Improving Macroeconomic Forecasting by Accounting for Parameter Instability in Large-N Factor Models

Bachelor Thesis

Louis Lejeune (434454)

Supervisor: Annika Schnücker | Co-reader: Dick van Dijk

Double Bachelor Econometrics & Economics

Erasmus School of Economics – Erasmus University Rotterdam

July 2019

Abstract

Dimensionality reduction techniques have been widely researched in the literature and applied in a diversity of contexts. Dynamic approaches to specific Principal Components (PC) and Partial Least Squares (PLS) models have been developed to better exploit the properties of time series data. However, the time invariance of the loadings is continuously challenged in macroeconomic frameworks dealing with large sets of variables. In this paper, we test for the significance of this time instability modelled as a structural break and two-state Markov regime switches, and we design extended PC and PLS models that account for such non-constancy of loadings in the factor construction. Our results support existing findings on the presence of time variance and bring evidence of improved forecasting potential, particularly in the short term with Markov regime-switching models and over longer time horizons through the inclusion of a structural break.

Contents

1 Introduction

Perpetual improvements in the availability and accessibility of vast amounts of data continuously give rise to new modelling and forecasting opportunities. Besides enlarging the spectrum of applications covered, that has led to increased interest for and potential of the use of econometric models to explain complex relationships. In particular, the use of high-dimensional time series data has received considerable attention in the context of macroeconomics. Besides extensive literature on the connections between large numbers of variables and their algorithmic integration into single models, growing research focuses on exploiting the aggregate information contained in such dataset by summarising it.

In this context as well as in many others, factor models are a popular choice for capturing this information into a few unobservable, or *latent*, variables referred to as factors. Such models, in their simplest static form, find applications ranging from economics, to chemistry, and even to more advanced emerging fields such as image recognition (see, e.g., Haaland & Thomas, 1988; Geladi, Isaksson, Lindqvist, Wold, & Esbensen, 1989). Advancements in econometric techniques allow for the dynamic characteristics of variables to be modelled in the construction of factors, particularly relevant to economic time series. Stock and Watson (2010) provide a comprehensive review of the different dynamic factor models and their implementation, spanning both univariate and multivariate analyses. The models, widely applied in the literature, centre upon the idea of exploiting the dynamic properties of the time series variables to be summarised. Indeed, static factor models neglect the importance of the lagged factors in the estimation. Consequently, correlation across time can be observed in the residual component; in particular, an autoregressive (AR) process is found to remain in the errors as a result of this misspecification. If the latent factors evolve through a time-series process, it follows that they can indirectly be explained by lagged variables themselves. The inclusion of lags of the cross-section circumvents the issue cause by static estimation in dynamic factor models.

However, little literature focuses on empirical applications of extended factor models meant to incorporate parameter instability in the construction of factors themselves. Many procedures are designed and studied theoretically to test for time instability in a diversity of ways, and in fact are found to be firmly relevant to many economic analyses, including inflation (Stock, 2003). The breakdown of static models, anticipated as a consequence of these findings, is also vastly explored (Stock & Watson, 2009). Nonetheless, despite the existence and general validity of the instability being established, extended models that account for it are much more scarcely estimated, computational difficulty being one of the main challenges encountered (Chow, Zu, J., Shifren, & Zhang, 2011).

In this paper, we take an empirical approach and both test for and model the potential parameter instability of factors to forecast inflation over time. We focus on two varieties of time instability: structural breaks, and more continuous time variance through regimes. The basic factor models extended in this analysis are chosen to rely on Principal Components (PC) and Partial Least Squares (PLS) analysis. The central research question is thus formulated as follows:

To what extent can inflation forecasts derived from PC and PLS factor models be improved by accounting for parameter instability?

In answering this question, differences arising between forecasts across extensions, are also expected to shed light on how the time instability of the inflation process can be characterised

The forecasting performance of all factor models is compared across static, dynamic and extended implementations. The factor estimation in PLS models, central to this paper, is based on a *supervised* dimensionality reduction technique. Whereas in alternative, unsupervised methods, the factors are constructed

independently of the variable to forecast and based solely on the common variation contained in the information set, PLS analysis is best suited to our purpose as it attempts to summarise the co-movement of the data with the target variable (Maitra & Yan, 2008). One such alternative factor estimation technique is PC, a very common and relatively simple dimensionality reduction technique that is used as a reference for the performance of PLS models.

Static versions of both types of factor estimation methods are implemented. In addition, the PLS analysis is further explored with a dynamic approach. As mentioned earlier, integrating the time series properties of the variables modelled into the estimation factors is greatly relevant in this context. In the dynamic PLS models, the information set is therefore enlarged with lagged variables. This is done in the aim of capturing the autocorrelation of variables in the factors, while in the alternative case of static models, this is modelled separately in the forecasting model beside selected factors. For all versions of the basic models, the work of Bai and Ng (2008) and Fuentes, Poncela, and Rodriguez (2015) is taken as guidance.

All of the models are then extended to allow for parameter instability. The first extension consists in testing for structural breaks according to the work of Han and Inoue (2015). The procedure considers a break in the construction of factors in aggregate – rather than their relationship with specific variables of the large information set only – and as such is applied individually to each single estimated basic model (for separate estimation samples) as the test requires the use of factors obtained through basic models. Unlike with many other existing tests, no candidate breakpoint location must be specified. The models are re-estimated by accounting for the relevant instability if found to be significant using adjusted critical values derived by Andrews (1993). Second, a more complex and flexible extension is explored to allow for a series of cyclical breaks in the parameters; specifically, the factors are estimated according to separate processes in a two-state space defined by a Markov regime-switching model. The approach employed relies on the method elaborated by Liu and Chen (2016) and is applied to each chosen estimation sample. The convergence of the algorithm depends on the significance of the differences between the alternating regimes and the two factor processes they define.

The relevance of this paper is threefold. First, as suggested earlier, the analysis fills an apparent gap in the academic literature by providing empirical support for the usefulness of some methods whose theoretical aspects have been studied at large. Second, with the availability of increasingly more econometric techniques, it serves a purpose in determining whether one particular approach brings additional predictive power to existing models. The two extensions considered can be seen to incorporate time instability in a similar way: allowing for a structural break as well as for regime switches ultimately comes down to introducing a particular weighing of the time dimension in the factor estimation; in the same way, both can be seen to rely on a two-state space, where the latter extension enables switches in cycle while the former restricts the regimes to form a single chronological sequence. The relevance of this single idea is then put to the test with two different perspectives. Finally, the paper also presents practical, economic relevance through its forecasting scope. Concretely, the methods explored here have endless applicability; structural breaks serve to segregate useful information from all past, outdated one with different economic functioning in any analysis conducted and, in fact, two-state Markov regime-switching models are continuously being used by the Bank of Canada (Van Norden & Vigfusson, 1996).

The main findings made in this paper are the following. First, our results support conclusions drawn in earlier research regarding the existence of time variation in the the loadings of PC and PLS factor models applied in this large-N macroeconomic context. We further find evidence for the relevance of incorporating this non-constancy into the models by obtaining improved out-of-sample performance as a result. Most

notably, we suggest that the structural break and Markov regime-switching extensions complement each other; the former best captures the long-term factor structure of the inflation process, whereas the latter drastically improves the forecasting performance over shorter time horizons.

The structure of this paper is a follows. First, a review of existing literature relevant to the study is provided in the next section. In Section 3, the data selection and transformations are explained in detail. The methodology followed to build the various basic and extended models is then developed in Section 4, after which all results are presented and discussed extensively. Finally, the key findings are summarised and notable conclusions are drawn from the analysis in the last section.

2 Literature Review

A review of research already conducted around five topics greatly relevant to this study is presented here, namely factor models, PC and PLS analyses, dynamic improvements to them, structural breaks and Markov regime-switching models.

First, the possible and relevant applications of factor models are numerous. Indeed, they present two paramount advantages. On the one hand, they allow to capture an important part of the information contained wide datasets into a few constructed variables. That is particularly relevant in this macroeconomic framework where the number of series relevant to incorporate into models can grow very large and easily exceed that of time series observations. With factor models, Stock and Watson (2010) find that relationships can still be estimated simultaneously and consistently. On the other hand, the latency of the factors is of special interest, especially in this context where factors can then be seen as information-dense unobservable variables, which Stock and Watson (2002b) refer to as diffusion indices. These are in fact perpetually investigated by NBER[1] business cycle analysts for their potential as indicators.

In particular, PC analysis is by far one of the most widely used and investigated techniques developed for factor estimation. Driving reasons for this prevalence are the established performance of the method, the relative simplicity as well as its general robustness. Many empirical studies find the forecasting performance of such factor models to surpass that of various others such as univariate autoregressions, small vector autoregressions, and leading indicator models, and even more extensive models used in practice by governments (Matheson, 2006; Stock & Watson, 2002b). In addition, the work of De Mol, Giannonea, and Reichlin (2008) and Matteson and Tsay (2011) also show that complex Bayesian shrinkage and orthogonal components methods do not lead to much improvement over standard PC models, while requiring far more stringent regularity conditions. Most importantly, Stock and Watson (2002a) find that estimated factors are consistent even in the presence of time variation in the factor model and that the implied forecasts are asymptotically efficient.

Besides, as an alternative to PC analysis, factor estimation based on PLS, first introduced by Wold (1966), offers promising prospects in a forecasting framework. While arguing that the resulting factor decomposition is not optimal on mathematical and statistical grounds, Chin (1995) also supports the performance of the analysis for practical uses such as forecasting. PLS analysis is part of a class of supervised dimensionality reduction techniques, which have also widely been applied in the literature. Another such technique is Supervised Principal Components, in which only a portion of variables is selected to construct factors depending on the significance of their relationships with the target variable (Bair, Hastie, Paul, & Tibshirani, 2006). The approach therefore requires the setting of a certain threshold. In comparison, PLS has the advantages of being non-parametric and having a level of simplicity closely similar to PC analysis.

[1] National Bureau of Economic Research

Note that in line with this, a variety of methods have been designed to refine the number of relevant variables in the information set used to construct factors. Bai and Ng (2008) investigate in detail the ultimate forecasting performance of diverse well-documented procedures in which standard factor estimation is supplemented by variable selection in the same context as ours. Imposing sparsity has been found to lead to significant improvements of forecasts, most notably so in association with PLS analysis shown by Fuentes et al. (2015). However, while it may well be combined with the extensions presented, the variable selection angle is entirely disregarded in this paper as not directly relevant to the aim of the research.

A dynamic approach to the basic PC and PLS factor models employed is greatly relevant when dealing with time series variables. Dynamic factor models have originally been proposed by Geweke (1977). Sargent and Sims (1977) show that an important fraction of the variance of key U.S. quarterly macroeconomic variables such as output, employment, and prices can be explained by only a couple of dynamic factors. The work of Giannone, Reichlin, and Sala (2004) supports these early empirical findings and altogether form evidence of the relevance of dynamic factor models in macroeconomics. Forni, Hallin, Lippi, and Reichlin (2000) study the theoretical aspects of such models, providing conditions for identification and consistency.

The focus on accounting for time instability with two extensions also appears to be very relevant in view of existing literature. Indeed, the significance of the variation of factor loadings over time has been established in a diversity of settings. The seemingly improved specification of the capital asset pricing model (CAPM) through the inclusion of time-varying betas is a notable example (Jagannathan & Wang, 1996). In the same way, Stock and Watson (1996) find evidence of the prevalence of instability in postwar macroeconomic time series. For this reason, Bates, Plagborg-Møller, Stock, and Watson (2013), among others, model the factor loadings according to a dynamic equation with stochastic processes of various kinds in the errors to conduct their analysis. More prominently, modelling the time variation through sharp, one-off breaks in the series, both in one sequence and in cycles, has been widely found to be crucial in the macroeconomic forecasting framework at hand, instead of through such small oscillations – and drift – or other smooth changes over time (see Guidolin, 2016 for examples thereof).

Regarding structural breaks, countless studies focusing on the theoretical implications of ignoring this time instability, on the implementation of tests as well as on empirical findings thereof exist. Assessing the reliability of forecasts in the face of structural changes, Stock and Watson (2009) find that sufficient independence of the variation across series is necessary for the factor models not to break down – which, in the context of macroeconomic time series, is not a condition that most surely holds. In particular, Breitung and Eickmeier (2011) show that the presence of unaccounted breaks severely inflate the number of factors identified. In reaction to these findings, Bates et al. (2013) further characterise the type and magnitude of the temporal instabilities required for consistency.

Plenty of tests have since been designed. Many of them have a relatively narrow scope: some solely focus on specific individual factor loadings (Yamamoto & Tanaka, 2015); others require the setting of select breakdates to test. Among them, the test elaborated by Breitung and Eickmeier (2011) is based on the sample average of the Chow test statistics for each of the factor loadings. However, the simplicity of the implementation comes at the cost of both imposing the restrictive assumption of cross-sectional independence of the errors on the factor model – in contrast with the more flexible, approximate factor setup of Bai and Ng (2002) followed in this study – and lacking consistency in some special cases. To circumvent these limitations, Chen, Dolado, and Gonzalo (2014) propose a test that relies upon the search for parameter breaks in a regression of one estimated factor on the remaining ones, therefore only applicable to cases where a multitude is used. Instead, we here choose to resort to the test developed by Han and Inoue (2015),

in which the joint null hypothesis of time-invariant factor loadings is tested against the alternative that a nonnegligible fraction of them is not constant. As we allow for the breakdate to be unspecified, modified critical values uncovered by Andrews (1993) are used.

Key findings highlight the relevance of considering structural changes: Breitung and Eickmeier (2011) find evidence of breaks in over half of the series in 1984, associated with the Great Moderation in the U.S. and elsewhere. More vaguely, Ma and Su (2018) also detect a the handful of breaks in U.S. macroeconomic time series over the second half of the 20^{th} century. These results reflect changes in the structure of modern economies such as reduced credit constraints among consumers, changes in production technologies, or changes in (monetary) policy regimes (Stock, 2003).

Burns and Mitchell (1946) characterise non-linearity as a defining characteristic of the evolution of the business cycle through regime switches at turning points. Diebold and Rudebusch (1996) suggest the integration of the Markov regime-switching model developed by Hamilton (1989) into dynamic factor models in which the non-linearity is thereby tentatively encompassed. Such models have been found to be relevant in a diversity of economic time series analyses. For example, Kim and Nelson (1998) and Lo and Piger (2005) apply them to economic indicators of the state of the cycle and to the effectiveness of monetary policies respectively, yielding interesting economic insight regarding the significance of two expansionary and recessionary states.

Markov regime-switching models are therefore investigated in the closely related framework of inflation forecasting. A variety of implementations have been studied, each presenting their own computational difficulties. A quasi-maximum likelihood estimation procedure is provided in detail by Kuan (2002), for which a factor model must be specified with correct distribution of the errors. However, the absence of studies on the consequences of such choices makes it hard to apply it in this context. Alternatively, Guérin, Leiva-Leon, and Marcellino (2018) develop a new method referred to as Markov-switching three-pass regression filter, which incorporates regimes of loadings directly into the factor estimation. Mere tests of the regimes switches also exist, relying for instance on a quasi-likelihood ratio statistic of transition probabilities in different samples or on the sensitivity of posterior probabilities to the prior with a Bayesian approach. In this paper, we focus on building factor models in conjunction with a two-state space defined by a Markov regime-switching model as designed by Liu and Chen (2016) and adapting it to our setup. The method uses an iterative approach to estimate the loading space in either regime and categorising the observations through a Viterbi alogrithm (see Section 4.3 for details). The clustering obtained then allows for two separate factor processes to be modelled.

Finally, we note that many other possible extensions exist beside structural breaks and Markov regime-switching models to account for time instability of the factor loadings. Relevant examples include threshold models, smooth transition models and Kalman filtering. However, these models are not explored for particular reasons. Threshold models require key assumptions about the driving variables behind the regime switches among a large cross-section, whereas these forces are latent in Markov models. In fact, threshold models can be seen as a subcase where states are directly observable. As for smooth transition models and Kalman filtering, the nature of the time variation imposed is fundamentally different from the one-off breaks in the series defined by our two extensions as the underlying state space is continuous. We here limit ourselves to less flexible methods investigating the existence of two – sequential or cyclical – regimes that are mutually exclusive, also in the interest of identification in the application at hand.

3 Data

For the complete analysis presented in this paper, a dataset filled with 132 monthly U.S. macroeconomic time series is used. The data, put together and elaborated on by Stock and Watson (2005), have been repeatedly used in the literature because of their established quality and reliability.[2] Given the uniform use of all variables, only the fully balanced sample is considered, covering all periods from January 1960 to December 2003 and resulting in a total of 528 observed time periods. Data manipulations are performed following the authors' suggestions and as replicated in Bai and Ng (2008); indeed, many macroeconomic variables exhibit non-stationarity of various orders and therefore need to be transformed adequately for models to yield interpretable economic insight.

The information set is always taken as a whole as the basis for factor construction. Among the data series, the growth of the Consumer Price Index for All Urban Consumers, all items (CPI-I:all) is the most frequently used statistic as representative of inflation over time and is thus picked as the target variable. In particular, the inflation is assumed to be integrated of order 2, hence the following forms are at the centre of this study:

$$y_{t+h} = \frac{1200}{h}(x_{t+h} - x_t) - 1200(x_t - x_{t-1}) \tag{1}$$

$$z_t = 1200(x_t - x_{t-1}) - 1200(x_{t-1} - x_{t-2}) \tag{2}$$

where x_t and y_{t+h} represent the natural logarithm of the index and the target variable to forecast with horizon h respectively, and z_t is a known instance of the target variable to be used as regressor in the forecasting equation or in the factor estimation (see Section 4.1).

Given the length of the period covered, we conduct our research on a variety of subsamples. The interest of doing so has both an economic and an econometric aspect: inconsistencies arising between different times and lengths of forecast samples can shed light on the time-changing characteristics of the inflation process, and discrepancies in the relevance of the extensions considered can highlight the importance of the size of the estimation sample. We consider the same seven combinations of estimation and forecast subsamples as defined by Bai and Ng (2008) and replicated by Fuentes et al. (2015) with one addition to it, as displayed in Table 1.

Table 1: Estimation and forecast subsamples

	Estimation subsample	Forecast subsample
M1	1960:03 to 1970:03-h	1970:03 to 1980:12
M2	1960:03 to 1980:03-h	1980:03 to 1990:12
M3	1960:03 to 1990:03-h	1990:03 to 2000:12
M4	1960:03 to 1970:03-h	1970:03 to 1990:12
M5	1960:03 to 1970:03-h	1970:03 to 2000:12
M6	1960:03 to 1980:03-h	1980:03 to 2000:12
M7	1960:03 to 1990:03-h	1990:03 to 2003:12
M8	1970:03 to 1990:03-h	1990:03 to 2000:12

For reasons to be elaborated on in Sections 4.2 and 4.3, not all subsamples are relevant to the two types of extended models presented. Specifically, the settings where the estimation sample is the smallest, i.e.

[2]The data can be retrieved from Mark Watson's website: `http://www.princeton.edu/~mwatson/wp.html`

those consisting of 10 years of data, are disregarded, leaving the analysis to focus on five pairs of subsamples.

4 Methodology

Section 4.1 first introduces the broad concept of factor models and the chosen forecasting model, as well as how these function as a system; a detailed description of different implementations of PC and PLS models is presented in Subsections 4.1.1 and 4.1.2. Next, the test for and models based on a structural break and Markov regime switches are dealt with in the two respective following sections.

4.1 Factor Models and Forecasting Framework

Widely used in the context of large sets of explanatory variables, factor models rely on the assumption that a group of N variables can be summarised by a small number of constructed latent factors. The residual information contained in the original variables is then captured by a second, idiosyncratic component, assumed to drive specific variable shocks beyond the common movement of the series. The following representation introduces the relevant notation at time $t \in \{1, \dots, T\}$:

$$\mathbf{x}_t = \mathbf{f}_t \mathbf{\Lambda} + \boldsymbol{\epsilon}_t \tag{3}$$

where $\mathbf{x}_t$ is an N-dimensional row vector of variables summarised by a set of $k < N$ factors $\mathbf{f}_t = [f_{t1} f_{t2} \dots f_{tk}]$, $\boldsymbol{\epsilon}_t$ is a vector of disturbances, and $\mathbf{\Lambda}$ is a $(k \times N)$-matrix of factor loadings. More specifically, the row vectors $\boldsymbol{\lambda}_1', \boldsymbol{\lambda}_2', \dots \boldsymbol{\lambda}_k'$ of $\mathbf{\Lambda}$ contain weights assigned to each of the variables in the cross-section and are used to construct each of the k factors. In this setting, both the common and idiosyncratic components are assumed to be orthogonal across all variables and time periods, i.e. $E(\mathbf{f}_t \epsilon_{si}) = 0$ for all $i \in \{1, \dots, N\}$ and $t, s \in \{1, \dots, T\}$. Note that in approximate factor models, unlike in strict ones, the error terms are allowed to exhibit weak cross-correlation, and have been shown not to interfere with the consistency of factor estimates (Doz, Giannone, & Riechlin, 2006).

PC and PLS analysis are used as basic factor models throughout the paper. The former mostly serves as a reference point for the performance of all other models to be investigated, while the latter lies at the centre of the study as it fits the goal of macroeconomic forecasting best.

At each time $t \in \{1, \dots, T\}$ of any forecasting sample, estimated factors $\hat{\mathbf{f}}_t$ are obtained and are then used to forecast the target variable of choice. In each iteration, all available information serves to derive the factor loadings and imply updated factors. In other words, as time advances, the estimation sample grows to include past periods, resulting in a model with expanding window. The forecasting model in question, in its most general form, is defined as follows:

$$y_{t+h} = \mu + \hat{\mathbf{f}}_t \boldsymbol{\beta}(L) + z_t \boldsymbol{\phi}(L) + u_{t+h} \tag{4}$$

where y_{t+h} represents the target variable forecast with time horizon h, and L is the lag operator that introduces lags[3] of each explanatory variable into the equation. Apart from a particular implementation where the forecasting model is modified (see Subsection 4.1.2), the number of lags of both the factors and the target variable is set according to the Bayesian Information Criterion (BIC) up to maximum of six where combinations leading to identification issues are disregarded, following standard choices made by Bai and Ng (2008) and Fuentes et al. (2015). This number is allowed to fluctuate over time as well as across the

[3]It should be noted that throughout this paper, lags refer to all instances of a certain variable; in other words, any number of lags mentioned regarding regressors in an equation such as 4 includes the variable at time t itself.

two types of regressors to allow for maximum flexibility. However, it is fixed to a single value across the different factors (in case they are more than one) – otherwise, the number of regressions to estimate would grow exponentially with the number of factors with no apparent added value.

The basic PC and PLS models do not take any kind of time instability into account as the structure of the estimation – as made explicit in Subsections 4.1.1 and 4.1.2 – remains constant over time. That will only change with the features introduced in the next sections.

4.1.1 Principal Components

To summarise the information contained in a large dataset, PC analysis seeks to account for as much common movement as possible into a few factors. In other words, it has as an objective to maximise the variance the factors account for in the cross-section. To achieve this, the estimated factors loadings $\hat{\mathbf{\Lambda}}_t'$ are derived from the eigenvalue decomposition of the matrix $\mathbf{X}_{1:t}'\mathbf{X}_{1:t}$, where $\mathbf{X}_{1:t}$ is the $(t \times N)$-matrix of times series data (equivalent to $\mathbf{X}$ where the last $T - t$ rows of unavailable information at time $t \in \{1, \dots, T\}$ are cut off). The matrix of factors loadings is then constructed by assembling the k eigenvectors, corresponding to the k largest eigenvalues (sorted in decreasing order), as its columns $\boldsymbol{\lambda}_{t1}, \boldsymbol{\lambda}_{t2}, \dots \boldsymbol{\lambda}_{tk}$.

Details of the exact procedure followed are chosen to be as similar as possible to those found in Bai and Ng (2008) to allow for maximum referability of the results derived from this basic model. The number of factors k is selected ex-post for the entire model so as to maximise its predictive ability, which in this case comes down to minimising the Mean Squared Error (MSE; see Section 5 for details on the used metrics), up to a maximum of 10.

Also, each of the predictor variables in $\mathbf{X}_t$ must be standardised. That is common practice in PC analysis to prevent any difference in the means and scales across variables to affect the factor estimation; in other words, for the model to capture the most cross-correlation contained in the information set, as opposed to covariance. As factors are estimated in each iteration, the standardisation is applied column-wise to $\mathbf{X}_t$ before each eigenvalue decomposition is performed.

Note that the simplicity of the procedure for PC analysis comes at a cost. Indeed, while we here deal with time series data, this method ignores the time dimension in its estimation. As a result, no properties of the time series is exploited, and the model can therefore be seen as static. The advantage of such models, however, is that they are found to be quite robust to misspecification, as can be expected given the limited number of parameters involved (Boivin & Ng, 2005).

4.1.2 Partial Least Squares

The dimensionality reduction technique central to this paper is PLS. This type of analysis differs from the former in the way the estimated factor loadings $\hat{\mathbf{\Lambda}}_t'$ are obtained. While PC models completely disregards the forecasting objective in estimating the factors, PLS requires the use of the target variable. As such, the method seeks to capture as much of the relevant variance contained in the information set; in other words, the most co-movement between the predictor variables and the target.

To do so, the PLS technique relies on the same principle as PC analysis does, but the loadings are rather derived from the eigenvalue decomposition of the matrix $\mathbf{M}_t = \mathbf{X}_{1:t-h}'\mathbf{Y}_{h+1:t}\mathbf{Y}_{h+1:t}'\mathbf{X}_{1:t-h}$. In line with notation already introduced, $\mathbf{X}_{1:t-h}$ is a $(t - h \times N)$-matrix of panel data, and $\mathbf{Y}_{h+1:t}$ is a $(t - h)$-vector whose entries contain the target variable to be forecast h periods ahead, such that corresponding rows of the two variables are linked through the forecasting framework. Altogether, the two variables span all information available at time t, and the eigenvalue decomposition is thus based on the maximum amount of

data. Note that the subscript t of the matrix $\mathbf{M}$ denotes the number of time periods used to construct it as all observations from the start of the estimation sample to the current period are made use of during the iteration.

Note that unlike PC, this technique involves an iterative process to find the k factors. Indeed, the factor loadings $\hat{\lambda}_1$, used to estimate the first factor, are retrieved from the decomposition of $\mathbf{M}_t$ as the eigenvector corresponding to its largest eigenvalue. To proceed to the estimation of the remaining factors, two algorithms exist: the NIPALS algorithm, in which variables are deflated based on a linear combination of the predictor variables (Maitra & Yan, 2008); and another, more complex algorithm that relies on the use of regressions. In this context, the latter algorithm is selected, in part because of its more intuitive interpretability. Besides, details of this procedure, as well as of the implementation of PLS more broadly outlined in this subsection, are chosen according to the work of Fuentes et al. (2015). The reasons behind this choice are again the referability of the results derived from basic models, but also the quality of the models set up in this macroeconomic forecasting context.

The iterative algorithm used works as follows. In each iteration, the obtained factor loadings $\hat{\lambda}_1$ are used to derive the first factor, to be used as a regressor observation later in the forecasting equation 4. Forming a vector together with previously derived equivalents, the first factors are used in two regressions: one of $\mathbf{X}_{1:t}$ on the full vector derived, and another of $\mathbf{Y}_{h+1:t}$ on the first $1 : t - h$ elements of the vector, as regressing its full version while preserving the correspondence of rows between target and predictor variables would otherwise require the use of out-of-sample observations of the target variable. As mentioned earlier, in all steps, as little data available as possible is left out in order for the estimation to be the most accurate and up-to-date in each iteration. Next, the matrix $\mathbf{M}_t$ is updated using the residuals of the two regressions (where again, the residuals of the first regression must be cut off for the dimensions to match) and serves for the second eigenvalue decomposition to be performed and the vector of second PLS factors to be derived. The process is then applied repeatedly for the extraction of all remaining factors. In this way, each additional factor is constructed so as to capture the most co-movement between the information and the target variable yet unexplained by the common component.

For the same reasons as presented in the previous subsection, standardising is also necessary in the PLS technique. Crucially, the two variables $\mathbf{X}_{1:t-h}$ and $\mathbf{Y}_{h+1:t}$ used to construct the original $\mathbf{M}_t$ are standardised column-wise. In addition, the entire information set available at time t ought to be standardised separately as it serves to estimate each factor. Logically, as solely these three variables are used throughout, a single initial standardisation ensures the smoothness of the estimation of all factors (in every time iteration).

In the forecasting framework at hand, two approaches to modelling the original matrix $\mathbf{M}_t$ are employed, and lead up to three different PLS models in total. Specifically, models follow from a simple static approach, in line with the PC model described in Subsection 4.1.1, and from a dynamic approach, more interesting in this context as it exploits the dynamics of time series data, in which two distinct possibilities are explored.

Static approach:

a) In the simplest PLS version, the lags of the target variable are only resorted to in the second step of the procedure, in the forecasting model. In the estimation of the factors, only the original information set full of its 132 macroeconomic time series is used in combination with the isolated target variable. The exclusion of any dynamic relationships of the target from the factor estimation leads to the static approach to PLS, potentially resulting in weaker predictive power of the most important regressors in equation 4 that are the factors (and lags thereof).

Dynamic approach:

b) A first option to derive PLS factors by exploiting the dynamics of the series consists in directly expanding the predictor dataset to be used in the factor estimation with lags of the target variable. This then serves as a substitute for any lags present in the forecasting equation 4. The objective sought here is to improve the estimated factors by including these additional predictors in the linear combinations. On the other hand, there exists a risk that the size of the information set, no matter the quality of the variables contained in it, reduces the usefulness of the lags to be insignificant, which instead could bear more power as stand-alone regressors in the forecasting model. As a consequence, the performance of this approach is expected to worsen with the importance of an AR-type of process for explaining the target variable.

   Consistent with what is stated earlier in the section, the number of lags added to the predictor series is set independently in each iteration and can go up to a maximum of six. In the altered setup of this dynamic approach, however, this number cannot be chosen according to BIC anymore. It is then rather set so as to minimise the squared distance between the estimated and observed target variable in each period, in line with the forecast metric to be used.

c) Another possibility to approach the PLS factor estimation dynamically is to aim for the estimated factors to summarise the residual part of the target variable, while using the lags of the target in the forecasting equation 4 similarly to the static approach. More explicitly, an AR(p) process is fitted to the target variable before computing the matrix $\mathbf{M}_t$, whose residuals only are used in the construction. The number of lags p is set in each iteration, in the same way as the size of the enlargement in PLS (b), and up to a maximum of six, in line with all other choices of lags. Meanwhile, the predictor information set is kept standard in the factor estimation. The objective sought in this approach is clear: removing the autocorrelation of the target variable series, constructing PLS factors using this unexplained information contained in the lags, and modelling the AR-like properties of the series in complete isolation of the factor estimation.

Following the analysis of the three PLS implementations conducted by Fuentes et al. (2015), the number of factors considered is limited to two. The authors indeed argue that out of a total of five factors, the best forecasting results are generally found without exceeding this upperbound. Note that while many methods have been proposed in the literature to set k by balancing the amount of extracted information and efficiency of estimation, for instance through the use of information criteria as Bai and Ng (2002) propose, allowing for this number to be determined by the forecasting performance of the ultimate models has commonly been done by Stock and Watson (1998, 2002b).

4.2 Structural Break

As argued in Section 2, the methodology employed to test for the existence of a break closely follows that developed by Han and Inoue (2015) as it presents the most interest in the application at hand. Specifically, the test locates the most significant location, if any, for a structural change Q_0 in the construction of the factors in the estimation sample. Once the test is performed, the instability is taken into account by shrinking the sample in the relevant way, and results of the four extended PC and PLS models can be obtained. In this case, although we do not focus on the pre-break subsample, the matrix of factor loadings can be seen as varying over time – $\mathbf{\Lambda}_t$ – as it abruptly changes from the threshold date found by the test onwards. As such, the factor model given in equation 3 above becomes

$$\mathbf{x}_t = \begin{cases} \mathbf{f}_t \mathbf{\Lambda}_{pre} + \boldsymbol{\epsilon}_t & \text{for } t < Q_0 \\ \mathbf{f}_t \mathbf{\Lambda}_{post} + \boldsymbol{\epsilon}_t & \text{for } t \geq Q_0 \end{cases} \tag{5}$$

The test allows for serial, cross-sectional correlation and heteroskedasticity in the idiosyncratic shocks $\boldsymbol{\epsilon}_t$, the knowledge of which is not required in the implementation.

The test is performed on factors obtained under the null hypothesis of no parameter shifts, i.e. $H_0 : \mathbf{\Lambda}_{pre} = \mathbf{\Lambda}_{post}$. Note however that this does not mean that a break in every single factor loading is required for the time instability to be significant nor does it imply it in any case; instead, the test should be seen as testing for an important change in the way the information set can be summarised and in the group of variables that drive the common variation. Intuitively, a break in only a small portion of parameters can be found significant if its magnitude is important enough to compensate for the constancy of other loadings. Therefore, the finding of a significant shift should not be interpreted as implying a change associated with all variables either.

In their paper, Han and Inoue (2015) show that the existence of a structural break in the factor loadings necessarily entails the presence of one in the second moments of factors in the two subsamples it creates. Using this fact, the authors are able to reduce a high-dimensional problem (for large N) to a simpler one. They propose a Wald-like test statistic thus based on the pre- and post-break subsample means of estimated factors $\hat{\mathbf{f}}_t \hat{\mathbf{f}}_t'$, for which they define the following vector:

$$\mathbf{A}(Q, \hat{\mathbf{F}}_T) = \text{vec}\left(\sqrt{T}\left(\frac{1}{Q}\sum_{t=1}^{Q} \hat{\mathbf{f}}_t \hat{\mathbf{f}}_t' - \frac{1}{T-Q}\sum_{t=Q+1}^{T} \hat{\mathbf{f}}_t \hat{\mathbf{f}}_t'\right)\right) \tag{6}$$

where $\hat{\mathbf{F}}_T$ denotes the $(k \times T)$-matrix of factors $\hat{\mathbf{f}}_t$ for all $t \in \{1, \ldots, T\}$ as its rows. The covariance matrix of the vector $\mathbf{V}(Q, \hat{\mathbf{F}}_T)$, as defined by its unrestricted long-run estimate in Section A.1, then serves to compute the test statistic:

$$\sup_{Q \in [Q_1, Q_2]} W_T(Q, \hat{\mathbf{F}}_T) \equiv \sup_{Q \in [Q_1, Q_2]} \mathbf{A}(Q, \hat{\mathbf{F}}_T)' \, \mathbf{V}(Q, \hat{\mathbf{F}}_T)^{-1} \, \mathbf{A}(Q, \hat{\mathbf{F}}_T) \tag{7}$$

where Q, as can be understood from the formulas, denotes the last date that precedes the break. The optimal result found with the Wald test statistic therefore leads to the value $Q_0 = Q + 1$ as best breakpoint location.

The test is performed on the factors obtained from the initial estimation sample; in other words, it is only operated in the first iteration of all forecasting models. The structural break analysis is therefore not dynamic. We are here concerned with the effect of accounting for the sole existence of time instability, instead of investigating its evolving significance through time and potentially having to compare initial breakpoint found to other, more recent breaks arising as the estimation window expands. Besides, the application of the test to the set of samples considered in this paper allows for interesting insights to be derived without overcomplicating the extended models. In this context, T thus refers to the size of the estimation sample throughout this section. The results are nonetheless used to re-estimate models for all eight samples presented in Section 3. This is indeed relevant as an improvement of the ultimate out-of-sample performance of the model can itself be seen as evidence of a structural change in the construction of factors (Stock, 2003). In addition, the procedure might yield interest insight if the same time instability in an estimation sample turns out to improve the forecasting performance over two different time windows.

As can be seen from equation 7, the candidate breakpoint locations Q are comprised in the interval

$[Q_1, Q_2]$. This has two important implications. First, the maximum cutting of the observations up to Q_2 can lead to identification issues in the application at hand. To avoid this, this date is set so as to ensure that the size of the remaining, inclusive sample matches at least that of the smallest estimation samples in Table 1. Consequently, the setting of Q_2 forces us to leave out the analysis on the smallest estimation samples, i.e. M1, M4, M5 and M7. As for Q_1, it is set two years after the beginning of the period used. Picking an earlier date presents little interest as the analysis should not focus on the usefulness of a few data points only, while it has the disadvantage of decreasing the power of the test (Andrews, 1993). The resulting intervals chosen are summarised in Table 2. Second, as the procedure provides multiple opportunities for the null hypothesis to be rejected, it follows logically that standard critical values cannot be employed here (Stock, 2003). In fact, the Wald statistic has the same asymptotic distribution as the conventional supreme Wald test proposed by Andrews (1993), who presents an extended panel of critical values that fits this framework.

Table 2: Structural break test intervals

	Q_1 to Q_2
M2,M6	1962:03 to 1970:03
M3	1962:03 to 1980:03
M8	1972:03 to 1980:03

Once the optimal breakpoint location Q_0 has been established, the time variance is introduced by constructing the factor usign information contained in the post-break subsample only. In the subsequent estimation of the forecasting model, the instability is also taken into account by solely regressing post-break data points. In this way, not only is the structure of the factors in relation with the information set non-constant, but their role as explanatory variables also is.

With time-varying loadings, factors can be seen as representing different latent variables across subsamples and can be assumed to have different meanings. Setting the number of factors used, the number of lags included in the regression, and estimating their coefficients in the forecasting equation independently as a final step therefore follows logically.

4.3 Markov Regime Switching

The methodology followed in this second extension to the factor basic models covered in Section 4.1 is mostly inspired by the work of Liu and Chen (2016). With this extension, the time-varying factor model can best be represented as follows:

$$\mathbf{x}_t = \begin{cases} \mathbf{f}_t \mathbf{\Lambda}_1 + \boldsymbol{\epsilon}_t & \text{for } s_t = 1 \\ \mathbf{f}_t \mathbf{\Lambda}_2 + \boldsymbol{\epsilon}_t & \text{for } s_t = 2 \end{cases} \tag{8}$$

where each time period t is characterised by a state s_t, which corresponds to either of two mutually exclusive states simply referred to as 1 and 2. We refer to Section 2 for the economic motivation for adopting a two-state space. As for econometric aspects, considering a higher number of states would likely lead to identification issues given the dimensions of the data at hand, besides computational difficulties.

In the forecasting framework at hand, the method finds an estimate of the two-state state vector $\hat{\mathbf{s}}$. The estimation is again performed once per model with the initial information available, such that T is defined

as the size of the estimation sample. Broadly put, the procedure consists in an iterative algorithm in the four following steps:

Step 1: Set the elements of the state vector to some initial values $\hat{s}_1, \hat{s}_2, \ldots, \hat{s}_T$.

Step 2: Given $\hat{\mathbf{s}}$, compute variables $\hat{\pi}_{j,k}^{(l)}$, $\hat{\pi}_k$, $\hat{\boldsymbol{\mu}}_k$, $\hat{d}_k$, $\hat{\mathbf{Q}}_k$ and $\hat{\mathbf{B}}_k$ for both regimes to estimate the regime-specific factor loading spaces, where

$$\hat{\pi}_{j,k}^{(l)} = \frac{\sum_1^{T-l} I(s_t = j), I(s_{t+l} = k)}{\sum_{t=1}^{T-l} I(s_t = j)}$$

$$\hat{\pi}_k = \frac{\sum_{t=1}^{T} I(s_t = k)}{T}$$

for $j, k \in \{1, 2\}$.

Step 3: Given the variables obtained in Step 2, derive new estimate of state vector $\hat{\mathbf{s}}$ by maximising $G_T(\mathbf{s}, \mathbf{B}_1, \mathbf{B}_2, \mathbf{X}))$ using the Viterbi algorithm.

Step 4: Repeat Steps 2 and 3 until convergence (or a maximum number of iterations) is reached.

First, the initialisation is done in the following way: the inflation observations that are below the mean of the series are set to belong to one state, while the periods of above-average inflation belong to the other. This approach seems reasonable as the resulting distribution should balance the number of each state well, and as regime switches are likely to be accompanied by changes of level in the target macroeconomic series if the models turn out to be relevant. Besides, Liu and Chen (2016) suggest a systematic method to initialise the state vector, but the computational difficulty of the method is anticipated to yield a useless result in this large-N context.

Second, a series of variables is computed in order for transition probabilities and probability density of the residual factor component. The l-period transition probability from state j to k, $\hat{\pi}_{jk}^{(l)}$, the unconditional probability of transitioning to state k, $\hat{\pi}_k$, are both directly derived from the estimation sample. Consistently, $\boldsymbol{\Sigma}_{\mathbf{X}}(l)$ represents the l-order autocovariance of the predictor variables $\mathbf{x}_t$ whose observations are contained in the estimation sample, and is derived for each state as follows:

$$\hat{\boldsymbol{\Sigma}}_{\mathbf{X},k}(l) = \frac{\sum_{t=1}^{T-l} \sum_{j=1}^{2} (\mathbf{x}_t - \hat{\boldsymbol{\mu}}_k)'(\mathbf{x}_{t+l} - \hat{\boldsymbol{\mu}}_j) I(\hat{s}_t = k, \hat{s}_{t+l} = j)}{\sum_{t=1}^{T-l} I(\hat{s}_t = k)} \tag{9}$$

The quadratic versions of l-order covariance matrices are then summed over a certain number of leads l_0 as represented in equation 10. The autocorrelation is often at its strongest at small time lags, hence the parameter is arbitrarily fixed at 4 for all regimes. In any case, the estimation of the loading space to be described below is relatively insensitive to the choice of l_0 (Lam, Yao, & Bathia, 2011; Chang, Guo, & Yao, 2015).

$$\hat{\mathbf{M}}_k = \sum_{l=1}^{l_0} \hat{\boldsymbol{\Sigma}}_{\mathbf{X},k}(l) \hat{\boldsymbol{\Sigma}}_{\mathbf{X},k}(l)' \tag{10}$$

where $\hat{\boldsymbol{\mu}}_k$ is the N-dimensional row vector made of all series means for each regime k. The matrix defined here for each state is an equivalent to the $\mathbf{M}$ matrices introduced earlier for PC and PLS models, on which the factor estimation relies through eigenvalue decomposition. This specific representation can be seen

as forming the basis of a unsupervised dimensionality reduction technique, but similarly to PLS analysis, extracting factors from it comes down to maximising the quadratic variance of some variables. The equation is independent of any subscript t as this specific factor estimation is resorted to only once to obtain a state vector spanning the estimation sample.

From the matrices $\hat{\mathbf{M}}_k$ are defined $\hat{\mathbf{Q}}_k$ and $\hat{\mathbf{B}}_k$, matrices whose columns are the eigenvectors corresponding to the first $\hat{d}_k$ and last $(N^\star - \hat{d}_k)$ largest eigenvalues respectively, such that for a given regime, the two span orthogonal spaces. The eigenvalue-ratio method of (Lam & Yao, 2012) is used to set up $\hat{d}_k$ in the following way:

$$\hat{d}_k = \underset{1 \leq i \leq N^\star/2}{\operatorname{argmin}} \frac{\hat{\lambda}_{k,i+1}}{\hat{\lambda}_{k,i}} \tag{11}$$

where $\hat{\lambda}_k, i$ is the i^{th} largest eigenvalue of $\hat{\mathbf{M}}_k$. In other words, the approach consists in searching for the point of smallest incremental portion of variance captured by an additional eigenvector. Only the first $N^\star/2$ possibilities are considered instead of all $N^\star$ to prevent approximations of small numbers (among the last eigenvalues) to negatively affect the obtained result.

Note that in our context, the number of variables is particularly large, sometimes even larger than the time dimension. As the algorithm involves the full eigenvalue decomposition and use of "last" eigenvectors in $\hat{\mathbf{B}}_k$, we slightly modify to ensure orthogonality of the eigenvectors and thereby the correct span of the column space of the matrix. In this perspective, the scalar $N^\star$ corresponds to the rank of the autocovariance matrix of $\mathbf{X}$ and is used above, instead of simply taking the number of series N as specified by Liu and Chen (2016).

There are two main reasons for such a factor estimation in the second step of the algorithm, instead of the PC and PLS methods developed in Section 4.1. Crucially, as opposed to PLS analysis, the factor estimation makes use of all data points contained in the estimation sample rather than disregarding h observations. That is due to its sole focus on the information set, in a similar way to PC analysis. In addition, the theoretical properties of the algorithm have been studied in detail by Liu and Chen (2016) in this specific setup only, including its convergence. In fact, a faster convergence rate has been obtained by setting the number of factors to that of eigenvectors used in the stronger state, where the strength of a state is defined by its 2-norm.[4] In other words, it is set to $\hat{d} = \hat{d}_{\tilde{k}}$ where $\tilde{k} = \underset{k=1,2}{\operatorname{argmax}} \|\hat{\mathbf{M}}_k\|_2$. This type of factor construction is thus used repeatedly in the iterative algorithm employed to estimate the state space, but is however not used dynamically as the estimation window expands.

As a third step, the derived variables allow for a new state vector $\hat{\mathbf{s}}$ to be estimated in the hope of converging towards optimality using the Viterbi algorithm, which is explained in Algorithm 1. The procedure involves the density function $G_n(\mathbf{s}, \mathbf{B}_1, \mathbf{B}_2, \mathbf{X})$, which is based on the variation of the information set not captured by the selected regime-specific factors, i.e. the residual component of the factor model, as designated by $\hat{\mathbf{B}}'_{s_t}\mathbf{x}'_t$ at time t. For simplicity, the reader is referred to Section A.2 for the exact shape of the likelihood function. Intuitively, the algorithm can be seen as a forward progression towards the best path through states 1 and 2, where a path $\mathbf{P}_{n,k}$ defines a sequence of regimes spanning the first $n \in \{1, \dots, T\}$ time periods and ending in state k. The notation n is chosen to avoid confusion with previously used index t that advances with the expanding window. The index here runs through the estimation sample and serves an iterative algorithm rather than a dynamic procedure.

[4] The 2-norm of a matrix corresponds to its largest singular value.

Algorithm 1: Viterbi algorithm for estimating the state vector $\hat{\mathbf{s}}$

Initialise one-period path for each regime: $P_{1,1} = 1$ and $P_{1,2} = 2$

for $n = 2, \ldots, T$ **do**

  for $k = 1, 2$ **do**

    Step 1: Find most likely $(n-1)$-period path to precede transition to state k

$$s^{\star}_{n,k} = \underset{j=1,2}{\operatorname{argmax}} \left[G_{n-1}(\mathbf{P}_{n-1,j}) + \log(\pi_{j,k} f(\mathbf{B}_k' \mathbf{x}_n') \right]$$

    Step 2: Create n-period path with final state k

$$\mathbf{P}_{n,k} = \begin{bmatrix} \mathbf{P}_{n-1,s^{\star}_{n,k}} \\ k \end{bmatrix}$$

  end

end

Set full estimated state vector to optimal obtained path: $\hat{\mathbf{s}} = \underset{k=1,2}{\operatorname{argmax}} \, G_T(\mathbf{P}_{T,k}, \mathbf{B}_1, \mathbf{B}_2, \mathbf{X})$

Finally, the fourth step of the iterative procedure consists in checking for convergence and potentially repeating the last two steps, unless it exceeds the maximum number of iterations that we set to 50. To perform this check, we refer to the two convergence criteria suggested by Liu and Chen (2016) and made explicit in Appendix A.2.2.

After the Markov regime-switching model has been estimated on the initial sample to derive the state vector, the factors are constructed and the forecasting equation is estimated in each iteration. Forecasts are constructed based on a weighted average of two forecasts conditional on the current state using the relevant transition probabilities, as can be represented by the following formula:

$$\hat{y}_{t+h} = \hat{\pi}^{(h)}_{j,1} \hat{y}^{(1)}_{t+h} + \hat{\pi}^{(h)}_{j,2} \hat{y}^{(2)}_{t+h} \tag{12}$$

where $\hat{y}^{(1)}_{t+h}$ and $\hat{y}^{(2)}_{t+h}$ are forecasts made assuming a transition to state 1 and 2 respectively, and where $j \in \{1, 2\}$ is the relevant state in period t. The two state-specific forecasts derived are compared ex-post to the true value of the target variable every time the estimation window expands, and the state of the passing period is set to that of the model that yielded the closest prediction on its own. This new state is then considered "observed", in the same way as the estimated states initially outputted by the algorithm. The transition probabilities are then updated by making use of all past periods.

Similarly to the structural break analysis, the Markov regime-switching models are only estimated for some of the samples presented in Section 3. Namely, the algorithm is applied to the different estimation samples focus on estimation samples. Besides, once again for a matter of identification, the four smallest estimation samples are left out of this analysis. The state vector is estimated separately for each time horizon on which it depends only once, as unlike the structural break test, the procedure relies on its single and own other kind of initial factor estimation technique (see 4.3 for argumentation).

5 Results

The forecast performance is evaluated based on Mean Squared Error (MSE) of the predictions, a very common and key measure in a forecasting context. However, for better comparability of the models, a relative metric is ultimately employed instead of an absolute one that would hinder the analysis across samples of different sizes. The Relative Mean Squared Error (RMSE) is therefore introduced as the ratio of the MSE obtained over that of a benchmark model as in 13, which is here chosen to be an AR(4)-like model for referability to Bai and Ng (2008) and Fuentes et al. (2015); specifically, we regress the target variable y_{t+h} on z_t as well as three lags thereof for each setting of sample and forecast horizon h (see Section 3 for details on the variables).

$$\text{RMSE (model investigated)} = \frac{\text{RMSE (model investigated)}}{\text{RMSE (benchmark model)}} \tag{13}$$

As a result, a value below 1 indicates an improvement over the benchmark whereas ratio above unity points at a lack of incremental explanatory power.

This section is structured as follows. First, results for basic PC and PLS models are presented. Then, models extended with a structural break and Markov regime-switching models are analysed in Sections 5.2 and 5.3.

5.1 Basic PC and PLS Models

Table 3 shows the RMSE values obtained for all models, along with the optimal number of factors k among possible values set (see 4.1 for details). A few observations follow from the results gathered.

First, regarding the out-of-sample performance of competing methods, it can be noted that PLS analysis outperforms PC, as anticipated in a forecasting framework. The models are found to perform better in about 72% of the cases, although most often by a relatively small margin. Most strikingly, the method is observed to particularly enrich the amount of relevant information contained per factor; while the performance of PLS models peaks for more than half of the settings when one single factor is included, PC models require more than 3 factors on average to achieve the same range of RMSE values. Efficiency gains are therefore obtained by performing PLS analysis.

Second, interesting observations can be made regarding the PLS models that aim at capturing the dynamic properties of the target series through its lags directly in the factor estimation. Clearly, the last approach explored, PLS (c), appears to be very relevant to inflation forecasting compared to both static PLS analysis and across the board as the method is found to outperform other PLS models in about 88% of the cases while achieving the best RMSE of all in more than 60%.

In contrast, the other dynamic model implemented, PLS (b), generally gives the worst results in aggregate; in fact, it is found to perform worse than the benchmark model in the vast majority of samples for shorter-term forecasting ($h = 1$ and $h = 6$). The distinct shape of the forecasting equation 4 in the unique case of PLS (b) is likely to account for this observation: as mentioned earlier, as the construction of factors essentially relies on linear combinations of the predictor variables, the average weight assigned to each decreases with the N-dimension of the factor model. With our large information set, modelling these dynamic characteristics out of the factor estimation seems to lead to better results as otherwise their role in the forecasting is too sharply reduced. The findings made here therefore also point to the expected disproportionate importance of target lags in the inflation process as compared to other individual series. Ultimately, the dynamic properties are better modelled by including an AR-like regressors in the forecasting model while making use

Table 3: RMSE and best number of factors in basic models

(a) $h = 1$

	PC		PLS (a)		PLS (b)		PLS (c)	
M1	**0.934**	*3*	1.101	*1*	1.324	*1*	0.935	*1*
M2	0.954	*6*	0.979	*1*	1.022	*1*	**0.920**	*1*
M3	**0.883**	*4*	0.955	*1*	1.120	*2*	0.928	*1*
M4	0.964	*3*	1.049	*1*	1.191	*1*	**0.945**	*1*
M5	0.954	*3*	1.031	*1*	1.184	*2*	**0.941**	*1*
M6	0.937	*6*	0.968	*1*	1.072	*2*	**0.920**	*1*
M7	0.940	*4*	1.020	*1*	1.179	*2*	**0.932**	*1*
M8	**0.915**	*4*	1.054	*1*	1.114	*2*	0.943	*1*

(b) $h = 6$

	PC		PLS (a)		PLS (b)		PLS (c)	
M1	**0.655**	*4*	0.703	*1*	1.116	*2*	0.664	*1*
M2	**0.626**	*1*	0.635	*1*	0.987	*2*	0.632	*1*
M3	0.658	*1*	0.627	*1*	1.175	*2*	**0.624**	*2*
M4	**0.676**	*1*	0.701	*1*	1.129	*2*	0.678	*1*
M5	0.667	*1*	0.682	*1*	1.116	*2*	**0.665**	*1*
M6	0.621	*1*	**0.618**	*1*	0.992	*2*	0.619	*1*
M7	0.665	*1*	0.680	*1*	1.136	*2*	**0.664**	*1*
M8	0.697	*1*	0.633	*2*	1.159	*2*	**0.628**	*1*

(c) $h = 12$

	PC		PLS (a)		PLS (b)		PLS (c)	
M1	**0.667**	*1*	0.696	*1*	0.994	*1*	0.689	*1*
M2	0.604	*6*	0.592	*1*	0.868	*2*	**0.575**	*2*
M3	0.719	*2*	0.605	*2*	1.183	*2*	**0.588**	*2*
M4	0.642	*1*	**0.632**	*1*	0.955	*2*	0.639	*1*
M5	0.645	*1*	**0.629**	*1*	0.963	*2*	0.635	*1*
M6	0.621	*6*	0.593	*2*	0.897	*2*	**0.572**	*2*
M7	0.650	*1*	**0.630**	*1*	0.988	*2*	0.639	*1*
M8	0.760	*1*	0.674	*2*	1.208	*2*	**0.636**	*2*

(d) $h = 24$

	PC		PLS (a)		PLS (b)		PLS (c)	
M1	**0.554**	*4*	0.565	*1*	0.823	*1*	0.563	*1*
M2	**0.506**	*6*	0.521	*2*	0.695	*2*	**0.506**	*1*
M3	0.599	*3*	0.555	*2*	1.202	*2*	**0.540**	*2*
M4	**0.544**	*6*	0.559	*1*	0.786	*2*	**0.544**	*1*
M5	0.557	*6*	0.563	*1*	0.818	*2*	**0.549**	*1*
M6	0.536	*6*	0.525	*2*	0.765	*2*	**0.509**	*2*
M7	0.565	*6*	0.567	*1*	0.843	*2*	**0.551**	*1*
M8	0.690	*3*	0.605	*2*	1.221	*2*	**0.586**	*2*

Note: Each pair of columns contains the best RMSE and number of factors associated with it. Bold entries highlight the model with lowest RMSE in each row.

of the residual, autocovariance-free component of the target series in the factor estimation as in PLS (c).

Overall, the table displays a clear pattern with regards to factor estimation as a whole: in comparison with the benchmark model, the improvements the factor models considered bring are more substantial the higher the time horizon h at hand. Indeed, while fairly close for $h = 1$, the derived MSE values of our models are almost half those of the benchmark for $h = 24$, in line with findings on factor models from the broader literature (Matheson, 2006).

Given the similarity of our approach to that of Bai and Ng (2008) and Fuentes et al. (2015), as mentioned throughout Section 4.1, a comparison of the results obtained are in order. In what might appear strange at first sight, values seem to differ to various degrees across the board. Two possibly complementary explanations are suggested. On the one hand, the authors might have performed data-cleaning procedures such as through outlier detection before estimating the factors and model. Indeed, this approach is very implicitly mentioned by Stock and Watson (2005) when introducing the data. However, given the lack of transparency on the details of the method used and the fact that any reference to it is absent from the other papers referred to, the procedure is here intentionally left out of the implementation. On the other hand, with regards to basic PLS models developed by Fuentes et al. (2015), the methodology employed might slightly differ. Specifically, the technique they describe to extract eigenvectors from the quadratic correlation between the information set and the target series and construct factors therewith appears to involve the use

of out-of-sample observations of the target (in the matrix **M**). This is however in conflict with the principle of pseudo out-of-sample forecasting. Consequently, the number of time periods used to derive the matrix in question decreases with the horizon in our case, whereas they keep it independent of it. We again refer to Section 4.1 for an extensive description of our implementation of the models and the logic behind it.

As a result of this, it comes as no surprise that the MSE values they obtain for PLS models are lower, and more significantly so the higher h. Besides, in view of the first discrepancy in the methodology, all values can be expected to differ slightly in no guaranteed direction, which is in line with the results of all PC models, as well as of PLS models in Table 3a, for which the out-of-sample explanation is unlikely to have much weight.

5.2 Models with Structural Breaks

In this section, we present the results of the extended PC and PLS models when allowing for a structural break in the previously defined intervals. The forecasting performance of these models with time-varying factor loadings is investigated in the same way as in the former section: the RMSE values corresponding to the best configuration in terms of k are shown in Table 5. First, however, we analyse the relevant outputs from the structural break test, namely the optimal breakpoint date Q_0 found and the significance of the associated Wald statistic derived. These results are displayed in Table 4. Note that since samples $M2$ and $M6$ share the same estimation sample, their test results are the same unless the best forecasting performance is reached with a different number of lags for both; in that case, the two respective tests are applied to different sets of factors and are therefore independent.

Note that as explained in the methodology, the significance of the statistics derived is investigated using adjusted critical values elaborated by (Andrews, 1993). These vary with the interval of candidate dates tested as well as the number of factors estimated. The relevant critical values can easily be derived from the information given in Table 2 and 5 in combination with the extensive tables displayed in the author's paper.

From Table 4 follows the observation that the time-variance of factor loadings is generally significant in the inflation process. Indeed, most tests performed suggest that time instability in the factor structure needs to be taken into account to achieve correct specification of the models. As a whole, the results indicate a parameter shift somewhere in the middle of the broad time window – from March 1962 to March 1980 – being investigated in pieces. While some outputs point to earlier and later dates, the outputs altogether suggest a structural change towards the turn of the 1970s given the restricted intervals tested; results for the first sample (M2 and M6) converge towards a late date among possible locations up to March 1970, while those for the last sample (M8) are rather concentrated close to the earlier bound of March 1972. Nonetheless, these differences may well altogether suggest the presence of several breaks in the series.

A few interesting insights can be derived by interpreting the significance of the test statistics. First, while the values derived for PC models can be expected to grow higher with the number of factors included, they are found to be strikingly more significant in relative terms as well. This means that unlike the entire variance contained in the information set, the common movement between the predictor and target variables is subject to far less non-constancy in the factor loadings. This makes intuitive sense since, if the economy is thought to evolve through various stages, the information captured by PC factors is likely to focus mostly on the variance across these stages and thereby exhibit a time instability in the factor structure. Structural breaks would therefore match turning points of the economic cycle, and modelling this time variance is of primary importance. In contrast, the relevant portion of the variance to the inflation might not change shape at these points. If PLS factors assign much weight to leading indicators, these

Table 4: Best breakpoint location and test statistic

(a) $h = 1$

	PC		PLS (a)		PLS (b)		PLS (c)	
M2	1966:07	*48.570**	1969:03	*3.255*	1969:03	*3.001*	1966:07	*11.641*
M3	1963:11	*151.692**	1969:11	*8.432*	1969:11	*12.019**	1969:11	*9.688**
M6	1966:07	*48.570**	1969:03	*3.255*	1969:03	*3.001*	1962:04	*2.117*
M8	1973:04	*92.000**	1975:09	*3.299*	1973:01	*22.153**	1976:03	*3.557*

(b) $h = 6$

	PC		PLS (a)		PLS (b)		PLS (c)	
M2	1962:09	*36.808**	1969:11	*8.690**	1969:12	*9.962**	1969:11	*8.684**
M3	1970:01	*7.248*	1969:11	*17.477**	1969:09	*16.621**	1969:11	*17.562**
M6	1962:09	*36.808**	1969:11	*8.690**	1969:12	*9.962**	1969:11	*8.684**
M8	1975:05	*2.336*	1975:05	*4.726*	1973:01	*23.439*	1975:05	*4.729*

(c) $h = 12$

	PC		PLS (a)		PLS (b)		PLS (c)	
M2	1962:09	*35.813**	1969:07	*17.569**	1969:19	*17.027**	1969:09	*17.760**
M3	1970:01	*6.262*	1969:06	*23.293**	1969:09	*21.437**	1969:07	*22.925**
M6	1962:09	*35.813**	1969:07	*17.569**	1969:10	*17.027**	1969:09	*17.760**
M8	1975:05	*2.278*	1973:01	*18.268**	1973:02	*19.178**	1973:03	*17.209**

(d) $h = 24$

	PC		PLS (a)		PLS (b)		PLS (c)	
M2	1963:08	*74.040**	1969:09	*12.469**	1969:09	*17.665**	1969:05	*17.949**
M3	1965:09	*75.390**	1969:06	*28.792**	1969:03	*27.239**	1969:03	*28.804**
M6	1963:08	*74.040**	1969:06	*18.118**	1969:09	*17.665**	1969:05	*17.949*
M8	1975:05	*2.073*	1973:07	*15.110**	1973:02	*15.602**	1973:02	*15.056**

Note: Each pair of columns contains the best breakpoint location found and its corresponding Wald test statistic, for which an asterisk is used to represent the significance.

might well drive inflation through both expansionary and recessionary economic times; in other words, the correlation between macroeconomic variables might not be as time changing. Parameter shifts could instead correspond to other sorts of structural changes in this case, found to be of smaller significance in the PLS factor loadings.

Second, it appears that the impact of a potential structural break on forecasting with horizon of higher order is found to be more severe. As is clear from earlier results, factor models are observed to present most interest over the benchmark for longer forecast horizon. These results further suggest the ability PC and PLS models to capture the driving forces behind inflation over longer time horizon, with increased sensitivity to the non-constancy of parameters. Contrary to statements made earlier, the high significance of the statistics derived could indicate the existence of sudden sharp breaks in the long-term factor structure of inflation.

Finally, the test statistics on PLS (b) models also show somewhat larger degrees of time instability in the factor loadings for some settings (mostly with short-term horizons). While not so clear in the numbers, this observation might result from the fact that inflation lags, found to be very relevant in direct modelling of inflation by interpreting Table 3, also exhibit particularly strong time instability in their relationship with current inflation among all variables of the enlarged information set. The PLS models might therefore greatly benefit from allowing for time-variant coefficients in its AR process in the forecasting model of extended models. The results gathered in Table 5 are analysed below in connection with these observations.

Table 5: RMSE and best number of factors in models extended with structural break

(a) $h = 1$

	PC		PLS (a)		PLS (b)		PLS (c)	
M2	**0.912***	*6*	0.979	*1*	1.023	*1*	0.944	*2*
M3	**0.881***	*5*	1.049	*1*	1.106*	*2*	0.994	*1*
M6	**0.918***	*6*	0.988	*1*	1.085	*1*	0.955	*1*
M8	**0.914***	*3*	1.076	*1*	1.087*	*2*	1.014	*1*

(b) $h = 6$

	PC		PLS (a)		PLS (b)		PLS (c)	
M2	0.653	*4*	**0.645**	*1*	1.011	*2*	0.644	*1*
M3	0.686	*1*	0.625*	*1*	1.163*	*2*	**0.624**	*1*
M6	0.650	*4*	0.624	*1*	0.999	*2*	**0.623**	*1*
M8	0.695*	*1*	**0.639**	*1*	1.123	*2*	0.641	*1*

(c) $h = 12$

	PC		PLS (a)		PLS (b)		PLS (c)	
M2	**0.576***	*4*	0.582*	*2*	0.810*	*2*	0.596	*2*
M3	0.696*	*1*	0.600*	*2*	1.099*	*2*	**0.585***	*2*
M6	0.596*	*4*	0.599	*2*	0.832*	*2*	**0.585**	*2*
M8	0.682*	*1*	0.574*	*2*	1.023*	*2*	**0.566***	*2*

(d) $h = 24$

	PC		PLS (a)		PLS (b)		PLS (c)	
M2	**0.504***	*6*	0.582	*1*	0.682*	*2*	0.572	*2*
M3	0.586*	*3*	0.537*	*2*	1.099*	*2*	**0.529***	*2*
M6	**0.531***	*6*	0.576	*2*	0.740*	*2*	0.566	*2*
M8	0.653*	*1*	**0.558***	*2*	1.096*	*2*	0.569*	*2*

Note: Each pair of columns contains the best RMSE with its number of factors. Bold entries highlight the superior model in each row, and an asterisk indicates an improvement over the equivalent basic model.

The RMSE values provide support for the comments made regarding the structural break test results. Clearly, the extension is key to PC models, for which the improvements over the basic version are non-negligible, in line with the proposed explanation of highly volatile information it aims to capture. In contrast, improvements of PLS models are more scarce. Nonetheless, the consistent decrease in the RMSE values for high forecast horizons goes against the general findings in the literature regarding the robustness of PLS analysis to structural changes. Overall, the relevance of incorporating time instability in the factor loadings, and specifically the adequacy of accounting for a structural break, is supported by the findings. The same holds for the non-constancy of the regression coefficients as PLS (b) is in fact found to benefit the most from extending the models in this manner.

5.3 Markov Regime-Switching Models

The procedure described to estimate the state vector for the three distinct estimation samples is found to be greatly applicable to the dataset. Indeed, convergence is obtained within 15 to 25 iterations, and only very small changes are observed across the last iterations when the limit is reached. The obtained vectors often present three to five long sequential regimes, spanning an average of about three years, with only few occurrences of regimes sustained for less than a year. Note that this is obtained no matter how rapid the alternation of regimes is in the initial state vector. While no extensive study is conducted to determine the

robustness of the applied method to the setting of starting states – for the interested reader, we refer to Liu and Chen (2016) –, the sole key condition on the initialisation appears to be that the vector should be sufficiently well balanced. Sticking to the methodology described in the paper, the following results on the forecasting performance of the models are obtained:

Table 6: RMSE and best number of factors in Markov regime-switching models

(a) $h = 1$

	PC		PLS (a)		PLS (b)		PLS (c)	
M2	0.927*	*2*	0.950*	*1*	**0.875***	*2*	0.904*	*2*
M3	0.892	*3*	0.913*	*2*	1.138	*2*	**0.877***	*2*
M6	0.914*	*2*	0.937*	*1*	0.920*	*2*	**0.910***	*2*
M8	**0.894***	*6*	0.958*	*1*	1.076*	*2*	0.919*	*1*

(b) $h = 6$

	PC		PLS (a)		PLS (b)		PLS (c)	
M2	**0.622***	*8*	0.649	*2*	1.084	*2*	0.634	*2*
M3	0.625*	*4*	0.607*	*1*	1.179	*2*	**0.516***	*1*
M6	**0.619***	*8*	0.628	*2*	1.116	*2*	0.632	*1*
M8	0.572*	*9*	0.589*	*2*	1.310*	*2*	**0.522***	*1*

(c) $h = 12$

	PC		PLS (a)		PLS (b)		PLS (c)	
M2	0.677	*6*	0.732	*2*	1.078	*2*	**0.675**	*2*
M3	0.683*	*2*	0.703	*2*	1.370	*2*	**0.618**	*2*
M6	0.701	*6*	0.740	*2*	1.105	*2*	**0.665**	*2*
M8	0.702*	*2*	0.714	*2*	1.595	*2*	**0.616***	*2*

(d) $h = 24$

	PC		PLS (a)		PLS (b)		PLS (c)	
M2	0.700	*3*	0.677	*2*	1.001	*2*	**0.666**	*2*
M3	**0.638**	*3*	0.775	*2*	1.523	*2*	0.672	*1*
M6	0.696	*3*	0.675	*2*	1.042	*2*	**0.665**	*2*
M8	**0.749**	*5*	0.897	*2*	1.807	*2*	0.803	*1*

Note: Each pair of columns contains the best RMSE with its number of factors. Bold entries highlight the best model in each row, and an asterisk indicates an improvement over the equivalent basic model.

Several comments are in order. Conspicuously, the findings are more sparse than for other model configurations. In fact, a line can be drawn between the short-term and long-term forecasting performance. Improvements are indeed mostly found in Tables 6a and 6b. This can be interpreted in a variety ways. Computational difficulty can be seen as a cause for the inferior performance of higher-order horizon forecasting. Indeed, the computational difficulty of the procedure followed particular to small time horizons results from the relatively limited time dimension of the estimation sample, which is critical to deriving the right transition probabilities as fewer observations are available for a higher h. Also, some of the steps clearly appear more suitable to $h = 1$ and closest values, such as the continuous update of the state vector based on the quality of forecasts made, which can be expected to best fit one-step ahead forecasting.

Another interpretation of this term discrepancy however yields interesting economic insight. In the method applied, it can be expected that in case of relevance of the two-state factor and inflation processes, poorer forecasts tend to be obtained the higher the probability of regime switches to take place within the time horizon. This is inevitably true in light of the shape of estimated state vectors, consistently displaying a relative stability. Given the average regime duration of a few years' time, such events are likely to happen, while the method is not able to easily locate such turning point exactly. In contrast, in this instance of state space, the short-term forecasting, which anticipates almost exclusively same-state transitions given the very low cross-state probabilities, the segregation of two types of processes appear to be very beneficial to the out-of-sample estimation.

6 Conclusion

In this paper, we focus on the forecasting performance of large-N PC and PLS factor models and investigate possible improvements. Widely studied in the literature, both static and dynamic versions are implemented. The dynamic approach to the latter, supervised dimensionality reduction technique is of particular relevance in this framework as we deal with time series inflation forecasts. In this paper, we consider extensions of these models to allow for time instability of the factor loadings, which have scarcely been applied empirically in the literature while macroeconomic factor models at hand have extensively been studied and found to be characterised by non-constant weights. Specifically, the extensions consider the existence of sharp changes in the factors loadings through regimes, either sequential through a structural break or alternating through two-state Markov regime switches.

The main finding made throughout the paper is the relevance and usefulness of accounting for time variance in such manner. Supporting a large diversity of research conducted in this context, both of the extensions first lead to the conclusion that the time instability is well present in the PC as well as PLS factor structure of inflation. The significance is established through both a structural break test and the convergence of the Markov state space estimation. Beyond these confirmatory observations, we find that incorporating this time instability into the factor estimation and forecasting model do lead to improved out-of-sample performance.

Besides, while the significance of the break appears largely superior in the PC factor loadings, possibly due to the different economic nature of the variance it aims to capture, improvements of extended models over basic ones are not only found for this common factor estimation technique. As opposed to the clear-cut robustness of PLS analysis to structural breaks argued for in the literature, we observe consistent drastic benefits from modelling an optimal structural break in PLS models. Interestingly, a dynamic implementation in which the target lags are only used in the factor construction exhibits very strong time instability. From this, it follows that the non-constancy of the relationship between the current inflation and its lagged values is of primary importance. Apart from this fact, no notable difference is observed between the relative improvements obtained with static versions as compared to dynamic ones throughout the analysis.

Most remarkably, the findings across the different configurations of the factor models appear to be complementary. The investigation of the out-of-sample performance of basic models suggests that both PC and PLS models capture the long-term factor structure of inflation best, and extended models show that potential structural changes are critical in this context. On the other hand, the analysis of Markov regime-switching models shows that short-term forecasting models best gain from the extension. In addition, the state space defined in the estimation exhibits slow cyclical regime switches that most often last for a few years. The models can therefore be seen as most useful in conjunction with each other, where the latter is used for short-term forecasting and the former focuses on precisely identifying the structural changes to be encountered in the longer run. Not only is their relevance complementary, but their implications might also be; indeed, there might exist a few distant structural breaks that align with the state spaces estimated.

A couple of limitations of the proposed methods are investigated here. Critically, doubt could be cast on whether the time-varying nature of the factor structure really is the driver of the improved results obtained. Indeed, creating a forecasting model that allows for two separate processes and from which the best is picked boils down to allowing for more flexibility in the model. However, two arguments are put forward to counter this. First, the inclusion of more parameters does not guarantee better results; forecasts obtained from both models might well be worse than those of a single, better-trained model if correctly specified. In view of the relative small size of the estimation samples analysed, however, obtaining improved out-of-sample

performance strongly supports the better specification of extended models. Second, the breakpoint dates and state space are not determined by any arbitrary choice but respectively tested and found to converge in a factor estimation framework. Only if the significance of modelling two separate processes was left out, such limitation would arise.

Besides, the procedures followed in this paper for taking into account the time variance of parameters is restricted to a static approach. A "dynamic" version of these extended dynamic factor models could be interesting to investigate; the structural break interval would progress in parallel with the expanding window and the state vector would be fully re-estimated in each iteration. In our case, the break test interval is fixed and contained in the estimation sample, and initially unknown states are determined by the forecasting performance instead of by re-estimation of the state space over time. The dynamic approach to these extensions, if relevant, would be completely different as the time instability found in one period could potentially conflict with previous determination. In this sense, the static approach makes intuitive sense as we essentially stick to a certain type of time period clustering once estimated, but the dynamic alternative has more potential as such clustering may well evolve over time and not hold over the long forecasting samples considered.

To conclude this paper, we offer some suggestions for further research. Related to the complementarity of the extensions considered, research could be conducted to compared the estimation outputs from both extensions, instead of solely linking back the forecast results to those of the basic models. Besides, using a higher time dimension for such analysis is also of great interest. Among others, this would enable for more breaks to be considered as well as more Markov regimes. Finally, the research could be broadened to a greater diversity of dynamic PC and PLS models; in particular, considering popular models in which the loadings themselves follow an AR model supplemented by a stochastic process would be of great interest in this analysis.

References

Andrews, D. W. (1993). Tests for parameter instability and structural change with unknown change point: A corrigendum. *Econometrica*, *61*(4), 821-856.

Bai, J., & Ng, S. (2002). Determine the number of factors in approximate factor models. *Econometrica*, *70*(1), 191-221.

Bai, J., & Ng, S. (2008). Forecasting economic time series using targeted predictors. *Journal of Econometrics*, *146*(2), 304-317.

Bair, E., Hastie, T., Paul, D., & Tibshirani, R. (2006). Prediction by supervised principal components. *Journal of the American Statistical Association*, *101*(473), 119-137.

Bates, B. J., Plagborg-Møller, M., Stock, J. H., & Watson, M. W. (2013). Consistent factor estimation in dynamic factor models with structural instability. *Journal of Econometrics*, *177*(2), 289-304.

Boivin, J., & Ng, S. (2005). Understanding and comparing factor-based forecasts. *International Journal of Central Banking*, *1*(3), 117-151.

Breitung, J., & Eickmeier, S. (2011). Testing for structural breaks in dynamic factor models. *Journal of Econometrics*, *163*(1), 71-84.

Burns, A. F., & Mitchell, W. C. (1946). Measuring business cycles. *National Bureau of Economic Research*.

Chang, J., Guo, B., & Yao, Q. (2015). High dimensional stochastic regression with latent factors, endogeneity and nonlinearity. *Journal of Econometrics*, *189*(2), 297-312.

Chen, L., Dolado, J. J., & Gonzalo, J. (2014). Detecting big structural breaks in large factor models. *Journal of Econometrics*, *180*(1), 30-48.

Chin, W. W. (1995). Partial least squares is to LISREL as principal components analysis is to common factor analysis. *Technology studies*, *2*(2), 315-319.

Chow, S. M., Zu, J., Shifren, K., & Zhang, G. (2011). Dynamic factor analysis models with time-varying parameters. *Multivariate Behavioral Research*, *46*(2), 303-339.

De Mol, C., Giannonea, D., & Reichlin, L. (2008). Forecasting using a large number of predictors: Is Bayesian shrinkage a valid alternative to principal components? *Journal of Econometrics*, *146*(2), 318-328.

Diebold, F. X., & Rudebusch, G. D. (1996). Measuring business cycles: A modern perspective. *The Review of Economics and Statistics*, *78*(1), 67-77.

Doz, C., Giannone, D., & Riechlin, L. (2006). A quasi maximum likelihood approach for large approximate dynamic factor. *Review of Economics and Statistics*, *94*(4), 1014-1024.

Forni, M., Hallin, M., Lippi, M., & Reichlin, L. (2000). The generalized dynamic-factor model: Identification and estimation. *Review of Economics and Statistics*, *82*(4), 540-554.

Fuentes, J., Poncela, P., & Rodriguez, J. (2015). Sparse partial least squares in times series for macroeconomic forecasting. *Journal of Applied Econometrics*, *30*(4), 576-595.

Geladi, P., Isaksson, H., Lindqvist, L., Wold, S., & Esbensen, K. (1989). Principal component analysis of multivariate images. *Chemometrics and Intelligent Laboratory Systems*, *5*(3), 209-220.

Geweke, J. (1977). The dynamic factor analysis of economic time series. *Latent variables in socio-economic models*.

Giannone, D., Reichlin, L., & Sala, L. (2004). Monetary policy in real time. *NBER Macroeconomics Annual*, *19*, 161-200.

Guidolin, M. (2016). *Modelling, estimating and forecasting financial data under regime (Markov) switching.* Retrieved from `http://didattica.unibocconi.it/mypage/dwload.php?nomefile=Lecture_7_`

-_Markov_Switching_Models20130520235704.pdf

Guérin, P., Leiva-Leon, D., & Marcellino, M. (2018). Markov-switching three-pass regression filter. *Journal of Business Economic Statistics*, 1-18.

Haaland, D. M., & Thomas, E. V. (1988). Partial least-squares methods for spectral analyses. 1. relation to other quantitative calibration methods and the extraction of qualitative information. *Analytical chemistry*, *60*(11), 1193-1202.

Hamilton, J. D. (1989). A new approach to the economic analysis of nonstationary time series and the business cycle. *Econometrica*, *57*(2), 357-384.

Han, X., & Inoue, A. (2015). Tests for parameter instability in dynamic factor models. *Econometric Theory*, *31*(5), 1117-1152.

Jagannathan, R., & Wang, Z. (1996). The conditional CAPM and the cross-section of expected returns. *The Journal of finance*, *51*(1), 3-53.

Kim, C. J., & Nelson, C. R. (1998). Business cycle turning points, a new coincident index, and tests of duration dependence based on a dynamic factor model with regime switching. *Review of Economics and Statistics*, *80*(2), 188-201.

Kuan, C. M. (2002). *Lecture on the Markov switching model.* Retrieved from `http://homepage.ntu.edu.tw/~ckuan/pdf/Lec-Markov_note_spring%202010.pdf`

Lam, C., & Yao, Q. (2012). Factor modeling for high-dimensional time series: inference for the number of factors. *The Annals of Statistics*, *40*(2), 694-726.

Lam, C., Yao, Q., & Bathia, N. (2011). Estimation of latent factors for high-dimensional time series. *Biometrika*, *98*(4), 901-918.

Liu, X., & Chen, R. (2016). Regime-switching factor models for high-dimensional time series. *Statistica Sinica*, 1427-1451.

Lo, M. C., & Piger, J. (2005). Is the response of output to monetary policy asymmetric? evidence from a regime-switching coefficients model. *Journal of Money, Credit and Banking*, 865-886.

Ma, S., & Su, L. (2018). Estimation of large dimensional factor models with an unknown number of breaks. *Journal of Econometrics*, *207*(1), 1-29.

Maitra, S., & Yan, J. (2008). Principle component analysis and partial least squares: Two dimension reduction techniques for regression. *Applying Multivariate Statistical Models*, *79*, 79-90.

Matheson, T. D. (2006). Factor model forecasts for New Zealand. *International Journal of Central Banking*, *2*(2), 169-237.

Matteson, D. S., & Tsay, R. S. (2011). Dynamic orthogonal components for multivariate time series. *Journal of the American Statistical Association*, *106*(496), 1450-1463.

Sargent, T. J., & Sims, C. A. (1977). Business cycle modeling without pretending to have too much a priori economic theory. *New methods in business cycle research*, *1*, 145-168.

Stock, J. H. (2003). The econometric analysis of business cycles. *Medium Econometrische Toepassingen*, *11*, 23-26.

Stock, J. H., & Watson, M. W. (1996). Evidence on structural instability in macroeconomic time series relations. *Journal of Business & Economic Statistics*, *14*(1), 11-30.

Stock, J. H., & Watson, M. W. (1998). *Diffusion indexes.* NBER Working Paper No. 6702. Retrieved from `https://www-nber-org.eur.idm.oclc.org/papers/w6702.pdf`

Stock, J. H., & Watson, M. W. (2002a). Forecasting using principal components from a large number of predictors. *Journal of the American statistical association*, *97*(460), 1167-1179.

Stock, J. H., & Watson, M. W. (2002b). Macroeconomic forecasting using diffusion indexes. *Journal of Business & Economic Statistics*, *20*(2), 147-162.

Stock, J. H., & Watson, M. W. (2005). *Implications of dynamic factor models for VAR analysis.* NBER Working Paper No. 11467. Retrieved from `https://www-nber-org.eur.idm.oclc.org/papers/w11467.pdf`

Stock, J. H., & Watson, M. W. (2009). Forecasting in dynamic factor models subject to structural instability. *The Methodology and Practice of Econometrics. A Festschrift in Honour of David F. Hendry*, 173-205.

Stock, J. H., & Watson, M. W. (2010). *Dynamic factor models.* Retrieved from `https://dash.harvard.edu/bitstream/handle/1/28469541/dynamicfactormodels_0_0.pdf?sequence=1`

Van Norden, S., & Vigfusson, R. J. (1996). *Regime-switching models: A guide to the Bank of Canada Gauss procedures.* Bank of Canada Working Paper 96-3. Retrieved from `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.2.3391&rep=rep1&type=pdf`

Wold, H. (1966). Estimation of principal components and related models by iterative least squares. *Multivariate Analysis*, 391-420.

Yamamoto, Y., & Tanaka, S. (2015). Testing for factor loading structural change under common breaks. *Journal of Econometrics*, *189*(1), 187-206.

A Appendix

A.1 Structural Break

A.1.1 Test Statistic

In relation with Aaa used in the Wald statistic of the structural break test from Han and Inoue (2015), the covariance matrix Sss is defined here. First, we introduce the following functions:

$$\hat{\mathbf{\Omega}}_1(Q, \hat{\mathbf{F}}_T) = \hat{\mathbf{\Gamma}}_{1,0}(Q, \hat{\mathbf{F}}_T) + \sum_{j=1}^{Q-1} \kappa\left(\frac{j}{S_{(Q)}}\right) (\hat{\mathbf{\Gamma}}_{1,j}(Q, \hat{\mathbf{F}}_T) + \hat{\mathbf{\Gamma}}_{1,j}(Q, \hat{\mathbf{F}}_T)') \tag{14}$$

$$\hat{\mathbf{\Omega}}_2(Q, \hat{\mathbf{F}}_T) = \hat{\mathbf{\Gamma}}_{2,0}(Q, \hat{\mathbf{F}}_T) + \sum_{j=1}^{T-Q-1} \kappa\left(\frac{j}{S_{(T-Q)}}\right) (\hat{\mathbf{\Gamma}}_{2,j}(Q, \hat{\mathbf{F}}_T) + \hat{\mathbf{\Gamma}}_{2,j}(Q, \hat{\mathbf{F}}_T)') \tag{15}$$

$$\hat{\mathbf{\Gamma}}_{1,j}(Q, \hat{\mathbf{F}}_T) = \frac{1}{Q} \sum_{t=j+1}^{Q} \text{vec}(\hat{\mathbf{f}}_t \hat{\mathbf{f}}_t' - \boldsymbol{I}_k)\ \text{vec}(\hat{\mathbf{f}}_{t-j} \hat{\mathbf{f}}_{t-j}' - \boldsymbol{I}_k)' \tag{16}$$

$$\hat{\mathbf{\Gamma}}_{2,j}(Q, \hat{\mathbf{F}}_T) = \frac{1}{T-Q} \sum_{t=j+Q+1}^{T} \text{vec}(\hat{\mathbf{f}}_t \hat{\mathbf{f}}_t' - \boldsymbol{I}_k)\ \text{vec}(\hat{\mathbf{f}}_{t-j} \hat{\mathbf{f}}_{t-j}' - \boldsymbol{I}_k)' \tag{17}$$

$$\mathbf{V}(Q, \hat{\mathbf{F}}_T) = \frac{1}{\pi} \hat{\mathbf{\Omega}}_1(Q, \hat{\mathbf{F}}_T) + \frac{1}{1-\pi} \hat{\mathbf{\Omega}}_2(Q, \hat{\mathbf{F}}_T) \tag{18}$$

where π is defined such that $Q = \lfloor \pi T \rfloor$, and where $\kappa\left(\frac{t}{b_{(T)}}\right)$ represents the following real-valued kernel function, called Quadratic Spectral:

$$\kappa(x) = \frac{25}{12\pi^2 x^2} \left(\frac{\sin(6\pi x/5)}{6\pi x/5} - \cos(6\pi x/5) \right) \tag{19}$$

where S is a smoothing parameter called the bandwidth $b_{(T)} = 1.3221(c * T)^{1/5}$, with the parameter c is arbitrarily set to 0.5

A.2 Markov Regime Switching

A.2.1 Density Function of the Residual Factor Component

Density function used in the Viterbi algorithm for $n = 2, \ldots, T$:

$$G_n(\mathbf{s}, \mathbf{B}_1, \mathbf{B}_2, \mathbf{X}) = \log(\pi_{s_1} f(\mathbf{B}_{s_1}' \mathbf{x}_1')) + \sum_{t=2}^{n} \log(\pi_{s_{t-1}, s_t} f(\mathbf{B}_{s_t}' \mathbf{x}_t'))) \tag{20}$$

where

$$f(\mathbf{B}_{s_t}' \mathbf{x}_t') = \frac{1}{\sqrt{(2\pi)^p |\mathbf{\Sigma}_{\mathbf{B}, s_t}|}}\ \exp\left(-\frac{(\mathbf{x}_t - \boldsymbol{\mu}_{s_t}) \mathbf{B}_{s_t} \mathbf{\Sigma}_{\mathbf{B}, s_t}^{-1} \mathbf{B}_{s_t}' (\mathbf{x}_t - \boldsymbol{\mu}_{s_t})'}{2} \right) \tag{21}$$

where p is the column dimension of $\mathbf{B}_{s_t}$ and $|\mathbf{A}|$ refers to the determinant of the matrix $\mathbf{A}$, and where the autocovariance matrix of $\mathbf{B}, s_t$ is estimated as follows:

$$\hat{\mathbf{\Sigma}}_{\mathbf{B}, k} = \frac{\sum_{t=1}^{T} \hat{\mathbf{B}}_k' (\mathbf{x}_t - \boldsymbol{\mu}_k)' (\mathbf{x}_t - \boldsymbol{\mu}_k) \hat{\mathbf{B}}_k I(\hat{s}_t = k)}{\sum_{t=1}^{T} I(\hat{s}_t = k) - 1} \tag{22}$$

A.2.2 Convergence Criteria

In order to check for convergence of the iterative algorithm used to estimate the state vector (step 4), the following criteria are derived:

$$|\frac{G_n(\hat{\mathbf{s}}^{(r)}) - G_n(\hat{\mathbf{s}}^{(r+1)})}{G_n(\hat{\mathbf{s}}^{(r)})}| < c_1 \tag{23}$$

$$\frac{1}{2}\left(\hat{\mathbf{Q}}_1^{(r)}, \hat{\mathbf{Q}}_1^{(r+1)}) + D(\hat{\mathbf{Q}}_2^{(r)}, \hat{\mathbf{Q}}_2^{(r+1)})\right) < c_2 \tag{24}$$

where r represents the iteration of the algorithm, c_1 and c_2 are prescribed small constants set to 10^{-5} and

$$D(\mathbf{H}_1, \mathbf{H}_2) = \sqrt{1 - \frac{1}{\max(d_1, d_2)}\text{tr}(\mathbf{H}_1\mathbf{H}_1'\mathbf{H}_2\mathbf{H}_2')} \tag{25}$$

with d_1 and d_2 being the column dimensions of $\mathbf{H}_1$ and $\mathbf{H}_2$ respectively.

A.3 Code

The entire code used to derive the results presented throughout the paper is included in this last section of the Appendix. After importing the data series (see Section 3 for details and accessibility), the *Data2003.m* should first be run to store the data after performing the relevant transformations; a variety of functions have been created to that effect: *firstDiff.m*, *secondDiff.m*, *lnFirstDiff.m* and *lnSecondDiff.m*.

The other, core *main.m* script then manipulates the series in the adequate manner to a set of parameters and allows to run the four basic models with all settings of samples and forecast horizons, possibly extended with a structural break or Markov regime switches, as well as the benchmark model to be compared to. The following structure best outlines the use of each program in relation to the extensions:

- Extension 1: Structural break
  - *find SB.m*: Algorithm that serves to find the best breakpoint location and investigate its significance for each model run
    - *Omega.m*: set of complex formulas involved in the derivation the covariance matrix of the **A** vector the statistic is based on
    - *Gamma.m* similar to the above and called in *Omega.m*
    - *kernel.m* similar the above and called in *Gamma.m*
- Extension 2: Markov regime switches
  - *find MS.m*: Algorithm that aims to estimate the optimal state vector for the three distinct estimation samples considered
    - *Gsum.m*: used in Viterbi algorithm to pick most likely path through regimes
    - *logf.m*: similar to the above and called in *Gsum.m*
    - *D.m*: used to check for convergence of iterative search for optimal state vector

A.3.1 Data importation and transformations

```
% Import data manually and run this script the first time
% Code tranforms series contained in the information set

dataSet=table2array(Database2003(:,2:end)); % array without date column

infoSet=dataSet(13:end,:); % removing observations before Jan 1960
infoSetTransfo=["lnFirstDiff"   "lnFirstDiff"   "lnFirstDiff"   "lnFirstDiff"   "lnFirstDiff" ...
    "lnFirstDiff"   "lnFirstDiff"   "lnFirstDiff"   "lnFirstDiff"   "lnFirstDiff" ...
    "lnFirstDiff"   "lnFirstDiff"   "lnFirstDiff"   "lnFirstDiff"   "lnFirstDiff" ...
    "lnFirstDiff"   "lnFirstDiff"   "lnFirstDiff"   "lv"    "firstDiff" "firstDiff" ...
    "firstDiff" "lnFirstDiff"   "lnFirstDiff"   "firstDiff" "firstDiff" "lnFirstDiff" ...
    "lnFirstDiff"   "lnFirstDiff"   "lnFirstDiff"   "lnFirstDiff"   "lnFirstDiff" ...
    "lnFirstDiff"   "lnFirstDiff"   "lnFirstDiff"   "lnFirstDiff"   "lnFirstDiff" ...
    "lnFirstDiff"   "lnFirstDiff"   "lnFirstDiff"   "lnFirstDiff"   "lnFirstDiff" ...
    "lnFirstDiff"   "lnFirstDiff"   "lnFirstDiff"   "lnFirstDiff"   "lv"    "firstDiff" ...
    "lv"    "lv"    "ln"    "ln"    "ln"    "ln"    "ln"    "ln"    "ln"    "ln"    "ln" ...
    "ln"    "lv"    "lv"    "lv"    "lv"    "lnFirstDiff"   "lnFirstDiff"   "lnFirstDiff"  ...
    "lnFirstDiff"   "lnFirstDiff"   "firstDiff" "lnSecondDiff"  "lnSecondDiff"  "lnSecondDiff" ...
    "lnFirstDiff"   "lnSecondDiff"  "lnSecondDiff"  "lnSecondDiff"  "lnSecondDiff"  "lv" ...
    "lnSecondDiff"  "firstDiff" "lnFirstDiff"   "lnFirstDiff"   "firstDiff" "lnFirstDiff" ...
    "firstDiff" "firstDiff" "firstDiff" "firstDiff" "firstDiff" "firstDiff" "firstDiff" ...
    "firstDiff" "firstDiff" "lv"    "lv"    "lv"    "lv"    "lv"    "lv"    "lv"    "lv" ...
    "lnFirstDiff"   "lnFirstDiff"   "lnFirstDiff"   "lnFirstDiff"   "lnFirstDiff" ...
    "lnSecondDiff"  "lnSecondDiff"  "lnSecondDiff"  "lnSecondDiff"  "lnSecondDiff" ...
    "lnSecondDiff"  "lv"    "lnSecondDiff"  "lnSecondDiff"  "lnSecondDiff"  "lnSecondDiff" ...
    "lnSecondDiff"  "lnSecondDiff"  "lnSecondDiff"  "lnSecondDiff"  "lnSecondDiff" ...
    "lnSecondDiff"  "lnSecondDiff"  "lnSecondDiff"  "lnSecondDiff"  "lnSecondDiff" ...
    "lnSecondDiff"  "lnSecondDiff"  "lnSecondDiff"  "firstDiff"];

for col=1:size(infoSet,2)
    if infoSetTransfo(col)=="ln"
        infoSet(:,col)=log(infoSet(:,col));
    elseif infoSetTransfo(col)=="firstDiff"
        infoSet(:,col)=firstDiff(infoSet(:,col));
    elseif infoSetTransfo(col)=="lnFirstDiff"
        infoSet(:,col)=lnFirstDiff(infoSet(:,col));
    elseif infoSetTransfo(col)=="lnSecondDiff"
        infoSet(:,col)=lnSecondDiff(infoSet(:,col));
    end
end

infoSet=infoSet(3:end,:); % balancing sample after losing up to first two observations

targetSeries=dataSet(:,115);

save('Data2003.mat','infoSet','targetSeries');
clear
```

```
function B = firstDiff(A)
```

```
% FIRSTDIFF Changes elements in vector A to correspond to first differences
% and adjusts length

n=length(A);
B=NaN(n,1);
for i=2:n
    B(i)=A(i)-A(i-1);
end

end
```

```
function C = secondDiff(A)
% LNSECONDDIFF Changes elements in vector A to correspond to second
% differences of ln and adjusts length

n=length(A);
B=[A NaN(n,1) NaN(n,1)];
j=1;
while j<3
    for i=1+j:n
        B(i,j+1)=B(i,j)-B(i-1,j);
    end
    j=j+1;
end
C=B(:,3);
end
```

```
function C = lnFirstDiff(A)
% LNFIRSTDIFF Changes elements in vector A to correspond to first
% differences of ln and adjusts length

n=length(A);
B=log(A);
C=NaN(n,1);
for i=2:n
    C(i)=B(i)-B(i-1);
end

end
```

```
function C = lnSecondDiff(A)
% LNSECONDDIFF Changes elements in vector A to correspond to second
% differences of ln and adjusts length

n=length(A);
B=[log(A) NaN(n,1) NaN(n,1)];
j=1;
while j<3
    for i=1+j:n
        B(i,j+1)=B(i,j)-B(i-1,j);
```

```
    end
    j=j+1;
end
C=B(:,3);
end
```

A.3.2 Main script and models

```
% Load saved data file (infoSet, targetSeries) and set relevant indices
% for subsamples, forecast horizons, SB test interval, MS samples and
% correspondence
load Data2003v2.mat
sampleSet=[0 121 250;0 241 370;0 361 490;0 121 370;0 121 490;0 241 490;0 121 526;120 361 490];
h_all=[1,6,12,24];
QSetSB=[0 0;25 121;25 241;0 0;0 0;25 121;0 0;145 241];
MSsampleSet=[2,3,8];
MSrefSet=[0; 1; 2; 0; 0; 1; 0; 3];

% Set parameters: maximum number of factors in each model
k_PC_max=10;
k_plsA_max=2;
k_plsB_max=2;
k_plsC_max=2;

% Choose extension (0 for basic models, 1 for SB, 2 for MS)
extension=0;


% ----------------------------------------------------------------


% MS: derive state vector for the three relevant samples (dependent on h)
if extension==2
    Sresults=NaN(526,3,4); % 526 max length of sample, 3 MS samples, 4 h
    Pi0results=NaN(2,3,4); % 2 unconditional probabilities
    Piresults=NaN(2,2,3,4); % 2x2 transition probabilities

    for subsample=1:8

        MSref=MSrefSet(subsample);
        % Detect and skip smallest samples
        if MSref==0
            continue
        end

        % Use relevant MS sample among three to set up variables
        MSsample=MSsampleSet(MSref);

        cutStart=sampleSet(MSsample,1);
        fctlb=sampleSet(MSsample,2);
        fctub=sampleSet(MSsample,3);
```

```
        for h_index=1:4

            h=h_all(h_index);
            X=infoSet(1+cutStart:fctlb-h,:);

            % Find state vector and probabilities and store them
            [S,Pi0,Pi] = find_MS(X,h);
            Sresults(1+cutStart:fctlb-h,MSref,h_index)=S;
            Pi0results(:,MSref,h_index)=Pi0;
            Piresults(:,:,MSref,h_index)=Pi(:,:,2); % only need h-order transition probabilities

        end
    end
end


% ------------------------------------------------------------------


% Run all models for all settings and store following results
results=NaN(8,4,4,4); % 8 subsamples, 4 models, 4 for mse-k-Q0(ext1)-Wbest(ext1), 4 horizons
y_results=NaN(526,4,8,4); % 526 max time index, 4 models, 8 subsamples, 4 horizons

for h_index=1:4

    h=h_all(h_index);

    for subsample=1:8

        MSref=MSrefSet(subsample);
        % Detect and skip smallest samples for extensions 1 and 2
        if extension~=0
            if MSref==0
                continue
            end
        end

        % Set sample-specific variables
        cutStart=sampleSet(subsample,1);
        fctlb=sampleSet(subsample,2);
        fctub=sampleSet(subsample,3);
        Q1andQ2=QSetSB(subsample,:)-cutStart; % reindex sample (8) to start at one

        % Build X - InfoSet has 526 rows after manipulations
        X=infoSet(1+cutStart:fctub-h,:);

        % Build y - TargetSet has 540 rows, series have not been transformed
        z=infoSet(1+cutStart:fctub,115);
        y_log=log(targetSeries(1+cutStart:end));
        fctlb=fctlb-cutStart; % reindex sample (8) bounds to start at one in method
        fctub=fctub-cutStart;
        y_big=1200/h*(y_log(h+2:fctub+14)-y_log(2:fctub+14-h))-1200*(y_log(2:fctub+14-h)- ...
            y_log(1:fctub+14-h-1));
        y_big=[NaN(h+1,1);y_big];
```

```
    y=y_big(15:end);

    % Compute factors with approach a-b-c
    mse_AR4=run_AR4(z,y,fctlb,fctub,h);


% Run PC for all k depending on extension chosen and store results
    mse_PC_all=NaN(k_PC_max,1);
    y_est_PC_all=NaN(fctub,k_PC_max);
    if extension==0
        for k_PC=1:k_PC_max
            [mse_PC_all(k_PC),y_est_PC_all(:,k_PC)]=run_PC(z,y,X,fctlb,fctub,h_index, ...
                k_PC,extension);
        end
    elseif extension==1
        Q0_all=NaN(k_PC_max,1);
        Wbest_all=NaN(k_PC_max,1);
        for k_PC=1:k_PC_max
            [mse_PC_all(k_PC),y_est_PC_all(:,k_PC),Q0_all(k_PC),Wbest_all(k_PC)]= ...
                run_PC(z,y,X,fctlb,fctub,h_index,k_PC,extension,Q1andQ2);
        end
    elseif extension==2
        for k_PC=1:k_PC_max
            [mse_PC_all(k_PC),y_est_PC_all(:,k_PC)]=run_PC(z,y,X,fctlb,fctub,h_index, ...
                k_PC,extension,[],[],[],Sresults(1+cutStart:fctlb+cutStart-h,MSref, ...
                h_index),Piresults(:,:,MSref,h_index));
        end
    end

    [mse_PC,best]=min(mse_PC_all);
    results(subsample,1,2,h_index)=best;
    results(subsample,1,1,h_index)=mse_PC/mse_AR4;
    y_results(1+cutStart:fctub+cutStart,1,subsample,h_index)=y_est_PC_all(:,best);
    if extension==1
        results(subsample,1,3,h_index)=Q0_all(best)+cutStart;
        results(subsample,1,4,h_index)=Wbest_all(best);
    end


% Run PLS A for all k depending on extension chosen and store results
    mse_plsA_all=NaN(k_plsA_max,1);
    y_est_plsA_all=NaN(fctub,k_plsA_max);
    if extension==0
        for k_plsA=1:k_plsA_max
            [mse_plsA_all(k_plsA),y_est_plsA_all(:,k_plsA)]=run_plsA(z,y,X,fctlb,fctub, ...
                h_index,k_plsA,extension);
        end
    elseif extension==1
        Q0_all=NaN(k_plsA_max,1);
        Wbest_all=NaN(k_plsA_max,1);
        for k_plsA=1:k_plsA_max
            [mse_plsA_all(k_plsA),y_est_plsA_all(:,k_plsA),Q0_all(k_plsA), ...
                Wbest_all(k_plsA)]=run_plsA(z,y,X,fctlb,fctub,h_index,k_plsA, ...
                extension,Q1andQ2);
```

```
        end
    elseif extension==2
        for k_plsA=1:k_plsA_max
            [mse_plsA_all(k_plsA),y_est_plsA_all(:,k_plsA)]=run_plsA(z,y,X,fctlb,fctub, ...
                h_index,k_plsA,extension,[],[],[],Sresults(1+cutStart:fctlb+cutStart-h, ...
                MSref,h_index),Piresults(:,:,MSref,h_index));
        end
    end

    [mse_plsA,best]=min(mse_plsA_all);
    results(subsample,2,2,h_index)=best;
    results(subsample,2,1,h_index)=mse_plsA/mse_AR4;
    y_results(1+cutStart:fctub+cutStart,2,subsample,h_index)=y_est_plsA_all(:,best);
    if extension==1
        results(subsample,2,3,h_index)=Q0_all(best)+cutStart;
        results(subsample,2,4,h_index)=Wbest_all(best);
    end


% Run PLS B for all k depending on extension chosen and store results
    mse_plsB_all=NaN(k_plsB_max,1);
    y_est_plsB_all=NaN(fctub,k_plsB_max);
    if extension==0
        for k_plsB=1:k_plsB_max
            [mse_plsB_all(k_plsB),y_est_plsB_all(:,k_plsB)]=run_plsB(z,y,X,fctlb,fctub, ...
                h_index,k_plsB,extension);
        end
    elseif extension==1
        Q0_all=NaN(k_plsB_max,1);
        Wbest_all=NaN(k_plsB_max,1);
        for k_plsB=1:k_plsB_max
            [mse_plsB_all(k_plsB),y_est_plsB_all(:,k_plsB),Q0_all(k_plsB), ...
                Wbest_all(k_plsB)]= run_plsB(z,y,X,fctlb,fctub,h_index,k_plsB, ...
                extension,Q1andQ2);
        end
    elseif extension==2
        for k_plsB=1:k_plsB_max
            [mse_plsB_all(k_plsB),y_est_plsB_all(:,k_plsB)]=run_plsB(z,y,X,fctlb,fctub, ...
                h_index,k_plsB,extension,[],[],[],Sresults(1+cutStart:fctlb+cutStart-h, ...
                MSref,h_index),Piresults(:,:,MSref,h_index));
        end

    end

    [mse_plsB,best]=min(mse_plsB_all);
    results(subsample,3,2,h_index)=best;
    results(subsample,3,1,h_index)=mse_plsB/mse_AR4;
    y_results(1+cutStart:fctub+cutStart,3,subsample,h_index)=y_est_plsB_all(:,best);
    if extension==1
        results(subsample,3,3,h_index)=Q0_all(best)+cutStart;
        results(subsample,3,4,h_index)=Wbest_all(best);
    end
```

```
    % Run PLS C for all k depending on extension chosen and store results
        mse_plsC_all=NaN(k_plsC_max,1);
        y_est_plsC_all=NaN(fctub,k_plsC_max);
        if extension==0
            for k_plsC=1:k_plsC_max
                [mse_plsC_all(k_plsC),y_est_plsC_all(:,k_plsC)]=run_plsC(z,y,X,fctlb,fctub, ...
                    h_index,k_plsC,extension);
            end
        elseif extension==1
            Q0_all=NaN(k_plsC_max,1);
            Wbest_all=NaN(k_plsC_max,1);
            for k_plsC=1:k_plsC_max
                [mse_plsC_all(k_plsC),y_est_plsC_all(:,k_plsC),Q0_all(k_plsC), ...
                    Wbest_all(k_plsC)]=run_plsC(z,y,X,fctlb,fctub,h_index,k_plsC, ...
                    extension,Q1andQ2);
            end
        elseif extension==2
            for k_plsC=1:k_plsC_max
                [mse_plsC_all(k_plsC),y_est_plsC_all(:,k_plsC),]=run_plsC(z,y,X,fctlb,fctub, ...
                    h_index,k_plsC,extension,[],[],[],Sresults(1+cutStart:fctlb+cutStart-h, ...
                    MSref,h_index),Piresults(:,:,MSref,h_index));
            end
        end

        [mse_plsC,best]=min(mse_plsC_all);
        results(subsample,4,2,h_index)=best;
        results(subsample,4,1,h_index)=mse_plsC/mse_AR4;
        y_results(1+cutStart:fctub+cutStart,4,subsample,h_index)=y_est_plsC_all(:,best);
        if extension==1
            results(subsample,4,3,h_index)=Q0_all(best)+cutStart;
            results(subsample,4,4,h_index)=Wbest_all(best);
        end

    end

end
```

```
function [mse,y_est] = run_AR4(z,y,fctlb,fctub,h)
% RUN_AR4 Returns MSE and vector of estimated y's in the forecast sample
% for the benchmark model AR(4)

n_lags=4;
y_est=NaN(fctub,1);

for i=fctlb:fctub

    % Build lagged y's
    z_reg=NaN(i-h,n_lags);
    for j=0:n_lags-1
        z_reg(j+1:i-h,j+1)=z(1:i-h-j);
    end
    predictors=[ones(i-h,1) z_reg];
```

```
    % Derive next forecast
    b=regress(y(h+1:i-h),predictors(1:i-2*h,:));
    y_est(i)=predictors(i-h,:)*b;

end

T=fctub-fctlb+1;
mse=1/T*sum((y_est(fctlb:fctub)-y(fctlb:fctub)).^2);

end
```

```
function [mse,y_est_output,Q0,Wbest] = run_PC(z,y,X,fctlb,fctub,h_index,k,extension, ...
    Q1andQ2,Q0,Wbest,S,Pi)
% RUN_PC Returns MSE and vector of estimated y's in the forecast sample
% for PC approach

h_all=[1,6,12,24];
h=h_all(h_index);

n_lags_y_max=6;
n_lags_Z_max=6;

% Define two-state space proper to extension 2 and enlarged y output needed as a result
if extension==2
    states=[1;2];
    y_est_missing=NaN(h-1,2);
else
    states=1;
end
y_est=NaN(fctub,sum(states));

% Start factor estimation and forecasting process with expanding window
for i=fctlb:fctub

    if extension==2
        % Use newly-available information to define past state i-h (based
        % on best prediction made at i-2*h) and adjust relevant transition
        % probabilities
        if i>fctlb
            fromstate=S(i-2*h);
            if i>=fctlb+h
                [~,tostate]=min([y_est(i-h,1)-y(i-h); y_est(i-h,2)-y(i-h)].^2);
            else
                [~,tostate]=min([y_est_missing(i-fctlb,1)-y(i-h); ...
                    y_est_missing(i-fctlb,2)-y(i-h)].^2);
            end
            S=[S;tostate];
            if fromstate==1
                candidates=-sum(S-2);
                Pi(1,:)=Pi(1,:)*candidates/(candidates+1);
                Pi(1,tostate)=Pi(1,tostate)+1/(candidates+1);
            else
```

```
            candidates=sum(S-1);
            Pi(2,:)=Pi(2,:)*candidates/(candidates+1);
            Pi(2,tostate)=Pi(2,tostate)+1/(candidates+1);
        end
    end

    Weight=NaN(i-h,i-h,2);
    Weight(:,:,2)=diag(S-1);
    Weight(:,:,1)=diag(-(S-2));
end

for state=states(1):states(end)

    % Prepare standardised X for factor construction
    X_state=X(1:i-h,:);
    if extension==2
        X_state=Weight(:,:,state)*X_state;
    end
    X_m=normalize(X_state);

    % Construct k factors
    M=X_m'*X_m;
    [V,D]=eig(M,'vector');
    [~,indices]=maxk(D,k);
    W=V(:,indices);
    Z=X_m*W;

    % ----------------
    if extension==1

        W = find_SB(Z,Q1andQ2);
        [~,best]=max(W);
        Q0=best+Q1;
        Wbest=W(best);

        [mse,y_est_output,Q0,Wbest]=run_PCvExtvFINAL(z(Q0:end),y(Q0:end),X(Q0:end,:), ...
            fctlb-Q0+1,fctub-Q0+1,h_index,k,0,Q1andQ2,Q0,Wbest);
        y_est_output=[NaN(Q0-1,1);y_est_output];
        return

    end
    % ----------------

    % Build largest set of regressors possible
    Z_reg=NaN(i-h,k*n_lags_Z_max);
    y_reg=NaN(i-h,n_lags_y_max);
    for j=0:n_lags_Z_max-1
        Z_reg(j+1:i-h,j*k+1:(j+1)*k)=Z(1:i-h-j,:);
    end
    for j=0:n_lags_y_max-1
        y_reg(j+1:i-h,j+1)=z(1:i-h-j);
    end

    % Find right number of lags and forecast y - n_lags allowed to vary across
```

```
        % y and Z (and allowed to vary in each iteration)
        fcts=NaN(n_lags_y_max,n_lags_Z_max);
        bics=NaN(n_lags_y_max,n_lags_Z_max);
        for n_lags_y=1:n_lags_y_max
            for n_lags_Z=1:n_lags_Z_max
                predictors=[ones(i-h,1) y_reg(:,1:n_lags_y) Z_reg(:,1:k*n_lags_Z)];
                n_lags_max=max([n_lags_y;n_lags_Z]);
                [b,~,e]=regress(y(n_lags_max+h:i-h),predictors(n_lags_max:i-2*h,:));
                fcts(n_lags_y,n_lags_Z)=predictors(i-h,:)*b;
                T=length(e);
                bics(n_lags_y,n_lags_Z)=log(1/T*e'*e)+size(predictors,2)*log(T)/T;
            end
        end
        [temp,index_y]=min(bics);
        [~,index_Z]=min(temp);
        y_est(i,state)=fcts(index_y(index_Z),index_Z);

        % Make h-1 missing h-order out-of-sample forecasts between estimation and forecast
        % samples used to derive a gap-free state vector
        if extension==2
            if i==fctlb && h>1
                n_lags_y=index_y(index_Z); % best number of lags based on relevant state portion
                                           % of full estimation sample
                n_lags_z=index_Z;
                predictors=[ones(h-1,1) y_reg(fctlb-2*h+1:fctlb-h-1,1:n_lags_y) ...
                    Z_reg(fctlb-2*h+1:fctlb-h-1,1:k*n_lags_z)];
                b=regress(y(fctlb-h+1:fctlb-1),predictors);
                y_est_missing(:,state)=predictors*b;
            end
        end

    end

    % Compute final forecast as a convex combination of the two state-specific ones
    % in extension 2
    if extension==2
        lastk=S(i-h);
        prob1=Pi(lastk,1);
        y_est(i,3)=prob1*y_est(i,1)+(1-prob1)*y_est(i,2);
    end

end

% Compute MSE
T=fctub-fctlb+1;
mse=1/T*sum((y_est(fctlb:fctub,end)-y(fctlb:fctub)).^2);
y_est_output=y_est(:,end);

end
```

```
function [mse,y_est_output,Q0,Wbest] = run_plsA(z,y,X,fctlb,fctub,h_index,k_tot,extension, ...
    Q1andQ2,Q0,Wbest,S,Pi)
% RUN_PLSA Returns MSE and vector of estimated y's in the forecast sample
```

```
% for PLS with approach A

h_all=[1,6,12,24];
h=h_all(h_index);

n_lags_y_max=6;
n_lags_Z_max=6;

% Define two-state space proper to extension 2 and enlarged y output needed as a result
if extension==2
    states=[1;2];
    y_est_missing=NaN(h-1,2);
else
    states=1;
end
y_est=NaN(fctub,sum(states));

% Start factor estimation and forecasting process with expanding window
for i=fctlb:fctub

    if extension==2
        % Use newly-available information in forecasting sample only to
        % to define past state i-h and adjust relevant transition
        % probabilities
        if i>fctlb
            fromstate=S(i-2*h);
            if i>=fctlb+h
                [~,tostate]=min([y_est(i-h,1)-y(i-h); y_est(i-h,2)-y(i-h)].^2);
            else
                [~,tostate]=min([y_est_missing(i-fctlb,1)-y(i-h); y_est_missing(i-fctlb,2)- ...
                    y(i-h)].^2);
            end
            S=[S;tostate];
            if fromstate==1
                candidates=-sum(S-2);
                Pi(1,:)=Pi(1,:)*candidates/(candidates+1);
                Pi(1,tostate)=Pi(1,tostate)+1/(candidates+1);
            else
                candidates=sum(S-1);
                Pi(2,:)=Pi(2,:)*candidates/(candidates+1);
                Pi(2,tostate)=Pi(2,tostate)+1/(candidates+1);
            end
        end

        Weight=NaN(i-h,i-h,2);
        Weight(:,:,2)=diag(S-1);
        Weight(:,:,1)=diag(-(S-2));
    end

    for state=states(1):states(end)

        % Prepare standardised X for factor construction
        X_state=X(1:i-h,:);
        y_state=y(h+1:i-h);
```

```
if extension==2
    X_state=Weight(:,:,state)*X_state;
    y_state=Weight(h+1:end,h+1:end,state)*y_state;
end
X_m=normalize(X_state(1:end-h,:));
y_m=normalize(y_state);
X_reg=normalize(X_state);

% Construct k factors
Z=NaN(i-h,k_tot);
for k=1:k_tot
    M=X_m'*y_m*y_m'*X_m;
    [V,D]=eig(M,'vector');
    [~,index]=max(D);
    w=V(:,index);
    Z(:,k)=X_reg*w; % preceding factor as regressor, using all data available
    eX=NaN(size(X_reg));
    for col=1:size(X_reg,2)
        [~,~,eX(:,col)]=regress(X_reg(:,col),Z(:,k));
    end
    [~,~,ey]=regress(y_m,Z(1:end-h,k));
    X_reg=eX;
    X_m=eX(1:end-h,:);
    y_m=ey;
end

% ----------------
if extension==1

    W = find_SB(Z,Q1andQ2);
    [~,best]=max(W);
    Q0=best+Q1;
    Wbest=W(best);

    [mse,y_est_output,Q0,Wbest]=run_plsAvExtvFINAL(z(Q0:end),y(Q0:end),X(Q0:end,:), ...
        fctlb-Q0+1,fctub-Q0+1,h_index,k_tot,0,Q1andQ2,Q0,Wbest);
    y_est_output=[NaN(Q0-1,1);y_est_output];
    return

end
% ----------------

% Build largest set of regressors possible
Z_reg=NaN(i-h,k_tot*n_lags_Z_max);
y_reg=NaN(i-h,n_lags_y_max);
for j=0:n_lags_Z_max-1
    Z_reg(j+1:i-h,j*k_tot+1:(j+1)*k_tot)=Z(1:i-h-j,:);
end
for j=0:n_lags_y_max-1
    y_reg(j+1:i-h,j+1)=z(1:i-h-j);
end

% Find right number of lags and forecast y - n_lags allowed to vary for y and Z
% and allowed to vary in each iteration
```

```
        fcts=NaN(n_lags_y_max,n_lags_Z_max);
        bics=NaN(n_lags_y_max,n_lags_Z_max);
        for n_lags_y=1:n_lags_y_max
            for n_lags_Z=1:n_lags_Z_max
                predictors=[ones(i-h,1) y_reg(:,1:n_lags_y) Z_reg(:,1:k_tot*n_lags_Z)];
                [b,~,e]=regress(y(h+1:i-h),predictors(1:i-2*h,:));
                fcts(n_lags_y,n_lags_Z)=predictors(i-h,:)*b;
                e=e(~isnan(e));
                T=length(e);
                bics(n_lags_y,n_lags_Z)=log(1/T*e'*e)+size(predictors,2)*log(T)/T;
            end
        end
        [temp,index_y]=min(bics);
        [~,index_Z]=min(temp);
        y_est(i,state)=fcts(index_y(index_Z),index_Z);

        % Make h-1 missing h-order out-of-sample forecasts between estimation and forecast
        % samples used to derive a gap-free state vector
        if extension==2
            if i==fctlb && h>1
                n_lags_y=index_y(index_Z); % best number of lags based on relevant state
                                           % portion of full estimation sample
                n_lags_z=index_Z;
                predictors=[ones(h-1,1) y_reg(fctlb-2*h+1:fctlb-h-1,1:n_lags_y) ...
                Z_reg(fctlb-2*h+1:fctlb-h-1,1:k*n_lags_z)];
                b=regress(y(fctlb-h+1:fctlb-1),predictors);
                y_est_missing(:,state)=predictors*b;
            end
        end

    end

    % Compute final forecast as a convex combination of the two state-specific ones
    % in extension 2
    if extension==2
        lastk=S(i-h);
        prob1=Pi(lastk,1);
        y_est(i,3)=prob1*y_est(i,1)+(1-prob1)*y_est(i,2);
    end

end

% Compute MSE
T=fctub-fctlb+1;
mse=1/T*sum((y_est(fctlb:fctub,end)-y(fctlb:fctub)).^2);
y_est_output=y_est(:,end);

end
```

```
function [mse,y_est_output,Q0,Wbest] = run_plsB(z,y,X,fctlb,fctub,h_index,k_tot,extension, ...
    Q1andQ2,Q0,Wbest,S_input,Pi)
% RUN_PLSB Returns MSE and vector of estimated y's in the forecast sample
% for PLS with approach B
```

```
h_all=[1,6,12,24];
h=h_all(h_index);

n_lags_y_max=6;
n_lags_Z_max=6;
r_pre=size(X,2); % pre-enlargement column dimension of X

% Extend X to include maximum number of y's and y lags
Xe=[X NaN(fctub-h,n_lags_y_max)];
for j=0:n_lags_y_max-1
    Xe(j+1:end,r_pre+j+1)=z(1:end-h-j); % adding y lags from z
end

% Define two-state space proper to extension 2 and enlarged y output needed as a result
if extension==2
    states=[1;2];
    y_est_missing=NaN(h-1,2);
else
    states=1;
end
y_est_all=NaN(fctub,n_lags_y_max,sum(states)); % output matrix with columns corresponding to each
                                               % choice of n_lags_y (from 1 to max)
mse_all=NaN(n_lags_y_max,1); % output vector with elements corresponding to each choice of
                             % n_lags_y (from 1 to max)
T_mse=fctub-fctlb+1;

% Start factor estimation and forecasting process with expanding window
for n_lags_y=1:n_lags_y_max

    % Use cut to remove unknown observations due to lags for factor construction
    cut=n_lags_y-1;

    if extension==2
        S=S_input;
    end

    for i=fctlb:fctub

        if extension==2
            % Use newly-available information in forecasting sample only to define past state
            % i-h and adjust relevant transition probabilities
            if i>fctlb
                fromstate=S(i-2*h);
                if i>=fctlb+h
                    [~,tostate]=min([y_est_all(i-h,n_lags_y,1)-y(i-h); ...
                        y_est_all(i-h,n_lags_y,2)- y(i-h)].^2);
                else
                    [~,tostate]=min([y_est_missing(i-fctlb,1)-y(i-h); ...
                        y_est_missing(i-fctlb,2)-y(i-h)].^2);
                end
                S=[S;tostate];
                if fromstate==1
                    candidates=-sum(S-2);
```

```
            Pi(1,:)=Pi(1,:)*candidates/(candidates+1);
            Pi(1,tostate)=Pi(1,tostate)+1/(candidates+1);
        else
            candidates=sum(S-1);
            Pi(2,:)=Pi(2,:)*candidates/(candidates+1);
            Pi(2,tostate)=Pi(2,tostate)+1/(candidates+1);
        end
    end

    Weight=NaN(i-h,i-h,2);
    Weight(:,:,2)=diag(S-1);
    Weight(:,:,1)=diag(-(S-2));
end

for state=states(1):states(end)

    % Prepare standardised X for factor construction
    X_state=Xe(1+cut:i-h,1:r_pre+n_lags_y);
    y_state=y(1+cut+h:i-h);
    if extension==2
        X_state=Weight(1+cut:end,1+cut:end,state)*X_state;
        y_state=Weight(1+h+cut:end,1+h+cut:end,state)*y_state;
    end
    X_m=normalize(X_state(1:end-h,:));
    y_m=normalize(y_state);
    X_reg=normalize(X_state);

    % Construct k factors
    Z=NaN(i-h-cut,k_tot);

    for k=1:k_tot
        M=X_m'*y_m*y_m'*X_m;
        [V,D]=eig(M,'vector');
        [~,index]=max(D);
        w=V(:,index);
        Z(:,k)=X_reg*w; % preceding factor as regressor, using all data available
        eX=NaN(size(X_reg));
        for col=1:size(X_reg,2)
            [~,~,eX(:,col)]=regress(X_reg(:,col),Z(:,k));
        end
        [~,~,ey]=regress(y_m,Z(1:end-h,k));
        X_reg=eX;
        X_m=eX(1:end-h,:);
        y_m=ey;
    end
    Z=[NaN(cut,k_tot);Z];

    % ----------------
    % Find best breakpoint location and re-estimate model by cutting pre-break
    % observations
    if extension==1

        W = find_SB(Z,Q1andQ2);
        [~,best]=max(W);
```

```
            Q0=best+Q1;
            Wbest=W(best);

            [mse,y_est_output,Q0,Wbest]=run_plsBvExtvFINAL(z(Q0:end),y(Q0:end), ...
                X(Q0:end,:),fctlb-Q0+1,fctub-Q0+1,h_index,k_tot,0,Q1andQ2,Q0,Wbest);
            y_est_output=[NaN(Q0-1,1);y_est_output];
            return

        end
        % ----------------

        % Build largest set of regressors possible
        Z_reg=NaN(i-h,k_tot*n_lags_Z_max);
        for j=0:n_lags_Z_max-1
            Z_reg(j+1:i-h,j*k_tot+1:(j+1)*k_tot)=Z(1:i-h-j,:);
        end

        % Find right number of lags and forecast y - n_lags set to be the same for y and Z
        % and allowed to vary in each iteration
        fcts_bics=NaN(n_lags_Z_max,2);
        for n_lags_Z=1:n_lags_Z_max
            predictors=[ones(i-h,1) Z_reg(:,1:k_tot*n_lags_Z)];
            [b,~,e]=regress(y(h+1:i-h),predictors(1:i-2*h,:));
            fcts_bics(n_lags_Z,1)=predictors(i-h,:)*b;
            e=e(~isnan(e));
            T_bic=length(e);
            fcts_bics(n_lags_Z,2)=log(1/T_bic*e'*e)+size(predictors,2)*log(T_bic)/T_bic;
        end
        [~,index]=min(fcts_bics(:,2));
        y_est_all(i,n_lags_y,state)=fcts_bics(index,1);

        % Make h-1 missing h-order out-of-sample forecasts between estimation and forecast
        % samples used to derive a gap-free state vector
        if extension==2
            if i==fctlb && h>1
                n_lags_z=index; % best number of lags based on relevant state portion of
                                % full estimation sample
                predictors=[ones(h-1,1) Z_reg(fctlb-2*h+1:fctlb-h-1,1:k*n_lags_z)];
                b=regress(y(fctlb-h+1:fctlb-1),predictors);
                y_est_missing(:,state)=predictors*b;
            end
        end

    end

    % Compute final forecast as a convex combination of the two state-specific ones
    % in extension 2
    if extension==2
        lastk=S(i-h);
        prob1=Pi(lastk,1);
        y_est_all(i,n_lags_y,3)=prob1*y_est_all(i,n_lags_y,1)+(1-prob1)* ...
            y_est_all(i,n_lags_y,2);
    end
```

```
    end

    % Compute MSE for given n_lags_y
    mse_all(n_lags_y)=1/T_mse*sum((y_est_all(fctlb:fctub,n_lags_y,end)-y(fctlb:fctub)).^2);

end

% Select optimal n_lags_y based on model with min MSE
[mse,n_lags_y_opt]=min(mse_all);
y_est_output=y_est_all(:,n_lags_y_opt,end);

end
```

```
function [mse,y_est_output,Q0,Wbest] = run_plsC(z,y,X,fctlb,fctub,h_index,k_tot,extension, ...
    Q1andQ2,Q0,Wbest,S_input,Pi)
% RUN_PLSA Returns MSE and vector of estimated y's in the forecast sample
% for PLS with approach A

h_all=[1,6,12,24];
h=h_all(h_index);

n_lags_AR=6;
n_lags_y_max=6;
n_lags_Z_max=6;

% Define two-state space proper to extension 2 and enlarged y output needed as a result
if extension==2
    states=[1;2];
    y_est_missing=NaN(h-1,2);
else
    states=1;
end
y_est_all=NaN(fctub,n_lags_AR,sum(states)); % output matrix with columns corresponding
                                            % to each choice of n_lags_y (from 1 to max)
mse_all=NaN(n_lags_AR,1); % output vector with elements corresponding to each choice of
                          % n_lags_y (from 1 to max)
T_mse=fctub-fctlb+1;

% Start factor estimation and forecasting process with expanding window
for p=1:n_lags_AR

    if extension==2
        S=S_input;
    end

    for i=fctlb:fctub

        % Use cut to remove unknown observations due to lags for factor construction
        cut=p-1;

        if extension==2
            % Use newly-available information in forecasting sample only to define past
            % state i-h and adjust relevant transition probabilities
```

```
        if i>fctlb
            fromstate=S(i-2*h);
            if i>=fctlb+h
                [~,tostate]=min([y_est_all(i-h,p,1)-y(i-h); y_est_all(i-h,p,2)-y(i-h)].^2);
            else
                [~,tostate]=min([y_est_missing(i-fctlb,1)-y(i-h); ...
                    y_est_missing(i-fctlb,2)- y(i-h)].^2);
            end
            S=[S;tostate];
            if fromstate==1
                candidates=-sum(S-2);
                Pi(1,:)=Pi(1,:)*candidates/(candidates+1);
                Pi(1,tostate)=Pi(1,tostate)+1/(candidates+1);
            else
                candidates=sum(S-1);
                Pi(2,:)=Pi(2,:)*candidates/(candidates+1);
                Pi(2,tostate)=Pi(2,tostate)+1/(candidates+1);
            end
        end

        Weight=NaN(i-h,i-h,2);
        Weight(:,:,2)=diag(S-1);
        Weight(:,:,1)=diag(-(S-2));
    end

    for state=states(1):states(end)

        % Get residuals from AR(p) to be used as y_m
        y_AR=y(1+cut:i-h);
        y_AR_reg=NaN(i-h-cut,p);
        for j=0:p-1
            y_AR_reg(:,j+1)=z(p-j:i-h-j);
        end
        pred_AR=[ones(i-h-cut,1) y_AR_reg];
        [~,~,e_AR]=regress(y_AR,pred_AR);

        % Prepare standardised X for factor construction
        X_state=X(1+cut:i-h,:);
        e_AR_state=e_AR(h+1:end);
        if extension==2
            X_state=Weight(1+cut:end,1+cut:end,state)*X_state;
            e_AR_state=Weight(1+cut+h:end,1+cut+h:end,state)*e_AR_state;
        end
        X_m=normalize(X_state(1:end-h,:));
        y_m=normalize(e_AR_state);
        X_reg=normalize(X_state);

        % Construct k factors
        Z=NaN(i-h-cut,k_tot);
        for k=1:k_tot
            M=X_m'*y_m*y_m'*X_m;
            [V,D]=eig(M,'vector');
            [~,index]=max(D);
            w=V(:,index);
```

```
    Z(:,k)=X_reg*w; % preceding factor as regressor, using all data available
    eX=NaN(size(X_reg));
    for col=1:size(X_reg,2)
        [~,~,eX(:,col)]=regress(X_reg(:,col),Z(:,k));
    end
    [~,~,ey]=regress(y_m,Z(1:end-h,k));
    X_reg=eX;
    X_m=eX(1:end-h,:);
    y_m=ey;
end
Z=[NaN(cut,k_tot);Z];

% ----------------
if extension==1

    W = find_SB(Z,Q1andQ2);
    [~,best]=max(W);
    Q0=best+Q1;
    Wbest=W(best);

    [mse,y_est_output,Q0,Wbest]=run_plsCvExtvFINAL(z(Q0:end),y(Q0:end), ...
        X(Q0:end,:),fctlb-Q0+1,fctub-Q0+1,h_index,k_tot,0,Q1andQ2,Q0,Wbest);
    y_est_output=[NaN(Q0-1,1);y_est_output];
    return

end
% ----------------

% Build largest set of regressors possible
Z_reg=NaN(i-h,k_tot*n_lags_Z_max);
y_reg=NaN(i-h,n_lags_y_max);
for j=0:n_lags_Z_max-1
    Z_reg(j+1:i-h,j*k_tot+1:(j+1)*k_tot)=Z(1:i-h-j,:);
end
for j=0:n_lags_y_max-1
    y_reg(j+1:i-h,j+1)=z(1:i-h-j);
end

% Find right number of lags and forecast y - n_lags allowed to vary for y and Z
% and allowed to vary in each iteration
fcts=NaN(n_lags_y_max,n_lags_Z_max);
bics=NaN(n_lags_y_max,n_lags_Z_max);
for n_lags_y=1:n_lags_y_max
    for n_lags_Z=1:n_lags_Z_max
        predictors=[ones(i-h,1) y_reg(:,1:n_lags_y) Z_reg(:,1:k_tot*n_lags_Z)];
        [b,~,e]=regress(y(h+1:i-h),predictors(1:i-2*h,:));
        fcts(n_lags_y,n_lags_Z)=predictors(i-h,:)*b;
        e=e(~isnan(e));
        T=length(e);
        bics(n_lags_y,n_lags_Z)=log(1/T*e'*e)+size(predictors,2)*log(T)/T;
    end
end
[temp,index_y]=min(bics);
[~,index_Z]=min(temp);
```

```
            y_est_all(i,p,state)=fcts(index_y(index_Z),index_Z);

            % Make h-1 missing h-order out-of-sample forecasts between estimation and forecast
            % samples used to derive a gap-free state vector
            if extension==2
                if i==fctlb && h>1
                    n_lags_y=index_y(index_Z); % best number of lags based on relevant state
                                               % portion of full estimation sample
                    n_lags_z=index_Z;
                    predictors=[ones(h-1,1) y_reg(fctlb-2*h+1:fctlb-h-1,1:n_lags_y) ...
                        Z_reg(fctlb-2*h+1:fctlb-h-1,1:k*n_lags_z)];
                    b=regress(y(fctlb-h+1:fctlb-1),predictors);
                    y_est_missing(:,state)=predictors*b;
                end
            end

        end

        % Compute final forecast as a convex combination of the two state-specific ones
        % in extension 2
        if extension==2
            lastk=S(i-h);
            prob1=Pi(lastk,1);
            y_est_all(i,p,3)=prob1*y_est_all(i,p,1)+(1-prob1)*y_est_all(i,p,2);
        end

    end

    % Compute MSE for given n_lags_y
    mse_all(p)=1/T_mse*sum((y_est_all(fctlb:fctub,p,end)-y(fctlb:fctub)).^2);

end

% Select optimal n_lags_y based on model with min MSE
[mse,n_lags_AR_opt]=min(mse_all);
y_est_output=y_est_all(:,n_lags_AR_opt,end);

end
```

A.3.3 Structural break

```
function W = find_SB(Z,Q1andQ2)
% FIND_SB Returns the vector of test statistics for each possible
% breakpoint location Q within [Q1,Q2] in a given matrix of factors Z

T=size(Z,1);
k=size(Z,2);

Q1=Q1andQ2(1);
Q2=Q1andQ2(2);

W=NaN(Q2-Q1+1,1);
```

```
for q=Q1:Q2

    index=q-Q1+1;
    p=q/T;

    % Compute A vector
    A1=zeros(k);
    A2=zeros(k);
    for i=1:q
        A1=A1+Z(i,:)'*Z(i,:);
    end
    for i=q+1:T
        A2=A2+Z(i,:)'*Z(i,:);
    end
    A=sqrt(T)*(1/q*A1-1/(T-q)*A2);
    A=A(:);

    % Compute covariance matrix V and test statistic vector W
    V=1/p*Omega(1,q,Z)+1/(1-p)*Omega(2,q,Z);
    W(index)=A'/V*A;

end
```

```
function [value] = Omega(type,q,Z)
% OMEGA Function used in deriving the covariance matrix of the A vector

T=size(Z,1);
value=Gamma(type,0,q,Z);

if type==1

    for j=1:q-1
        value=value+kernel(j,q)*(Gamma(1,j,q,Z)+Gamma(1,j,q,Z)');
    end

elseif type==2

    for j=1:T-q-1
        value=value+kernel(j,T-q)*(Gamma(2,j,q,Z)+Gamma(2,j,q,Z)');
    end

end
```

```
function [value] = Gamma(type,j,q,Z)
% GAMMA Function used in deriving the covariance matrix of the A vector

T=size(Z,1);
k=size(Z,2);
value=zeros(k^2);
```

```
if type==1

    for i=j+1:q
        factor1=Z(i,:)'*Z(i,:)-eye(k);
        factor2=Z(i-j,:)'*Z(i-j,:)-eye(k);
        value=value+factor1(:)*factor2(:)';
    end
    value=1/q*value;

elseif type==2

    for i=j+q+1:T
        factor1=Z(i,:)'*Z(i,:)-eye(k);
        factor2=Z(i-j,:)'*Z(i-j,:)-eye(k);
        value=value+factor1(:)*factor2(:)';
    end
    value=1/(T-q)*value;

end
```

```
function [value] = kernel(j,t)
% Quadratic-spectral kernel function

denom=1.3221*(0.2*t)^(1/5); % TEMPPP set c2 as 0.2
x=j/denom;

value=25/(12*pi^2*x^2)*(sin(6*pi*x/5)/(6*pi*x/5)-cos(6*pi*x/5));
```

A.3.4 Markov regime switches

```
function [S,Pi0,Pi] = find_MS(X,h)
% FIND_MS Returns the optimal state variables and transition probabilities
% for a given estimation sample and forecast horizon

[T,N]=size(X);
L0=4;

% Initialise S
S=(X(:,115)>mean(X(:,115)))+1;

% Initiate while loop
bool=true;
iteration=0;

while bool

    iteration=iteration+1;

    % Indicator functions for unconditional probabilities
    indic2=S-1;
```

```
indic1=ones(T,1)-indic2;
total1=sum(indic1);
total2=sum(indic2);

% Indicator functions for four L-order transition probabilities plus
% h-order probabilities to be outputted for use in run_(model)
indic11=zeros(T,L0+1);
indic12=zeros(T,L0+1);
indic21=zeros(T,L0+1);
indic22=zeros(T,L0+1);
for i=1:T-L
    transition=[S(i);S(i+L)];
    if transition==[1;1]
        indic11(i,L)=1;
    elseif transition==[1;2]
        indic12(i,L)=1;
    elseif transition==[2;1]
        indic21(i,L)=1;
    else
        indic22(i,L)=1;
    end
end
for i=1:T-h
    transition=[S(i);S(i+h)];
    if transition==[1;1]
        indic11(i,L0+1)=1;
    elseif transition==[1;2]
        indic12(i,L0+1)=1;
    elseif transition==[2;1]
        indic21(i,L0+1)=1;
    else
        indic22(i,L0+1)=1;
    end
end

% Compute unconditional probabilities
Pi0=NaN(2,1);
Pi0(1)=total1/T;
Pi0(2)=1-Pi0(1);

% Compute transition probabilities
Pi=NaN(2,2,2);
orders=[1 L0+1]; % only relevant to store first order for procedure and h-order for output
for type=1:2
    order=orders(type);
    Pi(1,1,type)=sum(indic11(:,order))/total1;
    Pi(1,2,type)=sum(indic12(:,order))/total1;
    Pi(2,1,type)=sum(indic21(:,order))/total2;
    Pi(2,2,type)=sum(indic22(:,order))/total2;
end


% Compute L-order covariance matrices used to construct M
mu=[X'*indic1/total1 X'*indic2/total2];
```

```
CovX1=zeros(N,N,L0);
CovX2=zeros(N,N,L0);
for L=1:L0
    for i=1:T-L
        CovX1(:,:,L)=CovX1(:,:,L)+(X(i,:)'-mu(:,1))*(X(i+L,:)'-mu(:,1))'*indic11(i,L) ...
            +(X(i,:)'-mu(:,1))*(X(i+L,:)'-mu(:,2))'*indic12(i,L);
        CovX2(:,:,L)=CovX2(:,:,L)+(X(i,:)'-mu(:,2))*(X(i+L,:)'-mu(:,1))'*indic21(i,L)+ ...
            (X(i,:)'-mu(:,2))*(X(i+L,:)'-mu(:,2))'*indic22(i,L);
    end
    CovX1(:,:,L)=CovX1(:,:,L)/sum(indic1(1:T-L));
    CovX2(:,:,L)=CovX2(:,:,L)/sum(indic2(1:T-L));
end

% Build M matrix
M1=0;
M2=0;
for L=1:L0
    M1=M1+CovX1(:,:,L)*CovX1(:,:,L)';
    M2=M2+CovX2(:,:,L)*CovX2(:,:,L)';
end

% Compute d scalar
Nstar=min(rank(M1),rank(M2));
[V1,D1]=eig(M1,'vector');
[V2,D2]=eig(M2,'vector');
D1=sort(D1,'descend');
D2=sort(D2,'descend');
temp1=D1(2:floor(Nstar/2)+1)./D1(1:floor(Nstar/2));
temp2=D2(2:floor(Nstar/2)+1)./D2(1:floor(Nstar/2));
[~,d1]=min(temp1);
[~,d2]=min(temp2);

norm1=sqrt(sum(sum(M1.^2)));
norm2=sqrt(sum(sum(M2.^2)));
if norm1>norm2
    d=d1;
else
    d=d2;
end

% Define Q1 and Q2 matrices of eigenvectors used to check convergence
[~,indices1]=maxk(D1,d);
[~,indices2]=maxk(D2,d);
Q=NaN(N,d,2);
Q(:,:,1)=V1(:,indices1);
Q(:,:,2)=V2(:,indices2);

% Define B1 and B2 and covariances to be used in likelihood function
p=Nstar-d;
[~,indices1]=maxk(D1,d+p);
[~,indices2]=maxk(D2,d+p);
B=NaN(N,p,2);
B(:,:,1)=V1(:,indices1(d+1:end));
B(:,:,2)=V2(:,indices2(d+1:end));
```

```
    CovB=zeros(p,p,2);
    for i=1:T
        CovB(:,:,1) = CovB(:,:,1) + (B(:,:,1)'*(X(i,:)'-mu(:,1))*(X(i,:)'-mu(:,1))' ...
            *B(:,:,1)*indic1(i));
        CovB(:,:,2) = CovB(:,:,2) + (B(:,:,2)'*(X(i,:)'-mu(:,2))*(X(i,:)'-mu(:,2))' ...
            *B(:,:,2)*indic2(i));
    end
    CovB(:,:,1) = CovB(:,:,1) / (total1-1);
    CovB(:,:,2) = CovB(:,:,2) / (total2-1);

    % Perform Viterbi algorithm based on current variables derived
    Stest=[1 2];
    for i=2:T
        values=NaN(2,2);
        for k=1:2
            for j=1:2
                values(j,k)= log(Pi(j,k)) + logf(X(i,:)',mu(:,k),B(:,:,k),CovB(:,:,k)) ...
                    + Gsum(Stest(:,j),X,mu,B,CovB,Pi0,Pi);
            end
        end
        [~,indices]=max(values);
        Stest=[Stest(:,indices);1 2];
    end
    values=[Gsum(Stest(:,1),X,mu,B,CovB,Pi0,Pi);Gsum(Stest(:,2),X,mu,B,CovB,Pi0,Pi)];
    [Gbest,index]=max(values);
    S=Stest(:,index);

    % Check for convergence
    if iteration>1
        condition1=1/2*(D(Qold(:,:,1),Q(:,:,1))+D(Qold(:,:,2),Q(:,:,2)));
        condition2=abs((Gold-Gbest)/Gold);
        if condition1<0.1 || condition2<0.2 || iteration==50
            bool=false;
        end
    end
    Qold=Q;
    Gold=Gbest;

end
```

```
function value = Gsum(S,X,mu,B,CovB,Pi0,Pi)
% GSUM Function weighing between the values obtained with the f likelilood
% function using transition probabilities

T=length(S);

value = log(Pi0(S(1))) + logf(X(1,:)',mu(:,S(1)),B(:,:,S(1)),CovB(:,:,S(1)));

if T>1
    for i=2:T
        value = value + log(Pi(S(i-1),S(i))) + logf(X(i,:)',mu(:,S(i)),B(:,:,S(i)), ...
            CovB(:,:,S(i)));
```

```
    end
end
```

```
function value = logf(xt,mu,B,CovB)
% LOGF Log-likelihood function based on the Gaussian density function

p=size(B,2);

value = - 1/2*log((2*pi)^(p)) - 1/2*log(det(CovB)) - 1/2*(B'*(xt-mu))'/CovB*B'*(xt-mu);
```

```
function value = D(H1,H2)
% D Function based on two successive output matrices of the iterative
% algorithm used to check for convergence of the estimated state vector

value=sqrt(1-(1/max(size(H1,2),size(H2,2)))*trace(H1*H1'*H2*H2'));
```