ERASMUS UNIVERSITY ROTTERDAM

ERASMUS SCHOOL OF ECONOMICS

BACHELOR'S THESIS BSC^2 ECONOMETRICS/ECONOMICS

Forecasting Inflation Using Many Predictors

A comparison between linear and non-linear dimensionality reduction techniques

Fabio Hoxha

(423293)

Supervisor: dr. A.M. Schnücker

Second assessor: prof. dr. D.J.C. van Dijk

Date: July 6, 2019

– Page intentionally left blank –

Abstract

This thesis analyzes the performance of linear and non-linear dimensionality reduction techniques in the context of inflation forecasting. In particular, Principal Component Analysis (PCA) and three Partial Least Squares (PLS) variants are used as representative of linear methods. Non-linear models include Squared Principal Components (SQPC), Kernel PCA, and Kernel PLS. The findings indicate that factor models substantially improve the forecasting performance in comparison to univariate autoregressive models. More importantly, although this paper finds non-linear models to dominate the linear ones in specific subsamples and forecast horizons, it concludes that there is rather limited room for improvement in the forecasting methodology concerning non-linear models. A key reason is that these models come with the major drawback that their performance is highly dependent on the choice of hyperparameters.

Contents

1. Introduction

The abundance of data in the field of economics and beyond has led to a series of publications on how to effectively use large datasets for forecasting purposes (e.g., Bok, Caratelli, Giannone, Sbordone, and Tambalotti (2018); Diebold (2003); Stock and Watson (2002a)). Along with carrying more valuable information, data-rich environments generally introduce difficulties to traditional econometric models. In particular, the so-called curse of dimensionality, i.e., when the number of parameters to estimate is larger than the number of observations, and the multicollinearity between different variables can make it impossible to estimate simple multiple regression models or hinder their efficiency (Fuentes et al., 2015). In the field of finance and macroeconomics, the use of factor models has gained widespread attention in overcoming such issues. This thesis focuses on the use of linear and non-linear factor construction techniques in the context of macroeconomic forecasting.

The main intuition behind factor models is as follows. Given a large set of potential explanatory variables, factor models first reduce the dimensionality of the dataset by extracting a small number of latent factors. These factors are then used to explain or forecast some variable of interest. This is different from conventional regression models, in which the entire set of explanatory variables is used. There exist multiple techniques for constructing the latent factors, each leading to a variation of the general class of factor models. In general, these can be divided into two broad categories: (1) unsupervised techniques and (2) supervised techniques.

Unsupervised techniques extract common factors without considering the target variable to be forecasted. One of the most popular unsupervised techniques is the Principal Component Analysis (PCA), initially introduced by Pearson (1901). PCA projects data into a lower-dimensional subspace comprised of orthogonal components (factors), which maximize the explained variability of the original dataset. Stock and Watson (2002a), for instance, apply the PCA method to forecast the Federal Reserve Board's Index of Industrial Production based on a set of 149 potential predictors. Their empirical findings indicate outperformance of the models incorporating PCA as compared to univariate autoregressions and small-scale vector autoregressions. Similar findings for other macroeconomic series, such as inflation, also arise in Stock and Watson (2002b) and Bai and Ng (2008), among others. Another popular factor estimation technique in the literature is the Dynamic Principal Component Analysis (DPCA). While PCA assumes that data are time-independent, DPCA deals with time-series data that have a non-zero autocorrelation, by enlarging the original set of predictors with lagged values of the variables

(Vanhatalo, Kulahci, & Bergquist, 2017). Several studies compare the predictive performance of PCA- and DPCA-based methods, including Boivin and Ng (2005), Eickmeier and Ziegler (2008), and Stock and Watson (2006). However, no clear conclusions seem to arise.

The other class of factor extraction methods concerns supervised techniques. These aim to address one of the main criticisms concerning unsupervised techniques, that they do not account for the purpose of explaining a particular target variable while extracting the common factors. Factors extracted via PCA, for example, explain the vast amount of variation in the set of predictors. However, they do not necessarily explain a large proportion of the variation in the target variable (Fuentes et al., 2015). To mitigate this drawback of PCA, or unsupervised methods in general, Fuentes et al. (2015) and Groen and Kapetanios (2009) propose the implementation of Partial Least Squares (PLS) and its sparse extension in macroeconomic forecasting. PLS is a dimension reduction technique, initially introduced by Wold (1966). In contrast to PCA, the objective of PLS is in optimizing the covariance between the factors and the target variable. Fuentes et al. (2015) propose three PLS variants (one static and two dynamic) that account for the time-series structure of the data. Their results indicate that some PLS implementations outperform PCA in forecasting inflation using a set of 132 predictor variables, especially for the medium-term forecasting horizons. Other approaches classified as supervised techniques and used in the context of macroeconomic forecasting include targeted predictors (Bai & Ng, 2008) and ad-hoc procedures (Boivin & Ng, 2006).

All methods described above have one common feature in their respective baseline representations: the latent factors are extracted from linear combinations of the predictor variables. Additionally, all the studies mentioned above use linear regression models to explain the relationship between the target variable and the common factors. Effectively, this assumes a linear link between the target variable to be forecasted and the original set of potential predictors. However, based on the expectation of non-linear relationships between the variables in a real-world economy, this might be a quite restrictive assumption (Giovannelli, 2012). Non-linearities can arise due to various reasons. As a matter of fact, many macroeconomic variables show different behaviors during the expansion and contraction phases of the business cycle, which gives rise to a simple regime-switching-like form of non-linearity (LeBaron, 1994). Another possible reason is related to the asymmetric future response of macroeconomic variables to a positive or negative shock today. Several publications in the literature provide evidence in favor of non-linear relationships between various economic variables. The results in Sarel (1996), for

instance, suggest non-linear effects of inflation on economic growth. The findings of Eggoh and Khan (2014) also support such a relationship. Other relevant instances include, but are not limited to: (1) Hung (2009), who suggests non-linearity in the relationship between financial development and economic growth, and (2) Bekiros, Gupta, and Kyei (2016), who find evidence for non-linear associations between economic and firm-level uncertainty measures and stock market volatility.

Despite this existing evidence in favor of non-linear relationships, economic time series have been usually forecasted using linear models (Giovannelli, 2012; Kock, Teräsvirta, et al., 2011). Publications focusing on non-linear methods generally employ (small-scale) regime-switching models or smooth transition regression models (Kock et al., 2011). With a few exceptions (e.g., Bai and Ng (2008); Exterkate, Groenen, Heij, and van Dijk (2016); Giovannelli (2012)), non-Bayesian and non-linear methods for dealing with data-rich environments are barely existent in the macroeconomic forecasting literature. As a first step in this regard, Bai and Ng (2008) propose a variation to the conventional PCA framework: Squared Principal Components (SQPC). SQPC allows for a non-linear function between the original variables and the estimated factors. In a nutshell, Bai and Ng (2008) suggest to apply PCA to the set of predictor variables enlarged by the inclusion of their squared terms. This way, the authors allow the factors to be possibly non-linear functions of the predictors, without altering the linear relationship between the target variable and factors assumed in other related studies. Their empirical analysis on inflation forecasting indicates that SQPC performs better than a linear PCA model in many samples and for different forecast horizons.

Although the framework of SQPC in Bai and Ng (2008) is a step into incorporating non-linearities in the dimensionality reduction framework, the method is still restrictive in the sense that it cannot capture complex non-linear relationships. More specifically, SQPC only allows non-linearities to arise from quadratic transformations of the original variables. In this regard, two key findings in Giovannelli (2012) motivate further research on the topic: (1) the use of non-linear Kernel PCA (KPCA) in extracting common factors significantly outperforms the linear PCA, and (2) a non-linear forecasting equation has no significantly better performance than a simple linear regression. Giovannelli (2012), however, does not consider supervised dimensionality reduction techniques in his work. Therefore, inspired by the finding of Fuentes et al. (2015) that PLS outperforms PCA, the main aim of this thesis is to investigate whether non-linear extensions of the PLS and PC framework can provide further performance increases

when applied to inflation forecasting. In addition to considering a squared extension of the PLS (i.e., SQPLS), this thesis also proposes the use of a more general form of non-linear PLS: the non-linear Kernel PLS (KPLS), as defined in Rosipal and Clancy (2003). Therefore, this paper revisits the PCA and PLS methodology of Fuentes et al. (2015) and compares the predictive performance to that of the previously mentioned non-linear models (i.e., SQPLS, KPCA, and KPLS).

Inflation is considered as one of the most difficult macroeconomic variables to forecast (Stock & Watson, 1999). Therefore, as argued by Bai and Ng (2008), testing different models on inflation forecasting provides an adequate framework for comparison. Improvements in the forecasting of inflation are therefore highly necessary, as the best-performing models might also be implemented in other forecasting exercises. As such, this research is not only relevant from an academic perspective but might also benefit professionals who heavily rely on macroeconomic, or more generally, financial time-series forecasts. In particular, this paper applies the KPCA and KPLS framework in inflation forecasting, which to the best of my knowledge is not yet applied in this setting. Moreover, this paper provides a thorough comparison of 12 different models (four linear and eight non-linear) by considering not only the final predictive performance but also several robustness and sensitivity analyses. Furthermore, the comparison of linear and non-linear methods provides an implicit investigation of the existence of non-linear relationships between a broad variety of macroeconomic variables and inflation.

Next to the above discussion, this thesis also aims to acknowledge the differences in the forecasters' preferences and their degree of risk aversion. As an example, forecasters using the models presented in this paper might have different attitudes towards the forecast errors: some might be more sensitive to large errors than small errors, while others are equally concerned (Stock & Watson, 1998). Likewise, depending on the application, one might be more sensitive to an under-estimation of future inflation as compared to an over-estimation (Diebold & Mariano, 2002). For this reason, in addition to using the mean squared prediction error (a standard measure in the literature), this thesis also ranks the models under investigation based on other loss functions, such as the mean absolute error, the cubic loss, the piece-wise linear loss, and the asymmetric squared loss functions, as well as it proposes two new asymmetric loss functions.

This paper performs an empirical application using the well-known Stock and Watson (2005) dataset. The data is composed of 132 macroeconomic variables for the United States economy, measured at a monthly frequency, during the time span from January 1960 to Decem-

ber 2003. The findings support the idea that factor models substantially improve the forecasting accuracy as compared to univariate autoregressions. In particular, for the medium-term forecast horizons of 12 and 24 months, there is a 40-50% decrease in the mean squared prediction error. Next, I find evidence that two linear PLS variants consistently give better forecasting performance than linear PCA, supporting the use of supervised factor extraction techniques in the field of macroeconomic forecasting. Concerning the comparison between linear and non-linear models, I find that non-linear methods (in particular, KPLS with a polynomial Kernel function) perform significantly better when constructing 6-months ahead inflation forecasts. However, it should be noted that the sensitivity analysis suggests that Kernel methods are highly unstable and their success strongly depends on the choice of Kernel hyperparameters. For the other forecast horizons, there is either no significant evidence for the over-performance of non-linear models or the results are mixed and sample-dependent. Finally, this paper finds that the model choice highly depends on the forecaster's goal and their attitude towards the prediction error.

The remainder of this thesis is structured as follows. Section 2 describes the linear and non-linear factor extraction methodology. An empirical application is presented in Section 3. Section 4 concludes, while additional results are summarized in the Appendix of this paper.

2. Methodology

Assume that we are given data on a potentially large number of predictors $X_t = (X_{1t},...,X_{Nt})'$, where $t = 1,...,T$, and N is the total number of predictors. Our aim is to forecast the so-called target variable, y_{T+h}, given the information set at time T. In other words, we are interested in generating a h-step-ahead forecast for y_t at time T. Since the number of predictors can be large, we do not directly use the original predictors in a forecasting model. Instead, we first extract K, $K << N$, common factors of X_t, denoted by $\hat{F}_t$. Then, we assume the following forecasting equation for y_t:

$$y_{t+h} = \omega + \phi(L)y_t + \beta'(L)\hat{F}_t + \varepsilon_{t+h}, \tag{1}$$

where ω is a constant, $\phi(L)$ and $\beta'(L)$ are the lag polynomials corresponding to y_t and $\hat{F}_t$, respectively, while ε_{t+h} is an error term.

Different methods used to construct the K unobserved factors, $\hat{F}_t$, will be considered. First, the methods used in Fuentes et al. (2015) will be briefly discussed, followed by the non-linear squared and Kernel extensions.

2.1. Principal Components

The first dimensionality reduction technique considered concerns the application of PCA on the predictor matrix $X = (X_1, ..., X_T)'$, where each row represents an observation and each column a predictor variable. The factors $\hat{F}_t$, in this case, correspond to the first K principal components of the matrix X. That is, the k-th factor, $\hat{F}^k$, can be obtained as $\hat{F}^k = Xe_k$, where e_k is the eigenvector of $X'X$ corresponding to its k-th largest eigenvalue. Such factors are orthogonal and aim to maximize the proportion of variance explained in the set of original variables, X. Since PCA is not scale-invariant (e.g., see Bai and Ng (2008)), each predictor variable, i.e., each column of X, is standardized to have zero mean and unit variance before performing PCA.

2.2. Partial Least Squares

While PCA produces factors that maximize the proportion of explained variance, PLS is a dimensionality reduction technique that extracts orthogonal components based on the covariance between the predictors and the target variable. This is an iterative approach, in the sense that the factors are obtained sequentially. According to Fuentes et al. (2015), the first PLS factor is obtained as $\hat{F}^1 = Xe_1$, where e_1 can be found by solving the following optimization problem:

$$e_1 \;=\; \arg\max_{e_1} \; e_1{}'X_h{}'Y_hY_h{}'X_he_1, \qquad \text{subject to} \quad e_1{}'e_1 = 1, \tag{2}$$

where $Y_h = (y_{h+1}, ..., y_T)'$ and $X_h = (X_1, ..., X_{T-h})'$. It can be shown that e_1 is the first eigenvector of the matrix $X_h{}'Y_hY_h{}'X_h$. To see this, we first write down the Lagrangian corresponding the above maximization problem:

$$\mathcal{L} = e_1{}'X_h{}'Y_hY_h{}'X_he_1 - l(e_1{}'e_1 - 1). \tag{3}$$

Taking the derivative with respect to e_1 in Equation 3, we get:

$$\frac{\partial \mathcal{L}}{\partial e_1} = 2X_h{}'Y_hY_h{}'X_he_1 - 2le_1. \tag{4}$$

Setting the expression in Equation 4 equal to zero yields the following equality:

$$Me_1 = le_1, \tag{5}$$

where $M = X_h{}'Y_h Y_h{}'X_h$, such that e_1 is by definition the eigenvector of M corresponding to its largest eigenvalue.

To find the second PLS factor, Y_h is regressed on the first factor and a constant term and is replaced by the residuals of this regression. Additionally, each column of X and X_h is also regressed over the first component and a constant, and afterward replaced by the residuals of this regression. Then, the second PLS factor is obtained by applying the eigenvalue decomposition on the updated X_h and Y_h. This process is similarly repeated until the last factor is obtained.

To be noted is that, similarly to PCA, PLS is not scale-invariant. Therefore, all variables in X, X_h and Y_h are standardized prior to constructing M and deriving the first PLS factor.

Following the same approach as in Fuentes et al. (2015), I estimate three variants of the PLS that take into account the time series structure of the data. These can be summarized as follows:

(a) Factors are obtained by using the original matrix X and the vector Y, as defined above. In this case, Equation 1 contains lags of the target variable.

(b) Prior to applying PLS, X is replaced by X_e, which expands the original set of predictors, X, by lags of the target variable. The target Y remains the same as in approach (a). In this case, Equation 1 does not include lags of the target, since they are already included in X_e and, as such, are accounted for when constructing the latent factors.

(c) Factors are computed using the original matrix X and Y^{res}, where Y^{res} is the residual series obtained from fitting an AR(p) process to the target variable Y. In this approach, Equation 1 contains lags of the target variable.

In what follows, the three PLS approaches introduced above will be referred to as PLS(a), PLS(b), and PLS(c), respectively.

2.3. Squared Methods

In the spirit of Bai and Ng (2008), who introduce the SQPC approach to account for non-linear factor estimation, the first extension to the methodology of Fuentes et al. (2015) that this paper considers is the Squared Partial Least Squares (SQPLS). The main intuition behind SQPLS is to replace X by X_{SQ}, which augments X with columns corresponding to the squared variables in X. That is, $X_{SQ,t} = \{X_{it}, X_{it}^2\}$, such that, if X has N columns, then X_{SQ} has $N_{SQ} = 2N$ columns.

Computing the latent factors using SQPLS is equivalent to PLS if we replace X by X_{SQPLS} in all steps. Analogous to PLS, approaches (a), (b) and (c) will be also implemented for SQPLS.

According to Bai and Ng (2008), the purpose of including the squared terms is two-fold. First, a simple non-linear structure is introduced. Secondly, the factors are linear combinations of both the original and squared variables, thus capturing the effects of both levels and volatility in the latent factors. This might increase the explanatory and predictive power of the factors.

For comparison purposes, this paper also employs the SQPC method. For SQPC, the estimation procedure is analogous to that of linear PC introduced in Section 2.1 in case that X_{SQ} substitutes X.

2.4. Kernel Methods

Although the squared methods proposed above introduce non-linearities in the PCA and PLS frameworks, they are still very restricted in the sense that the feature space over which components are formed only extends the space spanned by the original variables with their squared components. To further generalize this approach, I put forward the Kernel Principal Component (KPC) and Kernel Partial Least Squares (KPLS) methods, which are able to incorporate not only higher-order transformations in the set of predictor variables (as compared to the squared methods), but also interaction terms and other non-linear transformations, such as the vector norm. This non-linear technique is highly relevant beyond its flexibility in capturing complex non-linear behaviors. In particular, the squared methods compute explicit mappings to the higher dimensional space, a procedure that fails to account for the high dimensionality of the problem, which may, in turn, lead to computational difficulties as the number of predictors increases. As we will see, Kernel methods are specially developed to overcome such issues.

Several implementations of the KPC and KPLS are proposed in the literature. For comparison purposes, I choose to implement KPLS similarly to Rosipal and Clancy (2003) because their methodology closely resembles the PLS variant of Fuentes et al. (2015). Again, for comparability, I apply the same trick that Rosipal and Clancy (2003) propose in the KPLS framework to derive KPC, similarly as Giovannelli (2012). The intuition behind these two methods is to first map the original predictor data into a higher-dimensional (possibly even infinite-dimensional) feature space, $\mathcal{F}$, corresponding to a reproducing Kernel Hilbert space (RKHS), and then apply the PCA or PLS in $\mathcal{F}$ (Lee, Yoo, Choi, Vanrolleghem, & Lee, 2004; Rosipal & Clancy, 2003).

Let $\Phi(.)$ be a non-linear function that maps an input vector from the input space to $\mathcal{F}$.

Then, each row (observation) of X, $X_t \in \mathcal{R}^N$, can be mapped into the feature space $\mathcal{F}$ as follows:

$$X_t^* = \Phi(X_t), \qquad t = 1, ..., T. \tag{6}$$

Of course, one could apply Equation 6 to each row of X (analogous to the previous case where we add the squared terms), construct $X^* = (X_1^*, ..., X_T^*)'$, and afterward apply PCA of PLS using X^* instead of X. However, as argued by Lee et al. (2004), the mapping $\Phi(.)$ might not always be computationally tractable (e.g., due to infinite-dimensionality). In this regard, the advantage of Kernel methods is that the latter computations need not be carried.

In what follows, the KPC and KPLS methods employed in this paper will be discussed in more detail, together with the choice of the Kernel functions.

2.4.1. Kernel Principal Components

Section 2.1 argues that the principal components of X are obtained as linear combinations between X and the eigenvectors of $X'X$. In a similar manner, one can apply the same procedure using X^* to obtain KPC. However, as argued above, explicit computations of the mapping function $\Phi(.)$ are not required. To alleviate such computations in KPC, the results of Stock and Watson (2002a), among others, are particularly useful. The authors show that, for linear PCA, in case $T < N$, there is a more computationally efficient way of obtaining principal components. Instead of computing the eigenvectors of the $N \times N$ matrix $X'X$, one can also apply an eigenvector decomposition of the (smaller) $T \times T$ matrix XX'. In the latter case, the eigenvectors correspond to the principal components of X. Incorporating this procedure to KPC, we can obtain the i-th component, $\hat{F}_{i,KPC}^*$, by solving the following eigenvalue problem:

$$K\hat{F}_{i,KPC}^* = \lambda_i \hat{F}_{i,KPC}^*, \tag{7}$$

where $K = X^*X^{*\prime}$ and λ_i denotes the i-th largest eigenvalue of K.

Although Stock and Watson (2002a) motivate the usefulness of this finding from an efficiency point of view, the equivalence of the two methods is crucial for deriving KPC. In particular, calculating the entries of $X^*X^{*\prime}$ requires no explicit evaluation of the mapping function $\Phi(.)$. To see this, note that the (i, j)-th entry of $X^*X^{*\prime}$ is defined as the dot product between $\Phi(X_i)$ and $\Phi(X_j)$, i.e., $\Phi(X_i)\Phi(X_j)'$. This implies that only computations of the dot product in the feature space $\mathcal{F}$ are required to derive KPC. Applying the so-called Kernel trick, the value of the dot

product between two vectors in $\mathcal{F}$ can be evaluated without performing any computations in the feature space by using a Kernel function of the form

$$k(a,b) = \langle \Phi(a), \Phi(b) \rangle, \tag{8}$$

where the operator $\langle x, y \rangle$ represents the dot product between x and y (Aronszajn, 1950). Effectively, this implies that we can apply PCA in $\mathcal{F}$ by using simple linear algebra as in the linear PCA method, hence avoiding the explicit computation of the non-linear mapping (Rosipal & Clancy, 2003; Rosipal & Trejo, 2001).

Following Lee et al. (2004) and Schölkopf, Smola, and Müller (1998), the mapped data in $\mathcal{F}$ are assumed to have zero-mean. This can be achieved by substituting K in Equation 7 with

$$\tilde{K} = (I_T - \frac{1}{T} 1_T 1_T{}') \; K \; (I_T - \frac{1}{T} 1_T 1_T{}'), \tag{9}$$

where I_T is a T-dimensional identity matrix and 1_T is a $T \times 1$ vector with all elements set to one.

2.4.2. Kernel Partial Least Squares

Remember that, as discussed in Section 2.2, the first PLS component is extracted by a linear combination of X and the first eigenvector of M. Similarly, in the case of transformed predictors in the feature space $\mathcal{F}$, we need to solve the following eigenvalue problem:

$$M^* e_1 = \lambda_1 e_1 \quad \Longleftrightarrow \quad X^{*\prime} Y Y' X^* e_1 = \lambda_1 e_1. \tag{10}$$

Multiplying both sides of Equation 10 by X^*, and using the fact that the first component is defined as $\hat{F}^{*1} = X^* e_1$, we have

$$X^* X^{*\prime} Y Y' \hat{F}^{*1} = \lambda \hat{F}^{*1}, \tag{11}$$

such that $\hat{F}^{*1}$ is the first eigenvector of $X^* X^{*\prime} Y Y'$. Note that, as in KPC, the (i,j)-th entry in the matrix $X^* X^{*\prime}$ is defined as the dot product between $\Phi(X_i)$ and $\Phi(X_j)$, i.e., $\Phi(X_i)\Phi(X_j)'$. Therefore, only the dot product in the space $\mathcal{F}$ is required to obtain the KPLS factors, for which the Kernel trick is applied similarly to the case of KPC.

To be noted is the fact that in KPLS, as defined here, we can not implement approaches (a), (b) and (c) introduced in Section 2.2. To see this, note that in KPLS, we are unable to explicitly derive the eigenvectors (or, weights) used to compute the factors due to untractable

computations in the feature space. As such, since we make use of the trick that $\hat{F}^{*1} = X^* e_1$, KPLS actually maximizes the contemporaneous covariance between the predictors and the target variable at time t, instead of the covariance between the predictors at time t and the h-step ahead target. For this reason, one can argue that there is a trade-off in choosing between PLS and KPLS: PLS can take into account the time-series structure of the variables but not the non-linear relationships, while the opposite holds for KPLS.

As in KPC, KPLS also assumes the mapped data to have zero mean. The same approach (see Equation 9) is used to centralize the transformed variables in space $\mathcal{F}$.

2.4.3. Kernel Functions

Several Kernel functions have appeared in the literature. A Kernel function defines a valid mapping into the feature space via the dot product if it satisfies Mercer's theorem (Mercer, 1909). According to Exterkate et al. (2016) and Giovannelli (2012), the choice of the Kernel function is crucial for the performance of the forecasting model. This is because the choice of the Kernel function implicitly represents an assumption of the forecaster concerning the data generating process (Giovannelli, 2012). Considering that both papers focus on macroeconomic forecasting, I follow their approach and evaluate the performance of (1) the polynomial and (2) the Gaussian radial basis Kernel functions, as defined in Exterkate et al. (2016). Both these functions always satisfy Mercer's theorem (Lee et al., 2004). Given two input vectors, a and b, the polynomial Kernel function is defined as

$$k(a,b) = (a'b + 1)^2, \tag{12}$$

while the Gaussian radial basis Kernel can be written down as

$$k(a,b) = \exp\left(-\frac{||a-b||^2}{2}\right), \tag{13}$$

where the operator $||x||$ represents the norm of vector x. Following Exterkate et al. (2016), each observation is scaled by $1/\sigma$, $\sigma > 0$, to control for their relative importance in the Kernel function.

The choice of the scaling parameter σ is very important for the performance of Kernel methods. According to Giovannelli (2012), overestimation of such parameter makes the Kernels unable to capture nonlinearities in the data, while underestimation means that the

Kernels are highly sensitive to noise in the training samples. In the spirit of Exterkate et al. (2016) and Stock and Watson (2002b), to optimise σ, I perform a grid search over the space $\sigma \in \{0.5\sigma_0, \sigma_0, 2\sigma_0, 4\sigma_0, 8\sigma_0\}$, where $\sigma_0 = \sqrt{(N+2)/2}$ for the polynomial Kernel, and $\sigma_0 = \sqrt{c_N}/\pi$ for the Gaussian radial basis Kernel, where c_N is the 95-th percentile of the χ^2 distribution with N degrees of freedom. The intuition behind these grid values is based on the smoothness assumption concerning the Kernel function. In particular, the higher the value of σ, the lower the effect of higher-order terms, and the smoother the prediction function. As it is naturally desirable for the first-order terms to have a larger effect than higher-order ones, Exterkate (2013) shows that these grid values satisfy the relation while keeping the search space relatively small for computational efficiency; for a detailed explanation see Exterkate (2013).

Deviating from the literature, I refrain from using the well-known k-fold cross-validation procedure for hyperparameter estimation due to its computation expensiveness. Instead, this paper follows a more simplistic approach (as in Giovannelli (2012)), by evaluating the performance of each model for every (fixed) value of σ and reporting only the best-performing one.[1]

To illustrate how Kernel functions define a valid mapping into the higher-dimensional space via the dot product, I consider deriving a Kernel function that represents the feature space in SQPC and SQPLS, as an example. Remember that, as discussed in Section 2.3, squared methods define a higher-dimensional feature space by augmenting the original variables with their squared terms. As such, a valid Kernel function that represents the dot product between observations i and j, $i, j = 1, ..., T$, in this space, must satisfy the following:

$$k(X_i, X_j) = \Phi(X_i)'\Phi(X_j),$$

in which $\Phi(x) = (x, x^2)'$. Applying the transformation Φ to observations i and j, we obtain

$$\begin{aligned} k(X_i, X_j) &= \langle\ (X_{1i}, ..., X_{Ni}, X_{1i}^2, ..., X_{Ni}^2), (X_{1j}, ..., X_{Nj}\ ,\ X_{1j}^2, ..., X_{Nj}^2)\ \rangle && (14) \\ &= (X_{1i}X_{1j} + ... + X_{Ni}X_{Nj}) + (X_{1i}^2X_{1j}^2 + ... + X_{Ni}^2X_{Nj}^2) && (15) \\ &= X_i'X_j + (X_i \circ X_j)'(X_i \circ X_j), && (16) \end{aligned}$$

where the operator $\circ$ represents the Hadamard (or, element-wise) product. Important to note is

[1] In fact, the procedure used here can be thought of as a combination of that in Giovannelli (2012) and in Exterkate et al. (2016). On the one hand, I only display the best-performing model like Giovannelli (2012); however, unlike the latter author, who uses a search space $\{1, ..., 1000\}$ for the adjustable Kernel parameter, I adopt the domain proposed by Exterkate et al. (2016).

Table 1: Overview of all models

	Model	Abbreviation	Short Description
1.	Principal Component Analysis	PCA	Factors are extracted using the linear PCA technique.
2.	Partial Least Squares – Variant (a)	PLS (a)	Factors are extracted using a static version of PLS.
3.	Partial Least Squares – Variant (b)	PLS (b)	Factors are extracted using a dynamic version of PLS.
4.	Partial Least Squares – Variant (c)	PLS (c)	Factors are extracted using a dynamic version of PLS.
5.	Squared Principal Component Analysis	SQPC	Factors are extracted using the squared extension of the PCA technique.
6.	Squared Partial Least Squares – Variant (a)	SQPLS (a)	Factors are extracted using the squared extension of PLS (a).
7.	Squared Partial Least Squares – Variant (b)	SQPLS (b)	Factors are extracted using the squared extension of PLS (b).
8.	Squared Partial Least Squares – Variant (c)	SQPLS (c)	Factors are extracted using the squared extension of PLS (c).
9.	Kernel Principal Components using the Polynomial Kernel function	KPC1	Kernel extension of PCA, based on the Polynomial Kernel function.
10.	Kernel Principal Components using the Gaussian radial-basis Kernel function	KPC2	Kernel extension of PCA, based on the Gaussian radial-basis Kernel function.
11.	Kernel Partial Least Squares using the Polynomial Kernel function	KPLS1	Kernel extension of PLS, based on the Polynomial Kernel function.
12.	Kernel Partial Least Squares using the Gaussian radial-basis Kernel function	KPLS2	Kernel extension of PLS, based on the Gaussian radial-basis Kernel function.

that, although SQPC can be estimated using both an explicit mapping or the Kernel function in Equation 16, it is impossible to estimate variants (a), (b), and (c) of SQPLS without computing the mapping into the feature space, for the same reasons discussed above. Therefore, in the empirical analysis that follows, I refrain from using the Kernel function derived in Equation 16, and instead compute the explicit transformation in both SQPC and SQPLS.

3. Empirical Application

3.1. Data and the Forecasting Procedure

The different models presented in Section 2 (see Table 1 for an overview) are compared with each other in terms of forecasting accuracy using the dataset of Stock and Watson (2005). The original dataset includes a total of 132 macroeconomic variables measured at a monthly frequency and spans the time period from January 1959 to December 2003. All variables are

transformed to achieve stationarity by taking logs, and first or second differences, as suggested in the online appendix of Fuentes et al. (2015). Additionally, to avoid missing values, the sample used in this study is set to begin in January 1960, which yields a total of 528 observations.

The target variable to be forecasted is the annualized change in inflation (from here onwards, referred to as inflation for simplicity), which is defined as

$$y_t^h = \frac{1200}{h}\log\left(\frac{CPI_t}{CPI_{t-h}}\right) - 1200\log\left(\frac{CPI_{t-h}}{CPI_{t-h-1}}\right), \tag{17}$$

where CPI refers to the Consumer Price Index. Additionally, let

$$z_t = 1200\log\left(\frac{CPI_t}{CPI_{t-1}}\right) - 1200\log\left(\frac{CPI_{t-1}}{CPI_{t-2}}\right). \tag{18}$$

Then, the forecasting equation for y_t^h (analogous to Equation 1) is defined as

$$y_t^h = \omega + \phi(L)z_{t-h} + \beta'(L)\hat{F}_{t-h}. \tag{19}$$

The orders of the lag polynomials $\phi(L)$ and $\beta'(L)$ are independently determined by the Bayesian Information Criterion (BIC), with the maximum number of lags set to six if the sample size permits, and four otherwise.[2] The forecast horizons, h, considered in this paper are 1, 6, 12, and 24 months.

Following Bai and Ng (2008), for PC, SQPC and KPC, the number of factors is fixed to 10. For the linear PLS models, following Fuentes et al. (2015), I do not fix the number of factors to some particular value. Instead, the number of factors is set to what yields the best forecasting performance, limiting the search to two components. Finally, the number of KPLS components is set to one due to the computationally expensive estimation procedure concerning the grid search for the adjustable Kernel coefficient.

Next to the above, PLS and SQPLS approaches (b) and (c) have an additional unspecified parameter. For approach (b) this is the number of lags of the target variable, z_t, used to enlarge X, while for approach (c) this corresponds to the lag order of the AR model fitted to the target y_t^h. These parameters are not fixed to any particular value. Instead, I perform a grid search over the set $\{1,...,6\}$, and allow these parameters to be determined by the final forecasting performance.

[2] The BIC formula used here corresponds to that in Bai and Ng (2008): BIC = $\log(\hat{\sigma}^2) + n\frac{\log(T)}{T}$, where n is the number of explanatory variables, T is the sample size, and $\hat{\sigma}^2$ is the sum of squared residuals of the model, divided by T.

Table 2: Estimation and forecast subsamples

Subsample	Estimation Subsample	Forecast Subsample
M1	03/1960 to 03/1970 – h	03/1970 to 12/1980
M2	03/1960 to 03/1980 – h	03/1980 to 12/1990
M3	03/1960 to 03/1990 – h	03/1990 to 12/2000
M4	03/1960 to 03/1970 – h	03/1970 to 12/1990
M5	03/1960 to 03/1970 – h	03/1970 to 12/2000
M6	03/1960 to 03/1980 – h	03/1980 to 12/2000
M7	03/1960 to 03/1970 – h	03/1970 to 12/2003

* h refers to the forecast horizon.

In the spirit of Fuentes et al. (2015) and Bai and Ng (2008), I use an AR(4)-type model of the form

$$y_t^h = \mu + \sum_{i=0}^{3} \theta_i z_{t-h-i} + \epsilon_t, \tag{20}$$

as a benchmark. To compare the forecasting performance among models, I employ the relative mean squared forecast error (RMSE), defined as

$$\text{RMSE (Method)} = \frac{\text{MSE (Method)}}{\text{MSE (Benchmark)}}, \tag{21}$$

such that a value less than one indicates that the model under consideration beats the benchmark.

To account for the time-varying relationships between the target and predictor variables, this paper follows the same approach as in Fuentes et al. (2015), and evaluates the forecasting performance over seven forecast subsamples. All forecasts are generated using an expanding window, in which the start of the training sample is always set as March, 1960. For instance, when generating a one-step-ahead forecast for March, 1970, the estimation sample used to extract factors and estimate the forecasting equation consists of 120 observations from March, 1960 up to February, 1970. For the next forecast, the sample is extended by one observation, up to March, 1970. The procedure is carried on similarly until the end of the forecast sample is reached. An overview of these estimation and forecast subsamples is given in Table 2.

3.2. Main Results

Table 3 reports the forecasting performance results for $h = 1$ and $h = 6$, while results concerning $h = 12$ and $h = 24$ are given in Table 4. Note that Tables 3 and 4 display the results in the follow-

ing way: Panel A includes linear methods, Panel B concerns the squared extensions, while Panel C represents the Kernel extensions. The discussion of these empirical results will be structured in two parts. First, the performance of the linear methods will be discussed, accompanied with a comparison to the findings of Fuentes et al. (2015). Then, a comparison between linear and non-linear methods will be considered.

3.2.1. Linear Models of Fuentes et al. (2015)

Beginning with $h = 1$, as shown in Panel A of Table 3, many values close to one indicate that the methods under consideration have difficulties in beating the AR(4) benchmark, which is particularly true for PC, PLS(a) and PLS(c). Nevertheless, in all seven subsamples, PLS(c) yields the best forecasting performance, with values in-between around 0.92 and 0.95. Besides, PLS(b) shows the most inferior performance among all linear models in every subsample. Although the figures for $h = 1$ do not perfectly mirror those of Fuentes et al. (2015), all the findings discussed above remain the same.

For $h = 6$, in contrast to Fuentes et al. (2015), who find PLS(a) to be the best-performing model in all subsamples, the findings presented in Panel A of Table 3 indicate that PLS(c) yields superior forecasting accuracy in six subsamples, while PLS(a) outperforms the other methods only in subsample M6. That being said, we should keep in mind that the differences in RMSE between these two methods are minor, varying from 0.001 to around 0.03. Next, apart from M1, these results match with those of Fuentes et al. (2015) in the sense that both PLS(a) and PLS(c) provide better performance than PC, although the values presented here are consistently higher than those reported in Fuentes et al. (2015).

For $h = 12$, as can be seen from Panel A of Table 4, PLS(c) outperforms the other linear methods in four out of the seven subsamples (M1-M3 and M6), while PLS(a) yields the lowest RMSE in the other three subsamples. Similar to the horizon $h = 6$, PLS(a) and PLS(c) always show superior performance in comparison to PC, while PLS(b) is the worst-performing model in all subsamples. All these findings are similar to what is suggested in Fuentes et al. (2015), despite that RMSEs tend to deviate, especially for PLS(a) and PLS(c).

For $h = 24$, results in Panel A of Table 4 indicate that PLS(a) outperforms the other linear methods in the first subsample only, while PLS(c) yields the lowest RMSE in all other subsamples. Again, the differences in RMSE between PLS(a) and PLS(c) are minor, and both methods consistently outperform PC. PLS(b) tends to be the inferior model in all subsamples.

Based on the above discussion, some key results can be distinguished. First, PLS(c) tends to be the best-performing linear model in terms of forecasting accuracy for almost all forecast horizons and subsamples, followed by PLS(a), which seems to perform only marginally worse. Second, PLS(b) turns out to be an inferior model, sometimes even yielding RMSEs larger than one, indicating less forecasting accuracy than the benchmark AR(4). This is a relevant issue also in Fuentes et al. (2015), where the authors argue that this is due to PLS(b) giving weights to all predictors, such that since the cross-sectional dimension is large, the target and its lags are only allocated a relatively small weight. Finally, it can be seen that both PLS(a) and PLS(b) perform better than PC, especially for the medium-term forecast horizons, thus indicating the advantage of supervised dimensionality reduction techniques in the context of inflation forecasting.

As mentioned, the forecasting performance results reported here tend to deviate considerably from what is shown in Fuentes et al. (2015), although many conclusions remain the same. There are several possible reasons to explain such differences. For the sake of transparency and for future reference, I mention only a few of the most plausible arguments:

1. First, Fuentes et al. (2015) do not mention whether data should be demeaned or standardized before applying the dimensionality reduction techniques. Based on related literature and forecasting performance, this paper chooses to standardize variables to have zero mean and unit variance.

2. For PLS(b), it is unclear how many lags of the target variable are used to enlarge the predictor matrix, while for PLS(c), it is not stated which order of the AR model should be fitted to the target variable. Furthermore, it is ambiguous as to what the paper refers to as the target variable (i.e., y_t^h or z_t).

3. The procedure for extracting the second PLS factor in Fuentes et al. (2015) is not very precisely written. In particular, the authors do not acknowledge the fact that, when extracting PLS components, there are two variants of the predictor matrix (i.e., X and X_h in Section 2).

4. Regarding the forecasting procedure, Fuentes et al. (2015) do not explicitly indicate whether an expanding or rolling window approach should be used. Based on related literature, I incorporate the former. Additionally, although Fuentes et al. (2015) argues that the lag order selection for the forecasting equation is based on the BIC, the maximum number of lags over which BIC is minimized is not clear.

Table 3: RMSEs for all models, $h = 1$ and $h = 6$

Panel A: Linear Methods								
	h = 1				**h = 6**			
Period	PC (k=10)	PLS (a)	PLS (b)	PLS (c)	PC (k=10)	PLS (a)	PLS (b)	PLS (c)
M1	1.037	1.101	1.305	0.934	0.686	0.703	1.121	0.672
M2	0.971	0.979	1.024	0.920	0.708	0.635	0.990	0.633
M3	0.941	0.955	1.121	0.928	0.726	0.627	1.173	0.624
M4	1.015	1.050	1.178	**0.946**	0.708	0.701	1.135	0.683
M5	1.000	1.032	1.167	**0.942**	0.699	0.683	1.121	0.669
M6	0.959	0.969	1.072	0.922	0.695	0.619	0.994	0.620
M7	0.986	1.020	1.163	**0.934**	0.698	0.680	1.142	0.668
Panel B: Squared Methods								
	h = 1				**h = 6**			
Period	SQPC (k=10)	SQPLS (a)	SQPLS (b)	SQPLS (c)	SQPC (k=10)	SQPLS (a)	SQPLS (b)	SQPLS (c)
M1	1.150	1.283	1.472	1.038	0.831	0.686	1.242	0.657
M2	1.004	0.957	1.042	0.916	0.690	0.623	1.342	0.623
M3	**0.875**	0.941	1.101	0.903	0.626	0.623	1.298	0.633
M4	1.080	1.132	1.254	0.993	0.770	0.687	1.432	0.671
M5	1.045	1.097	1.254	0.978	0.746	0.673	1.407	0.660
M6	0.966	0.947	1.093	0.915	0.668	0.612	1.314	0.614
M7	1.020	1.078	1.262	0.963	0.742	0.676	1.426	0.664
Panel C: Kernel Methods								
	h = 1				**h = 6**			
Period	KPC1 (k=10)	KPC2 (k=10)	KPLS1 (k=1)	KPLS2 (k=1)	KPC1 (k=10)	KPC2 (k=10)	KPLS1 (k=1)	KPLS2 (k=1)
M1	0.961	0.964	**0.919**	0.922	**0.633**	0.665	0.789	0.783
M2	**0.909**	0.939	0.922	0.993	**0.609**	0.627	0.854	0.850
M3	0.909	0.926	0.883	0.921	**0.597**	0.658	0.751	0.807
M4	0.972	0.963	0.981	0.992	**0.655**	0.677	0.837	0.833
M5	0.962	0.959	0.967	0.982	**0.647**	0.667	0.829	0.825
M6	**0.909**	0.937	0.912	0.980	**0.605**	0.621	0.838	0.835
M7	0.953	0.949	0.974	0.979	**0.648**	0.666	0.831	0.826

Notes: Best performance for each subsample-forecast horizon pair is highlighted in bold. For PC and SQPC, the number of factors is fixed to 10. For Kernel methods, the number of factors is fixed to one. For the other methods, the number of factors is chosen based on the final predictive performance, limiting the search to 2 components. The hyperparameter σ, which is present in Kernel methods, is optimized based on the final predictive performance based on a grid search over $\sigma \in \{0.5\sigma_0, \sigma_0, 2\sigma_0, 4\sigma_0, 8\sigma_0\}$, where $\sigma_0 = \sqrt{(N+2)/2}$ for the polynomial Kernel, and $\sigma_0 = \sqrt{c_N}/\pi$ for the Gaussian radial basis Kernel, where c_N is the 95-th percentile of the χ^2 distribution with N degrees of freedom.

3.2.2. Linear Models Against Non-linear Extensions

We now turn to the comparison between linear and non-linear models. As indicated by the RMSEs in Tables 3 and 4, the results in this regard are somewhat mixed. For the forecast

Table 4: RMSEs for all models, $h = 12$ and $h = 24$

Panel A: Linear Methods								
	h = 12				**h = 24**			
Period	PC (k=10)	PLS (a)	PLS (b)	PLS (c)	PC (k=10)	PLS (a)	PLS (b)	PLS (c)
M1	0.706	0.696	0.994	0.692	0.789	**0.565**	0.824	0.566
M2	0.622	0.592	0.875	0.577	0.522	0.515	0.695	0.493
M3	0.774	0.603	1.179	**0.586**	0.632	0.560	1.206	0.541
M4	0.657	0.632	0.946	0.641	0.665	0.560	0.779	**0.550**
M5	0.662	0.630	0.955	0.638	0.664	0.564	0.812	**0.555**
M6	0.637	0.588	0.902	0.574	0.539	0.520	0.766	**0.504**
M7	0.664	0.631	0.980	0.640	0.666	0.567	0.838	**0.557**
Panel B: Squared Methods								
	h = 12				**h = 24**			
Period	SQPC (k=10)	SQPLS (a)	SQPLS (b)	SQPLS (c)	SQPC (k=10)	SQPLS (a)	SQPLS (b)	SQPLS (c)
M1	0.838	0.711	1.113	0.712	1.160	0.620	0.917	0.636
M2	0.582	**0.542**	1.085	0.542	0.604	0.502	0.829	**0.490**
M3	0.629	0.657	1.297	0.665	**0.514**	0.521	1.292	0.550
M4	0.701	**0.616**	1.176	0.625	0.902	0.564	0.992	0.577
M5	0.691	**0.615**	1.177	0.624	0.873	0.564	1.016	0.576
M6	0.584	**0.553**	1.106	0.556	0.593	0.509	0.889	0.509
M7	0.696	**0.623**	1.217	0.634	0.886	0.571	1.046	0.583
Panel C: Kernel Methods								
	h = 12				**h = 24**			
Period	KPC1 (k=10)	KPC2 (k=10)	KPLS1 (k=1)	KPLS2 (k=1)	KPC1 (k=10)	KPC2 (k=10)	KPLS1 (k=1)	KPLS2 (k=1)
M1	**0.662**	0.667	1.066	1.062	0.602	0.602	0.674	0.674
M2	0.631	0.629	0.900	0.936	0.639	0.616	0.552	0.551
M3	0.669	0.721	0.718	0.730	0.615	0.636	0.974	0.977
M4	0.638	0.642	0.989	0.988	0.632	0.627	0.618	0.616
M5	0.640	0.645	0.964	0.964	0.632	0.632	0.646	0.644
M6	0.637	0.640	0.874	0.902	0.640	0.625	0.612	0.611
M7	0.647	0.650	0.955	0.955	0.640	0.639	0.642	0.640

Notes: Best performance for each subsample-forecast horizon pair is highlighted in bold. For PC and SQPC, the number of factors is fixed to 10. For Kernel methods, the number of factors is fixed to one. For the other methods, the number of factors is chosen based on the final predictive performance, limiting the search to 2 components. The hyperparameter σ, which is present in Kernel methods, is optimized based on the final predictive performance based on a grid search over $\sigma \in \{0.5\sigma_0, \sigma_0, 2\sigma_0, 4\sigma_0, 8\sigma_0\}$, where $\sigma_0 = \sqrt{(N+2)/2}$ for the polynomial Kernel, and $\sigma_0 = \sqrt{c_N}/\pi$ for the Gaussian radial basis Kernel, where c_N is the 95-th percentile of the χ^2 distribution with N degrees of freedom.

horizon $h = 1$ (see Table 3), non-linear methods dominate in subsamples M1-M3 and M6, while linear methods perform better in the remaining three subsamples. In M1-M3 and M6, the use of non-linear methods reduces the RMSE by 0.02 to 0.05, as compared to the best-performing

Table 5: Diebold-Mariano test (based on MSE loss) for a formal pariwise comparison of the predictive accuracy of the best-performing linear and non-linear models for every subsample and forecast horizon

Period	h=1	h=6	h=12	h=24
M1	2.127 **	1.597 *	1.133	-1.582 *
	(0.02)	(0.06)	(0.13)	(0.06)
M2	2.012 **	1.562 *	1.111	0.193
	(0.02)	(0.06)	(0.13)	(0.42)
M3	4.416 ***	0.369	-0.554	0.343
	(0.00)	(0.36)	(0.29)	(0.37)
M4	-3.077 ***	2.144 **	0.913	-0.985
	(0.00)	(0.02)	(0.18)	(0.16)
M5	-2.856 ***	1.533 *	0.728	-0.558
	(0.00)	(0.06)	(0.23)	(0.29)
M6	2.176 **	0.726	0.559	-0.198
	(0.01)	(0.23)	(0.29)	(0.42)
M7	-2.947 ***	1.512 *	0.391	-0.879
	(0.00)	(0.07)	(0.35)	(0.19)

Notes: Symbols *, **, and *** indicate significance at the 10%, 5%, and 1% level, respectively. The *p*-values given in parentheses. Positive test statistics indicate better performance of non-linear methods; negative test statistics indicate better performance of linear methods.

linear model. Nonetheless, even in the case of over-performance, there is no clear winner among non-linear models.

Surprisingly, for $h = 6$ (see Table 3), KPC with a polynomial Kernel function beats all other models in every subsample. RMSEs for this best-performing model are quite consistent among different samples, in a range of around 0.60 to 0.65.

For $h = 12$ (see Table 4), PLS(c) dominates all models only in subsample M3. In the remaining six subsamples, non-linear methods show superior performance. In particular, KPC with a polynomial Kernel function performs best in the first subsample, while SQPLS(a) dominates all models in the remaining cases.

Finally, for the horizon $h = 24$ (see Table 4), non-linear methods perform better than the linear ones only in subsamples M2 and M3. The corresponding superior models in these cases are SQPLS(c) and SQPC, respectively. In the remaining subsamples, linear PLS methods tend to provide more accurate forecasts, as indicated by the RMSE.

Based on the above discussion, it is not straightforward to identify a winning class of models. To summarize, I find that, besides from the horizon $h = 6$, results are mixed up: linear methods over-perform their non-linear extensions in some subsamples, while the opposite holds

true in the remaining cases. For $h = 6$, however, it seems that we can identify a consistently better model, that being the KPC with a polynomial Kernel function. Another important finding concerns the rather poor performance of KPLS methods, especially for the longer-term forecast horizons. This relates directly to the discussion in the methodology section regarding KPLS, where I acknowledge that this method, as defined here, is only able to optimize the contemporaneous covariance between the predictors and the target variable. This drawback should certainly become more evident as h increases, a feature which is visible in these results.

Although this discussion sheds light on the comparison between the linear and non-linear classes of models, I have not yet formally tested the differences in forecasting accuracy. To address this, Table 5 reports the test statistics and their corresponding p-values of the Diebold-Mariano (DM) test (see Diebold and Mariano (2002) for details) to formally compare the predictive performance of the best-performing linear model and the best-performing non-linear model in each subsample and forecast horizon.

As an example of how to read Table 5, I consider the case of $h = 1$ and subsample M1. In Table 3, I noted that in this case, the best-performing linear model is PLS(c) and the best-performing non-linear model is KPLS with a polynomial Kernel function. Table 5 now applies the DM test over the squared forecast errors of these two models. Positive entries indicate that non-linear methods (in this case, KPLS using a polynomial Kernel function) perform better, and negative entries suggest the opposite.

From the results in Table 5, it is immediately apparent that the forecast accuracy for $h = 12$ and $h = 24$ is not significantly different between the linear and non-linear models. Of particular importance is the horizon $h = 6$, as I previously found that KPC with a polynomial Kernel function out-performs in all subsamples. This observation is confirmed here by the positive DM test statistics. However, we can see that the higher accuracy of these forecasts is significant only at the 10% level in four subsamples, and at the 5% level in one subsample. In the remaining two subsamples, the performance is not significantly different from what linear methods yield. Finally, for $h = 1$, we can see that in all subsamples in which non-linear methods dominate, the performance gain is significant at the 5% level. However, in the remaining subsamples, linear models show statistically significant superior performance even at the 1% level, as indicated by the DM test.

3.3. Robustness Checks

The majority of the models considered in this paper are subject to a number of unspecified parameters, which I choose based on the final forecasting performance. This is, of course, possible only from the benefit of hindsight. Practitioners using such models must rely on other approaches to optimize over such parameters, given the available data. Examples include minimizing some in-sample loss function or preferably implementing (forecast-oriented) cross-validation techniques. Therefore, it is essential for forecasters to realize how robust are the results presented over the previous sections with respect to these adjustable parameters.

Additionally, all results presented so far are based on RMSE. Indeed, forecasters might have different tolerances toward forecast errors that are not reflected in the mean squared prediction error. Examples include asymmetric risk attitudes towards small and large errors or between over- and under-estimation of the target variable. Therefore, it is crucial to evaluate the forecasts obtained via different models based on several loss functions that reflect such preferences.

In what follows, I aim to address the issues mentioned in the above discussion. I start with a sensitivity analysis of the forecasting performance of linear and non-linear models concerning their adjustable parameters. Next, additional loss functions will be considered alongside RMSE.

3.3.1. Robustness of the Models of Fuentes et al. (2015)

In this section, I discuss the robustness of the forecasting results of the linear models with respect to their adjustable parameters. Since the results presented in Section 3.2.1 suggest that PLS(c) is one of the best-performing linear models, while PLS(b) is inferior in most subsample-forecast horizon pairs, I choose to focus our attention on PLS(c) here. Meanwhile, results corresponding to PLS(b) are included in Section A of the Appendix.

Table 6 displays the sensitivity analysis results for PLS(c) concerning the lag order, p, of the AR(p) process fitted to the target variable. A quick glance at the results suggests that there is no consistency in the optimal lag order p that yields the best forecasting performance. Nevertheless, some interesting results can be mentioned. The average p's that yield the best results for the 1-, 6-, 12-, and 24-month horizons are approximately 4, 4, 2, and 3, respectively. Therefore, one can argue that there is a decreasing trend in the optimal p with respect to the forecast horizon. Another evidence for this is that $p = 6$ yields the worst performance in all subsamples for $h = 24$ and in six out of seven subsamples for $h = 12$. On the contrary, $p = 1$

Table 6: RMSEs for PLS (c) for every value p of the AR(p) process fitted to the target variable

	$p = 1$	$p = 2$	$p = 3$	$p = 4$	$p = 5$	$p = 6$
			Subsample M1			
h = 1	1.051	0.973	0.963	0.937	0.939	**0.934**
h = 6	0.696	0.718	0.698	**0.672**	0.673	0.673
h = 12	**0.692**	0.711	0.706	0.708	0.700	0.723
h = 24	**0.566**	0.609	0.579	0.598	0.670	0.790
			Subsample M2			
h = 1	0.924	**0.920**	0.922	0.922	0.927	0.928
h = 6	0.642	0.658	0.648	**0.633**	0.635	0.635
h = 12	0.583	0.578	**0.577**	0.580	0.610	0.643
h = 24	0.502	0.493	**0.493**	0.508	0.550	0.586
			Subsample M3			
h = 1	0.951	0.939	0.942	0.938	0.928	**0.928**
h = 6	0.633	**0.624**	0.632	0.642	0.649	0.655
h = 12	0.592	0.589	0.588	0.589	**0.586**	0.587
h = 24	**0.541**	0.580	0.582	0.609	0.671	0.730
			Subsample M4			
h = 1	1.005	0.963	0.960	**0.946**	0.950	0.948
h = 6	0.701	0.720	0.705	**0.683**	0.684	0.685
h = 12	**0.641**	0.655	0.644	0.662	0.669	0.677
h = 24	0.558	0.569	0.551	**0.550**	0.602	0.680
			Subsample M5			
h = 1	0.995	0.957	0.955	**0.942**	0.945	0.943
h = 6	0.684	0.701	0.687	**0.669**	0.671	0.672
h = 12	**0.638**	0.651	0.640	0.656	0.662	0.669
h = 24	0.562	0.572	0.555	**0.555**	0.606	0.682
			Subsample M6			
h = 1	0.929	**0.922**	0.924	0.922	0.924	0.924
h = 6	0.626	0.640	0.631	**0.620**	0.623	0.624
h = 12	0.578	0.575	**0.574**	0.576	0.600	0.627
h = 24	0.505	**0.504**	0.504	0.521	0.565	0.604
			Subsample M7			
h = 1	0.981	0.946	0.945	**0.934**	0.936	0.934
h = 6	0.685	0.701	0.686	**0.668**	0.670	0.671
h = 12	**0.640**	0.654	0.644	0.659	0.664	0.669
h = 24	0.565	0.574	0.558	**0.557**	0.606	0.679

Note: Best performance for each subsample-forecast horizon pair is highlighted in bold.

yields the highest RMSE in six subsamples for the horizon $h = 1$.

Next, it should be noted that $p = 5$ and $p = 6$ are optimal in only one and two subsample-forecast horizon pairs, respectively. Additionally, the exclusion of these lag orders from the analysis would not lead to a substantial deterioration in the forecasting performance. This

Table 7: RMSEs of KPC1 for every value of the tunable parameter, σ, in the search space

	$\sigma = 0.5\sigma_0$	$\sigma = \sigma_0$	$\sigma = 2\sigma_0$	$\sigma = 4\sigma_0$	$\sigma = 8\sigma_0$
			Subsample M1		
h = 1	1.470	1.376	0.969	**0.961**	0.963
h = 6	1.539	1.246	**0.633**	0.654	0.662
h = 12	1.238	0.767	**0.662**	0.672	0.666
h = 24	2.188	0.734	0.612	0.604	**0.602**
			Subsample M2		
h = 1	1.041	**0.909**	0.957	0.956	0.957
h = 6	1.036	0.782	**0.609**	0.620	0.625
h = 12	1.142	0.894	0.635	**0.631**	0.639
h = 24	1.119	0.954	0.655	0.647	**0.639**
			Subsample M3		
h = 1	0.939	0.924	**0.909**	0.920	0.924
h = 6	0.645	**0.597**	0.636	0.653	0.657
h = 12	0.730	**0.669**	0.691	0.713	0.719
h = 24	0.842	0.651	**0.615**	0.630	0.634
			Subsample M4		
h = 1	1.246	1.197	0.978	**0.972**	0.973
h = 6	1.137	0.969	**0.655**	0.670	0.675
h = 12	1.188	0.828	**0.638**	0.640	0.641
h = 24	1.676	0.853	0.642	0.636	**0.632**
			Subsample M5		
h = 1	1.196	1.150	0.964	**0.962**	0.964
h = 6	1.085	0.925	**0.647**	0.661	0.666
h = 12	1.148	0.813	**0.640**	0.642	0.644
h = 24	1.614	0.838	0.641	0.637	**0.632**
			Subsample M6		
h = 1	1.013	**0.909**	0.940	0.945	0.947
h = 6	0.974	0.749	**0.605**	0.615	0.620
h = 12	1.079	0.857	0.639	**0.637**	0.645
h = 24	1.083	0.914	0.651	0.646	**0.640**
			Subsample M7		
h = 1	10.055	1.134	0.955	**0.953**	0.956
h = 6	26.155	0.992	**0.648**	0.660	0.664
h = 12	23.399	0.901	**0.647**	0.648	0.650
h = 24	24.165	0.924	0.649	0.644	**0.640**

Notes: This table displays the RMSE of KPC1 (i.e., KPC using a polynomial kernel function) for every value of the Kernel tunable coefficient, σ, that the performance is evaluated on. In this case, $\sigma \in \{0.5\sigma_0, \sigma_0, 2\sigma_0, 4\sigma_0, 8\sigma_0\}$, where $\sigma_0 = \sqrt{(N+2)/2}$, in which N represents the total number of potential predictors. Best performance for each subsample-forecast horizon pair is highlighted in bold.

suggests that, in practical (real-time forecasting) applications, the forecaster might also reduce the search set to $\{1, ..., 4\}$ for a more computationally efficient procedure.

Overall, the values displayed in Table 6 indicate that there is considerable variation in

RMSE among the different values of p, such that the forecasting results are not invariant to p. Therefore, the results in Section 3.2.1 must be interpreted with caution and forecasters should be careful when optimizing p in a real-time forecasting exercise. It is left for future research to determine whether the same conclusion would hold true if the selection of p is automatically incorporated in the forecasting procedure, for instance, by means of cross-validation.

3.3.2. Robustness of the Kernel Methods

In this section, the robustness of the Kernel models with respect to the choice of the adjustable parameter σ is discussed. Due to the superior performance found in Section 3.2.2, the focus here is on KPC1 (i.e., KPC using a polynomial Kernel function). Results concerning the other Kernel methods are displayed in Section B of the Appendix.

Table 7 displays the sensitivity analysis results concerning the adjustable parameter, σ, used in KPC1. It is immediately apparent that the forecasting performance is highly sensitive to the choice of σ for all subsamples and forecast horizons. As an example, RMSE ranges from 0.64 to around 24.17 in subsample M7 for $h = 24$. This confirms the findings of Exterkate et al. (2016) and Giovannelli (2012) that the forecasting accuracy of the Kernel methods is highly dependent on the adjustable parameter. The worst performance is generally achieved when using $\sigma = 0.5\sigma_0$. Again, this is in line with Giovannelli (2012), who argues that underestimating σ leads Kernel methods over-fitting to noise in the training sample and consequentially performing poorly out-of-sample.

It is important to note that all of the findings discussed above also hold for the particular case of $h = 6$, in which we found KPC1 to be the best-performing model in all subsamples. Therefore, such conclusions should be interpreted with great caution, and it is difficult to extrapolate them to an actual real-time forecasting exercise without additional knowledge. Therefore, it is left for future research to investigate inflation forecasting using Kernel methods, which incorporates an automated selection procedure for the parameter σ (e.g., via cross-validation).

3.3.3. Forecast Evaluation with Different Loss Functions

In addition to the RMSE as a measure of forecast accuracy, this paper aims to account for heterogeneous preferences of forecasters concerning the prediction error by using other loss functions. Diebold and Mariano (2002), among others, argue for the relevance of considering different loss functions when evaluating and comparing forecasts, as the standard method of MSE might

poorly asses the actual economic loss associated with forecast errors. In this regard, Stock and Watson (1998) use loss functions of the form

$$C_q = \sum_{t=1}^{T} |e_t|^q, \qquad \text{for} \quad q \in \{1, 1.5, 2, 2.5, 3\}, \tag{22}$$

where T is the forecast sample size and e_t is the forecast error in time t.[3] Stock and Watson (1998) use these different values of q to represent the various types of economic losses associated with forecast errors. In particular, the authors argue that such a loss function with $q = 1$ uniformly penalizes across large and small errors and therefore represent a forecaster who is equally concerned about small and large forecast errors. For $q = 3$, this function most heavily penalizes large errors (Stock & Watson, 1998).

This paper chooses to implement and report the loss function in Equation 22 only in the two extreme cases of $q = 1$ and $q = 3$ for conciseness. From this point onward, I will refer to these as $CF2$ and $CF3$, respectively. To be noted is that for $q = 2$, the function in Equation 22 corresponds to the MSE, which I will refer to as $CF1$. Based on the loss functions proposed in Stock and Watson (1998), this paper extends the forecast evaluation methodology by including asymmetric loss functions. In particular, I propose the following two additional loss functions:

$$CF4 = \sum_{t=1}^{T} \left[\mathbb{1}(e_t > 0) \, |e_t| \; + \; \mathbb{1}(e_t \leq 0) \, |e_t|^3 \right], \tag{23}$$

and

$$CF5 = \sum_{t=1}^{T} \left[\mathbb{1}(e_t > 0) \, |e_t|^3 \; + \; \mathbb{1}(e_t \leq 0) \, |e_t| \right], \tag{24}$$

where $\mathbb{1}(.)$ represents the indicator function, and the rest are defined as in Equation 22. The expression in Equation 23 penalizes under-estimations of the target variable more heavily than over-estimations, while the opposite holds true for what is presented in Equation 24. This is because, as mentioned, the cubic loss penalizes errors more heavily than the solely taking their absolute value.

According to Diebold and Mariano (2002), in the majority of real-life scenarios where forecasts guide decision-making, the associated loss is generally asymmetric. As such, $CF4$ and $CF5$ might be highly-relevant evaluation tools for choosing an appropriate forecasting model,

[3] The forecast error is computed as the forecasted value minus the realization of the target variable.

Table 8: Best performing model for each subsample and forecast horizon according to different cost functions

	$CF1$	$CF2$	$CF3$	$CF4$	$CF5$
			Subsample M1		
h = 1	KPLS1	KPLS2	PLS (c)	PLS (c)	KPLS1
h = 6	KPC1	KPC1	KPC1	PLS (c)	KPC1
h = 12	KPC1	KPC1	KPC1	KPC1	PC
h = 24	PLS (a)	PLS (c)	PLS (c)	KPLS1	PLS (c)
			Subsample M2		
h = 1	KPC1	PLS (c)	KPC1	SQPLS (c)	KPLS1
h = 6	KPC1	KPC1	KPC1	KPLS1	KPC1
h = 12	SQPLS (a)	PLS (c)	SQPLS (c)	PLS (c)	SQPC
h = 24	SQPLS (c)	PLS (c)	SQPLS (c)	KPLS1	SQPC
			Subsample M3		
h = 1	SQPC	SQPC	SQPC	KPC1	PLS (c)
h = 6	KPC1	KPC1	KPC1	KPC1	PLS (c)
h = 12	PLS (c)	PLS (c)	PLS (c)	KPC1	PC
h = 24	SQPC	SQPLS (a)	PLS (c)	SQPLS (c)	PC
			Subsample M4		
h = 1	PLS (c)	KPC2	PLS (c)	SQPLS (c)	KPLS1
h = 6	KPC1	KPC1	KPC1	PLS (c)	KPC1
h = 12	SQPLS (a)	SQPLS (a)	SQPLS (a)	PLS (c)	SQPC
h = 24	PLS (c)	PLS (c)	PLS (c)	KPLS1	PC
			Subsample M5		
h = 1	PLS (c)	PLS (c)	PLS (c)	SQPLS (c)	KPLS1
h = 6	KPC1	KPC1	KPC1	PLS (c)	KPC1
h = 12	SQPLS (a)	PLS (c)	SQPLS (a)	PLS (c)	SQPC
h = 24	PLS (c)	PLS (c)	PLS (c)	KPLS1	PC
			Subsample M6		
h = 1	KPC1	PLS (c)	KPLS1	KPLS1	KPLS1
h = 6	KPC1	SQPC	KPC1	PLS (c)	KPC1
h = 12	SQPLS (a)	PLS (c)	SQPLS (a)	PLS (c)	SQPC
h = 24	PLS (c)	PLS (c)	SQPLS (c)	KPLS1	SQPC
			Subsample M7		
h = 1	PLS (c)	PLS (c)	PLS (c)	SQPLS (c)	KPLS1
h = 6	KPC1	KPC1	KPC1	PLS (c)	KPC1
h = 12	SQPLS (a)	PLS (c)	SQPLS (a)	PLS (c)	SQPC
h = 24	PLS (c)	PLS (c)	PLS (c)	KPLS1	PC

depending on the specific applications. As an example, banks or other financial institutions, who rely on inflation rate forecasts when determining the interest rates on their mortgages or other securities, might benefit from an over-estimation of future inflation in order to mitigate the inflation rate risk. For this reason, CF4 might give a better representation of this economic

loss than symmetric functions.

As a point of reference to compare the asymmetric loss functions $CF4$ and $CF5$ introduced above, this paper also evaluates the models based on other well-established asymmetric loss functions in the literature. In particular, the piece-wise linear and the asymmetric squared loss function, as defined in Elliott and Timmermann (2004), will be considered. The piece-wise linear loss function is defined as

$$L = \sum_{t=1}^{T} \Big[\mathbb{1}(e_t > 0)\, |e_t|\, (1-\delta) \;+\; \mathbb{1}(e_t \leq 0)\, |e_t|\, \delta \Big], \tag{25}$$

and the asymmetric squared loss can be written as

$$L = \sum_{t=1}^{T} \Big[\mathbb{1}(e_t > 0)\, {e_t}^2\, (1-\delta) \;+\; \mathbb{1}(e_t \leq 0)\, {e_t}^2\, \delta \Big], \tag{26}$$

where $\delta \in (0,1)$ and the rest are defined as in Equation 22. Although Elliott and Timmermann (2004) evaluate the expressions in Equations 25 and 26 for δ ranging from 0.1 to 0.9 with increments of 0.1, for conciseness this paper only reports the two extreme cases. Note that for $\delta = 0.1$, these loss functions most heavily penalize over-estimations of the target variable (as in $CF5$), while for $\delta = 0.9$, they assign more weight to under-estimations (as in $CF4$).

Table 8 shows the best-performing model for each subsample and forecast horizon, as indicated by the different performance measures employed in this paper.[4] At first glance, we can see that different loss functions tend to favor different models for many subsamples and forecast horizons. According to $CF2$ and $CF3$, PLS(c) again shows superior performance in comparison to the other linear methods. In fact, PLS(c) dominates all other models (both linear and non-linear) in 15 out of 28 subsample-forecast horizon pairs. For the remaining cases, results show that non-linear models tend to dominate. KPC using a polynomial Kernel function again shows superior performance for $h = 6$ according to both $CF2$ and $CF3$.

The asymmetric loss functions, $CF4$ and $CF5$, provide greater support in favor of non-linear models. In particular, both $CF4$ and $CF5$ indicate that non-linear models are the best-performing in 18 out of 27 subsample-forecast horizon pairs. According to $CF4$, all the remaining scenarios are dominated by PLS(c), while $CF5$ surprisingly indicates a rather good performance for PC, especially for the horizons $h = 12$ and $h = 24$.

[4] Table 8 only reports the best-performing models. For the interested reader, Section C in the Appendix provides the actual results based on different performance measures, relative to the AR(4) benchmark, for each model.

Results concerning the piece-wise linear and asymmetric squared loss functions are included in Section C of the Appendix to conserve space. As can be seen, these loss functions lead to somewhat similar conclusions as $CF4$ and $CF5$. In particular, the highest similarity is found between $CF4$ and the asymmetric squared loss with $\delta = 0.9$ and especially between $CF5$ and the asymmetric squared loss with $\delta = 0.1$. The number of subsample-forecast horizon combinations in which these pairs loss functions lead to the same result are 15 and 22 out of 28, respectively.

4. Conclusion

This thesis sheds light on the comparison between linear and non-linear dimensionality reduction techniques in the context of inflation rate forecasting using big datasets. In particular, I consider PCA, PLS(a), PLS(b) and PLS(c), introduced in Fuentes et al. (2015), as representative of linear models. In terms of non-linear methods, I consider the squared extensions of the models mentioned above, together with KPC and KPLS. The performance of each of these models is evaluated over seven subsamples using the well-known Stock and Watson (2005) database. Besides, sensitivity analyses concerning unspecified model parameters are performed, and the predictive accuracy is not only measured by the popular RMSE, but also in terms of four additional loss functions, which aim to represent the economic loss associated with the forecast error in different applications.

Our empirical application yields several worth-noting results. First, I find evidence that PLS(c) tends to be the best-performing linear model in terms of forecasting accuracy for nearly all forecast horizons and their corresponding subsamples. PLS(a) follows the lead by only slightly higher RMSEs. It should be noted, however, that the latter model might be more appropriate in real-time applications, as it is prone to less unspecified parameters, which make PLS(c) prone to robustness issues. Comparing linear models to their non-linear extensions, I find that there is no significant difference in predictive accuracy for the horizons $h = 12$ and $h = 24$. While for $h = 1$, conclusions are somewhat mixed-up and subsample-dependent, I find that KPC with a polynomial Kernel function outperforms all other models in every subsample for $h = 6$. This performance gain is significant (at least) at the 10% level in 5 out of 7 subsamples. Nevertheless, it should be acknowledged that, based on our sensitivity analysis, it is difficult to extrapolate this success to a real-time scenario, since the performance of KPC and KPLS is very sensitive to the choice of the adjustable Kernel parameter. Finally, our findings suggest that forecasters should not focus solely on RMSE when comparing models in terms of forecasting precision. In-

stead, a proper choice of the loss function should be context-dependent. In our case, I find that the symmetric loss functions under consideration lead to roughly similar conclusions, whereas asymmetric functions provide greater support towards non-linear models.

To conclude, it is difficult to choose a winning model class. Empirical evidence suggests that different models yield the best forecasting performance in different time periods and for different horizons. Although I find that KPC with a polynomial Kernel function is dominant for the horizon $h = 6$, further research needs to be carried to realize whether this finding is due to sample-specific characteristics or if there are genuinely non-linear links present in the data generating process. For now, it is safe to say that the non-linear class of models, as defined here, does not appear to provide sufficiently better results to justify their computationally demanding estimation procedure, and more importantly, their inherently unstable performance associated to hyperparameter estimation.

Notwithstanding, this thesis is prone to certain limitations. First, and more importantly, many unspecified parameters are chosen based on the final predictive performance. As already mentioned, this is, of course, not possible in a real-time forecasting exercise. Not only this, but our methodology also fixes such parameters to be constant over time, which might hinder the forecasting ability of the models. As a suggestion for future research to address these issues, I recommend the implementation of a cross-validation technique. Next, we should acknowledge that due to the large dataset used in this study, many predictor variables are only assigned a small weight when constructing latent factors. This might as well be a source of bias for our results. A possible solution would be the implementation of regularization techniques. This would be particularly interesting for Kernel methods, as to the best of our knowledge, the two approaches have not yet been combined in the literature. Next, based on the finding that different models are the best-performing in different samples, an interesting direction of future research arises: investigating whether regime-switching models that nest the different models considered in this paper consistently improve the forecasting accuracy. Finally, we should acknowledge the growth in popularity of machine learning techniques. In particular, autoencoders have been recently cited in the literature as useful dimensionality reduction tools that can capture arbitrary non-linear relationships. As a suggestion for future research, I put forward the investigation of these artificial neural networks as alternatives to the models that this paper considers.

References

Aronszajn, N. (1950). Theory of reproducing kernels. *Transactions of the American Mathematical Society*, *68*(3), 337–404.

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

Bekiros, S., Gupta, R., & Kyei, C. (2016). On economic uncertainty, stock market predictability and nonlinear spillover effects. *The North American Journal of Economics and Finance*, *36*, 184–191.

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

Boivin, J., & Ng, S. (2006). Are more data always better for factor analysis? *Journal of Econometrics*, *132*(1), 169–194.

Bok, B., Caratelli, D., Giannone, D., Sbordone, A. M., & Tambalotti, A. (2018). Macroeconomic nowcasting and forecasting with big data. *Annual Review of Economics*, *10*, 615–643.

Diebold, F. X. (2003). Big data dynamic factor models for macroeconomic measurement and forecasting. *Advances in Economics and Econometrics: Theory and Applications*, 115–122.

Diebold, F. X., & Mariano, R. S. (2002). Comparing predictive accuracy. *Journal of Business & Economic Statistics*, *20*(1), 134–144.

Eggoh, J. C., & Khan, M. (2014). On the nonlinear relationship between inflation and economic growth. *Research in Economics*, *68*(2), 133–143.

Eickmeier, S., & Ziegler, C. (2008). How successful are dynamic factor models at forecasting output and inflation? a meta-analytic approach. *Journal of Forecasting*, *27*(3), 237–265.

Elliott, G., & Timmermann, A. (2004). Optimal forecast combinations under general loss functions and forecast error distributions. *Journal of Econometrics*, *122*(1), 47–79.

Exterkate, P. (2013). Model selection in kernel ridge regression. *Computational Statistics & Data Analysis*, *68*, 1–16.

Exterkate, P., Groenen, P. J., Heij, C., & van Dijk, D. (2016). Nonlinear forecasting with many predictors using kernel ridge regression. *International Journal of Forecasting*, *32*(3), 736–753.

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

Giovannelli, A. (2012). Nonlinear forecasting using large datasets: Evidences on us and euro area economies. *CEIS Working Paper*.

Groen, J. J., & Kapetanios, G. (2009). Revisiting useful approaches to data-rich macroeconomic forecasting. *Federal Reserve Bank of New York Staff Report*, *327*.

Hung, F.-S. (2009). Explaining the nonlinear effects of financial development on economic growth. *Journal of Economics*, *97*(1), 41–65.

Kock, A. B., Teräsvirta, T., et al. (2011). Forecasting with nonlinear time series models. *Oxford Handbook of Economic Forecasting*, 61–87.

LeBaron, B. (1994). Chaos and nonlinear forecastability in economics and finance. *Philosophical Transactions of the Royal Society of London. Series A: Physical and Engineering Sciences*, *348*(1688), 397–404.

Lee, J.-M., Yoo, C., Choi, S. W., Vanrolleghem, P. A., & Lee, I.-B. (2004). Nonlinear process monitoring using kernel principal component analysis. *Chemical Engineering Science*, *59*(1), 223–234.

Mercer, J. (1909). *Functions of positive and negative type and their connection with the theory of integral equations, philosophical transaction of the royal society of london, ser.* A.

Pearson, K. (1901). Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, *2*(11), 559–572.

Rosipal, R., & Clancy, D. (2003). Kernel partial least squares for nonlinear regression and discrimination. *Neural Network World*, *13*(3), 291–300.

Rosipal, R., & Trejo, L. J. (2001). Kernel partial least squares regression in reproducing kernel hilbert space. *Journal of Machine Learning Research*, *2*(Dec), 97–123.

Sarel, M. (1996). Nonlinear effects of inflation on economic growth. *IMF Staff Papers*, *43*(1), 199–215.

Schölkopf, B., Smola, A., & Müller, K.-R. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, *10*(5), 1299–1319.

Stock, J. H., & Watson, M. W. (1998). *A comparison of linear and nonlinear univariate models for forecasting macroeconomic time series* (Tech. Rep.). National Bureau of Economic Research.

Stock, J. H., & Watson, M. W. (1999). Forecasting inflation. *Journal of Monetary Economics*, *44*(2), 293–335.

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* (Tech. Rep.). National Bureau of Economic Research.

Stock, J. H., & Watson, M. W. (2006). Forecasting with many predictors. *Handbook of Economic Forecasting*, *1*, 515–554.

Vanhatalo, E., Kulahci, M., & Bergquist, B. (2017). On the structure of dynamic principal component analysis used in statistical process monitoring. *Chemometrics and Intelligent Laboratory Systems*, *167*, 1–11.

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

Appendices

A. Robustness of PLS(b)

Table 9: RMSEs for PLS (b) for every number L of lags of the target used to enlarge the predictor matrix

	L = 1	L = 2	L = 3	L = 4	L = 5	L = 6
			Subsample M1			
h = 1	1.391	1.325	1.333	1.345	1.339	**1.305**
h = 6	1.168	1.144	1.146	1.130	1.161	**1.121**
h = 12	**0.994**	1.053	1.042	1.018	1.019	1.019
h = 24	0.834	**0.824**	0.869	0.872	0.873	0.847
			Subsample M2			
h = 1	1.041	1.030	1.025	**1.024**	1.034	1.024
h = 6	1.042	1.040	**0.990**	1.020	1.026	1.030
h = 12	0.880	0.876	0.878	**0.875**	0.880	0.885
h = 24	**0.695**	0.703	0.697	0.696	0.699	0.699
			Subsample M3			
h = 1	**1.121**	1.148	1.135	1.135	1.158	1.138
h = 6	1.231	1.230	**1.173**	1.218	1.220	1.231
h = 12	1.185	**1.179**	1.182	1.183	1.183	1.187
h = 24	1.226	1.214	1.211	**1.206**	1.212	1.208
			Subsample M4			
h = 1	1.232	1.192	1.200	1.200	1.201	**1.178**
h = 6	1.164	1.151	**1.135**	1.138	1.156	1.136
h = 12	0.959	0.965	0.959	**0.946**	0.949	0.951
h = 24	0.780	**0.779**	0.785	0.783	0.786	0.789
			Subsample M5			
h = 1	1.211	1.200	1.186	1.208	1.197	**1.167**
h = 6	1.152	1.141	**1.121**	1.127	1.143	1.127
h = 12	0.967	0.972	0.967	**0.955**	0.957	0.960
h = 24	0.814	**0.812**	0.818	0.816	0.819	0.821
			Subsample M6			
h = 1	**1.072**	1.084	1.081	1.083	1.074	1.072
h = 6	1.046	1.044	**0.994**	1.024	1.029	1.034
h = 12	0.907	0.903	0.905	**0.902**	0.906	0.911
h = 24	0.768	0.773	0.767	**0.766**	0.769	0.769
			Subsample M7			
h = 1	1.202	1.198	1.180	1.205	1.190	**1.163**
h = 6	1.169	1.160	**1.142**	1.142	1.157	1.148
h = 12	0.993	0.999	0.993	**0.980**	0.984	0.986
h = 24	0.839	**0.838**	0.842	0.840	0.843	0.846

Note: Best performance for each subsample-forecast horizon pair is highlighted in bold.

B. Robustness of Kernel Methods

Table 10: RMSEs of KPC2 for every value of the tunable parameter, σ, in the search space

	$\sigma = 0.5\sigma_0$	$\sigma = \sigma_0$	$\sigma = 2\sigma_0$	$\sigma = 4\sigma_0$	$\sigma = 8\sigma_0$
			Subsample M1		
h = 1	0.975	0.967	0.965	0.964	**0.964**
h = 6	0.701	0.673	0.667	0.665	**0.665**
h = 12	0.687	0.670	0.668	0.667	**0.667**
h = 24	0.617	0.605	0.603	0.602	**0.602**
			Subsample M2		
h = 1	0.954	**0.939**	0.959	0.959	0.959
h = 6	0.640	0.630	0.628	0.627	**0.627**
h = 12	**0.629**	0.636	0.639	0.640	0.640
h = 24	**0.616**	0.632	0.636	0.637	0.638
			Subsample M3		
h = 1	0.939	0.939	0.931	0.927	**0.926**
h = 6	0.669	0.660	0.659	0.659	**0.658**
h = 12	0.742	0.726	0.722	0.721	**0.721**
h = 24	0.674	0.645	0.638	0.636	**0.636**
			Subsample M4		
h = 1	0.974	**0.963**	0.974	0.974	0.974
h = 6	0.694	0.681	0.678	0.677	**0.677**
h = 12	0.646	0.642	0.642	**0.642**	0.642
h = 24	**0.627**	0.630	0.630	0.631	0.631
			Subsample M5		
h = 1	0.968	**0.959**	0.966	0.965	0.965
h = 6	0.684	0.671	0.668	0.668	**0.667**
h = 12	0.651	0.646	0.645	0.645	**0.645**
h = 24	**0.632**	0.632	0.632	0.632	0.632
			Subsample M6		
h = 1	0.949	**0.937**	0.950	0.949	0.949
h = 6	0.634	0.624	0.622	0.621	**0.621**
h = 12	**0.640**	0.644	0.645	0.645	0.645
h = 24	**0.625**	0.636	0.637	0.638	0.639
			Subsample M7		
h = 1	0.958	**0.949**	0.958	0.957	0.957
h = 6	0.682	0.669	0.667	0.666	**0.666**
h = 12	0.656	0.651	0.650	0.650	**0.650**
h = 24	0.639	0.639	**0.639**	0.639	0.639

Notes:

1. This table displays the RMSE of KPC2 (i.e., using a Gaussian radial basis kernel function) for every value of the Kernel tunable coefficient, σ, that the performance is evaluated on. In this case, $\sigma \in \{0.5\sigma_0, \sigma_0, 2\sigma_0, 4\sigma_0, 8\sigma_0\}$, where $\sigma_0 = \sqrt{c_N}/\pi$, in which c_N represents the 95-th percentile of the χ^2 distribution with N (i.e., number of potential predictors) degrees of freedom.

2. Best performance for each subsample-forecast horizon pair is highlighted in bold.

Table 11: RMSEs of KPLS1 for every value of the tunable parameter, σ, in the search space

	$\sigma = 0.5\sigma_0$	$\sigma = \sigma_0$	$\sigma = 2\sigma_0$	$\sigma = 4\sigma_0$	$\sigma = 8\sigma_0$
			Subsample M1		
h = 1	2.398	1.030	0.925	**0.919**	0.924
h = 6	1.426	1.096	0.865	0.794	**0.789**
h = 12	1.095	1.080	1.069	1.067	**1.066**
h = 24	1.513	0.871	0.698	0.678	**0.674**
			Subsample M2		
h = 1	0.990	**0.922**	0.961	0.992	0.998
h = 6	1.147	1.030	0.912	0.865	**0.854**
h = 12	**0.900**	0.989	0.957	0.934	0.936
h = 24	0.716	0.596	**0.552**	0.557	0.554
			Subsample M3		
h = 1	0.923	**0.883**	0.885	0.901	0.902
h = 6	0.772	**0.751**	0.787	0.802	0.806
h = 12	0.727	**0.718**	0.732	0.727	0.729
h = 24	1.000	0.990	0.979	**0.974**	0.979
			Subsample M4		
h = 1	1.748	1.018	**0.981**	0.990	0.994
h = 6	1.146	0.990	0.882	0.841	**0.837**
h = 12	0.992	1.026	1.002	**0.989**	0.989
h = 24	1.120	0.734	0.627	0.621	**0.618**
			Subsample M5		
h = 1	1.612	0.997	**0.967**	0.976	0.979
h = 6	1.105	0.962	0.868	0.832	**0.829**
h = 12	0.969	0.997	0.977	**0.964**	0.965
h = 24	1.112	0.755	0.655	0.649	**0.646**
			Subsample M6		
h = 1	0.968	**0.912**	0.943	0.970	0.973
h = 6	1.084	0.980	0.884	0.847	**0.838**
h = 12	**0.874**	0.945	0.921	0.900	0.902
h = 24	0.758	0.653	**0.612**	0.616	0.614
			Subsample M7		
h = 1	1.656	1.057	0.976	**0.974**	0.979
h = 6	1.075	0.965	0.872	0.835	**0.831**
h = 12	0.986	0.996	0.968	**0.955**	0.956
h = 24	1.097	0.749	0.651	0.645	**0.642**

Notes:

1. This table displays the RMSE of KPLS1 (i.e., using a polynomial kernel function) for every value of the Kernel tunable coefficient, σ, that the performance is evaluated on. In this case, $\sigma \in \{0.5\sigma_0, \sigma_0, 2\sigma_0, 4\sigma_0, 8\sigma_0\}$, where $\sigma_0 = \sqrt{(N+2)/2}$, in which N represents the total number of potential predictors.

2. Best performance for each subsample-forecast horizon pair is highlighted in bold.

Table 12: RMSEs of KPLS2 for every value of the tunable parameter, σ, in the search space

	$\sigma = 0.5\sigma_0$	$\sigma = \sigma_0$	$\sigma = 2\sigma_0$	$\sigma = 4\sigma_0$	$\sigma = 8\sigma_0$
			Subsample M1		
h = 1	0.966	0.944	0.922	0.922	**0.922**
h = 6	0.812	**0.783**	0.787	0.785	0.784
h = 12	**1.062**	1.064	1.063	1.063	1.063
h = 24	0.689	0.683	**0.674**	0.674	0.674
			Subsample M2		
h = 1	**0.993**	0.998	1.000	1.000	1.000
h = 6	0.851	**0.850**	0.850	0.850	0.850
h = 12	0.943	0.942	0.936	0.936	**0.936**
h = 24	0.566	0.554	**0.551**	0.551	0.551
			Subsample M3		
h = 1	0.965	0.926	0.926	0.921	**0.921**
h = 6	0.815	0.809	0.807	0.807	**0.807**
h = 12	0.751	0.735	0.731	0.730	**0.730**
h = 24	1.025	0.988	0.978	0.978	**0.977**
			Subsample M4		
h = 1	0.999	1.002	**0.992**	0.993	0.993
h = 6	0.845	**0.833**	0.835	0.834	0.834
h = 12	0.992	0.992	0.988	0.988	**0.988**
h = 24	0.630	0.622	**0.616**	0.616	0.616
			Subsample M5		
h = 1	0.996	0.991	0.983	**0.982**	0.982
h = 6	0.836	**0.825**	0.827	0.827	0.826
h = 12	0.970	0.968	0.964	0.964	**0.964**
h = 24	0.661	0.651	**0.644**	0.645	0.645
			Subsample M6		
h = 1	0.990	0.982	0.982	0.980	**0.980**
h = 6	0.837	0.836	0.835	0.835	**0.835**
h = 12	0.912	0.908	0.903	0.902	**0.902**
h = 24	0.630	0.615	**0.611**	0.611	0.611
			Subsample M7		
h = 1	0.983	0.985	**0.979**	0.980	0.980
h = 6	0.834	**0.826**	0.829	0.828	0.828
h = 12	0.962	0.959	0.955	0.955	**0.955**
h = 24	0.657	0.646	**0.640**	0.640	0.640

Notes:

1. This table displays the RMSE of KPLS2 (i.e., using a Gaussian radial basis kernel function) for every value of the Kernel tunable coefficient, σ, that the performance is evaluated on. In this case, $\sigma \in \{0.5\sigma_0, \sigma_0, 2\sigma_0, 4\sigma_0, 8\sigma_0\}$, where $\sigma_0 = \sqrt{c_N}/\pi$, in which c_N represents the 95-th percentile of the χ^2 distribution with N (i.e., number of potential predictors) degrees of freedom.

2. Best performance for each subsample-forecast horizon pair is highlighted in bold.

C. Forecasting performance based on alternative loss functions

Table 13: Best performing model for each subsample and forecast horizon according to the piece-wise linear and asymmetric squared loss functions

	Piece-wise Linear ($\delta = 0.1$)	Piece-wise Linear ($\delta = 0.9$)	Asymmetric Squared ($\delta = 0.1$)	Asymmetric Squared ($\delta = 0.9$)
		Subsample M1		
h = 1	KPC1	SQPLS (c)	KPLS1	PLS (c)
h = 6	KPC1	PC	KPC1	SQPLS (c)
h = 12	PLS (c)	KPC2	PC	KPC1
h = 24	PLS (c)	KPLS2	PLS (c)	KPLS2
		Subsample M2		
h = 1	PLS (c)	KPLS1	PLS (c)	SQPLS (c)
h = 6	SQPC	PLS (c)	KPC1	PLS (c)
h = 12	SQPC	PLS (c)	SQPC	PLS (c)
h = 24	SQPC	KPLS1	SQPC	KPLS1
		Subsample M3		
h = 1	PLS (c)	KPLS1	PLS (c)	KPC1
h = 6	PLS (c)	KPC1	PLS (c)	KPC1
h = 12	PC	KPC1	PC	KPC1
h = 24	PC	KPLS1	PC	SQPLS (c)
		Subsample M4		
h = 1	KPC2	SQPLS (c)	KPLS1	PLS (c)
h = 6	KPC1	SQPLS (c)	KPC1	SQPLS (c)
h = 12	SQPC	PLS (c)	SQPC	PLS (c)
h = 24	SQPC	KPLS2	PC	KPLS2
		Subsample M5		
h = 1	PLS (c)	SQPLS (c)	PLS (c)	PLS (c)
h = 6	PC	KPC1	PC	SQPLS (c)
h = 12	SQPC	PLS (c)	SQPC	PLS (c)
h = 24	SQPC	KPLS2	PC	KPLS2
		Subsample M6		
h = 1	PLS (c)	KPLS1	PLS (c)	KPLS1
h = 6	PC	KPC1	KPC1	PLS (c)
h = 12	SQPC	PLS (c)	SQPC	PLS (c)
h = 24	SQPC	KPLS1	SQPC	PLS (c)
		Subsample M7		
h = 1	PLS (c)	SQPLS (c)	PLS (c)	PLS (c)
h = 6	PC	SQPLS (c)	KPC1	SQPLS (c)
h = 12	PC	PLS (c)	PC	PLS (c)
h = 24	PC	KPLS1	PC	KPLS1

Table 14: CF2 for all models relative to benchmark, $h = 1$ and $h = 6$

	Panel A: Linear Methods							
	h = 1				**h = 6**			
Period	PC (k=10)	PLS (a)	PLS (b)	PLS (c)	PC (k=10)	PLS (a)	PLS (b)	PLS (c)
M1	1.052	1.020	1.089	0.992	0.883	0.920	1.105	0.903
M2	0.951	0.976	1.028	**0.935**	0.850	0.840	1.023	0.845
M3	0.973	0.997	1.080	0.955	0.928	0.849	1.211	0.834
M4	1.017	1.014	1.065	0.980	0.882	0.898	1.105	0.892
M5	1.005	1.007	1.073	**0.972**	0.884	0.884	1.127	0.883
M6	0.956	0.980	1.051	**0.940**	0.863	0.835	1.071	0.835
M7	0.997	1.000	1.077	**0.965**	0.876	0.872	1.135	0.872
	Panel B: Squared Methods							
	h = 1				**h = 6**			
Period	SQPC (k=10)	SQPLS (a)	SQPLS (b)	SQPLS (c)	SQPC (k=10)	SQPLS (a)	SQPLS (b)	SQPLS (c)
M1	1.103	1.069	1.115	1.021	0.971	0.911	1.140	0.890
M2	0.980	0.961	1.037	0.942	0.820	0.825	1.169	0.830
M3	**0.925**	0.968	1.022	0.946	0.854	0.851	1.271	0.856
M4	1.048	1.029	1.074	0.993	0.913	0.886	1.185	0.878
M5	1.016	1.012	1.080	0.980	0.898	0.875	1.221	0.871
M6	0.956	0.960	1.043	0.942	**0.825**	0.826	1.210	0.832
M7	1.003	1.005	1.095	0.969	0.889	0.867	1.220	0.862
	Panel C: Kernel Methods							
	h = 1				**h = 6**			
Period	KPC1 (k=10)	KPC2 (k=10)	KPLS1 (k=1)	KPLS2 (k=1)	KPC1 (k=10)	KPC2 (k=10)	KPLS1 (k=1)	KPLS2 (k=1)
M1	0.986	0.989	0.967	**0.967**	**0.861**	0.884	0.939	0.938
M2	0.950	0.946	0.967	0.970	**0.819**	0.839	0.945	0.943
M3	0.949	0.956	0.939	0.949	**0.797**	0.895	0.853	0.973
M4	0.980	**0.978**	0.988	0.982	**0.862**	0.879	0.955	0.953
M5	0.973	0.974	0.978	0.980	**0.862**	0.879	0.957	0.956
M6	0.946	0.950	0.957	0.972	0.833	0.852	0.952	0.951
M7	0.965	0.966	0.983	0.984	**0.852**	0.867	0.948	0.947

Note: Best performance for each subsample-forecast horizon pair is highlighted in bold.

Table 15: CF2 for all models relative to benchmark, $h = 12$ and $h = 24$

	Panel A: Linear Methods							
	h = 12				**h = 24**			
Period	PC (k=10)	PLS (a)	PLS (b)	PLS (c)	PC (k=10)	PLS (a)	PLS (b)	PLS (c)
M1	0.843	0.847	0.945	0.831	0.831	0.763	0.882	**0.747**
M2	0.829	0.817	0.984	**0.797**	0.793	0.778	0.879	**0.756**
M3	0.979	0.831	1.178	**0.824**	0.872	0.857	1.182	0.853
M4	0.830	0.845	0.966	0.837	0.827	0.778	0.890	**0.759**
M5	0.857	0.849	0.998	**0.839**	0.838	0.790	0.938	**0.771**
M6	0.875	0.825	1.034	**0.812**	0.820	0.798	0.965	**0.781**
M7	0.851	0.852	1.017	**0.844**	0.839	0.791	0.956	**0.776**
	Panel B: Squared Methods							
	h = 12				**h = 24**			
Period	SQPC (k=10)	SQPLS (a)	SQPLS (b)	SQPLS (c)	SQPC (k=10)	SQPLS (a)	SQPLS (b)	SQPLS (c)
M1	0.921	0.851	1.002	0.847	0.966	0.795	0.933	0.809
M2	0.798	0.802	1.075	0.806	0.815	0.789	0.978	0.785
M3	0.898	0.899	1.245	0.898	0.786	**0.778**	1.177	0.815
M4	0.858	**0.826**	1.062	0.836	0.914	0.801	1.001	0.813
M5	0.867	0.842	1.092	0.850	0.897	0.803	1.027	0.812
M6	0.832	0.836	1.126	0.839	0.812	0.796	1.029	0.802
M7	0.871	0.844	1.113	0.854	0.908	0.809	1.040	0.818
	Panel C: Kernel Methods							
	h = 12				**h = 24**			
Period	KPC1 (k=10)	KPC2 (k=10)	KPLS1 (k=1)	KPLS2 (k=1)	KPC1 (k=10)	KPC2 (k=10)	KPLS1 (k=1)	KPLS2 (k=1)
M1	**0.830**	0.834	1.035	1.032	0.777	0.778	0.828	0.829
M2	0.862	0.864	0.966	0.983	0.860	0.855	0.812	0.821
M3	0.878	0.964	0.871	0.918	0.872	0.905	1.102	1.154
M4	0.843	0.845	1.003	1.004	0.831	0.829	0.833	0.833
M5	0.864	0.867	0.985	0.994	0.845	0.844	0.887	0.887
M6	0.894	0.899	0.944	0.973	0.876	0.873	0.910	0.916
M7	0.860	0.862	0.985	0.986	0.850	0.849	0.877	0.877

Note: Best performance for each subsample-forecast horizon pair is highlighted in bold.

Table 16: CF3 for all models relative to benchmark, $h = 1$ and $h = 6$

	Panel A: Linear Methods							
	h = 1				**h = 6**			
Period	PC (k=10)	PLS (a)	PLS (b)	PLS (c)	PC (k=10)	PLS (a)	PLS (b)	PLS (c)
M1	0.918	1.164	1.551	**0.826**	0.533	0.488	1.145	0.453
M2	1.015	1.028	0.961	0.939	0.590	0.457	0.957	0.449
M3	0.878	0.888	1.099	0.879	0.455	0.398	0.967	0.395
M4	0.971	1.104	1.348	**0.909**	0.558	0.507	1.150	0.484
M5	0.959	1.081	1.323	**0.905**	0.540	0.490	1.110	0.469
M6	0.987	1.001	1.034	0.928	0.561	0.438	0.921	0.431
M7	0.943	1.063	1.305	**0.894**	0.545	0.496	1.137	0.478
	Panel B: Squared Methods							
	h = 1				**h = 6**			
Period	SQPC (k=10)	SQPLS (a)	SQPLS (b)	SQPLS (c)	SQPC (k=10)	SQPLS (a)	SQPLS (b)	SQPLS (c)
M1	1.095	1.537	1.990	0.983	0.723	0.471	1.394	0.444
M2	1.056	1.003	0.993	0.929	0.585	0.448	1.505	0.444
M3	**0.806**	0.888	1.186	0.840	0.380	0.386	1.122	0.393
M4	1.088	1.308	1.574	1.014	0.638	0.496	1.696	0.478
M5	1.059	1.264	1.563	0.998	0.612	0.481	1.627	0.465
M6	1.012	0.979	1.074	0.920	0.553	0.432	1.427	0.429
M7	1.030	1.229	1.554	0.978	0.612	0.493	1.678	0.478
	Panel C: Kernel Methods							
	h = 1				**h = 6**			
Period	KPC1 (k=10)	KPC2 (k=10)	KPLS1 (k=1)	KPLS2 (k=1)	KPC1 (k=10)	KPC2 (k=10)	KPLS1 (k=1)	KPLS2 (k=1)
M1	0.912	0.917	0.855	0.857	**0.428**	0.457	0.660	0.642
M2	**0.867**	0.984	0.872	1.025	**0.437**	0.448	0.760	0.741
M3	0.853	0.879	0.810	0.893	**0.380**	0.400	0.569	0.568
M4	0.976	0.961	0.993	1.006	**0.468**	0.486	0.715	0.705
M5	0.964	0.955	0.977	0.995	**0.456**	0.473	0.698	0.689
M6	0.865	0.970	**0.864**	1.003	**0.423**	0.433	0.730	0.712
M7	0.952	0.943	0.983	0.988	**0.464**	0.480	0.708	0.697

Note: Best performance for each subsample-forecast horizon pair is highlighted in bold.

Table 17: CF3 for all models relative to benchmark, $h = 12$ and $h = 24$

	Panel A: Linear Methods							
	h = 12				**h = 24**			
Period	PC (k=10)	PLS (a)	PLS (b)	PLS (c)	PC (k=10)	PLS (a)	PLS (b)	PLS (c)
M1	0.594	0.551	1.120	0.549	0.910	0.420	0.780	**0.417**
M2	0.460	0.387	0.788	0.404	0.337	0.312	0.518	0.298
M3	0.519	0.383	1.048	**0.360**	0.378	0.287	1.099	**0.271**
M4	0.516	0.451	0.949	0.460	0.611	0.373	0.676	**0.360**
M5	0.507	0.439	0.933	0.448	0.601	0.372	0.695	**0.359**
M6	0.451	0.375	0.780	0.391	0.341	0.310	0.563	0.297
M7	0.513	0.444	0.960	0.453	0.602	0.375	0.718	**0.361**
	Panel B: Squared Methods							
	h = 12				**h = 24**			
Period	SQPC (k=10)	SQPLS (a)	SQPLS (b)	SQPLS (c)	SQPC (k=10)	SQPLS (a)	SQPLS (b)	SQPLS (c)
M1	0.773	0.574	1.409	0.576	1.671	0.486	1.000	0.497
M2	0.425	0.350	1.163	**0.349**	0.493	0.294	0.686	**0.277**
M3	0.364	0.398	1.194	0.406	0.274	0.286	1.301	0.304
M4	0.579	**0.444**	1.399	0.450	1.063	0.382	1.022	0.391
M5	0.558	**0.433**	1.366	0.440	1.029	0.379	1.037	0.389
M6	0.408	**0.344**	1.131	0.345	0.477	0.296	0.730	**0.284**
M7	0.562	**0.443**	1.421	0.451	1.041	0.385	1.076	0.394
	Panel C: Kernel Methods							
	h = 12				**h = 24**			
Period	KPC1 (k=10)	KPC2 (k=10)	KPLS1 (k=1)	KPLS2 (k=1)	KPC1 (k=10)	KPC2 (k=10)	KPLS1 (k=1)	KPLS2 (k=1)
M1	**0.513**	0.515	1.119	1.114	0.464	0.463	0.535	0.535
M2	0.444	0.439	0.869	0.904	0.446	0.409	0.330	0.328
M3	0.423	0.453	0.505	0.524	0.347	0.360	0.631	0.630
M4	0.465	0.470	1.004	0.989	0.461	0.448	0.427	0.425
M5	0.456	0.461	0.968	0.955	0.456	0.446	0.436	0.433
M6	0.432	0.429	0.833	0.856	0.440	0.408	0.354	0.351
M7	0.469	0.473	0.962	0.953	0.463	0.454	0.435	0.432

Note: Best performance for each subsample-forecast horizon pair is highlighted in bold.

Table 18: CF4 for all models relative to benchmark, $h = 1$ and $h = 6$.

	Panel A: Linear Methods							
	h = 1				**h = 6**			
Period	PC (k=10)	PLS (a)	PLS (b)	PLS (c)	PC (k=10)	PLS (a)	PLS (b)	PLS (c)
M1	0.751	1.035	1.550	**0.720**	0.865	0.815	1.847	**0.762**
M2	1.279	1.092	1.009	1.007	1.414	0.883	1.479	0.863
M3	1.007	1.018	1.367	1.020	1.227	0.896	2.337	0.830
M4	0.969	1.065	1.335	0.963	1.139	0.846	1.773	**0.838**
M5	0.959	1.050	1.327	0.957	1.137	0.835	1.769	**0.827**
M6	1.189	1.055	1.069	1.065	1.374	0.859	1.516	**0.840**
M7	0.963	1.061	1.322	0.966	1.121	0.828	1.670	**0.821**
	Panel B: Squared Methods							
	h = 1				**h = 6**			
Period	SQPC (k=10)	SQPLS (a)	SQPLS (b)	SQPLS (c)	SQPC (k=10)	SQPLS (a)	SQPLS (b)	SQPLS (c)
M1	0.859	1.487	1.978	0.767	1.316	0.799	2.696	0.777
M2	1.313	1.067	1.156	**0.974**	1.302	0.944	2.669	0.919
M3	0.905	1.029	1.322	0.931	0.849	0.803	2.421	0.799
M4	1.047	1.341	1.628	**0.924**	1.334	0.870	2.855	0.849
M5	1.023	1.298	1.598	**0.925**	1.293	0.856	2.792	0.836
M6	1.201	1.048	1.199	0.981	1.231	0.909	2.579	0.886
M7	1.028	1.290	1.643	**0.933**	1.245	0.856	2.720	0.838
	Panel C: Kernel Methods							
	h = 1				**h = 6**			
Period	KPC1 (k=10)	KPC2 (k=10)	KPLS1 (k=1)	KPLS2 (k=1)	KPC1 (k=10)	KPC2 (k=10)	KPLS1 (k=1)	KPLS2 (k=1)
M1	0.895	0.896	0.958	0.953	0.792	0.808	0.868	0.869
M2	1.185	1.230	1.011	1.416	1.006	1.051	**0.849**	1.400
M3	**0.768**	0.955	0.776	0.937	**0.583**	0.931	0.740	1.104
M4	1.014	1.036	1.184	1.183	0.899	0.927	1.141	1.146
M5	0.987	1.020	1.151	1.151	0.891	0.920	1.121	1.140
M6	1.090	1.156	**0.945**	1.309	0.977	1.022	0.841	1.357
M7	1.008	1.026	1.163	1.160	0.903	0.930	1.138	1.170

Note: Best performance for each subsample-forecast horizon pair is highlighted in bold.

Table 19: CF4 for all models relative to benchmark, $h = 12$ and $h = 24$

	Panel A: Linear Methods							
	h = 12				**h = 24**			
Period	PC (k=10)	PLS (a)	PLS (b)	PLS (c)	PC (k=10)	PLS (a)	PLS (b)	PLS (c)
M1	0.925	0.771	1.167	0.763	1.511	0.598	0.895	0.587
M2	1.011	0.622	1.160	**0.566**	1.060	0.599	1.274	0.515
M3	1.333	0.860	2.355	0.801	1.334	0.880	2.827	0.842
M4	0.962	0.741	1.359	**0.732**	1.470	0.625	1.270	0.590
M5	0.982	0.747	1.393	**0.737**	1.469	0.638	1.323	0.603
M6	1.049	0.640	1.257	**0.588**	1.096	0.630	1.439	0.551
M7	0.985	0.758	1.362	**0.746**	1.450	0.639	1.317	0.598
	Panel B: Squared Methods							
	h = 12				**h = 24**			
Period	SQPC (k=10)	SQPLS (a)	SQPLS (b)	SQPLS (c)	SQPC (k=10)	SQPLS (a)	SQPLS (b)	SQPLS (c)
M1	1.249	0.806	1.630	0.800	2.811	0.683	1.225	0.689
M2	1.001	0.724	2.196	0.695	1.953	0.718	2.096	0.634
M3	0.874	0.872	2.491	0.838	0.824	0.808	3.178	**0.791**
M4	1.155	0.762	2.302	0.764	2.758	0.713	2.050	0.716
M5	1.141	0.764	2.302	0.766	2.699	0.719	2.147	0.722
M6	0.986	0.732	2.199	0.705	1.846	0.736	2.242	0.649
M7	1.123	0.780	2.292	0.788	2.641	0.727	2.252	0.724
	Panel C: Kernel Methods							
	h = 12				**h = 24**			
Period	KPC1 (k=10)	KPC2 (k=10)	KPLS1 (k=1)	KPLS2 (k=1)	KPC1 (k=10)	KPC2 (k=10)	KPLS1 (k=1)	KPLS2 (k=1)
M1	**0.721**	0.731	1.093	1.088	0.600	0.599	**0.518**	0.518
M2	0.874	0.914	0.849	0.998	0.741	0.740	**0.478**	0.484
M3	**0.656**	0.990	0.679	0.840	0.792	1.009	0.991	1.137
M4	0.765	0.793	0.994	1.030	0.671	0.671	**0.516**	0.516
M5	0.769	0.799	0.982	1.022	0.683	0.684	**0.536**	0.537
M6	0.871	0.919	0.839	0.985	0.771	0.772	**0.544**	0.553
M7	0.801	0.828	1.040	1.043	0.699	0.699	**0.533**	0.534

Note: Best performance for each subsample-forecast horizon pair is highlighted in bold.

Table 20: CF5 for all models relative to benchmark, $h = 1$ and $h = 6$

	Panel A: Linear Methods							
	h = 1				**h = 6**			
Period	PC (k=10)	PLS (a)	PLS (b)	PLS (c)	PC (k=10)	PLS (a)	PLS (b)	PLS (c)
M1	1.146	1.202	1.276	0.964	0.442	0.408	0.811	0.382
M2	0.819	0.974	0.684	0.700	0.390	0.374	0.822	0.343
M3	0.749	0.749	0.788	**0.689**	0.284	0.325	0.511	**0.275**
M4	0.976	1.067	1.054	0.844	0.378	0.423	0.897	0.390
M5	0.963	1.041	1.068	0.844	0.366	0.411	0.861	0.381
M6	0.823	0.952	0.775	0.720	0.372	0.365	0.776	0.337
M7	0.925	1.003	1.061	0.818	0.370	0.417	0.930	0.384
	Panel B: Squared Methods							
	h = 1				**h = 6**			
Period	SQPC (k=10)	SQPLS (a)	SQPLS (b)	SQPLS (c)	SQPC (k=10)	SQPLS (a)	SQPLS (b)	SQPLS (c)
M1	1.404	1.568	1.762	1.065	0.519	0.392	0.831	0.363
M2	0.865	0.929	0.650	0.730	0.412	0.346	1.041	0.320
M3	0.715	0.741	0.981	0.746	0.324	0.347	0.634	0.341
M4	1.136	1.244	1.283	0.908	0.411	0.398	1.076	0.369
M5	1.101	1.199	1.294	0.904	0.401	0.391	1.068	0.364
M6	0.855	0.912	0.756	0.749	0.397	0.343	1.034	0.320
M7	1.029	1.139	1.265	0.875	0.408	0.403	1.093	0.372
	Panel C: Kernel Methods							
	h = 1				**h = 6**			
Period	KPC1 (k=10)	KPC2 (k=10)	KPLS1 (k=1)	KPLS2 (k=1)	KPC1 (k=10)	KPC2 (k=10)	KPLS1 (k=1)	KPLS2 (k=1)
M1	0.929	0.949	**0.678**	0.738	**0.333**	0.367	0.609	0.585
M2	0.642	0.803	**0.606**	0.673	**0.315**	0.318	0.603	0.569
M3	0.804	0.807	0.783	0.848	0.326	0.325	0.473	0.453
M4	0.901	0.870	**0.741**	0.785	**0.351**	0.365	0.582	0.565
M5	0.899	0.873	**0.756**	0.800	**0.347**	0.359	0.571	0.553
M6	0.692	0.817	**0.645**	0.710	**0.314**	0.316	0.584	0.552
M7	0.873	0.852	**0.762**	0.793	**0.347**	0.358	0.564	0.546

Note: Best performance for each subsample-forecast horizon pair is highlighted in bold.

Table 21: CF5 for all models relative to benchmark, $h = 12$ and $h = 24$

	Panel A: Linear Methods							
	h = 12				**h = 24**			
Period	PC (k=10)	PLS (a)	PLS (b)	PLS (c)	PC (k=10)	PLS (a)	PLS (b)	PLS (c)
M1	**0.308**	0.386	0.726	0.390	0.211	0.231	0.449	**0.203**
M2	0.276	0.292	0.649	0.311	0.209	0.279	0.354	0.267
M3	**0.266**	0.285	0.460	0.268	**0.243**	0.270	0.649	0.260
M4	0.285	0.325	0.677	0.338	**0.208**	0.261	0.385	0.250
M5	0.281	0.320	0.653	0.333	**0.210**	0.261	0.400	0.250
M6	0.271	0.290	0.618	0.307	0.211	0.277	0.379	0.268
M7	0.284	0.319	0.709	0.333	**0.217**	0.267	0.435	0.258
	Panel B: Squared Methods							
	h = 12				**h = 24**			
Period	SQPC (k=10)	SQPLS (a)	SQPLS (b)	SQPLS (c)	SQPC (k=10)	SQPLS (a)	SQPLS (b)	SQPLS (c)
M1	0.323	0.393	0.857	0.395	0.276	0.272	0.508	0.318
M2	**0.232**	0.247	0.716	0.257	**0.197**	0.233	0.372	0.215
M3	0.276	0.318	0.571	0.321	0.253	0.269	0.759	0.289
M4	**0.263**	0.299	0.767	0.308	0.222	0.254	0.439	0.263
M5	**0.263**	0.300	0.745	0.307	0.224	0.255	0.458	0.264
M6	**0.236**	0.255	0.692	0.266	**0.202**	0.237	0.404	0.228
M7	**0.275**	0.303	0.823	0.311	0.260	0.260	0.493	0.268
	Panel C: Kernel Methods							
	h = 12				**h = 24**			
Period	KPC1 (k=10)	KPC2 (k=10)	KPLS1 (k=1)	KPLS2 (k=1)	KPC1 (k=10)	KPC2 (k=10)	KPLS1 (k=1)	KPLS2 (k=1)
M1	0.356	0.358	1.127	1.121	0.347	0.347	0.428	0.589
M2	0.306	0.292	0.887	0.866	0.405	0.356	0.327	0.324
M3	0.345	0.345	0.487	0.487	0.322	0.322	0.627	0.625
M4	0.322	0.320	0.985	0.958	0.386	0.355	0.416	0.412
M5	0.323	0.322	0.938	0.914	0.382	0.354	0.428	0.424
M6	0.309	0.298	0.845	0.813	0.399	0.355	0.354	0.349
M7	0.321	0.319	0.915	0.893	0.385	0.357	0.428	0.425

Note: Best performance for each subsample-forecast horizon pair is highlighted in bold.

D. MATLAB Code

Listing 1: Main file

```
%% Get dataset ready for analysis

clc;
clear;

load data_2003.mat

t = datevec(date);
year = t(:,1);
month = t(:,2);

transf = ["DLN";"DLN";"DLN";"DLN";"DLN";"DLN";"DLN";"DLN";"DLN";"DLN";"DLN";"DLN";"DLN";"DLN";"DLN";"DLN";"DLN";"DLN";"LV";"
      DLV";"DLV";"DLV";"DLN";"DLN";"DLV";"DLV";"DLN";"DLN";"DLN";"DLN";"DLN";"DLN";"DLN";"DLN";"DLN";"DLN";"DLN";"DLN";"DLN";"
      DLN";"DLN";"DLN";"DLN";"DLN";"DLN";"DLN";"LV";"DLV";"LV";"LV";"LN";"LN";"LN";"LN";"LN";"LN";"LN";"LN";"LN";"LN";"LV";"LV
      ";"LV";"LV";"DLN";"DLN";"DLN";"DLN";"DLN";"DLV";"D2LN";"D2LN";"D2LN";"DLN";"D2LN";"D2LN";"D2LN";"D2LN";"LV";"D2LN";"DLV
      ";"DLN";"DLN";"DLV";"DLN";"DLV";"DLV";"DLV";"DLV";"DLV";"DLV";"DLV";"DLV";"DLV";"DLV";"LV";"LV";"LV";"LV";"LV";"LV";"LV";"LV
      ";"DLN";"DLN";"DLN";"DLN";"DLN";"D2LN";"D2LN";"D2LN";"D2LN";"D2LN";"D2LN";"LV";"D2LN";"D2LN";"D2LN";"D2LN";"D2LN";"D2LN
      ";"D2LN";"D2LN";"D2LN";"D2LN";"D2LN";"D2LN";"D2LN";"D2LN";"D2LN";"D2LN";"D2LN";"DLV"];

% Transform variables according to the appendix of Fuentes et al. (2015):
count = 1;
for y = 1960:2003 % for yeach year in sample
    for m = 1:12 % for all months Jan-Dec
        ind = find(year == y & month == m); % index in data matrix corresponding to y and m

        for x = 1:size(data,2) % iterate over each column/variable to perform transformation

            if isequal(transf{x,1}, 'LV')
                final_data(count,x) = data(ind,x);
            elseif isequal(transf{x,1}, 'LN')
                final_data(count,x) = log(data(ind,x));
            elseif isequal(transf{x,1}, 'DLV')
                final_data(count,x) = (data(ind,x) - data(ind-1,x));
            elseif isequal(transf{x,1}, 'DLN')
                final_data(count,x) = log(data(ind,x)/data(ind-1,x));
            elseif isequal(transf{x,1}, 'D2LN')
                final_data(count,x) = (log(data(ind,x)/data(ind-1,x)) - log(data(ind-1,x)/data(ind-2,x)));
            end

        end

        count = count + 1;
    end
end

date2 = datevec(date);

ind1 = find(year==1960 & month==1);
ind2 = find(year==2003 & month==12);
date = date(ind1:ind2,:);

predictors = final_data;


year = date2(:,1);
month = date2(:,2);

```

```
52
53  periods = {'M1';'M2';'M3';'M4';'M5';'M6';'M7'};
54  horizons = [1;6;12;24];
55  nr_pc_factors = 10;
56
57
58  %% Final forecasts for all models and all forecast horizons (only best-performing)
59
60  tic
61  % Dimensions of 'results_all' correspond to PERFORMANCE MEASURE, METHOD, H,
62  % and SUBSAMPLE, respectively
63  results_all = cell(5,12,4,7);
64  forecast_series = cell(12,4,7);
65
66  asymmetric_squared = cell(12,4,7);
67  asymmetric_linear = cell(12,4,7);
68
69  for hor_index = 1:4
70      hor_index
71      for p = 1:7 % iterate over each period to forecast
72
73          h = horizons(hor_index,1); % current iteration forecast horizon
74          per = periods{p,1}; % current iteration estimation/forecast subsample
75
76          % Compute the target variable Y and Z:
77          x = 115; % column 115 in the data/predictor matrix corresponds to CPI
78          count = 1;
79          for y = 1960:2003
80              for m = 1:12
81                  ind = find(year == y & month == m);
82                  if ind-h-1 > 0
83                      target_y(count,1) = (1200/h)*log(data(ind,x)/data(ind-h,x)) - 1200*log(data(ind-h,x)/data(ind-h-1,x));
84                  else
85                      target_y(count,1) = NaN;
86                  end
87                  target_z(count,1) = 1200*( log(data(ind,x)/data(ind-1,x)) - log(data(ind-1,x)/data(ind-2,x)) );
88
89                  count = count + 1;
90              end
91          end
92
93
94          % PC(10)
95          [forecast,~,rmse,p1,p2,p3,p4,as_sq,as_lin] = forecast_target_2(predictors,target_z,target_y,date,'PC',h,per,
                nr_pc_factors,[]);
96          results_all{1,1,hor_index,p} = rmse;
97          results_all{2,1,hor_index,p} = p1;
98          results_all{3,1,hor_index,p} = p2;
99          results_all{4,1,hor_index,p} = p3;
100         results_all{5,1,hor_index,p} = p4;
101         forecast_series{1,hor_index,p} = forecast;
102         asymmetric_squared{1,hor_index,p} = as_sq;
103         asymmetric_linear{1,hor_index,p} = as_lin;
104
105         % SQPC(10)
106         [forecast,~,rmse,p1,p2,p3,p4,as_sq,as_lin] = forecast_target_2(predictors,target_z,target_y,date,'SQPC',h,per,
                nr_pc_factors,[]);
107         results_all{1,5,hor_index,p} = rmse;
108         results_all{2,5,hor_index,p} = p1;
109         results_all{3,5,hor_index,p} = p2;
110         results_all{4,5,hor_index,p} = p3;
111         results_all{5,5,hor_index,p} = p4;
```

```
112         forecast_series{5,hor_index,p} = forecast;
113         asymmetric_squared{5,hor_index,p} = as_sq;
114         asymmetric_linear{5,hor_index,p} = as_lin;
115
116         % PLS(a)
117         forecasts = cell(2,1);
118         [forecasts{1,1},~,rmse_1,p1_1,p2_1,p3_1,p4_1,as_sq_1,as_lin_1] = forecast_target_2(predictors,target_z,target_y,date,'
                PLS_a',h,per,1,[]);
119         [forecasts{2,1},~,rmse_2,p1_2,p2_2,p3_2,p4_2,as_sq_2,as_lin_2] = forecast_target_2(predictors,target_z,target_y,date,'
                PLS_a',h,per,2,[]);
120         [rmse] = min(rmse_1,rmse_2);
121         if rmse == rmse_1
122             mIND = 1;
123         else
124             mIND = 2;
125         end
126         p1 = min(p1_1,p1_2);
127         p2 = min(p2_1,p2_2);
128         p3 = min(p3_1,p3_2);
129         p4 = min(p4_1,p4_2);
130         as_sq = min(as_sq_1,as_sq_2);
131         as_lin = min(as_lin_1,as_lin_2);
132         results_all{1,2,hor_index,p} = rmse;
133         results_all{2,2,hor_index,p} = p1;
134         results_all{3,2,hor_index,p} = p2;
135         results_all{4,2,hor_index,p} = p3;
136         results_all{5,2,hor_index,p} = p4;
137         forecast = forecasts{mIND,1};
138         forecast_series{2,hor_index,p} = forecast;
139         asymmetric_squared{2,hor_index,p} = as_sq;
140         asymmetric_linear{2,hor_index,p} = as_lin;
141
142
143         % SQPLS(a)
144         forecasts = cell(2,1);
145         [forecasts{1,1},~,rmse_1,p1_1,p2_1,p3_1,p4_1,as_sq_1,as_lin_1] = forecast_target_2(predictors,target_z,target_y,date,'
                SQPLS_a',h,per,1,[]);
146         [forecasts{2,1},~,rmse_2,p1_2,p2_2,p3_2,p4_2,as_sq_2,as_lin_2] = forecast_target_2(predictors,target_z,target_y,date,'
                SQPLS_a',h,per,2,[]);
147         [rmse] = min(rmse_1,rmse_2);
148         if rmse == rmse_1
149             mIND = 1;
150         else
151             mIND = 2;
152         end
153         p1 = min(p1_1,p1_2);
154         p2 = min(p2_1,p2_2);
155         p3 = min(p3_1,p3_2);
156         p4 = min(p4_1,p4_2);
157         as_sq = min(as_sq_1,as_sq_2);
158         as_lin = min(as_lin_1,as_lin_2);
159         results_all{1,6,hor_index,p} = rmse;
160         results_all{2,6,hor_index,p} = p1;
161         results_all{3,6,hor_index,p} = p2;
162         results_all{4,6,hor_index,p} = p3;
163         results_all{5,6,hor_index,p} = p4;
164         forecast = forecasts{mIND,1};
165         forecast_series{6,hor_index,p} = forecast;
166         asymmetric_squared{6,hor_index,p} = as_sq;
167         asymmetric_linear{6,hor_index,p} = as_lin;
168
169         pls_b_performances = NaN(6,5);
```

```
170         pls_c_performances = NaN(6,5);
171         sqpls_b_performances = NaN(6,5);
172         sqpls_c_performances = NaN(6,5);
173
174         pls_b_performances_sq = NaN(6,9);
175         pls_c_performances_sq = NaN(6,9);
176         sqpls_b_performances_sq = NaN(6,9);
177         sqpls_c_performances_sq = NaN(6,9);
178
179         pls_b_performances_lin = NaN(6,9);
180         pls_c_performances_lin = NaN(6,9);
181         sqpls_b_performances_lin = NaN(6,9);
182         sqpls_c_performances_lin = NaN(6,9);
183
184
185         pls_b_forecasts = cell(6,1);
186         pls_c_forecasts = cell(6,1);
187         sqpls_b_forecasts = cell(6,1);
188         sqpls_c_forecasts = cell(6,1);
189
190         parfor extra_par = 1:6  %  iterate over possible values
191             % PLS(b)
192             forecasts = cell(2,1);
193             [forecasts{1,1},~,rmse_1,p1_1,p2_1,p3_1,p4_1,as_sq_1,as_lin_1] = forecast_target_2(predictors,target_z,target_y,
                    date,'PLS_b',h,per,1,extra_par);
194             [forecasts{2,1},~,rmse_2,p1_2,p2_2,p3_2,p4_2,as_sq_2,as_lin_2] = forecast_target_2(predictors,target_z,target_y,
                    date,'PLS_b',h,per,2,extra_par);
195             [rmse] = min(rmse_1,rmse_2);
196             if rmse == rmse_1
197                 mIND = 1;
198             else
199                 mIND = 2;
200             end
201             p1 = min(p1_1,p1_2);
202             p2 = min(p2_1,p2_2);
203             p3 = min(p3_1,p3_2);
204             p4 = min(p4_1,p4_2);
205             as_sq = min(as_sq_1,as_sq_2);
206             as_lin = min(as_lin_1,as_lin_2);
207             pls_b_performances(extra_par,:) = [rmse,p1,p2,p3,p4];
208             pls_b_performances_sq(extra_par,:) = as_sq';
209             pls_b_performances_lin(extra_par,:) = as_lin';
210             forecast = forecasts{mIND,1};
211             pls_b_forecasts{extra_par,1} = forecast;
212
213             % SQPLS(b)
214             forecasts = cell(2,1);
215             [forecasts{1,1},~,rmse_1,p1_1,p2_1,p3_1,p4_1,as_sq_1,as_lin_1] = forecast_target_2(predictors,target_z,target_y,
                    date,'SQPLS_b',h,per,1,extra_par);
216             [forecasts{2,1},~,rmse_2,p1_2,p2_2,p3_2,p4_2,as_sq_2,as_lin_2] = forecast_target_2(predictors,target_z,target_y,
                    date,'SQPLS_b',h,per,2,extra_par);
217             [rmse] = min(rmse_1,rmse_2);
218             if rmse == rmse_1
219                 mIND = 1;
220             else
221                 mIND = 2;
222             end
223             p1 = min(p1_1,p1_2);
224             p2 = min(p2_1,p2_2);
225             p3 = min(p3_1,p3_2);
226             p4 = min(p4_1,p4_2);
227             as_sq = min(as_sq_1,as_sq_2);
```

```
228             as_lin = min(as_lin_1,as_lin_2);
229             sqpls_b_performances(extra_par,:) = [rmse,p1,p2,p3,p4];
230             sqpls_b_performances_sq(extra_par,:) = as_sq';
231             sqpls_b_performances_lin(extra_par,:) = as_lin';
232             forecast = forecasts{mIND,1};
233             sqpls_b_forecasts{extra_par,1} = forecast;
234
235
236             % PLS(c)
237             forecasts = cell(2,1);
238             [forecasts{1,1},~,rmse_1,p1_1,p2_1,p3_1,p4_1,as_sq_1,as_lin_1] = forecast_target_2(predictors,target_z,target_y,
                    date,'PLS_c',h,per,1,extra_par);
239             [forecasts{2,1},~,rmse_2,p1_2,p2_2,p3_2,p4_2,as_sq_2,as_lin_2] = forecast_target_2(predictors,target_z,target_y,
                    date,'PLS_c',h,per,2,extra_par);
240             [rmse] = min(rmse_1,rmse_2);
241             if rmse == rmse_1
242                 mIND = 1;
243             else
244                 mIND = 2;
245             end
246             p1 = min(p1_1,p1_2);
247             p2 = min(p2_1,p2_2);
248             p3 = min(p3_1,p3_2);
249             p4 = min(p4_1,p4_2);
250             as_sq = min(as_sq_1,as_sq_2);
251             as_lin = min(as_lin_1,as_lin_2);
252             pls_c_performances(extra_par,:) = [rmse,p1,p2,p3,p4];
253             pls_c_performances_sq(extra_par,:) = as_sq';
254             pls_c_performances_lin(extra_par,:) = as_lin';
255             forecast = forecasts{mIND,1};
256             pls_c_forecasts{extra_par,1} = forecast;
257
258             % SQPLS(c)
259             forecasts = cell(2,1);
260             [forecasts{1,1},~,rmse_1,p1_1,p2_1,p3_1,p4_1,as_sq_1,as_lin_1] = forecast_target_2(predictors,target_z,target_y,
                    date,'SQPLS_c',h,per,1,extra_par);
261             [forecasts{2,1},~,rmse_2,p1_2,p2_2,p3_2,p4_2,as_sq_2,as_lin_2] = forecast_target_2(predictors,target_z,target_y,
                    date,'SQPLS_c',h,per,2,extra_par);
262             [rmse] = min(rmse_1,rmse_2);
263             if rmse == rmse_1
264                 mIND = 1;
265             else
266                 mIND = 2;
267             end
268             p1 = min(p1_1,p1_2);
269             p2 = min(p2_1,p2_2);
270             p3 = min(p3_1,p3_2);
271             p4 = min(p4_1,p4_2);
272             as_sq = min(as_sq_1,as_sq_2);
273             as_lin = min(as_lin_1,as_lin_2);
274             sqpls_c_performances(extra_par,:) = [rmse,p1,p2,p3,p4];
275             sqpls_c_performances_sq(extra_par,:) = as_sq';
276             sqpls_c_performances_lin(extra_par,:) = as_lin';
277             forecast = forecasts{mIND,1};
278             sqpls_c_forecasts{extra_par,1} = forecast;
279         end
280
281         [aba, abaIND] = min(pls_b_performances);
282         rmse=aba(1,1); p1=aba(1,2); p2=aba(1,3); p3=aba(1,4); p4=aba(1,5);
283         aba = min(pls_b_performances_sq);
284         as_sq = aba';
285         aba = min(pls_b_performances_lin);
```

```
286         as_lin = aba';
287         results_all{1,3,hor_index,p} = rmse;
288         results_all{2,3,hor_index,p} = p1;
289         results_all{3,3,hor_index,p} = p2;
290         results_all{4,3,hor_index,p} = p3;
291         results_all{5,3,hor_index,p} = p4;
292         forecast_series{3,hor_index,p} = pls_b_forecasts{abaIND(1,1),1};
293         asymmetric_squared{3,hor_index,p} = as_sq;
294         asymmetric_linear{3,hor_index,p} = as_lin;
295
296         [aba, abaIND] = min(pls_c_performances);
297         rmse=aba(1,1); p1=aba(1,2); p2=aba(1,3); p3=aba(1,4); p4=aba(1,5);
298         aba = min(pls_c_performances_sq);
299         as_sq = aba';
300         aba = min(pls_c_performances_lin);
301         as_lin = aba';
302         results_all{1,4,hor_index,p} = rmse;
303         results_all{2,4,hor_index,p} = p1;
304         results_all{3,4,hor_index,p} = p2;
305         results_all{4,4,hor_index,p} = p3;
306         results_all{5,4,hor_index,p} = p4;
307         forecast_series{4,hor_index,p} = pls_c_forecasts{abaIND(1,1),1};
308         asymmetric_squared{4,hor_index,p} = as_sq;
309         asymmetric_linear{4,hor_index,p} = as_lin;
310
311
312         [aba, abaIND] = min(sqpls_b_performances);
313         rmse=aba(1,1); p1=aba(1,2); p2=aba(1,3); p3=aba(1,4); p4=aba(1,5);
314         aba = min(sqpls_b_performances_sq);
315         as_sq = aba';
316         aba = min(sqpls_b_performances_lin);
317         as_lin = aba';
318         results_all{1,7,hor_index,p} = rmse;
319         results_all{2,7,hor_index,p} = p1;
320         results_all{3,7,hor_index,p} = p2;
321         results_all{4,7,hor_index,p} = p3;
322         results_all{5,7,hor_index,p} = p4;
323         forecast_series{7,hor_index,p} = sqpls_b_forecasts{abaIND(1,1),1};
324         asymmetric_squared{7,hor_index,p} = as_sq;
325         asymmetric_linear{7,hor_index,p} = as_lin;
326
327         [aba, abaIND] = min(sqpls_c_performances);
328         rmse=aba(1,1); p1=aba(1,2); p2=aba(1,3); p3=aba(1,4); p4=aba(1,5);
329         aba = min(sqpls_c_performances_sq);
330         as_sq = aba';
331         aba = min(sqpls_c_performances_lin);
332         as_lin = aba';
333         results_all{1,8,hor_index,p} = rmse;
334         results_all{2,8,hor_index,p} = p1;
335         results_all{3,8,hor_index,p} = p2;
336         results_all{4,8,hor_index,p} = p3;
337         results_all{5,8,hor_index,p} = p4;
338         forecast_series{8,hor_index,p} = sqpls_c_forecasts{abaIND(1,1),1};
339         asymmetric_squared{8,hor_index,p} = as_sq;
340         asymmetric_linear{8,hor_index,p} = as_lin;
341
342
343
344         kpc_polynomial_performances = NaN(6,5);
345         kpc_radial_performances = NaN(6,5);
346         kpls_polynomial_performances = NaN(6,5);
347         kpls_radial_perfromances = NaN(6,5);
```

```
348
349         kpc_polynomial_performances_sq = NaN(6,9);
350         kpc_radial_performances_sq = NaN(6,9);
351         kpls_polynomial_performances_sq = NaN(6,9);
352         kpls_radial_perfromances_sq = NaN(6,9);
353
354         kpc_polynomial_performances_lin = NaN(6,9);
355         kpc_radial_performances_lin = NaN(6,9);
356         kpls_polynomial_performances_lin = NaN(6,9);
357         kpls_radial_perfromances_lin = NaN(6,9);
358
359         kpc_polynomial_forecasts = cell(6,1);
360         kpc_radial_forecasts = cell(6,1);
361         kpls_polynomial_forecasts = cell(6,1);
362         kpls_radial_forecasts = cell(6,1);
363
364        % Initialize search set for adjustable kernel parameters
365         sigma_0 = sqrt(67);
366         kernel_pars_polynomial = [0.5*sigma_0; sigma_0; 2*sigma_0; 4*sigma_0; 8*sigma_0];
367         sigma_0 = chi2inv(0.95,132)/pi;
368         kernel_pars_radial = [0.5*sigma_0; sigma_0; 2*sigma_0; 4*sigma_0; 8*sigma_0];
369
370        % Analysis limited to 1 factor only due to "expensive" computation
371         parfor e = 1:length(kernel_pars_polynomial)
372             % KPC Polynomial
373             [forecast,~,rmse,p1,p2,p3,p4,as_sq,as_lin] = forecast_target_2(predictors,target_z,target_y,date,'KPC_polynomial',
                    h,per,1,kernel_pars_polynomial(e,1));
374             kpc_polynomial_performances(e,:) = [rmse,p1,p2,p3,p4];
375             kpc_polynomial_performances_sq(e,:) = as_sq';
376             kpc_polynomial_performances_lin(e,:) = as_lin';
377             kpc_polynomial_forecasts{e,1} = forecast;
378
379             % KPC Radial
380             [forecast,~,rmse,p1,p2,p3,p4,as_sq,as_lin] = forecast_target_2(predictors,target_z,target_y,date,'KPC_radial',h,
                    per,1,kernel_pars_radial(e,1));
381             kpc_radial_performances(e,:) = [rmse,p1,p2,p3,p4];
382             kpc_radial_performances_sq(e,:) = as_sq';
383             kpc_radial_performances_lin(e,:) = as_lin';
384             kpc_radial_forecasts{e,1} = forecast
385
386
387             % KPLS Polynomial
388             [forecast,~,rmse,p1,p2,p3,p4,as_sq,as_lin] = forecast_target_2(predictors,target_z,target_y,date,'KPLS_polynomial'
                    ,h,per,1,kernel_pars_polynomial(e,1));
389             kpls_polynomial_performances(e,:) = [rmse,p1,p2,p3,p4];
390             kpls_polynomial_performances_sq(e,:) = as_sq';
391             kpls_polynomial_performances_lin(e,:) = as_lin';
392             kpls_polynomial_forecasts{e,1} = forecast;
393
394             % KPLS Radial
395             [forecast,~,rmse,p1,p2,p3,p4,as_sq,as_lin] = forecast_target_2(predictors,target_z,target_y,date,'KPLS_radial',h,
                    per,1,kernel_pars_radial(e,1));
396             kpls_radial_perfromances(e,:) = [rmse,p1,p2,p3,p4];
397             kpls_radial_perfromances_sq(e,:) = as_sq';
398             kpls_radial_perfromances_lin(e,:) = as_lin';
399             kpls_radial_forecasts{e,1} = forecast;
400         end
401
402         [aba, abaIND] = min(kpc_polynomial_performances);
403         rmse=aba(1,1); p1=aba(1,2); p2=aba(1,3); p3=aba(1,4); p4=aba(1,5);
404         aba = min(kpc_polynomial_performances_sq);
405         as_sq = aba';
```

```
406         aba = min(kpc_polynomial_performances_lin);
407         as_lin = aba';
408         results_all{1,9,hor_index,p} = rmse;
409         results_all{2,9,hor_index,p} = p1;
410         results_all{3,9,hor_index,p} = p2;
411         results_all{4,9,hor_index,p} = p3;
412         results_all{5,9,hor_index,p} = p4;
413         forecast_series{9,hor_index,p} = kpc_polynomial_forecasts{abaIND(1,1),1};
414         asymmetric_squared{9,hor_index,p} = as_sq;
415         asymmetric_linear{9,hor_index,p} = as_lin;
416
417
418         [aba, abaIND] = min(kpc_radial_performances);
419         rmse=aba(1,1); p1=aba(1,2); p2=aba(1,3); p3=aba(1,4); p4=aba(1,5);
420         aba = min(kpc_radial_performances_sq);
421         as_sq = aba';
422         aba = min(kpc_radial_performances_lin);
423         as_lin = aba';
424         results_all{1,10,hor_index,p} = rmse;
425         results_all{2,10,hor_index,p} = p1;
426         results_all{3,10,hor_index,p} = p2;
427         results_all{4,10,hor_index,p} = p3;
428         results_all{5,10,hor_index,p} = p4;
429         forecast_series{10,hor_index,p} = kpc_radial_forecasts{abaIND(1,1),1};
430         asymmetric_squared{10,hor_index,p} = as_sq;
431         asymmetric_linear{10,hor_index,p} = as_lin;
432
433
434         [aba, abaIND] = min(kpls_polynomial_performances);
435         rmse=aba(1,1); p1=aba(1,2); p2=aba(1,3); p3=aba(1,4); p4=aba(1,5);
436         aba = min(kpls_polynomial_performances_sq);
437         as_sq = aba';
438         aba = min(kpls_polynomial_performances_lin);
439         as_lin = aba';
440         results_all{1,11,hor_index,p} = rmse;
441         results_all{2,11,hor_index,p} = p1;
442         results_all{3,11,hor_index,p} = p2;
443         results_all{4,11,hor_index,p} = p3;
444         results_all{5,11,hor_index,p} = p4;
445         forecast_series{11,hor_index,p} = kpls_polynomial_forecasts{abaIND(1,1),1};
446         asymmetric_squared{11,hor_index,p} = as_sq;
447         asymmetric_linear{11,hor_index,p} = as_lin;
448
449
450         [aba, abaIND] = min(kpls_radial_perfromances);
451         rmse=aba(1,1); p1=aba(1,2); p2=aba(1,3); p3=aba(1,4); p4=aba(1,5);
452         aba = min(kpls_radial_perfromances_sq);
453         as_sq = aba';
454         aba = min(kpls_radial_perfromances_lin);
455         as_lin = aba';
456         results_all{1,12,hor_index,p} = rmse;
457         results_all{2,12,hor_index,p} = p1;
458         results_all{3,12,hor_index,p} = p2;
459         results_all{4,12,hor_index,p} = p3;
460         results_all{5,12,hor_index,p} = p4;
461         forecast_series{12,hor_index,p} = kpls_radial_forecasts{abaIND(1,1),1};
462         asymmetric_squared{12,hor_index,p} = as_sq;
463         asymmetric_linear{12,hor_index,p} = as_lin;
464
465     end
466 end
467
```

```
468
469 save('final_results_1', 'results_all', 'forecast_series')
470 save('nonlinear_loss_results', 'asymmetric_squared', 'asymmetric_linear')
471
472 toc
473
474 % Generate LaTex tables for RMSE:
475 latex_table_1(results_all,1,1,2);
476 latex_table_1(results_all,1,3,4);
477
478 % Generate LaTex tables for the other 4 performance measures:
479 for ii = 2:5
480     latex_table_1(results_all,ii,1,2);
481     latex_table_1(results_all,ii,3,4);
482 end
483
484 latex_table_3(results_all);
485
486
487 %%% Addition after draft version: include piecewise linear & asymmetric loss
488 %%% functions as defined in Elliott & Timmermann, 2004 for comparison to the
489 %%% other 2 asymmetric loss functions introduced here.
490 %%
491 %%% Evaluate the new loss functions for theta = 0.1, 0.2, ..., 0.9
492 %% asymmetric_squared_loss = NaN(9,12,4,7);
493 %% piecewise_linear_loss = NaN(9,12,4,7);
494
495
496
497 %% Robustness of Fuentes et al. (2015) PLS (a) and (b) models -- only RMSE:
498
499 pls_b_results = cell(6,1);
500 pls_c_results = cell(6,1);
501
502 for extra_par = 1:6
503
504     extra_par
505
506     pls_b_curr = NaN(7,4);
507     pls_c_curr = NaN(7,4);
508
509     for hor_index = 1:4
510         for p = 1:7 % iterate over each period to forecast
511
512             h = horizons(hor_index,1); % current iteration forecast horizon
513             per = periods{p,1}; % current iteration estimation/forecast subsample
514
515             % Compute the target variable Y and Z:
516             x = 115; % column 115 in the data/predictor matrix corresponds to CPI
517             count = 1;
518             for y = 1960:2003
519                 for m = 1:12
520                     ind = find(year == y & month == m);
521                     if ind-h-1 > 0
522                         target_y(count,1) = (1200/h)*log(data(ind,x)/data(ind-h,x)) - 1200*log(data(ind-h,x)/data(ind-h-1,x));
523                     else
524                         target_y(count,1) = NaN;
525                     end
526                     target_z(count,1) = 1200*( log(data(ind,x)/data(ind-1,x)) - log(data(ind-1,x)/data(ind-2,x)) );
527
528                     count = count + 1;
529                 end
```

```
            end


            % PLS(b)
            [~,~,rmse_1,~,~,~] = forecast_target_2(predictors,target_z,target_y,date,'PLS_b',h,per,1,extra_par);
            [~,~,rmse_2,~,~,~] = forecast_target_2(predictors,target_z,target_y,date,'PLS_b',h,per,2,extra_par);
            rmse = min(rmse_1,rmse_2);
            pls_b_curr(p,hor_index) = rmse;

            % PLS(c)
            [~,~,rmse_1,~,~,~] = forecast_target_2(predictors,target_z,target_y,date,'PLS_c',h,per,1,extra_par);
            [~,~,rmse_2,~,~,~] = forecast_target_2(predictors,target_z,target_y,date,'PLS_c',h,per,2,extra_par);
            rmse = min(rmse_1,rmse_2);
            pls_c_curr(p,hor_index) = rmse;

        end
    end


    pls_b_results{extra_par,1} = pls_b_curr;
    pls_c_results{extra_par,1} = pls_c_curr;

end

save('fuentes_robustness_results_1', 'pls_b_results', 'pls_c_results')

% Generate LaTex tables for the robustness of the results of Fuentes:
latex_table_2(pls_b_results,'b');
latex_table_2(pls_c_results,'c');


%% Robustness of Kernel models -- RMSE only:

kpc1_results = cell(5,7,4); %dimensions: extra_param; subsample; horizon;
kpc2_results = cell(5,7,4);
kpls1_results = cell(5,7,4);
kpls2_results = cell(5,7,4);


for hor_index = 1:4
    hor_index

    for p = 1:7 % iterate over each period to forecast

        h = horizons(hor_index,1); % current iteration forecast horizon
        per = periods{p,1}; % current iteration estimation/forecast subsample

        % Compute the target variable Y and Z:
        x = 115; % column 115 in the data/predictor matrix corresponds to CPI
        count = 1;
        for y = 1960:2003
            for m = 1:12
                ind = find(year == y & month == m);
                if ind-h-1 > 0
                    target_y(count,1) = (1200/h)*log(data(ind,x)/data(ind-h,x)) - 1200*log(data(ind-h,x)/data(ind-h-1,x));
                else
                    target_y(count,1) = NaN;
                end
                target_z(count,1) = 1200*( log(data(ind,x)/data(ind-1,x)) - log(data(ind-1,x)/data(ind-2,x)) );

                count = count + 1;
            end
```

```
592         end
593
594
595         sigma_0 = sqrt(67);
596         kernel_pars_polynomial = [0.5*sigma_0; sigma_0; 2*sigma_0; 4*sigma_0; 8*sigma_0]; %; 16*sigma_0; 32*sigma_0; 64*
                sigma_0; 128*sigma_0];
597         sigma_0 = chi2inv(0.95,132)/pi;
598         kernel_pars_radial = [0.5*sigma_0; sigma_0; 2*sigma_0; 4*sigma_0; 8*sigma_0]; %; 16*sigma_0; 32*sigma_0; 64*sigma_0;
                128*sigma_0];
599
600         % Analysis limited to 1 factor only due to "expensive" computation
601         for e = 1:length(kernel_pars_polynomial)
602             % KPC Polynomial
603             [~,~,rmse,~,~,~,~] = forecast_target_2(predictors,target_z,target_y,date,'KPC_polynomial',h,per,1,
                    kernel_pars_polynomial(e,1));
604             kpc1_results{e,p,hor_index} = rmse;
605
606             % KPC Radial
607             [~,~,rmse,~,~,~,~] = forecast_target_2(predictors,target_z,target_y,date,'KPC_radial',h,per,1,kernel_pars_radial(e
                    ,1));
608             kpc2_results{e,p,hor_index} = rmse;
609
610
611             % KPLS Polynomial
612             [~,~,rmse,~,~,~,~] = forecast_target_2(predictors,target_z,target_y,date,'KPLS_polynomial',h,per,1,
                    kernel_pars_polynomial(e,1));
613             kpls1_results{e,p,hor_index} = rmse;
614
615             % KPLS Radial
616             [~,~,rmse,~,~,~,~] = forecast_target_2(predictors,target_z,target_y,date,'KPLS_radial',h,per,1,kernel_pars_radial(
                    e,1));
617             kpls2_results{e,p,hor_index} = rmse;
618         end
619
620     end
621 end
622
623 save('kernel_sensitivity_results_1', 'kpc1_results', 'kpc2_results', 'kpls1_results', 'kpls2_results')
624
625
626 kpc1 = cell(5,1);
627 kpc2 = cell(5,1);
628 kpls1 = cell(5,1);
629 kpls2 = cell(5,1);
630
631 for i = 1:5
632     kpc1_curr = NaN(7,4);
633     kpc2_curr = NaN(7,4);
634     kpls1_curr = NaN(7,4);
635     kpls2_curr = NaN(7,4);
636
637     for m=1:7
638         for h=1:4
639             kpc1_curr(m,h) = kpc1_results{i,m,h};
640             kpc2_curr(m,h) = kpc2_results{i,m,h};
641             kpls1_curr(m,h) = kpls1_results{i,m,h};
642             kpls2_curr(m,h) = kpls2_results{i,m,h};
643         end
644     end
645
646     kpc1{i,1} = kpc1_curr;
647     kpc2{i,1} = kpc2_curr;
```

```
    kpls1{i,1} = kpls1_curr;
    kpls2{i,1} = kpls2_curr;
end

save('kernel_sensitivity_results_final', 'kpc1', 'kpc2', 'kpls1', 'kpls2')

latex_table_4(kpc1,'KPC1');
latex_table_4(kpc2,'KPC2');
latex_table_4(kpls1,'KPLS1');
latex_table_4(kpls2,'KPLS2');


%% Diebold-Mariano (DM) Tests

clear;
clc;

load data_2003.mat

t = datevec(date);
year = t(:,1);
month = t(:,2);

load final_results_1.mat


latex_table_5(results_all,forecast_series,data,date)
```

Listing 2: Function that generates and evaluates forecasts

```
function [forecast,forecast_AR4,RMSE,perf1,perf2,perf3,perf4,as_sq,as_lin] = forecast_target_2(predictors,target_z,target_y,
    date,method,h,subsample,nr_factors,extra_parameter)
% Forecast h-step ahead based on Principal Component method
% Note: this function's input are the "cleaned" data of 528 observations.


dt = date;
date = datevec(date);
year = date(:,1);
month = date(:,2);

% Define forecast sample: (US dataset)
if isequal(subsample, 'M1')
    est_end_year = 1970; for_end_year = 1980;
elseif isequal(subsample, 'M2')
    est_end_year = 1980; for_end_year = 1990;
elseif isequal(subsample, 'M3')
    est_end_year = 1990; for_end_year = 2000;
elseif isequal(subsample, 'M4')
    est_end_year = 1970; for_end_year = 1990;
elseif isequal(subsample, 'M5')
    est_end_year = 1970; for_end_year = 2000;
elseif isequal(subsample, 'M6')
    est_end_year = 1980; for_end_year = 2000;
elseif isequal(subsample, 'M7')
    est_end_year = 1970; for_end_year = 2003;
end
```

```
27
28
29
30 % "Expanding Window" forecasting. Final date extends by 1 month at a time:
31
32 cc = 1;
33
34 for y = est_end_year:for_end_year
35     for m = 1:12
36
37         if y == est_end_year && ( m == 1 || m == 2 )
38             % Do not do anything for the first 2 months because forecast
39             % sample always begins in March of 'est_end_year'.
40
41         else
42             % Generate the variables according to the appropriate
43             % estimation sample:
44             est_end_date = datetime([y,m,1]) - calmonths(h);
45             est_end_date = datevec(est_end_date);
46             est_end_y = est_end_date(1,1);
47             est_end_m = est_end_date(1,2);
48
49             est_st_index = find(year==1960 & month==3);
50             est_end_index = find(year==est_end_y & month==est_end_m);
51
52             X = predictors(est_st_index:est_end_index,:);
53             Y = target_y(est_st_index:est_end_index,:);
54             Z = target_z(est_st_index:est_end_index,:);
55
56             dt1 = dt(est_st_index:est_end_index,:);
57
58             X = timetable(dt1,X);
59             Y = timetable(dt1,Y);
60             Z = timetable(dt1,Z);
61
62
63             % Compute latent factors based on 'method': (Here can also use
64             % case/switch commands to check for method)
65
66             % Linear Methods:
67             if isequal(method, 'PC')  % Principal Component (PC) method
68                 F = extract_factors_PC(X,dt1,nr_factors,0); % input 0 in the end indicates linear PC (instead of SQPC)
69
70             elseif isequal(method, 'PLS_a')  % PLS(a) method
71                 F = extract_factors_PLS(X,Y,Z,h,dt1,nr_factors,'a',[],[],0); % input 0 in the end indicates linear PLS
72
73             elseif isequal(method, 'PLS_b')  % PLS(b) method
74                 F = extract_factors_PLS(X,Y,Z,h,dt1,nr_factors,'b',extra_parameter,[],0);
75
76             elseif isequal(method, 'PLS_c')  % PLS(c) method
77                 F = extract_factors_PLS(X,Y,Z,h,dt1,nr_factors,'c',[],extra_parameter,0);
78
79             % Squared Methods:
80             elseif isequal(method, 'SQPC')  % Squared PC (SQPC) method
81                 F = extract_factors_PC(X,dt1,nr_factors,1); % input 1 in the end indicates SQPC (instead of linear PC)
82
83             elseif isequal(method, 'SQPLS_a')  % Squared PLS(a) method
84                 F = extract_factors_PLS(X,Y,Z,h,dt1,nr_factors,'a',[],[],1); % input 1 in the end indicates SQPLS
85
86             elseif isequal(method, 'SQPLS_b')  % Squared PLS(b) method
87                 F = extract_factors_PLS(X,Y,Z,h,dt1,nr_factors,'b',extra_parameter,[],1);
88
```

```
89          elseif isequal(method, 'SQPLS_c')  % Squared PLS(c) method
90              F = extract_factors_PLS(X,Y,Z,h,dt1,nr_factors,'c',[],extra_parameter,1);
91
92          % Kernel Methods
93          elseif isequal(method, 'KPC_polynomial') % KPC with polynomial kernel function
94              F = extract_factors_KPC(X,dt1,nr_factors,'polynomial',extra_parameter); % 'extra_parameter' in this case
                    corresponds to the tunable parameter of the kernel function
95
96          elseif isequal(method, 'KPC_radial') % KPC with gaussian radial basis kernel function
97              F = extract_factors_KPC(X,dt1,nr_factors,'radial',extra_parameter);
98
99          elseif isequal(method, 'KPLS_polynomial') % KPLS with polynomial kernel function
100             F = extract_factors_KPLS(X,Y,dt1,'polynomial',extra_parameter);
101
102         elseif isequal(method, 'KPLS_radial') % KPLS with gaussian radial basis kernel function
103             F = extract_factors_KPLS(X,Y,dt1,'radial',extra_parameter);
104
105         end
106
107
108
109         % Estimate forecasting equation; lags must be chosen based on
110         % BIC; try combinaitons 1-6 for lags of both "variables":
111         BICs = NaN(36,1);
112         lag_comb = NaN(36,2);
113
114         x_matrices = cell(6,6);
115         X_matrices_forecast = cell(6,6); % store x matrix for every lag combination to later use in forecasting
116         b_vectors_forecast = cell(6,6); % store regression coefficients for every lag combination to later use in
                forecasting
117
118         count = 1;
119
120         f_1 = table2array(F);
121         z_1 = table2array(Z);
122         y_1 = table2array(Y);
123
124         maxL1 = 6;
125         maxL2 = 6;
126
127         for lag1 = 1:maxL1  % 6 % lags of Z (that is, lags of the "target")
128             for lag2 = 1:maxL2  %6 % lags of factors F
129
130                 x_matrix = [];
131                 x_matrix_f = [];
132
133                 if ~isequal(method, 'PLS_b') && ~isequal(method, 'SQPLS_b') % add lags of the 'target' to the forecasting
                        equation only if we are not using (b) approach
134                     for i = 1:lag1  % lags of z
135                         a = z_1(1:end-(i+h-1),:);
136                         a1 = NaN(i+h-1,1);
137                         a = [a1; a];
138                         x_matrix = [x_matrix a];
139
140                         a = z_1(1:end-(i-1),:);
141                         a1 = NaN(i-1,1);
142                         a = [a1; a];
143                         x_matrix_f = [x_matrix_f a];
144                     end
145                 end
146
147                 for i = 1:lag2  % lags of F (these should be added to the forecasting equation in all cases)
```

```
148                         a = f_1(1:end-(i+h-1),:);
149                         a1 = NaN(i+h-1,size(a,2));
150                         a = [a1; a];
151                         x_matrix = [x_matrix a];
152
153                         a = f_1(1:end-(i-1),:);
154                         a1 = NaN(i-1,size(a,2));
155                         a = [a1; a];
156                         x_matrix_f = [x_matrix_f a];
157                     end
158
159                     x_matrix = [ones(size(x_matrix,1),1), x_matrix]; % add constant to the forecasting regression
160                     y_matrix = y_1;
161
162                     bb1 = ~isnan(x_matrix);
163                     bb2 = sum(bb1);
164                     nr_obs = min(bb2);
165
166                     if nr_obs > size(x_matrix,2)
167                         [b,~,res] = regress(y_matrix, x_matrix); % estimate forecasting equation coefficients
168
169                        % Store relevant matrices for later use
170                         X_matrices_forecast{lag1,lag2} = x_matrix_f;
171                         b_vectors_forecast{lag1,lag2} = b;
172                         x_matrices{lag1,lag2} = x_matrix;
173
174                         T = sum(~isnan(res)); % number of observations used in the regression
175                         n = size(x_matrix,2) ;
176
177                         % Compute the Bayes Information Criterion
178                         res = res(end-T+1:end,:);
179                         bic = log((res'*res)/T);
180                         bic = bic + n*log(T)/T;
181                         %bic = bic + 2*n/T;  % an also widely used criterion
182                         BICs(count,1) = bic;
183
184                     else % this is used to implicitly restrict the lag order to 4 when sample size does not permit 5 or 6 lags
                               (useful for h-24)
185                         BICs(count,1) = 10^6; % set BIC to a very large value in case the lag combination is infeasible based
                                on sample size
186
187                     end
188
189                     lag_comb(count,:) = [lag1,lag2];
190
191                     count = count + 1;
192                 end
193             end
194
195             [~,minIND] = min(BICs);
196             L1 = lag_comb(minIND,1);
197             L2 = lag_comb(minIND,2);
198
199
200             % Generate the h-step-ahead forecast for the target variable
201             % using the best model as indicated by BIC:
202             x_matrix = X_matrices_forecast{L1,L2};
203             x_matrix = [1, x_matrix(end,:)];
204             b = b_vectors_forecast{L1,L2};
205
206             f = x_matrix*b;
207             forecast(cc,1) = f;
```

```
208
209
210
211             %=========================================================%
212
213             % SIMPLE AR(4) BENCHMARK (estimation + forecast):
214
215             y_matrix = y_1;
216
217             x_matrix = [];
218             x_matrix_f = [];
219
220             for i = 1:4
221                 a = z_1(1:end-(i+h-1),:);
222                 a1 = NaN(i+h-1,1);
223                 a = [a1; a];
224                 x_matrix = [x_matrix a];
225
226                 a = z_1(1:end-(i-1),:);
227                 a1 = NaN(i-1,1);
228                 a = [a1; a];
229                 x_matrix_f = [x_matrix_f a];
230             end
231
232
233             x_matrix = [ones(size(x_matrix,1),1), x_matrix]; % add a constant to the AR(4)
234             b = regress(y_matrix, x_matrix); % estimate AR(4) regression coefficients
235
236             % Forecast from AR(4):
237             x_matrix = [1 x_matrix_f(end,:)];
238             fAR4 = x_matrix*b;
239             forecast_AR4(cc,1) = fAR4;
240
241
242             cc = cc + 1; % increment cc for the next expanding window iteration
243         end
244
245     end
246 end
247
248
249
250 % Calculate the RMSE of the "method" against the AR(4) benchmark:
251 for_st_index = find(year==est_end_year & month==3);
252 for_end_index = find(year==for_end_year & month==12);
253
254 target_actual = target_y(for_st_index:for_end_index,:);
255
256 MSE_method = mean((forecast-target_actual).^2);
257 MSE_AR4 = mean((forecast_AR4-target_actual).^2);
258 RMSE = MSE_method/MSE_AR4;
259
260
261 % Calculate additional "alternative" performance measures:
262 performance_measures = evaluate_performance_alternative( (forecast-target_actual), (forecast_AR4-target_actual) );
263 perf1  = performance_measures(1,1);  % 1) mean absolute error
264 perf2 = performance_measures(2,1);  % 2) cubic loss
265 perf3 = performance_measures(3,1);  % 3) asymmetric 1 (cubic loss for negative errors)
266 perf4 = performance_measures(4,1);  % 4) asymmetric 2 (cubic loss for positive errors)
267
268 as_sq = performance_measures(5:13,1);  % asymmetric squared loss for all values of theta
269 as_lin = performance_measures(14:22,1);  % piecewise linear loss for all values of theta
```

```

end
```

Listing 3: Function that extracts latent factors using linear and squared PCA

```
function [factors] = extract_factors_PC(X,dates,nr_factors,squared_indicator)
% If 'squared_indicator' equals 1, the perform SQPC. Otherwise perform the
% liear PC.

x_mat = table2array(X);

if squared_indicator == 1 % enlarge predictor matrix in case of SQPC
    x_mat_2 = NaN(size(x_mat,1),size(x_mat,2)); % this matrix will contain the "squared" components and will be latter merged
          with 'x_mat'

    for i = 1:size(x_mat,2)
        x_mat_2(:,i) = x_mat(:,i).^2;
    end

    x_mat = [x_mat x_mat_2];
end


%Standardize each variable prior to performing PCA
for i = 1:size(x_mat,2)
    x_mat(:,i) = ( x_mat(:,i) - mean(x_mat(:,i)) ) / std(x_mat(:,i));
end


% Perform eigenvalue decomposition of X'X and use
% the eigenvectors corresponding to 10 largest eigenvalues
% to compute the 10 PCs (alternative approach also possible...)

%[V,D] = eig(x_mat*x_mat','vector'); % alternative approch, both equivalent
[V,D] = eig(x_mat'*x_mat,'vector');
[~,ind] = sort(D);
V_sorted = V(:,ind);
lambda = V_sorted(:,end-(nr_factors-1):end); % eigenvectors corresponding to 'nr_factors' largest eigenvectors
%F = lambda; % alternative approch, both equivalent
F = x_mat*lambda;

factors = timetable(dates,F);

end
```

Listing 4: Function that extracts latent factors using linear and squared PLS

```
function[factors] = extract_factors_PLS(X,Y,Z,h,dates,nr_factors,approach,lags_b,p_c,squared_indicator)
% Input 'approach' can take values 'a', 'b' and 'c' corresponding to which
% version of PLS one is using to extract the latent factors. If
% 'squared_indicator' equals 1, then perform SQPLS, otherwise perform
% linear PLS.
```

```
6
7
8  % If performing SQPLS, first enlarge the predictor matrix with the squared
9  % terms.
10 if squared_indicator == 1
11     X_new = table2array(X);
12     X_new_2 = NaN(size(X_new,1),size(X_new,2));
13     for i = 1:size(X_new,2)
14         X_new_2(:,i) = X_new(:,i).^2;
15     end
16     X_new = [X_new X_new_2];
17     X = timetable(dates,X_new);
18 end
19
20
21 % If the approach is PLS(b), first check if the number of lags of the target
22 % variable to be included in the enlarged predictor matrix is valid (i.e.,
23 % between 1 and 6)
24 if isequal(approach, 'b')
25     indicator_b = (lags_b >= 1) & (lags_b <= 6);
26     if indicator_b == 0
27         msg = ('Invalid lag order.');
28         error(msg)
29     end
30 end
31
32 % If the approach is PLS(c), first check if the lag order og the AR(p) to
33 % be fitted to the target variable is valid (i.e., between 1 and 6)
34 if isequal(approach, 'c')
35     indicator_c = (p_c >= 1) & (p_c <= 6);
36     if indicator_c == 0
37         msg = ('Invalid lag order.');
38         error(msg)
39     end
40 end
41
42
43
44 % If the approach is PLS(b), enlarge the predictor matrix with lags_b lags
45 % of the target
46 if isequal(approach, 'b') % first enlarge the X matrix when
47     X_new = table2array(X);
48     z1 = table2array(Z);
49
50     for i = 1:lags_b
51         a = z1(1:end-(i-1),:);
52         a1 = NaN(i-1,1);
53         a = [a1; a];
54         X_new = [X_new a];
55     end
56     X = timetable(dates,X_new);
57 end
58
59
60 % If the approach is PLS(c), replace Y with the residuals from an AR(p)
61 % process fitted to the target variable Y; in this case, p = 'p_c':
62 if isequal(approach, 'c')
63     Y_new = table2array(Y);
64     y_lagged = [];
65     for i = 1:p_c
66         a = Y_new(1:end-i,:);
67         a1 = NaN(i,1);
```

```
        a = [a1; a];
        y_lagged = [y_lagged a];
    end
    [~,~,res] = regress(table2array(Y), y_lagged);
    Y = timetable(dates, res);
end



factors = NaN(size(table2array(X),1), nr_factors); % initialize the matrix that will contain extracted factors

% Iteratively extract the factors
for n = 1:nr_factors

    % Construct the "X" and "Y_h" matrices used to construct M based on the
    % approach to be used.
    if n == 1
        if isequal(approach, 'a')
            xx = table2array(lag(X,h));
            xx = xx(1+h:end,:);
            yy = table2array(Y);
            yy = yy(1+h:end,:);


        elseif isequal(approach, 'b')
            xx = table2array(lag(X,h));

            ind = ~isnan(xx(:,end));
            ind2 = find(ind==1);

            xx = xx(ind2(1,1):end,:);
            yy = table2array(Y);
            yy = yy(ind2(1,1):end,:);


        elseif isequal(approach, 'c')
            xx = table2array(lag(X,h));
            yy = table2array(Y);

            ind = ~isnan(xx(:,end));
            ind2 = find(ind==1);

            ind_ = ~isnan(yy);
            ind2_ = find(ind_==1);

            final_ind = max(ind2(1,1),ind2_(1,1));

            xx = xx(final_ind:end,:);
            yy = yy(final_ind:end,:);


        else  % throw an error if the input approach is not valid
            msg = ('Invalid PLS approach.');
            error(msg)

        end


        % Standardize before constructing the first factor
        for i = 1:size(xx,2)
            xx(:,i) = ( xx(:,i) - mean(xx(:,i)) ) / std(xx(:,i));
        end
```

```
        yy = ( yy - mean(yy) ) / std(yy);

        predictors = table2array(X);
        target = table2array(Y);
        for i = 1:size(predictors,2)
            predictors(:,i) = ( predictors(:,i) - nanmean(predictors(:,i)) ) / nanstd(predictors(:,i));
        end
        target = ( target - nanmean(target) ) / nanstd(target);

        X = timetable(dates,predictors);
        Y = timetable(dates,target);

    end


    % Extract the n-th factor:
    aaa = xx'*yy;
    M = aaa*aaa'; % M matrix constructed as in Fuentes et al. (2015)
    [V,D] = eig(M, 'vector'); % compute eigenvalues and eigenvectors of M
    [~,max_ind] = max(D); % find position of the largest eigenvalue
    lambda = V(:,max_ind); % eigenvector corresponding to largest eigenvalue of M

    F = table2array(X)*lambda; % construct n-th factor
    ff = xx*lambda;
    factors(:,n) = F; % store F in the final factor matrix


    % Update X and Y as the residuals of the simple regressions of Y on the
    % previous PLS component and of each col of X on the previous PLS
    % component, respectively:

    predictors = table2array(X);
    predictors_new = NaN(size(predictors,1), size(predictors,2));
    for i = 1:size(predictors,2)
        [~,~,res] = regress(predictors(:,i), F); % constant is included in these regressions
        predictors_new(:,i) = res;
    end
    X = timetable(dates,predictors_new); % update aach column of X with the residual series of the above regressions

    [~,~,yy] = regress(yy,  ff);
    for i = 1:size(xx,2)
        [~,~,xx(:,i)] = regress(xx(:,i), ff);
    end

end


factors = timetable(dates,factors); % output factor matrix in timetable format

end
```

Listing 5: Function that extracts latent factors using KPC

```
function [factors] = extract_factors_KPC(X,dates,nr_factors,kernel_name,kernel_parameter)

x_mat = table2array(X);

```

```
% Standardize each variable prior to performing KPCA
for i = 1:size(x_mat,2)
    x_mat(:,i) = ( x_mat(:,i) - mean(x_mat(:,i)) ) / std(x_mat(:,i));
end


% Construct the "kernel matrix"
T = size(x_mat,1);
K = NaN(T,T);

for i = 1:T
    for j = 1:T
        vec_i = x_mat(i,:)'; % transpose i-th obs. vec. to get column vector
        vec_j = x_mat(j,:)'; % transpose j-th obs. vec. to get column vector

        if isequal(kernel_name, 'polynomial')
            K(i,j) = polynomial_kernel(vec_i,vec_j,kernel_parameter);

        elseif isequal(kernel_name, 'radial')
            K(i,j) = gaussian_radial_basis_kernel(vec_i,vec_j,kernel_parameter);

        else
            msg = ('Invalid Kernel name.');
            error(msg)

        end

    end
end


% Before applying KPCA, mean cetering in the high-dimensional space should
% be performed:
I = eye(T);
mm = I - (1/T)*ones(T,1)*ones(1,T);
K = mm*K*mm;


% Calculate the eigenvectors of K and set the factors equal to the
% eigenvectors of K corresponding to the 'nr_factors' largest eigenvectors:
[V,D] = eig(K,'vector');
[D_sorted,ind] = sort(D);
V_sorted = V(:,ind);
vv = D_sorted(end-(nr_factors-1):end,1);
lambda = V_sorted(:,end-(nr_factors-1):end); % eigenvectors corresponding to 'nr_factors' largest eigenvectors

F = lambda;

factors = timetable(dates,F);

end
```

Listing 6: Function that extracts latent factors using KPLS

```
function [factors] = extract_factors_KPLS(X,Y,dates,kernel_name,kernel_parameter)


```

```
x_mat = table2array(X);
y_mat = table2array(Y);

% Check for missing values and remove before analysis to avoid NaN matrices
% (e.g., relevant for h=12 and h=24)
nr_nans = sum(isnan(y_mat));
ind = nr_nans + 1;

x_mat = x_mat(ind:end,:);
y_mat = y_mat(ind:end,:);

% Standardize each variable prior to performing KPLS
for i = 1:size(x_mat,2)
    x_mat(:,i) = ( x_mat(:,i) - nanmean(x_mat(:,i)) ) / nanstd(x_mat(:,i));
end
y_mat = ( y_mat - nanmean(y_mat) ) / nanstd(y_mat);


% Construct the "kernel matrix":
T = size(x_mat,1);
K = NaN(T,T);

for i = 1:T
    for j = 1:T
        vec_i = x_mat(i,:)'; % transpose i-th obs. vec. to get column vector
        vec_j = x_mat(j,:)'; % transpose j-th obs. vec. to get column vector

        if isequal(kernel_name, 'polynomial')
            K(i,j) = polynomial_kernel(vec_i,vec_j,kernel_parameter);

        elseif isequal(kernel_name, 'radial')
            K(i,j) = gaussian_radial_basis_kernel(vec_i,vec_j,kernel_parameter);

        else
            msg = ('Invalid Kernel name.');
            error(msg)

        end

    end
end


% Centralize the mapped data in the feature space using the procedure
% explained in Rosipal and Clancy (2002):
I = eye(T);
mm = I - (1/T)*ones(T,1)*ones(1,T);
K = mm*K*mm;


% Construct the KPLS component
M = K*y_mat*(y_mat');
[V,D] = eig(M, 'vector'); % compute eigenvalues and eigenvectors of M
[~,max_ind] = max(D); % find position of the largest eigenvalue
lambda = V(:,max_ind); % eigenvector corresponding to largest eigenvalue of M
F = lambda;

F = [NaN(nr_nans,1);F]; % add back NaN values, if any in the initial target vector (relevant for h=12 and h=24)

factors = timetable(dates,F);

end
```

Listing 7: Function that evaluates the polynomial kernel function

```
function [out] = polynomial_kernel(vector1,vector2,sigma)
% This function computes the output of the polynomial kernel for two input
% vectors, 'vector1' and 'vector2', given the tuning parameter sigma (scalar).
% Note: 'vector1' and 'vector2' should be both given as column vectors.

out = ( (vector1/sigma)'*(vector2/sigma) + 1 )^2;

end
```

Listing 8: Function that evaluates the Gaussian radial-basis kernel function

```
function [out] = gaussian_radial_basis_kernel(vector1,vector2,sigma)
% This function computes the output of the radial basis kernel for two input
% vectors, 'vector1' and 'vector2', given the tuning parameter sigma (scalar).
% Note: 'vector1' and 'vector2' should be both given as column vectors.

a = ( norm(vector1/sigma - vector2/sigma) )^2;

out = exp( - a/2 );

end
```

Listing 9: Function that performs the Diebold-Mariano test

```
function [test_statistic,p_val] = diebold_mariano_test(sq_prediction_errors_1, sq_prediction_errors_2)
% Diebold-Mariano test statistic and its corresponding p-value.

d = sq_prediction_errors_1 - sq_prediction_errors_2;
T = size(d,1);

M = round(T^(1/3));

[autocorrelations] = autocorr(d, 'NumLags', M);
sum_autocorrelations = 2*sum(autocorrelations(2:end)) + autocorrelations(1,1);

numerator = mean(d);
denominator = sqrt(sum_autocorrelations/T);
test_statistic = numerator/denominator;

% DM test follows standard normal distribution. Calculate p-value based on
% the cdf of N(0,1):
if numerator >= 0
    p_val = 1 - normcdf(test_statistic,0,1);
else
    p_val = normcdf(test_statistic,0,1);
end

end
```

Listing 10: Function that computes alternative performance measures

```
function [performances] = evaluate_performance_alternative(error_model, error_benchmark)

% 1) mean absolute error
% 2) cubic loss
% 3) asymmetric 1 (cubic loss for negative errors)
% 4) asymmetric 2 (cubic loss for positive errors)

performances = NaN(4,1);

indicator_pos_model = (error_model > 0);
indicator_pos_benchmark = (error_benchmark > 0);


% Cost function 1:
mae_model = mean(abs(error_model));
mae_benchmark = mean(abs(error_benchmark));
performances(1,1) = mae_model/mae_benchmark;

% Cost function 2:
cl_model = mean(abs(error_model).^3);
cl_benchmark = mean(abs(error_benchmark).^3);
performances(2,1) = cl_model/cl_benchmark;

% Cost function 3:
as1_model = mean( indicator_pos_model.*abs(error_model) + (ones(size(indicator_pos_model,1),1) - indicator_pos_model).*(abs(
    error_model).^3) );
as1_benchmark = mean( indicator_pos_benchmark.*abs(error_benchmark) + (ones(size(indicator_pos_benchmark,1),1) -
    indicator_pos_benchmark).*(abs(error_benchmark).^3) );
performances(3,1) = as1_model/as1_benchmark;

% Cost function 4:
as2_model = mean( indicator_pos_model.*(abs(error_model).^3) + (ones(size(indicator_pos_model,1),1) - indicator_pos_model).*(
    abs(error_model)) );
as2_benchmark = mean( indicator_pos_benchmark.*(abs(error_benchmark).^3) + (ones(size(indicator_pos_benchmark,1),1) -
    indicator_pos_benchmark).*(abs(error_benchmark)) );
performances(4,1) = as2_model/as2_benchmark;


% Elliot & Timmermann, 2004 asymmetric loss functions: (1) asymmetric
% squared loss, and (2) piecewise linear loss.

% Asymmetric Squared Loss:
count = 1;
for theta = 0.1:0.1:0.9
    as_squared = mean( indicator_pos_model.*(abs(error_model).^2).*(1-theta) + (ones(size(indicator_pos_model,1),1) -
        indicator_pos_model).*(abs(error_model).^2).*theta );
    as_squared_benchmark = mean( indicator_pos_benchmark.*(abs(error_benchmark).^2).*(1-theta) + (ones(size(
        indicator_pos_benchmark,1),1) - indicator_pos_benchmark).*(abs(error_benchmark).^2).*theta );
    performances(4+count,1) = as_squared/as_squared_benchmark;
    count = count + 1;
end

% Piecewise Linear Loss:
count = 1;
for theta = 0.1:0.1:0.9
    as_linear = mean( indicator_pos_model.*(abs(error_model)).*(1-theta) + (ones(size(indicator_pos_model,1),1) -
        indicator_pos_model).*(abs(error_model)).*theta );
    as_linear_benchmark = mean( indicator_pos_benchmark.*(abs(error_benchmark)).*(1-theta) + (ones(size(
        indicator_pos_benchmark,1),1) - indicator_pos_benchmark).*(abs(error_benchmark)).*theta );
    performances(13+count,1) = as_linear/as_linear_benchmark;
```

```
53      count = count + 1;
54  end
55
56  end
```