# How to speed up MHEs - project ideas

Federico Oliva
lvofrc95@outlook.it



## Problem Statement

This document aims to present the general MHE problem and the implementation used in [7, 8]. The goal is to stress the main issues regarding the computational burden required by the algorithm, and to propose some ideas to speed it up. Such ideas come from the main topics discussed during the May 2022 EECI Spring School *Sparsity and Big Data in Control, System Identification and Machine Learning*, held in Toulouse.

### ✍The MHE problem

The problem tackled by the *Moving Horizon Estimators* (MHEs) is the design of robust observers, intending to reconstruct the plant states that are not directly measurable by sensors. Roughly speaking MHEs handles the estimation process by minimising an optimisation problem, defined as a model matching problem on the system state and exploiting a buffer of output measurements [4, 6, 10–12, 16]. The MHE can be interpreted as an injection-like observer, as described in [9], but in general its structure is different from the more common *Luenberger observer* and *Kalman filter* [3, 14]. Generally speaking MHEs can handle both linear and nonlinear systems. Due to its optimisation-based approach, the main drawback of the MHE is the computational cost. Indeed, the more complex the model and the longer the measurement buffer, the slower the optimisation process. This section aims to define the MHE problem and to highlight the main reasons of its computational complexity.

## ✍️MHE overview

Consider a general nonlinear system in the following form:

$$\boldsymbol{\xi}_{k+1} = f(\boldsymbol{\xi}_k, \boldsymbol{u}_k) + \boldsymbol{w}_k, \tag{1a}$$

$$\boldsymbol{y}_k = h(\boldsymbol{\xi}_k, \boldsymbol{u}_k) + \boldsymbol{\nu}_k, \tag{1b}$$

where $\boldsymbol{\xi}_k \in \mathbb{R}^n$ is the state vector, $\boldsymbol{u}_k \in \mathbb{R}^m$ is the control input, and $\boldsymbol{y}_k \in \mathbb{R}^p$ is the measured output. The vectors $(\boldsymbol{w}_k, \boldsymbol{\nu}_k)$ represent additive model and measurement disturbances. The index $k$ stands for the time instant at which the model is considered.

In the rest of the report it is assumed that inputs and outputs of plant (1) are sampled every $T_s$ seconds, i.e. $T_s$ is the sampling time of (1). It is also considered a down-sampling of the I/O signals to be saved into the MHE buffer (moving window data), such that samples at time instants $t_k$ are buffered, with $k \in \mathbb{Z}$, and $t_k - t_{k-1} = N_{T_s} \cdot T_s$ (multiple of sampling time) for some $N_{T_s} \in \mathbb{N}_{\geq 1}$. The input $\boldsymbol{u}_k$ is assumed to be constant within sampling time intervals, i.e. it is piece-wise constant. Define the (down-sampled) output $\mathrm{Y}_k = [\boldsymbol{y}_{k-N+1}, \ldots, \boldsymbol{y}_k]^T \in \mathbb{R}^{(N\times p)}$ and input $\mathrm{U}_k = [\boldsymbol{u}_{k-N+1}, \ldots, \boldsymbol{u}_k]^T \in \mathbb{R}^{(N\times m)}$ vectors of (1), where $\boldsymbol{y}_j \triangleq \boldsymbol{y}(t_j)$ and $\boldsymbol{u}_j = \boldsymbol{u}(t_j)$, and with $j \in \{k - N + 1, .., k\}$. $N$ is the number of samples defining the moving window data (buffer). The *N-lifted* system [15] operator is defined as

$$\mathrm{H}_k(\boldsymbol{\xi}_{k-N+1}, \boldsymbol{u}(\cdot)) = \begin{bmatrix} h(\boldsymbol{\xi}_{k-N+1}, \boldsymbol{u}_{k-N+1}) \\ h \circ f(\boldsymbol{\xi}_{k-N+2}, \boldsymbol{u}_{k-N+2}) \\ \vdots \\ h \circ f(\boldsymbol{\xi}_{k-1}, \boldsymbol{u}_{k-1}). \end{bmatrix} \tag{2}$$

where $\circ$ denotes the composition operation. Within the framework of MHEs, the estimation problem consists in finding the solution, at each time step $k$, to the following minimisation problem [9]:

$$\min_{\hat{\boldsymbol{\xi}}_{k-N+1}} \mathrm{V}_k(\mathrm{Y}_k, \mathrm{H}_k(\hat{\boldsymbol{\xi}}_{k-N+1}, \boldsymbol{u}(\cdot))), \tag{3}$$

for which a quadratic cost function $V_k$ was selected as

$$V_k \triangleq \sum_{j=1}^{N} (\mathrm{Y}_k^j - \hat{\mathrm{H}}_k^j)^T W_j (\mathrm{Y}_k^j - \hat{\mathrm{H}}_k^j), \tag{4}$$

where $\hat{H}_k = \mathrm{H}_k(\hat{\boldsymbol{\xi}}_{k-N+1}, \boldsymbol{u}(\cdot))$, $W_i \in \mathbb{R}^{p\times p}$ are symmetric and positive definite weight matrices, $\mathrm{Y}_k^j$ and $\hat{\mathrm{H}}_k^j$ are the *j-th* rows of the matrices $Y_k$ and $\hat{H}_k$, respectively. The structure of this MHE outlines the one described in [16], with the exception that a *single-shooting* rather than a *simultaneous* MHE observer is considered, i.e. the optimisation is performed on the variable $\hat{\boldsymbol{\xi}}_{k-N+1}$ only, and not simultaneously on the vector $[\hat{\boldsymbol{\xi}}_{k-N+1}, \ldots, \hat{\boldsymbol{\xi}}_k]$. This yields the loss of constraints $||\hat{\boldsymbol{\xi}}_{j+1} - f(t_{j+1}, t_j, \hat{\boldsymbol{\xi}}_j, \boldsymbol{u}(\cdot))|| = 0$, $\forall j = \{k - N + 1, \ldots, k\}$, resulting in an unconstrained optimisation problem. Please note that no *terminal cost* is considered as in [5, 11, 12], since forward propagation of the previous estimate is obtained through (1) as in [16]. Practical convergence of the estimation error can be ensured if (1) is *N-observable* according to the Definition provided in [1].

# MHE computational issues

The MHE described in 1.2 presents a main issue, namely its computational burden grows as $N$ and the complexity of model $f(\cdot)$ grow. In fact, in order to carry on the minimisation process, (2) shall be evaluated at each iteration of the optimisation algorithm. Thus, function $f(\cdot)$ of (1) is computed $N$ times every iteration. Clearly, the longer the buffer and the more complex the model, the slower the whole process.

## ✍️Potential approaches

The potential approaches to tackle the computational burden problem are the following:

1. **Reduce the buffer length $N$:** by doing so, less model integrations are required. This is the approach proposed in [7, 8], where output filtering and adaptive sampling are exploited to speed up the MHE.
2. **Reduce the model complexity:** the main issue with the forward integration of (1) lies in the complexity of the $f(\cdot)$ mapping. A potential remedy would be to simplify $f(\cdot)$ by finding an equivalent or approximated model.
3. **Speeding up the optimisation algorithm:** another remedy to the slowness of the process could be to improve the efficiency of the chosen optimisation algorithm. MHE is a strongly non-convex optimisation problem, if no specific structure is assumed for the mapping $f(\cdot)$. In [7, 8] the optimisation process relied on a blind usage of MATLAB `fminunc` method, where for instance no information on the gradient of $f(\cdot)$ was used. Two ways can be outlined to improve the MHE performance, as far as optimisation algorithms are concerned:
   (a) **Reduce iterations:** this simple approach consists in reducing the number of iterations for each step of the algorithm. This approach is described for instance in [4], where stability issues are discussed.
   (b) **Exploit structure:** this approach would try to investigate the structure of the MHE problem in order to see if some property (i.e. spasrity) can be exploited to speed up the process.

In the next paragraphs, the focus will be on the last two bullets of the list above, namely on how to improve the efficiency of the optimisation algorithm and how to lower the model complexity. The paths proposed take inspiration from the topics covered during the May 2022 EECI Spring School *Sparsity and Big Data in Control, System Identification and Machine Learning*, held in Toulouse.

## 💡Conjugate function

As reported in 2.1, the MHE problem is highly non convex, due to the presence of mapping $f(\cdot)$ in its structure. A first approach one could take is trying to relax the MHE problem by considering a convex approximation of the cost function $\mathrm{V}_k$. Consider a general function $g(x)$ with no convexity assumptions on it. Given $g(x)$, the following definition holds:

**Definition 1 (Conjugate function)** *Given a general function $g(x)$, its conjugate is defined as*

$$g^\star(y) = \sup_x (y^T x - g(x)) \tag{5}$$

An important result states the following:

**Remark 1** *Consider a general function $g(x)$ and its conjugate $g^\star(y)$. Now, $g^\star(y)$ is convex even if $g(x)$ is not.*

One could decide to minimise over the conjugate function instead, but there isn't in general any assumption on how tight the convex approximation is with respect to the former cost function $g(x)$. An interesting result in this direction is the following:

**Proposition 1 (Convex Envelope)** *Consider a general function $g(x)$. The following holds*

1. *$g(x)$ is convex: the double conjugate $g(z)^{\star\star}$ returns again the initial function $g(x)$*
2. *$g(x)$ is not convex: the double conjugate $g(z)^{\star\star}$ returns the tightest convex underestimator, namely the convex envelope*

The main issue with this approach is that there is no guarantee that the unique minimum of the convex envelope coincides with the global minimum of the initial cost function. The implementation steps could be the following:

- Test the convex envelope solution on a toy model such as the Van der Pol oscillator (both state and parameter estimation)
- Study if assumptions on the model structure can lead to statements on the goodness of the approximation through the convex envelope

## 💡Sparsity on Hessian function

As described in 1.2 the considered implementation of the MHE problem yields an unconstrained optimisation problem. Moreover, in [7, 8] its solution has been carried out by simply using the MATLAB method `fminunc`. As reported in the official MATLAB help regarding `fminunc` suggest to explicitly provide the algorithm with a Gradient or an Hessian. Indeed this makes sense as `fminunc` exploits a gradient-like optimisation algorithm. In the following this kind of algorithms are recalled.

**Gradient-like optimisation algorithms** Consider a general unconstrained optimisation problem in the following form:

$$\min_x \ f(x), \ x \in \mathbb{R}^n \tag{6}$$

At this stage there is no assumption on the structure of the functional cost $f(x)$. The following well-known results define a set of necessary conditions for a point $\bar{x}$ to be a local minimum for $f(x)$:

**Theorem 1 (First-Order Necessary Condition)** *Let $f \in C^1$. If $\bar{x} \in \mathbb{R}^n$ is a local minimum for problem (6), then it holds $\nabla f(\bar{x}) = 0$.*

**Theorem 2 (Second-Order Necessary Condition)** *Let $f \in C^2$. If $\bar{x} \in \mathbb{R}^n$ is a local minimum for problem (6), then it holds :*

1. $\nabla f(\bar{x}) = 0$
2. $d^T \nabla^2 f(\bar{x}) d \geq 0 \ \forall d \in \mathbb{R}^n$

These necessary conditions are used in the design of algorithms for the solution of unconstrained optimisation problems. Generally speaking the structure of these algorithms outlines the following structure:

$$x_{k+1} = x_k + \alpha_k d_k, \tag{7}$$

where $d_k \in \mathbb{R}^n$ is the search direction, with $||d_k|| = 1$, and $\alpha_k$ is the step-size. Two of the most used algorithms are the *Gradient method* and the *Newton's method*:

- **Gradient Method:** this is the simplest method and it is based on the first-order Taylor approximation of $f(x)$, namely
$$f(x_k + \alpha d) = f(x_k) + \alpha \nabla f(x_k)^T d, \ \alpha > 0, \ ||d|| = 1, \tag{8}$$
resulting in $d_k = -\dfrac{\nabla f(x_k)}{||\nabla f(x_k)||}$.
- **Newton's method:** this approach is similar to the previous one but it exploits the second-order Taylor approximation of $f(x)$, namely
$$f(x_k + h) = f(x_k) + \nabla f(x_k)^T h + \frac{1}{2} h^T \nabla^2 f(x_k) h, \tag{9}$$
where $h$ is the displacement of the new solution with respect to the previous one. By setting to zero the gradient in $h$ of (9) the update term becomes $h = -\nabla^2 f(x_k)^{-1} \nabla f(x_k)$. In a single step the algorithm computes both the search direction and the step-size. Indeed, the computation of the Hessian can be demanding as the problem dimension grows.

**Sparsity in the Hessian estimation** As described above, if the Gradient and/or the Hessian can be computed for the functional cost $f(x)$, the solution of problem (6) can speed up considerably. As far as MATLAB optimisation solvers are concerned, the algorithm options relying on Gradient and Hessian are the following options:

- `fminunc`: the solver can use a trust-region algorithm[1] to solve the optimisation problem, if at least the Gradient and optionally the Hessian are provided.
- `fmincon`: the solver can either use a trust-region algorithm with the previous requirements, or an interior-point algorithm which requires optionally both the Gradient and the Hessian.

The goal of this solution is to exploit some structure in the Hessian to actually estimate it[2]. More specifically, if the Hessian is sparse, it can be described through a Cholesky factorization, namely

$$\nabla^2 f(x) = PLL^T P, \tag{10}$$

where $P$ is a permutation matrix[3] and $L$ is lower-triangular. This structure could be used to solve a linear problem and define the Newton step. This can be done if the gradient is available.

The main points on this solution would be the following:

- Consider the detailed problem to find the Hessian (find paper from which notes where inspired)

[1] Roughly speaking, trust-region algorithms first proceed determining the step-size, and then the search direction

[2] See EECI M12 notes

[3] A permutation matrix is a square binary matrix that has exactly one entry of 1 in each row and each column and 0 elsewhere.

- Consider how to get the gradient. Is the analytical way the only viable or could we use an optimisation approach as well?
- Consider if `fminunc` and `fmincon` are the best options within which integrating the Gradient/Hessian (interior-point is for convex problems...) or if a custom optimiser could be an option too (maybe use ADMM to solve the Hessian-finding problem?)

### 💡Koopman operator

As reported in 2.1 a potential approach to speed up the MHE problem solution consists in reducing the complexity of the model used as the prior knowledge to compute $\hat{\mathrm{H}}_k$. A first idea could be to exploit a linearised version of the general model described in (1). However, as the MHE considers a buffer of measurement over the system trajectory, the validity of such linearised system cannot in general be taken from granted for the whole set of points considered. A possible approach to keep the lower complexity of a linear model yet properly describing the system without restricting the analysis to a limited set of working conditions could be represented by the *Koopman operator* theory [2].

Consider a general system in the form of (1). We define *observables* as those quantities obtained from the output mapping $h(\cdot)$, namely evaluations of a specific function of the system state. Depending on the system structure, the state may or may not uniquely determine the value of the observables. Therefore, the key question one could start from is whether or not the state evolution can be inferred from the knowledge of the observables. An useful tool in this direction is represented by the Koopman operators, which are defined as follows:

**Definition 2 (Koopman operator)** *Consider a general discrete-time system as in* (1)*, with a scalar output mapping* $h(\cdot) : \mathcal{S} \to \mathbb{R}$*. The Koopman operator* $U$ *is defined as a linear transformation* $Uh(x) = h \circ f(x)$*, where* $\circ$ *denotes the composition operation. As the composition operation is linear, it follows that the Koopman operator is linear too.*

Definition 2 holds also for continuous-time systems, with the only difference that the Koopman operator $U^t h(x)$ is defined as a set of parametrised operators in the time variable $t$. Roughly speaking, the Koopman operator performs a lifting of the system dynamics from the state space to the observable state. The main advantage is that the new evolution is linear, the drawback is that the space of observables is infinite dimensional. This dynamics description is based on the concept of *eigenfunction-eigenvalue pair of the Koopman operator*, which is defined as follows:

**Definition 3 (Eigenfunction-Eigenvalue pair)** *Consider a complex-valued observable* $\phi_j : \mathcal{S} \to \mathbb{C}$ *of a dynamical system, and a complex number* $\lambda_j \in \mathbb{C}$*. The pair* $(\phi_j, \lambda_j)$ *is said to be an eigenfunction-eigenvalue pair of the Koopman operator* $U$ *if it holds*

$$U^t \phi_j = e^{\lambda_j t} \phi_j \tag{11}$$

This report doesn't want to be a detailed discussion of Koopman operators, but the take-home message is that several classes of nonlinear systems can be described as an infinite-dimensional linear combination of these Koopman

eigenfunctions, yelding a linear evolution rule. Indeed, identifying these eigenfunctions isn't trivial. In [13] a data-driven approach is proposed to approximate Koompan operators on a restricted subspace defined by a dictionary of functions.

Although the Koopman operators approach could actually speed up the dynamics propagation, the MHE problem still needs to map the trajectories back in the state space in order to perform the estimation. Again in [13], it is proposed to use a local approximation by means of fractional functions (Loewner matrices) to perform the mapping from the dictionary back to the state space.

A potential workplan for this Koompan-based solution to speed-up the MHE could be the following:

- Compute the Koompman operator for a defined system (e.g. a Van der Pol oscillator) exploiting the approach proposed in [13].
- Use the Koopman operator to forward propagate the model dynamics during the solution of the MHE optimisation problem. Once the trajectory has been evaluated over the whole MHE buffer, get back to the state space (Loewner matrices) to actually compute $\hat{H}_k$.
- It could be worth studying how to use this approach if parameters were also to be estimated.

# References


[1] D. Aeyels. On the number of samples necessary to achieve observability. *Systems & Control Letters*, 1981. 2

[2] Hassan Arbabi. Introduction to koopman operator theory of dynamical systems. Available at https://www.mit.edu/~arbabi/research/KoopmanIntro.pdf. 6

[3] Jon H. Davis. *Luenberger Observers*, pages 245–254. Birkhäuser Boston, 2002. 1

[4] Wei Kang. Moving horizon numerical observers of nonlinear control systems. *IEEE Transactions on Automatic Control*, 2006. 1, 3

[5] Peter Kühl, Moritz Diehl, Tom Kraus, Johannes P. Schlöder, and Hans Georg Bock. A real-time algorithm for moving horizon state and parameter estimation. *Computers & Chemical Engineering*, 2011. 2

[6] H. Michalska and D.Q. Mayne. Moving horizon observers and observer-based control. *IEEE Transactions on Automatic Control*, 1995. 1

[7] Federico Oliva and Daniele Carnevale. Moving horizon estimator with filtering and adaptive sampling. *submitted to the 18th IFAC Workshop Control Applications for Optimisation*, 2022. 1, 3, 4

[8] Federico Oliva and Daniele Carnevale. On the implementation of adaptive and filtered mhe. *Arxiv*, 2022. 1, 3, 4

[9] J.W. Grizzle P.E. Moraal. Observer design for noninear systems with discrete-time measurements. *IEEE Transactions on Automatic Control*, 1995. 1, 2

[10] Julian D. Schiller, Sven Knüfer, and Matthias A. Müller. Robust stability of suboptimal moving horizon estimation using an observer-based candidate solution. *IFAC-PapersOnLine*, 2021. 1

[11] Dan Sui, Tor Arne Johansen, and Le Feng. Linear moving horizon estimation with pre-estimating observer. *IEEE Transactions on Automatic Control*, 2010. 1, 2

[12] Rata Suwantong, Sylvain Bertrand, Didier Dumur, and Dominique Beauvois. Stability of a nonlinear moving horizon estimator with pre-estimation. In *2014 American Control Conference*, 2014. 1, 2

[13] Mario Sznaier. A convex optimization approach to learning koopman operators. *Arxiv*, 2021. 7

[14] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. The MIT Press, 2006. 1

[15] R. Tousain, E. van der Meche, and O. Bosgra. Design strategy for iterative learning control based on optimal control. *Proceedings of the 40th IEEE Conference on Decision and Control (Cat. No.01CH37228)*, 2001. 2

[16] Andrew Wynn, Milan Vukov, and Moritz Diehl. Convergence guarantees for moving horizon estimation based on the real-time iteration scheme. *IEEE Transactions on Automatic Control*, 2014. 1, 2