Feasibility of standard and novel solvers in atmospheric tomography for the ELT

B. Stadler[a], R. Biasi[b], M. Manetti[b], and R. Ramlau[a]

[a]Johannes Kepler University Linz, Altenbergerstrasse 69, Linz, Austria
[b]Microgate, Via Waltraud-Gebert-Deeg 3e, Bolzano, Italy

ABSTRACT

The new generation of ground-based extremely large telescopes require highly efficient algorithms to achieve an excellent image quality in a large field of view. These systems rely on adaptive optics, where one aims to compensate in real-time the rapidly changing optical distortions in the atmosphere. Due to the growth of telescope sizes, the computational load is increasing drastically. Thus, current algorithms become inappropriate and there is a big interest in the development of more efficient solvers. In this work, we compare a novel method called Finite Element Wavelet Hybrid Algorithm to the frequently used MVM within the framework of MAORY, an instrument of the Extremely Large Telescope. We look in detail at the performance of both algorithms in terms of floating point operations, memory usage and parallelization possibilities. On that basis, we determine a possible real-time computing hardware architecture for both algorithms.

Keywords: atmospheric tomography, feasibility study, FEWHA, MVM

1. INTRODUCTION

Many next generation adaptive optics (AO) systems, such as Multi-Conjugate AO (MCAO), Laser-Tomography AO (LTAO) or Multi-Object AO (MOAO), require the reconstruction of the turbulence profile, which is called atmospheric tomography. The dimension of this problem depends on the number of subapertures of the used wavefront sensors and on the number of degrees of freedom of the correcting mirrors, which are in general higher for bigger telescopes. Moreover, the solution has to be computed in real-time. Altogether, efficient solution methods are of great interest for this kind of problems.

Atmospheric tomography has been widely studied during the last years.[1–3] Mathematically, the problem is ill-posed, i.e., there is an unstable relation between measurement data and the solution.[4] Hence, regularization techniques must be applied. A common way of dealing with this problem is the Bayesian formulation, where the statistical information regarding the turbulence model and sensor noise can be incorporated.

So far, the standard method for atmospheric tomography is the matrix-vector multiplication (MVM).[5] The computational costs of the MVM scales at $\mathcal{O}(n^2)$, where n is the dimension of the AO system. These dimension is increasing drastically in the next generation of ground-based telescopes, as e.g., the Extremely Large Telescope (ELT). That is why alternative solvers, like the Finite Element Wavelet Hybrid Algorithm (FEWHA), become a feasible alternative.

Within this work, we want to show the benefits and drawbacks of a standard matrix-vector multiplication approach compared to the iterative approach of FEWHA. We give a detailed comparison of the two algorithms in terms of computational performance. In particular, we compare floating point operations (FLOPs), memory usage and parallelization possibilities. This theoretical study is evaluated on the test case of MAORY, the adaptive optical relay for some of the first light instruments of the ELT. For such a big AO system the large benefits in terms of computational performance of an iterative approach become visible. Based on this study, we determine a possible hardware architecture for both algorithms to meet the real-time requirements of MAORY.

This paper is structured as follows. In Section 2 we give a short introduction to atmospheric tomography and the Bayesian framework. Section 3 is devoted to the description of the the MVM and FEWHA. The performance

Further author information: (Send correspondence to B. Stadler)
B. Stadler: E-mail: bernadett.stadler@indmath.uni-linz.ac.at

analysis of the algorithms, in particular, floating point operations, memory usage and parallelization possibilities, are presented in Section 4. In Section 5 we look at a specific test case, namely MAORY, and determine a particular hardware architecture for the real-time computing (RTC) unit. Finally, a conclusion and an outlook are given in Section 6.

2. ATMOSPHERIC TOMOGRAPHY

Atmospheric tomography is the fundamental problem in many AO systems. Assuming a layered model of the atmosphere, the goal of the atmospheric tomography problem is to reconstruct the turbulent layers from the wavefront sensor measurements.[6]

The atmospheric tomography problem is defined by

$$s = s_g = A\phi \text{ for } g = 1, ..., G, \tag{1}$$

where G are the number of guide stars, $\phi = (\phi_1, ..., \phi_L)$ denote the L turbulent layers, s the sensor measurements and A is the tomographic operator. Assuming the presence of a Shack-Hartmann (SH) sensor, A is a concatenation of the SH operator Γ and a geometric propagation operator P in the direction of the guide star. This leads to the following equivalent formulation of Equation (1)

$$s_g = \Gamma_g P_g \phi \text{ for } g = 1, ..., G. \tag{2}$$

A common procedure is to formulate the problem in the Bayesian framework, where statistical information regarding the turbulence model and noise can be utilized. Let $\boldsymbol{S}$ and $\boldsymbol{\phi}$ be random variables corresponding to the vectors of measurements and turbulence layers, respectively. Further, we assume the presence of noise modelled as a random variable $\boldsymbol{\eta}$. Altogether, this leads to a re-formulation of Equation (1)

$$\boldsymbol{S} = A\boldsymbol{\phi} + \boldsymbol{\eta}. \tag{3}$$

Prior and noise models are typically Gaussian, therefore, the maximum a posteriori (MAP) estimate provides an optimal point estimate for the solution.[7] It is obtained by solving the linear system of equations

$$(A^T C_\eta^{-1} A + C_\phi^{-1})\phi = A^T C_\eta^{-1} s, \tag{4}$$

where C_ϕ^{-1} and C_η^{-1} are the inverse covariance matrices of layers $\boldsymbol{\phi}$ and noise $\boldsymbol{\eta}$, respectively. We assume that the layers are zero centered and uncorrelated, which implies a block-diagonal structure of C_ϕ. The noise covariance matrix C_η is given as a block-diagonal matrix with respect to the wavefront sensors.[8]

3. SOLVERS FOR ATMOSPHERIC TOMOGRAPHY

In this work, we focus on two different methods for solving Equation (4). The first one is the frequently used MVM. The second is a novel, iterative method called FEWHA.

3.1 Matrix Vector Multiplication

The standard approach to solve Equation (4) is called MVM, where the inverse of the discretized left-hand side matrix is computed explicitly by

$$R := (A^T C_\eta^{-1} A + C_\phi^{-1})^{-1} A^T C_\eta^{-1} \tag{5}$$

and then multiplied with the sensor measurements. Typically, a mirror fitting operator F is combined with the atmospheric reconstruction, mapping sensor measurements onto mirror shapes

$$a = (FR)s. \tag{6}$$

In the closed loop case, the computation of pseudo open loop (POL) measurements, based on the previous DM shape $a^{(-1)}$, is required in a first step

$$s^o = s + \Gamma a^{(-1)}. \tag{7}$$

The calculation of FR is often referred to as soft real-time, since the re-computation has to be done whenever the specific geometry (directions θ_g and layer altitudes h_ℓ), certain noise levels (entries of C_η) or the turbulence parameters (entries of C_ϕ) change. In contrast, the POL computations and the multiplication with the vector of sensor measurements s, which is done at approximately 500 - 1000 Hz, is called hard real-time.

3.2 Finite Element Wavelet Hybrid Algorithm

The Finite Element Wavelet Hybrid Algorithm (FEWHA) is an iterative approach to compute the MAP estimate. The main idea is to use compactly supported orthonormal wavelets for representing the turbulent layers, because the properties in the frequency domain allow a completely diagonal approximation of the penalty term in (4). The fitting term has no sparse representation in the wavelet domain. Hence, it is discretized using a continuous piecewise bilinear basis of finite elements.[9]

By combining the two representations we obtain the discretization of the MAP estimate in the wavelet domain

$$(\hat{A}^T C_\eta^{-1} \hat{A} \boldsymbol{W}^{-1} + \alpha D)c = \boldsymbol{W}^{-T} \hat{A}^T C_\eta^{-1} s, \tag{8}$$

where $\hat{A}$ is the atmospheric tomography operator in the finite element domain and $\boldsymbol{W}$ is a mapping between the finite element and the wavelet domain. The operator C_η^{-1} denotes the inverse covariance matrix of the noise and D is a diagonal approximation of C_ϕ^{-1} in the frequency domain. As we have an approximation of C_ϕ^{-1} we introduce a scalar parameter α for tuning the balance between the fitting and the regularizing terms.[10] The vector c is a concatenation of all wavelet coefficients of all turbulence layers and the vector s is the concatenation of all SH sensor measurements from all guide star directions.
For the sake of simplicity we denote the left-hand side operator of Equation (8) by

$$M := (\hat{A}^T C_\eta^{-1} \hat{A} \boldsymbol{W}^{-1} + \alpha D). \tag{9}$$

Equation (8) is solved by applying a few iterations of a Jacobi preconditioned conjugate gradient (PCG) algorithm with the operator M. The PCG is started with an initial guess provided from the previous solution, which we refer to as warm restart.

The basic operations of the algorithm are the discrete wavelet transform (DWT), bilinear interpolation and matrix-vector multiplications. The essential input of the algorithm is the measurement data s and the output is the surface of the mirrors a. The wavelet reconstructor is outlined in the algorithm below. The superscript indices (-1), (0) and (1) denote the previous, the current and the next step of the open or closed loop, respectively. The superscript indices $(0, 1)$ denote an intermediate step.

Algorithm 3.1: Reconstruction Algorithm

Input : $s = (s_g)_{g=1}^G$ (measurement vector)
$c^{(0)} = (c_\ell)_{\ell=1}^L$ (wavelet coefficient vector)
$r^{(0)}$ (residual coefficients vector)
$b^{(0)}$ (right-hand side vector)
$a^{(-1)}, a^{(0)}$ (current and previous DM shapes)
gain (scalar parameter to control weight between current and reconstructed DM shape)

Output : $a^{(1)}$ (next DM shapes)

if *loop* = *closed* **then**
  $s = s + \boldsymbol{\Gamma} a^{(-1)}$ `/* compute pseudo-open loop data */`
$b^{(1)} = \mathbf{W}^{-T} \mathbf{A}^T \mathbf{C}_\eta^{-1} s$ `/* compute new RHS */`
$r^{(0,1)} = b^{(1)} - \mathbf{M} c^{(0)} = (b^{(1)} - b^{(0)}) + r^{(0)}$ `/* update residual */`
$(c^{(1)}, r^{(1)}) = \mathrm{PCG}(c^{(0)}, r^{(0,1)})$ `/* apply PCG */`
$\tilde{a} = \mathbf{F} \mathbf{W}^{-1} c^{(1)}$ `/* obtain mirror shapes */`
if *loop* = *closed* **then**
  $a^{(1)} = a^{(0)} + \mathrm{gain} \cdot (\tilde{a} - a^{(-1)})$ `/* closed loop control */`
else if *loop* = *open* **then**
  $a^{(1)} = (1 - \mathrm{gain}) \cdot a^{(0)} + \mathrm{gain} \cdot \tilde{a}$ `/* open loop control */`

4. PERFORMANCE ANALYSIS

This section is dedicated to the complexity analysis of the real-time computing (RTC) algorithms. The algorithm implementations follow the specifications provided in Section 3. An estimation of the system memory capabilities, necessary to implement the most demanding tasks, is also reported. The analysis treats the hard real-time as well as the soft real-time tasks.

Before we go deeper into the performance analysis we need to define the parameters that determine the complexity of the algorithms. They are listed in the following table.

Description	**Variable**
Number of guide stars	G
Number of LGS	G_{LGS}
Number of NGS	G_{NGS}
Number of layers	L
Number of mirrors	M
Number of subapertures	n_s^2
Number of layer discretization points	n_{lay}^2
Number of actuators	n_a^2
Number of PCG iterations	n_{iter}
Number of modes used in the modal description of each layer	n_{ml}

Table 1. Important parameters for performance analysis.

4.1 Floating point operations

4.1.1 Hard real-time analysis

The hard real-time computational costs for the MVM are dominated by two contributions, which require about the same amount of operations: the POL slopes computation and the computation of the deformable mirror commands.

Computation step	**Floating Point Operations**
POL Data Computation	$2n_s^2n_a^2-n_s^2+6n_s$
Applying (FR) matrix	$2n_a^2n_s^2-n_a^2$

Table 2. HRT FLOPs for MVM.

The most time consuming steps for FEWHA, as it is presented in Algorithm 3.1, are the application of M in the atmospheric tomography step and partially in computing the right hand side and the fitting step. An important feature of FEWHA is the ability to present almost all of its components in a matrix-free way, which leads to a substantial reduction in terms of FLOPs.

Computation step	**Floating Point Operations**
POL Data Computation	$(8n_s^2+2n_s)G$
Right Hand Side Operator	$[14G_{LGS}+2G_{NGS}+6G]n_s^2+2Gn_s+[(L-1)(8.5G_{LGS}+10G_{NGS})+L(G-1)](n_s+1)^2+(L+1)$ $+\frac{88}{3}(4n_{lay}^2-1)+n_{lay}^2L$
Update Residual	$2n_{lay}^2L$
PCG iterations	$\{[14G_{LGS}+2G_{NGS}+12G]n_s^2+4Gn_s+[(L-1)(15.6G_{LGS}+18G_{NGS})+L(G-1)](n_s+1)^2$ $+L\frac{176}{3}(4n_{lay}^2-1)+13n_{lay}^2L\}n_{iter}$
Fitting	$8(L-1)M(n_s+1)^2+L\frac{88}{3}(4n_{lay}^2-1)+Ln_{lay}^2$
Control	$3n_a^2$

Table 3. FLOPs for FEWHA.

4.1.2 Soft real-time analysis

The main contribution in terms of computational costs for the soft real-time part of the MVM is the update of the FR matrix and there, in particular, the matrix inversion. The following computational costs are estimated assuming that the matrix inversion is performed through Cholesky decomposition.

Computation step	Floating Point Operations
Computation of R	$n_s^6+4n_{ml}n_s^4L+(2n_{ml}^2L-n_{ml}L+2)n_s^2$
Computation of F	$n_a^3+(2n_{ml}+2n_{ml}L+n_{opt}-1)n_{phi}^2+(2n_{ml}^2L+n_{opt}^2n_{ml}L-3n_{ml}L)n_a$
Computation of FR	$(2n_{ml}L-1)n_a^2n_s^2$

Table 4. SRT FLOPs for MVM.

One benefit of FEWHA is that there is no additional soft real-time. Because it is an iterative approach, there is no need to pre-compute the FR matrix and the whole computation is done in hard real-time.

4.2 Memory usage

This work only takes into account the main contributions in terms of matrices required to perform the MVM. This is, primarily, storing the huge FR matrix. This matrix has a dimension of $n_a^2 \times n_s^2$, which can become very big, as e.g., for ELT-sized problems. An optimization in terms of memory usage is possible, but usually involves a trade-off with respect to the computational time. Here, no optimization strategies are considered.

As it was the case for the FLOPs analysis, for FEWHA the heaviest part is applying the matrix M in the PCG step. The feature of presenting almost all of its components in a matrix-free way leads also to a substantial reduction in storage of the components. Table 5 shows in detail the memory usage for FEWHA.

Computation step	Memory Usage
POL Data Computation	$(2n_s^2+n_sn_{phi})G$
Right Hand Side Operator	$G+[0.7G_{LGS}+G_{NGS}](L-1)(n_s+1)^2+Ln_{lay}^2$
Update Resiudal	$L2^{2J_l}$
PCG	$2Gn_s^2+G+[0.7G_{LGS}+G_{NGS}](L-1)(n_s+1)^2+Gn_sn_{phi}+4Ln_{lay}^2$
Fitting	$G(L-1)(n_s+1)^2+2Ln_{lay}^2$
Control	Mn_a^2

Table 5. Memory usage for FEWHA.

4.3 Parallelization

The hard- and soft real-time computational costs for the MVM are dominated by three contributions: computing the POL slopes and the deformable mirror commands in hard real-time and the FR matrix update in soft real-time. All these operations consists of matrix-vector multiplications and can be easily and efficiently parallelized using standard linear algebra libraries.

For FEWHA parallelization is slightly more difficult. In fact, the algorithm allows two types of parallelization, namely, global and local. By global parallelization we understand the block decomposition of the algorithm as a whole. Local parallelization, on the other hand, refers to the parallelization of the individual operators, as e.g., the wavelet transform. The two strategies are not exclusive. In fact, the combination of these two leads to a very efficient parallelization scheme. However, there is a high level of data exchange between the threads during run-time. To avoid communication overhead, a shared memory system is preferred, i.e., a system where all threads have access to the same block of physical memory. We illustrate the global parallelization strategy on the left-hand side operator M in Figure 1. The operator M, as defined in Equation (9), is applied once per PCG iteration and is the computationally heaviest part of the algorithm. Other parts of the algorithm consist of similar steps and, therefore, can be parallelized analogously. The parallelization is applied over layers L and guide stars G. However, after certain steps it is necessary to exchange the data between all threads (highlighted with dashed lines), which is not optimal for certain hardware architectures.

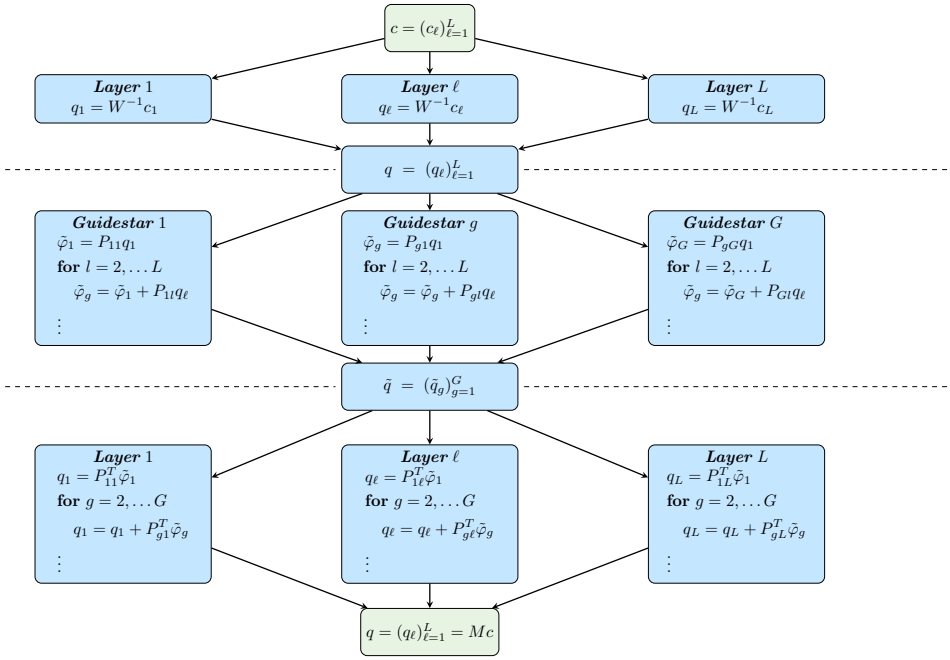

Figure 1. Parallelization of operator M

In the local parallelization scheme, we compute the SH operator Γ, the bilinear interpolation P and the discrete wavelet transform W in parallel. These operators share the same basic characteristics. They consist of a sequence of repeated instructions applied to a grid of values that can be computed independently of each other. In parallel computing, this is referred to as the single instruction, multiple data (SIMD) paradigm.[11]

5. NUMERICAL EXAMPLE

In this section, we apply the results of the theoretical performance analysis of Section 4 on the example of MAORY,[12] an adaptive optics module for the ELT based on MCAO. MAORY will use at least two deformable mirrors (including the deformable mirror in the telescope) designed to correct for different layers of turbulence. Table 6 shows all important parameter values for analyzing the computational performance of the algorithms.

Description	**Variable**	**Value**
Number of guide stars	G	9
Number of LGS	G_{LGS}	6
Number of NGS	G_{NGS}	3
Number of layers	L	6
Number of mirrors	M	4
Number of subapertures	n_s^2	80^2
Number of layer discretization points	n_{lay}^2	128^2
Number of actuators	n_a^2	81^2
Number of PCG iterations	n_{iter}	4
Number of modes used in the modal description of each layer	n_{ml}	300

Table 6. Important parameters for performance analysis.

5.1 Floating point operations

Figure 2 shows the hard real-time (HRT) and soft real-time (SRT) FLOPs for the MVM (in green) and FEWHA (in blue) for the test case of MAORY. We can observe the benefits of FEWHA compared to the MVM. For hard real-time FEWHA is 12 times faster than the MVM. For the soft real-time part, since FEWHA does not involve any significant computation, there is an evident advantage with respect to the MVM algorithm.

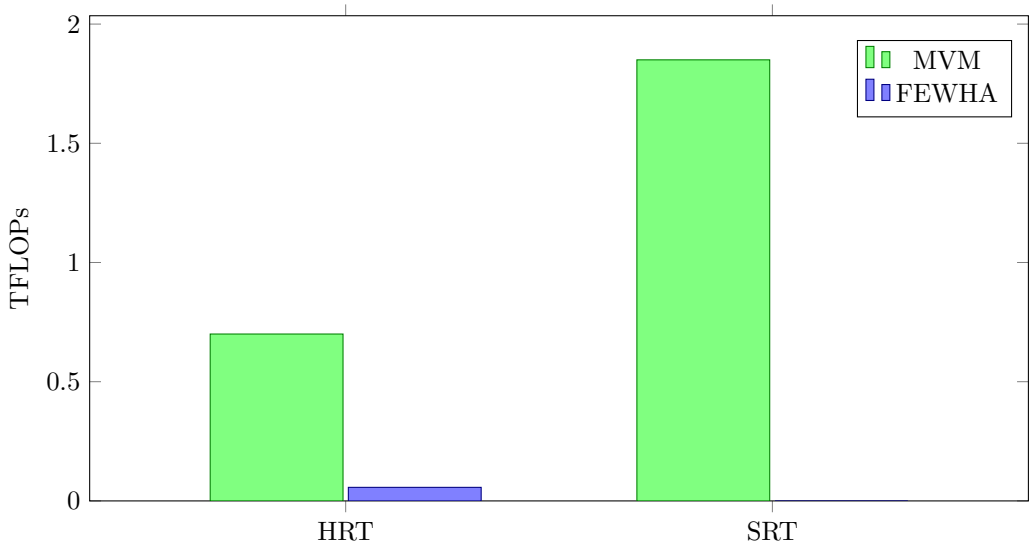

Figure 2. FLOPs for MVM and FEWHA

5.2 Memory usage

The memory usage for both algorithms is listed in Table 7. We made the assumption of having single precision (32 bit) floating point numbers. The substantial reduction of storage for FEWHA compared to the MVM is due to the ability of FEWHA to present the components in a matrix-free way.

	MVM	**FEWHA**
Hard real-time	3 GB	15 MB
Soft real-time	50 GB	-
Total	**53 GB**	**15 MB**

Table 7. Memory Usage MAORY.

5.3 RTC hardware architecture

On the basis of the computational throughput requirements analyzed above we determined a possible hardware architecture. Moreover, we have based the architectural design on the experience accumulated within the GreenFlash research activity.[13] The outcome of the GreenFlash study, in which CPU, FPGA and GPU technologies have been compared, has indicated that the most suitable technology for the computational engine of an ELT-class RTC is the GPU. Another important outcome of the study is that the FPGA technology is still superior for the smart-interfacing, i.e., to provide the interfaces between sensors and mirrors and the computational core.

Taking into account the computational load determined for the MVM in Figure 2, a single GPU would not achieve the computational throughput needed to meet the real-time requirements of MAORY. Thus, we suggest to use an off-the-shelf product, namely, the DGX-1 supercomputer by NVIDIA.[14] This workstation is equipped with 8 Tesla V100 GPUs connected by a dedicated ultra high speed bus (NVLINK). We suggest to use the DGX-1 for the hard as well as the soft real-time engine. Since the computational load of FEWHA is significantly lower, in theory, one Tesla V100 should provide enough computational power to meet the real-time requirements. We already started the implementation of FEWHA on a GPU to validate this assumption. Moreover, for FEWHA

no additional soft real-time engine is needed. Altogether, this leads to significant lower hardware requirements for FEWHA compared to the MVM.

6. CONCLUSION AND FUTURE WORK

In this paper, we compared the frequently used MVM method to a novel, iterative approach called FEWHA. We have studied the computational performance of both algorithms in terms of FLOPs, memory usage and parallelization possibilities. We observed benefits in terms of parallelization for the MVM. Since it is based on simple matrix-vector operations, parallelization is very easy. In contrast, for FEWHA it is slightly more complicated and a shared memory system would be preferable to deal with the high level of data exchange between the threads during run-time. We applied this theoretical performance analysis to the test case of MAORY, a first light instrument of the ELT. There, we were able to show the significant performance benefits of FEWHA over the MVM. Finally, we derived a possible RTC hardware architecture for the two algorithms.

Since our feasibility study showed that the performance benefits of FEWHA over the MVM are significant, we started the development of FEWHA on GPU. The implementation in CUDA C/C++ on a Tesla V100 is still an ongoing activity. Moreover, we are working on analysing the AO performance of the two algorithms for different parameter settings. In future, we want to be able to provide a detailed comparison of the MVM and FEWHA for several benchmarks.

ACKNOWLEDGMENTS

This work is funded by the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie Grant Agreement No. 765374.

REFERENCES

[1] Tokovinin, A. and Viard, E., "Limiting precision of tomographic phase estimation," *JOSA A* **18**, 873–882 (04 2001).

[2] Tokovinin, A., Louarn, M. L., and Sarazin, M., "Isoplanatism in a multiconjugate adaptive optics system," *J. Opt. Soc. Am. A* **17**, 1819–1827 (Oct 2000).

[3] Ellerbroek, B. L. and Vogel, C. R., "Inverse problems in astronomical adaptive optics," *Inverse Problems* **25**, 063001 (apr 2009).

[4] Davison, M., "The ill-conditioned nature of the limited angle tomography problem," *SIAM Journal on Applied Mathematics* **43**(2), 428–448 (1983).

[5] Marchetti, E., Hubin, N. N., Fedrigo, E., Brynnel, J., Delabre, B., Donaldson, R., Franza, F., Conan, R., Louarn, M. L., Cavadore, C., Balestra, A., Baade, D., Lizon, J.-L., Gilmozzi, R., Monnet, G. J., Ragazzoni, R., Arcidiacono, C., Baruffolo, A., Diolaiti, E., Farinato, J., Vernet-Viard, E., Butler, D. J., Hippler, S., and Amorin, A., "MAD: the ESO multiconjugate adaptive optics demonstrator," in [*Adaptive Optical System Technologies II*], Wizinowich, P. L. and Bonaccini, D., eds., **4839**, 317 – 328, International Society for Optics and Photonics, SPIE (2003).

[6] Dainty, C., "Book Review: Imaging Through Turbulence. by Michael C. Roggemann and Byron Welsh," *Optical Engineering* **35**(11), 3361 – 3361 (1996).

[7] Fusco, T., Conan, J.-M., Rousset, G., Mugnier, L. M., and Michau, V., "Optimal wave-front reconstruction strategies for multiconjugate adaptive optics.," *Journal of the Optical Society of America. A, Optics, image science, and vision* **18 10**, 2527–38 (2001).

[8] Béchet, C., Le Louarn, M., Clare, R., Tallon, M., Tallon-Bosc, I., and Thiébaut, É., "Closed-loop ground layer adaptive optics simulations with elongated spots : impact of modeling noise correlations," (2010).

[9] Yudytskiy, M., Helin, T., and Ramlau, R., "Finite element-wavelet hybrid algorithm for atmospheric tomography," *J. Opt. Soc. Am. A* **31**, 550–560 (Mar 2014).

[10] Helin, T. and Yudytskiy, M., "Wavelet methods in multi-conjugate adaptive optics," *Inverse Problems* **29**, 085003 (jul 2013).

[11] Cardoso, J. M., Coutinho, J. G. F., and Diniz, P. C., "Chapter 2 - high-performance embedded computing," in [*Embedded Computing for High Performance*], Cardoso, J. M., Coutinho, J. G. F., and Diniz, P. C., eds., 17 – 56, Morgan Kaufmann, Boston (2017).

[12] Schreiber, L., Diolaiti, E., Arcidiacono, C., Baruffolo, A., Bregoli, G., Cascone, E., Cosentino, G., Esposito, S., Felini, C., Foppiani, I., Ciliegi, P., Feautrier, P., and Torroni, P., "Dimensioning the MAORY real time computer," in [*Adaptive Optics Systems V*], Marchetti, E., Close, L. M., and Véran, J.-P., eds., **9909**, 1353 – 1363, International Society for Optics and Photonics, SPIE (2016).

[13] Gratadour, D., Dipper, N., Biasi, R., Deneux, H., Bernard, J., Brule, J., Dembet, R., Doucet, N., Ferreira, F., Gendron, E., Laine, M., Perret, D., Rousset, G., Sevin, A., Bitenc, U., Geng, D., Younger, E., Andrighettoni, M., Angerer, G., and Rouaud, C., "Green flash: energy efficient real-time control for ao," 99094I (07 2016).

[14] NVIDIA, "Nvidia dgx-1: The essential instrument for ai research." https://www.nvidia.com/en-us/data-center/dgx-1/ (2019).