Preprints of the 19th World Congress
The International Federation of Automatic Control
Cape Town, South Africa. August 24-29, 2014

FPGA Implementation of Rao-Blackwellized Particle Filter and its Application to Sensorless Drive Control

Václav Šmídl [*]
Robert Nedvěd, Tomáš Košan and Zdeněk Peroutka [**]

[*] *Institute of Information Theory and Automation, Prague, Czech Republic, (e-mail: smidl@utia.cas.cz)*
[**] *University of West Bohemia, Pilsen, Czech Republic (e-mail: peroutka@ieee.org*

AbstractRao-Blackwellied particle filter is a stochastic filter combining Kalman filters with particle filters. It is suitable for models that could be decomposed into linear and nonlinear part. Since the conditionally linear part can be solved by the Kalman filter, the sequential Monte Carlo is run only on the non-linear subspace. The resulting algorithm is a parallel evaluation of multiple Kalman filters with resampling. The parallel nature of this algorithm allows for very efficient implementation in hardware supporting parallel computation processes. In this contribution, we present implementation of the algorithm in the Field Programmable Gate Array (FPGA). Due to the used model and optimized implementation, the execution time of the filter is in units of microseconds and scales very favorably with the number of particles. This is demonstrated experimentally on a laboratory prototype of sensorless drive with permanent magnet synchronous machine (PMSM) of rated power of 10.7kW.

Keywords: Particle filtering/Monte Carlo methods; Software for system identification; Adaptive control -applications

1. INTRODUCTION

Control of systems with incomplete state observation is typically based on the use of state estimator or observer. The theory of observer design is very rich with many well known algorithms (Simon, 2006). A prominent example of a state observer is the (extended) Kalman filter that is often used in practice (Qin and Badgwell, 2003). However the Kalman filter is known to be unsuitable for non-linear and especially for non-Gaussian systems (Ristic et al., 2004). The particle filter (Gordon et al., 1993) was proposed as a more general algorithm for estimation of such systems. However, computational cost of the particle filter is typically much greater than that of the Kalman filter for the same system. The computation cost typically prevents its use in real-time application, or high-dimensions. Computational speed of the particle filter in low dimensions can be improved by the use of dedicated hardware (Athalye et al., 2005). However, its use in high dimensions is still very problematic (Quang et al., 2010).

Rao-Blackwellization is a technique that allows to improve accuracy of a particle filter in cases where the system permits for an analytical solution (Doucet et al., 2000), yielding a Rao-Blackwellized particle filter (RB-PF). This theory has been applied to state-space model in (Schön et al., 2005) under the name marginalized particle filter and has been successfully applied in many applications especially in target tracking and communications. However, its application in real-time control is still rare. One reason is that the RB-PF is still considered to be too computationally expensive.

In this paper, we show that the RB-PF can be efficiently implemented on a field programmable gate array (FPGA) which is nowadays a standard component of real-time control systems. Both key elements of the agorithm, i.e. the Kalman filter and the particle filter, has already been implemented in FPGA, in (Lee and Salcic, 1997) and (Athalye et al., 2005), respectively. Implementation of the RB-PF algorithm without the resampling operation was already published in (Šmídl et al., 2013). In this contribution, we present implementation of the complete algorithm including systematic resampling.

Performance of the algorithm is demonstrated on the task of sensorless control of a PMSM drive. Sensorless drive control is a task of speed, torque or position control of a drive without speed or position sensor on the rotor (Vas, 1998). We will show that the presented algorithm is able to operate the drive even at ultra-low speed. All presented experiments were made on a laboratory prototype of rated power of 10.7kW.

2. RAO-BLACKWELLIZED PARTICLE FILTERING

Rao-Blackwellized particle filtering is a technique of Bayesian filtering, where all unknowns are represented by probability density functions (distributions). By *Bayesian Filtering* we mean the recursive evaluation of the filtering distribution, $p(x_t|y_{1:t})$, using Bayes rule:

$$p(x_t|y_{1:t}) = \frac{p(y_t|x_t)p(x_t|y_{1:t-1})}{p(y_t|y_{1:t-1})}, \tag{1}$$

$$p(x_t|y_{1:t-1}) = \int p(x_t|x_{t-1})p(x_{t-1}|y_{1:t-1})dx_{t-1}, \tag{2}$$

where $p(x_1|y_0)$ is the prior distribution, and $y_{1:t} = [y_1, \ldots, y_t]$ denotes the set of all observations.

Equations (1)–(2) are analytically tractable only for a limited set of models. The most notable example of an analytically tractable model is linear Gaussian for which (1)–(2) are equivalent to the Kalman filter. For other models, (1)–(2) need to be evaluated approximately.

2.1 Kalman filtering

An important special case of the Bayesian filtering is the Kalman filter. The filter is derived for a linear model with Gaussian distributed errors:

$$\begin{aligned} x_t &\sim \mathcal{N}(Ax_{t-1}, Q), \\ y_t &\sim \mathcal{N}(Cx_t, R), \end{aligned} \tag{3}$$

where A and C are matrices of system dynamics and observation operator, respectively. Q and R are covariance matrices of appropriate dimensions. The posterior distribution (1) is then approximated by:

$$p(x_t|y_{1:t}) \approx \mathcal{N}(\hat{x}_t, P_t). \tag{4}$$

Its statistics $\hat{x}_t, P_t$ are evaluated recursively as follows:

$$\hat{x}_t = A\hat{x}_{t-1} - K\left(y_t - C\hat{x}_{t-1}\right). \tag{5}$$

$$R_y = CP_{t-1}C' + R_t, \tag{6}$$

$$K = S_{t-1}C'R_y^{-1}, \tag{7}$$

$$P_t = S_{t-1} - S_{t-1}C'R_y^{-1}CS_{t-1}, \tag{8}$$

$$S_t = AP_tA' + Q_t. \tag{9}$$

An important quantity for further development in this work is the predictive density of the observations

$$p(y_t|y_{1:t-1}) = \mathcal{N}(C\hat{x}_{t-1}, R_y). \tag{10}$$

2.2 Particle filtering

The particle filtering is based on approximation of the posterior (1) by an empirical probability density function

$$p(x_{1:t}|y_{1:t}) \approx \sum_{i=1}^{N} w_t^{(i)}\delta(x_{1:t} - x_{1:t}^{(i)}), \tag{11}$$

where $x_{1:t}^{(i)}$, $i = 1, \ldots, N$, are samples of the state space trajectory. Assimilation of the measured data is then achieved via sampling-importance-resampling procedure, where the weights can be computed recursively,

$$w_t^{(i)} \propto w_{t-1}^{(i)} \frac{p(y_t|x_t)p(x_t|x_{t-1})}{q(x_t|y_t)}. \tag{12}$$

Good proposal function and resampling strategy are necessary steps preventing degeneracy of the particle filter (12), Doucet et al. (2001). We will implement the systematic procedure since it can be implemented in one sweep through the particles with $O(N)$ complexity. The principle of the resampling procedure is demonstrated in Figure 1. The first step is to evaluate the number of copies for each particle. This is achieved by generating N "random" samples, $s_0 \ldots s_N$ between zero and one. The key idea of systematic resampling is to randomly choose only the first number s_0 from interval $\langle 0, \frac{1}{N} \rangle$ and then create a deterministic grid with step size $s_i = s_{i-1} + 1/(N+1)$. The number of copies of the ith particle is given by the number of indexes j for which it holds that $\sum_{i=1}^{i-1} w_i < s_j < \sum_{i=1}^{i} w_i$.

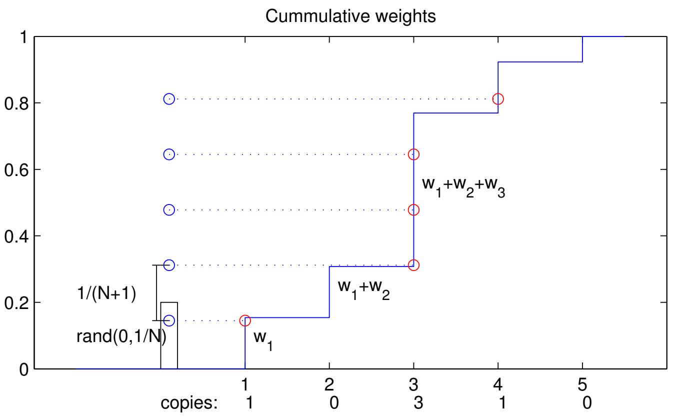

Figure 1. Illustration of the systematic resampling algorithm.

However, the resampling operation introduces an additional stochastic variance of the estimates and it is advantageous to perform it only when necessary. The most common condition for resampling is evaluation of the effective sample size $n_{eff} = \left(\sum_{i=1}^{n}(w_t^{(i)})^2\right)^{-1}$ and triggering the resampling operation only when it falls under a chosen threshold (e.g. $n_{eff} < 0.9n$).

2.3 Rao-Blackwellized particle filtering

Approximation (11) is unnecessary if the system has a linear Gaussian part (Schön et al., 2005). In such a case, it is possible to split the state into linear and non-linear part, $x_t = [x_t^l, x_t^n]$, such that

$$x_{t+1}^l = A(x_t^n)x_t^l + B(x_t^n)u_t + \epsilon_{l,t}, \tag{13}$$

$$x_{t+1}^n = f(x_t, u_t, \epsilon_{n,t}), \tag{14}$$

$$y_t = C(x_t^n)x_t^l + Du_t + \epsilon_{y,t}. \tag{15}$$

Here, $\epsilon_{l,t}$ and $\epsilon_{y,t}$ are assumed to be Gaussian-distributed with zero mean and known covariance matrix. Function $f()$ is an arbitrary non-linearity and $\epsilon_{n,t}$ can have an arbitrary distribution. Note that if x_t^n was known, equations (13) and (15) form a linear Gaussian model that can be estimated by the Kalman filter. The resulting estimate would be in the form of Gaussian density with mean and covariance conditioned on the non-linear state.

The idea of the RB-PF is to approximate the posterior density of the non-linear part by the empirical density (11). The full posterior density is then approximated by the chain rule of probability calculus as follows:

$$\begin{aligned} p(x_t^l, x_t^n|y_{1:t}) &= p(x_t^l|x_{1:t}^n, y_{1:t})p(x_t^n|y_{1:t}), \\ &= \sum_{i=1}^{n} w_t^{(i)}\mathcal{N}(\hat{x}_t^{l(i)}, P_t^{(i)})\delta(x_t^{n(i)} - x_t^n). \end{aligned}$$

```
Algorithm 1 Algorithm of the Rao-Blackwellized particle filter.
Initialize: generate $\hat{x}_0^{l(i)}, P_t^{(i)}, x_0^{n(i)}$ and set $w_0^{(i)} = 1/N$.
On-line: At each time step:
(1) For each particle:
    (a) Predict non-linear part of the state using (14),
    (b) Evaluate $A(x^{n(i)}), B(x^{n(i)}), C(x^{n(i)})$,
    (c) Execute the Kalman filter (5)–(9) to obtain $\omega_t^{(i)}, P_t^{(i)}$.
    (d) Evaluate non-normalized weight $\tilde{w}_t$, (10) and auxiliary variables for required moments e.g. $\sum \tilde{w}_t \hat{x}_t$.
(2) Normalize weights $w_t = \tilde{w}_t / \sum_{i=1}^n \tilde{w}_t$ and required moments,
(3) Evaluate the number of offsprings for each particle using systematic resampling,
(4) Copy the particles with more than one offspring to the positions of particles without offsprings.
```

Here, $x_t^{n(i)}$, $i = 1, \dots n$, are the samples from the non-linear state and $\mathcal{N}(\hat{x}_t^{l(i)}, P_t^{(i)})$ is the posterior density of the ith Kalman filter associated with the ith particle of the non-linear state. The weight

$$w_t^{(i)} \propto p(y_t | y_{1:t-1}, x_t^{n(i)}) w_{t-1}^{(i)},$$

is proportional to the marginal likelihood of the associated Kalman filters (10). Estimates of the unknown state are typically evaluated as the first moment:

$$\hat{x}_t = \frac{\sum_i \tilde{w}_t^{(i)} \hat{x}_t^{(i)}}{\sum_i \tilde{w}_t^{(i)}}.$$

Remark 1. Theoretical optimal performance of the particle filter is achieved for perfect random sampling. This may be difficult to implement in real-time and also potentially undesired due to different results of estimation for different realization of the random samples. Therefore, we intentionally draw samples of 'random variables' from a buffer of stored values of fixed length.

3. FPGA IMPLEMENTATION OF THE RB-PF

Algorithm 1 is intentionally written in four steps. Each of these steps will be implemented as a block of the FPGA, the first two blocks are implemented as a pipeline. The advantage of this solution is that all data goes through the same calculation process and no additional FPGA resources are needed when the number of particles is increased.

All calculations are done in floating-point format, not the IEEE 754 standard, but a shorter type with 5 bits for the exponent and 16 bits for the mantisa. This reductions of all number representations saves a lot of FPGA resources and increases the calculation speed, at the price of reduced accuracy and lower dynamic range. However, this reduction is still sufficient for evaluation of the RB-PF algorithm.

In our implementation, the data acquisition and the control algorithm is implemented in the DSP which simplifies and speed-up control design and saves the FPGA resources. The DSP sends the observed quantities and the calculated actions to the FPGA via the I/O buffer and triggers computation of the RB-PF estimates. It also reads

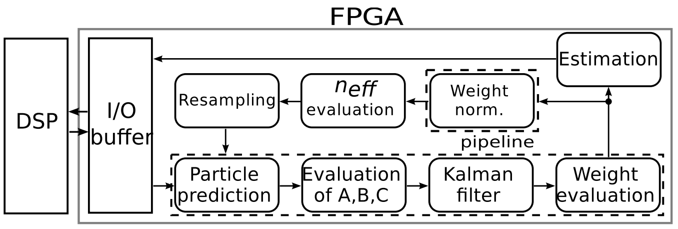

Figure 2. Block diagram of the FPGA implementation of the RB-PF algorithm. Dashed lines denote pipelined blocks of the algorithm.

the results of RB-PF calculation and uses the estimated states as inputs to the controller. The architecture of the hardware implementation of RB-PF is displayed in Fig. 2. The first component is the above described I/O buffer, where all variables and constants are stored. It is interconnected with the DSP via a 16-bit data and 9-bit address bus, with maximum data transfer rate depending on the DSP speed. The I/O buffer also contains status and control register, which is used for triggering the calculation and signaling errors or calculation state. The next blocks of FPGA implementation of RB-PF are described in details in the following sections.

3.1 Particle prediction and correction

The first pipeline of the algorithm evaluates particle prediction and correction. The first part is the particle generator. Its function is to mark each particle going through the pipeline. This marking allows to detect the last particle reaching the end of the pipeline, which finishes the calculation process. As shown in Fig.3, two additional particles are present, one at the beginning and one at the end. These two particles are not used in the calculation, but they ensure correct evaluation of the particles between them. Specifically, the first particle leads the way through the pipeline. Since it is the first particle in the pipeline, all pipeline stages are pointing to this particle and change its value until another particle number gets in, therefore this particle is corrupted and contains incorrect values. The same issue is with the last particle, which closes the pipeline. The particle generator also adds a random sample to each particle $x_t^{n(i)}$ before it is send it to the matrix update block. The random number is not generated by the hardware, but it is predefined in a ROM table, which saves the FPGA resources. The ROM table is implemented as a large buffer of samples from the chosen probability distribution. Each draw of the random number shifts the actual position of the buffer. The length of the buffer is chosen as a large prime number (997 in our case) to extend the generator period as long as possible. The sine and cosine functions are not calculated either, but are also precalculated in the ROM table. Sine table consists of 8192 values that describes a quarter of the sine wave, which implies resolution of 0.01°. The internal logic only chooses the requested value of the argument. This solution allows both sine and cosine calculation being done in one clock tact.

The block of the Kalman filter was implemented using manually tuned evaluation of equations (5)–(10) for a one dimensional state and two dimensional observation. For higher dimensional Kalman filters, it may be advantageous

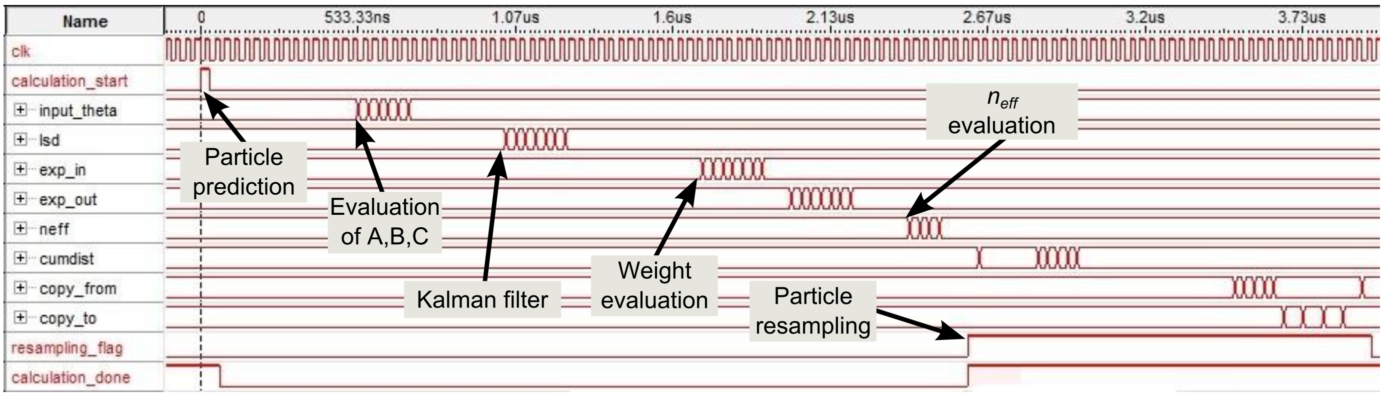

Figure 3. Wave diagram of the FPGA implementation of the RB-PF algorithm. Arrows point to the instant where blocks of the algorithm from Figure 2 starts.

to use more general techniques described e.g. in (Sudarsanam, 2010). The implemented Kalman filter consist of many addition and multiplication, taking altogether 26 clock tacts. The weight evaluation block contains the most expensive operation of the algorithm, the exp function. This function was implemented using Altera MegaWizard manager. To achieve maximum calculation speed, the exp function is clocked by both the rising and falling edge and it takes only 9 clock tacts instead of 18 when single edge clocked.

This block also evaluates accumulators of $\sum_i \tilde{w}_t^{(i)}$ for normalization, $\sum_i (\tilde{w}_t^{(i)})^2$ for evaluation of n_{eff}, and $\sum_i \tilde{w}_t^{(i)} \hat{x}_t$ for state estimate.

3.2 Normalization and resampling

The sum of the particle weights is accumulated in the previous pipeline. When the last particle reaches the end of the previous pipeline, their sum is ready and a weight normalization pipeline is started to divide all weights by their sum. At the same time the particle with the highest weight value is selected in the estimation block and it's estimate is sent into the I/O buffer, the calculation_done flag is set, signaling that the state estimate is ready for use by the DSP for calculation of the control action. After the weight normalization, the number of efficient particles is calculated and compared to the selected threshold level, and the resampling step is started if needed. The resampling block is not pipelined but implemented as a state machine with states: (i) initialization, (ii) update of the current sample s_i, (iii) comparison with cumulative sum of the weights, (iv) delete particle, and (v) copy particle. States (ii)-(v) are evaluated N times, taking one tact of the clock each time.

4. EXPERIMENTAL RESULTS

The presented implementation was developed for application of the RB-PF for sensorless control of PMSM drive, which was presented in (Šmídl and Peroutka, 2012). We briefly review specific forms of the general equations in Section 2.

4.1 RB-PF for the PMSM drive

A commonly used model of a PMSM is the following first order model for time step Δt:

$$i_{d,t+1} = a_d i_{d,t} + b_d i_{q,t}\omega_t + c_d u_{d,t} + \epsilon_{d,t}, \tag{16}$$
$$i_{q,t+1} = a_q i_{q,t} - f_q\omega_t - b_q i_{d,t}\omega_t + c_q u_{q,t} + \epsilon_{q,t}, \tag{17}$$
$$\omega_{me,t+1} = \omega_{me,t} + \epsilon_{\omega,t}, \tag{18}$$
$$\vartheta_{e,t+1} = \vartheta_{e,t} + \omega_{me,t}\Delta t + \epsilon_{\vartheta,t}. \tag{19}$$

Here, i_d, i_q, u_d and u_q represent components of stator current and voltage vector in the rotating reference frame, respectively; ω_{me} is electrical rotor speed and ϑ_e is electrical rotor position. Constants a_d, a_q, b_d, b_q, c_d, c_q, and f_q and known parameters of a particular machine. Noise terms $\epsilon_{d,t}, \epsilon_{q,t}, \epsilon_{\omega,t}, \epsilon_{\vartheta,t}$, aggregate errors caused by inaccurate discretization, uncertainties in parameters (e.g. due to temperature changes, saturation), unobserved physical effects (such as the unknown load, dead-time effects, nonlinear voltage drops on power electronics devices).

We will use a reduced order model of the PMSM drive, i.e. the state variable being only $x_t = [\omega_{me,t}, \vartheta_{e,t}]$. The state equations are (18)–(19) and (16)–(17) are the observation. Since we observe only the stator currents in stationary reference frame, $i_{\alpha,t}$ and $i_{\beta,t}$, they are transformed into the rotating d-q reference frame as follows:

$$i_{d,t} = i_{\alpha,t}\cos\vartheta_{e,t} + i_{\beta,t}\sin\vartheta_{e,t}, \tag{20}$$
$$i_{q,t} = -i_{\alpha,t}\sin\vartheta_{e,t} + i_{\beta,t}\cos\vartheta_{e,t}. \tag{21}$$

This transformation is done for each particle with angle $\vartheta_{e,t}^{(i)}$. The measurement errors are assumed to be non-correlated Gaussian with variances $var(\epsilon_{d,t}) = r_i$, $var(\epsilon_{q,t}) = r_i$.

Such a model is a special case of the model (13)–(15) with assignments $x^l \equiv \omega_{me}$, $x^n \equiv \vartheta_e$, and

$$\begin{aligned} A &= 1, \\ y_t &= [i_{d,t} - a_d i_{d,t-1} - c_d u_{d,t-1}, \\ &\quad i_{q,t} - a_q i_{q,t-1} - c_q u_{q,t-1}]^T, \\ C &= [b_d i_{q,t}, -(f_q + b_q i_{d,t})]^T. \end{aligned}$$

4.2 Field oriented control of the drive

The drive control is based on the conventional vector control (cascade PI control) in Cartesian coordinates in rotating reference frame (d,q) linked to a rotor flux linkage

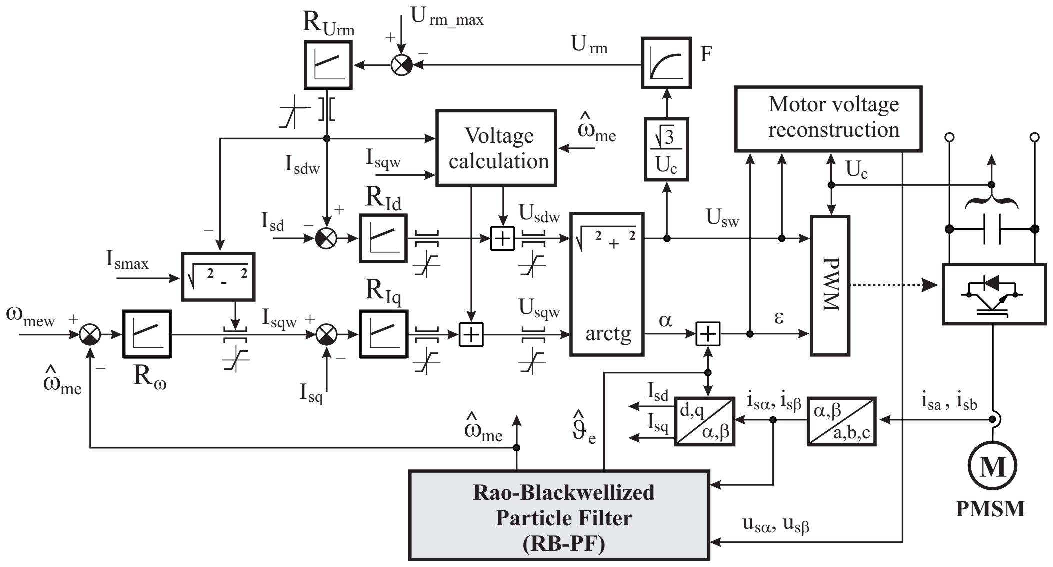

Figure 4. Investigated sensorless control of a PMSM drive with the proposed RB-PF

vector, Fig. 4. An input to the drive controller is the commanded electrical rotor speed ω_{mew} which is controlled by the PI controller R_{ω}. Output of R_{ω} is the demanded torque component I_{sqw} of the stator current vector. The torque (I_{sqw}) and flux (I_{sdw}) currents are controlled by the PI controllers $\mathrm{R}_{\mathrm{Isd}}$ and $\mathrm{R}_{\mathrm{Isq}}$, respectively. The flux weakening is secured by the PI controller $\mathrm{R}_{\mathrm{Urm}}$ which controls the PWM modulation depth (signal U_{rm}) and commands the flux current I_{sdw}. The current controllers are supported by feed-forward voltage calculation (block 'voltage calculation') which computes the components of the required stator voltage vector in (d,q) frame using a simplified model of the PMSM in steady-state. The components of the stator current vector $(i_{s\alpha}, i_{s\beta})$ and the reconstructed stator voltage vector $u_{s\alpha}, u_{s\beta}$ in the stationary reference frame are inputs to the RB-PF. The stator voltage vector is reconstructed from the measured dc-link voltage and known switching combination of the voltage-source converter. The RB-PF output is the estimated electrical rotor speed $\hat{\omega}_{me}$ and the electrical rotor position $\hat{\vartheta}_e$.

4.3 Experimental evaluation

The proposed sensorless drive control with the presented implementation has been tested on a laboratory prototype of PMSM drive of rated power of 10.7kW. The sampling period of the controller was 125 μs. The developed prototype of the PMSM drive was during the experimental tests operated in both sensored and sensorless mode. All results are obtained with the RB-PF with 5 particles and covariance matrices $q_{\omega} = 0.1$, $q_{\vartheta} = 0.003$, $r = 0.05$.

The drive was operated in both sensored and sensorless mode under a triangular speed profile with the commanded mechanical rotor speed of ± 50 rpm. Results of estimation of the RB-PF in open loop (sensored mode) are displayed in Fig. 5 left and sensorless mode in Figure 5 right. Note that the drive trajectory in the sensorless mode of control is almost identical to that in the sensored mode.

In comparsion with the traditional EKF, the RB-PF was able to achieve lower speed of secure operation. However, reliable operation at standstill was not yet achieved. We investigate the use of more complex models of the drive with linear structure (13), e.g. . Implementation of such model in the presented scheme would require to replace only the first step of Algorithm 1, all other steps would be implemented identically.

5. CONCLUSION

The need for reliable state observers arises commonly in control of systems with imperfect observation. The Rao-Blackwellized particle filter has been shown to be an efficient estimator for many practical problems. However, its use in real-time control is still limited. One reason is its perception as a computationally costly algorithm. In this paper, we have demonstrated that this algorithm can be implemented efficiently on a field programmable gate array (FPGA). Due to parallel nature of the RB-PF, its computational cost is comparable to implementation of a Kalman filter for the same model. The most attractive feature of the proposed FPGA implementation is its linear increase of the computational speed with the number of particles. Specifically, an additional particle extends computational time of the algorithm by five tacts of the FPGA clock. This allows to implement the filter for problems with high sampling frequency.

Feasibility of the presented implementation was demonstrated on the problem of sensorless control of a PMSM drive. The model of the drive is suitable for approximation by the RB-PF. The proposed implementation of the RB-PF algorithm allowed to test real-time performance of the closed loop with many more particles than ever before.

ACKNOWLEDGEMENTS

This work was supported by the European Regional Development Fund and Ministry of Education, Youth and Sports of the Czech Republic under project No. CZ.1.05/2.1.00/03.0094: Regional Innovation Centre for

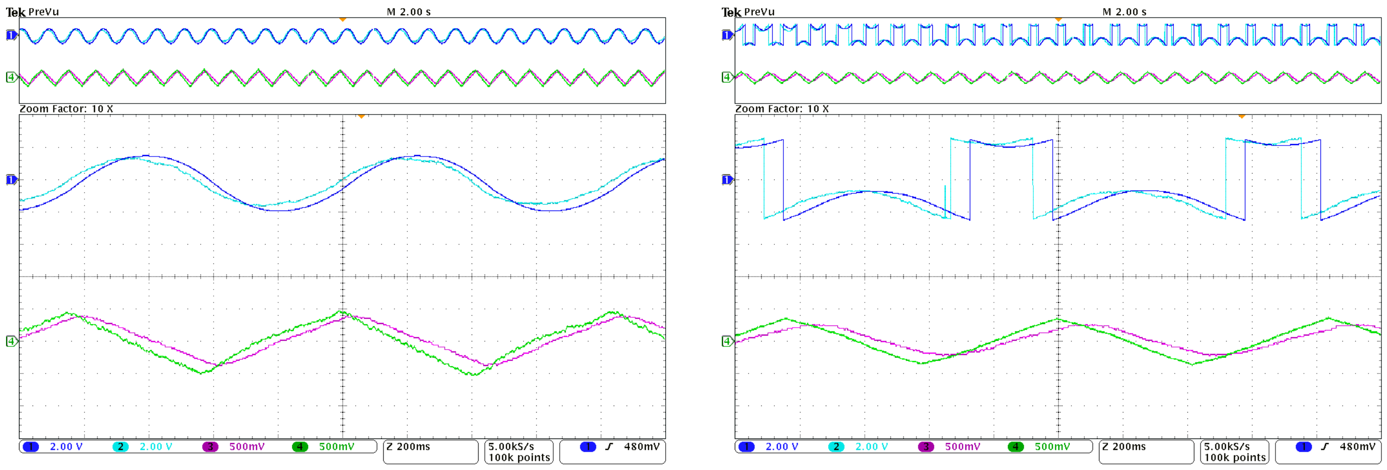

Figure 5. Experimental result for triangular profile of the commanded mechanical rotor speed of ±50 rpm (i.e. 2Hz electrical rotor speed). **Left:** Sensored mode (open loop estimation) of the drive control. **Right**: sensorless mode (estimation in closed loop). *ch1*: electrical rotor position (sensor) [144 deg/div], *ch2*: estimated electrical rotor position (RB-PF) [144 deg/div], *ch3*: rotor speed (sensor) [30 rpm/div], *ch4*: estimated rotor speed (RB-PF) [30 rpm/div].

Electrical Engineering (RICE) and by the Czech Science Foundation under project No. P102/11/0437.

REFERENCES

Athalye, A., Bolic, M., Hong, S., and Djuric, P. (2005). Generic Hardware Architectures for Sampling and Resampling in Particle Filters. *EURASIP JOURNAL ON APPLIED SIGNAL PROCESSING*, 17, 2888.

Doucet, A., de Freitas, N., and Gordon, N. (eds.) (2001). *Sequential Monte Carlo Methods in Practice*. Springer.

Doucet, A., de Freitas, N., Murphy, K., and Russell, S. (2000). Rao-Blackwellised particle filtering for dynamic Bayesian networks. In *Proc. of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, 176–183.

Gordon, N., Salmond, D., and Smith, A. (1993). Novel approach to nonlinear/non-Gaussian Bayesian state estimation. In *Radar and Signal Processing, IEE Proceedings F*, volume 140, 107–113. IEE.

Lee, C. and Salcic, Z. (1997). High-performance fpga-based implementation of kalman filter. *Microprocessors and Microsystems*, 21(4), 257–265.

Qin, S.J. and Badgwell, T.A. (2003). A survey of industrial model predictive control technology. *Control engineering practice*, 11(7), 733–764.

Quang, P.B., Musso, C., and Le Gland, F. (2010). An insight into the issue of dimensionality in particle filtering. In *Information Fusion (FUSION), 2010 13th Conference on*, 1–8. IEEE.

Ristic, B., Arulampalam, S., and Gordon, N. (2004). *Beyond the Kalman Filter: Particle Filters for Tracking Applications*. Artech House Publishers.

Schön, T., Gustafsson, F., and Nordlund, P.J. (2005). Marginalized particle filters for mixed linear/nonlinear state-space models. *IEEE Transactions on Signal Processing*, 53, 2279–2289.

Simon, D. (2006). *Optimal state estimation: Kalman, H infinity, and nonlinear approaches*. Wiley.

Sudarsanam, A. (2010). *Analysis of Field Programmable Gate Array-Based Kalman Filter Architectures*. Ph.D. thesis, University of Utah.

Vas, P. (1998). *Sensorless vector and direct torque control.* Oxford University Press New York.

Šmídl, V., Nedvěd, R., Košan, T., and Peroutka, Z. (2013). Fpga implementation of marginalized particle filter for sensorless control of pmsm drives. In *Proceedings of the IECON 2013 - 39th Annual Conference of the IEEE Industrial Electronics Society*, 8219–8224. Vienna, Austria.

Šmídl, V. and Peroutka, Z. (2012). Marginalized particle filter for sensorless control of pmsm drives. In *Proc. of the 38th Annual Conference of the IEEE Industrial Electronics Society*, 1877–1882. Montreal, Canada.