This is an Accepted Manuscript version of the following article, accepted for publication in:

J. Olaizola, C. -S. Bouganis, E. S. de Argandoña, A. Iturrospe and J. M. Abete, "Real-Time Servo Press Force Estimation Based on Dual Particle Filter," in *IEEE Transactions on Industrial Electronics*, vol. 67, no. 5, pp. 4088-4097, May 2020.

DOI: https://doi.org/10.1109/TIE.2019.2921292

Real-time Servo Press Force Estimation based on dual Particle Filter

***Abstract*—The ability to monitor the quality of the metal forming process as well as the machine's condition is of significant importance in modern industrial processes. In the case where a physical device (i.e. sensor) cannot be deployed due to the characteristics of the system, models that rely on the estimation of both the applied force and the dynamic behavior of the machine (i.e. system) are adopted. The development of such models and the corresponding algorithms used to estimate the above mentioned quantities has attracted the interest of the community. The main contribution of this paper is the estimation of a servo press force by employing a novel dual particle filter (dPF)-based algorithm, achieving a maximum relative error in the force estimation of 3.6%. Moreover, to address real-time performance requirements, the paper proposes a field programmable gate array (FPGA)-based accelerator that improves the sampling rate by a factor of 200 compared to a processor-based solution, thus enabling the deployment of the system in many realistic scenarios.**

***Index Terms*—dPF, dual particle filter, FPGA, model-based soft sensor, MBSS, state, unknown input.**

I. Introduction

MONITORING plays a crucial role in modern industrial processes and machines. It provides useful information about the condition of the machine and the process quality. Furthermore, monitoring can detect signs of malfunction that may cause the machine's failure and its subsequent downtime.

In the specific case of a servo press, a press that is driven by a servo motor, the process force has a high impact in the quality of the manufactured part. The monitoring of the force can reveal aspects related to the quality of the produced part, such as information about the profile of the applied force that it was subjected to during a stroke. The industry's classical approach for monitoring press force is to place sensors on machine components. Due to ease of installation and the little wear they suffer compared with other locations, such sensors are commonly installed on the press frame and the connecting rods [1]. However, force sensors located in forming dies provide more precise information, since they are closer to the process. When the die is replaced to produce a new design, these sensors are usually replaced, which increases the cost of monitoring [1]. These sensors also experience higher levels of wear than those mounted on the frame or connecting rods. It is also well known that force sensors tend to lose their initial calibration over the working cycles, and they need periodic maintenance for recalibration [2].

To address the increased costs and the recalibration need associated with physical sensors, other solutions, such as model-based soft(ware) sensors (MBSS), can be adopted, which take advantage of the already existing sensors and a model of the targeted machine. A MBSS is a software-based sensor that uses a mathematical representation (model) of the machine and/or process and some measurements to provide estimations [3]. Models are described by state variables (also known as states) that determine the behavior of the system. At the same time, the system can be driven by external inputs. Those inputs can be either known or unknown depending on whether they are measured or unmeasured. In the case of servo presses, the press force is modelled as an input. The signals utilized by the control of the system can be used by soft sensors to estimate states and the unknown input (unmeasured force) of the servo press.

In the last couple of decades, several works have successfully applied MBSSs to the monitoring of industrial processes and machines to estimate states and unknown inputs of systems. Regarding the estimation of states, Park et al. proposed a Kalman filter (KF)-based soft sensor for the dynamic compensation of the spindle integrated force sensors of a machine tool [4]. They employed the KF to remove the influences of structural modes that are present in the cutting force measurement, thus compensating the deviations of the force sensor. The extended KF (EKF) has also been applied in multiple industrial applications. Chinniah et al. proposed an EKF approach for failure monitoring of a high performance hydrostatic actuation system [5]. Measuring the position and the speed of the piston and the angular speed of the electric motor, they estimated the effective compression resistance of the system.

With respect to the estimation of unknown inputs, Tsai et al. developed an unknown input observer (UIO) for estimating the load torque of a direct current DC permanent magnet motor employed in a manufacturing process [6]. They measured the motor's currents to estimate the load torque through the UIO. More recently, Reid et al. presented an augmented-state KF for estimating the cutting force of a longwall shearer [7]. However, approaches based on KF assume that the distribution of the estimated magnitudes of the input signals follow a Gaussian distribution, and they fail to perform well in the case of non-Gaussian estimations, as claimed by Gordon et al. [8].

To overcome this limitation, Monte Carlo (MC) methods have been proposed. MC-based soft sensors are more suitable according to Candy et al. [9], since they make fewer assumptions than KF based approaches about the nature of the process, particularly when the distribution of the estimations over time is unknown. In addition, MC methods do not use a process covariance matrix, which influences the performance of the algorithm and is not straightforward to determine [10]. Taking this approach, Karandikar et al. predicted the turning tool life of a machining process by means of the Markov chain Monte Carlo (MCMC) approach [11]. The estimations were validated under different cutting conditions for the prediction of the remaining tool life. He et al. implemented a particle filters (PF) soft sensor for the prognosis and health management of an automated machining process [12]. The authors were able to detect a fault on a bearing of the machine before it led in serious damage. Other important publications that address the PF-based estimation of states (and/or parameters) in nonlinear and non-Gaussian models include those presented by Dahlin et al. [13], Kantas et al. [14] and Imani et al. [15]. PF has also been combined with other algorithms for estimating multiple model elements simultaneously. Imani et al. proposed a PF approach in combination with a maximum likelihood method for the simultaneous estimation of the states and parameters of Boolean gene regulatory networks [16]. The authors took advantage of noisy ribonucleic acid sequencing time series data to accomplish their estimation objectives. Olivier et al. presented a dual particle filter (dPF) algorithm for state and parameter estimation that could be applied to a run-off-mine ore mill process [17]. One of the PFs estimates the states, while the other estimates the parameters. Both share their estimations in every iteration. Mejri et al. proposed a PF-based simultaneous state and parametrized input estimation for chaotic systems [18]. Although the work tackles the simultaneous estimation of states and unknown inputs, it is not able to deal with time-varying unknown inputs due to its parametrization. In the dual PF scheme, the state particles are decoupled and weighted separately from the parameter particles, improving the accuracy of the estimations [19]. A simultaneous estimation of varying unknown input and states was proposed by Khan et al. [20] in their particle swarm optimization (PSO), with which they obtained an estimation of the unknown input by using a optimization scheme at each iteration. The main drawback of the PSO is that quality of estimations and convergence are highly sensitive to the definition of the algorithm's parameters, as stated by Zhang et al. [21].

Taking advantage of the flexibility of MC methods, this paper presents a novel dPF algorithm that is able to estimate simultaneously a time-varying unknown input (process force) and states (the angular position, speed and acceleration) of the crankshaft of the servo press. At each time step k, one PF estimates the unknown input (PF-UI), while the other PF estimates the states (PF-S). At the end of each time step, the PFs share their results.

Moreover, as MC-based methods, and therefore the proposed dPF algorithm, are subject to a high computational cost, mainly due to the number of computations they have to perform to evaluate the likelihood weight of particles [22]. To address this, the paper also presents a field programmable gate array (FPGA)-based accelerator to enable real-time monitoring and estimation of the system's states and the unknown input.

The rest of the paper is organized as follows. Section II presents the model of the servo press. Section III describes the proposed dPF-based approach, starting from the conventional PF and showing the enhancements carried out for the simultaneous estimation of an unknown input and states. Section IV presents the tuning of the dPF parameters and the hardware implementation of the algorithm, comparing the achieved performance with the one obtained by a software-based solution. Section V shows the experimental results of two different experiments, along with the error between the measured and estimated forces. Finally, Section VI provides the conclusions that were extracted from the developed work.

II. Servo Press Model

A servo press is an electromechanical system formed by an electrical motor and a mechanical subsystem, which is composed of all the mechanical components of the press. Most servo presses use permanent magnets synchronous machines (PMSM) to drive the mechanical part of the press [23]. Field oriented control is the most utilized control scheme in many industrial applications. It uses the servomotor's electrical (voltage and current) signals and its angular position to control servo press operation.

The mechanical part of the servo press is based on a ram-crank mechanism, as illustrated in Fig. 1. Its transmission chain is comprised of a gearbox, a crankshaft, two connecting rods and a ram, which transforms the electric torque τ_e generated by the servomotor into a linear force.

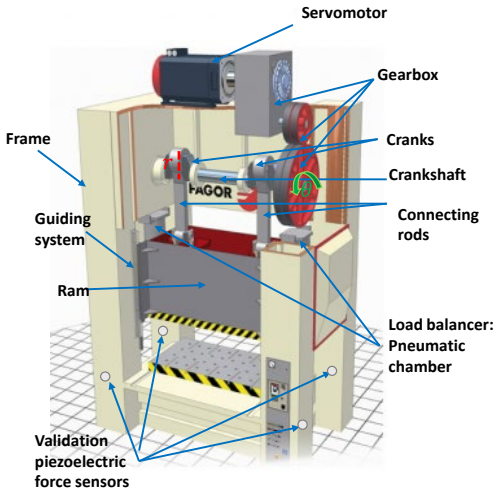

Fig. 1. Computer aided design (CAD) representation of Fagor Arrasate's SDM2-400-2400-1200 servo press.

The modelling of a ram-crank type servo press was carried out by the Lagrange method presented in [24] and [25]. A simplified discrete dynamic model of the servo press is shown in (1) (the complete model is defined in Appendix A). All the inertial terms of the servo press components have been

included in the nonlinear function $m_k(\cdot)$, and the process force and the system's state vector are defined as F_{stroke_k}, and $\boldsymbol{x}_k$, respectively. The system's state vector $\boldsymbol{x}_k$ is composed of the angular position θ_k, angular speed $\dot{\theta}_k$ and angular acceleration $\ddot{\theta}_k$ of the crankshaft. $f_{\theta_k}(\cdot)$ is a function that transforms force into torque, and t_s is the discrete time interval. τ_{fric_k}, F_{lb_k} and F_{fric_k} are the friction torque, the force of the load balancer that compensates the weight of the ram and the friction force of the system, respectively, while $\boldsymbol{\omega}_k$ is the process noise vector that represents the model's uncertainties. The known inputs $\boldsymbol{u}_k$ of the model are τ_{e_k}, τ_{fric_k}, F_{lb_k} and F_{fric_k}, while F_{stroke_k} is the unknown input d_k.

$$m\left(\theta_{k-1}, \dot{\theta}_{k-1}, \boldsymbol{\omega}_k\right)\ddot{\theta}_k = \tau_{e_k} - \tau_{fric_k} - f_\theta\left(F_{lb_k}, F_{fric_k}, F_{stroke_k}\right)$$

$$\begin{bmatrix}\theta_k \\ \dot{\theta}_k \\ \ddot{\theta}_k\end{bmatrix} = \begin{bmatrix}\theta_{k-1} + \dot{\theta}_{k-1} t_s \\ \dot{\theta}_{k-1} + \ddot{\theta}_{k-1} t_s \\ \dfrac{\left(\tau_{e_k} - \tau_{fric_k} - f_{\theta_k}\left(F_{lb_k}, F_{fric_k}, F_{stroke_k}\right)\right)}{m_k\left(\theta_{k-1}, \dot{\theta}_{k-1}, \boldsymbol{\omega}_k\right)}\end{bmatrix} \quad (1)$$

III. dPF Architecture

A. Overview of PF

PF is a sequential Monte Carlo method that is frequently used for recursively estimating (in k iterations) a system's states based on the posterior probability of generated particles x_k^i. These particles are selected by importance sampling techniques [8], which associate a likelihood weight $W_{x_k}^i$ to the generated (sampled) particle x_k^i. The likelihood weight of each particle $\{x_k^i, W_x^i\}$ represents the probability of that particle being sampled from the probability density function defined by the measurements, as expressed in (2).

$$p(x_k, |\boldsymbol{Y}_k) \approx \sum_{i=1}^{N_x} W_{x_k}^i \, \delta(x_k - \hat{x}_k^i) \quad (2)$$

where $\boldsymbol{Y}_k = \{y_1, y_2, .., y_k\}$ is a set of accumulated measurements up to time k and is defined as $y_k = h_k(\boldsymbol{x}_k) + v_k$, where $h_k(\cdot)$ and v_k are the measurement function and measurements noise, respectively. δ is the Dirac delta function, and N_x is the number of particles.

PF employs four steps to carry out the estimation of states: the Prediction & Update step, the Normalization step, the Resampling step and the Averaging step, as described in Algorithm I, where a general PF is detailed with a single

Algorithm I
States PF Steps

For $\boldsymbol{i} = \boldsymbol{1}..\boldsymbol{N_x}$ – Prediction & Update loop

(a) Propagate previous estimated state particles $\boldsymbol{x}_{k-1}^i = \hat{\boldsymbol{x}}_{k-1}^i$ through the model $g_k(\cdot)$ to obtain predicted states particles $\bar{\boldsymbol{x}}_k^i = g_k\left(\hat{\boldsymbol{x}}_{k-1}^i, \boldsymbol{u}_k, \boldsymbol{\omega}_k\right)$

(b) Sample proposal state particles from predicted state particles with a variance vector $\boldsymbol{q}_x$ associated with $\boldsymbol{\omega}_k$, e.g., using a normal distribution $\tilde{\boldsymbol{x}}_k^i \sim N(\bar{\boldsymbol{x}}_k^i, \boldsymbol{q}_x)$

(c) Calculate weights of proposal state particles using a weighting function for updating the particles, e.g., using a Gaussian function $W_{x_k}^i = \frac{1}{\sqrt{2\pi\sigma_x}} e^{\frac{\left(y_k - h_k(\tilde{x}_k^i)\right)^2}{2\sigma_x}}$, σ_x being the variance of the measurement associated to v_k.

End

(d) Normalize weights $W_{x_k}^i = \frac{W_{x_k}^i}{\sum_{j=1}^{N_x} W_{x_k}^j}$

(e) Calculate the cumulative sum of the weights $CW_x^i = \sum_{i=1}^{N_x} W_{x_k}^i$

For $\boldsymbol{i} = \boldsymbol{1}..\boldsymbol{N_x}$ – Resampling loop

(f) Generate uniformly distributed random sample $r_u \sim U(0,1)$

For $\boldsymbol{j} = \boldsymbol{1}..\boldsymbol{N_x}$

(g) Resample state particles $\hat{\boldsymbol{x}}_k^i = \tilde{\boldsymbol{x}}_k^{(r_u \le CW_x^j)}$

End

End

(h) Calculate estimated state vector $\hat{\boldsymbol{x}}_k$ using an average function of the resampled particles: mean, mode or median.

Go to step (a) for the next iteration

available measurement y_k.

As the conventional PF is only able to estimate the states of the system, the next subsection tackles the simultaneous estimation of states and a single unknown input.

B. dPF for Simultaneous States and Unknown Input Estimation

In this work, we propose a modified version of a dPF algorithm [26], extending the PF described in Algorithm I, that estimates the states of the system (PF-S), with an additional PF for the estimation of a single unknown input (PF-UI), as shown in Algorithm II. The key point of this approach lies in the sharing of the estimations between the two PFs. Fig. 2 illustrates the block diagram of the dPF.

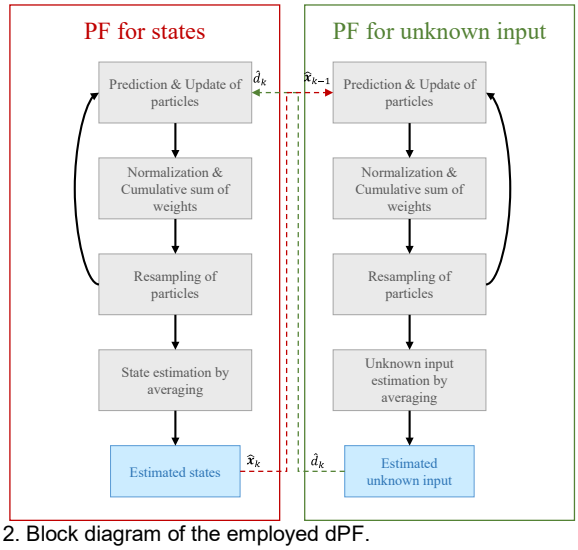

Fig. 2. Block diagram of the employed dPF.

In the dPF, the two PFs must include the estimation of the other PF in the Prediction & Update step, while keeping the rest of the steps the same. The dPF begins with the PF-UI. Let us consider a system model that can be either linear or non-linear, which is in dynamic equilibrium, with known inputs $\boldsymbol{u}_k$ and a single unknown input d_k, as in (3).

$$f_k(\boldsymbol{x}_{k-1}, \boldsymbol{u}_k, d_k, \boldsymbol{\omega}_k) = 0 \quad (3)$$

For each k iteration, the PF-UI samples N_d proposal particles for the unknown input $\tilde{d}_k^i$ based on previous estimations of both unknown input particles $\hat{d}_{k-1}^i$ and state vector $\hat{\boldsymbol{x}}_{k-1}$ estimated by the PF-S, as in (4).

$$\tilde{d}_k^i \sim p\left(\tilde{d}_k \middle| \hat{d}_{k-1}, \hat{\boldsymbol{x}}_{k-1}\right) \quad i = 1,2,..N_d \quad (4)$$

To calculate the likelihood weight $W_{d_k}^i$ for each $\tilde{d}_k^i$ proposal particle, both $\tilde{d}_k^i$ and $\hat{\boldsymbol{x}}_{k-1}$ are propagated through the dynamic equilibrium model (3), obtaining a deviation Φ_k^i each $\tilde{d}_k^i$ produces, as in (5).

$$f_k\left(\hat{\boldsymbol{x}}_{k-1}, \boldsymbol{u}_k, \tilde{d}_k^i, \boldsymbol{\omega}_k\right) = \Phi_k^i \quad (5)$$

Then, $W_{d_k}^i$ is evaluated based on Φ_k^i and increases in value as Φ_k^i approximates to 0 (6).

$$W_{d_k}^i \sim p\left(\Phi_k \middle| \tilde{d}_k, \hat{\boldsymbol{x}}_{k-1}\right) \tag{6}$$

Once the rest of the PF-UI steps are computed, the PF-S samples its proposal particles $\tilde{\boldsymbol{x}}_k^i$ using the current iteration's predicted particles $\bar{\boldsymbol{x}}_k^i$. The predicted particles are obtained by propagating the previous iteration's $\hat{\boldsymbol{x}}_{k-1}^i$ and the current iteration's estimated unknown input $\hat{d}_k$, as in (7).

$$\begin{aligned} \bar{\boldsymbol{x}}_k^i &= g_k\left(\hat{\boldsymbol{x}}_{k-1}^i, \boldsymbol{u}_k, \hat{d}_k, \boldsymbol{\omega}_k\right) \\ \tilde{\boldsymbol{x}}_k^i &\sim p(\tilde{\boldsymbol{x}}_k | \bar{\boldsymbol{x}}_k) \;\; i = 1,2,..N_x \end{aligned} \tag{7}$$

The weights $W_{x_k}^i$ are then calculated, as in Algorithm I, taking advantage of the available state measurement.

ALGORITHM II
UNKNOWN INPUT PF STEPS

For $i = 1..N_d$ – Prediction & Update loop

(a) Sample proposal unknown input particles from estimated unknown input and a variance q_d , e.g., $\tilde{d}_k^i \sim N(\hat{d}_{k-1}^i, q_d)$. q_d is associated to the maximum value d_k can get.

(b) Propagate proposal unknown input particles $\tilde{d}_k^i$ and the previous estimation of $\hat{\boldsymbol{x}}_{k-1}$ through the dynamic equilibrium model to get the deviation $\Phi_k^i = f_k\left(\hat{\boldsymbol{x}}_{k-1}, \boldsymbol{u}_k, \tilde{d}_k^i, \boldsymbol{\omega}_k\right)$.

(c) Calculate weights of proposal particles $\tilde{d}_k^i$ using a weighting function for updating the particles, e.g., $W_{d_k}^i = \frac{1}{\sqrt{2\pi\sigma_d}} e^{-\frac{\left(0-\Phi_k^i\right)^2}{2\sigma_d}}$. σ_d is associated to the maximum value Φ_k^i can have.

End

(d) Normalize weights $W_{d_k}^i = \frac{W_{d_k}^i}{\sum_{j=1}^{N_d} W_{d_k}^j}$

(e) Calculate the cumulative sum of the weights $CW_d^i = \sum_{i=1}^{N_d} W_{d_k}^i$

For $i = 1..N_d$ – Resampling loop

(f) Generate uniformly distributed random sample $r_u \sim U(0,1)$

For $j = 1..N_d$

(g) Resample updated particles $\hat{d}_k^i = \tilde{d}_k^{(r_u \le CW_d^j)}$

End

End

(h) Calculate estimated unknown input $\hat{d}_k$ using an average function of the resampled particles: mean, mode or median.

Go to step (a) for the next iteration

Regarding the servo press use case, the dynamic equilibrium model that allows the evaluation of the unknown input F_{stroke_k} is shown in (8).

$$m_k(\theta_{k-1}, \dot{\theta}_{k-1}, \boldsymbol{\omega}_k)\ddot{\theta}_{k-1} - \left(\tau_{e_k} - \tau_{fric_k} - f_{\theta_k}\left(F_{lb_k}, F_{fric_k}, F_{stroke_k}\right)\right) = 0 \tag{8}$$

Fig. 3 depicts the block diagram of the estimation procedure for the servo press model, showing the signals used to feed the model and the estimations obtained by means of the dPF.

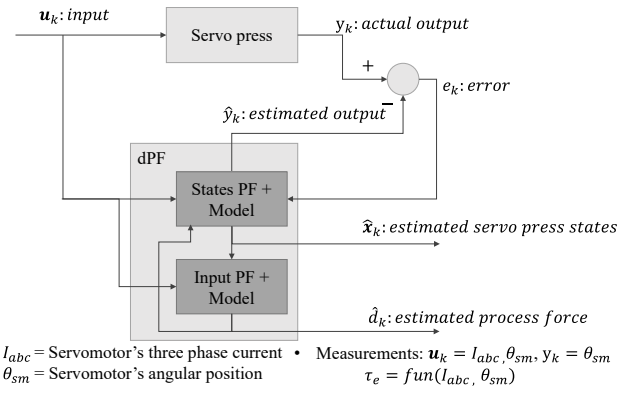

Fig. 3. Block diagram of the estimation procedure.

The next section tackles the tuning of the number of particles and the variance vectors of the dPF, as well as the acceleration of the algorithm by means of an FPGA-based hardware implementation.

IV. TUNING AND ACCELERATION OF THE DPF

A. Tuning of the dPF

Some parameters of the dPF must be tuned according to the available measurements and the characteristics of the model it is interacting with. The variance vectors used in the weighting function $\boldsymbol{\sigma}_x, \boldsymbol{\sigma}_d$ and variance vectors used in the proposal sampling $\boldsymbol{q}_x, \boldsymbol{q}_d$ must be tuned based on the possible variance of the estimation magnitudes during the working cycle of the servo press.

Moreover, the number of particles for both PF-UI and PF-S, N_x and N_d provide a trade-off between computational cost and the quality of the estimate. Different combinations of particle numbers have been tested to find the minimum values for N_x and N_d that provide an acceptable estimation of the process force. The utilized metric is given by (9).

$$e = \sqrt{\frac{\sum_{k=1}^{L}\left(S_{F_k}\left(\hat{d}_k - F_{stroke_k}\right)\right)^2}{L}} \tag{9}$$

This metric captures the mean error (e) in the estimated process force, weighted by the actual position of the force, giving higher weight at the maximum force point that is located at the bottom dead center (180º of the angular position of the crankshaft), which is critical to the safety of the machine. L is the number of recorded k samples, and S_F is the scaling factor employed. Fig. 4 shows the application of the mentioned metric graphically. The scaling factor is depicted by means of the green arrow, which emphasizes an incremental importance as the estimation approaches 180º.

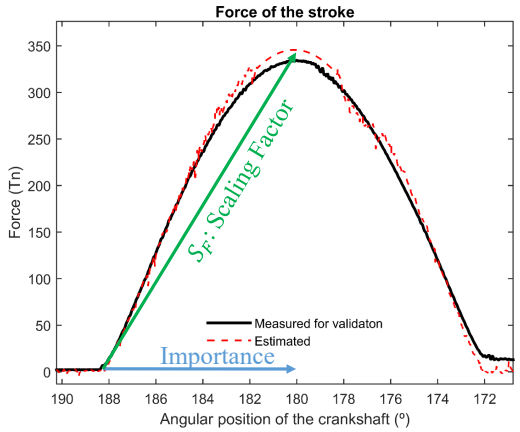

Fig. 4. Estimated and measured process forces.

The dPF algorithm was executed 100 times for each particle combination in four experiments to obtain the mean error value of the executions. The evaluation underlined the influence of the values of N_x and N_d in the estimations. Fig. 5 shows the estimation error defined by (9) for all the tested combinations of N_x and N_d. It can be noted that N_d has a bigger impact than N_x in the estimation error. Based on this graph, the combination of $N_d = 1000$ and $N_x = 300$ was selected, since the error obtained with larger values of N_d and

N_x did not improve the estimation error significantly, while they increased the computational cost of the dPF.

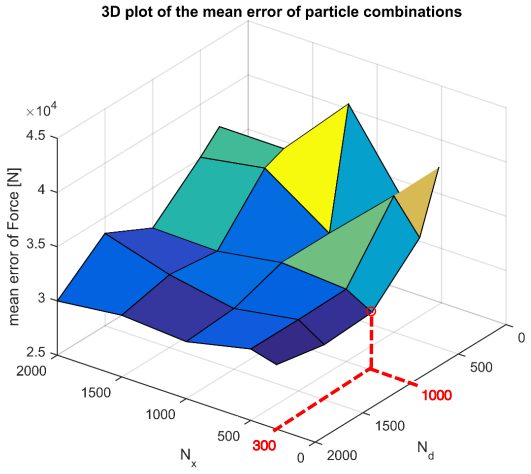

Fig. 5. Three-dimensional plot of the estimation error evaluation for different N_x and N_d combinations.

The sampling time used for the data acquisition was 250 µs for experiments of 10 s. The dPF was executed using a personal computer (PC) equipped with an Intel® Core™ i7-4700MQ central processing unit (CPU) using MATLAB® in place of the servo press PC used for loading metal forming programs. On average, the execution of the dPF for a single machine cycle under the chosen N_x and N_d takes 1012 s, which corresponds to 25.3 ms to process each sample. Therefore, this execution time does not permit a real-time estimation of the process force and states of the servo press, since it is more than the sampling time.

The next section addresses the development of a system that accelerates the proposed algorithm to allow a real-time estimation of both the process force and the states of the servo press.

B. Hardware Selection and Methodology

As already mentioned, the dPF may be subject to high computational cost due to the number of particles used in the algorithm, which makes the dPF time-consuming. This is because the CPU implementation operates on the particles sequentially in the Prediction & Update and the Resampling loops, although the particles do not have any dependencies upon one another.

The independence of the particles opens the possibility of executing the iterations of those calculations in parallel by using modern hardware solutions that employ parallelization.

Multicore CPUs, graphics processing units (GPU) and FPGAs are able to carry out operations in parallel. According to Mingas et al., multicore CPUs provide limited parallelism per chip and were originally manufactured for sequential code, so they cannot tackle applications that demand a high degree of parallelism [27]. GPUs and CPUs have fixed hardware architectures, while FPGAs are reconfigurable, which facilitates custom hardware architecture. A GPU has a predefined amount of on-chip memory per processing core, while an FPGA provides more flexibility, as it can allocate a custom amount of on-chip memory to each processing block [28]. When data is stored on-chip, communication between the various hardware blocks of the device is faster than when it is stored off-chip, due to the latency associated with off-chip memory communication. In recent years, hardware manufacturers have moved towards hybrid solutions that combine some of the mentioned hardware solutions to exploit their benefits in a single device. These devices allow sequential code to run on traditional CPUs and the parallelizable code to run in FPGAs to accelerate execution. In this context, the hardware implementation of the dPF algorithm was tested and validated by means of the Xilinx Zynq UltraScale+ MPSoC ZCU106 Evaluation Kit, which integrates a quad-core ARM® Cortex™ applications processor and an FPGA.

The global hardware architecture of the algorithm is presented in Fig. 6. The dPF algorithm was implemented exclusively in the programmable logic (PL), but the estimated unknown input and states were shared through the processing system (PS), and those results were simultaneously extracted from the PS universal asynchronous receiver–transmitter (UART) for verification. The Gaussian and uniform random number generators (RNG) were also implemented in the FPGA.

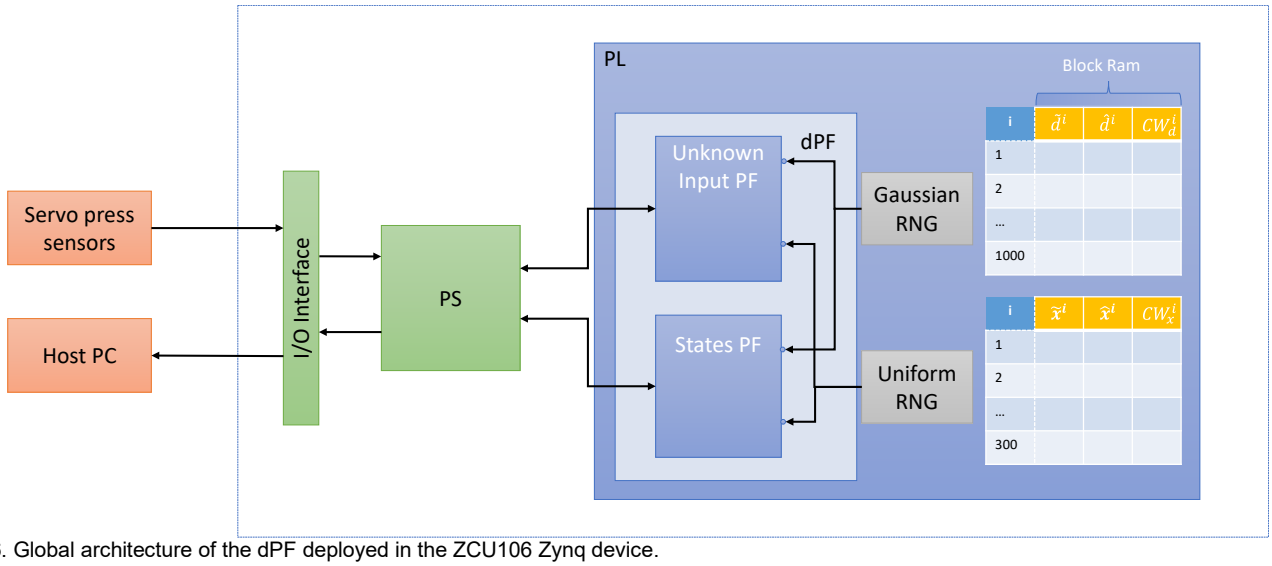

Fig. 6. Global architecture of the dPF deployed in the ZCU106 Zynq device.

The parallelization of the mentioned dPF loops was carried out through pipelining. Pipelining is a technique of executing subsequent instructions or commands that overlap in time. In a pipelined scheme, the received data is processed in a different hardware block at each clock cycle. Fig. 7(a) and Fig. 7(b) show a non-pipelined and a pipelined series of instructions, respectively and emphasizes the number of clock cycles used in both processing schemes.

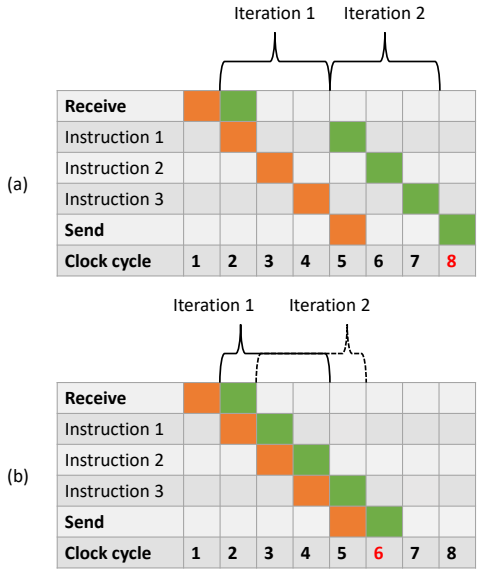

Fig. 7. Illustrations of two data processing schemes: (a) non-pipelined series of instructions and (b) pipelined series of instructions.

The time consumed by a pipelined loop is given by (10):

$$t_t = l_i + t_e \tag{10}$$

where t_t is the total execution time consumed processing the pipelined instructions, which is given by the initial latency l_l and the execution time of the instructions t_e.

When designing a hardware architecture, two aspects must be taken into account: the required latency of the system and the available hardware resources on the FPGA. The required latency is the time period in which the algorithm has to process the received data to produce and output. If real-time data processing is required, the required latency will be defined by the sampling time of the data acquisition. The resource consumption of the hardware implementation is tightly related to the time requirements established by the required latency. The faster (i.e. fewer clock cycles) the processing, the more resources are used. Therefore, the designer of the architecture must try to optimize the design to satisfy the latency time by using a minimum of resources on the FPGA.

Table I contains the FPGA's resource utilization and the total execution time of the two PFs involved in the dPF after the placement of the generated hardware blocks and the routing of the circuits carried out in the hardware implementation. The hardware implementation of the dPF was carried out using Vivado HLS®. The required time was defined by the sampling time of the data acquisition $t_s = 250\ \mu s$, and the analysed resources are lookup tables (LUT), flip flops (FF), digital signal processing (DSP) modules and block RAM (BRAM). The first three are computational resources, i.e. they are used to implement the computations of the algorithm, while the last is a memory resource, i.e. on-chip memory. It should be noted that single precision arithmetic was used throughout the algorithm. Arithmetic precision also has an impact on resource utilization. As the arithmetic precision increases, more resources are utilized for computing and data storage. The resource utilization presented in Table I achieves a faster execution time than the required latency, which guarantees that the pursued real-time estimation of the mentioned magnitudes. This was not achieved by software processing, as shown in Table I.

TABLE I
POST PLACE & ROUTE RESOURCE UTILIZATION AND EXECUTION TIME SPECIFICATIONS

Piece of Code	LUTs	FFs	DSPs	BRAM	Execution Time
Unknown Input PF	46176	30101	353	4	115.35 µs
States PF	48581	31712	353	4	37.73 µs
Total	94757	61813	706	8	153.08 µs
Available on Board	230400	460800	1728	624	
Percentage	**41.13%**	**13.41%**	**40.85%**	**1.28%**	
Execution time in software					25.3 ms
Required latency					< 250 µs

C. Resampling step Modification for Hardware Implementation

To predict the execution time of the implemented algorithm to guarantee the required latency, the code must be deterministic. For that reason, step (g) of the Resampling loops presented in Algorithm I and Algorithm II must be modified, since its operation is a non-deterministic **while** loop.

A bounded binary search resampling is raised to make step (g) deterministic, as shown in Table II.

TABLE II
BOUNDED BINARY SEARCH RESAMPLING

(g)
Initialize $L = 1, R = N_x$
For $\boldsymbol{n = 1..13}$
  Set: $p = floor(L + R/2)$
  if $r_u > CW_x^p$ **then** $L = p$
  else if $r_u < CW_x^p$ **then** $R = p$
  else if $r_u == CW_x^p$ **then** $p = p - 1$
End
Copy the sampled p particle in the ith position of particles vector: $\hat{\boldsymbol{x}}_k^i = \tilde{\boldsymbol{x}}_k^{(p)}$

The number of iterations of the **For** loop was set iteratively, while checking that the binary search converged to a single particle number represented by p. The scheme shown in Table II was also applied to the resampling of the unknown input particles.

V. EXPERIMENTAL RESULTS

Four experiments were carried out in the real servo press to validate the proposed dPF. Four different tonnage strokes were applied, and the servomotor's currents I_{abc} and angular position θ_{sm} were measured to obtain τ_e through Park's transform [29]. Crankshaft's angular position θ was used as a measured output in the dPF, obtained through $\theta = \theta_{sm}/\eta$, where η is the reduction ratio of the gearbox. The validation

force signal was measured through four Vario® force piezoelectric sensors of Brankamp® located on each column of the frame of the servo press, as shown in Fig. 1.

The variance coefficients were defined, as shown in (11).

$$\sigma_x = 0.1 \qquad \sigma_d = 4 \cdot 10^8$$
$$\boldsymbol{q}_x = \begin{bmatrix} 8 \cdot t_s^4 \\ 8 \cdot t_s^2 \\ 8 \end{bmatrix} \qquad q_d = [4 \cdot 10^8] \tag{11}$$

where t_s is the sampling time interval of the CNC.

Fig. 8 shows the estimation results of two experiments with the mentioned combination of particles. Table III collects the estimation results and estimation errors achieved regarding the process force and the energy of the four experiments at the maximum force position at 180º of the angular position of the crankshaft.

TABLE III
RELATIVE ESTIMATION ERROR WITH RESPECT TO THE MEASUREMENTS

Experiment	Force at 180º			Energy at 180º		
	Estimated	Measured	Error	Estimated	Measured	Error
(a)	343.33 Tn	334.51 Tn	2.64%	3676.0 J	3609.0 J	1.88%
(b)	235.35 Tn	227.20 Tn	3.60%	1730.0 J	1702.0 J	1.66%
(c)	193.91 Tn	188.29 Tn	2.98%	1205.0 J	1189.0 J	1.35%
(d)	90.23 Tn	88.47 Tn	1.99%	279.3 J	273.6 J	2.08%

VI. CONCLUSIONS AND FUTURE WORKS

This paper presents a method to estimate the process force and the states of a servo press through a novel dPF construction. The proposed algorithm takes advantage of the already available signals used by the CNC, thereby avoiding the need to introduce new sensors to the system. The results show that the estimated states and process force approximate well the measured validation signals. The hardware implementation of the dPF allows a real-time estimation of the press force and states, allowing the system to carry out fast corrective actions.

Regarding the metal forming industry, the proposed work could give rise to control strategies that act on the processes to improve production and the quality of the formed parts. Furthermore, the estimation of the press forces and states can also reveal information about the condition of the machine, thus allowing machine failures and downtime to be prevented. Moreover, the developed algorithm and its hardware implementation can be ported to other machines and processes by modifying the underlying model while keeping the rest of the infrastructure the same.

APPENDIX A

Fig. 9 illustrates the schematic design of the servo press.

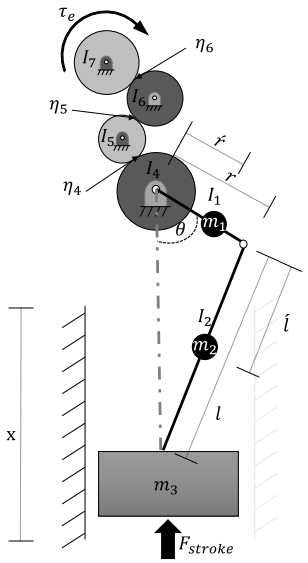

Fig. 9. Schematic representation of the mechanical components of the servo press.

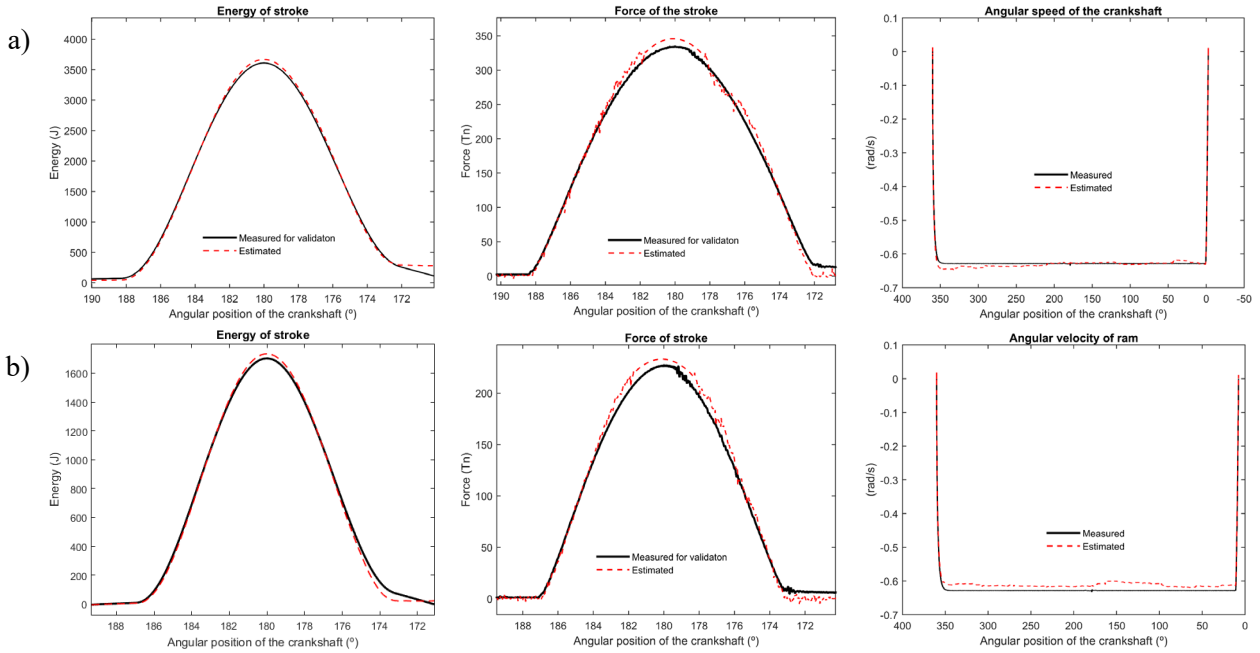

Fig. 8. Estimation results for two experiments recorded in the real servo press. The energy and force of the stroke and the angular speed of the crankshaft are shown: a) Experiment 1 and b) Experiment 2.

The complete model of the servo press is shown in (12), where the general torque equation is described. The model was developed based on works presented in [24] and [25].

$$\begin{aligned}
&\ddot{\theta}\left(I_1 + m_1\acute{r}^2 + \frac{I_2 r^2 cos^2\theta}{c^2} + \frac{(l-\acute{l})^2}{l^2} m_2 r^2 cos^2\theta\right.\\
&+ sin^2\theta\left(m_2 r^2 \frac{(r\acute{l}\cos\theta + lc)^2}{l^2 c^2} + m_3 r^2 \frac{(r\cos\theta + c)^2}{c^2}\right) + I_4\\
&\left.+ I_5\eta_4^2 + I_6(\eta_4\eta_5)^2 + I_7(\eta_4\eta_5\eta_6)^2\right)\\
&+ \frac{1}{2}\dot{\theta}^2\left(\frac{2I_2 r^4 cos^3\theta\, sin\theta}{c^4} - \frac{2I_2 r^2 cos\theta\, sin\theta}{c^2}\right.\\
&- \frac{2m_2 r^2 cos\theta\, sin\theta\,(l-\acute{l})^2}{l^2}\\
&+ 2\,sin\theta\,cos\theta\left(m_2 r^2 \frac{(r\acute{l}\cos\theta + lc)^2}{l^2 c^2} + m_3 r^2 \frac{(r\cos\theta + c)^2}{c^2}\right)\\
&+ 2sin^2\theta\left(\frac{m_2 r^3 \acute{l}\, sin\theta\,(r\acute{l}\cos\theta + lc)(r\cos\theta - c)(r\cos\theta + c)}{l^2 c^4}\right.\\
&\left.\left.+ \frac{m_3 r^3\, sin\theta\,(r\cos\theta - c)^2(r\cos\theta + c)^2}{c^4}\right)\right)\\
&+ \mathrm{g}\left(sin\theta\,(m_1\acute{r} + m_2 r + m_3 r) + sin\theta\,cos\theta\left(\frac{m_2 r^2\acute{l}}{lc} + \frac{m_3 r^2}{c}\right)\right)\\
&= \tau_e\eta - \tau_{fric} - \left(F_{stroke} + F_{lb} + F_{fric}\right) r\,sin\theta\left(1 + \frac{r\cos\theta}{c}\right)
\end{aligned} \tag{12}$$

All symbols in (12) are defined in Table IV, and the values of the constants were obtained from a computer aided design (CAD).

TABLE IV
LIST OF SYMBOLS

$\theta, \dot{\theta}, \ddot{\theta}\ (rad, \frac{rad}{s}, \frac{rad}{s^2})$	Angular position, speed and acceleration of the crankshaft.
$\mathrm{g} = 9.81\frac{m}{s^2}$	Gravitational acceleration constant.
$I_1 = 80.72\frac{kg}{m^2}$	Inertia of the crankshaft.
$m_1 = 424\ kg$	Mass of the crankshaft.
$I_2 = 132.72\frac{kg}{m^2}$	Inertia of the connecting rods.
$m_2 = 825.94\ kg$	Mass of the connecting rods.
$m_3 = 11600\ kg$	Mass of the ram.
$I_4 = 105.77\frac{kg}{m^2}$	Inertia of the 1st gear of the gearbox.
$I_5 = 19.007\frac{kg}{m^2}$	Inertia of the 2nd gear of the gearbox.
$I_6 = 3.58\frac{kg}{m^2}$	Inertia of the 3rd gear of the gearbox.
$I_7 = 7.38\frac{kg}{m^2}$	Inertia of the servomotor axis.
$\eta_4 = \frac{60}{17}$	Reduction ratio of the 1st gear of the gearbox.
$\eta_5 = \frac{60}{17}$	Reduction ratio of the 2nd gear of the gearbox.
$\eta_6 = \frac{71}{18}$	Reduction ratio of the 3rd gear of the gearbox.
$\eta = \eta_4\eta_5\eta_6$	Reduction ratio of the gearbox.
$r = 0.2\ m$	Radius of the crankshaft
$\acute{r} = 0.07621\ m$	Distance between crankshaft's rotary axis and its mass center.
$l = 1.05\ m$	Length of the connecting rod.
$\acute{l} = 0.32703\ m$	Distance between connecting rod's rotary axis and its mass center.
$\tau_e(Nm)$	Electric torque of the servomotor.
$\tau_{fric}(Nm)$	Friction torque of the servo press. $\tau_{fric} = sgn(\dot{\theta})((32\eta)^{-\frac{\dot{\theta}}{5000}} - 3400\dot{\theta})$
$F_{stroke}(N)$	Force of the servo press.
$F_{lb}(N)$	Force of the load balancer. $F_{lb} = ((3\mathrm{x})2900 + m_3)grsin(\theta)\left(1 + \frac{rcos(\theta)}{c}\right)$
$F_{fric}(N)$	Friction force of the guiding system. $F_{fric} = sgn(\mathrm{v})((140)^{-\frac{\mathrm{v}}{254}} - 565\mathrm{v})$
$\mathrm{x}(m)$	Linear displacement of ram.
$\mathrm{v}(\frac{m}{s})$	Linear speed of the ram.
c	$c = \sqrt{l^2 - r^2\sin^2\theta}$

ACKNOWLEDGEMENTS

REFERENCES

[1] T. Altan and A. E. Tekkaya, *Sheet metal forming: fundamentals*. Asm International, 2012.

[2] R. Kumme, O. Mack, B. Bill, C. Gossweiler, H. Haab, and K. I. AG, "Dynamic properties and investigations of piezoelectric force measuring devices," *VDI Berichte*, vol. 1685, pp. 161–172, 2002.

[3] R. Doraiswami and L. Cheded, "Robust model-based soft sensor: design and application," *IFAC Proceedings Volumes*, vol. 47 DOI 10.3182/20140824-6-ZA-1003.00245, no. 3, pp. 5491–5496, 2014.

[4] S. S. Park and Y. Altintas, "Dynamic compensation of spindle integrated force sensors with Kalman filter," *Journal of Dynamic Systems, Measurement, and Control*, vol. 126, DOI 10.1115/1.1789531, no. 3, pp. 443–452, 2004.

[5] Y. Chinniah, R. Burton, and S. Habibi, "Failure monitoring in a high performance hydrostatic actuation system using the extended Kalman filter," *Mechatronics*, vol. 16, DOI 10.1016/j.mechatronics.2006.04.004, no. 10, pp. 643–653, 2006.

[6] M.-C. Tsai, E.-C. Tseng, and M.-Y. Cheng, "Design of a torque observer for detecting abnormal load," *Control Engineering Practice*, vol. 8, DOI 10.1016/S0967-0661(99)00160-4, no. 3, pp. 259–269, 2000.

[7] A. W. Reid, P. McAree, P. Meehan, and H. Gurgenci, "Longwall shearer cutting force estimation," *Journal of Dynamic Systems, Measurement, and Control*, vol. 136, DOI 10.1115/1.4026326, no. 3, pp. 031008-1-031008-9, 2014.

[8] N. Gordon, B. Ristic, and S. Arulampalam, "Beyond the kalman filter: Particle filters for tracking applications," *Artech House, London*, vol. 830, p. 5, 2004.

[9] J. V. Candy, *Bayesian signal processing: classical, modern, and particle filtering methods*, John Wiley & Sons, vol. 54, 2016.

[10] L. Zanni, S. Sarri, M. Pignati, R. Cherkaoui, and M. Paolone, "Probabilistic assessment of the process-noise covariance matrix of discrete Kalman filter state estimation of active distribution networks," in *2014 International Conference on Probabilistic Methods Applied to Power Systems (PMAPS)*, DOI 10.1109/PMAPS.2014.6960646, pp. 1–6, 2014.

[11] J. M. Karandikar, A. E. Abbas, and T. L. Schmitz, "Tool life prediction using Bayesian updating. Part 2: Turning tool life using a Markov Chain Monte Carlo approach," *Precision Engineering*, vol. 38, DOI 10.1016/j.precisioneng.2013.06.007, no. 1, pp. 18–27, 2014.

[12] C. He, J. Li, and G. Vachtsevanos, "Prognostics and health management of an automated machining process," *Mathematical Problems in Engineering*, vol. 2015, DOI 10.1155/2015/651841, pp. 1–10, 2015.

[13] J. Dahlin and F. Lindsten, "Particle filter-based Gaussian process optimisation for parameter inference," *IFAC Proceedings Volumes*, vol. 47, DOI 10.3182/20140824-6-ZA-1003.00278, no. 3, pp. 8675–8680, 2014.

[14] N. Kantas, A. Doucet, S. S. Singh, J. Maciejowski, N. Chopin, and others, "On particle methods for parameter estimation in state-space models," *Statistical science*, vol. 30, DOI 10.1214/14-STS511, no. 3, pp. 328–351, 2015.

[15] M. Imani, S. F. Ghoreishi, D. Allaire, and U. M. Braga-Neto, “MFBO-SSM: Multi-fidelity Bayesian optimization for fast inference in state-space models,” AAAI, 2019.

[16] M. Imani and U. M. Braga-Neto, “Particle filters for partially-observed Boolean dynamical systems,” *Automatica*, vol. 87, DOI 10.1016/j.automatica.2017.10.009, pp. 238–250, 2018.

[17] L. E. Olivier, B. Huang, and I. K. Craig, “Dual particle filters for state and parameter estimation with application to a run-of-mine ore mill,” *Journal of Process Control*, vol. 22, DOI 10.1016/j.jprocont.2012.02.009, no. 4, pp. 710–717, 2012.

[18] S. Mejri, A. S. Tlili, and N. B. Braiek, “Particle filter for state and unknown input estimation of chaotic systems,” *Proceedings Engineering & Technology-Vol*, vol. 4, pp. 67–72, 2013.

[19] F. Mustière, M. Bolic, and M. Bouchard, “Speech enhancement based on nonlinear models using particle filters,” *IEEE transactions on neural networks*, vol. 20, DOI 10.1109/TNN.2009.2033367, no. 12, pp. 1923–1937, 2009.

[20] S. I. Mohammad Aminul Islam Khan and F. Khan, “Simultaneous state and input estimation of non-linear process with unknown inputs using particle swarm optimization particle filter (PSO-PF) algorithm,” in *Fifth International Conference on Chemical Engineering (ICChE 2017) Process Modelling, Safety and Control*, 2017.

[21] Y. Zhang, S. Wang, and G. Ji, “A comprehensive survey on particle swarm optimization algorithm and its applications,” *Mathematical Problems in Engineering*, vol. 2015, DOI 10.1155/2015/931256, 2015.

[22] S. Liu, G. Mingas, and C.-S. Bouganis, “Parallel resampling for particle filters on FPGAs,” in *2014 International Conference on Field-Programmable Technology (FPT)*, DOI 10.1109/FPT.2014.7082775, pp. 191–198, 2014.

[23] S. Yu, Y. Liu, and L. Li, “Comparative life cycle assessment of servo press and flywheel press,” in *Re-engineering Manufacturing for Sustainability*, Springer, DOI 10.1007/978-981-4451-48-2_84, pp. 515–521, 2013.

[24] R. Halicioglu, L. C. Dulger, and A. T. Bozdana, “Modelling, design, and implementation of a servo press for metal-forming application,” *The International Journal of Advanced Manufacturing Technology*, DOI 10.1007/s00170-016-9947-8, pp. 1–12, 2016.

[25] J. Olaizola, J. M. Abete, A. Iturrospe, and E. Saenz de Argandoña, “Modelling, simulation and validation of a ram-crank servo driven press,” in *19th Scientific Convention on Engineering and Architecture*, pp. 112–123, 2018.

[26] S. Thrun, D. Fox, W. Burgard, and F. Dellaert, “Robust Monte Carlo localization for mobile robots,” *Artificial intelligence*, vol. 128, DOI 10.1016/S0004-3702(01)00069-8, no. 1–2, pp. 99–141, 2001.

[27] G. Mingas and C.-S. Bouganis, “Population-based MCMC on multi-core CPUs, GPUs and FPGAs,” *IEEE Transactions on Computers*, vol. 65, DOI 10.1109/TC.2015.2439256, no. 4, pp. 1283–1296, 2016.

[28] G. Mingas, L. Bottolo, and C.-S. Bouganis, “Particle MCMC algorithms and architectures for accelerating inference in state-space models,” *International Journal of Approximate Reasoning*, vol. 83, DOI 10.1016/j.ijar.2016.10.011, pp. 413–433, 2017.

[29] R. H. Park, “Two-reaction theory of synchronous machines generalized method of analysis-part I,” *Transactions of the American Institute of Electrical Engineers*, vol. 48, DOI 10.1109/T-AIEE.1929.5055275, no. 3, pp. 716–727, 1929.