PARTICLE FLOW FOR PARTICLE FILTERING

Yunpeng Li[⋆]*, Lingling Zhao*[†] *and Mark Coates*[⋆]

[⋆] Dept. of Electrical and Computer Engineering, McGill University, Montréal, Québec, Canada
[†] School of Computer Science and Technology, Harbin Institute of Technology, Harbin, China
E-mail: yunpeng.li@mail.mcgill.ca; zhaoll@hit.edu.cn; mark.coates@mcgill.ca

ABSTRACT

Particle flow algorithms have been developed as an alternative to particle filtering. In these algorithms, there is no importance sampling, and particles are migrated from the prior to the posterior via a "flow", described by differential equations. Aside from a few special cases, implementations involve multiple approximations, and their impact on the accuracy of the estimates is not clearly understood. In this paper, we propose algorithms that use particle flow procedures to construct an importance sampling distribution within a standard particle filter. The resultant algorithms retain the statistical consistency of sequential Monte Carlo methods, but acquires the desirable properties of particle flow techniques. We report the results of a multiple target tracking simulation study that combines highly informative measurements with a reasonably high-dimensional state space, leading to a challenging scenario for particle filters. Of the filters we test, the particle flow particle filter provides the smallest tracking error and achieves the largest average effective sample size.

Index Terms— Sequential Monte Carlo, Particle Flow, High Dimensional Filtering, Optimal Proposal Distribution

1. INTRODUCTION

Sequential Monte Carlo methods track the posterior distribution in time by updating a set of weighted particles when new data become available. When the state dimension is high or observations are highly informative, it is often the case that almost all particles have extremely small weights after a few timesteps. This particle degeneracy phenomenon, associated with the curse of dimensionality [1], leads to poor representation of the posterior and inaccurate state estimates.

Particle flow algorithms [2, 3] migrate particles from the prior distribution to the posterior distribution, via a "flow" that is specified through a partial differential equation. There is no sampling (or resampling); there is some evidence that the approach of migrating particles can lead to improved filter performance for high-dimensional state spaces, avoiding the severe particle degeneracy.

Daum et al. have derived numerous particle flow algorithms, with differences arising through the assumptions about the dynamic model and the method employed to solve the differential equations [3–5]. An "exact" particle flow solution exists for the case when the prior and posterior distributions are both Gaussian [6] and the observation model is linear. We refer to this algorithm as the exact Daum and Huang filter (EDH), and a detailed description of its implementation is provided in [7]. A computationally intensive variation of EDH that computes the flow for each particle is proposed in [8] and is referred to as the localized exact DH filter (LEDH). In practice, we would like to apply the EDH to non-Gaussian settings with non-linear observation models. This necessitates the introduction of local linearizations and approximations of covariance matrices.

For more general non-Gaussian problems, the non-zero diffusion particle flow filter (NZDDH) models the particle flow with a stochastic differential equation and uses the diffusion term to simplify the Fokker-Planck equation that describes the density evolution [3]. Implementation of this filter also requires approximations to calculate the flow. For all particle flow filters, although the particle flow is expressed as a continuous partial differential equation, implementation requires that we compute it using discrete steps, and this introduces further approximation error.

An alternative method is to use particle flow ideas to generate a proposal distribution within a particle filter. We then retain desirable statistical consistency properties of the particle filter, but can exploit the ability of particle flow procedures to find regions where the posterior is dense. In [9], Bunch and Godsill develop an approximate Gaussian particle flow importance sampling procedure which we refer to as GPFIS. The algorithm performs impressively, but it performs numerous importance weight updates every time step, and each of these involves a computationally expensive solution to a Sylvester equation. The optimal transport methods described in [10] also involve particle migration to derive a proposal distribution, but they involve the design of a complex transport map and this can be challenging for many filtering scenarios. In [11], we proposed the use of particle flow in an auxiliary particle filter framework (PF-APF) to sample auxiliary variables and construct an importance sampling distribution. However, the design of an appropriate proposal distribution can be difficult in high dimensions.

In this paper, we propose a simple way to incorporate particle flow techniques within the standard particle filtering framework. We demonstrate with an example simulation that the approach can improve accuracy and increase the effective sample size. The paper is organized as follows: Section 2 provides the problem statement; section 3 describes the particle flow particle filter; section 4 describes the simulation setup and presents results; and section 5 provides a summary.

2. PROBLEM STATEMENT

Consider the task of nonlinear filtering with the following models:

$$x_k = g(x_{k-1}, v_k) \tag{1}$$
$$z_k = h(x_k, w_k)\,. \tag{2}$$

Here the unobserved state x_k evolves according to the dynamic model described by $g()$, z_k is the observation which is linked to x_k through a nonlinear measurement model $h()$. v_k is the process noise and w_k is the measurement noise term. The nonlinear filtering task is to track the marginal posterior distribution $p(x_k|z_{1:k})$, where $z_{1:k} = \{z_1, \ldots, z_k\}$ is a sequence of observations collected up to time step k.

3. PARTICLE FLOW IMPORTANCE SAMPLING

Suppose that at the $(k-1)$-th time step, we have a set of N_p particles $\{x^i_{k-1}\}_{i=1}^{N_p}$ and weights $\{w^i_{k-1}\}_{i=1}^{N_p}$ representing the posterior distribution at time $k-1$. After propagating particles using the dynamic model, we obtain weighted samples $\{\tilde{\mu}^i, w^i_{k-1}\}_{i=1}^{N_p}$ that represent the prior distribution at time k.

The proposed particle flow particle filter consists of two steps. First, the particle flow equations are employed to migrate particles from the prior to the posterior. After applying the mapping, we have generated a set of weighted particles $\{\mu^i, w^i_{k-1}\}_{i=1}^{N_p}$. The proposed particle flow particle filter algorithm considers the generated particles as being drawn from a proposal distribution. Due to the nature of the particle flow procedure, we anticipate that the proposal distribution is reasonably well matched to the posterior. Our key challenge is evaluating the proposal density $q(x_k|x_{k-1}, z_k)$ at μ^i. We address this by designing particle flow procedures which possess the one-to-one mapping property.

3.1. Exact Particle Flow

We model the particle flow process as a background stochastic process $\mu(\lambda)$ for $\lambda \in [0, 1]$, such that the distribution of $\mu(0)$ is the prior distribution of x_k and the distribution of $\mu(1)$ is the posterior distribution of x_k. For simplicity we drop the parameter λ for μ when the pseudo-time step λ is clearly defined in the context. Discretized pseudo-time integration is needed in the implementation. We set a sequence of discrete steps with possibly varying step sizes $[\Delta\lambda(1), \Delta\lambda(2), \ldots, \Delta\lambda(N_\lambda)]$.

When both the prior and posterior distribution are Gaussian and the observation model is linear, an exact expression for the particle flow can be derived for EDH [6]. The flow is described by the equation:

$$\frac{d\mu}{d\lambda} = \zeta(\mu, \lambda) = A(\lambda)\mu + b(\lambda) \tag{3}$$

where

$$A(\lambda) = -\frac{1}{2}PH^T(\lambda HPH^T + R)^{-1}H, \tag{4}$$
$$b(\lambda) = (I + 2\lambda A)[(I + \lambda A)PH^T R^{-1} z_k + A\bar{\mu}_0]. \tag{5}$$

Here $\bar{\mu}_0$ is the mean of the prior distribution. P is the covariance matrix of the prediction error for the prior distribution, which can be estimated by the sample covariance matrix, or through an extended or unscented Kalman filter (EKF/UKF). For nonlinear models, H is the linearization of the measurement model , i.e. $H = \dfrac{\partial h(\mu, 0)}{\partial \mu}$. R is the covariance matrix of the measurement error. Pseudocodes of two typical algorithms in this class, the EDH [7] and the LEDH, are both presented in [8]. In LEDH, the slope $A^i(\lambda)$ and the offset $b^i(\lambda)$ of the drift term $\zeta(\mu^i, \lambda) = A^i(\lambda)\mu^i + b^i(\lambda)$ are calculated for each particle μ^i.

3.2. Proposal Density Evaluation

As discussed above, the critical step is the evaluation of the proposal density for a given particle. After the discretized particle flow process, we can view the migrated particle μ^i as being drawn from a proposal distribution $q(\mu^i|x^i_{k-1}, z_k)$. If the function constructed by the discretized particle flow, $\mu^i = T(\tilde{\mu}^i)$ is one-to-one (injective), then we can evaluate the proposal density as follows:

$$\begin{aligned} q(\mu^i|x^i_{k-1}, z_k) &= p(\tilde{\mu}^i|x^i_{k-1}, z_k) \\ &= p(\tilde{\mu}^i|x^i_{k-1}) \end{aligned} \tag{6}$$

The first equation holds because of the one-to-one mapping between $\tilde{\mu}^i$ and μ^i. The second equation holds because $\tilde{\mu}^i$ is generated solely from the dynamic model.

The proposal density evaluation is a one-step calculation of the probability density based on the dynamic model, which adds negligible computational cost to the particle flow algorithm. We can then evaluate the importance weights of each particle as

$$w^i_k = \frac{p(\mu^i|x^i_{k-1})p(z_k|\mu^i)}{p(\tilde{\mu}^i|x^i_{k-1})} w^i_{k-1} \tag{7}$$

The pseudocode of the particle flow particle filter algorithm (PF-PF) based on EDH is presented in Algorithm 1. And the algorithm based on LEDH is shown in Algorithm 2.

Algorithm 1: Particle flow particle filtering (EDH).

1: Initialization: Draw $\{x_0^i\}_{i=1}^{N_p}$ from the prior $p_0(x)$;
2: Set $\{w_0^i\}_{i=1}^{N_p} = \frac{1}{N_p}$;
3: **for** k = 1 to T **do**
4: Estimate $\bar{\mu}$ and P using the sample mean and the sample covariance matrix, EKF, or UKF;
5: **for** $i = 1, \ldots, N_p$ **do**
6: Propagate particles $\tilde{\mu}^i = g(x_{k-1}^i, v_k)$;
7: Set $\mu^i = \tilde{\mu}^i$;
8: **end for**
9: Set $\lambda = 0$;
10: **for** $j = 1, \ldots, N_\lambda$ **do**
11: Set $\lambda = \lambda + \Delta\lambda(j)$;
12: Calculate $A(\lambda)$ and $b(\lambda)$ using $\bar{\mu}$;
13: Migrate $\bar{\mu}$: $\bar{\mu} = \bar{\mu} + \Delta\lambda(j)(A(\lambda)\bar{\mu} + b(\lambda))$;
14: **for** $i = 1, \ldots, N_p$ **do**
15: Migrate particles: $\mu^i = \mu^i + \Delta\lambda(j)(A(\lambda)\mu^i + b(\lambda))$;
16: **end for**
17: **end for**
18: **for** $i = 1, \ldots, N_p$ **do**
19: Set $x_k^i = \mu^i$;
20: $w_k^i = \dfrac{p(x_k^i|x_{k-1}^i)p(z_k|x_k^i)}{p(\tilde{\mu}^i|x_{k-1}^i)} w_{k-1}^i$;
21: **end for**
22: **for** $i = 1, \ldots, N_p$ **do**
23: Normalize $w_k^i = w_k^i / \sum_{s=1}^{N_p} w_k^s$;
24: **end for**
25: Estimate $\hat{x}_k$ from $\{x_k^i, w_k^i\}$;
26: (Optional) Resample $\{x_k^i, w_k^i\}_{i=1}^{N_p}$ and regularize to obtain $\{x_k^i, \frac{1}{N}\}_{i=1}^{N_p}$;
27: **end for**

Here we discuss the conditions the function $T(\cdot)$ is one-to-one for the PFPF based on EDH; PFPF based on LEDH can be addressed similarly. Consider two values $\mu_1 \neq \mu_2$, and the action of one update with step size $\Delta\lambda$ in the discretized exact Gaussian particle flow (line 15 in Algorithm 1). We have $\mu_1' = \mu_1 + \Delta\lambda(A(\lambda)\mu_1 + b(\lambda))$ and $\mu_2' = \mu_2 + \Delta\lambda(A(\lambda)\mu_2 + b(\lambda))$. For the EDH, $A(\lambda)$ and $b(\lambda)$ are the same for all μ^i, since linearization is performed at $\bar{\mu}$. If $\mu_1' \neq \mu_2'$ for all λ, then $T(\cdot)$ must be one-to-one. If $\mu_1' = \mu_2'$, then $(\mu_2 - \mu_1) = -\Delta\lambda A(\lambda)(\mu_2 - \mu_1)$. This equality holds only if $(\mu_2 - \mu_1)$ is an eigenvector of $A(\lambda)$ and its corresponding eigenvalue ρ_A satisfies $\Delta\lambda = \frac{-1}{\rho_A}$. We can ensure that this is not the case in several ways. $|A(\lambda)|$ can be proved to be bounded thus $|\rho_A|$ is bounded. If we choose $\Delta\lambda$ sufficiently small, then $\Delta\lambda < 1/\max|\rho_A(\lambda)|$ for any λ, and $\mu_1' \neq \mu_2'$. Alternatively, at the cost of increased computational expense, we can explicitly evaluate eigenvalues of $A(\lambda)$ at each pseudo-time step and choose the value of $\Delta\lambda$ accordingly.

Algorithm 2: Particle flow particle filtering (LEDH). (Replaces line 5–17 of Algorithm 1).

6: **for** $i = 1, \ldots, N_p$ **do**
7: Calculate $\bar{\mu}^i = g(x_{k-1}^i, 0)$;
8: Propagate particles $\tilde{\mu}^i = g(x_{k-1}^i, v_k)$;
9: Set $\mu^i = \tilde{\mu}^i$;
10: **end for**
11: Set $\lambda = 0$;
12: **for** $j = 1, \ldots, N_\lambda$ **do**
13: Set $\lambda = \lambda + \Delta\lambda(j)$;
14: **for** $i = 1, \ldots, N_p$ **do**
15: Calculate $A^i(\lambda)$ and $b^i(\lambda)$ using $\bar{\mu}^i$;
16: Migrate $\bar{\mu}^i$: $\bar{\mu}^i = \bar{\mu}^i + \Delta\lambda(j)(A^i(\lambda)\bar{\mu} + b^i(\lambda))$;
17: Migrate particles: $\mu^i = \mu^i + \Delta\lambda(j)(A^i(\lambda)\mu^i + b^i(\lambda))$;
18: **end for**
19: **end for**

4. SIMULATION AND RESULTS

4.1. Simulation setup

We adapt the multi-target simulation setup proposed in [12]. M targets move according to the dynamic model $x_k^{(m)} = Fx_{k-1}^{(m)} + v_k^{(m)}$, where $x_k^{(m)} = (\mathrm{x}_k^{(m)}, \mathrm{y}_k^{(m)}, \dot{\mathrm{x}}_k^{(m)}, \dot{\mathrm{y}}_k^{(m)})^T$ contains the position and velocity of target m. $F \in \mathbb{R}^{4\times4}$ is the transition matrix and $v_k^{(m)} \sim N(0, \sigma_v^2 V)$ is the noise vector. 25 acoustic amplitude sensors are deployed in a region of size 40 m$\times$40 m. Each target emits a sound of amplitude A, which is received by sensor s at position ζ^s with an amplitude

$$\bar{z}^s(x_k) = \sum_{m=1}^{M} \frac{A}{||(\mathrm{x}_k^{(m)}, \mathrm{y}_k^{(m)})^T - \zeta^s||^\kappa + d_0}.$$

The noisy measurement z_k^s of sensor s is modeled as

$$z_k^s = \bar{z}^s(x_k) + w_k^s$$

where $x_k = (x_k^{(1)}; x_k^{(2)}; x_k^{(3)}; x_k^{(4)})$ is the overall state vector of dimension 16, $w_k^s \sim N(0, \sigma_w^2)$.

In the simulation, $M = 4$, $A = 10$, $\sigma_v^2 = 0.05$, $\kappa = 1$, $d_0 = 0.1$, $F = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$, $V = \begin{bmatrix} 1/3 & 0 & 0.5 & 0 \\ 0 & 1/3 & 0 & 0.5 \\ 0.5 & 0 & 1 & 0 \\ 0 & 0.5 & 0 & 1 \end{bmatrix}$. We set $\sigma_w^2 = 0.01$, indicating that measurements are highly informative. The initial states of targets are $[12, 6, 0.001, 0.001]^T$, $[32, 32, -0.001, -0.005]^T$, $[20, 13, -0.1, 0.01]^T$ and $[15, 35, 0.002, 0.002]^T$. The simulation is implemented in Matlab and is conducted for 100 trials.

4.2. Parameter values for the filtering algorithms

We adopt the exponentially spaced step sizes recommended in [3]. 29 λ values are chosen with the constant ratio of step sizes being 1.2 and the initial step size is approximately 0.001. The exact value is chosen to guarantee that the sum of step sizes is equal to 1. The covariance matrix of the prior distribution is estimated with an EKF running in parallel. The redraw strategy in [8] is used for the EDH, LEDH, and NZDDH filters at the beginning of each time step. The diffusion term for the GPFIS algorithm is set to 0 as suggested in [9].

For all algorithms, we sample the initial mean from a Gaussian centered at the true initial states with variance 10 for positions and 1 for velocities. The covariance of the dynamic noise is modeled as $\begin{bmatrix} 3 & 0 & 0.1 & 0 \\ 0 & 3 & 0 & 0.1 \\ 0.1 & 0 & 0.03 & 0 \\ 0 & 0.1 & 0 & 0.03 \end{bmatrix}$, larger than that used to generate tracks. Resampling is performed when the effective sample size (ESS) is less than $\frac{N_p}{2}$. A small regularization noise is added after resampling. $N_p = 500$ particles are used in all tested algorithms except the bootstrap particle filter (BPF),

4.3. Experimental results

Figure 1 shows the average position errors at each time step for the various tracking algorithms we compare. PF-PF (LEDH) exhibits the smallest average tracking error. BPF with 1 million particles has slightly larger average error. However, the computational cost of BPF with 1 million particles is twice the cost of PF-PF (LEDH) (Table 1). LEDH has the third best performance but EDH has relatively large average error, indicating that the proposal distribution constructed from EDH may not be close to the posterior distribution, thus PFPF (EDH) performs poorly. NZDDH results in constantly lost tracks due to large movements of individual particles. So average errors are not shown here. GPFIS has similar tracking error with BPF with 100000 particles and is the most computationally expensive algorithm. EKF has the worst average tracking performance among those displayed in Figure 1.

Table 1. Average error and execution time per step. Results are produced with an Intel Xeon E5-4650 2.70GHz CPU.

Algorithm	PF-PF (LEDH)	PF-PF (EDH)	LEDH	EDH	GPFIS	EKF	BPF (10^5)	BPF (10^6)
Avg. error (m)	6.54	10.2	7.03	10.8	8.86	15.7	8.83	7.43
Avg. exec. time (s)	3.48	0.03	3.44	0.02	239	0.003	0.73	7.20

Figure 2 compares the average ESS among all tested algorithms with importance sampling. BPF with 100000 particles has an average ESS around 2, showing a strong discrepancy between the prior and the posterior, which requires an even greater amount of particles for BPF. GPFIS exhibits a high variation in ESS. PF-PF (LEDH) with 500 particles maintains a particle cloud with the highest effective sample size for most of time. And it is very efficient compared with GPFIS.

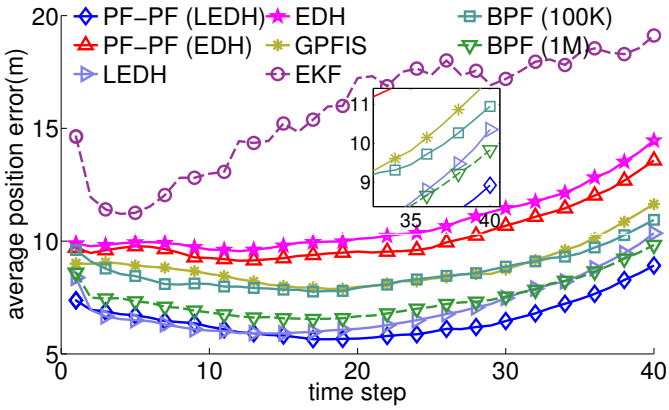

Fig. 1. Average errors at each time step.

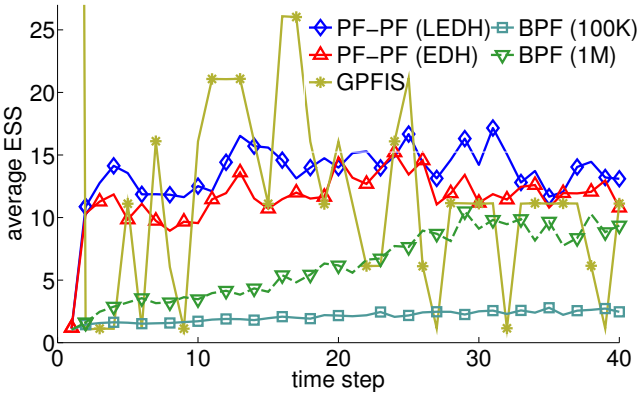

Fig. 2. Average ESS at each time step.

5. CONCLUSIONS

In this paper, we have proposed particle filtering algorithms that uses particle flow methods to construct the proposal distribution. By designing particle flow approaches with the property of one-to-one mapping, we can evaluate the important weights in an efficient manner. In contrast to a previous similar approach [9], the computational overhead is modest, and the accuracy is similar or superior. In a simulation setup where the average effective sample size of BPF with one million particles is less than 10, PF-PF (LEDH) with 500 particles has the smallest tracking error and maintains a particle cloud with a higher effective sample size. This demonstrates that PF-PF based on LEDH is capable of producing better particle presentations of posterior distributions than other filtering algorithms with much higher computational cost.

6. REFERENCES

[1] T. Bengtsson, P. Bickel, and B. Li, "Curse-of-dimensionality revisited: Collapse of the particle filter in very large scale systems," in *Probability and Statistics: Essays in Honor of David A. Freedman*, D. Nolan and T. Speed, Eds., vol. 2, pp. 316–334. Institute of Mathematical Statistics, Beachwood, OH, USA, 2008.

[2] F. Daum and J. Huang, "Nonlinear filters with log-homotopy," in *Proc. SPIE Signal and Data Processing of Small Targets*, San Diego, CA, USA, Sep. 2007, p. 669918.

[3] F. Daum and J. Huang, "Particle flow with non-zero diffusion for nonlinear filters," in *Proc. SPIE Conf. Signal Proc., Sensor Fusion, Target Recog.*, Baltimore, MD, USA, May 2013, p. 87450P.

[4] F. Daum, J. Huang, and A. Noushin, "Coulomb's law particle flow for nonlinear filters," in *Proc. SPIE Conf. Signal Proc., Sensor Fusion, Target Recog.*, San Diego, CA, USA, Sep. 2011, p. 81370B.

[5] F. Daum and J. Huang, "Small curvature particle flow for nonlinear filters," in *Proc. SPIE Signal and Data Processing of Small Targets*, Baltimore, MD, USA, May 2012, p. 83930A.

[6] F. Daum, J. Huang, and A. Noushin, "Exact particle flow for nonlinear filters," in *Proc. SPIE Conf. Signal Proc., Sensor Fusion, Target Recog.*, Orlando, FL, USA, Apr. 2010, p. 769704.

[7] S. Choi, P. Willett, F. Daum, and J. Huang, "Discussion and application of the homotopy filter," in *Proc. SPIE Conf. Signal Proc., Sensor Fusion, Target Recog.*, Orlando, FL, USA, May 2011, p. 805021.

[8] T. Ding and M. J. Coates, "Implementation of the daum-huang exact-flow particle filter," in *Proc. IEEE Statistical Signal Processing Workshop (SSP)*, Ann Arbor, MI, USA, Aug. 2012, pp. 257–260.

[9] P. Bunch and S. Godsill, "Approximations of the optimal importance density using Gaussian particle flow importance sampling," *J. Amer. Statist. Assoc.*, 2015, accepted.

[10] Sebastian Reich, "A guided sequential Monte Carlo method for the assimilation of data into stochastic dynamical systems," in *Recent Trends in Dynamical Systems*, vol. 35, pp. 205–220. Springer Basel, 2013.

[11] Y. Li, L. Zhao, and M. J. Coates, "Particle flow auxiliary particle filter," in *Proc. Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, Cancun, Mexico, Dec. 2015, accepted.

[12] O. Hlinka, O. Sluciak, F. Hlawatsch, P. M. Djuric, and M. Rupp, "Distributed Gaussian particle filtering using likelihood consensus," in *Proc. Intl. Conf. Acoustics, Speech and Signal Proc. (ICASSP)*, Prague, Czech Republic, May 2011, pp. 3756–3759.