Dynamic Mixture of Experts Models for Online Prediction

Parfait Munezero¹,²  Parfait Munezero, Mattias Villani³,⁴  and Robert Kohn⁵  
¹Department of Statistics, Stockholm University, Stockholm, Sweden; ²Data Insights Support Team, Ericsson, Stockholm, Sweden; ³Department of Computer and Information Science, Linköping University, Linköping, Sweden; ⁴UNSW Business School, University of New South Wales, Sydney, Australia

ABSTRACT
A mixture of experts models the conditional density of a response variable using a mixture of regression models with covariate-dependent mixture weights. We extend the finite mixture of experts model by allowing the parameters in both the mixture components and the weights to evolve in time by following random walk processes. Inference for time-varying parameters in richly parameterized mixture of experts models is challenging. We propose a sequential Monte Carlo algorithm for online inference based on a tailored proposal distribution built on ideas from linear Bayes methods and the EM algorithm. The method gives a unified treatment for mixtures with time-varying parameters, including the special case of static parameters. We assess the properties of the method on simulated data and on industrial data where the aim is to predict software faults in a continuously upgraded large-scale software project.

1. Introduction
A mixture of experts (ME) model (Jordan and Jacobs 1994) provides a flexible framework for expressing the distribution of a response variable conditional on a set of covariates. It models continuous or discrete response variables using a finite mixture model with covariate-dependent component models and mixture weights. The component models are commonly referred to as the experts and the mixture weights models as gates. Generally, the experts are regression models with the response’s conditional density from the exponential family; see Gormley and Frühwirth-Schnatter (2018) for a concise introduction to ME models and Yuksel, Wilson, and Gader (2012, sec. IX) for an extensive list of application areas.

ME models have been extended in many ways. Hunter and Young (2012) relax the parametric assumption of the component models and propose a semiparametric inference methodology for mixtures of linear regression models. Villani, Kohn, and Nott (2012) extend the component models to density functions outside the exponential family and use Bayesian variable selection in all parts of the model. Jacobs, Peng, and Tanner (1997) propose a Bayesian hierarchical model in which the conditional density of the response is expressed as a mixture of ME components. Rasmussen and Ghahramani (2002) propose an infinite mixture of Gaussian process experts. Zevei, Meir, and Adler (1996) and Carvalho and Tanner (2005a, 2005b) extend the ME model framework to autoregressive time series data where the covariates may include lagged values of the response and they study the properties of the maximum likelihood estimator. Wood, Jiang, and Tanner (2002) propose a Bayesian inference methodology for ME models and allow smoothing spline components. In longitudinal data applications, Muthén and Shedden (1999) consider fixed and random effect covariates in both components and mixture weights. Quiroz and Villani (2013) propose a ME model with static parameters for longitudinal data but where the subjects are allowed to dynamically change mixture components over time.

While many existing ME models discussed in the previous paragraph are very flexible, they are still too restrictive for many applications, in particular when: (i) the data is an irregularly spaced time series or arrives in batches; (ii) the conditional density of the response tends to change over time. Our application to predicting software faults, in Section 4, is one such typical example taken from industry. Software is generally upgraded at irregular times—a week may pass without any release followed by two or three releases in the next week—depending on the bugs reported in previous versions, the complexity of the added features in the release, or business related factors. Also, the distribution of the response variable naturally changes as the software matures: the developers in the project change over time, user behavior changes, and new technologies emerge which demand adaptation. All these issues make standard ME models for time series data impractical.

We extend the class of ME models to dynamic mixture of experts (DME) models to propose an online (real-time) predictive model with time-varying parameters. The new class is particularly suited for unstructured streaming data, but may also be used for equally spaced time series with time-varying distributions. The proposed DME models have the general form of varying-regression coefficient models (Hastie and Tibshirani...
the regression coefficients in both the mixture components and the mixture weights are allowed to vary over time through a latent process. The latent process can be modeled as a discrete first order Markov process with static (Liehr et al. 1999; Kohlmorgen et al. 2000) or time-varying (Wang et al. 2003) transition probabilities. Alternatively, it can be modeled through random walk processes (West, Harrison, and Migon 1985; Fahrmeir, Kneib, and Lang 2004; Fahrmeir and Kneib 2011) as in Section 2.1. The density for the response variable is modeled conditional on the value of the latent state. This allows the model to adapt locally to any abrupt changes through time (Fahrmeir, Kneib, and Lang 2004). Also, it facilitates designing an adaptive online (real-time) Bayesian predictive inference in which the uncertainty is updated sequentially as new data arrive and predictions are computed based on the recently updated posterior density—the online posterior; see Section 3.

Inference in mixture models is challenging. One challenge is the identifiability issue caused by the invariance of the likelihood under permutation of component labels and parameters. Constraints on the regression coefficients have been imposed to identify ME models, see (Jiang and Tanner 1999) and Appendix B, supplementary materials. However, identifiability is of less importance in applications where prediction is the main objective as the predictive distribution is unaffected by label switching (Geweke 2007). Another important issue in mixture models is the number of components. One strand of the literature use nonparametric Bayes approaches with infinite mixtures typically modeled by stick-breaking processes (Hjort et al. 2010), or reversible jump MCMC (Richardson and Green 1997) to obtain the posterior over the number of mixture components. Another line of research uses model comparison methods to select the optimal number of components (Geweke and Keane 2007; Celeux, Frühwirth-Schnatter, and Robert 2019). In addition to these issues, the structure of DME models brings additional challenges. DME are often richly parameterized with time-varying regression coefficients in both the components and the mixture weights. This leads to a high dimensional and complex target posterior that can only be properly explored by carefully designed numerical methods.

We propose online inference based on Sequential Monte Carlo (SMC) methods (Doucet, Godsill, and Andrieu 2000; Del Moral, Doucet, and Jasra 2006) to address these issues. The performance of SMC methods depends on the proposal/importance distribution from which parameters are sampled. Given the complexity of DME models, off-the-shelf SMC methods based on commonly used proposal distributions such as the bootstrap filter will perform poorly. Our main contribution is a proposal distribution tailored to the class of DME models. The potentially high-dimensional regression coefficients influence the conditional density only through the low-dimensional linear predictors transformed by the link functions. This makes it possible to combine the linear Bayes method (West, Harrison, and Migon 1985) and ideas from the expectation and maximization (EM) algorithm principle (Dempster, Laird, and Rubin 1977) to build the proposal distribution efficiently. Our algorithm builds on the marginal particle filter algorithm (Klaas, De Freitas, and Doucet 2005) for online (real-time) prediction; offline inference using SMC algorithms can be done following ideas in Munezero (2021), where the proposed methodology can be considered as an integrated part of the particle smoother.

Standard inference for mixture models requires posterior draws of component allocation indicators or, in the approach of Carvalho et al. (2010), to keep track of the number of allocations for each component and conditional sufficient statistics for the mixture component parameters. Our methodology expresses the likelihood in its marginal form, which does not require sampling component indicators, hence, reducing considerably the dimension of the target posterior. The posterior’s dimension is reduced even further by using the discount factor approach (West, Harrison, and Migon 1985) that recursively estimates the innovation variance of the states at a particular time point as a function of a discount factor $0 < \alpha < 1$ and the filtered information from the most recent posterior. This means that the inference requires keeping track of only the regression coefficients through time.

The smoothness of the regression coefficients evolution is controlled via $\alpha$ which allows building static and dynamic models in a unified way just by changing the value of $\alpha$. Liu and West (2001) suggest that models with $0.95 \leq \alpha < 1$ are essentially static, and those with $\alpha < 0.95$ are dynamic, although this rule of thumb may vary depending on applications. Inference on the discount factor and the number of components is performed using the log predictive score, a marginal likelihood-based model comparison criteria (Geweke and Keane 2007; Villani, Kohn, and Giordani 2009).

The proposed methodology allows writing general computer code where a user can easily add a new model by supplying the first and second derivative of the component densities with respect to the linear predictors, which are the only key arguments of our procedure. Recent advances in automatic differentiation (Baydin et al. 2018) even removes the requirement of computing derivatives analytically.

The rest of the article is organized as follows. Section 2 introduces the dynamic mixture of experts model and the prior process. Section 3 presents the SMC algorithm based on a proposal distribution from linear Bayes theory. Section 4 presents an industrial application to online prediction of faults in a large-scale software project, where allowing parameters to evolve over time considerably improves predictive performance. Section 5 explores the properties of the inference method on simulated data. The final section concludes.

2. Dynamic Mixture of Experts

Let $D_j = (y_j, \bar{x}_j)$ represent data from a time dependent process observed at different time points $j = 1, \ldots, J$, where $y_j$ denotes the univariate response variable and $\bar{x}_j = (\bar{x}_j^{(1)}, \ldots, \bar{x}_j^{(P)})^\top$ is a $P$-dimensional covariate vector. The $D_j$ may contain only one observation as in standard time series applications, or it may be a data batch containing several observations as in the software upgrade process described in Section 4. We propose the dynamic mixture of experts model

$$f_j (y_j|\bar{x}_j, \omega_j, \lambda_j) = \sum_{k=1}^{K} \omega_{jk} (z_j) f_{jk} (y_j|\lambda_{jk} (x_j)),$$  (2.1)
for online (real time) prediction of $y_j$ given the value of the covariate $x_j$; $z_j$ and $x_j$ are subsets of $x_j$ of dimensions $Q$ and $P$, respectively. The $\lambda_{jk}(x_j)$ and $\omega_{jk}(z_j)$, $k = 1, \ldots, K$, are time-varying covariate-dependent parameter and mixture weight functions of the $k$th expert model, respectively, $\lambda_{j} = (\lambda_{j1}(x_j), \ldots, \lambda_{jk}(x_j))$ and $\omega_{j} = (\omega_{j1}(z_j), \ldots, \omega_{jk}(z_j))$. The covariates in the mixture weights can be distinct from the covariates in the experts.

Here, the experts represent the component models in the mixture (2.1) and depend on the structure of the response variable; they are typically density functions from the exponential family, for example, Gaussian if $y_j$ is continuous, or Poisson, binomial or negative binomial for count data, or multinomial if $y_j$ is categorical. However, as in Villani, Kohn, and Nott (2012), we allow the component models to be any well-behaved density functions, not necessarily limited to the exponential family, and the model parameter may be multidimensional with each of its components connected to the covariates through its own link function. By well-behaved densities we mean densities that are twice differentiable with respect to the parameters, and that satisfies the nondegeneracy condition in Jiang and Tanner (1999, Condition 1) so that the mixture of experts model is identified. Jiang and Tanner (1999) show that this condition is fulfilled for Poisson components and Section B in the Appendix, supplementary materials give a similar result for generalized Poisson components, which do not belong to the exponential family and are used in the empirical application in Section 4.

The component model parameters $\lambda_{jk} = \lambda_{jk}(x_j)$, $k = 1, \ldots, K$ are connected to their linear predictors through a link function $g$ as

$$
\eta_{jk} = g(\lambda_{jk}) = x_j^\top \beta_{jk},
$$

where $x_j = (1, x_j^{(1)}, \ldots, x_j^{(P)})^\top$ and $\beta_{jk} = (\beta_{jk}^{(0)}, \ldots, \beta_{jk}^{(P)})^\top$. For component models with more than one parameter, Equation (2.1) can be extended by linking each parameter to its own linear predictor; see Villani, Kohn, and Nott (2012). Furthermore, the mixture weights depend on the covariate $z_j$, through the multinomial logit link function

$$
\omega_{jk} = \frac{\exp(\psi_{jk})}{1 + \sum_{j=2}^P \exp(\psi_{jk})},
$$

with

$$
\psi_{jk} = z_j^\top \theta_{jk}, \quad k = 2, \ldots, K
$$

where $z_j = (1, z_j^{(1)}, \ldots, z_j^{(Q)})^\top$ and $\theta_{jk} = (\theta_{jk}^{(0)}, \ldots, \theta_{jk}^{(Q)})^\top$. Following standard practice we set $\psi_{j1} = 0$ for all $j$ in (2.3) to identify the model (Jiang and Tanner 1999, Remark 1). In the following, we refer to $\beta_{jk}$ and $\theta_{jk}$ as the regression coefficients in the component distributions and mixture weights, respectively, and to $\lambda_j = (\lambda_{j1}, \ldots, \lambda_{jK})$ and $\omega_j = (\omega_{j2}, \ldots, \omega_{jK})$ as the model parameters.

Mixture models are well-known to suffer from label switching, that is, invariance under permutations of the components. A common approach is to impose order restrictions on parameters of the mixture components Jiang and Tanner (1999), either before running MCMC or SMC to sample from the posterior, or by reordering the posterior draws after the sampling (Stephens 2000). Alternatively, in the Bayesian framework, the identifiability problem is addressed by designing informative priors (Malsiner-Walli, Frühwirth-Schnatter, and Grün 2017). However, our interest here is on predictive inference, where label switching is not a concern (Geweke 2007).

### 2.1. Prior Process

To simplify notation, we stack all the regression coefficients at time $j$ into one vector $y_j$ = ($\beta_j^1, \theta_j^1)^\top$, where $\beta_j = (\beta_j^1, \ldots, \beta_j^K)$ and $\theta_j = (\theta_j^2, \ldots, \theta_j^K)^\top$, and the linear predictors for all components into $\rho_j = (\eta_j^1, \psi_j^1)^\top$.

The prior for the $y_j$ is a random walk

$$
y_j = y_{j-1} + \epsilon_j, \quad \epsilon_j \sim N(0, U_j), \quad (2.5)
$$

with a predefined initial distribution $p(y_1)$, which allows it to vary over time. This prior process is commonly applied in dynamic models as a way of penalizing the regression coefficients from high fluctuations and avoiding overfitting (Fahrmeir, Kneib, and Lang 2004; Fahrmeir and Kneib 2011). In some applications it is sufficient to set $U_j = U$, which is a special case of (2.5). However, it is more useful for online inference to let $U_j$ change over time as it allows to update the prior with historic data recursively as more data batches are observed.

Fully Bayesian inference requires a prior for each $U_j$. Common priors are: (i) an inverse-Wishart density for a full matrix $U_j$ (Gamerman 1998), (ii) an inverse-gamma density (Fahrmeir, Kneib, and Lang 2004) or a random walk process (Lang, Frönk, and Fahrmeir 2002) on the elements of a diagonal $U_j$. An alternative to placing a prior on each $U_j$ is to approximate each $U_j$ recursively using the discount factor approach in West, Harrison, and Migon (1985). Let $C_j$ denote the posterior covariance of $y_j$ and set $U_j = (\alpha^{-1} - 1)C_j^{-1}$ for a given discount factor $0 < \alpha < 1$. A value of $\alpha$ close to one shrinks $U_j$ toward zero, leading to very little variation in $y_j$ over time; a value of $\alpha$ close to zero gives the regression parameters more flexibility and allows the model to adapt well to local fluctuations in the parameter; for instance, change points or level shifts in the parameter.

The discount factor approach has some advantages compared to a fully Bayesian approach. It is computationally much quicker as it avoids extra simulations from the posterior of $U_j$. The discount factor conveniently controls the smoothness of the parameter evolution through time with a single parameter, and it allows building static and dynamic models in a unified way just by changing the value of $\alpha$. Following Liu and West (2001), models with $0.95 \leq \alpha < 1$ are essentially static, and those with $\alpha < 0.95$ are dynamic. We use this approach in Sections 4 and 5.

Our inference methodology applies also to the case of a fully Bayesian approach where $U_j$ is estimated in an additional step using particle Markov chain Monte Carlo (Andrieu, Doucet, and Holenstein 2010) and SMC2 (Chopin, Jacob, and Papaspiliopoulos 2013) methods which allow inference in models with both fixed and time-varying (latent) parameters. It can also be used in the online parameter learning methodology of Carvalho et al. (2010).
3. Inference, Prediction, and Model Comparison

The state space model in Section 2 enables us to exploit the vast literature (Gordon, Salmond, and Smith 1993; Pitt and Shephard 1999; Doucet, Godsill, and Andrieu 2000; Klaas, De Freitas, and Doucet 2005; Doucet, Briers, and Sénécal 2006; Doucet and Johansen 2009) available on sequential Monte Carlo (SMC). SMC methods are particularly appropriate for sequential Monte Carlo (SMC) models with simple proposal distributions like the bootstrap filter (Gordon, Salmond, and Smith 1993) that tailors the true posterior, which is crucial for particle methods in high-dimensional parameter spaces.

3.1. The Marginal Particle Filter Approximation of the Online Posterior Distribution

The target density is the online posterior \( p(\gamma_j | D_{1:j}) \) updated sequentially in time using a prediction step

\[
p(\gamma_j | D_{1:j-1}) = \int p(\gamma_j | \gamma_{j-1}) p(\gamma_{j-1} | D_{1:j-1}) d\gamma_{j-1},
\]  

followed by a measurement update step using Bayes’ theorem

\[
p(\gamma_j | D_{1:j}) \propto f_j(\gamma_j | \tilde{x}_j, \gamma_{j-1}) p(\gamma_{j-1} | D_{1:j-1}),
\]  

(3.1)

(3.2)

to make prior-to-posterior updates. The function \( f_j(\cdot) \) is the response density defined in (2.1), \( D_{1:j} \) denotes the data observed until time \( j \), and \( p(\gamma_j | D_{1:j-1}) \) is the prior updated with all historic data observed before the data batch \( D_j \). Note that contrary to (2.1), now \( f_j(\cdot) \) is parametrized in terms of only the regression coefficients \( \gamma_j \); this is because all quantities required in the inference methodology discussed later are expressed in terms of \( \gamma_j \) only.

We are interested in the online predictive distribution \( p(y_j | \tilde{x}_j, y_{1:j-1}) \) which only depends on the filtering density up to time \( j-1 \) (Doucet, Godsill, and Andrieu 2000). However, the challenging part of the sequential inference in (3.1)–(3.2) is that the integral in (3.1) is only tractable for linear Gaussian models (West, Harrison, and Migon 1985; Gordon, Salmond, and Smith 1993). One way to sample from (3.2) is to use a particle filter algorithm. The particle filter is very attractive for real-time predictions; it allows to sample from intractable distributions and it does not require a scan of the full dataset every time a new observation becomes available.

We use the marginal particle filter of Klaas, De Freitas, and Doucet (2005) to generate a set of particles \( \{\gamma_j^m \}_{m=1}^M \) associated with the importance weights \( \{w_j^m \}_{m=1}^M \). Given the particle sample, any posterior expectation

\[
E(h(\gamma_j)) = \int h(\gamma_j)p(\gamma_j | D_{1:j})d\gamma_j
\]  

(3.3)
is approximated sequentially by

\[
\hat{E}(h(\gamma_j)) = \frac{\sum_{m=1}^M h(\gamma_j^m)w_j^m}{\sum_{m=1}^M w_j^m}.
\]  

(3.4)

The estimator \( (3.4) \) converges to \( E(h(\gamma_j)) \) as \( M \to \infty \) under some weak assumptions stated in Geweke (1989); see Doucet, De Freitas, and Gordon (2001), and Chopin (2004) for more results on the convergence of the particle filter.

The marginal particle filter proposes particles from the proposal distribution \( q(\gamma_j | D_{1:j}) \) and computes the importance weights as

\[
w_j^m \propto f_j(\gamma_j | \tilde{x}_j, \gamma_j^m) \sum_{h=1}^M w_{j-1}^h p(\gamma_j^m | \gamma_{j-1}^h) q(\gamma_j^m | D_{1:j}).
\]  

(3.5)

The estimation of the density in (3.1) follows Equation (3.4); that is,

\[
\hat{p}(\gamma_j | D_{1:j-1}) = \frac{\sum_{m=1}^M w_j^m p(\gamma_j | \gamma_j^m)}{\sum_{m=1}^M w_j^m}.
\]  

(3.6)

Notice that the importance weights in (3.5) depend on the likelihood \( f_j(\gamma_j | \tilde{x}_j, y_j) \) expressed using the covariates and the regression coefficients rather than the model parameters as in (2.1); the inference requires keeping track of the regression coefficients only.

Clearly the proposal density plays an important role. A proposal density which is inconsistent with the target posterior may lead to the particle degeneracy: The importance weights of only a few particles tend to be substantially different from zero, leading to very few effective samples. To mitigate this degeneracy issue, particles with low weights are discarded and replaced by copies of the particles with high weights. Various strategies for resampling particles are available in the literature (Gordon, Salmond, and Smith 1993; Liu and Chen 1998; Carpenter, Clifford, and Fearnhead 1999; Fearnhead and Clifford 2003) and Douc and Cappé (2005) compare some of these resampling schemes.

In the next section we use linear Bayes methods (West, Harrison, and Migon 1985) to construct a proposal \( q(\gamma_j | D_{1:j}) \) that is tailored to the true posterior, which is crucial for particle methods in high-dimensional parameter spaces.

3.2. A Computationally Fast Proposal Distribution for High-Dimensional Marginal Particle Filters

West, Harrison, and Migon (1985) develop a linear Bayes method (Goldstein and Wooff 2007) for dynamic generalized linear models with recursions for the posterior mean and covariance over time, making no assumptions on the distributional form of the posterior. Migon et al. (2013) use these recursive moments to design a multi-move proposal for MCMC targeting the joint smoothing posterior in dynamic generalized linear models. We combine the linear Bayes method in West, Harrison, and Migon (1985) with ideas from the EM algorithm Bishop (2006, chap. 9) to design a proposal distribution \( q(\gamma_j | D_{1:j}) \)
targeting the filtering density \( p(\mathbf{y}_j | D_{tj}) \) in dynamic mixture of experts models. The proposed method allows general mixture components outside the exponential family with any twice differentiable link function.

Similar to eq. (2.8) in West, Harrison, and Migon (1985), we can write the joint posterior of the regression coefficients and the linear predictors as

\[
p(\mathbf{y}_j, \rho_j | D_{tj}) = p(\rho_j | D_{tj}) p(\mathbf{y}_j | \rho_j, D_{tj-1}), \tag{3.7}
\]

where we recall that \( \rho_j = (\eta_j^\top, \psi_j^\top)^\top \) contains the linear predictors in all components and mixture weights. The second factor in (3.7) does not condition on \( D_t \) since \( \mathbf{y}_j \) only enters the likelihood function through the scalar-valued linear predictors in each component, \( \eta_{jk} = \mathbf{x}_j^\top \beta_{jk} \) and \( \psi_{jk} = \mathbf{z}_j^\top \theta_{jk} \) for \( k = 1, \ldots, K \). Our proposal is tailored to the posterior \( p(\mathbf{y}_j | D_{tj}) \) by using the following steps:

1. Approximate the prior \( p(\mathbf{y}_j | D_{tj-1}) \) using a Gaussian with mean and covariance computed from particles at time \( j-1 \).
2. Obtain the second factor in (3.7) by conditioning \( p(\mathbf{y}_j | D_{tj-1}) \) on the linear restrictions \( \rho_j \).
3. Propose from \( q(\mathbf{y}_j | D_{tj}) = N(\mu_j, H_j) \), where \( \mu_j \) and \( H_j \) are obtained from the law of iterated expectation and law of total variance on (3.7) using a Gaussian approximation of \( p(\rho_j | D_{tj}) \).

To give the details of the three steps, define \( \eta_j := X_j \beta_j \), where \( X_j = I_K \otimes x_j^\top \) and \( \psi_j := Z_j \theta_j \), where \( Z_j = I_K \otimes z_j^\top \); hence, we can compactly write \( \rho_j = W_j \mathbf{y}_j \) where \( \mathbf{y}_j = (\beta_j^\top, \theta_j^\top)^\top \), \( \rho_j = (\eta_j^\top, \psi_j^\top)^\top \) and

\[
W_j = \begin{pmatrix} I_K \otimes x_j^\top & 0 \\ 0 & I_K \otimes z_j^\top \end{pmatrix}. 
\]

We can use particles from time step \( j-1 \) to approximate \( \mathbf{y}_j | D_{tj-1} \sim N(\bar{\mathbf{y}}_j, \Sigma_{\mathbf{y}_j}) \), where

\[
\bar{\mathbf{y}}_j = \sum_{m=1}^{M} w_{j-1}^m y_{j-1}^m, \tag{3.8}
\]

\[
\Sigma_{\mathbf{y}_j} = U_j + \sum_{m=1}^{M} w_{j-1}^m \left( \mathbf{y}_{j-1}^m - \bar{\mathbf{y}}_j \right) \left( \mathbf{y}_{j-1}^m - \bar{\mathbf{y}}_j \right)^2,
\]

and then obtain the mean and covariance of the second factor of (3.7) by conditioning this distribution on the linear constraints \( \rho_j = W_j \mathbf{y}_j \) (Rue and Held 2005, eqs. 2.28–2.29) yielding

\[
E \left[ \mathbf{y}_j | \rho_j, D_{tj-1} \right] = \bar{\mathbf{y}}_j + \Sigma_{\mathbf{y_j} \rho_j} \Sigma_{\rho_j}^{-1} (\rho_j - \bar{\rho}_j)
\]

\[
V \left[ \mathbf{y}_j | \rho_j, D_{tj-1} \right] = \Sigma_{\mathbf{y}_j} - \Sigma_{\mathbf{y_j} \rho_j} \Sigma_{\rho_j}^{-1} \Sigma_{\rho_j} \Sigma_{\mathbf{y}_j},
\]

where \( \bar{\rho}_j = W_j \bar{\mathbf{y}}_j \), \( \Sigma_{\rho_j} = W_j \Sigma_{\mathbf{y}_j} W_j^\top \), \( \Sigma_{\mathbf{y_j} \rho_j} = W_j \Sigma_{\mathbf{y}_j} \Sigma_{\mathbf{y}_j} \rho_j = \Sigma_{\mathbf{y}_j} W_j^\top \). Now, the proposal is \( q(\mathbf{y}_j | D_{tj}) = N(\mu_j, H_j) \) with moments obtained from applying the law of iterated expectations and the law of total variance to (3.7),

\[
\mu_j = E_{\rho_j} \left[ E \left[ \mathbf{y}_j | \rho_j, D_{tj-1} \right] | D_{tj} \right] = \bar{\mathbf{y}}_j + \Sigma_{\mathbf{y}_j \rho_j} \Sigma_{\rho_j}^{-1} \left( E_{\rho_j} (\rho_j | D_{tj-1}) - \bar{\rho}_j \right) \tag{3.9}
\]

\[
H_j = E_{\rho_j} \left[ V \left[ \mathbf{y}_j | \rho_j, D_{tj-1} \right] | D_{tj} \right] = \Sigma_{\mathbf{y}_j} - \Sigma_{\mathbf{y}_j \rho_j} \Sigma_{\rho_j}^{-1} \Sigma_{\rho_j} \Sigma_{\mathbf{y}_j}. \tag{3.10}
\]

It remains to compute \( E_{\rho_j} (\rho_j | D_{tj}) \) and \( V_{\rho_j} (\rho_j | D_{tj}) \). A second order Taylor expansion of \( \log p(\rho_j | D_{tj}) \) around \( \bar{\rho}_j \) leads to the following approximations (Doucet, Godsill, and Andrieu 2000):

\[
V_{\rho_j} (\rho_j | D_{tj}) = \left[ -\nabla \nabla_{\rho_j} \log p(\rho_j | D_{tj}) \bigg|_{\rho_j = \bar{\rho}_j} \right]^{-1}, \tag{3.11}
\]

\[
E_{\rho_j} (\rho_j | D_{tj}) = \bar{\rho}_j + V_{\rho_j} (\rho_j | D_{tj}) \nabla \log p(\rho_j | D_{tj}) \bigg|_{\rho_j = \bar{\rho}_j}.
\]

Letting \( \pi_{jk} = \log \omega_{jk} f_{jk}(y_j | \lambda_{jk}) \), the gradient can be computed by direct calculation

\[
\nabla \nabla_{\rho_j} \log p(\rho_j | D_{tj}) = \sum_{k=1}^{K} \Pr(s_j = k | D_{tj}) \nabla \nabla_{\rho_j} \pi_{jk} = \Sigma_{\rho_j}^{-1} (\rho_j - \bar{\rho}_j),
\]

where \( \Pr(s_j = k | D_{tj}) \propto \omega_{jk} f_{jk}(y_j | D_{tj-1}, \lambda_{jk}) \) are the posterior probabilities of the observation \( y_j \) coming from component \( k \) (see Bishop 2006, chap. 9.3 for similar expressions for the EM algorithm). Similarly, the Hessian is,

\[
\nabla \nabla_{\rho_j} \log p(\rho_j | D_{tj}) = \sum_{k=1}^{K} \Pr(s_j = k | D_{tj}) \nabla \nabla_{\rho_j} \pi_{jk} = \Sigma_{\rho_j}^{-1}.
\]

Note that the component parameters \( \eta_{jk} \) and \( \psi_{jk} \) enter additively in \( \log \omega_{jk} f_{jk}(y_j | \lambda_{jk}) \); therefore, their gradients can be computed separately.

If the batches \( D_t \) contain several observations, then \( \mu_j \) and \( H_j \) can be computed by iterating the procedure described above over the observations in the batch; see Gamerman (1991) for a similar approach. Starting with the first observation, we proceed through the following iterations:

1. Compute \( \mu_j^{(i)} \) and \( H_j^{(i)} \) from Equation (3.9) and Equation (3.10).
2. Set \( \bar{\mathbf{y}}_j = \mu_j^{(i)} \) and \( \Sigma_{\mathbf{y}_j} = H_j^{(i)} \).
3. Return to Step 1 until the last observation in the batch.

### 3.3. Model Comparison and Prediction

Our model depends on the choice of the number of mixture components \( K \) and the discount factor \( \alpha \). We propose to infer these quantities using a sequential version of the marginal likelihood (Doucet, Godsill, and Andrieu 2000)

\[
p(\mathbf{y}_{1:t}) = p(\mathbf{y}_1) \prod_{j=2}^{t} p(\mathbf{y}_j | y_{1:j-1}), \tag{3.12}
\]
where
\[ p(y_j | y_{1:t-1}) = \int f_j(y_j | \bar{x}_j, y_j) \ p(y_j | D_{1:t-1}) \ dy_j. \] (3.13)

Given a sample of \( M \) particles \( \{y_{m}^{j*}\}_{m=1}^{M} \) and the corresponding importance weights \( \{w_{1:m}^{j*}\}_{m=1}^{M} \), the predictive distribution (3.13) is approximated as
\[ \hat{p}(y_j | y_{1:t-1}) = \sum_{m=1}^{M} w_{1:m}^{j*} f_j(y_j | \bar{x}_j, y_j^{m}). \]

where \( y_j^{m} \) are generated from the transition distribution \( p(y_j | y_{1:t-1}) \). Different predictive scores are defined as functions of (3.12). One particular example is the log predictive score
\[ \text{LPS} = \sum_{j=1}^{J} \log \hat{p}(y_j | y_{1:t-1}), \]
where \( J^* \in \{1, \ldots, J\} \). The LPS is generally sensitive to the initial distribution of the parameters (Villani, Kohn, and Giordani 2009). We therefore use the last \( l/2 \) data batches to compute the LPS for the models in Section 4; that is, \( J^* = l/2 \). We assume that the particle approximation to the marginal likelihood should be stable after \( j = l/2 \). Computing the LPS for different combinations of the number of mixture components \( K \) and the discount factor \( \alpha \) makes it possible to select good values for these model specification parameters.

4. Predicting Faults in Large-Scale Software Projects

Large-scale industrial software projects are continually upgraded to fix bugs and/or to add new features. The upgrades are generally at irregular times: in one week we may observe one release, and in the next, two or three releases depending on several factors such as the amount and severity of the bugs reported in previous versions, the complexity of the new features added to the software, and other business-related factors. Other key factors include the human interaction with the software and the technology evolution. The developers, the user behavior and technologies change over time. The dynamic mixture of experts model (2.1) is appropriate in this case. Different mixture components allow us to model the unknown variations/changes in the human interaction with the software and the time-varying parameters enable the model to adapt to the changes over time.

As the response variable is the number of faults \( y_t \) reported on the upgrade created at time \( t \), we propose a dynamic mixture of Poisson experts. Here each expert is a Poisson regression model with a covariate vector \( \bar{x}_j \) selected from six code complexity metrics that measure changes made in the source code. The metrics include: (i) The number of commits (NC) which represents the number of modifications done from the previous to the current version, (ii) the number of changed modules (CM), (iii) the number of faults corrected (NFC) per line of code which is the ratio of the total number of faults corrected and the total number of code lines excluding comments, (iv) the proportion of files written in C++ (CF), (v) the proportion of files written in Java (JP), and (vi) the file complexity (FC). The latter is a score calculated based on the number of control flows in the code, for example, if, for and while loop statements.

The aim is to build an online prediction model for the number of faults in a planned upgrade release. We use a software trouble reports dataset from a large-scale project at a major telecom company; the dataset contains a history of 1800 upgrades that were created during a period of 650 days (roughly 21 months). All covariates, excluding the CF, JP and NFC are integers ranging from zero to a value up to six order of magnitude. Therefore, to reduce the scale variations, we apply the log(1 + \( \bar{x}_j \)) transformation to the integer complexity metrics; after this transformation the highest value is no greater than 15.

To make it tractable to deal with the irregular times of fault reports, we partition time into short contiguous intervals \([\tau_0, \tau_1], [\tau_1, \tau_2], \ldots, [\tau_{j-1}, \tau_j]\), where \( \tau_0 = \min(t) < \tau_1 < \cdots < \tau_{j-1} < \tau_j = \max(t) \). The partition of time induces a partition of the original data into a sequence of batches \( D_j = \{y_j, \bar{X}_j\} \) which collect data for all upgrade packages created within the time interval \( t \in [\tau_{j-1}, \tau_j], j = 1, \ldots, J \). Batch \( D_j \) contains \( N_j \) data points, where \( y_{j*} = (y_{j1}, \ldots, y_{jN_j})^\top \) is a vector of the response observations in the batch and \( \bar{X}_j = (\bar{x}_{j1}, \ldots, \bar{x}_{jN_j})^\top \) is a vector of covariates \( \bar{x}_j \) for the data point \( i = 1, \ldots, N_j \). Figure 1 illustrates this data partition.

The time is partitioned into 30 days-long intervals, which leads to 21 intervals in total. Experimentation with intervals lengths of one week, two weeks and three months did not improve the LPS. We also assume the initial distribution \( y_1 \sim N(0, I) \), where \( I \) is the unit diagonal matrix.

Table 1 compares different fitted models based on their LPS. The table displays various dynamic models, with discount factor \( \alpha = 0.5 \), and their static versions, where \( \alpha = 0.99 \). The models in the table have different variables in the component models and the number of commits (NC) as the only covariate \( z \) in the mixture weights. To select \( z \), we fix the covariates in the component models to \( \tilde{X} \) and, starting from \( z = \tilde{X} \), we eliminate variables in \( z \) systematically based on the LPS.

Table 1 shows that dynamic models outperform static models, with a difference in LPS more that 40 for single component models and well above 100 for several of the multicomponent models. Also, there is a very large jump in LPS when going from one to two components, in particular for the dynamic versions. While two components seem to be sufficient for the dynamic models, the static models require more components and covariates. The dynamic model CM+FC with two components seems to perform well in terms of LPS since adding more complexity gives no significant increase in LPS. This model is therefore selected for further analysis.

To illustrate that our algorithm can be used also for models outside the exponential family we fit a single component dynamic generalized Poisson model (Famoye and Singh 2006) using CM and FC as covariates in both the mean and dispersion functions. Figure 2 displays the predictive distribution for the one and two-component versions of the selected CM+FC dynamic model, and the dynamic generalized Poisson model at three time points: \( j = 2, j = 10 \) and \( j = 21 \). The predictive distribution at the time point \( j \) is constructed using the posterior at the previous interval \( j - 1 \) and the batch \( D_j \) as test set.
Figure 1. (a) The upgrading process. An upgrade package (UP) at time $t$ is created by making $x_t$ changes on the previous version of the software (created at time $t-1$) and $y_t$ faults are reported on the version created at time $t$. (b) Process of grouping upgrades according to intervals partitioning the training time.

Table 1. LPS for different models fitted to the software trouble reports data.

| Component model       | Type   | 1       | 2       | 3       | 4       |
|-----------------------|--------|---------|---------|---------|---------|
| CM                    | Dynamic| -1550.71| -1192.66| -1179.44| -1173.35|
| CM + FC               | Dynamic| -1539.21| -1168.96| -1160.85| -1160.73|
| CM + FC + NC          | Dynamic| -1543.58| -1187.99| -1170.84| -1278.04|
| CM + FC + NC + NFC    | Dynamic| -1543.03| -1212.97| -1247.81| -1290.10|
| CM + FC + NC + NFC + JF| Dynamic| -1534.11| -1250.82| 1288.74 | -1272.16 |

NOTE: Results are based on a posterior sample of 100,000 particles. Bold values indicate results for dynamic models.

It is clear from Figure 2 that the distribution of the number of faults varies over time; there is a very large shift of probability mass toward a smaller number of faults as time evolve. The two-component CM + FC dynamic model adapts well to the dynamic variations in the data and gives very impressive predictions on the test data, while the one-component version does not perform well, agreeing with the LPS in Table 1. The one-component dynamic generalized Poisson model behaves very similarly to the two-component dynamic Poisson; the LPS of the generalized Poisson model is $-1189$.

To investigate the efficiency of the proposed SMC inference methodology, we fit the selected two-component CM + FC dynamic model using a particle filter with 1000 particles. This is at least an order of magnitude smaller than what can easily be afforded in real applications, but is used here to investigate how much the inferred predictive distribution varies over 100 independent runs with different seeds. Figure 3 shows that this variability is small; the figure also includes the predictive distribution from a single run with 100,000 particles to represent the ground truth. This shows that the proposed method is very efficient and even a small number of particles gives an adequate numerical precision for most applications.

5. Simulation Study

We perform several simulation experiments to study the performance of the proposed inference methodology on data
Figure 2. The evolution of the predictive distribution of one-component and two-components of the CM+FC dynamic Poisson ME model and the one-component dynamic generalized Poisson model fitted to the software fault data at three time points \( j = 2, j = 10, \) and \( j = 21. \)

Figure 3. Illustrating the efficiency of the proposed inference methodology. The shaded area represents the 95\% intervals of the predictions from 100 independent iterations of the particle filter with 1000 particles. The orange curve represents the prediction obtained from a particle filter with 100,000 particles.

generated from both static and dynamic data-generating processes (DGPs).

5.1. Simulation Experiments

The simulation experiments simulate data from the five data generating processes summarized in Table 2. Models M1, M3, and M3 all assume a batch data structure, which is common in industrial applications, where data are observed at irregular time points and aggregated into batches. The only source of time variation is in the parameter evolution; the parameter stays constant within a batch but may change across batches. Models M4 and M5 are mixtures of autoregressive experts (Carvalho and Tanner 2005b, 2007) with constant (M4) and time-varying parameters (M5). The latter two models are pure time series models with one data point observed at equidistant times, and where the response depends on its lagged values. We follow Carvalho and Tanner (2007) and use \( \log(y_{t-1} + 1) \) as lagged values, and parametrizing the autoregressive parameter as \( 1 - \exp(\beta) \), so that \( \beta \) can vary freely while at the same time ensuring that the process is stationary for every parameter value.

For each DGP, 50 datasets of 1000 observations are generated. For M1 to M3, data are generated sequentially over 10 time intervals, having 100 observations within each interval. For M4 and M5 a time series of length 1000 is generated. The first half of the data is used for training and the last for validation; all model comparisons are based on LPS values computed on the test set—the last half of generated data.

5.2. Inference of the Number of Mixture Components and the Discount Factor

The number of mixture components/experts \( K \) and the discount factor \( \alpha \) discussed in Section 2 are unknown. Inference of these quantities is a research area. Here, we use LPS to assess the performance of the proposed methodology on the inference of
these quantities. Several models with \( k = 1, 2, 3 \) Poisson components and \( \alpha \in \{0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.99\} \) are fitted to data generated from each of the models \( M_1, M_2, \) and \( M_3 \) DGP, and LPS is used to select the best model. The aim is to see if the proposed inference methodology is able to identify the underlying data generating process. Figure 4 displays the selection frequency of \( K \) and \( \alpha \) for all fitted models.

For \( M_4 \) and \( M_5 \), the most frequently selected model is the single component Poisson model with \( \alpha = 0.99 \), and \( \alpha = 0.4 \), respectively. While, for \( M_3 \), it is the model with \( \alpha = 0.6 \), and not the correct two components mixture model. This slight overestimation is not surprising as LPS is often observed to have a tendency to be generous with the number of components in a mixture without having a large impact on the final predictive density, see for example, Villani, Kohn, and Nott (2012).

### 5.3. Comparing Static and Dynamic Models

The data-generating process is generally unknown in real applications and the usual strategy in modeling the data is to fit static models. It is therefore interesting to evaluate how fitting a dynamic model would differ from its static version in the cases where the true data-generating process is static or dynamic.

We first consider the \( M_1, M_2, \) and \( M_3 \) data generating processes. Figure 5 compares the performance of the model (i) with \( K = K_{opt} \) and \( \alpha = \alpha_{opt} \), where \( K_{opt} \) and \( \alpha_{opt} \) are the values chosen from LPS and (ii) the corresponding static model with \( K = K_{opt} \) and \( \alpha = 0.99 \).

Figure 5 shows boxplots of the difference in the LPS values in the validation set for both models. For \( M_1 \) the average LPS difference between the selected and the static models is around zero, which shows that the dynamic model does not overfit on static data. On the other hand, for the two dynamic data generating processes, \( M_2 \) and \( M_3 \), the dynamic model selected in the validation step clearly outperforms the static model and the difference in LPS increases with the number of components.

Consider now the mixture of autoregressive Poisson experts models, \( M_4 \) and \( M_5 \). Static models are fitted using the MCMC algorithm in Villani, Kohn, and Nott (2012) with 10,000 MCMC iterations and dynamic models are fitted using 1000 particles. We partition the data into batches when running our algorithm, which also allows us to investigate the effect of the chosen batch size. Three different batch sizes are compared: 10, 25, and 50. The MCMC algorithm is also updated sequentially at each batch for comparability and for reducing computing times. The dynamic models are trained with different discount factors \( \alpha \in \{0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.99\} \) and the LPS is used to select the best model in the validation step.

Figure 6 displays boxplots of the LPS difference of the selected dynamic model and the static model. The average LPS difference between the selected dynamic model and the static model is around zero, which again shows that the dynamic model does not overfit on static data. On the other hand, for the two dynamic data generating processes, the dynamic model clearly outperforms the static model. Also, one can note that the data partition has minimal effect as there is not much variation in the LPS differences for the different batch partitions.

### 5.4. Efficiency of the Linear Bayes Proposal

The efficiency of particle filter algorithms is generally assessed based on the effective sample size

\[
ESS_j := \frac{1}{\sum_{m=1}^{M}(w_j^m)^2},
\]

where \( w_j^m \) are the importance weights computed at interval \( j \).

To assess the performance of the linear Bayes proposal strategy, we compare it with the local linearization proposal strategy (Doucet, Godsill, and Andrieu 2000); the state-of-the-art method of constructing proposal densities which approximates the target density (3.2) by a linear Gaussian distribution obtained from a second order Taylor expansion of the target density with respect to the regression coefficients \( \gamma_j \), for \( j \in \{1, \ldots, J\} \). The main difference between these two methods is that the local linearization method approximates the target density by a Gaussian density without the intermediate step of updating the linear predictors.

Figure 7 compares the effective sample size per second generated by the two strategies as a way of comparing their efficiency.
Figure 4. Fitting different models to the data generated from the three $M_1 - M_3$ DGPs. Each panel displays the number of times each model was selected based on the LPS. Results are based on 1000 particles.

Figure 5. Boxplot of the difference in LPS of the model selected in the validation step and the corresponding static model for the first three data-generating processes.

Figure 6. Boxplot of the difference in LPS of the model selected the dynamic model and the corresponding static model.
and computation time. The results are based on data simulated from $M_3$ and a posterior distribution approximated by 1000 particles and $\alpha = 0.5$. The figure shows that the linear Bayes proposal generates an effective sample size that is on average 10% higher than the local linearization. Both methods are quite fast; their computation time on a simple windows laptop with intel core i5 processor is less than 3 CPU min.

6. Conclusions

We introduce a general class of dynamic mixture of experts models for online predictions; the model allows the regression coefficients in each mixture component and weight to vary over time. The component models can be essentially any density function, not necessarily limited to the exponential family.

We propose an efficient SMC algorithm for sequential inference and online prediction that is tailored to handle the proposed model class with potentially high-dimensional parameter spaces. The algorithm handles models with static and dynamic parameters in a unified way.

The model is applied to online prediction of the number of faults in a continuously upgraded large-scale industrial software project. We show that allowing the parameters to evolve over time greatly improves the model's predictive performance. A simulation study documents that the proposed model selection procedure is (i) effective in reducing flexibility when data comes from a static single-component model, (ii) able to fit data from multi-component models with time-varying parameters, and (iii) it is fast and generates an effective sample size rate that is superior to the state-of-the-art particle filter which uses a proposal density designed via the local linearization of the target density.

Supplementary Materials

The electronic supplements include two files: (i) The appendices.pdf file provides details on the expressions of the gradient and the Hessian of the dynamic mixture of experts models and a discussion on the identifiability of the models applied in Section 4. (ii) The dmoef_code.zip includes source code files for reproducing the results in Section 5.

Acknowledgments

We thank the editor, the associate editor, and two referees for their valuable comments and suggestions that helped to improve the article. The author are grateful to the Data Insight team in Ericsson for providing the software fault data.

Disclosure Statement

The authors report that there are no competing interests to declare.

ORCID

Parfait Munezero http://orcid.org/0000-0002-3902-3846
Mattias Villani http://orcid.org/0000-0003-2786-2519
Robert Kohn http://orcid.org/0000-0002-3733-1474

References

Andrieu, C., Doucet, A., and Holenstein, R. (2010), “Particle Markov Chain Monte Carlo Methods,” Journal of the Royal Statistical Society, Series B, 72, 269–342. [259]
Baydin, A. G., Pearlmutter, B. A., Radul, A. A., and Siskind, J. M. (2018), “Automatic Differentiation in Machine Learning: A Survey,” Journal of Machine Learning Research, 18, 1–43. [258]
Bishop, C. M. (2006), Pattern Recognition and Machine Learning, New York: Springer. [260,261]
Carpenter, J., Clifford, P., and Fearnhead, P. (1999), “Improved Particle Filter for Nonlinear Problems,” IEE Proceedings-Radar, Sonar and Navigation, 146, 2–7. [260]
Carvalho, A. X., and Tanner, M. A. (2005a), “Mixtures-of-Experts of Autoregressive Time Series: Asymptotic Normality and Model Specification,” IEEE Transactions on Neural Networks, 16, 39–56. [257]
Carvalho, A. X., and Tanner, M. A. (2005b), “Modeling Nonlinear Time Series with Local Mixtures of Generalized Linear Models,” Canadian Journal of Statistics, 33, 97–113. [257,264]
Carvalho, A. X., and Tanner, M. A. (2007), “Modelling Nonlinear Count Time Series with Local Mixtures of Poisson Autoregressions,” Computational Statistics & Data Analysis, 51, 5266–5294. [264]
Carvalho, C. M., Lopes, H. F., Polson, N. G., and Taddy, M. A. (2010), “Particle Learning for General Mixtures,” Bayesian Analysis, 5, 709–740. [258,259]
Celeux, G., Frühwirth-Schnatter, S., and Robert, C. P. (2019), “Model Selection for Mixture Models—Perspectives and Strategies,” in Handbook of Mixture Analysis, eds. S. Frühwirth-Schnatter, G. Celeux, C. P. Robert, pp. 117–154, New York: Chapman and Hall/CRC. [258]
Chopin, N. (2004), “Central Limit Theorem for Sequential Monte Carlo Methods and Its Application to Bayesian Inference,” The Annals of Statistics, 32, 2385–2411. [260]
Chopin, N., Jacob, P. E., and Papaspiliopoulos, O. (2013), “SMC2: An Efficient Algorithm for Sequential Analysis of State Space Models,” Journal of the Royal Statistical Society, Series B, 75, 397–426. [259]
Del Moral, P., Doucet, A., and Jasra, A. (2006), “Sequential Monte Carlo Samplers,” Journal of the Royal Statistical Society, Series B, 68, 411–436. [258]
 Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977), “Maximum Likelihood from Incomplete Data via the EM Algorithm,” Journal of the Royal Statistical Society, Series B, 39, 1–22. [258]
Douc, R., and Cappé, O. (2005), “Comparison of Resampling Schemes for Particle Filtering,” in Proceedings of the 4th International Symposium on Image and Signal Processing and Analysis, ISPA 2005, pp. 64–69. IEEE. [260]
Doucet, A., Briers, M., and Sénécal, S. (2006), “Efficient Block Sampling Strategies for Sequential Monte Carlo Methods,” Journal of Computational and Graphical Statistics, 15, 693–711. [260]
Doucet, A., De Freitas, N., and Gordon, N. J. (2001), Sequential Monte Carlo Methods in Practice (Vol. 1), Springer. [260]
Doucet, A., Godsill, S., and Andrieu, C. (2000), “On Sequential Monte Carlo Sampling Methods for Bayesian Filtering,” Statistics and Computing, 10, 197–208. [258,260,261,265]
Doucet, A., and Johansen, A. M. (2009), “A Tutorial on Particle Filtering and Smoothing: Fifteen Years Later,” in Handbook of Nonlinear Filteringeds. D. Crisan and B. Rozovskii, pp. 656–704. Oxford: Oxford University Press. [260]
