Monthly Notices
of the
ROYAL ASTRONOMICAL SOCIETY

MNRAS **486,** 1774–1780 (2019)
Advance Access publication 2019 March 29

doi:10.1093/mnras/stz918

Efficient implementation of pseudo open-loop control for adaptive optics on Extremely Large Telescopes

A. G. Basden,★ D. Jenkins, T. J. Morris, J. Osborn and M. J. Townson
Department of Physics, Centre for Advanced Instrumentation, South Road, Durham DH1 3LE, UK

Accepted 2019 March 27. Received 2019 March 25; in original form 2018 July 11

ABSTRACT

Closed-loop adaptive optics systems that use minimum mean square error wavefront reconstruction require the computation of pseudo open-loop wavefront slopes. These techniques incorporate a knowledge of atmospheric statistics that must therefore be represented within the wavefront slope measurements. These pseudo open-loop slopes are computed from the sum of the measured residual slopes and the reconstructed slopes that would be given if the deformable mirror was flat, generally involving the multiplication of an interaction matrix with actuator demands from the previous time-step. When using dense algebra, this multiplication is computationally expensive for Extremely Large Telescopes, requiring a large memory bandwidth. Here, we show that this requirement can be significantly reduced, maintaining mathematical correctness and significantly reducing system complexity. This therefore reduces the cost of these systems and increases robustness and reliability.N

Key words: Instrumentation: adaptive optics – Instrumentation: miscellaneous.

1 INTRODUCTION

Adaptive optics (AO; Babcock 1953; Hardy 1998) is now a mainstream technology, and essential for the next generation of Extremely Large Telescopes (ELTs) , including the European Southern Observatory ELT, the Thirty Metre Telescope (TMT), and the Giant Magellan Telescope, which will have light collecting areas equivalent to primary mirror diameters of at least 20 m. To optimize AO performance, particularly for wide field of view systems and at low signal levels, it is necessary to use minimum variance wavefront reconstruction techniques (Ellerbroek, Gilles & Vogel 2003), also known as minimum mean square error or maximum a posteriori reconstruction. These methods use atmospheric statistics to optimize the wavefront reconstruction. However, to do this, they must rely on having open-loop slope measurements as input, i.e. those that represent the statistics of the atmosphere, rather than the residual slope measurements more commonly used by single conjugate AO (SCAO) systems. This can be achieved in two ways. Either the wavefront sensors (WFSs) can be operated in open-loop, as was the case for early phases of the CANARY instrument (Myers et al. 2008), or the system can use pseudo open-loop (POL) slope measurements that are reconstructed from the residual slope measurements and previous AO system states (effectively the shape of the deformable mirrors, DMs). Given that all proposed AO systems for the ELTs operate in closed-loop (or partial closed-loop in the case of MOSAIC Hammer et al. 2014), the latter technique must be used.

1.1 Pseudo open-loop slope computation

In a closed-loop AO system, the WFSs measure the residual wavefront slopes, i.e. the wavefront after correction by the DMs. Fortunately, when DMs are linear (or can be linearized), the shape of the DM can be estimated based on the DM demands from the previous frame (or frames, in the case of non-integer frame delay). To compute the POL slopes, the known DM shape is used to reconstruct the corresponding slope measurements that would be present if the DM surface was flat. This operation typically involves the multiplication of an interaction matrix with the DM demands. These slope measurements are then added to the residual slope measurements obtained from the WFSs to give the POL slopes:

$$\boldsymbol{s}_n^{\mathrm{POL}} = \boldsymbol{s}_n^{\mathrm{RES}} + \mathsf{P} \cdot \boldsymbol{a}_{\mathrm{n}-1}, \tag{1}$$

where s represents the wavefront slopes (POL and residual, respectively), P is the interaction matrix (which can be measured in a conventional way by poking the DM), and a represents the actuator demands from the previous frame (n−1). In the case where the AO loop delay is not equal to one frame, a linear combination of actuator demands from neighbouring frames should be used:

$$\boldsymbol{a} = (1-d)\boldsymbol{a}_{\mathrm{n-D}} + d\boldsymbol{a}_{\mathrm{n-(D+1)}}, \tag{2}$$

★ E-mail: a.g.basden@durham.ac.uk

where d is the fractional frame delay (between 0 and 1) and D is the total number of frames delay rounded down to the nearest integer. The total delay is therefore given by $d + D$.

Although for some AO systems, P can be sparse, for the ESO ELT it is a dense matrix, of size equal to the total number of slope measurements by the total number of actuators. This is because the ELT M4 mirror is treated as a modal DM, with internal electronics converting the applied modes (which we here call actuators for cohesiveness) into actual actuator values (after possible modifications in applied mode strengths to reduce stresses on the mirror surface). Therefore, the operation $\mathsf{P} \cdot \boldsymbol{a}$, which must be computed every frame, is a large dense matrix vector multiplication, requiring significant memory bandwidth (since the matrix has to be loaded from memory into the computational units every WFS frame, as it is too large to remain in memory cache). This operation, along with wavefront reconstruction, represents a bottleneck in the AO real-time control pipeline. The POL computation therefore represents a requirement for a significant additional hardware cost in the AO system design. We note that P is the same size as the reconstruction matrix, with required memory bandwidth scaling as the fourth power of telescope diameter.

In Section 2, we introduce a straightforward simplification that can be used to greatly reduce the computational requirements of POL computation, whilst maintaining numerical exactness. In Section 3, we discuss the implications that this method has, and additional requirements on the AO system supervisor. We also consider several AO system designs and estimate the hardware savings that this method delivers. We conclude in Section 4.

2 SIMPLIFICATION OF POL COMPUTATION

An AO real-time controller updates the DM demands at every WFS time-step. The DM demands, $\boldsymbol{a}$, required for a DM are typically computed using

$$\boldsymbol{a}_n = g\mathsf{R} \times \boldsymbol{s}_n^{\mathrm{POL}} + (1-g)\,\boldsymbol{a}_{n-1}, \tag{3}$$

where R is the reconstruction matrix (of size equal to the number of actuators by the number of slope measurements) and g is the loop gain. Expanding $\boldsymbol{s}^{\mathrm{POL}}$ gives

$$\boldsymbol{a}_n = \mathsf{R} \times \boldsymbol{s}_n^{\mathrm{RES}} + \mathsf{R} \times \mathsf{P} \times \boldsymbol{a}_{n-1} + (1-g)\,\boldsymbol{a}_{n-1}, \tag{4}$$

$$= g\mathsf{R} \times \boldsymbol{s}_n^{\mathrm{RES}} + \mathsf{Q} \times \boldsymbol{a}_{n-1} + (1-g)\,\boldsymbol{a}_{n-1}, \tag{5}$$

where Q is the pre-computed matrix multiplication product of R and P multiplied by g, a square matrix of size equal to the number of actuators squared. It should be noted that in this latter equation, the POL slopes are never explicitly computed, and therefore not available to the real-time control system. We hereafter refer to this approach as *Implicit POL*, while the conventional approach is referred to as *Explicit POL*. This approach was first suggested by Wang & Ellerbroek (2012), though did not include the gain term.

We can therefore consider the differences in computational requirements between these equations. In each, two matrix–vector multiplication operations are required. In the original equation, equation (3), computation of $\boldsymbol{s}^{\mathrm{POL}}$ requires multiplication of the matrix P (of size equal to the number of slopes by the number of actuators) by a vector of size equal to the number of actuators, followed by multiplication of R, (of size equal to P^T) by $\boldsymbol{s}^{\mathrm{POL}}$ (of size equal to the number of slopes). The proposed version, equation (5), requires multiplication of Q (a square matrix of size equal to the number of actuators), with the actuator vector, and then a multiplication by R. These matrices are summarized in Table 1.

Table 1. A table summarizing matrix dimensions for the implicit and explicit POL cases, with n_{act} giving the total number of actuators and n_{slopes} giving the total number of slopes. It should be noted that the number of slopes can be significantly larger than the number of actuators.

Matrix	Dimensions	Explicit POL	Implicit POL
R	$n_{\mathrm{act}} \times n_{\mathrm{slopes}}$	Yes	Yes
P	$n_{\mathrm{slopes}} \times n_{\mathrm{act}}$	Yes	No
Q	$n_{\mathrm{act}} \times n_{\mathrm{act}}$	No	Yes

2.1 Real-time control system designs

In order to compute the memory bandwidth requirement reduction that can be made by implicit POL computation, and hence the reduction in hardware requirements, we must consider designs for some proposed AO real-time control systems. We concentrate on ELT systems, including SCAO, laser tomographic AO (LTAO), multiconjugate AO (MCAO), and multiobject AO (MOAO), mapping these designs to currently available computational hardware. We note that the TMT architecture (Dunn et al. 2018) uses an explicit POL computation. A flexible real-time controller, such as the Durham AO real-time controller (DARC; Basden et al. 2010b), will be able to use both explicit and implicit POL computation, given sufficient underlying computational hardware.

2.1.1 ELT SCAO

The ELT SCAO real-time control system will be comprised of a single computational node, which receives WFS pixels, and computes the corresponding DM demands. When POL is not performed, it has been shown that a single Xeon Phi can process ELT SCAO at about 1.2 kHz (Jenkins, Basden & Myers 2018), in excess of the 1 kHz instrumental requirement. Table 2 gives the memory bandwidth requirements for both explicit and implicit POL computation, and Fig. 1 shows measured SCAO latency as a function of WFS frame rate for the explicit and implicit POL cases, and for the case without POL calculation. We find that the maximum WFS frame rate that can be processed using the explicit POL calculation is 500 Hz, while when using implicit POL calculation, this increases to 600 Hz, due to the reduced memory bandwidth requirement. The latency is also lower. Of course, the case without POL computation can reach highest frame rates (we show up to 750 Hz for the camera model that we use here) since memory bandwidths are significantly reduced. However, in this case, the AO control algorithm is sub-optimal since minimum variance reconstruction cannot be used. These measurements were taken using DARC, on an Intel Xeon Phi 7250 processor. The configuration of DARC is described by Jenkins, Basden & Myers (2019). In particular, we use a camera simulator to provide pixels in the same format as the ESO ELT WFSs.

For higher frame rates, the Xeon Phi used here is unable to process WFS information fast enough, and therefore dropped frames and eventual real-time control system failure result due to the increased latency. Although the memory bandwidth requirements are theoretically achievable using a single Xeon Phi processor (480 GBs^{-1}), or a quad-CPU Intel Scalable Processor design (512 GBs^{-1}), in practice, it is not possible to compute the POL computation within a single node while achieving higher frame rates or reduced latency, due to

Table 2. A table comparing real-time controller memory bandwidth requirements for the implicit and explicit POL computations. These numbers also include the wavefront reconstruction. LTAO and MCAO slope count does not include the NGSs, but as these are low order, the results are not affected significantly.

AO system	Number of slopes	Number of actuators	Frame rate	Bandwidth for explicit POL	Bandwidth for implicit POL
ELT SCAO	9232	5318	1 kHz	393 GBs^{-1}	310 GBs^{-1}
ELT LTAO	55 392	5326	500 Hz	1180 GBs^{-1}	647 GBs^{-1}
ELT MCAO	55 392	6326	500 Hz	1402 GBs^{-1}	781 GBs^{-1}
ELT MOAO	61 504	22 666	250 Hz	1722 GBs^{-1}	1515 GBs^{-1}
TMT NFIRAOS	35 808	7675	800 Hz	1859 GBs^{-1}	1068 GBs^{-1}

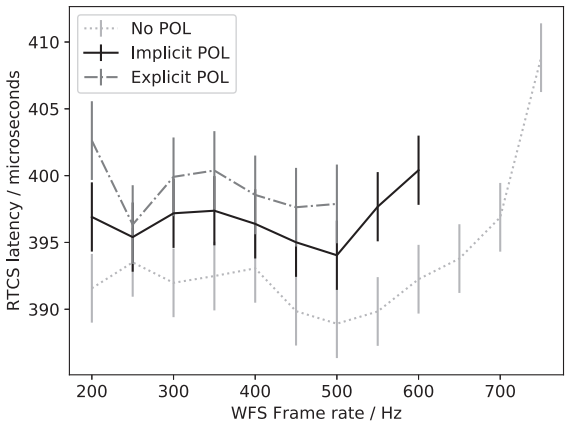

Figure 1. A figure showing AO system latency as a function of WFS frame rate for explicit and implicit POL computation. For comparison, the case without POL is also shown, though this would result in lower AO performance since minimum variance reconstruction could not be used. The explicit POL case cannot operate at frame rates above 500 Hz, while using implicit POL can extend the maximum frame rate to above 600 Hz.

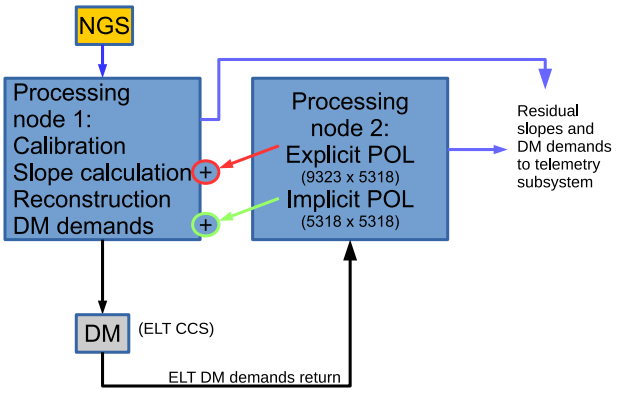

Figure 2. A possible ELT SCAO real-time control system architecture based on two CPU nodes. Explicit POL computation data flow is denoted by red arrows, while implicit POL computation data flow is denoted by the green arrows. The explicit POL computation result is required earlier in the pipeline, and therefore has more stringent latency requirements. Pixel and telemetry information flow is represented by the blue arrows, while DM demands are represented by the black arrows.

the other operations necessary (pixel reception, calibration, etc.). In particular, the Xeon Phi has poor single thread performance, so aspects of the real-time control system that cannot be parallelized, such as pixel acquisition, can significantly impact performance. Therefore, a second computational node could be used to receive the applied DM demands (as sent from the ELT central control system, CCS), and compute the component of the implicit or explicit POL slopes. This node would then send either slope adjustments (explicit POL) or actuator adjustments (implicit POL) to the first computational node, as shown in Fig. 2.

In this case, the use of implicit POL does not reduce the required hardware (two computational nodes are still required). However, AO system latency and jitter (variation in latency) can be improved since the implicit matrix–vector operation is smaller, and the result combined later, i.e. added to actuators after residual wavefront reconstruction, rather than being added to residual slopes before wavefront reconstruction. We note that a hardware accelerator, for example, a graphics processing unit (GPU) could be used in place of the second node, though here we consider only CPU-based designs.

In the case that there is a variation in arrival times of the DM demands sent from the ELT CCS (e.g. due to the network, or non-deterministic algorithms), the implicit POL method has a larger computational window available to mitigate the effect of this jitter.

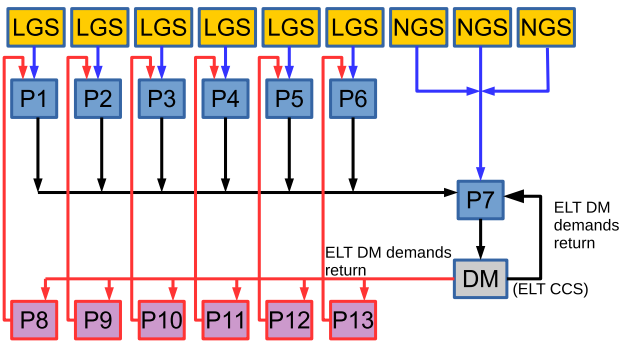

Figure 3. A possible ELT LTAO real-time control system architecture based on seven CPU nodes (or 13, in the case of explicit POL computation). Explicit POL computation data flow and extra components are shown in red (requiring six extra nodes, P8–P13). P represents a processing node (numbered from 1 to 13), with nodes 1–7 performing image calibration, slope calculation, and partial wavefront reconstruction, node P7 additionally performing DM vector summation, and nodes 7–13 computing POL information (all in node 7 in the case of implicit POL). The blue arrows represent pixel data; the black arrows show DM data.

2.1.2 ELT LTAO

The proposed ELT LTAO system uses six laser guide stars (LGSs) and between one and three low order natural guide stars (NGSs) to control the telescope DM via the CCS . Fig. 3 shows a schematic design for the real-time control system, with each WFS being

sent to a single processing node. These nodes each perform image calibration, residual slope calculation and partial wavefront reconstruction. Finally, these nodes compute partial DM vectors which are then collected by the NGS node (i.e. a 'gather' node) and summed together to give the final DM demands, before being sent to the DM .

In the case of implicit POL reconstruction, the POL computation is performed on this NGS node, and therefore only this node needs to receive the applied DM demands from the CCS . This POL calculation involves a square matrix–vector multiplication with dimensions equal to the number of actuators.

When using implicit POL calculation, the operations performed on nodes P1–P6 are essentially very similar in form to an SCAO calculation without any POL , i.e. a residual slope computation, and reconstruction using these. Fig. 1 shows that using Xeon Phi nodes, frame rates in excess of 750 Hz could be achieved here.

When using explicit POL calculation, on the WFS nodes, image calibration and residual slope computation are performed. The POL residuals are then added, followed by partial wavefront reconstruction, before computing the partial DM demands that are then collected and summed by the NGS node. These DM demands are sent to the ELT CCS . The ELT CCS will then return the actually applied demands.

If these applied demands are sent to the WFS nodes, P1–P6, the POL calculation for each WFS is then performed, so that the POL slope residuals can be added to the residual slope measurements. We note that the size of these operations on each WFS node is essentially identical to that of an SCAO system, and therefore as shown in Fig. 1, a maximum frame rate of about 500 Hz will be achievable using Xeon Phi nodes.

In the likely case that higher frame rates will be required, the applied demands can be sent from the CCS to additional nodes, P8–P13. On these nodes, DM demands are then multiplied by an interaction matrix to give the explicit POL slope residuals, to be sent to the WFS nodes for addition during the next frame.

The additional communications required for the explicit POL computation will reduce reliability. If a UDP protocol is used (including multicast), packets can go missing, while using a TCP protocol will increase jitter. In some situations, where delays mean that POL slope residuals are computed very close to the time at which they are required, timing issues may result, with different WFSs either being delayed, or using measurements from a previous frame. With the implicit POL approach (which is significantly simpler), this will not happen since all POL computation is performed within a single node. Simplicity within a real-time control system design is advantageous.

We note that in the explicit POL case, rather than using additional nodes to achieve higher frame rates, it would also be possible to use different computational hardware. For example, a quad-CPU Intel Xeon Gold or Platinum system would achieve slightly higher memory bandwidth than a Xeon Phi, with improved single-core performance. Unfortunately, the cost of a single such node would be at least double that of two Xeon Phi nodes, based on current prices. We also note that we find that theoretical performance metrics do not reliably relate to full real-time control system performance, when real camera packets are included.

In the case of explicit POL reconstruction, six additional nodes are required to perform this calculation (shown in red) before sending the slope adjustments to the corresponding WFS processing node. Each of these additional nodes must also receive the applied DM demands from the CCS . This therefore represents a significant

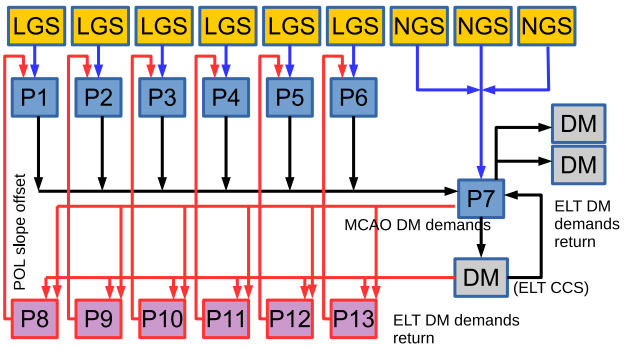

Figure 4. A possible ELT MCAO real-time control system architecture based on seven (or 13) CPU nodes. Explicit POL computation data flow and extra components are shown in red (requiring six extra nodes, P8–P13), and DM demands are required to be sent from the CCS and from node P7 to the additional nodes. P represents a processing node (numbered from 1 to 13). For the two additional DMs, it is assumed that the requested and applied DM demands are equal (as is usually the case). The blue arrows represent pixel data; the black arrows show DM data.

increase in complexity when compared with the implicit POL approach.

Telemetry information is not shown in Fig. 3 for clarity. However, the necessary telemetry information (discussed further in Section 3.3) includes residual slopes from nodes P1–6, POL slopes from nodes P1–6 when using explicit POL computation, and CCS return demands from P7.

We note that while a single node based on currently available hardware (e.g. a Xeon Phi, AMD EPYC or Intel Scalable processor solution) may be able to process a single WFS and its corresponding POL calculation (in the explicit case), such a solution would likely be unstable, with increased jitter and the risk of dropped frames, particularly at moderate frame rates (e.g. 500 Hz) and above. It would also be more susceptible to problems arising from the timing of arrival of CCS information. For this reason, Fig. 3 adds additional nodes for the POL computation.

2.2 ELT MCAO

From the point of view of the real-time control system, the ELT MCAO case is similar to the LTAO case. However, the differences are that the MCAO system will have an increased number of actuators to compute due to the two additional non-zero-conjugate DMs . We assume here that the DM demand return will not be returned from these DMs (as was the case for the CCS), but will be computed and known by node P7, as shown in Fig. 4. Therefore, in the case of implicit POL , these demands are not sent to other nodes, and are used for implicit POL computation on node P7. However, in the case of explicit POL , these demands will be sent from P7 to nodes P8–13, so that the explicit POL computations can be performed. This represents an additional increase in complexity, and greater potential for instability, particularly in the case of real world situations where Ethernet packets can get lost.

2.3 ELT MOAO

The proposed ELT MOAO system uses four LGS and four high-order NGS . In addition to the CCS actuators, 10 open-loop DMs are also used (with a lower spatial order than the CCS mirror, having

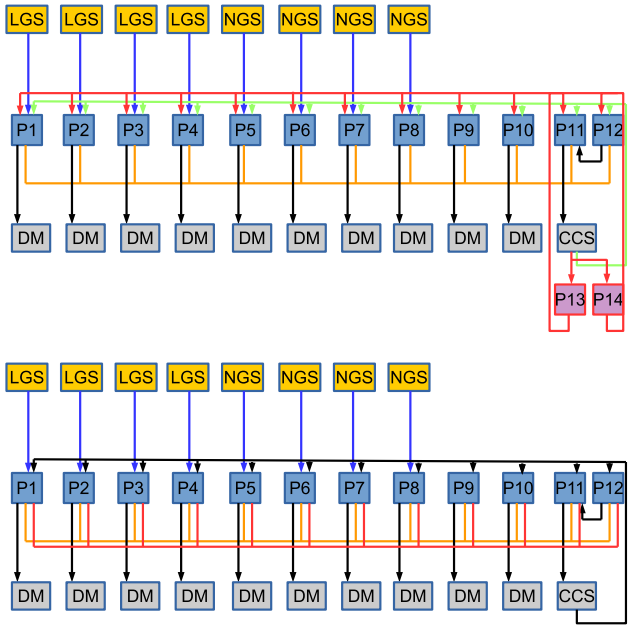

Figure 5. (a) A possible ELT MOAO real-time control system architecture based on 12 CPU nodes (or 14 nodes when explicit POL computation is required). The extra components and data flow required for explicit POL computation are shown in red. The orange lines represent the multicast of slope measurements between nodes. The green lines represent distribution of CCS demand output when implicit POL is used. The blue lines represent pixel flow and the black lines represent DM demand flow. (b) An alternative ELT MOAO architecture, using 12 CPU nodes in both cases. Here, in the explicit case, partial POL reconstruction is performed on each node (using the CCS demands). The resulting slopes are then broadcast to all other nodes (the red lines). Computation of POL slopes is shared equally between nodes, as otherwise, the bandwidth requirements for P11 and P12 are likely to be too great.

48 × 48 actuators). Fig. 5 shows possible schematic designs for the real-time control system. In each case, images from each WFS are sent to a single processing node, which computes the corresponding wavefront slopes. These slopes are then distributed (multicast) to all other nodes, so that all nodes now have all slope measurements. Wavefront reconstruction is performed (in a pipelined fashion, as soon as the corresponding slopes become available) on each node that has sole responsibility for commanding a single DM . The only caveat here is that control of the CCS is shared by two nodes due to the high memory bandwidth requirement, with node P12 passing its computation back to P11 so that the final CCS demands can be sent. We propose the use of two nodes for this task since a minimum of 327 GBs^{-1} is required (to be able to compute the reconstruction at loop rate, with higher bandwidth being necessary for reduced latency), which results in too low spare overhead for a single node, risking system instability and increased latency.

For implicit POL computation, the applied CCS demands are sent to all processing nodes where the POL calculation is performed.

For explicit POL computation, there are two possibilities. In the first case (Fig. 5a), two additional nodes are used for the explicit POL calculation (with the same memory bandwidth requirements as nodes P11 and P12), with these slopes then being broadcast to all other nodes, which add to the residual slopes to give the POL slopes. This results in extra cost and complexity compared with the implicit calculation where the CCS output is sent directly to the processing nodes.

In the second case for explicit POL (Fig. 5b), the applied CCS demands are sent to all nodes, which then equally share the computation of POL slopes. These POL slopes are then distributed to all other nodes. In this case, the number of residual slopes and POL slope offsets computed on each node differs and so a separate broadcast of each is required, reducing reliability.

We also note that an eight node design would also be possible here. In this case, since each MOAO DM is independent, the 22 666 actuator values would need to be spread between these nodes (approximately 2834 actuators per node), and then broadcast and collected. These nodes would then need to send DM demands to one or two DMs each. Since DM demands cannot be broadcast until all computation is completed (i.e. all slopes have been reconstructed), latency is increased compared with our first approach with one DM per node, where slopes can be broadcasted as they become available, in a pipelined fashion.

2.4 Summary of POL operations

Table 2 summarizes the relative size of these operations for different proposed AO systems.

It can be seen that in the LTAO and MCAO cases, the memory bandwidth requirement is reduced by nearly 50 per cent when using the implicit POL method.

For the MOAO case, we have assumed four LGS and four NGS each with 74 × 74 sub-apertures giving a total of $N_s = 61\,504$ slopes, which is the current MOSAIC baseline (Morris et al. 2018). We also assume that there are 10 MOAO DMs each with $A_M = 1634$ actuators (48 × 48), in addition to the ELT CCS (with $A_{CCS} = 5326$ actuators).

For the explicit POL MOAO case, the memory bandwidth required is therefore

$$((A_{CCS} + 10A_M) \times N_s + A_{CCS} \times N_s) \times 4f, \quad (6)$$

where f is the frame rate, and the factor of 4 is the number of bytes in a 32-bit floating point integer. The left part of this equation represents the residual slope reconstruction, while the right part ($A_{CCS} \times N_s$) represents the explicit POL slope computation. The explicit POL slope computation can be performed either on two separate nodes (Fig. 5a), or spread over the existing nodes (Fig. 5b).

For the implicit POL MOAO case, the memory bandwidth is

$$\left((A_{CCS} + 10A_M) \times N_s + A_{CCS}^2 + 10A_{CCS} \times A_M\right) \times 4f, \quad (7)$$

where the left part of this equation represents the residual slope reconstruction, and the right part ($A_{CCS}^2 + 10A_{CCS} \times A_M$) is the implicit POL computation. For the implicit case, the memory bandwidth required for the POL computation is approximately a third of that required in the explicit case.

2.4.1 *Considerations of sparsity in the interaction matrix*

Even in the case where P can be sparse, the method that we propose here is still relevant, particularly for MCAO and LTAO cases, as it reduces communication complexity since it removes the need to send the POL slopes back to the WFS nodes, thus reducing complexity and jitter, and increasing reliability. We also note that with a sparse P matrix, the post-processing of closed-loop residuals as a batch process (described in Section 3.2) to give POL slopes for tomography benefits from the sparsity (i.e. becomes a cheaper operation), as does the $\mathsf{Q} = \mathsf{R} \times \mathsf{P}$ matrix multiplication

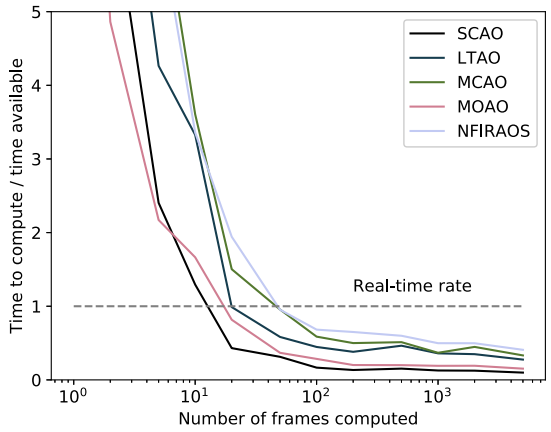

Figure 6. A figure showing the ratio of time to compute a batch of POL slopes to the time within which these slopes are produced, as a function of number of frames. When this ratio falls below unity, production of POL slopes is able to keep up with the real-time control system.

product, which then becomes achievable in shorter time-scales or with reduced hardware.

3 IMPLICATIONS

Our proposed implicit POL method means that POL slopes are not explicitly computed by the real-time control system. However, POL slopes are a necessity for parts of the AO system (such as tomography), and therefore we discuss these requirements, and solutions here.

3.1 Computation of *Q*

The implicit POL technique requires the computation of Q in equation (5). This operation is a matrix–matrix multiplication, with the matrix sizes equal to the number of actuators by the number of slopes. For the ELT SCAO case, we have $\mathsf{Q} = \mathsf{R} \cdot \mathsf{P}$ with R having dimensions 5318 × 9232 and P having dimensions 9232 × 5318. This can be performed in under a second on a standard Xeon server (dual E5-2630-v3 CPUs dating from 2015), and therefore would not require any additional hardware beyond what would be expected for an AO supervisor hardware system (used for calibration and soft real-time tasks).

For the LTAO case, these matrices increase in size to 55392 × 5326, and the computation of Q takes less than 1 min on this server. On a modern GPU , this would be far faster, and nearly trivial to implement (e.g. a call to a BLAS library). It is also a highly parallelizable problem and so could be spread across several CPU nodes using the Message Passing Interface as required. However, an update rate of once per minute is likely to be sufficient, so this is not a difficult operation. It is also not an operation within the real-time pipeline, so can be performed on hardware using a standard (non-real-time) operating system, and considerations of jitter are not necessary. The MCAO case is similar.

For the LTAO and MCAO cases, if a faster update rate is required, additional computational hardware would be necessary in the implicit POL case. However, this would be less than the additional nodes required to implement explicit POL (which would also introduce extra complexity and risk, including the necessity to pass DM demands back to the WFS nodes).

3.2 Atmospheric statistic computation

Computation of atmospheric statistics is necessary for building the minimum variance control matrix, which requires a knowledge of the C_n^2 profile, and estimates of Fried's parameter and the outer scale. This requires open-loop slope measurements (Ellerbroek & Vogel 2003). Fortunately, this information is not required in real-time on a frame by frame basis. Therefore, by recording contiguous frames of residual slopes and DM demands (a standard operation for facility AO systems), the POL slopes can be reconstructed after the event. The key advantage here is that POL slopes can be computed as a batch, i.e. several thousand frames of POL slopes can be computed simultaneously:

$$\mathsf{S}^{\mathrm{POL}} = \mathsf{S}^{\mathrm{RES}} + \mathsf{P} \times \mathsf{A}, \tag{8}$$

where S is a matrix comprised of many frames of slopes (dimensions equal to number of slopes by number of frames), and A is a matrix comprised of many frames of DM demands (dimensions equal to number of actuators by number of frames).

This calculaion involves a matrix–matrix multiplication, rather than a matrix–vector multiplication, and therefore is a compute-bound rather than a memory-bound operation. We have benchmarked this operation using a conventional server computer, with dual Xeon E5-2630-v3 CPUs (circa 2015), and find that a single server is able to keep up with the real-time data rates produced by the real-time control system when blocks of at least approximately 100 frames of slopes are processed simultaneously, as shown in Fig. 6. This corresponds to a delay of less than a second between implicit and explicit POL slopes, and this does not pose any problems for the AO control system. Table 3 summarizes the benchmark timings for the different cases given in Table 2. In all cases, it is possible to process 1 s of data in less than 1 s. On more modern hardware, processing time would be significantly reduced due to increased width vector processing units (e.g. AVX512). Fig. 6 shows the transition between memory-bound operation (for small batch sizes) to compute-bound operation. This figure shows the time taken to compute a batch of a given number of frames, and therefore the latency of this computation (from first frame to end of computation) will equal the time taken for the real-time control system to deliver this batch (i.e. frame period multiplied by batch size) plus the batch computation time.

We do not anticipate that additional hardware would be required for this task, as it could be performed by supervisory sub-system hardware (i.e. a soft-real-time system used for calibration and system supervision). Since this is not a hard real-time task, jitter in this operation is not critical.

3.3 Telemetry requirements

Using our implicit POL technique, to compute explicit POL slopes after the event, the AO system must record continuous frames of residual slopes and DM demands (actually applied, i.e. as returned from the CCS).

If an explicit POL technique is used, the AO system will usually record both residual and POL slopes (residual slopes being useful for telemetry displays and for computation of telescope offloading). In addition to this, DM demands would also usually be recorded, though it would be possible to compute these from the residual and POL slopes. In any case, the explicit POL technique requires capture of more telemetry information than the implicit case.

Table 3. A summary of computation time for explicit POL slopes derived from residual slopes and DM demands. These are computed using the SGEMM function available in the Intel Math Kernel Library.

AO system	Rate for 1000 frames	Rate for 5000 frames	Time to process 1 s data	MFLOP/frame
ELT SCAO	7.6 kHz	8.9 kHz	0.13 s	98
ELT LTAO	1.4 kHz	1.7 kHz	0.45 s	590
ELT MCAO	1.1 kHz	1.5 kHz	0.67 s	701
ELT MOAO	1.2 kHz	1.6 kHz	0.24 s	655
TMT NFIRAOS	1.5 kHz	1.9 kHz	0.52 s	550

3.4 Simplification of control system network

Another significant benefit to the implicit POL technique is a simplification of the real-time control system network requirements. As can be seen in Figs 2, 3, 4, and 5, fewer data paths between the real-time processing nodes are required. This increases system robustness since there are fewer packets that could get lost (and we assume that UDP packets will be used due to the stringent latency requirements ruling out any TCP retransmissions).

3.5 Slope linearization

Shack–Hartmann WFSs are non-linear due to **pixelization** effects. Although this effect is small, and usually ignored, under certain conditions, AO performance improvements can be achieved using a look-up table or polynomial fit to linearize slope measurements (Basden, Myers & Butterley 2010a). Solar AO WFSs can be particularly affected by this non-linearity.

Once the residual slopes have been linearized, it is not necessary to take any further linearization steps with either explicit or implicit POL schemes.

4 CONCLUSIONS

We have presented a straightforward and mathematically equivalent technique for the implicit computation of POL slopes for minimum variance AO systems, and considered the implications for common AO operation modes. This technique simplifies real-time control system design (hence reducing cost), increases robustness and reliability of the AO system, and leads to a reduction in required telemetry information. It leads to a simplified real-time control pipeline, which leads to lower latency and jitter. However when using the implicit scheme, if explicit slopes are required (for example, for atmospheric parameter estimation) an additional non-real-time computation is required, though we note that this still results in a lower server requirement (and hence cost) than the traditional explicit POL approach, and can be performed using a relatively low-end server, or shared with an existing AO supervisor server, e.g. that used for control matrix computation. Therefore, significant hardware and complexity savings can be achieved using implicit POL computation.

ACKNOWLEDGEMENTS

This work is funded by the UK Science and Technology Facilities Council grant ST/K003569/1 and consolidated grants ST/L00075X/1 and ST/P000541/1. This work is also sponsored through a grant from project 671662, a.k.a. Green Flash, funded by the European Commission under program H2020-EU.1.2.2 coordinated in H2020-FETHPC-2014.

REFERENCES

Babcock H. W., 1953, PASP, 65, 229
Basden A. G., Myers R., Butterley T., 2010a, Appl. Opt., 49, G1
Basden A. G., Geng D., Myers R., Younger E., 2010b, Appl. Opt., 49, 6354
Dunn J. et al., 2018, in Laird M. C. , Laura S., Dirk S., eds, Proc. SPIE Conf. Ser. Vol. 10703, Adaptive Optics Systems VI. SPIE, Bellingham, p. 1070317
Ellerbroek B. L., Vogel C. R., 2003, in Tyson R. K. , Lloyd-Hart M., eds, Proc. SPIE Conf. Ser. Vol. 5169, Astronomical Adaptive Optics Systems and Applications. SPIE, Bellingham, p. 206
Ellerbroek B. L., Gilles L., Vogel C. R., 2003, Appl. Opt., 42, 4811
Hammer F. et al., 2014, in Suzanne K. R., Ian S. M., Hideki T., eds, Proc. SPIE Conf. Ser. Vol. 9147, Ground-based and Airborne Instrumentation for Astronomy. SPIE, Bellingham, p. 914727
Hardy J., 1998, Adaptive Optics for Astronomical Telescopes. Oxford series in optical and imaging sciences. Oxford Univ. Press, Oxford
Jenkins D., Basden A. G., Myers R. M., 2019, MNRAS, 485, 5142
Jenkins D. R., Basden A., Myers R. M., 2018, MNRAS, 478, 3149
Morris T. et al., 2018, in Close L. M. , Schreiber L., Schmidt D., eds, Proc. SPIE Conf. Ser. Vol. 10703, Adaptive Optics Systems VI, SPIE, Bellingham, p. 1070316
Myers R. M. et al., 2008, in Hubin N., Max C., Wizinowich P.,eds, Proc. SPIE Conf. Ser. Vol. 7015, Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series. SPIE, Bellingham, p. 70150E
Wang L., Ellerbroek B., 2012, in Ellerbroek B., Marchetti E., Veran J.-P., eds, Proc. SPIE Conf. Ser. Vol. 8447, Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series. SPIE, Bellingham, p. 844723

This paper has been typeset from a TEX/LATEX file prepared by the author.