium run. While this may contradict the theoretical underpinnings of Cholesky-identified VAR models, these patterns emerge from the BVAR model as the recent post-COVID data show a positive correlation between interest rates and real activity in the short-run. For instance, after the pandemic the economy rebounded strongly during 2021-2022 when the Fed signaled hiking rates. Unlike Crump et. al (2021), this model spans periods when the Fed raised the federal funds rate from ZLB in March 2022 to 5.25-5.5 percent points in July 2023, the highest since 1980s and demand surged after the pandemic slump and supply chain disruptions healed. Rate hikes co-existed with robust output growth and tight labor market – quite different from the pre-pandemic historical trends. Observing these atypical episodes wherein expansions coincide with or follow higher rates, the scenario analyses from BVAR with COVID volatility model extrapolate and depict the stated trends for the first two years after the shock. Afterward, in the third year, economic activity becomes sluggish, unemployment rises, consumer sentiments worsen, volatility measured by the VIX index clambers, imports and exports decline, and Moody's seasoned corporate bond yields drop. So, even with the "exogenous" shock, recent swings in pandemic data can cause the subsequent few quarters to show GDP mildly rise in the very short-run, but the downturn from the third year surfaces once the model's dynamics override the short-term historical patterns of tight labor markets.

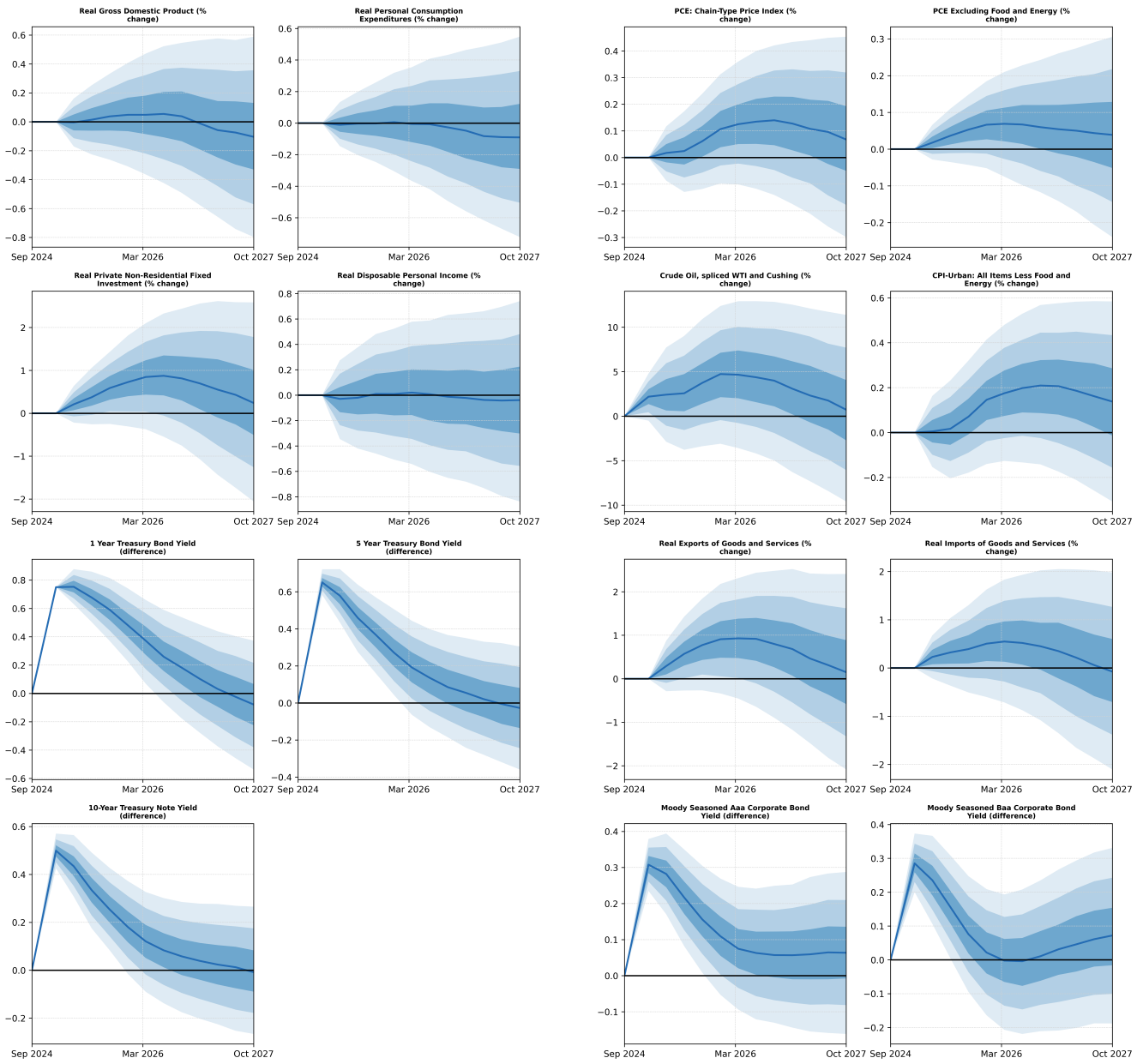

Figure 14: Response of a 75 bps increase in 1-year Treasury yield one quarter ahead over a period of 3 years where the dark blue lines are the posterior median values. This scenario affixes the macro variables to their unconditional forecasts when the Treasury yield hikes, but reacts endogenously next period onwards with a lag.

Financial Turmoil - Disentagling the Effects of Supply and Demand Shocks

Financial turbulence characterized 2015 – commodity and oil prices depressed, hitting a lowest of $30.31 since 2009, stock volatility spiked, major stock indices plunged, yields rose as investor demanded higher return for holding risky assets, US dollar appreciated as growing expectations of Fed rate hikes makes the dollar-denominated assets more attractive to global investor offering higher returns. Tighter financial conditions created a demand shock as borrowing rates rise, reducing consumer and business spending. Alternatively, falling oil and commodity prices

comprise of supply shocks which lower input costs for production. To analyze the effects of the 2015 financial turmoil, I first forecast all the variables in the system as if no financial turmoil occurred. In other words, I only use data until 2015-Q2 and construct baseline forecasts thereafter. Then I construct two scenarios - financial turmoil, and no-supply shock scenarios. The "financial turmoil" scenario conditions on the the observed financial market outcomes outcomes in 2015-Q3, such as Treasury and corporate bond yields, stock index, and volatility index. Figure 15 illustrates the responses of the variables. Notably, real GDP declines by 0.4 percent, unemployment rate rises by 0.1 percent at its peak, fully recovering after two years, corporate bond yields rise, headline and core inflation drop by around 0.2 and 0.3 percent respectively, recovering after three years. Most notably, the WTI crude oil price plunges by 22 percent before reverting to the baseline levels in three years. Moreover, VIX index spikes by around 20 percent, also diminishing consumer confidence as evident by nearly 3 percent drop in the University of Michigan Consumer Sentiment Index. The "financial turmoil" scenario shows that a combination of firm financial conditions (negative demand shock) and falling commodity prices (positive supply shock) diminished output and prices over the horizon.

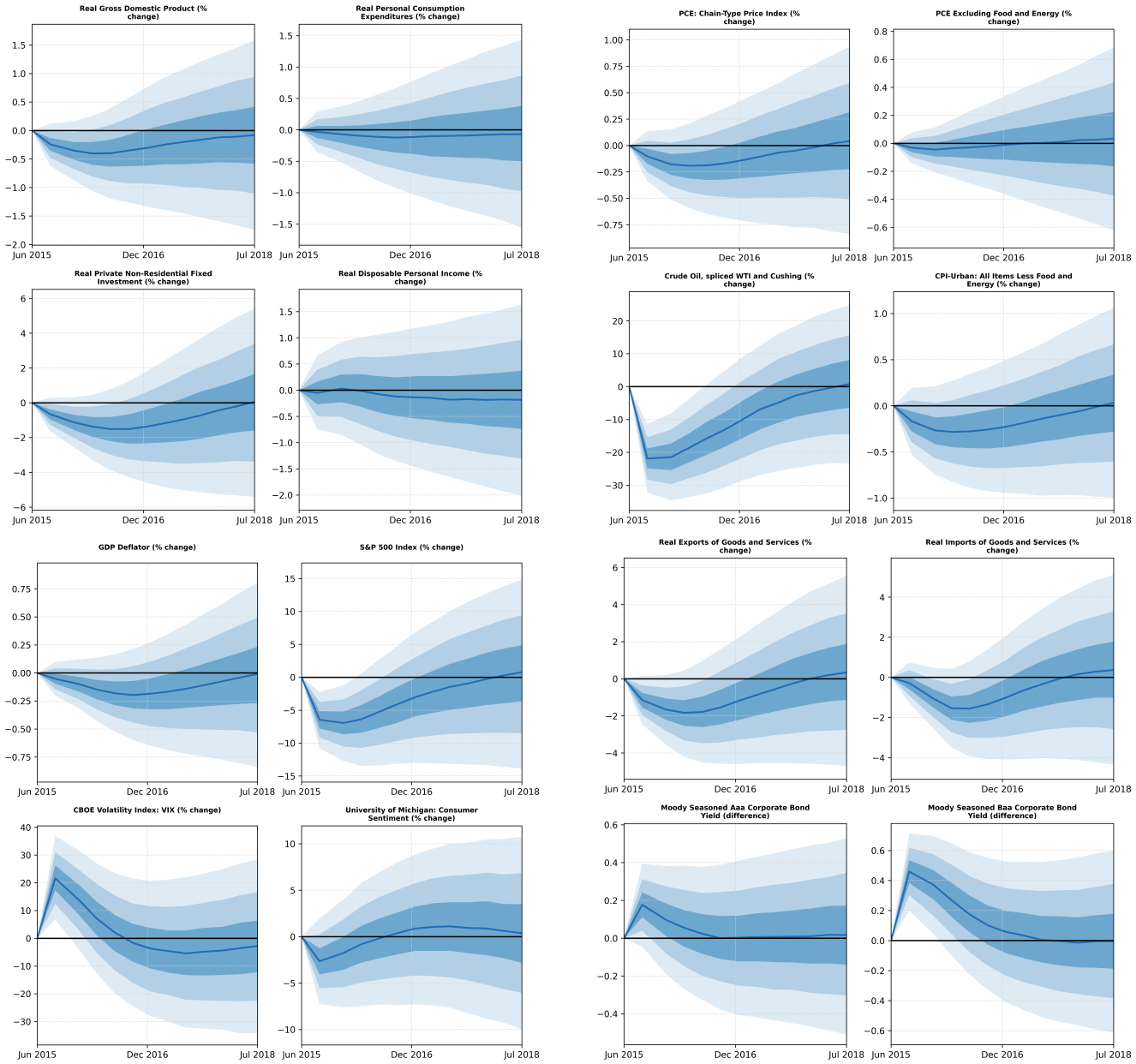

Figure 15: Financial turmoil scenario – the responses conditioned on the known paths of the financial variables in 2015-Q3.

To disentagle the elasticities of demand shock only, isolating the effects of supply shock, the second scenario is the "no-supply shock". Similar to the above scenario wherein I condition on the observed paths of financial variables, I exclude the WTI crude oil price from the list of conditioning variables and plot the elasticities in figure 16. Juxtaposing the elasticities of figure 16 with those in 15, the demand shock alone causes a deeper and prolonged decline in output as GDP and investment fall by 0.5 and 2 percent, respectively; and unemployment rate rises by 0.2 percent. Devoid of the positive supply shock, prices decline by lesser magnitude than in the first scenario. On the other hand, including the oil prices (first scenario) lowers the prices further as input costs slump. Furthermore, output grows as supply shock offsets some of the effects of the demand shocks.

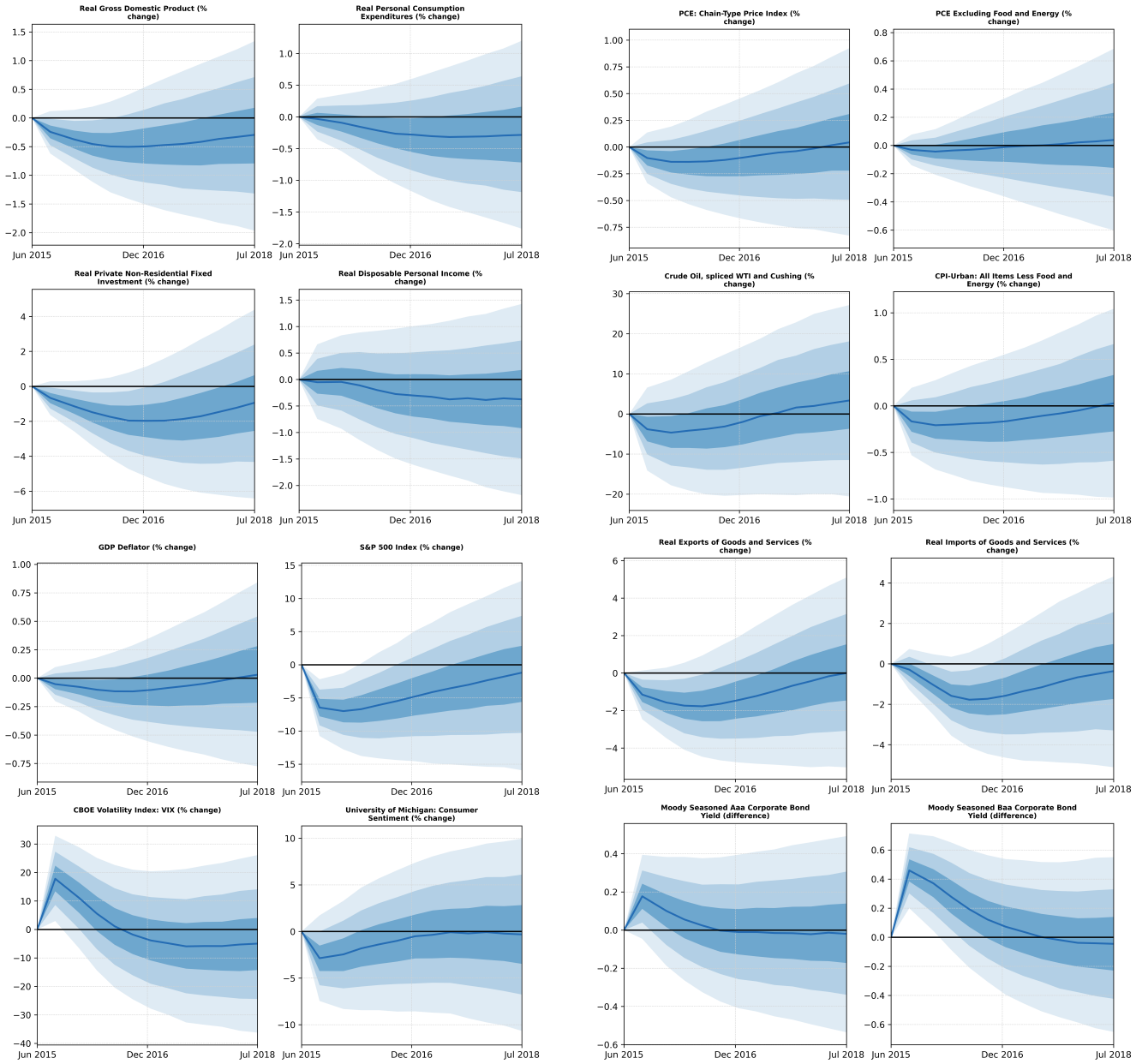

Figure 16: Financial turmoil excluding oil prices – conditioned on the observed values of all financial variables in 2015-Q3 except for WTI crude oil prices.

Pandemic Aftershocks: Detecting Structural Breaks During COVID and Why Inflation Defied Expectations

In addition to the examining the forecasts and impact of shocks, the model can detect structural breaks that occur when properties such as the mean, variance and covariances of variables significantly change over time. For instance, Furlanetto and Lepetit (2024) and Stock and Watson (2021) provide empirical evidence and theoretical reasons that the Phillips curve flattened in the pre-pandemic period. In early 2020, unemployment rate skyrocketed from 3.5 percent in February 20 to 15.8 percent in April 2020, while inflation remained subdued as core PCE inflation fell to 1 percent. Starting June 2020, inflationary pressures picked up as supply-chains were disrupted (due to congested

ports and dearth of semiconductor), commodity prices rose, fiscal stimulus boosted demand despite falling yet sky high unemployment of 11 percent. Inflation continued to accelerate while unemployment crept downwards, raising questions if the Phillips curve has steeped. Separately, from the policy front, the COVID-related Tax Relief Act of 2020 was enacted in late December 2020, authorizing additional Economic Impact Payments of up to $600 per eligible adult and $600 per qualifying child under the age of 17. While the quarterly growth rate of real GDP rebounded to 33.4 percent in Q3 after the economy reopened from the pandemic slump, it grew at 4 percent Q4 – a rate that aligns more closely with historical data. Likewise, consumer spending followed a similar trajectory – growing at a substantially lower pace 2.5 percent Q4 than in 41 percent in the prior quarter. These developments underscore the dynamic policy responses boosted by fiscal and monetary support and a rapid recovery in activity – whereas real GDP fell steeply and abruptly, it was temporary.

In light of the developments, I evaluate the structural changes in the dynamic system by first estimating the model in the entire dataset till Q3-2024, then I re-estimate the model till the break period of Q3-2020 in the *structuralBreak* file. With the BVAR estimates fitted in the entire data, I construct in-sample (IS) conditional forecasts from Q4-2020 till the end of the information set. Using the second model fitted till the break, I construct out-of-sample (OOS) conditional forecasts from Q4-2020 till Q3-2024 and let the historical correlations between variables in the data dictate the forecasts. In both cases, I build two types of conditional forecasts. First, I condition the out-of-sample forecasts on thirteen real activity variables, whose realized paths are known for the remainder for the information set after the break, Second, I again condition the forecasts given the realized paths of unemployment rate and real GDP only.

```
# List of real activity variables
realActivityVars = [
    'GDPC1',  # Real GDP
    'PCECC96', # Personal Consumption Expenditures
    'PRFIx',  # Real Private Residential Fixed Investment
    'PNFIx', # Real Private Non-Residential Fixed Investment
    'EXPGSC1',  # Real Exports of Goods and Services,
    'IMPGSC1',  # Real Imports of Goods and Services,
    'GCEC1',  # Real Government Consumption Expenditures and Gross Investment
    'INDPRO',  # Industrial Production Index
    'CUMFNS',  # Capacity Utilization: Manufacturing
    'UNRATE',  # Civilian Unemployment Rate
    'RCPHBS',  # Business Sector: Real Compensation Per Hour
    'HOUST',  # Housing Starts
    'DPIC96'  # Real Disposable Personal Income
]

# Settings for two conditioning schemes
Settings = [
    {
        'idxCond': Spec.index[Spec['SeriesID'].isin(realActivityVars)].tolist(),
        'labCond': 'real activity'  # Condition on real activity
    },
    {
        'idxCond': Spec.index[Spec['SeriesID'].isin(['GDPC1', 'UNRATE'])].tolist(),
        'labCond': 'GDP and unemployment rate'  # Condition on GDP and unemployment rate
    }
```

```
]

jBreak = np.where((dates.dt.year == 2020) & (dates.dt.month == 9))[0][0]  # Date of break
heff = T - jBreak  # Number of periods being forecasted

# Out of sample: model estimated through the break
# These results are out-of-sample in the sense that forecasts are generated for dates
    beyond the break
bvar_results_OOS = bvar.bvarGLP_covid(data_transformed[:jBreak, :], lags=lags,
    priors_params=priors_params, mcmc=1, MCMCconst=1, MNpsi=1, sur=0, noc=0, Ndraws=Ndraws,
    Ndrawsdiscard=discard, hyperpriors=1, Tcovid=Tcovid)

# In-sample: model estimated on full date
# These results are in-sample in the sense that forecasts are generated for the dates
    contained in this sample
Testim = np.nanmax(np.where(~np.isnan(data_transformed.sum(axis=1)))[0]) + 1
bvar_results_IS = bvar.bvarGLP_covid(data_transformed[:Testim, :], lags=lags,
    priors_params=priors_params, mcmc=1, MCMCconst=1, MNpsi=1, sur=0, noc=0, Ndraws=Ndraws,
    Ndrawsdiscard=discard, hyperpriors=1, Tcovid=Tcovid)

# Determine the number of draws in the MCMC simulation
ndraws = bvar_results_IS['mcmc']['beta'].shape[2]

############################## Get unconditional forecasts ##############################

for setting in Settings:
    idxCond = setting['idxCond']
    labCond = setting['labCond']

    # Generate conditional forecasts
    YCond = np.nan * np.ones((T, n))
    YCond[:jBreak, :] = data_transformed[:jBreak, :] # fill historical data up to the break
        for non-conditioning variables
    YCond[:, idxCond] = data_transformed[:, idxCond] # fill paths of conditioning variables
        for all time period

    # Initialize storage for conditional forecasts
    PredYIS = np.nan * np.ones((T, n, Ndraws - discard))
    PredYOOS = np.nan * np.ones((T, n, Ndraws - discard))

    for j in range(Ndraws - discard):  # Loop through the number of draws
        if (j % 1000) == 0 or j == Ndraws - 1:
            print(f"Processing draw {j} of {Ndraws - discard}...")
            sys.stdout.flush()


        # In-Sample (IS) forecasts
        beta_j = bvar_results_IS['mcmc']['beta'][:, :, j]
        Gamma_j = np.vstack((beta_j[1:, :], beta_j[0, :]))
```

```
        Su_j = bvar_results_IS['mcmc']['sigma'][:, :, j]
        PredYIS[:, :, j] = bvar.VARcf_DKcks(YCond, bvar_results_IS['lags']['lags'], Gamma_j,
            Su_j, 1)

        # In-Sample (IS) forecasts
        beta_j = bvar_results_OOS['mcmc']['beta'][:, :, j]
        Gamma_j = np.vstack((beta_j[1:, :], beta_j[0, :]))
        Su_j = bvar_results_OOS['mcmc']['sigma'][:, :, j]
        PredYOOS[:, :, j] = bvar.VARcf_DKcks(YCond, bvar_results_OOS['lags']['lags'],
            Gamma_j, Su_j, 1)

    # Growth rates (quarterly growth, annualized)
    dPredYIS = np.concatenate((np.nan * np.ones((1, n, Ndraws - discard)),
        (PredYIS[1:, :, :] - PredYIS[:-1, :, :]) * 4), axis=0)

    dPredYOOS = np.concatenate((np.nan * np.ones((1, n, Ndraws - discard)),
        (PredYOOS[1:, :, :] - PredYOOS[:-1, :, :]) * 4), axis=0)
```

I assess the magnitude of the difference in the in-sample and out-of-sample forecasts to examine if the pre-break model can accurately forecast post-break dynamics. In other words, do the economic and structural changes from before and after the COVID Tax Relief Act was enacted, affect economic and financial relationships? If the OOS forecasts deviate significantly from the observed data and IS forecasts, this indicates that the relationships estimated pre-break no longer hold, implying a structural break. Alternatively, if they are consistent, the relationships across the periods are stable.

The first panel of figure 17 depicts the trajectory of Core CPI inflation conditioned on GDP and unemployment rate, whereas that in the second panel conditions on thirteen real economic activity variables. The OOS forecasts are slightly closer to the actual data when conditioned on the thirteen real economic activity variables than when conditioned on only two of them. However, the OOS forecasts deviate significantly from from the IS forecasts, pointing evidence to structural breaks in CPI inflation rate. Furthermore, the OOS forecasts in both panels underestimate the inflationary pressures that brewed after the expansionary fiscal programs of the government pumped in stimulus checks and tax breaks to households and firms. The COVID–Related Tax Relief Act of 2020 is one prominent example that contributed to the demand-side factors fueling inflation. The BVAR model estimated up to Q3-2020 (before the break) is based on historical correlations, placing more weight on the past behavior of inflation, particularly when inflation expectations were stable, low and generally well-anchored. So, it's information set is devoid of the period when the the law was passed and the macro-financial relationships changed post-COVID, showing pronounced structural breaks in inflation forecasts. Likewise, the OOS conditional forecasts of Treasury yields and corporate bonds (not shown here) are lower than their actual counterparts, and the structural breaks are more pronounced. This is because Treasury yields and corporate bonds reflect expectations of inflation and monetary policy. Albeit, pre-COVID, these interest rates were low, the Fed aggressively pivoted in 2022-2023 by hiking rates to curb inflation. This structural shift was not embedded in the model estimated till the break.

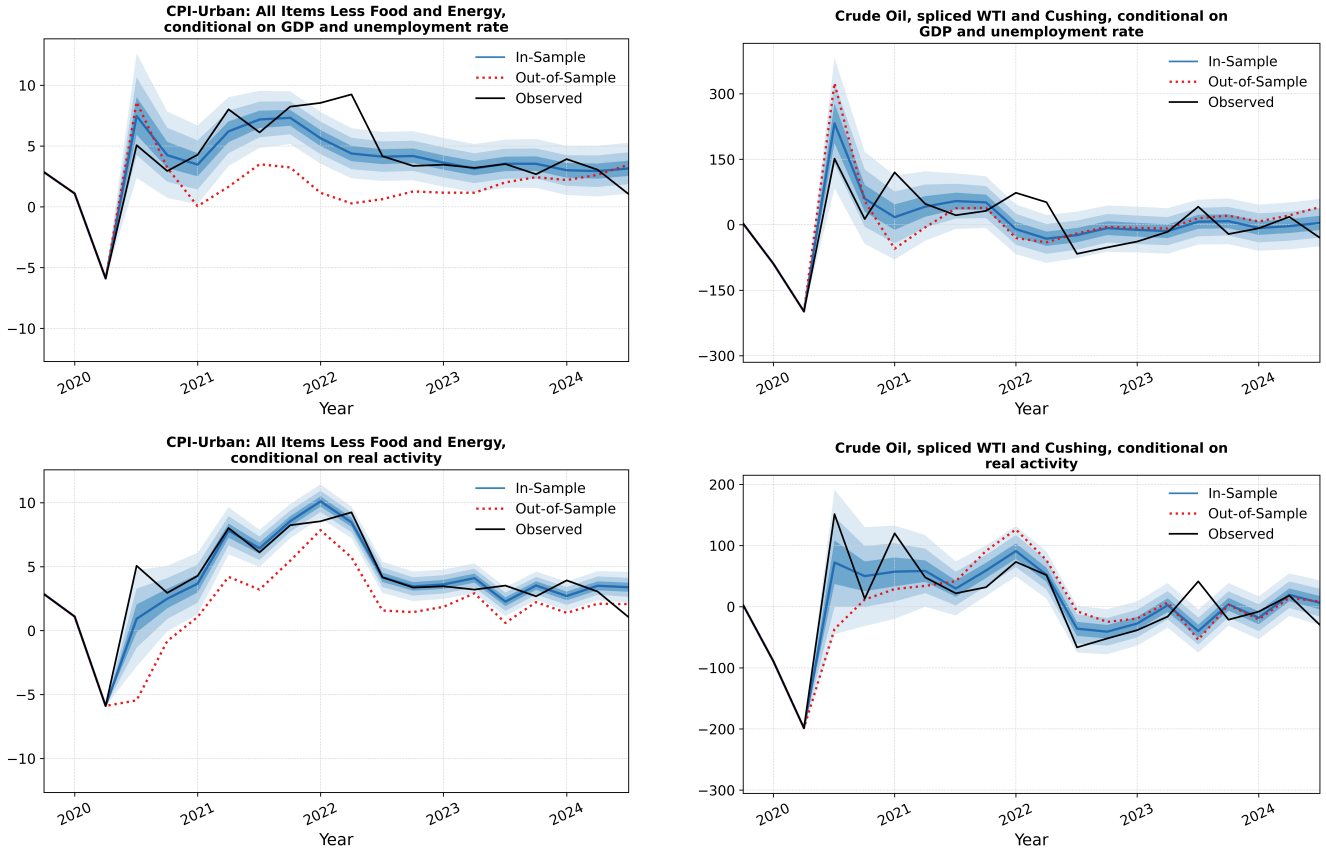

Figure 17: In-sample (IS) and out-of-sample (OOS) forecasts of core CPI and WTI oil prices are conditioned on the known paths of real GDP and unemployment rate in the top row, and conditioned on the known paths of thirteen real activity variables in the bottom row. The black lines are the historical values of core CPI and oil prices, whereas the blue shades depict the 60, 70 and 80 percent bands for the in-sample forecasts. The dark blue lines represent the median in-sample forecasts constructed using the model estimated till Q3-2024.The red dotted lines are the in-sample forecasts constructed using the model estimated till the break, Q3-2020.

Yet, the structural breaks aren't visually conspicuous in the forecasts of other macro and financial variables as evident in figure 18. These variables include crude oil prices, volatility index, residential and non-residential investment, and hourly compensation to name a few. As "fast-moving" variables, financial metrics adapt quicker to the new economic environment relative to the "slow-moving" macro variables, and tend to revert to the means after extreme events (for example, VIX spikes then collapses, stock indices drop then rally). Given that the pre-2020 data contains other crises periods such as the 2008 Global Financial Crisis, the BVAR with covid volatility model captures these dynamics, reducing the forecast errors between OOS conditional forecasts and realize data, and between OOS and IS forecasts.

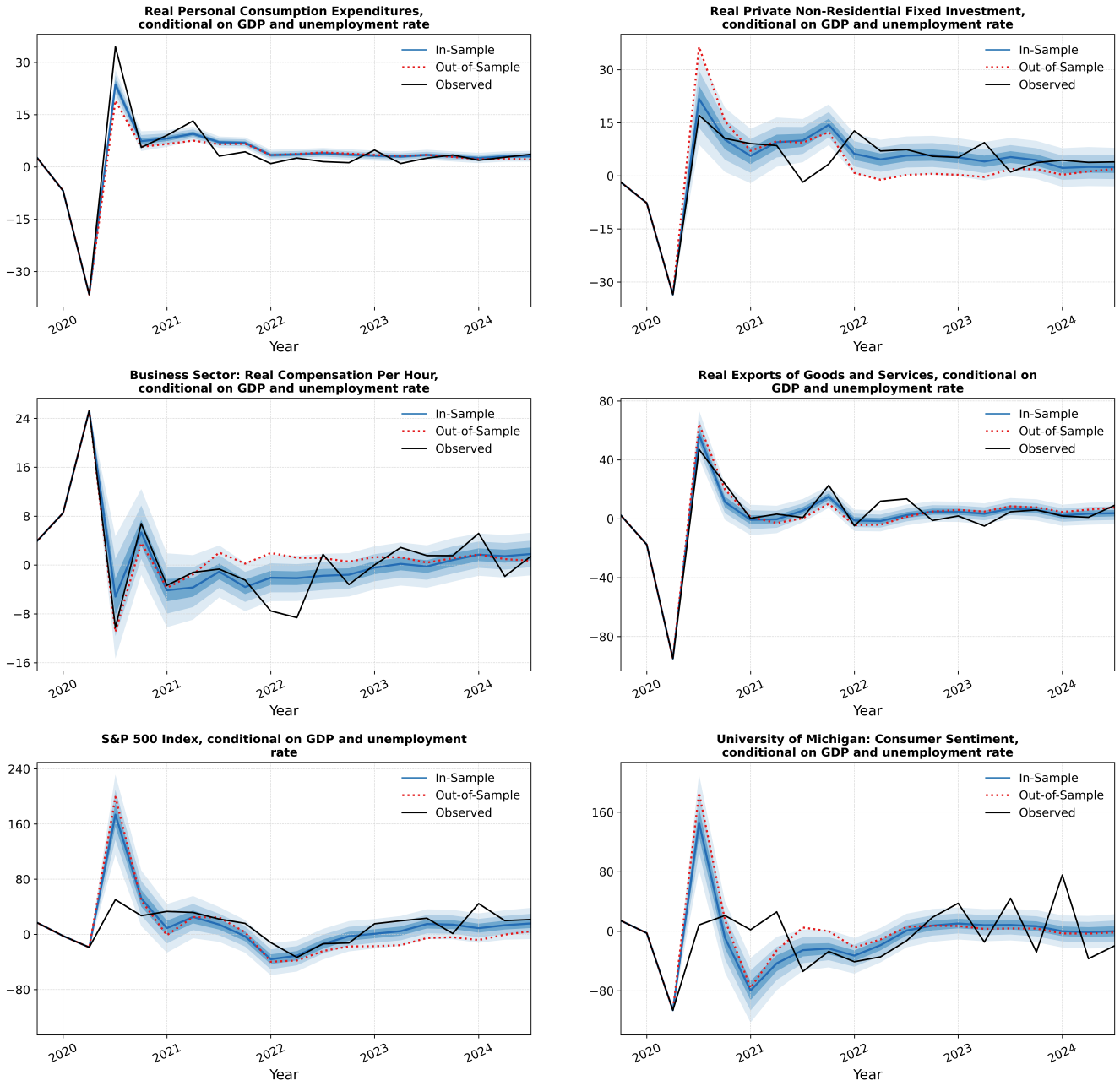

Figure 18: In-sample (IS) and out-of-sample (OOS) forecasts of macro-financial variables, conditional on the known paths of real GDP and unemployment rate. The black lines are the historical values of the variables, whereas the blue shades depict the 60, 70 and 80 percent bands for the in-sample forecasts.The dark blue lines represent the median in-sample forecasts constructed using the model estimated till Q3-2024.The red dotted lines are the in-sample forecasts constructed using the model estimated till the break, Q3-2020.

6. Replication Figures from Lenza and Primiceri (2022): How to estimate a VAR after March 2020

Lenza and Primiceri (2022) examine how the impulse responses functions, and conditional forecasts of variables change when we estimate the model with and without COVID-volatility. Their dataset comprises of unemployment rate, employment measured via total number of employees in non-farm payroll, and five pricing variables such as PCE Services and Core PCE. I attempt to replicate figures 2,3 and 4 of their paper using the functions in `covbayesvar` package and the python scripts. Whilst I used the exact dataset they employed for their analysis, we can also obtain the updated data directly from the FRED Economic Data repository, operated by St. Louis Fed. The *Data/LP Data Collection and Cleaning* has a short script to easily retrieve the data automatically from their website using an API, rather than manually downloading data separately.

```
from fredapi import Fred

fred = Fred(api_key='write_your_API_here')
file_name = 'LP_data.xlsx'


# monthly data from FRED-MD
series_list = [
    'CPIAUCSL',  # Consumer Price Index for All Urban Consumers: All Items
    'DDURRG3M086SBEA',  # Personal consumption expenditures: Durable goods (chain-type price
        index)
    'DGDSRG3M086SBEA',  # Personal consumption expenditures: goods (chain-type price index)
    'DNDGRG3M086SBEA',  # Personal consumption expenditures: Non-durable goods (chain-type
        price index)
    'DPCCRC1M027SBEA',  # Personal consumption expenditures excluding food and energy
        (Billions of dollars)
    'DSERRG3M086SBEA',  # Personal consumption expenditures: Services (chain-type price
        index)
    'INDPRO',  # Industrial Production Index, Index 2017=100
    'PAYEMS',  # All employees, total nonfarm, Thousands of Persons
    'PCE',  # Personal consumption expenditures, Billions of dollars
    'PCEDG',  # Personal Consumption Expenditures: Durable Goods
    'PCEND',  # Personal Consumption Expenditures: Non-Durable Goods
    'PCEPI',  # Personal consumption expenditures: Chain-type price index
    'PCEPILFE',  # Personal consumption expenditures: Chain-type price index excluding food
        and energy
    'PCES',  # Personal Consumption Expenditures: Services, Billions of Dollars
    'UNRATE',  # Civilian Unemployment Rate

]
data = DataReader(series_list, "fred" ,start=datetime(1947, 1, 1), end=datetime(2025, 2, 1))
```

Figure 19 presents the generalized impulse responses for key pricing and labor variables when unemployment rate hikes by 1 percentage point and when we place the unemployment rate first in the Cholesky identification scheme. In the top left panel, the unemployment rate jumps due to the shock, but gradually declines returning to its

baseline level. The red solid line depicts the posterior median responses when the model accounts for heightened volatility during the COVID months. These contrast with the grey dashed line, generated after estimating the model with constant volatility in the full sample till May 2021. Accounting for the COVID volatility, the responses of unemployment (top-left panel) decline shower relative to the responses from the constant volatility estimated estimated both till full-sample of May 2021 (grey dotted line) and the pre-pandemic sample February 2020 (blue dotted line). Therefore, the COVID volatility model suggests that the macro variables drop deeper for longer duration than those from the constant volatility model. Likewise, prices decline, indicating a persistent downward pressure on inflation. The COVID volatility model predicts a weaker decline than that from the conventional model. Although the responses are consistent with the notion that disruptions in the labor market lower demand and prices, the responses from the constant volatility model appear unrealistic, showing erratic movements that differ from the historical relationships. Hence, including the post-pandemic data without adjusting for time-varying volatility yields biased and unstable estimates, which juxtaposes with the more robust and credible estimates from the COVID-volatility model.

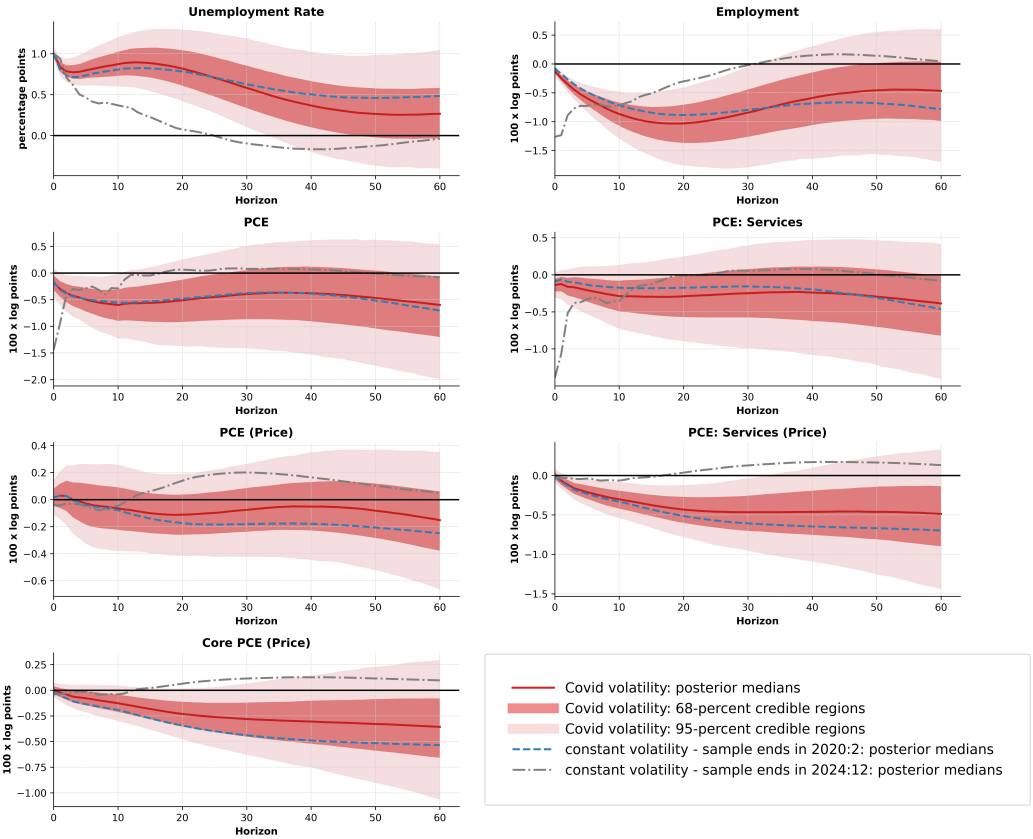

Figure 19: Generalized impulse responses of all macro variables when the unemployment rate rises by 1 percentage point. To identify the structural shocks using Cholesky decomposition, we order the unemployment rate first.

To generate the graphs in figure 19, refer to the scripts in *LP 2021* folder named *Baseline_May2021*, *CV_May2021*, *CV_Feb2020_May2021*, and *generate_figures*. For instance, first I load the data, and define the dates up to which we estimate the model, forecast horizons, and apply other transformations in the data as follows:

```
data['UNRATE'] = np.exp(data['UNRATE'] / 100)
# Real PCE: nominal PCE / PCE deflator
```

```
data['PCE_real'] = data['PCE'] / data['PCEPI']
# Real PCE services: nominal PCE services / PCE services deflator
data['PCE_services_real'] = data['PCES'] / data['DSERRG3M086SBEA']
# Selecting variables in the baseline model
indmacro = ['UNRATE', 'PAYEMS', 'PCE_real', 'PCE_services_real', 'PCEPI',
    'DSERRG3M086SBEA', 'PCEPILFE']
# Y-axis labels for IRF and Forecast plots
YLABELirf = ["percentage points", "100 x log points", "100 x log points", "100 x log
    points", "100 x log points",
             "100 x log points", "100 x log points"]
YLABELfcst = ["percentage points", "index", "index", "index", "index", "index", "index"]

# Choice of estimation sample, constant or varying volatility, and forecasting period
T0 = data.index[(data['DATE'].dt.year == 1988) & (data['DATE'].dt.month == 12)][0]     #
    beginning of estimation sample
T1estim = data.index[(data['DATE'].dt.year == 2021) & (data['DATE'].dt.month == 5)][0]    #
    end of estimation sample

T1av = T1estim       # date of last available data for forecasting
Tend = T1estim       # date of last available data in the dataset

# Position of the Feb 2020 observation
Tfeb2020 = data.index[(data['DATE'].dt.year == 2020) & (data['DATE'].dt.month == 2)][0]
Tcovid = Tfeb2020 - T0 + 1           # first time period of COVID (March 2020; set to
    "None" if constant volatility)

Tjan2019 = Tfeb2020 - 13    # initial date for conditional forecast plots
TendFcst = Tfeb2020 + 22 + 6
# TendFcst = Tfeb2020 + 71   # end date for projections (June 2022)
hmax = TendFcst - T1av    # corresponding maximum forecasting horizon

# Monthly VAR estimation
Ylev = data.loc[T0:T1estim, indmacro]
Ylog_df = 100 * np.log(Ylev)
Ylog = Ylog_df.to_numpy()
Time = data['DATE'].iloc[T0:]
T, n = Ylog.shape
```

Then, I estimate the model using the `bvarGLP_covid` function as shown previously, and estimate the generalized impulse response functions (GIRFs) to shock to the unemployment rate using the `bvarIrfs` function. The `bvarIrfs` computes the GIRFs using the Cholesky identification scheme to identify structural shocks, allowing us to analyze how a shock to one variables propagates through the system over time.

```

############################ generalized IRFs to an "unemployment" shock
    ############################
# Compute the IRFs
H = 60
M = bvar_results['mcmc']['beta'].shape[2]
```

```
Dirf1 = np.zeros((H+1, Ylog.shape[1], M))

for jg in range(M):
    Dirf1[:, :, jg] = bvar.bvarIrfs(bvar_results['mcmc']['beta'][:, :, jg],
        bvar_results['mcmc']['sigma'][:, :, jg], 1, H+1)
sIRF1 = np.sort(Dirf1, axis=2)
```

Figure 20 presents conditional forecasts under two different estimation methods: COVID volatility model till June 2020 in the left panel, and constant-volatility model using data up to February 2020 in the right panel. In other words, as of June 2020, these forecasts simulate the predictions for employment, and varying measures of prices, given the observed paths of unemployment rate (top row), including actual realizations from July 2020 - May 2021, and projections from the Blue Chip Consensus Forecasts for unemployment rate after June 2021. The shaded regions are the 68 and 95 percent credible intervals in both figures. These intervals are wider when we account for COVID volatility (left panel), indicating that the COVID-volatility model accounts for increased uncertainty during the pandemic. On the other hand, the forecast intervals from the constant-volatility model are narrower, implying more confidence in the forecasts and less uncertainty. Moreover, the realized data (black crosses) often fall outside or near the edges of the blue credible regions, connoting that the model underestimated the true variance of economic fluctuations. This contrasts with the wider forecast distributions of the COVID-volatility model that encompasses the realized data as they mostly fall inside the red shaded regions. Therefore, the COVID-volatility model provides more accurate probabilistic forecasts.

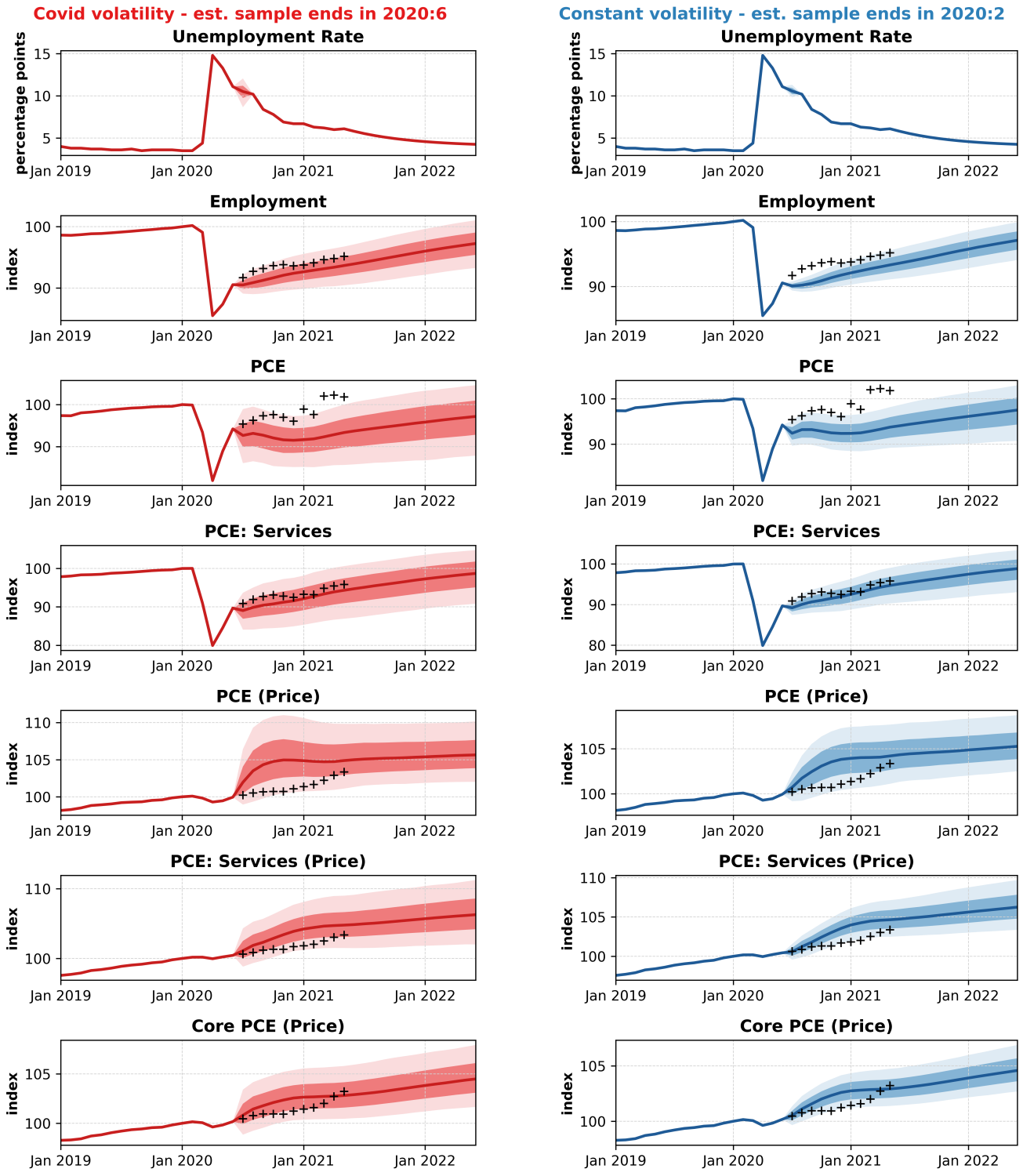

Figure 20: Conditional forecasts as of June 2020 given the realized paths of unemployment rate from June 2020 - May 2021 (black crosses), and the Blue Chip Consensus projections of unemployment rate from June 2021. The dark red and blue lines are the posterior median forecasts, and the shaded regions are the 68 and 95 percent credible posterior distributions.

```
############## conditional forecasts ################

# Create a matrix to store historical data (Jan 2019-May2021) and store future prokections
YYfcst = np.vstack([100 * np.log(data.loc[Tjan2019:T1av, indmacro].to_numpy()),
   np.full((hmax, n), np.nan)
])

# Condition on the Blue Chip forecasts of unemployment rate after July 2021
# start at the last known value of 5.8% and converge to 4% at an exponential decay of 0.85
    per month
YYfcst[-hmax:, 0] = 4 + (5.8 - 4) * (0.85 ** np.arange(hmax))
TTfcst = YYfcst.shape[0] # Total time periods for forecasting
M = bvar_results['mcmc']['beta'].shape[2] # Number of MCMC samples (posterior draws)
DRAWSY = np.full((n, TTfcst, M), np.nan)

# Forecasts: Loop Over MCMC Posterior Samples
for i in range(M):
   betadraw = bvar_results['mcmc']['beta'][:, :, i]
   G = np.linalg.cholesky(bvar_results['mcmc']['sigma'][:, :, i]).T


    # Handling Time-Varying Volatility
   if Tcovid is None:
       etapar = [1, 1, 1, 1] # Constant volatility case
       tstar = 1000000  # arbitrary large value to avoid COVID adjustment
   else:
       etapar = bvar_results['mcmc']['eta'][i, :]
       tstar = TTfcst - hmax + Tcovid - T # First period of COVID shocks
   varc, varZ, varG, varC, varT, varH = bvar.form_companion_matrices_covid(betadraw, G.T,
       etapar, tstar, n, lags,
                                                                           TTfcst)
   s00 = np.flip(YYfcst[:lags, :], axis=0).T.flatten().reshape(-1, 1)
   P00 = np.zeros((n * lags, n * lags))
   DrawStates, shocks = bvar.disturbance_smoother_var(
       YYfcst, varc, varZ, varG, varC, varT, varH, s00, P00, TTfcst, n, n * lags, n,
           'kalman'
   )
   DRAWSY[:, :, i] = DrawStates[:n, :]

IRFA = DRAWSY[:n, :, :]
IRFAsorted = np.sort(IRFA, axis=2)
```

Finally, figure 21 illustrates conditional forecast, conditioning on the Blue Chip Consensus Forecasts of unemployment released in June 2021. The left panel accounts for time varying volatility, implying that shocks were more extreme at the onset of the pandemic, but gradually decay over time. The right panel excludes all the data after February 2020, disregarding the unique patterns in the shocks observed during the pandemic, and assuming that the pandemic doesn't affect the volatility. Despite the differences in the modeling assumptions, the forecast distribution gauging uncertainty around the median forecasts are similar across both models. This is because the estimated volatility decay parameter is 0.8, tapering the shock volatility over time which returns to pre-pandemic levels. Thus,

by May 2021, the differences in volatility, reflected in the forecast intervals, have narrowed, yielding similar levels of forecast dispersion.

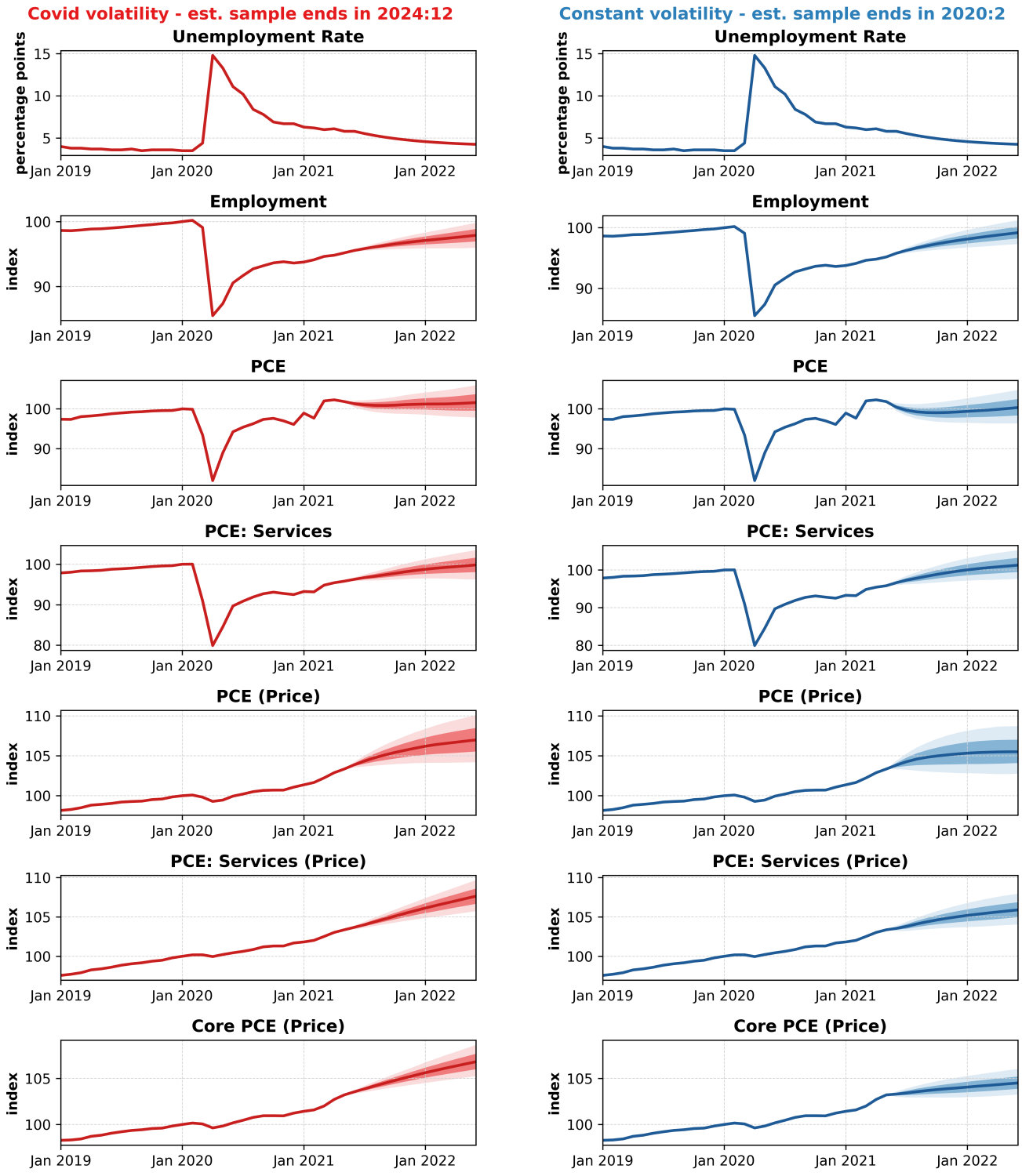

Figure 20: Conditional forecasts as of May 2021 given the realized paths of unemployment rate from June 2020 - May 2021, and the Blue Chip Consensus projections of unemployment rate from June 2021. The dark red and blue lines are the posterior median forecasts, and the shaded regions are the 68 and 95 percent credible posterior distributions.

8. Summary and Discussion

In this paper, I detailed the procedure to implement a large BVAR model with COVID volatility using the novel python's `covbayesvar` package, which is an amalgamation of the works of Giannone Lenza and Primiceri (2015), Banbura, Giannone and Lenza (2015), Crump et. al (2021), and Lenza and Primiceri (2021). I applied the model to the monthly and quarterly frequency medium-sized datasets, explained the functionalities of the most crucial Python functions, and conducted a Markov chain simulation exercise. Then, I presented use cases using both monthly and quarterly frequency data. Additionally, I replicated the figures from Lenza and Primiceri (2022) to demonstrate the distinguishing features of the forecasting the variables in the system with and without COVID volatility, and underscore that the model without COVID volatility underestimates the degree of uncertainty around the forecasts during COVID. To the best of my knowledge, this is the first open-source package which can estimate the model with COVID volatility and is versatile as it answers a breadth of policy questions via a reduced-form approach using a large number of variables.

Nevertheless, we can extend it to solve a diverse array of questions. A simple extension is to use a similar BVAR with COVID volatility to construct an exchange rate index for the US (similar to a trade-weighted US dollar index), which, to my knowledge, no one has constructed before using a BVAR model. That entails gathering a larger set of international-oriented data on bilateral exchange rates, import and export price indices, trade balance of goods on a balance of payment basis, and trade data on US imports and exports of goods. Although I run the model for the US only, we can apply this model for more than one country, and enhance the efficiency by parallelizing the code using the `joblib` package, exploiting all CPU cores. This is feasible and will drastically reduce the time to run the model, and produce and store the results. An alternative is to deploy a cluster of computers to radically speed up the estimation process, such as SageMaker Studio instances in Amazon Web Services.

References

1. Banbura, M., Giannone, D., & Lenza, M. (2015). Conditional forecasts and scenario analysis with vector autoregressions for large cross-sections. *International Journal of Forecasting, 31*(3), 739–756. https://doi.org/10.1016/j.ijforecast.2014.08.013

2. Giannone, D., Lenza, M., & Primiceri, G. E. (2015). Prior selection for vector autoregressions. *The Review of Economics and Statistics, 97*(2), 436–451. https://doi.org/10.1162/REST_a_00483

3. Crump, R. K., Eusepi, S., Giannone, D., Qian, E., & Sbordone, A. M. (2021). A large Bayesian VAR of the United States economy. *NY Fed Staff Report.* https://www.newyorkfed.org/research/staff_reports/sr976

4. Lenza, M., & Primiceri, G. (2022). How to estimate a VAR after March 2020. *Journal of Applied Econometrics, 37*(4), 688–699. https://doi.org/10.1002/jae.2895

5. Furlanetto, F., & Lepetit, A. (2024). *The slope of the Phillips curve. Finance and Economics Discussion Series, 2024-043.* Washington, DC: Board of Governors of the Federal Reserve System. https://doi.org/10.17016/FEDS.2024.043

6. Stock, J. H., & Watson, M. W. (2021). *Slack and cyclically sensitive inflation. Journal of Money, Credit, and Banking, 52*(s2), 393–428. https://doi.org/10.1111/jmcb.12757

7. Clements, M. P., & Galvao, A. B. (2024). Macroeconomic forecasting using BVARs. In *Handbook of Research Methods and Applications on Macroeconomic Forecasting* (Chapter 2, pp. 15–42). Cheltenham, UK: Edward Elgar Publishing. https://pureportal.strath.ac.uk/en/publications/macroeconomic-forecasting-using-bvars

8. Wozniak, T. (2024). bsvars: A Package for Bayesian Structural Vector Autoregressions in R. R package version 1.0. https://CRAN.R-project.org/package=bsvars

9. Lütkepohl, H., Shang, C., & Wozniak, T. (2024). Bayesian Structural VARs with Non-Normality and Heteroskedasticity. Working Paper.

10. Krüger, F. (2015). bvarsv: Bayesian Analysis of a Time-Varying Parameter Vector Autoregressive Model with Stochastic Volatility. R package version 1.0. https://CRAN.R-project.org/package=bvarsv

11. Primiceri, G. E. (2005). Time-varying structural vector autoregressions and monetary policy. The Review of Economic Studies, 72(3), 821–852. https://doi.org/10.1111/j.1467-937X.2005.00353.x

12. Kuschnig, N., & Vashold, L. (2021). BVAR: Bayesian Vector Autoregressive Models. R package version 1.2. https://CRAN.R-project.org/package=BVAR

13. Chan, J. C., Koop, G., Poirier, D. J., & Tobias, J. L. (2019). Bayesian Econometric Methods. *Cambridge University Press.* https://assets.cambridge.org/97811084/23380/frontmatter/9781108423380_frontmatter.pdf

14. Koop, G., & Korobilis, D. (2010). Bayesian multivariate time series methods for empirical macroeconomics. *Foundations and Trends in Econometrics*, 3(4), 267–358. https://doi.org/10.1561/0800000013

15. Quilis, E. M. (2022). BayVAR_R: Classical and Bayesian VAR Models. R package version 1.0. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4000589

16. McCracken, M. W., & Ng, S. (2016). FRED-MD: A monthly database for macroeconomic research. *Journal of Business & Economic Statistics*, 34(4), 574–589. https://doi.org/10.1080/07350015.2015.1086655

Appendix

A1. Description of monthly macro and financial variables

Series Name	Units	Transformation	isFinancial	Prior
Industrial Production Index	Index 2017=100	100×log	0	RW
Capacity Utilization: Manufacturing	Percent of Capacity	Raw	0	RW
Housing Starts	Thousands of Units	100×log	0	RW
Real Personal Income ex. Transfer Receipts	Billions of Chained 2012 Dollars	100×log	0	RW
Real Personal Consumption Expenditure	Index 2017=100	100×log	0	RW
Real Manufacturing and Trade Industries Sales	Millions of Chained 2017 Dollars	100×log	0	RW
All Employees, Total Nonfarm	Thousands of Persons	100×log	0	RW
Civilian Unemployment Rate	Percent	Raw	0	RW
Average Hourly Earnings of Production and Non-supervisory Employees	Dollars per Hour, Monthly	100×log	0	RW
Initial Claims		100×log	0	RW
CPI: All Items	Index 1982–1984=100	100×log	0	RW
CPI-Urban: All Items Less Food and Energy	Index	100×log	0	RW
PCE: Chain-Type Price Index	Index 2017=100	100×log	0	RW
PCE Services: Chain-Type Price Index	Index	100×log	0	RW
PPI by Commodity: Metals and Metal Products	Index	100×log	1	RW
Crude Oil, spliced WTI and Cushing	Dollars per Barrel	100×log	1	RW
10-Year Treasury Note Yield	Percent	Raw	1	RW
1 Year Treasury Bond Yield	Percent	Raw	1	RW
5 Year Treasury Bond Yield	Percent	Raw	1	RW
Federal Funds Rate	Percent	Raw	0	RW
Moody Seasoned Aaa Corporate Bond Yield	Percent	Raw	1	RW
Moody Seasoned Baa Corporate Bond Yield	Percent	Raw	1	RW
M2 Money Stock	Billions of Dollars	100×log	1	RW
S&P 500 Index	Index	100×log	1	RW
CBOE Volatility Index: VIX	Index	100×log	1	WN
Japan / U.S. Foreign Exchange Rate		100×log	1	RW
U.S. / U.K. Foreign Exchange Rate		100×log	1	RW
Canada / U.S. Foreign Exchange Rate		100×log	1	RW

A2. Description of quarterly macro and financial variables

Series Name	Units	Transformation	isFinancial	Prior
Real Gross Domestic Product	Billions of Chained 2017 Dollars	100×log	0	RW
Real Personal Consumption Expenditures	Billions of Chained 2017 Dollars	100×log	0	RW
Real Disposable Personal Income	Billions of Chained 2017 Dollars	100×log	0	RW
Real Private Residential Fixed Investment	Billions of Chained 2017 Dollars	100×log	0	RW
Real Private Non-Residential Fixed Investment	Billions of Chained 2017 Dollars	100×log	0	RW
Real Government Consumption Expenditures and Gross Investment	Billions of Chained 2017 Dollars	100×log	0	RW
Industrial Production Index	Index 2017=100	100×log	0	RW
Capacity Utilization: Manufacturing	Percent of Capacity	Raw	0	RW
Housing Starts	Thousands of Units	100×log	0	RW
All Employees, Total Nonfarm	Thousands of Persons	100×log	0	RW
Civilian Unemployment Rate	Percent	Raw	0	RW
Business Sector: Real Compensation Per Hour	Index 2017=100	100×log	0	RW
GDP Deflator	Index 2017=100	100×log	0	RW
PCE: Chain-Type Price Index	Index 2017=100	100×log	0	RW
PCE Excluding Food and Energy	Index 2017=100	100×log	0	RW
CPI: All Items	Index 1982–1984=100	100×log	0	RW
CPI-Urban: All Items Less Food and Energy	Index	100×log	0	RW
Crude Oil, spliced WTI and Cushing	Dollars per Barrel	100×log	1	RW
10-Year Treasury Note Yield	Percent	Raw	1	RW
1-Year Treasury Bond Yield	Percent	Raw	1	RW
5-Year Treasury Bond Yield	Percent	Raw	1	RW
Moody Seasoned Aaa Corporate Bond Yield	Percent	Raw	1	RW
Moody Seasoned Baa Corporate Bond Yield	Percent	Raw	1	RW
Real Exports of Goods and Services	Billions of Chained 2017 Dollars	100×log	0	RW
Real Imports of Goods and Services	Billions of Chained 2017 Dollars	100×log	0	RW
S&P 500 Index	Index	100×log	1	RW
CBOE Volatility Index: VIX	Index	100×log	1	WN
University of Michigan: Consumer Sentiment	Index 1st Quarter 1966=100	100×log	0	RW

A3. Entropic Tilting: Conditional Forecasts of Quarterly Data Anchored to Long Term Expectations

Section 6 introduced the framework of deploying entropic tilting to anchor the unconditional forecasts to long-term targets as defined in the Summary of economic Projections. Though I use monthly data in that example, we can apply the same method on quarterly data as well. This is crucial for mainly two reasons. Firstly, the FOMC reports projections in quarterly or annual terms, particularly for longer horizons spanning more than one year into the future. Using quarterly forecasts data ensures that the forecasts align with the policy horizon and the frequency at which the Fed communicates its targets. Secondly, it allows us to capture the most important metric of economic activity, real GDP, which is inherently a quarterly measure. Thirdly, monthly data can be volatile due to seasonal effects, measurement errors or short-term shocks, breeding noise that can distort long-run trends. Alternatively, quarterly data aggregates monthly observations, smooths out short-term fluctuations and provides a clearer signal of the underlying economic trends. This is advantageous if we condition on long-run targets where the emphasis is to capture persistent trends rather than transient movements. Therefore, *entropicTilting* file exemplifies how to generate the forecasts given the long-term targets and evaluate the joint predictive "tilted" distributions of such forecasts.

```
# Conditioning assumptions: SEP released on Dec 2024 SEP for 2027
# Center of the central tendency: midpoint of the range
```

```
# find the index of the PCE inflation rate, Federal Funds Rate, and Unemployment Rate in
    the Spec DataFrame
idxCVPCE = Spec[Spec['SeriesName'] == 'PCE: Chain-Type Price Index'].index[0]
valCVPCE = (2 + 2) / 2

# Core PCE
idxCVPCEcore = Spec[Spec['SeriesName'] == 'PCE Excluding Food and Energy'].index[0]
valCVPCEcore = (2 + 2) / 2

# Unemployment Rate
idxCVLR = Spec[Spec['SeriesName'] == 'Civilian Unemployment Rate'].index[0]
valCVLR = (4 + 4.4) / 2

# Real GDP
idxCVGDP = Spec[Spec['SeriesName'] == 'Real Gross Domestic Product'].index[0]
valCVGDP = (1.8 + 2) / 2
```

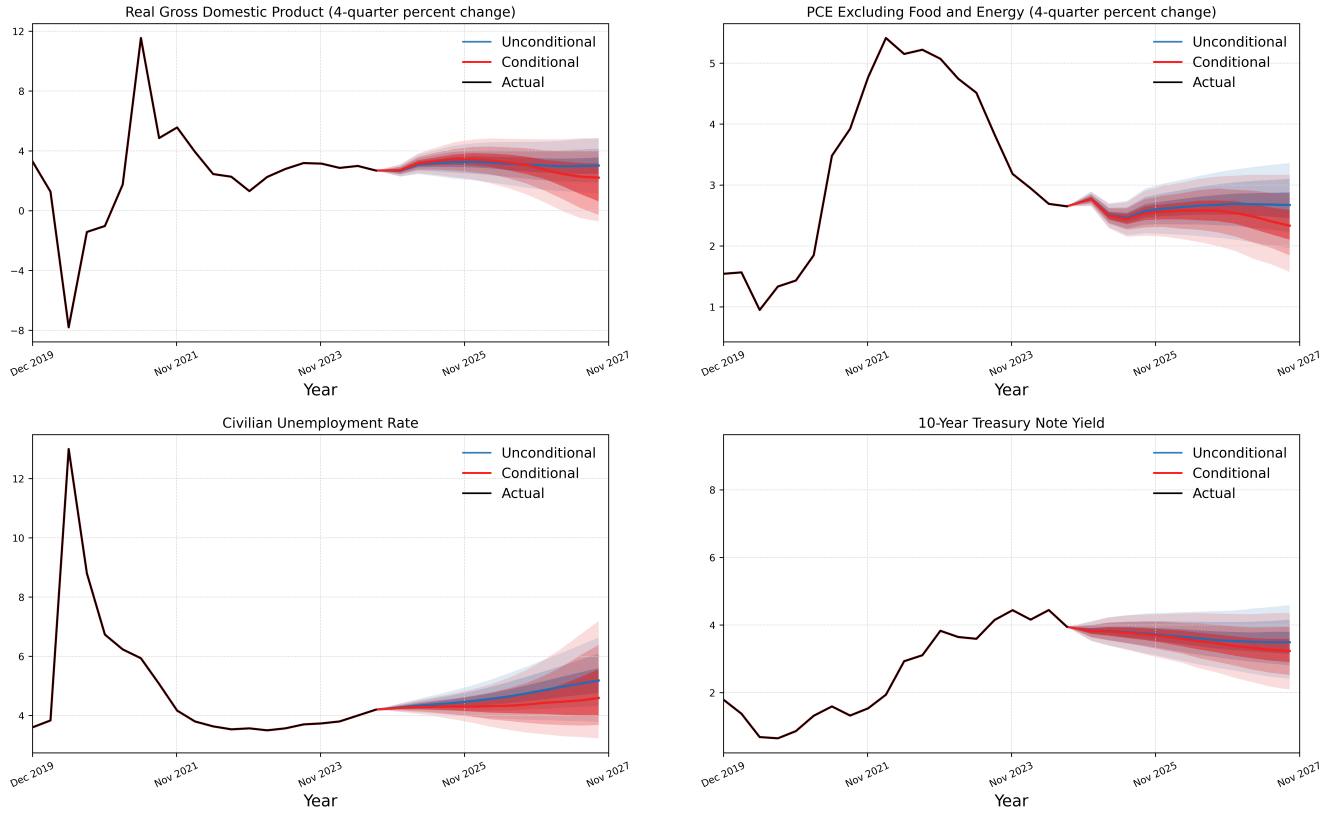

Figure A1: Unconditional forecasts and forecasts conditional on SEP projections of PCE inflation and core PCE inflation rate to 2 percent, unemployment rate to 4.2 percent, and real GDP growth rate to 1.9 percent 3 years into the future. These conditional forecasts are produced after tilting the forecast distribution such that the median of the forecast distribution anchors to the target SEP projections.

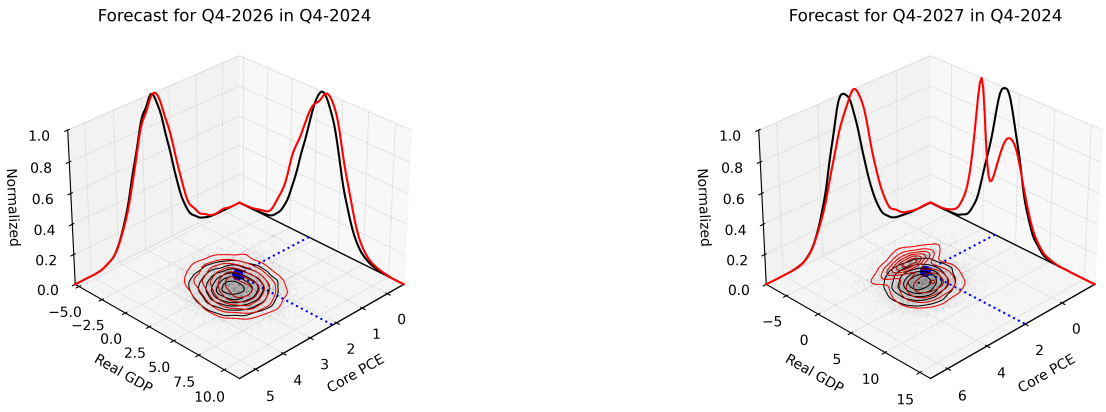

Figure A2: Joint forecast density functions of the forecast of the YoY growth rate of real GDP and PCE inflation for Q4-2026 (left), and Q4-2027 (right). The black lines are the unconditional (baseline) forecast distribution, where the contous in the centers represent the joint distribution, and those on the side panes are the marginal distributions. The red curves are the "tilted" distributions obtained after adjusting the original forecast distribution to anchor to the long-run targets - these are the average of the central tendency values present in the Summary of Economic Projections released on December 18, 2024. The blue dot in the surfaces are the midpoint of the central tendency values: real GDP growth and PCE inflation rate are 2 and 2.15 percent, respectively in 2026. Likewise, they are 1.9 and 2 percent, respectively in 2027.