Parallel Numerical Optimization for Fast Adaptive Nonlinear Moving Horizon Estimation

Tomáš Polóni, Boris Rohaľ-Ilkiv and Tor Arne Johansen

Abstract— **This paper proposes a novel strategy using parallel optimization computations for nonlinear moving horizon state estimation, and parameter identification problems of dynamic systems. The parallelization is based on the multi-point derivative-based Gauss-Newton search, as one of the most efficient algorithms for the nonlinear least-square problems. A numerical experiment is performed to demonstrate the parallel computations with the comparison to sequential computations.**

I. Introduction

Moving horizon estimation (MHE) of states in nonlinear systems is considered. This is a powerful estimation approach that is shown to have superior performance compared to Extended Kalman Filters (EKF) in many cases, and may have performance that is competitive to advanced nonlinear estimators such as particle filters (PF) and the Unscented Kalman Filter (UKF). Despite the merits of MHE, its practical applicability is limited by the need for a nonlinear numerical optimization algorithm to be executed at each sampling instant [1], [2], [3]. This may require powerful computers, in particular compared to EKF and UKF, or may limit the applicability to systems of relatively low order or slow sampling.

When selecting an algorithm for solving the MHE optimization problem at each sample, the following considerations can be made

- Computations must be done in real time at each sampling instant, and fast deterministic execution time is a high priority.
- The unknown state and the optimal estimate changes from one sample to the next. Using the optimization algorithm output from the previous sample should be used to initialize the optimization at the current sample, a procedure commonly known as warm start, since the changes from one sample to the next are usually reasonably small.
- In order to prioritize the requirement for fast and deterministic execution of the algorithm it may be acceptable that the optimum is tracked asymptotically and an optimal estimate is not computed exactly at each sample (this is commonly known as sub-optimal or approximate solution approaches).
- Although the algorithm requirement is relaxed to track the optimum only asymptotically, it should at the same time have a fast transient response. In some applications it is required that, the algorithm should adapt quickly to discontinuous jumps in states (e.g. due to unknown time-varying parameters being modeled as states in an augmented state space model) and initial errors.
- The algorithm should, as far as possible, be robust to initializations that may lead to local minima being tracked instead of global minima.
- The algorithm should be scalable such that it can be efficiently implemented on a range of multi-core processing architectures ranging from multi-core processors to massively parallel hardware such as GPUs and FPGAs.

The computational burden of a nonlinear optimization algorithm can be broken down, and the efficiency can be gained, by the problem redistribution to parallel processors/cores. In standard parallel algorithms, each processor computes a certain problem (subproblem) with two major steps: parallelization and synchronization [4]. The synchronization step is responsible for sufficient decrease of the objective function while the parallelization step produces candidate points and candidate directions. Derivative-based parallel optimization algorithms like Block-Jacobi [4], parallel variable distribution [5], [6], parallel gradient distribution [7] and parallel variable transformation [8] aim to decompose the original problem into significantly smaller subproblems that are treated in parallel. Solvers for the class of nonlinear least-square problems based on Levenberg-Marquardt method is proposed in [9] and the multisplitting strategy is proposed in [10]. Alternative derivative-free methods rely on point-based parallelization [11], [12], with Nelder-Mead algorithm [13].

Our approach combines point-based search with derivative-based Gauss-Newton method [14], where the synchronization step consists of taking the best point found by the C processors. The best point is the one for which the cost function has lowest value [7]. In the parallelization step the optimization algorithm starts from several initial points, where each initial point is allocated to one processor, but a point cannot be shared by more than one processor. This is also similar to the globalization technique with multistart concept that is mentioned in [15], [16]. The number of iterations computed on one processor is compared with a number of processors, each computing

This work is supported by the Slovak Research and Development Agency under projects LPP-0118-09 and APVV-0090-10.

Tomáš Polóni tomas.poloni@stuba.sk and Boris Rohaľ-Ilkiv boris.rohal-ilkiv@stuba.sk are with the Institute of Automation, Measurement and Applied Informatics at the Faculty of Mechanical Engineering, Slovak University of Technology, Bratislava, Slovakia

Tor Arne Johansen tor.arne.johansen@itk.ntnu.no is with the Department of Engineering Cybernetics at the Norwegian University of Science and Technology, Trondheim, Norway

one iteration. This is discussed for the moving horizon state and parameter estimation problem of the simulated nonlinear dynamic system. Published moving horizon observers (MHO) optimize the initial state of a model at the beginning of a moving data window [1], [2], [3]. The advantage of the proposed moving horizon observer is that the uncertainty and process noise is implicitly captured by the EKF pre-filtering and post-filtering, without introducing additional variables in optimization, which makes it more suitable for the fast mechatronic systems [17].

II. Model formulation

For the purpose of state estimation and parameter identification a dynamic system is described by a general continuous time-invariant augmented state-space equations

$$\dot{x}_s = \tilde{f}_c(x_s, p, u) + z_s \quad (1)$$

$$\dot{p} = z_p \quad (2)$$

where $\tilde{f}_c : \mathbb{R}^{n_{x_s}} \times \mathbb{R}^{n_p} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_{x_s}}$, $x_s \in \mathbb{R}^{n_{x_s}}$ is a state vector and $p \in \mathbb{R}^{n_p}$ is a vector of parameters. The state process noise vector is $z_s \in \mathbb{R}^{n_{x_s}}$ and the parameter process noise vector is $z_p \in \mathbb{R}^{n_p}$. Eq. (1) and (2) can be combined as

$$\dot{x} = f_c(x, u) + z \quad (3)$$

where $f_c : \mathbb{R}^{n_{x_s}+n_p} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_{x_s}+n_p}$ represents the augmented dynamics and $z = [z_s, z_p]^T$. The observation equation may be written as

$$y = h_c(x, u) + v \quad (4)$$

where $y \in \mathbb{R}^{n_y}$ is a vector of measurements and $h_c : \mathbb{R}^{n_{x_s}+n_p} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_y}$ is a continuous measurement function. The measurement errors are modeled with the noise term $v \in \mathbb{R}^{n_y}$. The most frequent situation encountered in practice is when the system is governed by continuous-time dynamics and the measurements are obtained at discrete time instances. For the problem formulation we consider the numerically discretized dynamic nonlinear system described by the equations

$$x_{t+1} = f(x_t, u_t) + z_t \quad (5)$$

$$y_t = h(x_t, u_t) + v_t \quad (6)$$

for $t = 0, 1, \ldots$, where $x_t \in \mathbb{R}^{n_x}$ is the augmented state vector, $u_t \in \mathbb{R}^{n_u}$ is the input vector and $z_t \in \mathbb{R}^{n_x}$ is the process noise vector. The state vector is observed through the measurement equation (6) where $y_t \in \mathbb{R}^{n_y}$ is the observation vector and $v_t \in \mathbb{R}^{n_y}$ is a measurement noise vector.

III. Moving horizon observer

In the basic moving horizon estimation formulation the statistics of the process and measurement noises z_t, v_t are assumed unknown. The function composition as the application of one function to the results of another like $f(f(x_{t-N}, u_{t-N}), u_{t-N+1})$ and $h(f(x_{t-N}, u_{t-N}), u_{t-N+1})$ can be written as $f^{u_{t-N+1}} \circ f^{u_{t-N}}(x_{t-N})$ and $h^{u_{t-N+1}} \circ f^{u_{t-N}}(x_{t-N})$ respectively, where "$\circ$" denotes function composition. The $N+1$ subsequent measurements of the outputs Y_t and inputs U_t up to time t with the $N+1$ measurement noise vector V_t is

$$Y_t = \begin{bmatrix} y_{t-N} \\ y_{t-N+1} \\ \vdots \\ y_t \end{bmatrix}; U_t = \begin{bmatrix} u_{t-N} \\ u_{t-N+1} \\ \vdots \\ u_t \end{bmatrix}; V_t = \begin{bmatrix} v_{t-N} \\ v_{t-N+1} \\ \vdots \\ v_t \end{bmatrix} \quad (7)$$

where $t = N+1, N+2, \ldots$. Neglecting process noise in the basic MHO formulation, the following algebraic map is defined

$$H_t(x_{t-N}) = \begin{bmatrix} h^{u_{t-N}}(x_{t-N}) \\ h^{u_{t-N+1}} \circ f^{u_{t-N}}(x_{t-N}) \\ \vdots \\ h^{u_t} \circ f^{u_{t-1}} \circ \ldots \circ f^{u_{t-N}}(x_{t-N}) \end{bmatrix}$$

$$Y_t = H_t(x_{t-N}) + V_t \quad (8)$$

Define the N-information vector at time t

$$I_t = [y_{t-N}^T, \ldots, y_t^T, u_{t-N}^T, \ldots, u_t^T]^T \quad (9)$$

The observer design problem is to reconstruct the vector x_{t-N} based on the information vector I_t. The basic formulation of such a problem is defined as the inverse mapping of Eq. (8). The unique existence and continuity of the solution depends on the function H_t. If Eq. (8) does not have a unique solution, the problem is ill-posed according to definitions of [18]. The solution vector x_{t-N} is in the case of uniform observability given by an over-determined set of full rank algebraic equations, where there are more equations than unknowns for which $n_x \leq N n_y$. The formulation can be under-determined if there is loss of rank if no persistence of excitation is present, or the system is not observable [19].

The cost function of the MHO optimization problem is in the meaning of the least-squares method defined as

$$J_t(\hat{x}_{t-N|t}, I_t) = \|\hat{x}_{t-N|t} - \bar{x}_{t-N|t}\|_S^2 + \|\hat{Y}_t - Y_t\|_W^2 \quad (10)$$

where

$$\hat{Y}_t = H_t(\hat{x}_{t-N|t}) \quad (11)$$

The cost function (10) comprises two squared norms where the first norm is weighted by the S matrix. The contribution of the N-step model response to the optimized vector $\hat{x}_{t-N}$ is expressed through the second norm weight matrix W. The first term in the given formulation can be viewed as an estimate of the arrival cost [3], [2].

The a priori state estimate used in the arrival cost at the beginning of the horizon is declared as $\bar{x}_{t-N|t}$ and is computed in a time instant t for the time instance $t-N$ by pre-filtering with an EKF [2]. The EKF is running at the beginning of horizon on the output data y_{t-N} which were measured in the $t-N$ time instance. This is the information which corrects the one-step prediction

$$\hat{x}_{t-N|t}^- = f(\hat{x}_{t-N-1|t-1}, u_{t-N-1}) \quad (12)$$

The a priori state estimate at the beginning of the horizon is computed as

$$\bar{x}_{t-N|t} = \hat{x}_{t-N|t}^- + K_{t-N|t}[y_{t-N} - h(\hat{x}_{t-N|t}^-)] \quad (13)$$

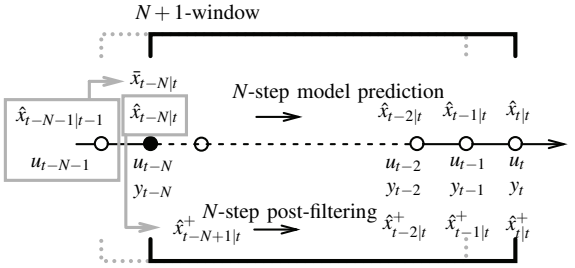

Fig. 1. Time sequences of state, input and output variables in $N+1$ Moving Horizon window

The covariance matrix is computed as

$$P^+_{t-N|t} = [I - K_{t-N|t}L_{t-N|t}]P^-_{t-N|t}[I - K_{t-N|t}L_{t-N|t}]^T + \\ + K_{t-N|t}M_{t-N|t}R_tM^T_{t-N|t}K^T_{t-N|t} \quad (14)$$

The other matrix computations necessary for the pre-filtering are done via regular EKF equations as explained in Appendix I (index t changes to $t-N|t$ and index $t-1$ changes to $t-N-1|t-1$). The only difference is that the EKF equations here are applied for the first time instance $t-N$ of the receding window. Note that with this pre-filtering the stochastic properties of the process noise and measurement noise are assumed known. Also note that neglecting process noise in (11) may lead to accuracy loss, and is made for reduced computations. The post-filtering using the EKF is further performed to propagate the estimate $\hat{x}_{t-N|t}$ to current time instance $\hat{x}^+_{t|t}$.

The schematic time sequence of the a priori state estimate vector $(\bar{x})$, state estimate vectors $(\hat{x})$, post-filtered state estimate vectors $(\hat{x}^+)$, input (u) and output (y) vectors on N-horizon are in Figure 1. The MHO algorithm, schematically shown in Figure 2 consists of three main computation parts: Pre-filtering, Optimizer, and N-step post-filtering. The Optimizer contains N-step model prediction and Cost function minimization blocks. The main computation engine is the optimization algorithm that performs the cost function minimization. The MHO algorithm with the pre-filtering and post-filtering can be summarized into following steps:

`Input`: Load the initial Datapool with measurement data and input data.

`Step 1`: Obtain the actual output measurement y_t, input u_t and update the Datapool.

`Step 2`: If the current time instance is $t = N+1$, set the initial values for $\hat{x}^+_{0|N}$ and $P^+_{0|N}$ (as in the case of EKF Eq. (27),(28)). Then according to Figure 1: $\hat{x}^+_{0|N} = \hat{x}_{0|N}$ (this value is set by the user)
Else for $t > N+1$: $\hat{x}^+_{t-N-1|t-1} = \hat{x}_{t-N-1|t-1}$ (this value is set from the last optimization run)

`Step 3`: Compute the a priori estimate with Eq. (12)

`Step 4`: Numerically integrate $P^-_{t-N|t}$ (as in the case of EKF Eq. (30) through Eq. (32))

`Step 5`: Compute the EKF gain matrix $K_{t-N|t}$ (as in the case of EKF according to Eq. (33))

`Step 6`: Compute the a posteriori state estimate $\hat{x}^+_{t-N|t}$ with Eq. (13) where $\hat{x}^+_{t-N|t} = \bar{x}_{t-N|t}$ (Block in Figure 2: Pre-filtering)

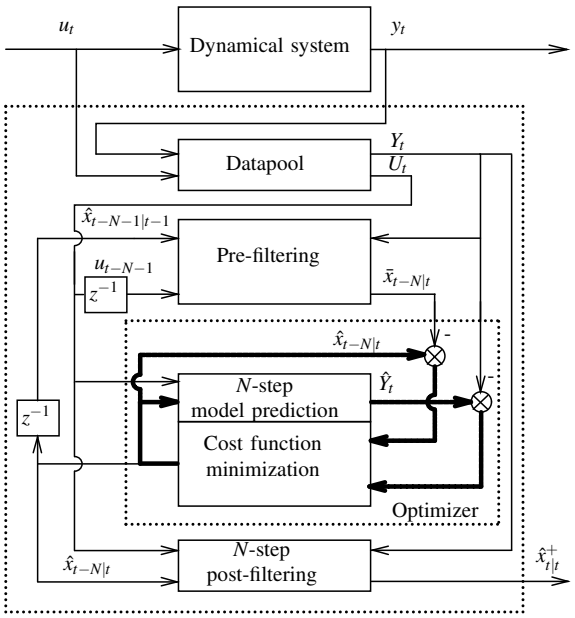

Fig. 2. Algorithm scheme of Moving Horizon Observer where z^{-1} is a one sample delay operator

`Step 7`: Minimize the cost function (10) to numerically compute the optimal state vector at the beginning of receding window $\hat{x}_{t-N|t}$. In the minimization routine the model is used through the Eq. (11). The initial condition for the optimization is $\hat{x}^{init}_{t-N|t} = \bar{x}_{t-N|t}$.

`Step 8`: Use the EKF to estimate the state from the beginning of receding window to the end of receding window as

$$\hat{x}^+_{t-N+i|t} = f_F^{u_{t-N+(i-1)}}(y_{t-N+i}, \hat{x}^+_{t-N+(i-1)|t}), \quad (15)$$

where $i = 1,2,\ldots,N$. The initial condition for the first step $(i=1)$ is $\hat{x}^+_{t-N|t} = \hat{x}_{t-N|t}$. (Block in Figure 2: N-step post-filtering)

`Step 9`: Check the cost function value $J_t(\hat{x}_{t-N|t})$ if it is below or above its threshold value δ_T. If $J_t(\hat{x}_{t-N|t}) > \delta_T$, turn off the pre-filtering in the cost function (10) through small gain in the S matrix, otherwise use the pre-filtering's nominal gain.

End of loop; Go to `Step 1`.

□

Step 9 is optional, but recommended for problems with sudden changes and jumps in states/parameters. The optimization algorithm is discussed in the next section.

IV. Optimization Algorithm

In this section, we consider the cost function $J_t(x)$ at time index t, where $x \in \mathbb{R}^{n_x}$.

`Step 1`: Generate C new initial points $\{\overline{x}^i_{t-N|t}\}_{i=1}^C = \{\overline{x}^1_{t-N|t}, ..., \overline{x}^C_{t-N|t}\}^T$ based on the pre-filtered state vector

$\overline{x}_{t-N|t}$. Use the fixed step-size Δ, to scatter the initial points $\overline{x}^i_{t-N|t}$, $i = 1 \dots C$ for the states and parameters

$$\{\overline{x}^i_{t-N|t}\}_{i=1}^{C} = \left\{ \begin{array}{c} \overline{x}_{t-N|t} \\ \overline{x}_{t-N|t} + q_1\Delta \\ \overline{x}_{t-N|t} - q_1\Delta \\ \vdots \\ \overline{x}_{t-N|t} + q_{n_x}\Delta \\ \overline{x}_{t-N|t} - q_{n_x}\Delta \end{array} \right\} \tag{16}$$

where $i = 1 \dots 2n_x + 1$ and q_k is a unit vector in a direction of $\overline{x}_{k,t-N|t}$, where $\overline{x}_{k,t-N|t}$ is a k element of vector $\overline{x}_{t-N|t}$, $k = 1 \dots n_x$.

Generate C solution candidates by the following procedure:

`Step 2`: Solve in parallel, for $i = 1,...,C$, the C Gauss-Newton-systems for the search direction $d^i_t \in \mathbb{R}^{n_x}$:

$$\nabla^2 J_t(\hat{x}^i_{t-N|t}) d^i_t = -\nabla J_t(\hat{x}^i_{t-N|t})^T F_t(\hat{x}^i_{t-N|t}) \tag{17}$$

where

$$F_t(x^i_{t-N|t}) = \begin{bmatrix} S^{1/2}(\hat{x}^i_{t-N|t} - \overline{x}_{t-N|t}) \\ W^{1/2}(Y_t - \hat{Y}_t) \end{bmatrix} \tag{18}$$

`Step 3`: In parallel, for $i = 1,...,C$, determine the step-length $\varepsilon^i_t > 0$ using a line search and make Newton-step update with Λ iterations (with the inclusion of `Step 2`):

$$\hat{x}^i_{t-N|t} = \overline{x}^i_{t-N|t} + \varepsilon^i_t d^i_t \tag{19}$$

`Step 3.1`: Pick the point $\hat{x}^1_{t-N|t},...,\hat{x}^C_{t-N|t}$ with minimum cost function value as the current estimate $\hat{x}_{t-N|t}$.

`Step 3.2`: Let $t \leftarrow t+1$, update with new data samples if available, and go to `Step 1`.

□

Remarks

- In `step 1`, the number of initial points C for the parallel computation is chosen as $C = 2n_x + 1$, however the design of Δ can be relaxed to a region where the optimal solution is likely to be found (hoping that this may lead to a negative jump in cost function value). Such relaxation can lead to a significant reduction of the initial points with lower C number. On the other side, one should maintain a significant diversity among the initial points such that there is a chance for the solution to escape if trapped in a local minimum. The design of new initial points through Eq. (16) requires the scaling of state vector x.
- In `step 2`, the gradient (or Jacobian) $\nabla J_t(\cdot)$ and the Hessian matrix $\nabla^2 J_t(\cdot)$ must be computed. Since the cost function J_t involves iterations involving the dynamic system model, it is clear that the computations are expensive and $\nabla^2 J_t(\cdot)$ is expected to be a dense matrix. Numerical finite difference calculation of $\nabla J_t(x)$ or symbolic (automatic) differentiation are typical alternatives, while a Gauss-Newton approximation for the Hessian is generally recommended since J is a least-squares type cost function $\nabla^2 J_t(\cdot) \approx A^i_t = (\nabla J_t(\cdot))^T \nabla J_t(\cdot)$. When solving the linear equation system (17), there is no particular sparseness structural properties to be exploited, but one should exploit the fact that A^i_t is symmetric and positive definite with factorization already given by $\nabla J_t(x)$, when solving (17):

$$(\nabla J_t(x^i_{t-N|t}))^T \nabla J_t(x^i_{t-N|t}) d^i_t = -\nabla J_t(x^i_{t-N|t})^T F_t(x^i_{t-N|t}) \tag{20}$$

- `Step 2` is naturally parallelized on C cores. If the number of available cores is much larger, one may achieve more efficient utilization of the processing resources by also parallelizing computation of gradient, and the linear algebraic methods for solving the Newton-system.
- While `Step 3` is naturally parallelized on C cores, further parallelization can be applied to the line search algorithm.
- State constraints of the form $\mathbb{X} = \{x \in \mathbb{R}^{n_x}, c(x) \leq 0\}$ for some smooth function c, can be included by replacing (17) with the Karush-Kuhn-Tucker (KKT) conditions. They are in general a mixture of equations and inequalities, and iterative approaches with additional mechanisms are generally needed to convert these into a set of equations at each iteration [14].

V. Simulations

In the following section, the simulation is performed to demonstrate the algorithm on the dynamical system data generated by the simulation of a nonlinear state-space model

$$\begin{aligned} \dot{x}_1 &= x_2 \exp\left[x_2 - 2x_1^2\right] - 1 + u + z_1 \\ \dot{x}_2 &= z_2 \\ y &= x_1 + v \end{aligned} \tag{21}$$

The state-space model consists of the nonlinear ordinary differential equation (ODE) system with the state x_1, unknown parameter x_2, input u, state process noise z_1, parameter process noise z_2 and continuous measurement y of x_1 with the continuous measurement noise v. The main goal of the simulation scenario is to investigate performance with a sudden change of the operating point. The fast system transient is triggered by the abrupt changes of parameter x_2.

The ODEs are solved numerically, where in the following experiments the explicit Heun's method [20] is used. The data are generated through the procedure for calculating the numerical solution to the initial value problem defined by the deterministic part of Eq. (21) with the initial condition of the state vector x_0,

$$\tilde{x}_{i+1} = x_i + h f_c(x_i, u_i) \tag{22}$$

$$x_{i+1} = x_i + \frac{h}{2}(f_c(x_i, u_i) + f_c(\tilde{x}_{i+1}, u_i)) \tag{23}$$

where i represents the numerical step index and h is the numerical step size. The additional process noise z_t is added at each sampling interval to simulate also the stochastic part, Eq. (5). The measurement equation is

$$y_t = x_{1,t} + v_t \tag{24}$$

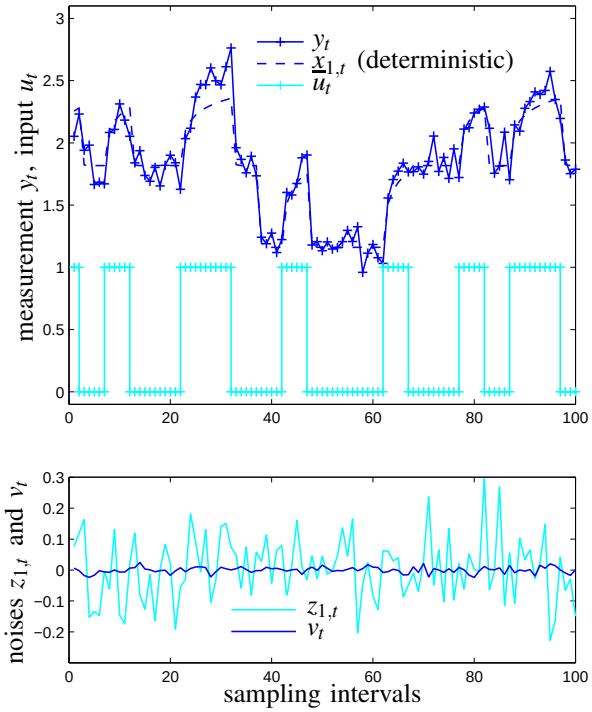

Fig. 3. Simulated data set with process and measurement noise (shown on the below subfigure).

Sampling interval T	1
Integration step h	0.1
Initial state $x_{1,0}$	0
Initial parameter $x_{2,0}$	0
Initial state x_1 variance $\Sigma_{1,0}$	1^2
Initial parameter x_2 variance $\Sigma_{2,0}$	5^2
Measurement noise:	
Standard deviation of the state x_1, σ_v	0.01
Process noise:	
Standard deviation of the state x_1, σ_1	0.1
Standard deviation of the parameter x_2, σ_2	0.02
Cost function value threshold δ_T	0.001
Norm gain parameter α, $W = diag(\alpha)$	0.005
Length of moving horizon N	3
Number of initial points C	3
Number of iterations Λ with single processor	3
Number of iterations Λ with three processors	1
Fixed step sizes Δ	$\{1,2,3\}$
Gain (pre-filtering on) θ	(10^{-4}) if $J_t(\hat{x}_{t-N\|t}) < \delta_T$
Gain (pre-filtering off) θ	(10^{-8}) if $J_t(\hat{x}_{t-N\|t}) > \delta_T$

TABLE I

SETTINGS OF NUMERICAL COMPUTATION

RSE	x_1	x_2
a	0.1588	9.1366
b	0.0983	8.8145
c	0.1184	7.5830

TABLE II

ROOT SQUARE ERROR OF STATES FROM SIMULATION SHOWN IN FIGURE 4

where y_t is the sampled measurement of state $x_{1,t}$ with added measurement noise v_t. During the simulation, a Pseudo Random Binary Signal (PRBS) is applied as the input u_t to excite the system. The simulated data considered in the following experiments are shown in Figure 3, where the process noise for x_1 was added with the properties $z_{1,t} \sim N(0, \sigma_1^2)$. The simulated stochastic properties of the measurement noise of measured x_1 is $v_t \sim N(0, \sigma_v^2)$.

a) Numerical setup: The settings of parameters needed for the numerical computation of the observer and optimization are listed in Table I. The measurement noise covariance is $R_t = \sigma_v^2$. The process noise spectral density matrix is $Q = diag\,[\Phi_1, \Phi_2]$, where the diagonal noise spectral densities are defined and computed as $\Phi_i = 1/T\sigma_i^2$. The initial state vector estimate is set to $\hat{x}_0^+ = [x_{1,0}, x_{2,0}]^T$. The initial covariance matrix of the initial state vector estimate error is $P_0^+ = diag\,[\Sigma_{1,0}, \Sigma_{2,0}]$, where the diagonal elements are in accordance to Eq. (28) computed as $\Sigma_i = (x_{i,0} - E[x_{i,0}])^2$ and set by the initial conditions. The weight matrices are set as $W = \alpha I$, where I is the identity matrix and $S = \theta P^{-1}$, motivated by [2]. The S matrix is time-varying since $P = P_{t-N|t}^+$, where θ is the gain parameter to turn on and off the pre-filtering. In the parallelization step, each sampling interval, the first processor starts from the point $\overline{x}_{t-N|t}^1 = \overline{x}_{t-N|t}$, the second processor starts from the point $\overline{x}_{t-N|t}^2 = \overline{x}_{t-N|t} + [0, \Delta]^T$, and the third processor starts from the point $\overline{x}_{t-N|t}^3 = \overline{x}_{t-N|t} + [0, -\Delta]^T$.

The quality of the algorithms is evaluated by the Root Square Error (RSE) computed for each state as

$$RSE_{x_k} = \|e_k\| = \sqrt{\sum_{t=1}^{n} e_{k,t}^2} \tag{25}$$

where $k = 1, 2$, $n = 100$, $e_{1,t} = y_t - \hat{x}_{1,t}$ and $e_{2,t} = x_{2,t} - \hat{x}_{2,t}$.

b) Estimation results with sequential computation, $C = 1$: Before any parallel computations are performed, the single-processor with sequential computations is evaluated in Figure 4. In this experiment the influence of pre-filtering on convergence is demonstrated. It can be seen that with the pre-filtering on (case "a"), the convergence during transients is retarded compared to the case with the pre-filtering off (case "b"), however in this case the estimates of x_1 and x_2 are more sensitive to the process noise. Improved filtering and rate of convergence is achieved when the pre-filtering is turned off during the transients (case "c"). The RSE index is computed and displayed in Table II. The cost function value is monitored during the computations and when it gets over the threshold value δ_T, the gain parameter θ changes to its off-value. The monitored cost function value with the threshold is shown in Figure 5. The influence of the number of iterations is shown on parameter x_2, in Figure 6, where it can be seen that three iterations have better converge rate of x_2 than one iteration.

c) Estimation results with parallel computation, $C = 3$: In the parallel computations, fast estimates of the state x_1 and parameter x_2 during the transients is shown in Figure 7. This figure compares the influence of different step-sizes Δ. The

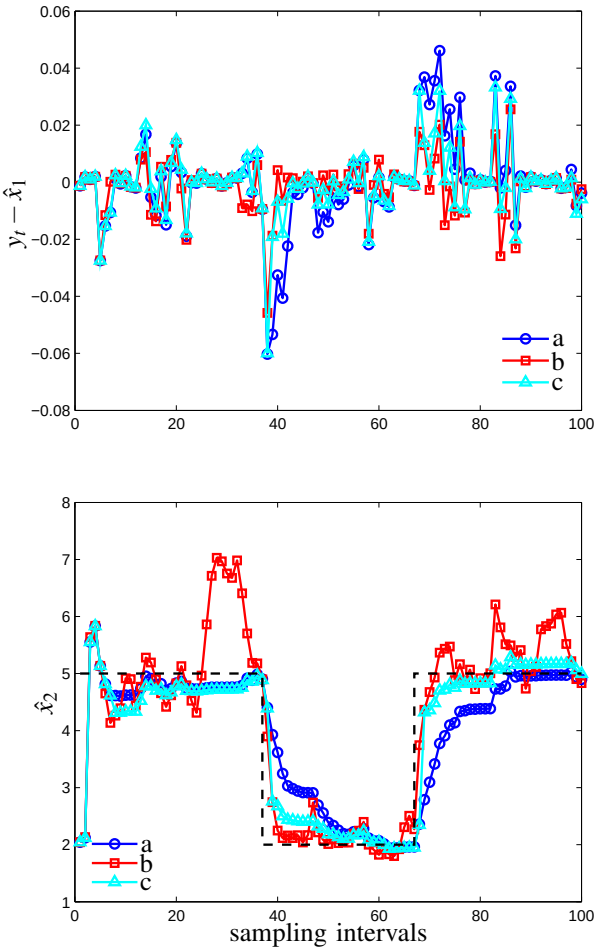

Fig. 4. Error of x_1 estimate and estimate of x_2 with single-processor optimization, number of iterations $\Lambda = 3$; a - pre-filtering on; b - pre-filtering off; c - pre-filtering off during transients

RSE	x_1	x_2
a	0.1245	7.8685
b	0.1205	7.5023
c	0.1119	7.2235

TABLE III

ROOT SQUARE ERROR OF STATES FROM SIMULATION SHOWN IN FIGURE 7

performance is compared for the step-size $\Delta = 1$ (case "a") and $\Delta = 2$ (case "b"), while the best performance is achieved for $\Delta = 3$ (case "c"). This is however a scenario where the algorithm can perform a "perfect jump" because the magnitude of change in the simulated system is exactly three. Such a step-size would be suitable for a parameter which randomly but repetitively changes between two constant values. The RSE index is computed and displayed in Table III.

The jumping logic among solutions from different processors for the case "a" (step-size $\Delta = 1$) is shown in Figure 8. Each parallel estimation started from a different initial point ends with a point which is evaluated in a cost function.

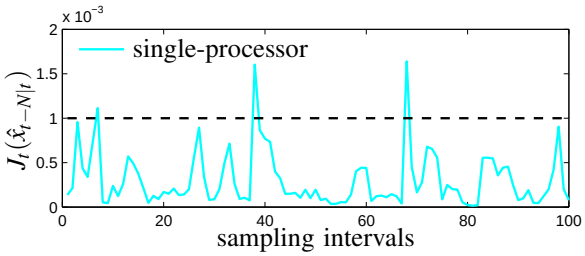

Fig. 5. Cost function value for the case "c" in Figure 4, threshold limit $\delta_T = 0.001$.

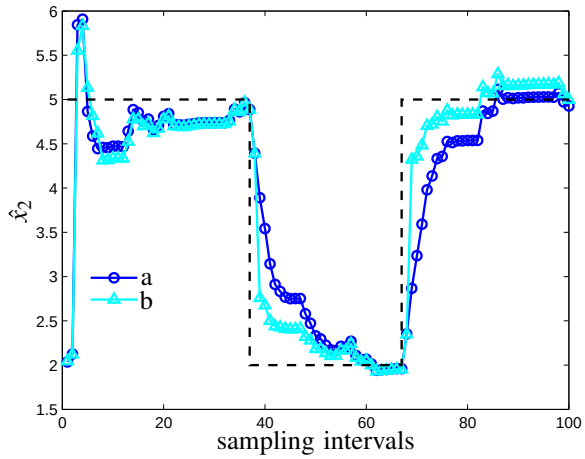

Fig. 6. Comparison of x_2 estimate with single-processor optimization, pre-filtering off during transients; a - number of iterations $\Lambda = 1$; b - number of iterations $\Lambda = 3$

The solution with the minimal cost function is marked with the red square. Recall that $\overline{x}^i_{t-N|t}$ with the upper index $i = 1$ represents the warm start initialization without any perturbation ($\overline{x}^1_{t-N|t} = \overline{x}_{t-N|t}$).

VI. CONCLUSION AND FUTURE WORK

This work demonstrates a novel approach and formulation of parallel optimization for an efficient moving horizon estimation. The efficiency lies in a strategy for evaluating several candidate initialization points and in the proposed parallel numerical optimization strategy that is based on the Gauss-Newton method. The simulation experiment demonstrates that the proposed parallelization and synchronization strategy performs more efficiently compared to sequential computation where the sequential approach uses more iterations, i.e. same number of computations as the parallel approach but the sequential approach needs more time.

The future work will focus more on the parallelization strategy of scattering the initial points scalability and real-time data testing.

APPENDIX I
EXTENDED KALMAN FILTER (EKF)

The EKF is used as a pre-filter and post-filter, therefore short summary of the assumptions and equations is introduced in this section. The following algorithm is in the

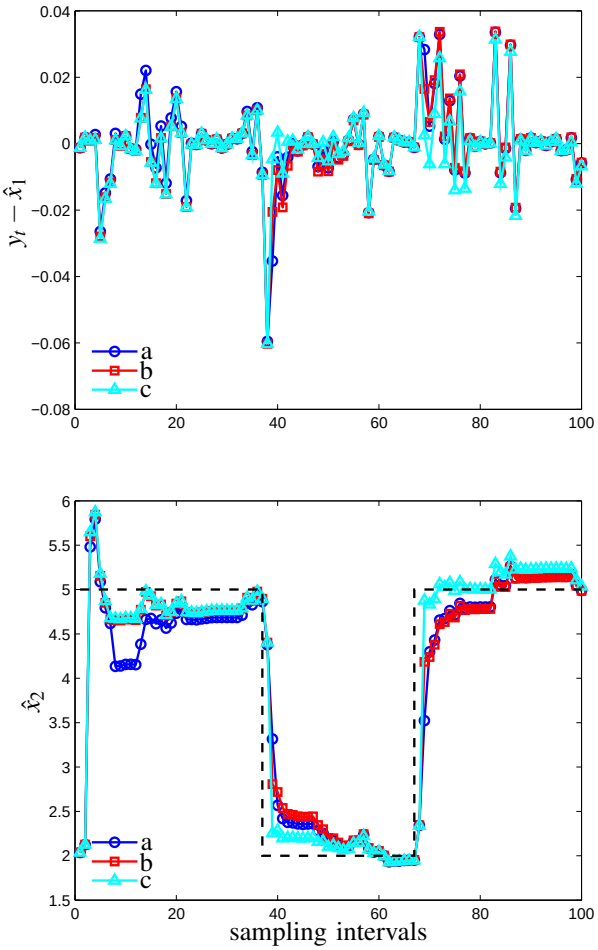

Fig. 7. Error of x_1 estimate and x_2 estimate with multi-processor optimization, number of processors $C = 3$, number of iterations $\Lambda = 1$; a - step-size $\Delta = 1$; b - step-size $\Delta = 2$; c - step-size $\Delta = 3$

literature known as continuous-discrete or hybrid EKF [21]. The dynamic system is given by (5) and (6). The main assumption about the process and the measurement noise is that they have the white noise properties, i. e. sequentially uncorrelated Gaussian distribution with zero mean

$$z_t \sim N(0, Q_t) \qquad v_t \sim N(0, R_t) \tag{26}$$

where Q_t is a process noise covariance matrix and R_t is a measurement noise covariance matrix. The initial condition of the state vector is assumed to be Gaussian distribution $x_0 \sim N(\hat{x}_0^+, P_0^+)$. The estimate of the state vector at $t = 0$ begins with the initial state vector estimate and with the initial covariance matrix of the initial state vector estimate error

$$\hat{x}_0^+ = E[x_0] \tag{27}$$
$$P_0^+ = E[(x_0 - \hat{x}_0^+)(x_0 - \hat{x}_0^+)^T] \tag{28}$$

From time instance $t-1$, the dynamic system (3) is simulatively propagated one step ahead as

$$\hat{x}_t^- = f(\hat{x}_{t-1}^+, u_{t-1}) \tag{29}$$

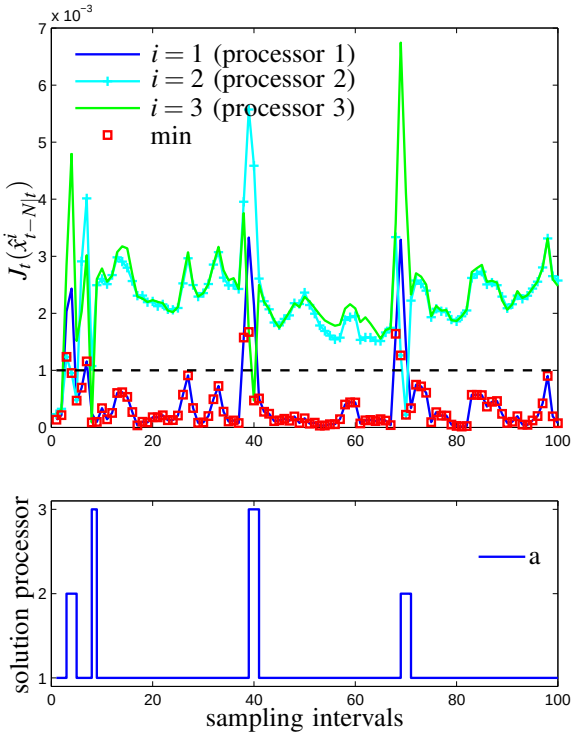

Fig. 8. Multi-processor cost function values for the case "a" in Figure 7 with picked minimum (red squares), threshold limit $\delta_T = 0.001$. Below figure shows the jumps among multi-processor solution points based on minimum cost function value in each processor at a time.

where $t = 1, 2, \ldots$. This one step prediction gives an a priori state estimate. The time update of the covariance matrix estimate is given by

$$\dot{P} = Z(\hat{x})P + PZ^T(\hat{x}) + Q \tag{30}$$

where

$$Z(\hat{x}) = \left.\frac{\partial f_c(x)}{\partial x}\right|_{x=\hat{x}} \tag{31}$$

and Q is a spectral density matrix, where $Q = \frac{1}{T}Q_t$. The covariance matrix estimate of state vector $\hat{x}_t^-$ estimation error is computed by simulative propagation of (30)

$$P_t^- = g(P_{t-1}^+, Z(\hat{x}_{t-1}^+)) \tag{32}$$

The EKF gain matrix is in time instant t

$$K_t = P_t^- L_t^T [L_t P_t^- L_t^T + M_t R_t M_t^T]^{-1} \tag{33}$$

and the measurement y_t is used for state vector estimation (a posteriori estimate)

$$\hat{x}_t^+ = \hat{x}_t^- + K_t[y_t - h(\hat{x}_t^-)] \tag{34}$$

The covariance matrix a posteriori estimate is updated as

$$P_t^+ = [I - K_t L_t] P_t^- [I - K_t L_t]^T + K_t M_t R_t M_t^T K_t^T \tag{35}$$

where

$$L_t = \left.\frac{\partial h(x_t)}{\partial x_t}\right|_{x_t = \hat{x}_t^-} \tag{36}$$

$$M_t = \left.\frac{\partial h(x_t)}{\partial v_t}\right|_{x_t = \hat{x}_t^-} \tag{37}$$

The presented EKF algorithm will be generally denoted as

$$\hat{x}_{t+1}^{+} = f_F(\hat{x}_t^{+}, u_t, y_{t+1}) \tag{38}$$

for $t = 0, 1, \ldots$, where $f_F : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$ represents the EKF dynamics. Its shorter notation $f_F^{u_t, y_{t+1}}(\hat{x}_t^{+})$ will also be used when convenient.

REFERENCES

[1] P. E. Moraal and J. W. Grizzle, "Observer design for nonlinear systems with discrete-time measurement," *IEEE Transactions on automatic control*, vol. 40, no. 3, pp. 395–404, 1995.

[2] C. V. Rao, J. B. Rawlings, and D. Q. Mayne, "Constrained state estimation for nonlinear discrete-time systems: Stability and moving horizon approximations," *IEEE Transactions on Automatic Control*, vol. 48, no. 2, pp. 246–258, 2003.

[3] A. Alessandri, M. Baglietto, and G. Battistelli, "Moving-horizon state estimation for nonlinear discrete-time systems: New stability results and approximation schemes," *Automatica*, vol. 44, no. 7, pp. 1753–1765, 2008.

[4] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. Athena Scientific, 1997.

[5] M. C. Ferris and O. L. Mangasarian, "Parallel variable distribution," *SIAM Journal on Optimization*, vol. 4, no. 4, pp. 815–832, 1994. [Online]. Available: http://link.aip.org/link/?SJE/4/815/1

[6] C. A. Sagastizábal and M. V. Solodov, "Parallel variable distribution for constrained optimization," *Comput. Optim. Appl.*, vol. 22, pp. 111–131, April 2002. [Online]. Available: http://dl.acm.org/citation.cfm?id=584529.584535

[7] O. L. Mangasarian, "Parallel gradient distribution in unconstrained optimization," *SIAM Journal on Control and Optimization*, vol. 33, pp. 1916–1925, 1995.

[8] M. Fukushima, "Parallel variable transformation in unconstrained optimization," *SIAM Journal on Optimization*, vol. 8, no. 3, pp. 658–672, 1998. [Online]. Available: http://link.aip.org/link/?SJE/8/658/1

[9] T. F. Coleman and P. E. Plassmann, "A parallel nonlinear least-squares solver: Theoretical analysis and numerical results," *SIAM Journal on Scientific and Statistical Computing*, vol. 13, no. 3, pp. 771–793, 1992. [Online]. Available: http://link.aip.org/link/?SCE/13/771/1

[10] R. A. Renaut and H. D. Mittelmann, "Parallel multisplittings for optimization," *Parallel Algorithms and Applications*, vol. 7, no. 1-2, pp. 17–27, 1995. [Online]. Available: http://www.tandfonline.com/doi/abs/10.1080/10637199508915519

[11] V. Torczon, "On the convergence of the multidirectional search algorithm," *SIAM Journal on Optimization*, vol. 1, no. 1, pp. 123–145, 1991. [Online]. Available: http://link.aip.org/link/?SJE/1/123/1

[12] D. Lee and M. Wiswall, "A parallel implementation of the simplex function minimization routine," *Computational Economics*, vol. 30, pp. 171–187, 2007, 10.1007/s10614-007-9094-2. [Online]. Available: http://dx.doi.org/10.1007/s10614-007-9094-2

[13] J. A. Nelder and R. Mead, "A simplex method for function minimization," *The Computer Journal*, vol. 7, no. 4, pp. 308–313, 1965. [Online]. Available: http://comjnl.oxfordjournals.org/content/7/4/308.abstract

[14] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed. Springer, 2006.

[15] M. A. Luersen and R. L. Riche, "Globalized Nelder-Mead method for engineering optimization," *Computers and Structures*, vol. 82, pp. 2251–2260, 2004.

[16] A. Koscianski and M. Luersen, "Globalization and parallelization of Nelder-Mead and Powell optimization methods," in *Innovations and Advanced Techniques in Systems, Computing Sciences and Software Engineering*, K. Elleithy, Ed. Springer Netherlands, 2008, pp. 93–98.

[17] T. Polóni, A. A. Eielsen, B. Rohal'-Ilkiv, and T. A. Johansen, "Moving horizon observer for vibration dynamics with plant uncertainties in nanopositioning system estimation," in *American Control Conference (ACC), 2012*, Montréal, Canada, 2012, pp. 3817–3824.

[18] A. N. Tikhonov and V. Y. Arsenin, *Solutions of Ill-posed Problems*. Wiley, 1977.

[19] D. Sui and T. A. Johansen, "Moving horizon observer with regularisation for detectable systems without persistence of excitation," *International Journal of Control*, vol. 84, no. 6, pp. 1041–1054, 2011.

[20] U. M. Ascher and L. R. Petzold, *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*. SIAM, 1998.

[21] A. Gelb, J. Joseph F. Kasper, J. Raymond A. Nash, C. F. Price, and J. Arthur A. Sutherland, *Applied optimal estimation*, A. Gelb, Ed. The. M.I.T. Press, 2001.