Jacobian-free coronagraphic wavefront control using nonlinear optimization

Scott D. Will,[a,b,c,*] Tyler D. Groff,[c] and James R. Fienup[a]

[a]University of Rochester, Institute of Optics, Rochester, New York, United States
[b]Space Telescope Science Institute, Baltimore, Maryland, United States
[c]NASA Goddard Space Flight Center, Greenbelt, Maryland, United States

Abstract. We describe an approach to coronagraphic focal-plane wavefront control that utilizes gradient-based nonlinear optimization along with analytical gradients obtained with algorithmic differentiation to find deformable mirror solutions. In addition to eliminating the cost of calculating a high-dimensional finite-difference Jacobian matrix, we show that this approach leads to improved asymptotic computational efficiency. With very high-actuator deformable mirrors such as the 128×128 actuators baselined for the Large UV/Optical/IR Surveyor mission concept, the proposed algorithm reduces memory consumption by approximately 95% compared to a Jacobian-based algorithm. © *2021 Society of Photo-Optical Instrumentation Engineers (SPIE)* [DOI: 10.1117/1.JATIS.7.1.019002]

Keywords: coronagraphy; high-contrast imaging; wavefront control; optimization.

Paper 20124 received Aug. 13, 2020; accepted for publication Jan. 20, 2021; published online Feb. 18, 2021; corrected Feb. 26, 2021.

1 Introduction

In recent years, direct imaging of exoplanets and exozodiacal dust clouds around nearby stars has generated significant interest in the astronomical community. Instruments called coronagraphs aim to enable this by suppressing bright starlight from a source centered on the optical axis while transmitting the signals from faint off-axis sources in a region of interest at the detector plane known as the dark zone. Direct imaging is made challenging, however, by the large flux ratios and small angular separations for astrophysical sources of interest. This is particularly true for terrestrial planets in the habitable zones of nearby Sun-like stars, for which the flux ratio can exceed 10^{10} at angular separations smaller than 0.1 arcseconds.[1] Two of the four flagship mission concepts submitted to the Astro2020 Decadal Survey, the Large UV/Optical/IR Surveyor (LUVOIR),[2] and Habitable Exoplanet Observatory (HabEx)[3] include direct imaging of Earth-like exoplanets as a primary scientific target.[4]

In this extremely high-contrast regime, optical wavefront errors on the order of tens of picometers from fabrication defects and misalignments introduce a bright speckle halo around the image of the star that overwhelms the faint image of an orbiting planet or circumstellar disk. This speckle halo evolves slowly over time in response to minute changes in the thermal and mechanical state of the observatory. Stellar coronagraphs rely critically on closed-loop wavefront sensing and control using deformable mirrors (DMs) to iteratively compensate for these aberrations over time and ensure that the scientific goals of the mission are achieved. For this reason, NASA's Exoplanet Exploration Program (ExEP) has identified wavefront sensing and control as a key technology for enhancing the capabilities of future space-based direct imaging missions.[5]

One key trade study for these missions will be the use of on-orbit wavefront sensing and control, where all computations associated with the sensing and control algorithms are carried out by the flight computer, versus ground in-the-loop operations, in which data and control commands are relayed to and from a ground-based computing node. On-orbit wavefront sensing and control is advantageous because DM commands can be updated more frequently without requiring ground communications; however, successfully implementing it is very challenging due to

*Address all correspondence to Scott Will, scott.will@rochester.edu

the demands placed on the flight computer by the wavefront sensing and control algorithms and the added complexity of deploying software in space. As a case in point, the Roman Space Telescope Coronagraph Instrument (CGI) modified its wavefront sensing and control operational mode to ground in-the-loop in response to the preliminary design review in order to reduce the complexity of its flight hardware and software.[6,7] Therefore, developing more computationally efficient algorithms is essential to providing a path toward feasibility for on-orbit operations.

To date, laboratory experiments have attained deepest contrast using model-based wavefront control algorithms such as stroke minimization (SM)[8] and electric field conjugation (EFC).[9] Such algorithms use a computer model of the coronagraph to solve an inverse problem in each wavefront control iteration given an estimate of the aberrated electric field measured at the coronagraph detector, to avoid noncommon path aberrations that would be introduced using a dedicated wavefront sensing instrument. SM and EFC find solutions by first forming a derivative matrix, called a Jacobian matrix, that models the electric field response of each pixel within the focal-plane control region to each DM actuator. This transforms the wavefront control inverse problem into a large linear system of equations that one then solves for the optimal command vector.

A disadvantage of SM and EFC is the computation required to evaluate and manipulate Jacobian matrices. Calculating a Jacobian matrix using a finite-difference approximation requires that the coronagraph model be evaluated, at minimum, once for each DM actuator; since the number of actuators for a modern coronagraph is typically in the thousands, this represents a substantial computational effort. Moreover, because the Jacobian matrix is only a first-order approximation of a nonlinear system, it must be periodically recalculated as the DM commands evolve away from the linearization point during closed-loop control. Worse still, in broadband observing scenarios, a separate Jacobian must be calculated for each controlled wavelength.

Software packages such as fast linear least-squares coronagraph optimization (FALCO)[10] have been developed specifically to address this issue by utilizing a coronagraph model highly optimized for speed; the practical improvement in calculation time depends on the specific type of coronagraph as well as the available computing resources. However, the fundamental fact remains that regardless of how rapidly one calculates the Jacobian matrix, its size becomes problematic when the number of DM actuators, the number of samples within the control region, or both become large. This, in turn, translates to stricter processing requirements for the flight computer aboard a future space mission. For current state-of-the-art laboratory testbeds such as the high contrast imaging testbed (HCIT)[11] and the high contrast imager for complex aperture telescopes (HiCAT)[12] that each use a pair of DMs with a combined actuator count on the order of a few thousand, this has not yet proven to be an obstacle. However, LUVOIR has identified a baseline requirement of two DMs each with 128×128 actuators, or up to 32,768 actuators in total under the assumption that all actuators are active and available for wavefront correction.[2] As we show later in the paper, in this case SM and EFC consume up to 35 GB of memory while computing wavefront control solutions, dominated by storage of Jacobian matrices and other arrays derived from Jacobians, which is a highly nontrivial requirement for space-qualified hardware.

In this paper, we present a new approach in which we compute the gradient of the objective function of the control problem using a technique known as reverse-mode algorithmic differentiation (RMAD)[13] and then use standard gradient-based optimization techniques to find wavefront control solutions. The computation of the gradient requires effort comparable to that of a single evaluation of the objective function,[13–16] which itself is comprised of one evaluation of the coronagraph model per controlled wavelength. Furthermore, gradient calculation only consists of inexpensive linear operations on gradient vectors; large matrices are not calculated or manipulated at any time. As a consequence, our approach is asymptotically more efficient in both memory consumption and CPU time as a function of the number of DM actuators and number of samples in the dark zone than methods based on a Jacobian matrix, while simultaneously eliminating the cost of calculating the Jacobian matrix beforehand. With a pair of 64×64 DMs and a small dark zone extending out to $12\lambda_0/D$, our proposed algorithm consumes approximately half as much memory as SM; with a pair of 128×128 DMs, our algorithm reduces memory consumption by more than 90%.

A closely related method is COFFEE,[17] which also utilizes gradient-based optimization with an analytically derived gradient, but which is based on a fully nonlinear model of the DM response, whereas our algorithm is founded on the same linearity approximation as SM and EFC. This approximation has the advantage that the proposed wavefront control objective function is globally convex, so a gradient-based optimization algorithm is guaranteed to converge to the globally optimal solution. Moreover, it enables the algorithm to utilize estimates of the electric field at the coronagraph detector, which are straightforward to obtain using techniques such as pairwise probing,[18] Kalman filtering,[19,20] or the self-coherent camera.[21] On the other hand, COFFEE has no such convergence guarantee and must use estimates of the amplitude and phase aberrations in the coronagraph entrance pupil (EP), which are more difficult to obtain.

This paper is structured as follows. In Sec. 2, we review the linearized control model for focal-plane wavefront control, the semianalytical coronagraph forward model, and the SM algorithm. In Sec. 3, we provide a basic overview of RMAD, which we use to differentiate the proposed objective function. In Sec. 4, we define the objective function for the proposed algorithm and derive its gradient. In Sec. 5, we show simulations of the proposed algorithm using an example coronagraph design and analyze its performance. Finally, in Sec. 6, we analyze the relative computational efficiency of the proposed algorithm compared to SM and offer perspectives on future approaches to system identification using the principles outlined in this paper.

1.1 *Notation*

In this paper, we primarily deal with discrete quantities such as DM actuator commands and arrays of samples of spatially varying fields and masks. These vector-valued quantities are denoted with boldface lettering. In certain contexts, such as when performing a Fourier transform operation, the input should be treated as a two-dimensional (2D) array rather than a one-dimensional vector; though we will not explicitly denote these two cases, the appropriate choice should be clear from context.

$\mathbf{A}^T$ and $\mathbf{A}^\dagger$ denote the ordinary transpose and Hermitian transpose of the matrix $\mathbf{A}$, respectively. $\Re\{\cdot\}$ and $\Im\{\cdot\}$ denote the real and imaginary parts of a complex variable. Finally, elementwise multiplication between a pair of vectors is denoted by the operator $\circ$; when this is omitted, ordinary matrix multiplication is assumed.

2 Review of Focal-Plane Coronagraphic Wavefront Control

In this section, we review the principles of coronagraph modeling and focal-plane wavefront control using a linearized model of the DM correction. We will use the material presented in Secs. 2.1 and 2.2 to derive an analytic gradient with respect to the DM correction later in Sec. 4.2, forming the basis for our proposed algorithm. In Secs. 2.3 and 2.4, we review the construction and implementation of conventional Jacobian-based wavefront control algorithms, focusing specifically on the SM algorithm.[8] Much of this section follows Groff et al.,[22] though the notation has been adapted to be consistent with the remainder of this paper.

2.1 *Linearized Deformable Mirror Model*

Figure 1 shows an unfolded optical layout of a Lyot-type coronagraph. For narrow-band light with wavelength λ, the electric field at the EP in the k'th control iteration can be written as

$$\mathbf{E}_{\mathrm{EP},k} = \mathbf{P} \circ \exp\{i\mathbf{g}\} \circ \exp\{i\boldsymbol{\phi}_{1,k}\}, \tag{1}$$

where $\mathbf{P}$ is the complex-valued EP function, $\mathbf{g}$ is the complex-valued aberration, and $\boldsymbol{\phi}_{1,k}$ is the phase imparted by the in-pupil deformable mirror (DM1). In practice, $\mathbf{P}$ contains both amplitude transmittance and phase errors estimated prior to commencing wavefront control; as a result, $\mathbf{g}$ primarily describes small residual aberrations.

We write the phase of the in-pupil DM as a combination of the phase in the previous iteration and an update in the current iteration, $\boldsymbol{\phi}_{1,k} = \boldsymbol{\phi}_{1,k-1} + \Delta\boldsymbol{\phi}_{1,k}$:

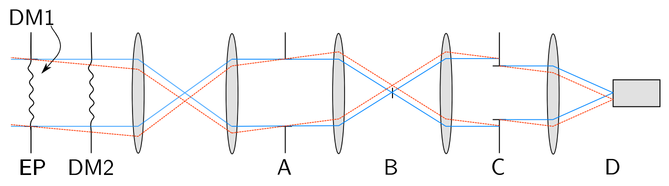


Fig. 1 Simplified, unfolded optical layout of a Lyot-type coronagraph with DMs for wavefront control. Planes EP, A, B, C, and D contain the coronagraph EP, pupil-plane mask, focal-plane mask, Lyot stop, and detector, respectively. Marginal rays from the on-axis host star are shown in blue, whereas rays from an off-axis planet are shown in red. Reproduced with modifications from Ref. 23, Fig. 6.

$$\mathbf{E}_{\mathrm{EP},k} = \mathbf{P} \circ \exp\{i\mathbf{g}\} \circ \exp\{i(\boldsymbol{\phi}_{1,k-1} + \Delta\boldsymbol{\phi}_{1,k})\}. \tag{2}$$

In the small-aberration regime, one can compute a first-order Taylor series expansion of both the aberrations and the DM update. Neglecting the cross-term between the two, we obtain

$$\mathbf{E}_{\mathrm{EP},k} \approx \mathbf{P} \circ \exp\{i\boldsymbol{\phi}_{1,k-1}\} \circ (1 + i\mathbf{g} + i\Delta\boldsymbol{\phi}_{1,k}). \tag{3}$$

The objective of the wavefront control problem is to determine a sequence of DM phase updates $\Delta\boldsymbol{\phi}$ that conjugate the unknown aberrations $\mathbf{g}$. To make the problem finite-dimensional, we write the phase update in terms of the actuator command vector $\mathbf{a}_{1,k}$ and influence functions for DM1, $\mathbf{F}_1$:

$$\Delta\boldsymbol{\phi}_{1,k} = \frac{4\pi}{\lambda}\sum_{j=1}^{N_{\mathrm{act}}} a_{1,j,k}\mathbf{f}_{1,j} = \frac{4\pi}{\lambda}\mathbf{F}_1\mathbf{a}_{1,k}, \tag{4}$$

where λ is the imaging wavelength, N_{act} is the total number of active DM actuators, and $\mathbf{f}_{1,j}$ is the influence function for the j'th actuator. We insert this expression into Eq. (3) and propagate from the coronagraph EP to the dark zone within the detector plane using the linear operator $\mathcal{C}\{\cdot;\lambda\}$ to obtain

$$\mathbf{E}_{\mathrm{DZ},k} \approx \mathcal{C}\{\mathbf{P} \circ \exp\{i\boldsymbol{\phi}_{1,k-1}\} \circ (1 + i\mathbf{g});\lambda\} + i\frac{4\pi}{\lambda}\sum_{j=1}^{N_{\mathrm{act}}} a_{1,j,k}\mathcal{C}\{\mathbf{P} \circ \exp\{i\boldsymbol{\phi}_{1,k-1}\} \circ \mathbf{f}_{1,j};\lambda\}. \tag{5}$$

We now make two definitions to simplify Eq. (5). First, we define the aberrated focal-plane electric field $\mathbf{E}_{\mathrm{ab},k}$ as

$$\mathbf{E}_{\mathrm{ab},k} \triangleq \mathcal{C}\{\mathbf{P} \circ \exp\{i\boldsymbol{\phi}_{1,k-1}\} \circ (1 + i\mathbf{g});\lambda\}. \tag{6}$$

In addition, letting N_{pix} denote the number of pixels inside the dark zone, we define the $N_{\mathrm{pix}} \times N_{\mathrm{act}}$ control Jacobian matrix for DM1, $\mathbf{G}_{1,k}$, as a matrix whose j'th column, $\mathbf{G}_{1,k}[j]$, corresponds to the j'th DM actuator and is given by

$$\mathbf{G}_{1,k}[j] \triangleq i\frac{4\pi}{\lambda}\mathcal{C}\{\mathbf{P} \circ \exp\{i\boldsymbol{\phi}_{1,k-1}\} \circ \mathbf{f}_{1,j};\lambda\}. \tag{7}$$

These two definitions enable us to write the electric field in the dark zone in the more manageable form

$$\mathbf{E}_{\mathrm{DZ},k} \approx \mathbf{E}_{\mathrm{ab},k} + \mathbf{G}_{1,k}\mathbf{a}_{1,k}. \tag{8}$$

As described in Sec. 1, the aberrated field $\mathbf{E}_{\mathrm{ab}}$ is estimated experimentally using techniques such as pairwise probing,[18] Kalman filtering,[19,20] or the self-coherent camera.[21]

2.1.1 *Calculating the Jacobian matrix*

Though it is possible to evaluate the columns of the Jacobian matrix analytically using Eq. (7), which requires N_{act} evaluations of the coronagraph model, in practice it is more straightforward to use a finite-difference approximation. In the least-expensive approximation, a forward finite difference, each column of $\mathbf{G}_{1,k}$ is computed as

$$\mathbf{G}_{1,k}[j] \approx \frac{\mathcal{C}\left\{\mathbf{P}\circ\exp\left\{i\left(\boldsymbol{\phi}_{1,k-1}+h\frac{4\pi}{\lambda}\mathbf{f}_{1,j}\right)\right\};\lambda\right\}-\mathcal{C}\{\mathbf{P}\circ\exp\{i(\boldsymbol{\phi}_{1,k-1})\};\lambda\}}{h}, \tag{9}$$

where h is a small real-valued step size. This requires only one additional evaluation of the model because the subtracted term is common to all columns. For large h, the approximation error is dominated by higher-order terms from the Taylor series expansion of the DM phase; for small h, floating-point error dominates. One can also use a more accurate central difference approximation with lower error from nonlinear terms, but at the cost of $2N_{\text{act}}$ model evaluations:

$$\mathbf{G}_{1,k}[j] \approx \frac{\mathcal{C}\left\{\mathbf{P}\circ\exp\left\{i\left(\boldsymbol{\phi}_{1,k-1}+h\frac{4\pi}{\lambda}\mathbf{f}_{1,j}\right)\right\};\lambda\right\}-\mathcal{C}\left\{\mathbf{P}\circ\exp\left\{i\left(\boldsymbol{\phi}_{1,k-1}-h\frac{4\pi}{\lambda}\mathbf{f}_{1,j}\right)\right\};\lambda\right\}}{2h}. \tag{10}$$

Because the performance of the wavefront control loop relies critically on the accuracy of the Jacobian matrix, one should carefully consider the approximation used to compute it.

Second, we note that since the Jacobian matrix, as defined by Eq. (7), is given in terms of the DM commands from the previous control iteration, the matrix should be recomputed each time the commands are updated. However, as we saw above, this requires a large number of coronagraph model evaluations since N_{act} is typically in the thousands. Therefore, one can instead linearize around the initial DM commands and reuse the same matrix for each control iteration, at the cost of slower control convergence.[8] Properly scheduling how often a Jacobian matrix should be recomputed to achieve optimal contrast and fast convergence is one aspect of the experimental validation process; for the purposes of this paper, we will assume the approach with the least computational overhead and only compute the Jacobian matrix once, using a forward finite difference as given in Eq. (9).

2.1.2 *Two deformable mirrors*

The situation as described up to this point becomes somewhat more complicated when a second deformable mirror (DM2) is introduced into an intermediate plane some distance along the optical axis from the EP, which enables multiwavelength control of both phase and amplitude aberrations over a symmetric dark zone.[8] The angular spectrum operator[24] $\mathcal{A}\{\mathbf{E};\Delta z,\lambda\}$ describes paraxial propagation of a monochromatic electric field $\mathbf{E}$ with wavelength λ over a distance Δz:

$$\mathcal{A}\{\mathbf{E};\Delta z,\lambda\} = \text{IFFT}\{\mathbf{H}(\Delta z,\lambda)\circ\text{FFT}\{\mathbf{E}\}\}, \tag{11}$$

where FFT$\{\cdot\}$ and IFFT$\{\cdot\}$ are the forward and inverse fast Fourier transform (FFT) operators, respectively. $\mathbf{H}$ is the angular spectrum transfer function in the Fourier (spatial frequency) domain:

$$\mathbf{H}(\Delta z,\lambda) = \exp\left\{i\frac{2\pi\Delta z}{\lambda}\sqrt{1-(\lambda\mathbf{f}_x)^2-(\lambda\mathbf{f}_y)^2}\right\}. \tag{12}$$

For the sake of notational brevity, we define the operator $\mathcal{P}$ as

$$\mathcal{P}\{\mathbf{E},\boldsymbol{\psi};\Delta z,\lambda\} \triangleq \mathcal{A}\{\boldsymbol{\psi}\circ\mathcal{A}\{\mathbf{E};\Delta z,\lambda\};-\Delta z,\lambda\}, \tag{13}$$

which represents propagation of $\mathbf{E}$ over distance Δz, elementwise multiplication with the complex-valued transmittance function $\boldsymbol{\psi}$, and subsequent propagation by the same distance in the reverse direction. The contribution from DM2 is then modeled by applying $\mathcal{P}$ to the electric field in Eq. (3) yielding the field in the plane immediately before the apodizing mask in plane A of Fig. 1:

$$\mathbf{E}'_{A,k} = \mathcal{P}\{\mathbf{E}_{\mathrm{EP},k}, \exp\{i\boldsymbol{\phi}_{2,k}\}; \Delta z_{\mathrm{DM}}, \lambda\}. \tag{14}$$

We write the phase from DM2 in the recursive form $\boldsymbol{\phi}_{2,k} = \boldsymbol{\phi}_{2,k-1} + \Delta\boldsymbol{\phi}_{2,k}$ as we did for DM1 earlier. Expanding $\mathbf{E}_{\mathrm{EP},k}$ using Eq. (3) and using the first-order approximation $\exp\{i\Delta\boldsymbol{\phi}_{2,k}\} \approx 1 + i\Delta\boldsymbol{\phi}_{2,k}$, we rewrite Eq. (14) as

$$\begin{aligned}\mathbf{E}'_{A,k} = {} & \mathcal{P}\{\mathbf{P} \circ \exp\{i\boldsymbol{\phi}_{1,k-1}\} \circ (1 + i\mathbf{g}), \exp\{i\boldsymbol{\phi}_{2,k-1}\}; \Delta z_{\mathrm{DM}}, \lambda\} \\ & + i\mathcal{P}\{\mathbf{P} \circ \exp\{i\boldsymbol{\phi}_{1,k-1}\} \circ \Delta\boldsymbol{\phi}_{1,k}, \exp\{i\boldsymbol{\phi}_{2,k-1}\}; \Delta z_{\mathrm{DM}}, \lambda\} \\ & + i\mathcal{P}\{\mathbf{P} \circ \exp\{i\boldsymbol{\phi}_{1,k-1}\}, \exp\{i\boldsymbol{\phi}_{2,k-1}\} \circ \Delta\boldsymbol{\phi}_{2,k}; \Delta z_{\mathrm{DM}}, \lambda\} \\ & - \mathcal{P}\{\mathbf{P} \circ \exp\{i\boldsymbol{\phi}_{1,k-1}\} \circ \mathbf{g}, \exp\{i\boldsymbol{\phi}_{2,k-1}\} \circ \Delta\boldsymbol{\phi}_{2,k}; \Delta z_{\mathrm{DM}}, \lambda\} \\ & - \mathcal{P}\{\mathbf{P} \circ \exp\{i\boldsymbol{\phi}_{1,k-1}\} \circ \Delta\boldsymbol{\phi}_{1,k}, \exp\{i\boldsymbol{\phi}_{2,k-1}\} \circ \Delta\boldsymbol{\phi}_{2,k}; \Delta z_{\mathrm{DM}}, \lambda\}.\end{aligned} \tag{15}$$

The final terms contain higher-order crossterms between the DM2 phase update and, respectively, the small aberrations and DM1 phase update, which we discard to obtain

$$\begin{aligned}\mathbf{E}'_{A,k} \approx {} & \mathcal{P}\{\mathbf{P} \circ \exp\{i\boldsymbol{\phi}_{1,k-1}\} \circ (1 + i\mathbf{g}), \exp\{i\boldsymbol{\phi}_{2,k-1}\}; \Delta z_{\mathrm{DM}}, \lambda\} \\ & + i\mathcal{P}\{\mathbf{P} \circ \exp\{i\boldsymbol{\phi}_{1,k-1}\} \circ \Delta\boldsymbol{\phi}_{1,k}, \exp\{i\boldsymbol{\phi}_{2,k-1}\}; \Delta z_{\mathrm{DM}}, \lambda\} \\ & + i\mathcal{P}\{\mathbf{P} \circ \exp\{i\boldsymbol{\phi}_{1,k-1}\}, \exp\{i\boldsymbol{\phi}_{2,k-1}\} \circ \Delta\boldsymbol{\phi}_{2,k}; \Delta z_{\mathrm{DM}}, \lambda\},\end{aligned} \tag{16a}$$

$$= \mathbf{E}'_{A,k,\mathrm{ab}} + \mathbf{E}'_{A,k,\mathrm{DM1}} + \mathbf{E}'_{A,k,\mathrm{DM2}}. \tag{16b}$$

Similarly to the single-DM case, the field in the EP consists of a linear combination of terms pertaining to the aberrations, an update term from DM1, and an update term from DM2. Also as before, we represent the DM2 control update as a linear combination of influence functions $\mathbf{F}_2$ weighted by the command vector $\mathbf{a}_2$:

$$\Delta\boldsymbol{\phi}_{2,k} = \frac{4\pi}{\lambda}\mathbf{F}_2\mathbf{a}_{2,k}. \tag{17}$$

Finally, the field at the detector is obtained by propagating the field in Eq. (16a) through the coronagraph:

$$\mathbf{E}'_{\mathrm{DZ},k} = \mathcal{C}\{\mathbf{E}'_{A,k}; \lambda\} = \mathbf{E}'_{\mathrm{ab},k} + \mathbf{G}'_{1,k}\mathbf{a}_{1,k} + \mathbf{G}'_{2,k}\mathbf{a}_{2,k}, \tag{18}$$

where the prime (′) is included to distinguish from the single-DM result in Eq. (8), and where we have defined the Jacobian matrices for each DM by propagating each individual influence function to the coronagraph detector plane. This can be written more succinctly as

$$\mathbf{E}'_{\mathrm{DZ},k} \approx \mathbf{E}'_{\mathrm{ab},k} + \mathbf{G}_k\mathbf{a}_k, \tag{19}$$

where the augmented Jacobian matrix and command vector are formed by concatenation:

$$\mathbf{G}_k = [\mathbf{G}'_{1,k} \quad \mathbf{G}'_{2,k}] \quad \mathbf{a}_k = \begin{bmatrix}\mathbf{a}_{1,k} \\ \mathbf{a}_{2,k}\end{bmatrix}. \tag{20}$$

In the remainder of this paper, we will focus on wavefront control using two DMs, and therefore will drop the primes (′) when referring to the quantities in Eq. (19) hereafter.

2.2 *Semianalytical Coronagraph Model*

In the previous section, we described an approximate linear decomposition of the focal-plane electric field in terms of a generic linear operator $\mathcal{C}\{\cdot; \lambda\}$ that represents propagation through the coronagraph. Though $\mathcal{C}\{\cdot;\lambda\}$ can be modeled in numerous ways depending on the specific coronagraph type of interest, in this paper, for simplicity, we restrict our attention to Lyot-type coronagraphs with small focal spot occulters, such as the classical Lyot coronagraph, apodized pupil Lyot coronagraph (APLC),[25] and hybrid Lyot coronagraph.[26] In this case, $\mathcal{C}\{\cdot;\lambda\}$ can be described by the semianalytical model developed by Soummer et al.,[27] which we review here.

Recall from Eq. (21) that with small aberrations and a pair of DMs, the electric field immediately before the apodizing mask can be written as a linear combination of terms that include the aberrations and the command updates for the in-pupil and out-of-pupil deformable mirror DM1 and DM2:

$$\mathbf{E}'_{A,k} = \mathbf{E}'_{A,k,\mathrm{ab}} + \mathbf{E}'_{A,k,\mathrm{DM1}} + \mathbf{E}'_{A,k,\mathrm{DM2}}. \tag{21}$$

The field immediately after the apodizing mask $\mathbf{A}$ is

$$\mathbf{E}_{A,k} = \mathbf{A} \circ \mathbf{E}'_{A,k}. \tag{22}$$

The field inside the opaque part of the occulter is given by

$$\mathbf{E}_{B,k} = \frac{\Delta x \Delta y}{\lambda} \mathbf{M} \circ \mathrm{MFT}\{\mathbf{E}_{A,k}; \mathbf{x}, \mathbf{y}, \boldsymbol{\theta}_x, \boldsymbol{\theta}_y\}, \tag{23}$$

where $\mathbf{M}$ is the complex transmittance of the focal-plane mask. Note that in this treatment, following Soummer et al.,[27] $\mathbf{M}$ is a small transmitting aperture so that $1 - \mathbf{M}$ is an opaque focal spot. MFT denotes the matrix Fourier transform:[27,28]

$$\mathrm{MFT}\{\mathbf{E}_{A,k}; \mathbf{x}, \mathbf{y}, \boldsymbol{\theta}_x, \boldsymbol{\theta}_y\} \triangleq \exp\{-i2\pi \boldsymbol{\theta}_x \mathbf{x}^T\} \mathbf{E}_{A,k} \exp\{-i2\pi \mathbf{y} \boldsymbol{\theta}_y^T\}, \tag{24}$$

where $(\mathbf{x}, \mathbf{y})$ and $(\boldsymbol{\theta}_x, \boldsymbol{\theta}_y)$ are the discrete coordinates for the pupil plane and occulter plane; also note that we treat $\mathbf{E}_{A,k}$ as a 2D array rather than as a vector in this expression. In paraxial Fourier optics, $(\boldsymbol{\theta}_x, \boldsymbol{\theta}_y)$ are proportional to spatial frequency via wavelength:

$$(\boldsymbol{\theta}_x, \boldsymbol{\theta}_y) = (\lambda \mathbf{f}_x, \lambda \mathbf{f}_y). \tag{25}$$

The occulter-plane coordinates are chosen so that the field $\mathbf{E}_{B,k}$ is computed only within the extent of the occulter, which makes the MFT very efficient relative to the FFT.[27] The field immediately after the Lyot stop $\mathbf{L}$ is calculated by applying Babinet's principle:

$$\mathbf{E}_{C,k} = \mathbf{L} \circ \left(\mathcal{R}\{\mathbf{E}_{A,k}\} - \frac{\Delta\theta_x \Delta\theta_y}{\lambda} \mathrm{MFT}\{\mathbf{E}_{B,k}; \boldsymbol{\theta}_x, \boldsymbol{\theta}_y, \mathbf{x}, \mathbf{y}\} \right). \tag{26}$$

Here, $\mathcal{R}\{\cdot\}$ denotes coordinate reversal (flipping) along both coordinate axes. The field contributed by the DMs at the detector is found using a final Fourier transformation:

$$\mathbf{E}_{\mathrm{DM},k} = \frac{\Delta x \Delta y}{\lambda} \mathrm{MFT}\{\mathbf{E}_{C,k}; \mathbf{x}, \mathbf{y}, \boldsymbol{\theta}_x, \boldsymbol{\theta}_y\}. \tag{27}$$

The total dark-zone electric field including the experimentally estimated aberrated field $\hat{\mathbf{E}}_{\mathrm{ab},k}$ is then

$$\mathbf{E}_{\mathrm{DZ},k} = \hat{\mathbf{E}}_{\mathrm{ab},k} + \mathbf{E}_{\mathrm{DM},k}[\mathbf{1}_{\mathrm{DZ}}], \tag{28}$$

where $\mathbf{1}_{\mathrm{DZ}}$ is the indicator function for the coronagraphic dark zone that has a value of unity for pixels inside the dark zone and zero elsewhere, and $\mathbf{E}_{\mathrm{DM},k}[\mathbf{1}_{\mathrm{DZ}}]$ is interpreted as the set of elements of $\mathbf{E}_{\mathrm{DM},k}$ that correspond to nonzero values of $\mathbf{1}_{\mathrm{DZ}}$, organized into a vector with length N_{pix}. Because this is the behavior of array indexing operations in several programming languages including Python, we will refer to this operation henceforth as "array indexing."

Finally, the integrated dark-zone intensity is

$$I_{\mathrm{DZ},k} = \mathbf{E}_{\mathrm{DZ},k}^{\dagger}\mathbf{E}_{\mathrm{DZ},k}. \tag{29}$$

2.3 *Stroke Minimization*

The SM algorithm[8] finds the least-norm actuator update that causes the integrated dark-zone intensity $I_{\mathrm{DZ},k}$ to reach a desired target $I_{T,k}$:

$$\arg\min_{\mathbf{a}_k} \quad \mathbf{a}_k^T\mathbf{a}_k \quad \text{subject to } I_{\mathrm{DZ},k} < I_{T,k}. \tag{30}$$

This is solved via the method of Lagrange multipliers by treating the constraint as an equality constraint and minimizing the scalar Lagrangian function

$$\mathcal{L}_k \triangleq \mathbf{a}_k^T\mathbf{a}_k + \mu(I_{\mathrm{DZ},k} - I_{T,k}). \tag{31}$$

Given an estimate of the aberrated dark-zone electric field $\mathbf{E}_{\mathrm{ab},k}$, the total intensity is written as

$$I_{\mathrm{DZ},k} = \mathbf{E}_{\mathrm{DZ},k}^{\dagger}\mathbf{E}_{\mathrm{DZ},k} \tag{32a}$$

$$= \mathbf{a}_k^T\Re\{\mathbf{G}_k^{\dagger}\mathbf{G}_k\}\mathbf{a}_k + 2\Re\{\mathbf{E}_{\mathrm{ab},k}^{\dagger}\mathbf{G}_k\}\mathbf{a}_k + \mathbf{E}_{\mathrm{ab},k}^{\dagger}\mathbf{E}_{\mathrm{ab},k} \tag{32b}$$

$$= \mathbf{a}_k^T\mathbf{M}_k\mathbf{a}_k + \mathbf{b}_k^T\mathbf{a}_k + d_k, \tag{32c}$$

where we have defined the auxiliary quantities

$$\mathbf{M}_k = \Re\{\mathbf{G}_k^{\dagger}\mathbf{G}_k\} \quad \mathbf{b}_k = 2\Re\{\mathbf{G}_k^{\dagger}\mathbf{E}_{\mathrm{ab},k}\} \quad d_k = \mathbf{E}_{\mathrm{ab},k}^{\dagger}\mathbf{E}_{\mathrm{ab},k}, \tag{33}$$

where the real part in the definition of $\mathbf{M}_k$ follows from the fact that $\mathbf{a}_k$ is purely real and $\mathbf{G}_k^{\dagger}\mathbf{G}_k$ is Hermitian. Note that $\mathbf{M}_k$ in this context should not be confused with the focal-plane mask transmittance in Eq. (23). Inserting Eq. (32c) into Eq. (31), it becomes apparent that the Lagrangian is a quadratic function of $\mathbf{a}_k$:

$$\mathcal{L}_k = \mathbf{a}_k^T\mathbf{a}_k + \mu(\mathbf{a}_k^T\mathbf{M}_k\mathbf{a}_k + \mathbf{b}_k^T\mathbf{a}_k + d_k - I_{T,k}). \tag{34}$$

To obtain the optimal set of actuator commands as a function of the Lagrange multiplier μ, we compute the gradient of the Lagrangian with respect to the actuator command vector:

$$\frac{\partial\mathcal{L}_k}{\partial\mathbf{a}_k^T} = 2\mathbf{a}_k + \mu[(\mathbf{M}_k + \mathbf{M}_k^T)\mathbf{a}_k + \mathbf{b}_k] \tag{35a}$$

$$= 2\mathbf{a}_k + \mu(2\mathbf{M}_k\mathbf{a}_k + \mathbf{b}_k). \tag{35b}$$

The critical point $\mathbf{a}_k^*(\mu)$ occurs where the gradient vanishes:

$$0 = 2\mathbf{a}_k^* + \mu(2\mathbf{M}_k\mathbf{a}_k^* + \mathbf{b}_k) \tag{36a}$$

$$\mathbf{a}_k^*(\mu) = -\frac{1}{2}\left(\frac{1}{\mu}\mathbf{I} + \mathbf{M}_k\right)^{-1}\mathbf{b}_k. \tag{36b}$$

The final step is to perform a line search on μ, increasing μ and recomputing $\mathbf{a}_k^*$ until the original constraint $I_{\mathrm{DZ},k} < I_{T,k}$ in Eq. (30) is satisfied. In practice, evaluating Eq. (36b) directly is not recommended because it necessitates inversion of a large matrix $\mathbf{M}_k$ with $N_{\mathrm{act}} \times N_{\mathrm{act}}$ elements, which requires $N_{\mathrm{act}}^3 + 2N_{\mathrm{act}}^2$ operations and has poor numerical stability properties.[29] A better approach is to solve the linear system

$$2\left(\frac{1}{\mu}\mathbf{I}+\mathbf{M}_k\right)\mathbf{a}_k=-\mathbf{b}_k, \tag{37}$$

using matrix decomposition methods; for this particular problem, we can exploit the fact that the matrix $2(\mathbf{I}/\mu+\mathbf{M}_k)$ is symmetric and positive definite, enabling it to be solved using the Cholesky decomposition using only $N_{\text{act}}^3/3+2N_{\text{act}}^2$ operations and with greatly improved numerical stability.[30] We will operate under the assumption of this strategy when we analyze the relationship between our proposed algorithm and SM later in Sec. 6.

Notice that the gradient of the Lagrangian in Eq. (35b) is given symbolically in terms of a matrix $\mathbf{M}_k$, which is itself the square of the Jacobian matrix $\mathbf{G}_k$ with dimension $N_{\text{pix}}\times N_{\text{act}}$. In other words, we first compute a quantity with $N_{\text{pix}}\times N_{\text{act}}$ degrees of freedom in order to evaluate a gradient vector with only N_{act} degrees of freedom. The reason for this extra effort is that we have written the gradient in terms of the derivative of a vector-valued variable, $\mathbf{E}_{\text{DM},k}[\mathbf{1}_{\text{DZ}}]$ in Eq. (28), with respect to another vector:

$$\mathbf{G}_k=\frac{\partial\mathbf{E}_{\text{DM},k}[\mathbf{1}_{\text{DZ}}]}{\partial\mathbf{a}_k}. \tag{38}$$

As we will show in Secs. 3 and 4, by manipulating only vector-valued derivatives of a scalar quantity instead, one can arrive at the desired gradient using far fewer model evaluations and eliminating the need to calculate $\mathbf{G}_k$ altogether.

2.4 *Broadband Stroke Minimization*

With a pair of DMs (one in-pupil and one out-of-pupil), the SM inverse problem can be augmented to enable multiwavelength wavefront control by adding intensity constraints for each control wavelength of interest.[8,22] It is impossible for all wavelength constraints to be simultaneously satisfied due to the inherent chromaticity of both the coronagraph and the DM phase update, so we introduce scalar, nonnegative weighting factors δ_ℓ to trade off between correction at the center wavelength λ_0 and correction at a given λ_ℓ. This is expressed by the modified Lagrangian function

$$\mathcal{L}_k'=\mathbf{a}_k^T\mathbf{a}_k+\mu\sum_{\ell=1}^{L}\delta_\ell(\mathbf{a}_k^T\mathbf{M}_{k,\ell}\mathbf{a}_k+\mathbf{b}_{k,\ell}^T\mathbf{a}_k+d_{k,\ell}-I_{T,k,\ell}). \tag{39}$$

The extra subscripts ℓ reflect the fact that the coronagraph response and aberrated electric field are functions of wavelength. Each term inside the summation is linear in all variables except for $\mathbf{a}_k$, so we can rewrite in a form identical to Eq. (34):

$$\mathcal{L}_k'=\mathbf{a}_k^T\mathbf{a}_k+\mu(\mathbf{a}_k^T\mathbf{M}_k'\mathbf{a}_k+\mathbf{b}_k'^T\mathbf{a}_k+d_k'-I_{T,k}'), \tag{40}$$

where

$$\mathbf{M}_k'\triangleq\sum_{\ell=1}^{L}\delta_\ell\mathbf{M}_{k,\ell}\quad\mathbf{b}_k'\triangleq\sum_{\ell=1}^{L}\delta_\ell\mathbf{b}_{k,\ell}\quad d_k'\triangleq\sum_{\ell=1}^{L}\delta_\ell d_{k,\ell}\quad I_{T,k}'\triangleq\sum_{\ell=1}^{L}\delta_\ell I_{T,k,\ell}. \tag{41}$$

The broadband solution $\mathbf{a}_k^*$ is then obtained using the same procedure as for the monochromatic case described in Sec. 2.3.

3 Algorithmic Differentiation

Algorithmic differentiation, also known as automatic differentiation, is a family of techniques for analytically evaluating the derivatives of numerical algorithms. By this we mean the following: given a finite sequence of differentiable operations implemented as a computer algorithm, which we term the forward model, algorithmic differentiation numerically evaluates the partial derivatives of the output with respect to the algorithm's inputs and intermediate variables. Unlike finite-difference methods, the derivative values yielded by algorithmic differentiation are accurate to machine precision.[16] Also, unlike symbolic techniques such as those used by computer algebra software tools such as Mathematica,[31] at no point is a closed-form expression for the derivative written down; instead, derivative values are propagated either through the forward model or backward through a related model that is constructed from the forward model. The former case is known as forward-mode algorithmic differentiation, whereas the latter is known as RMAD.[13,14] This enables simple and efficient handling of algorithms containing loops and conditional branching,[16] and as we will see shortly, this also enables highly complicated algorithms to be constructed and differentiated in a modular way that improves software reusability and testability.

The theory and practice of algorithmic differentiation techniques is a rich and complex area developed over the last several decades, and a full treatment is out of scope for this paper. In this section, we will instead provide a brief overview of the principles and consequences of algorithmic differentiation that are most relevant to the problem at hand, wavefront control of a time-varying coronagraph. In particular, we will focus on RMAD, which is especially suitable for computing derivatives of algorithms whose output is a scalar variable. For a rigorous introduction to RMAD and its application to optical phase retrieval, we refer the reader to Ref. 15. For a complete treatment of algorithmic differentiation, we refer the reader to Ref. 13.

With RMAD, the chain rule of calculus is recursively applied to generate an adjoint model that propagates partial derivative values in reverse order with respect to the forward model, beginning from the output. Each operation in the adjoint model corresponds to exactly one operation in the forward model, and the intermediate variables of the adjoint model, known as adjoint variables,[13] represent the derivative of the forward model with respect to the corresponding intermediate variable in the forward model, which we term forward variables. To compute a derivative using RMAD, one evaluates the forward model given a specific value for the inputs (termed the forward sweep[13]), passes the values of the forward variables to the adjoint model as parameters, and finally evaluates the adjoint model to calculate the value of each adjoint variable (termed the reverse sweep[13]). Figure 2 shows these concepts for a simple forward model.

Differentiation using a finite-difference approximation requires at least one forward sweep for each input variable, regardless of the number of output variables; each scalar element of a vector is treated as a separate variable. Conversely, differentiation using RMAD requires exactly one forward sweep and one reverse sweep per output variable, where, as before, each scalar element of a vector is treated separately, regardless of the number of inputs. Consequently, when the forward model consists of many inputs but only a single output, RMAD involves many times fewer model evaluations than finite-difference methods; this concept, known as the cheap gradient principle,[15] makes RMAD extremely effective for solving large-scale nonlinear optimization problems such as coronagraphic wavefront control.

Each operation in the adjoint model is related to its corresponding operation in the forward model as follows. If two forward variables are related by $\mathbf{y} = \mathbf{f}(\mathbf{x})$, then[15]

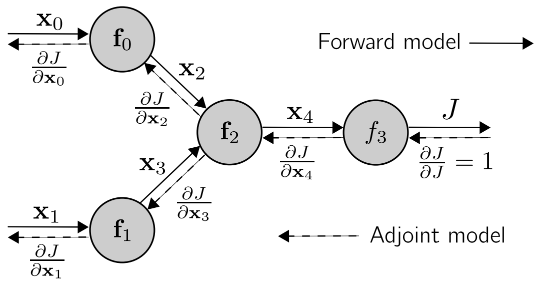

$\mathbf{x}_0$
$\mathbf{f}_0$
$\frac{\partial J}{\partial \mathbf{x}_0}$
Forward model
$\mathbf{x}_2$
$\frac{\partial J}{\partial \mathbf{x}_2}$
$\mathbf{f}_2$
$\mathbf{x}_4$
f_3
J
$\frac{\partial J}{\partial \mathbf{x}_4}$
$\frac{\partial J}{\partial J} = 1$
$\mathbf{x}_3$
$\frac{\partial J}{\partial \mathbf{x}_3}$
$\mathbf{x}_1$
$\mathbf{f}_1$
$\frac{\partial J}{\partial \mathbf{x}_1}$
Adjoint model

Fig. 2 Simple example demonstrating the basic principle of RMAD. The forward model, comprised of the differentiable functions $\mathbf{f}_n$, is evaluated left-to-right and computes the values of the forward variables $\mathbf{x}_n$ as well as a scalar output J. The adjoint model is then evaluated right-to-left and computes the derivatives of J with respect to each $\mathbf{x}_n$. A single evaluation of the forward model is referred to as a forward sweep; similarly, a single evaluation of the adjoint model is termed a reverse sweep. The operations of the adjoint model are related to $\mathbf{f}_n$ and $\mathbf{x}_n$ by Eq. (43).

$$\frac{\partial J}{\partial \mathbf{x}} = \frac{\partial J}{\partial \mathbf{y}} \frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}}, \tag{42}$$

where J is the scalar output of the forward model and $\partial \mathbf{f}(\mathbf{x})/\partial \mathbf{x}$ is interpreted as the partial derivative of $\mathbf{f}$ with respect to $\mathbf{x}$, evaluated around the particular value of $\mathbf{x}$ computed during the forward sweep. To simplify the notation, we define the adjoint variables $\bar{\mathbf{a}}$ such that for any forward variable $\mathbf{a}$, $\bar{\mathbf{a}}^\dagger \triangleq \partial J/\partial \mathbf{a}$, where $\dagger$ denotes the Hermitian transpose, so that

$$\bar{\mathbf{x}} = \left(\frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}}\right)^\dagger \bar{\mathbf{y}}. \tag{43}$$

By convention, if $\mathbf{a}$ is a column vector, $\partial J/\partial \mathbf{a}$ is a row vector; we include the Hermitian transpose in the definition of $\bar{\mathbf{a}}$ so that $\bar{\mathbf{a}}$ is also a column vector. Because $\mathbf{x}$ and $\mathbf{y}$ are vectors and J is a scalar, the adjoint variables $\bar{\mathbf{x}}$ and $\bar{\mathbf{y}}$ are gradient vectors; therefore, we refer to Eq. (43) as the gradient propagation rule.

The adjoint variables are linked by a linear transformation given by $(\partial \mathbf{f}(\mathbf{x})/\partial \mathbf{x})^\dagger$, which is the adjoint of the Jacobian matrix of the operation $\mathbf{f}$. However, to evaluate Eq. (43) does not necessitate that the Jacobian matrix be explicitly formed in general. Rather, for nearly all operations common for building models of optical systems, the gradient propagation rule has a simple closed-form expression. Table 1 includes several such examples; for a more exhaustive collection, see Ref. 15. For a simple example with a concrete forward model, see Appendix A.

The RMAD adjoint model may be constructed manually by utilizing Eq. (43), or by automated transformation of the computation graph associated with the forward model, such as the one shown in Fig. 2. The fully automatic case, which is implemented by frameworks such as TensorFlow[32] or Jax,[33] can be highly useful because it enables the programmer to specify only the forward model and then obtain partial derivatives for arbitrary model variables with no additional effort. Indeed, the ability to automatically and efficiently obtain derivatives of complicated models is one of the principal advantages of algorithmic differentiation over traditional techniques.

On the other hand, building the adjoint model manually, although more laborious, can enable the programmer to leverage computational optimizations that are difficult for, or inaccessible to, automated frameworks, such as retaining the minimal set of forward variables required for the reverse sweep. In the application considered here, computational demand is an important consideration; therefore, in Sec. 4.2, we explicitly construct the adjoint model for the proposed control problem.

Table 1 Gradient propagation rules for some operations commonly found in models of optical systems. Here, $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{a}$ are complex-valued vectors, $\mathbf{A}$ is a complex-valued matrix, and $\mathbf{1}_\mathcal{S}$ is an indicator function for the set $\mathcal{S}$ with a value of unity for elements in $\mathcal{S}$ and zero otherwise. For the FFT operation, we assume that $\mathbf{x}$ is reshaped into a 2D array. The equation number in Ref. 15 corresponding to each operation is shown on the right for convenience.

Forward model	Adjoint model	Ref. equation number[15]
$\mathbf{y} = \mathbf{x} + \mathbf{a}$	$\bar{\mathbf{x}} = \bar{\mathbf{y}}$	(47)
$\mathbf{y} = \mathbf{a} \circ \mathbf{x}$	$\bar{\mathbf{x}} = \mathbf{a}^* \circ \bar{\mathbf{y}}$	(48)
$\mathbf{y} = \mathbf{A}\mathbf{x}$	$\bar{\mathbf{x}} = \mathbf{A}^\dagger \bar{\mathbf{y}}$	(50)
$\mathbf{y} = a\mathbf{x}$	$\bar{\mathbf{x}} = a^* \bar{\mathbf{y}}$	(51)
$\mathbf{y} = \lvert\mathbf{x}\rvert^2$	$\bar{\mathbf{x}} = 2\mathbf{x} \circ \Re\{\bar{\mathbf{y}}\}$	(53)
$\mathbf{y} = \exp\{i\mathbf{x}\}$	$\bar{\mathbf{x}} = \Im\{\bar{\mathbf{y}} \circ \mathbf{y}^*\}$	(57)
$\mathbf{y} = \mathbf{x}[\mathbf{1}_\mathcal{S}]$	$\bar{\mathbf{x}}[\mathbf{1}_\mathcal{S}] = \bar{\mathbf{y}}$	(59)
$\mathbf{y} = \mathrm{FFT}\{\mathbf{x}\}$	$\bar{\mathbf{x}} = \mathrm{IFFT}\{\bar{\mathbf{y}}\}$	(95)

In this case, an additional advantage of algorithmic differentiation is that it can be applied recursively. Referring again to Fig. 2, the only requirements on the functions $\mathbf{f}_n$ are that they are differentiable and have known behavior (i.e., they cannot be black boxes); each $\mathbf{f}_n$ may itself consist of a complicated forward model. Evaluating the total forward model then involves evaluating the forward model for each $\mathbf{f}_n$ sequentially and passing input and output variables between models as necessary. Likewise, evaluating the total adjoint model simply involves evaluating the adjoint model for each $\mathbf{f}_n$ in the same manner.

This concept proves to be very powerful, because it enables one to build up very complicated differentiable models starting from elementary operations by forming self-contained building blocks that can be combined to form larger pieces, which can be themselves used to build more complicated models, and so on. Since each piece can be derived, implemented, and tested in isolation from the others, this makes model development and verification straightforward.[15] Appendix B contains several such examples that we encountered in Sec. 2, including the angular spectrum propagation operator and the matrix Fourier transform.

4 Proposed Algorithm

In this section, we will describe the application of gradient-based optimization with algorithmic differentiation to focal-plane wavefront control for coronagraphy. This section is structured as follows: in Sec. 4.1, we present a high-level overview of our proposed approach and describe its objective function. In Sec. 4.2, we derive the adjoint model of the objective function for RMAD. Finally, in Sec. 4.3, we discuss how to use the objective function and adjoint models to compute control solutions using gradient-based optimization.

4.1 *Fundamental Principles*

The basic premise of the proposed algorithm is to find wavefront control solutions by minimizing an appropriate scalar objective function J_k in each control iteration using gradient-based optimization:

$$\mathbf{a}_k^* = \arg \min_{\mathbf{a}_k} J_k(\mathbf{a}_k, \hat{\mathbf{E}}_{\mathrm{ab,k}}, I_{T,k}). \tag{44}$$

As in Sec. 2.3, $\mathbf{a}_k$, $\hat{\mathbf{E}}_{\mathrm{ab},k}$, and $I_{T,k}$ denote the actuator commands, the estimate of the aberrated electric field in the coronagraph dark zone, and the target intensity integrated over the dark zone, respectively. In this context, the asterisk denotes the optimality of the solution, not complex conjugation. The critical points of the objective function occur where the gradient with respect to $\mathbf{a}_k$ vanishes:

$$\frac{\partial J_k}{\partial \mathbf{a}_k^T} = \bar{\mathbf{a}}_k = 0. \tag{45}$$

Our approach is to use RMAD to construct an adjoint model for J_k that computes this gradient. Then, J_k and its adjoint model are provided to a gradient-based optimization algorithm that iteratively minimizes J_k to calculate the control solution $\mathbf{a}_k^*$. For now, it suffices to assume that the optimization algorithm is a black box that accepts an objective function and a function that computes its gradient and returns a solution. We will return to this topic in more detail in Sec. 4.3.

As described in Sec. 2.3, to solve the Lagrange multipliers problem in Eq. (31), the Lagrangian $\mathcal{L}_k$ is minimized with respect to $\mathbf{a}_k$ for a range of different values of the Lagrange multiplier μ, and of these, the least-norm $\mathbf{a}_k(\mu)$ that achieves the target integrated intensity is selected as the solution. The Lagrangian is a quadratic function of $\mathbf{a}_k$ as we showed in Eq. (34), and therefore each minimization subproblem is globally convex. Thus, it would be possible to directly replace the solution of each subproblem obtained by solving Eq. (37) with one obtained with gradient-based optimization, but we would be forced to repeat the optimization for many different values of μ. We would like to avoid this while still maintaining the desirable property of global convexity. One way to do so is to replace the linear intensity penalty in $\mathcal{L}_k$ with a quadratic penalty, forming a new objective function J_k with the same solutions:[14]

$$J_k = \mathbf{a}_k^T \mathbf{a}_k + \eta (I_{\mathrm{DZ},k} - I_{T,k})^2. \tag{46}$$

In this formulation, the scalar η is no longer interpreted as a Lagrange multiplier whose value is obtained by optimization. Instead, η is a penalty parameter that encodes the relative importance of minimizing actuator stroke and matching the target intensity in each control iteration, and whose value is chosen *a priori* as a parameter of the optimization problem. In practical terms, the advantage of framing the control problem in this way is that a solution is obtained by solving only a single optimization problem per control iteration. The downside is that the optimization problem becomes poorly conditioned as η becomes large,[14] which slows convergence to the solution in a given control iteration. We found that a good choice is $\eta = \eta_{00}/I_{T,k}^2$, where η_{00} is a constant, so that the penalty term in Eq. (46) has the form

$$\eta (I_{\mathrm{DZ},k} - I_{T,k})^2 = \eta_{00} \left(\frac{I_{\mathrm{DZ},k} - I_{T,k}}{I_{T,k}} \right)^2, \tag{47}$$

and is interpreted as the square of the fractional deviation of the modeled intensity $I_{\mathrm{DZ},k}$ from the target intensity $I_{T,k}$, scaled by η_{00}. This helps to ensure that the intensity penalty is adequately enforced as the system converges to high contrast and the target intensity becomes extremely small. For simplicity one can choose $\eta_{00} = 1$, but we explore the effect of different values later in Sec. 5.

The monochromatic objective function in Eq. (46) is extended to the multiwavelength case by introducing separate penalty terms for each wavelength of interest, weighted by the nonnegative coefficients δ_ℓ:

$$J_k' = \mathbf{a}_k^T \mathbf{a}_k + \eta \sum_\ell \delta_\ell (I_{\mathrm{DZ},k,\ell} - I_{T,k,\ell})^2. \tag{48}$$

4.2 *Adjoint Model*

In this section, we derive the adjoint model that evaluates the gradient of the objective function in Eq. (48) with respect to the control solution $\mathbf{a}_k$. We make extensive utilization of the gradient propagation rules derived in Ref. 15; therefore, we will not explicitly derive the rule associated with each forward model operation unless specified otherwise. For convenience, we refer again to Table 1, which lists the gradient propagation rules for the basic operations utilized in Sec. 2.

To begin, we use the fact that the derivative of the objective function with respect to itself is unity:

$$\overline{J_k'} = \frac{\partial J_k'}{\partial J_k'} = 1. \tag{49}$$

The gradient contribution from the actuator stroke penalty term in the objective function is

$$\overline{\mathbf{a}}_k = 2\mathbf{a}_k \overline{J_k'}. \tag{50}$$

Starting with Eq. (48) at the end of the forward model and working backward, we have the following. The adjoint dark-zone intensity $\overline{I}_{\mathrm{DZ},k,\ell}$ at the ℓ'th control wavelength is

$$\overline{I}_{\mathrm{DZ},k,\ell} = 2\eta\delta_\ell (I_{\mathrm{DZ},k,\ell} - I_{T,k,\ell}) \overline{J_k'}, \tag{51}$$

and the adjoint dark-zone electric field vector $\overline{\mathbf{E}}_{\mathrm{DZ},k,\ell}$ is

$$\overline{\mathbf{E}}_{\mathrm{DZ},k,\ell} = 2\mathbf{E}_{\mathrm{DZ},k,\ell} \circ \overline{I}_{\mathrm{DZ},k,\ell}. \tag{52}$$

We now propagate the gradient backward through the semianalytical coronagraph model in Eqs. (22)–(27). We recall from Eq. (28) that $\mathbf{E}_{\mathrm{DZ},k,\ell}$ is a vector of electric field values of pixels inside the dark zone $\mathbf{1}_{\mathrm{DZ}}$. Using the gradient propagation rule for the array indexing operator in Table 1, we initialize the adjoint electric field contributed by the DMs, $\overline{\mathbf{E}}_{\mathrm{DM},k,\ell}$, as a 2D array of zeros, and then populate it with the elements of $\overline{\mathbf{E}}_{\mathrm{DZ},k,\ell}$ as

$$\overline{\mathbf{E}}_{\mathrm{DM},k,\ell}[\mathbf{1}_{\mathrm{DZ}}] = \overline{\mathbf{E}}_{\mathrm{DZ},k,\ell}. \tag{53}$$

Using the gradient propagation rule associated with the matrix Fourier transformation derived in Appendix B.2, we next have

$$\overline{\mathbf{E}}_{C,k,\ell} = \frac{\Delta x \Delta y}{\lambda_\ell} \mathrm{IMFT}\{\overline{\mathbf{E}}_{\mathrm{DZ},k,\ell}; \boldsymbol{\theta}_x, \boldsymbol{\theta}_y, \mathbf{x}, \mathbf{y}\}, \tag{54}$$

where IMFT denotes the inverse matrix Fourier transformation, defined as

$$\mathrm{IMFT}\{\mathbf{E}; \boldsymbol{\theta}_x, \boldsymbol{\theta}_y, \mathbf{x}, \mathbf{y}\} \triangleq \exp\{i2\pi \mathbf{x}\boldsymbol{\theta}_x^T\}\mathbf{E}\ \exp\{i2\pi \boldsymbol{\theta}_y \mathbf{y}^T\}. \tag{55}$$

Similarly, moving backward to the occulter plane, we have

$$\overline{\mathbf{E}}_{B,k,\ell} = -\frac{\Delta\theta_x \Delta\theta_y}{\lambda_\ell} \mathrm{IMFT}\{\mathbf{L} \circ \overline{\mathbf{E}}_{C,k,\ell}; \mathbf{x}, \mathbf{y}, \boldsymbol{\theta}_x, \boldsymbol{\theta}_y\}. \tag{56}$$

The adjoint apodizer-plane field has two contributing terms: one from the coordinate-reversed term in Eq. (26) and one from the field impinging on the occulter in Eq. (23). Therefore, the desired adjoint variable is a linear combination of the two. Reversing the coordinates of a variable in the forward model also reverses the coordinates of its adjoint as derived in Appendix B.3; thus, we have

$$\overline{\mathbf{E}}_{A,k,\ell} = \mathcal{R}\{\mathbf{L} \circ \overline{\mathbf{E}}_{C,k,\ell}\} + \frac{\Delta x \Delta y}{\lambda_\ell} \mathrm{IMFT}\{\mathbf{M}^* \circ \overline{\mathbf{E}}_{B,k,\ell}; \boldsymbol{\theta}_x, \boldsymbol{\theta}_y, \mathbf{x}, \mathbf{y}\}, \tag{57}$$

where $\mathbf{M}^*$ is the complex conjugate of $\mathbf{M}$. Note that by omitting the complex conjugation of $\mathbf{L}$ in Eqs. (56) and (57), we have assumed that the Lyot stop transmittance is purely real-valued, which is true for every Lyot-type coronagraph at the time of writing. Allowing the apodizing mask to be complex-valued, the adjoint field immediately before the apodizer is

$$\overline{\mathbf{E}'}_{A,k,\mathrm{DM1},\ell} = \overline{\mathbf{E}'}_{A,k,\mathrm{DM2},\ell} = \mathbf{A}^* \overline{\mathbf{E}}_{A,k,\ell}. \tag{58}$$

At this point, we propagate the gradient through the adjoint model for the two DMs, using the forward model described in Sec. 2.1.2. Before proceeding, we remind ourselves that the operator $\mathcal{P}\{\cdot\}$ defined in Eq. (13) in fact consists of a pair of angular spectrum propagations with an intermediate multiplication. Because we are interested in finding the gradient with respect to an intermediate variable, it is easier to treat each of these operations separately. Hence, we expand $\mathbf{E}'_{A,k,\mathrm{DM1},\ell}$ in Eq. (21) as

$$\boldsymbol{\psi}_{\mathrm{DM1},k-1,\ell} \triangleq \exp\{i\boldsymbol{\phi}_{1,k-1,\ell}\}, \tag{59a}$$

$$\boldsymbol{\psi}_{\mathrm{DM2},k-1,\ell} \triangleq \exp\{i\boldsymbol{\phi}_{2,k-1,\ell}\}, \tag{59b}$$

$$\mathbf{E}_{\mathrm{EP},k-1,\mathrm{DM1},\ell} \triangleq \mathbf{P} \circ \boldsymbol{\psi}_{\mathrm{DM1},k-1,\ell}, \tag{59c}$$

$$\mathbf{E}_{\mathrm{EP},k,\mathrm{DM1},\ell} \triangleq i\mathbf{E}_{\mathrm{EP},k-1,\mathrm{DM1},\ell} \circ \Delta\boldsymbol{\phi}_{1,k,\ell}, \tag{59d}$$

$$\mathbf{E}'_{\mathrm{IP},k,\mathrm{DM1},\ell} \triangleq \mathcal{A}\{\mathbf{E}_{\mathrm{EP},k,\mathrm{DM1},\ell}; \Delta z_{\mathrm{DM}}, \lambda_\ell\}, \tag{59e}$$

$$\mathbf{E}_{\mathrm{IP},k,\mathrm{DM1},\ell} \triangleq \mathbf{E}'_{\mathrm{IP},k,\mathrm{DM1},\ell} \circ \boldsymbol{\psi}_{\mathrm{DM2},k-1,\ell}, \tag{59f}$$

$$\mathbf{E}'_{A,k,\mathrm{DM1},\ell} \triangleq \mathcal{A}\{\mathbf{E}_{\mathrm{IP},k,\mathrm{DM1},\ell}; -\Delta z_{\mathrm{DM}}, \lambda_\ell\}, \tag{59g}$$

where the subscript "IP" is an abbreviation for "intermediate plane." Similarly, for DM2, we have

$$\mathbf{E}'_{\mathrm{IP},k-1,\mathrm{DM2},\ell} \triangleq \mathcal{A}\{\mathbf{E}_{\mathrm{EP},k-1,\mathrm{DM1},\ell}; \Delta z_{\mathrm{DM}}, \lambda_\ell\}, \tag{60a}$$

$$\mathbf{E}_{\mathrm{IP},k-1,\mathrm{DM2},\ell} \triangleq \mathbf{E}'_{\mathrm{IP},k-1,\mathrm{DM2},\ell} \circ \boldsymbol{\psi}_{\mathrm{DM2},k-1,\ell}, \tag{60b}$$

$$\mathbf{E}_{\mathrm{IP},k,\mathrm{DM2},\ell} \triangleq i\mathbf{E}_{\mathrm{IP},k-1,\mathrm{DM2},\ell} \circ \Delta\boldsymbol{\phi}_{2,k,\ell}, \tag{60c}$$

$$\mathbf{E}'_{A,k,\mathrm{DM2},\ell} \triangleq \mathcal{A}\{\mathbf{E}_{\mathrm{IP},k,\mathrm{DM2},\ell}; -\Delta z_{\mathrm{DM}}, \lambda_\ell\}. \tag{60d}$$

From here, the adjoint model follows two separate paths to account for the fact that $\mathbf{E}'_{A,k,\mathrm{DM1},\ell}$ is a function only of $\Delta\boldsymbol{\phi}_{1,k,\ell}$, and likewise $\mathbf{E}'_{A,k,\mathrm{DM2},\ell}$ is only a function of $\Delta\boldsymbol{\phi}_{2,k,\ell}$. We additionally make use of the gradient propagation rule for the angular spectrum propagator derived in Appendix B.1. We begin with DM2:

$$\overline{\mathbf{E}}_{\mathrm{IP},k,\mathrm{DM2},\ell} = \mathcal{A}\{\overline{\mathbf{E}'}_{A,k,\mathrm{DM2},\ell}; \Delta z_{\mathrm{DM}}, \lambda_\ell\}, \tag{61a}$$

$$\overline{\Delta\boldsymbol{\phi}}_{2,k,\ell} = -i\mathbf{E}^*_{\mathrm{IP},k-1,\mathrm{DM2},\ell} \circ \overline{\mathbf{E}}_{\mathrm{IP},k,\mathrm{DM2},\ell}. \tag{61b}$$

Using the expansion in terms of influence functions in Eq. (17), the adjoint contribution from the ℓ'th wavelength is

$$\bar{\mathbf{a}}_{2,k,\ell} = \frac{4\pi}{\lambda_\ell}\mathbf{F}_2^\dagger \overline{\Delta\boldsymbol{\phi}}_{2,k,\ell}. \tag{62}$$

We now return to Eq. (59d) to propagate through the adjoint model for the DM1 update:

$$\overline{\mathbf{E}}_{\mathrm{IP},k,\mathrm{DM1},\ell} = \mathcal{A}\{\overline{\mathbf{E}'}_{A,k,\mathrm{DM1},\ell}; \Delta z_{\mathrm{DM}}, \lambda_\ell\}, \tag{63a}$$

$$\overline{\mathbf{E}'}_{\mathrm{IP},k,\mathrm{DM1},\ell} = \boldsymbol{\psi}^*_{\mathrm{DM2},k-1,\ell} \circ \overline{\mathbf{E}}_{\mathrm{IP},k,\mathrm{DM1},\ell}, \tag{63b}$$

$$\overline{\mathbf{E}}_{\mathrm{EP},k,\mathrm{DM1},\ell} = \mathcal{A}\{\overline{\mathbf{E}'}_{\mathrm{IP},k,\mathrm{DM},\ell}; -\Delta z_{\mathrm{DM}}, \lambda_\ell\}, \tag{63c}$$

$$\overline{\Delta\boldsymbol{\phi}}_{1,k,\ell} = -i\mathbf{E}^*_{\mathrm{EP},k-1,\mathrm{DM1},\ell} \circ \overline{\mathbf{E}}_{\mathrm{EP},k,\mathrm{DM1},\ell}, \tag{63d}$$

$$\bar{\mathbf{a}}_{1,k,\ell} = \frac{4\pi}{\lambda_\ell}\mathbf{F}_1^\dagger \overline{\Delta\boldsymbol{\phi}}_{1,k,\ell}. \tag{63e}$$

The total gradient is formed by concatenating the individual gradients for DM1 and DM2, summing the contributions from each control wavelength, and including the contribution from the $\mathbf{a}_k^T\mathbf{a}_k$ term in the objective function, yielding the final result:

$$\bar{\mathbf{a}}_k = 2\mathbf{a}_k + \sum_{\ell=1}^{L}\begin{bmatrix}\bar{\mathbf{a}}_{1,k,\ell}\\ \bar{\mathbf{a}}_{2,k,\ell}\end{bmatrix}. \tag{64}$$

4.3 *Computing Control Solutions*

At this point, we have derived a forward model that evaluates the objective function J'_k in Eq. (48), as well as an adjoint model that evaluates its gradient with respect to the DM actuator commands $\mathbf{a}_k$. In practice, both the forward model and adjoint model will be implemented as subroutines in a user's programming language of choice, parameterized by the focal-plane electric field estimate $\hat{\mathbf{E}}_{\mathrm{ab},k}$, the penalty parameter η, and the target dark-zone integrated intensity $I_{T,k}$. Given concrete values for $\hat{\mathbf{E}}_{\mathrm{ab},k}$, η, and $I_{T,k}$, these subroutines become functions of $\mathbf{a}_k$ alone. The parameterized forward model and adjoint model are provided to a gradient-based optimization algorithm along with a starting guess for the solution, which then attempts to minimize the objective function.

Though any gradient-based optimization algorithm is theoretically sufficient to solve the proposed control problem, the specific choice affects both the rate of convergence to the global optimum in each control iteration, as well as the memory efficiency. Quasi–Newton algorithms such as the well-known Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm utilize the computed gradient to iteratively approximate the inverse of the Hessian matrix of second derivatives and converge more rapidly than first-order methods such as gradient descent and the conjugate gradient method.[14] However, because the BFGS algorithm stores the full approximate inverse Hessian matrix with dimension $N_{\text{act}} \times N_{\text{act}}$, it can become memory-inefficient and slow when the number of actuators becomes large. The limited-memory BFGS algorithm (L-BFGS)[14] instead stores the $N_{\text{grad}} \ll N_{\text{act}}$ most recent gradient vectors, with which it implicitly computes inverse Hessian-vector products to determine the search direction. The optimal choice of N_{grad} is problem-dependent, but in many cases a value between 3 and 20 is acceptable.[14] Many software libraries provide standard implementations of a range of gradient-based optimization algorithms, including both BFGS and L-BFGS. For these reasons, we primarily employ the L-BFGS algorithm to compute solutions and will make this assumption when analyzing the computational complexity of the proposed algorithm in Sec. 6.

As we noted in Sec. 4, the proposed objective function is globally convex, so the control solution is independent of the choice of starting guess. However, a starting guess that is closer to the solution will help the optimization algorithm converge to the global optimum more rapidly; a good choice in this context is to provide the actuator solution obtained in the previous control iteration, with $\mathbf{a}_{-1} = 0$.

5 Simulation Results

In this section, we present simulation results using the proposed algorithm and compare its CPU time and memory efficiency to the SM algorithm. To compare the algorithms as fairly as possible, all simulations were performed on a computing server with 527 gigabytes of RAM and run on a single 3.2 GHz core. For simplicity, the simulations were noiseless and assumed perfect knowledge of the aberrated electric field.

5.1 *Coronagraph Design*

To place the simulation results in a context relevant to future high-contrast imaging missions, we simulated the small-angle APLC design[34] for LUVOIR architecture "A," submitted to the 2020 Astrophysics Decadal Survey (courtesy of Rémi Soummer). This design produces an annular dark zone extending from $3.5\lambda_0/D$ to $12\lambda_0/D$ with a nominal raw contrast of 10^{-10} over a 10% fractional bandpass. The EP mask, apodizing mask, and Lyot stop are each 1000×1000 arrays, and the focal-plane mask is an opaque circular spot with radius $3.4\lambda_0/D$. Figure 3 shows the coronagraph masks in each plane along with the aberration-free stellar image.

We simulated control over the full extent of the dark zone and assumed a center wavelength of $\lambda_0 = 550$ nm. The detector plane was Nyquist sampled at the shortest wavelength, resulting in a total of 1640 samples inside the control region. The two DMs were modeled after square MEMS DMs with Gaussian influence functions separated by a distance with Fresnel number 1562.5, which is consistent with the reference optical design in Ref. 35. We considered several

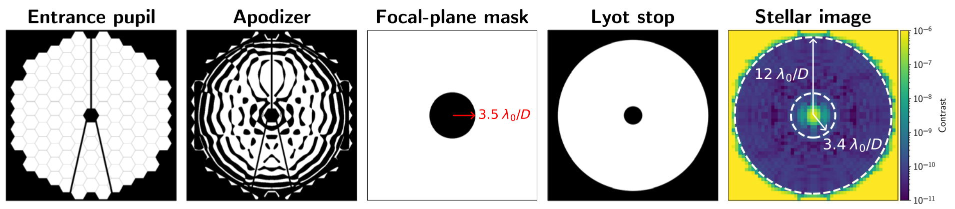


Fig. 3 Coronagraph masks for the LUVOIR-A APLC design used for simulating the proposed control algorithm described in Sec. 4. The aberration-free stellar image is shown in the rightmost panel, with the nominal inner and outer dark zone radii indicated in white.

different DM formats: 50×50, 64×64, and 128×128. These are representative of, respectively, Boston Micromachines 2K and 4K MEMS DMs available with current technology, and future DMs expected to be available by the projected LUVOIR launch date in the late 2030s.[2] We assumed for simplicity that all actuators were active and utilized by the control loop in every iteration; in a realistic scenario, some fraction of these actuators, primarily those outside the nominal beam radius at the pupil, will be disconnected from the control electronics and unavailable for wavefront control. Furthermore, actuators that have a sufficiently weak impact on the control problem can be ignored by examining the columns of the Jacobian matrix for SM, or the elements of the gradient vector $\bar{\mathbf{a}}_k$ for the proposed algorithm. Therefore, the DM formats considered in this paper serve primarily as concrete examples to illustrate the scaling properties of the two algorithms relative to one another.

5.2 *Comparison with Stroke Minimization*

We introduced a 5-nm peak-to-valley phase-only wavefront error with an inverse-square power spectral density into the EP of the coronagraph and simulated the proposed method and SM. Both algorithms used three control wavelengths across the 10% fractional bandpass of the coronagraph. The control weights δ_ℓ were chosen to have a value of unity at the center wavelength and 0.5 at the band-edge wavelengths. In each iteration, the target intensity was chosen as $I_{T,k} = 0.7 I_{T,k-1}$, which was informed by our experience with SM experiments on the HiCAT[12] testbed. A total of 25 iterations were performed for each algorithm, which was sufficient to converge to the target 10^{-10} contrast.

For the proposed method, the penalty parameter was chosen as $\eta_k = 1/I_{T,k}^2$ as described in Sec. 4.1. For numerical stability, the actuator commands were measured in units of waves at the center wavelength so that both terms in the objective function in Eq. (48) were within a few orders of magnitude of one another. The L-BFGS optimization algorithm[14] provided by the `scipy.optimize` package[36] was used to solve the optimization problem in each control iteration. In this implementation, the L-BFGS algorithm terminates when

$$\|\bar{\mathbf{a}}_k\|_\infty \triangleq \max_i |\bar{a}_{k,i}| \leq \varepsilon, \tag{65}$$

where $\bar{a}_{k,i}$ is the i'th element of the gradient vector $\bar{\mathbf{a}}_k$ and ε is a small real number. Smaller choices for ε yield numerical solutions closer to the global optimum, at the cost of longer convergence time. We found that acceptable solutions could be obtained with a relatively coarse value $\varepsilon = 10^{-3}$ and $N_{\text{grad}} = 10$ gradient vectors for the approximation of the inverse Hessian matrix.

For SM, the Lagrange multiplier line search was carried out using a multiplicative step size $\beta = \sqrt{2}$; this choice was also informed by the authors' experience with SM experiments on HiCAT. In other words, at the n'th step of the Lagrange multiplier line search with value μ_n and actuator solution $\mathbf{a}_k^*(\mu_n)$ calculated using Eq. (36b), if the integrated dark-zone intensity constraint $I_{\text{DZ},k} < I_{T,k}$ in Eq. (30) is not satisfied, then the next Lagrange multiplier value is chosen as $\mu_{n+1} = \beta\mu_n$. A logarithmic line search with $\beta > 1$ takes progressively larger step sizes as the number of search steps increases; therefore, it takes fewer search steps to reach the optimal value μ^* for which $I_{\text{DZ},k} = I_{T,k}$ than, for example, a high-resolution backtracking line search whose step size decreases as the optimal value is approached.[14] A side effect, though, is that such a line search will almost certainly overstep μ^* before terminating, producing a larger-than-desired intensity correction in each control iteration. In a noiseless simulation with perfect knowledge of the aberrated electric field and no model mismatch, this simply causes our implementation of SM algorithm to slightly overshoot the intensity constraint in the optimistic direction. However, in an experimental scenario with noise, estimation error, and model mismatch, overstepping μ^* can cause the control loop to become unstable.

Figure 4 shows the total CPU time and worst-case memory consumption for both SM and the proposed algorithm with each DM format, as well as achieved contrast and peak-to-valley actuator stroke as a function of control iteration. Figure 5 shows the final DM solutions computed by the proposed algorithm and SM. We note here that because all simulations were

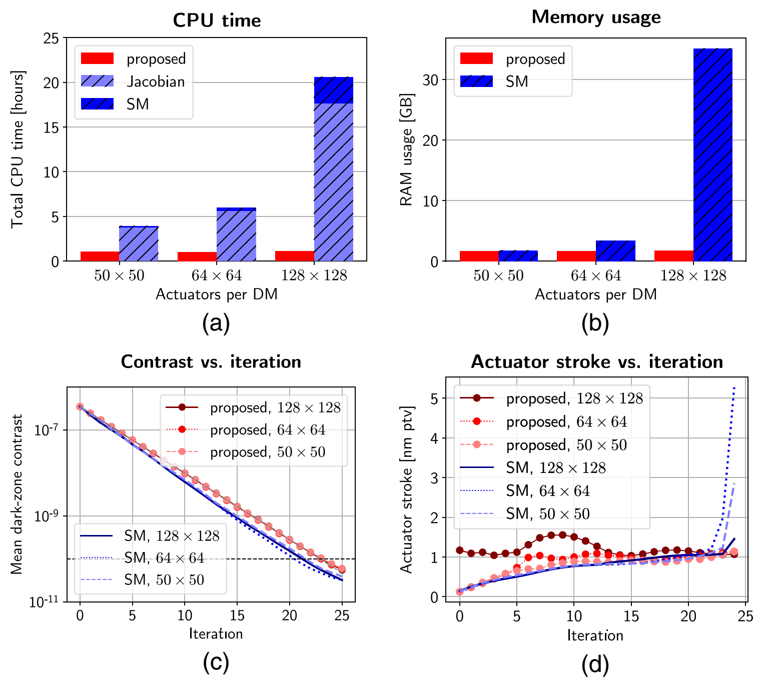
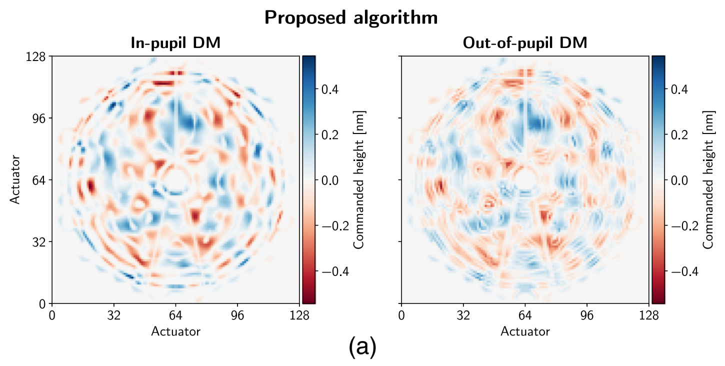


Fig. 4 (a) Total CPU time, (b) worst-case memory consumption, and (c) contrast and (d) peak-to-valley actuator stroke as a function of iteration for the proposed algorithm and SM with each DM actuator format. The CPU time associated with precomputing the Jacobian matrix for SM prior to the beginning of closed-loop control is also shown in light blue. All simulations were performed on identical single-core processors, so here the CPU time is a proxy for total demand placed on the processor by each algorithm. Because computation times in practice are a function of available resources, architecture, and implementation, the results in (a) serve to illustrate the relative, rather than absolute, performance of the two algorithms. In a real system, the duration of the control loop will be dominated by data acquisition times.

carried out on an identical single-core processor, the CPU times shown in Fig. 4 are a proxy for total floating-point operations; in a real flight system, the elapsed time of each control iteration will be dominated by exposure times for image acquisition. We measured memory consumption as a time series during execution of each simulation using the Python `memory-profiler` tool, from which we calculated the overall maximum at any given time. This represents the minimum memory requirement for a target platform to support execution of each simulation.

From these results, we can gather several important conclusions. First, the CPU time of the SM algorithm is dominated in all cases by the computation of the Jacobian matrix, shown in light blue. Once this has completed, the SM algorithm computes control solutions more rapidly than our proposed algorithm for the 50×50 and 64×64 formats. On the contrary, the CPU time of our algorithm is effectively invariant to the DM format; therefore, the overall computation of SM compared with our algorithm for these cases is tied directly to the degree to which the Jacobian evaluation calculations are minimized. However, as the problem size grows, a threshold is reached at which SM will be slower in CPU time than our algorithm regardless of the speed of Jacobian calculation; for the 128×128 case, even under the assumption of zero Jacobian calculation time, the difference in CPU time between the two algorithms is more than a factor of two. We reiterate that the CPU time in and of itself is not critical to the success of the control loop in a space environment due to the comparatively long times associated with data acquisition. Nevertheless, in order for a wavefront control algorithm to be feasibly implemented on flight hardware in an on-orbit operational scenario, minimizing the computational demand placed on the flight computer is an important consideration.

We also reiterate that these simulations were carried out under the least-expensive relinearization scenario for SM, which was computing the Jacobian once prior to the start of closed-



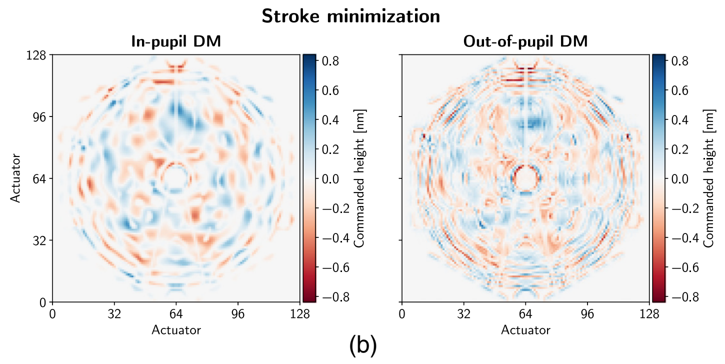


Fig. 5 Final DM actuator solutions for the 128×128 actuator format obtained by (a) the proposed algorithm and (b) SM. Note that the solutions from each algorithm are plotted using different color scales. While they share broadly similar low-order features, the SM solutions feature additional high-order components because the algorithm begins to compensate for diffraction effects from pupil features after the wavefront aberrations are fully corrected.

loop control. As experiments have demonstrated, a real control loop with model mismatches will require at least one Jacobian recalculation prior to convergence, and in many cases more than one.[7,37,38] Meanwhile, our proposed algorithm by construction is linearized around the current state of the DMs at all times, because the current commands simply serve as parameters to the forward and adjoint models.

Now we turn our attention to arguably the more important metric shown in Fig. 4: memory consumption. Here, because the Jacobian matrix is dense and has no particular structure to exploit, its memory footprint is unavoidable. For the smallest problem, the 50×50 format, the Jacobian matrix is manageable in size, and the worst-case memory consumption of SM and our proposed method are comparable (1.78 GB by SM versus 1.71 GB by the proposed algorithm). For the 64×64 format, the two algorithms differ by approximately a factor of two in memory consumption, with SM consuming ~3.34 GB versus 1.71 GB by our proposed algorithm. At the largest problem size, the 128×128 format, SM consumes ~35 GB of memory, whereas our proposed algorithm still only consumes ~1.75 GB: a reduction of ~95%. As was the case for the CPU time metrics, our proposed method's memory footprint is more or less invariant to the problem size, for reasons analyzed in Sec. 6.

5.3 *Dependence on Penalty Parameter*

We simulated the proposed algorithm with multiple values of the penalty parameter scaling factor η_{00} to examine its effect on the efficiency and contrast convergence of the proposed algorithm. Figure 6 shows these results. As discussed in Sec. 4.1, with a quadratic intensity penalty term, the solutions are guaranteed to achieve the target contrast within a tolerance determined by the value of the penalty parameter η. With a larger value of η_{00}, the algorithm adheres to the target more closely, but requires more numerical iterations of the optimization algorithm to find a solution, ultimately increasing CPU time. Conversely, decreasing η_{00} reduces CPU time, at the cost of greater variability in the achieved intensity.

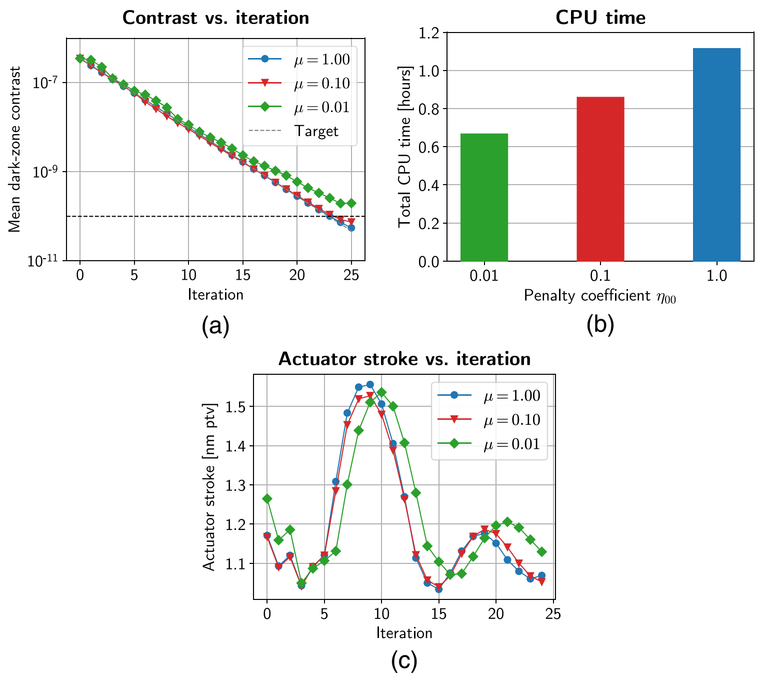


Fig. 6 Mean dark-zone contrast as (a) a function of iteration, (b) total elapsed CPU time, and (c) peak-to-valley actuator stroke as a function of iteration for three different values of the penalty parameter scaling factor η_{00} for the 128×128 DM actuator format. As η_{00} is reduced, our algorithm converges more rapidly to a solution in each control iteration, but the achieved contrast deviates more from the target contrast on average.

6 Discussion

6.1 *Computational Complexity*

In this section, we comment on the time and memory complexity of our proposed algorithm and SM. We note again that in this context, "time complexity" refers specifically to CPU load rather than the elapsed time of the control loop; in a real system, data acquisition times will be the dominant source of the duration of the loop.

6.1.1 *Time complexity*

We found that with the parameter values from Sec. 5, the CPU time performance of our algorithm was mostly driven by the angular spectrum propagations between the two DMs; this is the principal reason for which the CPU times in Fig. 4(a) varied so little with actuator count. Our pupil-plane arrays were 1000 pixels across and zero-padded by a factor of 1.05, giving a total array size $N_\mathbf{P} = 1050^2$. We observed no difference in the achieved contrast in various simulations using a range of padding factors between 1.05 and 2, indicating that a factor of 1.05 was sufficient to capture the behavior of DM2, which is separated from the pupil by a relatively large Fresnel number of $N_F = 1562.5$. By comparison, the dark zone only contained 1640 samples, and the DM command vector only had $N_\text{act} = 2 \times 128^2$ elements.

More concretely, our algorithm requires four angular spectrum operations to evaluate Eqs. (59e), (59g), (60a), and (60d) in the forward model and an additional three in the adjoint model in Eqs. (61a), (63a), and (63c), each involving a pair of FFTs and an elementwise multiplication; the dominant contribution is from the FFTs, whose time complexity is proportional to $N_\mathbf{P} \log_2(N_\mathbf{P})$ floating-point operations (flops).[28] Therefore, decreasing $N_\mathbf{P}$ either using lower-resolution masks or by further decreasing zero-padding margins impacts the most computationally expensive component of propagation.

Conversely, increasing the actuator count only impacts two operations in the forward model, Eqs. (4) and (17), their corresponding adjoint model operations in Eqs. (62) and (63e), and finally Eq. (64), all of which are linear in N_{act}[30] and are lower-dimensional in the first place. It is worth noting that we used the `hcipy` package[39] to model the DMs, which represents the influence function matrices $\mathbf{F}_1$ and $\mathbf{F}_2$ as sparse matrices with $N_{\mathrm{IF}} \ll N_{\mathbf{P}}$ nonzero elements in each column, where "IF" stands for "influence function," by exploiting the fact that the individual influence functions are highly localized, significantly reducing the actual flops for Eqs. (4), (17), (62), and (63e) to approximately $\frac{1}{2} N_{\mathrm{IF}} N_{\mathrm{act}}$ each, and where the factor of $1/2$ accounts for the fact that each matrix only corresponds to a single DM, whereas N_{act} is the total over both DMs. In the future, we will explore convolutional models for the DM surfaces as an alternative that does not necessitate the construction of influence function matrices at all. The time complexity of calculating solutions with the L-BFGS algorithm is similarly linear in N_{act}, requiring $N_{\mathrm{act}}(4N_{\mathrm{grad}} + 1)$ flops to evaluate the product between the inverse Hessian matrix approximation and the gradient each time the gradient is evaluated.[14]

The number of control wavelengths L also has a strong impact because each wavelength requires a separate evaluation of the coronagraph model $\mathcal{C}\{\cdot\}$ and its adjoint model, multiplying the angular spectrum operation count for each gradient evaluation by L. The number of L-BFGS iterations required to converge to a solution in each control iteration, which is problem dependent, has a similar effect by requiring multiple gradient evaluations, each of which involves a forward and reverse sweep; we denote the total number of gradient evaluations needed for L-BFGS convergence by N_{LBFGS}. Therefore, in this approximation, the time complexity of our proposed algorithm scales with actuator count and pupil array size as

$$t_{\mathrm{prop}} \propto N_{\mathrm{LBFGS}}[L[14N_{\mathbf{P}} \log_2(N_{\mathbf{P}}) + 7N_{\mathbf{P}} + 2N_{\mathrm{IF}}N_{\mathrm{act}} + N_{\mathrm{act}}] + N_{\mathrm{act}}(4N_{\mathrm{grad}} + 1)]. \tag{66}$$

The first term represents the cost of evaluating the gradient, which consists of, respectively, the seven angular spectrum propagations, each involving two FFTs and one elementwise multiplication, four products between the actuator command vector and influence function matrices, and the linear combination of gradient vectors at each wavelength in Eq. (64). The second term represents the cost of the L-BFGS algorithm as explained above.

For SM, evaluating each monochromatic Jacobian $\mathbf{G}_{\ell,k}$ with Eq. (9) requires $N_{\mathrm{act}} + 1$ evaluations of the operator $\mathcal{C}\{\cdot\}$. The particular implementation of $\mathcal{C}\{\cdot\}$ thus determines the required number of flops per DM actuator; with L control wavelengths, this value is multiplied by L. In our simulations in Sec. 5, for the sake of simplicity we used the same implementation of $\mathcal{C}\{\cdot\}$ used by our proposed algorithm. However, as we have noted, packages such as FALCO in fact use a separate forward model with greatly reduced array sizes, which reduces the number of flops for Jacobian calculation substantially. Reusing the Jacobian also amortizes this cost over multiple iterations. Second, computing $\mathbf{M}'_k$ from Eq. (41) requires $LN_{\mathrm{act}}^2 N_{\mathrm{pix}}$ flops.[30] Finally, solving the linear system in Eq. (37) using the Cholesky decomposition as explained in Sec. 2.3 requires $N_{\mathrm{act}}^3/3 + 2N_{\mathrm{act}}^2$ flops. This linear system must be solved separately for each Lagrange multiplier value μ encountered during the line search, multiplying this value by a problem-dependent constant factor N_μ.

Under the simplistic assumption that our proposed algorithm and SM use the same implementation for $\mathcal{C}\{\cdot\}$, the time complexity of SM grows approximately as

$$t_{\mathrm{SM}} \propto L(N_{\mathrm{act}} + 1)[8N_{\mathbf{P}} \log_2(N_{\mathbf{P}}) + 4N_{\mathbf{P}} + N_{\mathrm{IF}}N_{\mathrm{act}}] + LN_{\mathrm{act}}^2 N_{\mathrm{pix}} + N_\mu \left(2N_{\mathrm{act}}^2 + \frac{1}{3}N_{\mathrm{act}}^3\right). \tag{67}$$

The first term represents the contribution from the Jacobian calculation and is identical to Eq. (66) minus the operations from the adjoint model and L-BFGS algorithm, and with $N_{\mathrm{act}} + 1$ in place of N_{LBFGS}. The second term is the contribution from the calculation of $\mathbf{M}'_k$, and the final term is the contribution from the linear solver and Lagrange multiplier line search.

Comparing Eqs. (66) and (67), we can make several useful conclusions. First, under the assumptions above, SM is dominated by the complexity of evaluating the Jacobian matrix since, in our simulations, $N_{\mathbf{P}} \gg N_{\mathrm{act}}$. However, if the first term of Eq. (67) can be made very small by minimizing $N_{\mathbf{P}}$ as achieved by FALCO, the rightmost term dominates, which is cubic in N_{act} and grows much more rapidly than Eq. (66), which is only linear in N_{act}. When N_{act} becomes sufficiently large, this term becomes even larger than the contribution from the angular spectrum propagations in Eq. (66) regardless of how quickly the Jacobian matrix is computed; our results in Sec. 5 suggest that this inflection point is somewhere between the 64×64 and 128×128 DM formats.

6.1.2 *Memory complexity*

For our proposed algorithm, apart from storing the model parameters themselves, only four forward variables are needed for the reverse sweep: $\mathbf{E}_{\mathrm{DZ},k,\ell}$ in Eq. (52), $\mathbf{E}_{\mathrm{IP},k-1,\mathrm{DM2},\ell}$ in Eq. (61b), $\boldsymbol{\psi}_{\mathrm{DM2},k-1,\ell}$ in Eq. (63b), and $\mathbf{E}_{\mathrm{EP},k-1,\mathrm{DM1},\ell}$ in Eq. (63d). This means that the theoretical minimum total number of complex-valued array elements stored by the proposed algorithm, including the N_{grad} gradient vectors stored by L-BFGS (see Sec. 4.3), is $3LN_{\mathbf{P}} + LN_{\mathrm{pix}} + \frac{1}{2}N_{\mathrm{grad}}N_{\mathrm{act}}$, where the factor of $\frac{1}{2}$ accounts for the fact that the actuator gradient vectors are purely real. From this we can see that the array sizes $N_{\mathbf{P}}$, N_{act}, and N_{pix} compound additively.

For the worst-case simulation in Sec. 5, where $L = 3$, $N_{\mathbf{P}} = 1050^2$, $N_{\mathrm{pix}} = 1640$, and $N_{\mathrm{act}} = 2 \times 128^2$, assuming double-precision floating point representation, this totals to $\sim$0.15 GB. This value is dominated by the intermediate pupil-plane forward variables, which are much higher-dimensional than the gradient vector or dark zone as we discussed in Sec. 6.1.1; in fact, decreasing N_{act} to 2×64^2 only reduces the storage requirement to $\sim$0.148 GB. This analysis agrees with the results in Sec. 5, which showed that the memory consumption of the proposed algorithm was essentially invariant to N_{act}. In reality, the simulation of the proposed algorithm consumed $\sim$1.75 GB of storage during execution, most of which is accounted for by the numerical model itself and other simulation variables such as the control history.

On the other hand, SM must compute and store L separate complex-valued Jacobian matrices along with the real-valued matrix $\mathbf{M}_k'$ from Eq. (41). Solving the linear system in Eq. (37) through the Cholesky decomposition[30] generates two additional triangular matrices each with N_{act}^2 real-valued elements. Therefore, the minimum storage for SM is given by $LN_{\mathrm{pix}}N_{\mathrm{act}} + \frac{3}{2}N_{\mathrm{act}}^2$ complex numbers. Here, the problem is twofold: the quadratic dependence on N_{act} and the fact that simultaneous increases in N_{pix} and N_{act} compound multiplicatively rather than additively as before. Using the same values from above, this equates to $\sim$26.4 GB, a modest underestimate of the 35 GB actually consumed by the simulation, or 2.1 GB when $N_{\mathrm{act}} = 2 \times 64^2$, slightly less than the 3.34 GB used in reality. Our simple analysis does not account for details specific to the implementation of the linear solver such as temporary array allocations, which may explain the discrepancy between our predictions and the true values and meaning that these figures should be taken as a lower bound.

6.2 *System Identification*

Sun et al.[40,41] recently described an approach to system identification that learns the parameters of the Jacobian matrix for a given control problem from experimental data using a reinforcement learning approach based on the expectation-maximization (EM) algorithm, which reduces model mismatch and improves control convergence. However, the control model that generates the Jacobian matrix is fully described by a much smaller number of free parameters. In this paper, only the gradient with respect to the DM actuator update was derived, but it is straightforward to derive gradients with respect to many other model parameters, such as pupil transmittance, pupil shear, and the DM influence function, by defining appropriate adjoint variables in the adjoint model. Using these analytic gradients could enable a straightforward approach to system identification wherein the controller alternates between optimizing DM solutions to improve dark-zone contrast, and optimizing against the system parameters given the measured data history to tune the parameters of the forward model.

7 Conclusions and Future Work

In this paper, we describe an approach to focal-plane wavefront control that uses fast analytic gradients to efficiently obtain DM solutions without requiring the explicit computation of a Jacobian matrix. We tested the proposed algorithm in simulation using the small-angle APLC design for the LUVOIR-A mission concept and showed that it has improved asymptotic computational efficiency compared with current Jacobian-based algorithms, such as SM and EFC. The benefits are especially pronounced when the number of DM actuators are large; for the 128×128 case, our algorithm consumes $\sim 5\%$ of the RAM utilized by SM. By reducing the overall computational demands of wavefront control, we argue that our algorithm opens a path toward feasibility for on-orbit wavefront control, which will be a key trade study for future flagship observatories with a focus on direct imaging such as LUVOIR and HabEx. Future work will involve experimental demonstration of the proposed algorithm on a high-contrast optical testbed, computational enhancements to improve CPU time and memory efficiency, and the extension to system identification outlined in Sec. 6.2.

8 Appendix A: Reverse-Mode Algorithmic Differentiation: A Simple Example

Consider a scalar variable J defined as the output of a sequence of three operations, $J = f(\mathbf{g}(\mathbf{h}(\mathbf{x})))$, with a vector-valued input variable $\mathbf{x}$. For generality, we assume that the functions $\mathbf{g}$ and $\mathbf{h}$ each take a vector as input and produce a vector as output. To implement J as a numerical algorithm, we introduce the forward variables $\mathbf{x}_1$ and $\mathbf{x}_2$ to represent intermediate quantities in the computation of J:

$$\mathbf{x}_1 = \mathbf{h}(\mathbf{x}), \quad \mathbf{x}_2 = \mathbf{g}(\mathbf{x}_1), \quad J = f(\mathbf{x}_2). \tag{68}$$

The derivative $\partial J/\partial \mathbf{x}$ is expanded using the familiar chain rule of calculus

$$\frac{\partial J}{\partial \mathbf{x}} = \frac{\partial J}{\partial \mathbf{x}_2}\frac{\partial \mathbf{x}_2}{\partial \mathbf{x}_1}\frac{\partial \mathbf{x}_1}{\partial \mathbf{x}}. \tag{69}$$

Applying the definition of the adjoint variables from Sec. 3, we find that

$$\bar{\mathbf{x}}_2^\dagger \triangleq \frac{\partial J}{\partial \mathbf{x}_2} = \frac{\partial f}{\partial \mathbf{x}_2}, \tag{70}$$

$$\bar{\mathbf{x}}_1^\dagger \triangleq \frac{\partial J}{\partial \mathbf{x}_1} = \frac{\partial J}{\partial \mathbf{x}_2}\frac{\partial \mathbf{x}_2}{\partial \mathbf{x}_1} = \bar{\mathbf{x}}_2^\dagger \frac{\partial \mathbf{g}}{\partial \mathbf{x}_1}, \tag{71}$$

$$\bar{\mathbf{x}}^\dagger \triangleq \frac{\partial J}{\partial \mathbf{x}} = \frac{\partial J}{\partial \mathbf{x}_2}\frac{\partial \mathbf{x}_2}{\partial \mathbf{x}_1}\frac{\partial \mathbf{x}_1}{\partial \mathbf{x}} = \bar{\mathbf{x}}_1^\dagger \frac{\partial \mathbf{h}}{\partial \mathbf{x}}. \tag{72}$$

Transposing Eqs. (70)–(72), we can write this more succinctly as the following sequence:

$$\bar{\mathbf{x}}_2 = \left(\frac{\partial f(\mathbf{x}_2)}{\partial \mathbf{x}_2}\right)^\dagger, \quad \bar{\mathbf{x}}_1 = \left(\frac{\partial \mathbf{g}(\mathbf{x}_1)}{\partial \mathbf{x}_1}\right)^\dagger \bar{\mathbf{x}}_2, \quad \bar{\mathbf{x}} = \left(\frac{\partial \mathbf{h}(\mathbf{x})}{\partial \mathbf{x}}\right)^\dagger \bar{\mathbf{x}}_1. \tag{73}$$

With any particular value for $\mathbf{x}$, the values of $\mathbf{x}_1$, $\mathbf{x}_2$, and J are computed by the sequence in Eq. (68). The values of $\mathbf{x}$, $\mathbf{x}_1$, and $\mathbf{x}_2$ are then used to evaluate the sequence of operations in Eq. (73) from left to right, ultimately yielding the desired derivative, $\bar{\mathbf{x}}$. In the parlance introduced in Sec. 3, Eq. (68) describes a forward model that computes the forward variables $\mathbf{x}$, $\mathbf{x}_1$, and $\mathbf{x}_2$. Likewise, Eq. (73) comprises the adjoint model that evaluates the adjoint variables $\bar{\mathbf{x}}_2$, $\bar{\mathbf{x}}_1$, and $\bar{\mathbf{x}}$.

As discussed in Sec. 3, though the adjoint model in Eq. (73) is written in terms of the Jacobian matrices of the operations $\mathbf{g}$ and $\mathbf{h}$, in practice the Jacobian matrices need not be explicitly constructed; instead, the adjoint model usually will be written as a sequence of closed-form operations in a similar fashion to the forward model. For instance, consider a concrete example with the following forward model:

$$\mathbf{x}_1 = \exp\{i\mathbf{x}\}, \quad \mathbf{x}_2 = \mathrm{FFT}\{\mathbf{x}_1\}, \quad J = \|\mathbf{x}_2\|^2, \tag{74}$$

where $\|\cdot\|$ denotes the Euclidean norm. Using the well-known fact that $\partial J/\partial \mathbf{x}_2^\dagger = 2\mathbf{x}_2$, along with the gradient propagation rules in Table 1, we can write down the adjoint model easily as

$$\bar{\mathbf{x}}_2 = 2\mathbf{x}_2, \quad \bar{\mathbf{x}}_1 = \mathrm{IFFT}\{\bar{\mathbf{x}}_2\}, \quad \bar{\mathbf{x}} = \Im\{\bar{\mathbf{x}}_1 \circ \mathbf{x}_1^*\}, \tag{75}$$

where * denotes complex conjugation without transposing. Note that the adjoint model is a function of the forward variables $\mathbf{x}_1$ and $\mathbf{x}_2$; the values of these variables are cached during the forward sweep and passed as parameters to the adjoint model, which is then evaluated by the reverse sweep.

9 Appendix B: Additional Gradient Propagation Rules

9.1 *Angular Spectrum Propagator*

Let $\mathbf{Y} = \mathcal{A}\{\mathbf{X}; \Delta z, \lambda\}$. By Eq. (11), we can write this as the following sequence of operations:

$$\mathbf{Y}_1 = \mathrm{FFT}\{\mathbf{X}\}, \tag{76a}$$

$$\mathbf{Y}_2 = \mathbf{H}(\Delta z, \lambda) \circ \mathbf{Y}_1, \tag{76b}$$

$$\mathbf{Y} = \mathrm{IFFT}\{\mathbf{Y}_2\}. \tag{76c}$$

Using the gradient propagation rules in Table 1, the corresponding adjoint model is

$$\bar{\mathbf{Y}}_2 = \mathrm{FFT}\{\bar{\mathbf{Y}}\}, \tag{77a}$$

$$\bar{\mathbf{Y}}_1 = \mathbf{H}^*(\Delta z, \lambda) \circ \bar{\mathbf{Y}}_2, \tag{77b}$$

$$\bar{\mathbf{X}} = \mathrm{IFFT}\{\bar{\mathbf{Y}}_1\}. \tag{77c}$$

Recognizing from Eq. (12) that $\mathbf{H}^*(\Delta z, \lambda) = \mathbf{H}(-\Delta z, \lambda)$, we arrive at the final result:

$$\bar{\mathbf{X}} = \mathcal{A}\{\bar{\mathbf{Y}}; -\Delta z, \lambda\}, \tag{78}$$

which is simply an angular spectrum propagation of the adjoint variable $\bar{\mathbf{Y}}$ in the reverse direction.

9.2 *Matrix Fourier Transform*

Let $\mathbf{Y} = \mathrm{MFT}\{\mathbf{X}; \mathbf{x}, \mathbf{y}, \boldsymbol{\theta}_x, \boldsymbol{\theta}_y\}$. We proceed similarly to above, writing down the sequence of operations comprising Eq. (24) in sequential order:

$$\mathbf{Y}_1 = \mathbf{X} \exp\{-i2\pi \mathbf{y}\boldsymbol{\theta}_y^T\}, \tag{79a}$$

$$\mathbf{Y} = \exp\{-i2\pi \boldsymbol{\theta}_x \mathbf{x}^T\}\mathbf{Y}_1. \tag{79b}$$

We emphasize that for this operation, $\mathbf{X}$ is to be treated as a 2D array, and the multiplications above are matrix multiplications. The adjoint model is then

$$\bar{\mathbf{Y}}_1 = \exp\{i2\pi \mathbf{x}\boldsymbol{\theta}_x^T\}\bar{\mathbf{Y}}, \tag{80a}$$

$$\bar{\mathbf{X}} = \bar{\mathbf{Y}}_1 \exp\{i2\pi \boldsymbol{\theta}_y \mathbf{y}^T\}. \tag{80b}$$

Putting the two expressions together, we have

$$\bar{\mathbf{X}} = \exp\{i2\pi \mathbf{x}\boldsymbol{\theta}_x^T\}\bar{\mathbf{Y}} \exp\{i2\pi \boldsymbol{\theta}_y \mathbf{y}^T\}, \tag{81}$$

which represents an inverse Fourier transform from the coordinate axes $(\boldsymbol{\theta}_x, \boldsymbol{\theta}_y)$ to the axes $(\mathbf{x}, \mathbf{y})$, or more succinctly

$$\bar{\mathbf{X}} = \mathrm{IMFT}\{\bar{\mathbf{Y}}; \boldsymbol{\theta}_x, \boldsymbol{\theta}_y, \mathbf{x}, \mathbf{y}\}. \tag{82}$$

9.3 *Coordinate Reversal*

In Sec. 2.2, we denoted the operation of reversing the coordinates of a 2D array $\mathbf{X}$ by $\mathcal{R}\{\mathbf{X}\}$. This can be represented by a row permutation followed by a column permutation, both using the antidiagonal matrix

$$\mathbf{P} = \begin{bmatrix} 0 & 0 & \cdots & 0 & 1 \\ 0 & 0 & \cdots & 1 & 0 \\ \vdots & \vdots & & \vdots & \vdots \\ 1 & 0 & \cdots & 0 & 0 \end{bmatrix}, \tag{83}$$

so that

$$\mathbf{Y} = \mathbf{PXP}. \tag{84}$$

Since $\mathbf{P}^{\dagger} = \mathbf{P}$, we find that the gradient propagation rule is

$$\overline{\mathbf{X}} = \mathbf{P}\overline{\mathbf{Y}}\mathbf{P}, \tag{85a}$$

$$= \mathcal{R}\{\overline{\mathbf{Y}}\}. \tag{85b}$$

Acknowledgments

This work was supported in part by the National Aeronautics and Space Administration under Grant No. 80NSSC19K0120 issued through the Strategic Astrophysics Technology/Technology Demonstration for Exoplanet Missions Program (SAT-TDEM; PI: R. Soummer). The authors thank the reviewers for their helpful comments, as well as Rémi Soummer, Marshall Perrin, Laurent Pueyo, Hari Subedi, Roser Juanola-Parramon, Neil Zimmerman, Matthew Bolcar, Lee Feinberg, and Michael McElwain for many illuminating discussions. Disclosures: The authors declare no conflicts of interest.

References

1. M. Perryman, *The Exoplanet Handbook*, 2nd ed., Cambridge University Press (2018).
2. The LUVOIR Study Team, "LUVOIR," tech. rep., National Aeronautics and Space Administration (2019).
3. The Habitable Exoplanet Observatory Study Team, "Habitable exoplanet observatory final report," tech. rep., Jet Propulsion Laboratory (2019).
4. J. Tumlinson et al., "The next great observatories: how can we get there?" *Bull. AAS* **51**(7), 10 (2019).
5. National Aeronautics and Space Administration, "Exoplanet exploration program 2019 technology plan appendix," (2019).
6. I. Y. Poberezhskiy et al., "Wide Field Infrared Survey Telescope (WFIRST): coronagraph instrument engineering design and operational concept," *Proc. SPIE* **11443**, 114431V (2020).
7. H. Zhou et al., "Roman CGI testbed HOWFSC modeling and validation," *Proc. SPIE* **11443**, 114431W (2020).
8. L. Pueyo et al., "Optimal dark hole generation via two deformable mirrors with stroke minimization," *Appl. Opt.* **48**, 6296–6312 (2009).
9. A. Give'on, "A unified formailism [sic] for high contrast imaging correction algorithms," *Proc. SPIE* **7440**, 74400D (2009).
10. A. J. Eldorado Riggs et al., "Fast linearized coronagraph optimizer (FALCO) I: a software toolbox for rapid coronagraphic design and wavefront correction," *Proc. SPIE* **10698**, 10698V (2018).
11. E. Cady et al., "Demonstration of high contrast with an obscured aperture with the WFIRST-AFTA shaped pupil coronagraph," *J. Astron. Telesc. Instrum. Syst.* **2**(1), 011004 (2016).

12. R. Soummer et al., "High-contrast imager for complex aperture telescopes (HiCAT): 5. First results with segmented-aperture coronagraph and wavefront control," *Proc. SPIE* **10698**, 106981O (2018).
13. A. Griewank and A. Walther, *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*, 2nd ed., Society for Industrial and Applied Mathematics (2008).
14. J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer-Verlag, New York (1999).
15. A. S. Jurling and J. R. Fienup, "Applications of algorithmic differentiation to phase retrieval algorithms," *J. Opt. Soc. Am. A* **31**(7), 1348–59 (2014).
16. A. G. Baydin et al., "Automatic differentiation in machine learning: a survey," *J. Mach. Learn. Res.* **18**(153), 1–43 (2018).
17. B. Paul et al., "High-order myopic coronagraphic phase diversity (COFFEE) for wave-front control in high-contrast imaging systems," *Opt. Express* **21**(26), 31751–31768 (2013).
18. A. Give'on, B. D. Kern, and S. Shaklan, "Pair-wise, deformable mirror, image plane-based diversity electric field estimation for high contrast coronagraphy," *Proc. SPIE* **8151**, 815110 (2011).
19. T. D. Groff and N. J. Kasdin, "Kalman filtering techniques for focal plane electric field estimation," *J. Opt. Soc. Am. A* **30**(1), 128–39 (2013).
20. A. J. E. Riggs, N. J. Kasdin, and T. D. Groff, "Recursive starlight and bias estimation for high-contrast imaging with an extended Kalman filter," *J. Astron. Telesc. Instrum. Syst.* **2**(1), 011017 (2016).
21. P. Baudoz et al., "The self-coherent camera: a new tool for planet detection," *Proc. Int. Astron. Union* **1**(C200), 553–558 (2005).
22. T. D. Groff et al., "Methods and limitations of focal plane sensing, estimation, and control in high-contrast imaging," *J. Astron. Telesc. Instrum. Syst.* **2**, 011009 (2016).
23. S. D. Will and J. R. Fienup, "Field stop diffraction and sampling effects in apodized pupil Lyot coronagraphs," *J. Opt. Soc. Am. A* **37**, 629–642 (2020).
24. J. W. Goodman, *Introduction to Fourier Optics*, 4th ed., W.H. Freeman & Company, (2017).
25. M. N'Diaye et al., "Apodized pupil Lyot coronagraphs for arbitrary apertures. V. Hybrid shaped pupil designs for imaging Earth-like planets with future space observatories," *Astrophys. J.* **818**, 163–171 (2016).
26. D. C. Moody, B. L. Gordon, and J. T. Trauger, "Design and demonstration of hybrid Lyot coronagraph masks for improved spectral bandwidth and throughput," *Proc. SPIE* **7010**, 70103P (2008).
27. R. Soummer et al., "Fast computation of Lyot-style coronagraph propagation," *Opt. Express* **15**, 15935–15951 (2007).
28. A. S. Jurling, M. D. Bergkoetter, and J. R. Fienup, "Techniques for arbitrary sampling in two-dimensional Fourier transforms," *J. Opt. Soc. Am. A* **A35**, 1784–1796 (2018).
29. N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, 2nd ed., Society for Industrial and Applied Mathematics (2002).
30. G. H. Golub and C. F. Van Loan, *Matrix Computations*, 4th ed., Johns Hopkins University Press (2013).
31. Wolfram Research, Inc., *Mathematica, Version 12.1*, Wolfram Research, Inc., Champaign, IL (2020).
32. M. Abadi et al., "TensorFlow: large-scale machine learning on heterogeneous systems," 2015, tensorflow.org.
33. J. Bradbury et al., "JAX: composable transformations of Python+NumPy programs," (2018).
34. R. Juanola-Parramon et al., "The LUVOIR extreme coronagraph for living planetary systems (ECLIPS) II. Performance evaluation, aberration sensitivity analysis and exoplanet detection simulations," *Proc. SPIE* **11117**, 1111702 (2019).
35. Q. Gong et al., "Optical design of the extreme coronagraph for living planetary systems instrument for the LUVOIR mission study," *J. Astron. Telesc. Instrum. Syst.* **5**(2), 025002 (2019).
36. P. Virtanen et al., "SciPy 1.0: fundamental algorithms for scientific computing in Python," *Nat. Methods* **17**, 261–272 (2020).

37. E. Sidick et al., "Optimizing the regularization in broadband wavefront control algorithm for WFIRST coronagraph," *Proc. SPIE* **10400**, 1040022 (2017).
38. H. Zhou et al., "Wavefront control performance modeling with WFIRST shaped pupil coronagraph testbed," *Proc. SPIE* **10400**, 1040005 (2017).
39. E. H. Por et al., "High Contrast Imaging for Python (HCIPy): an open-source adaptive optics and coronagraph simulator," *Proc. SPIE* **10703**, 1070342 (2018).
40. H. Sun, J. N. Kasdin, and R. Vanderbei, "Identification and adaptive control of a high-contrast focal plane wavefront correction system," *J. Astron. Telesc. Instrum. Syst.* **4**(4), 049006 (2018).
41. H. Sun et al., "High-contrast integral field spectrograph (HCIFS): multi-spectral wavefront control and reduced-dimensional system identification," *Opt. Express* **28**(15), 22412–22423 (2020).

Scott D. Will is a PhD candidate at the Institute of Optics, University of Rochester and with the Makidon Optics Laboratory at the Space Telescope Science Institute. He has also worked with the Optics Branch at NASA Goddard Space Flight Center as part of the NASA Pathways Program. He received his BS degree in electrical engineering from the University at Buffalo in 2015. His research interests include wavefront sensing and control, computational imaging, and astronomical instrumentation.

Tyler D. Groff received his BS degree in mechanical engineering and astrophysics from Tufts University and his PhD in mechanical and aerospace engineering from Princeton University under an NESSF fellowship. He was the lab manager of the Princeton High Contrast Imaging Laboratory and the CHARIS instrument at Subaru Telescope. He is currently the lead engineer at Goddard Space Flight Center for the spectroscopy and polarization modes on the Roman Space Telescope Coronagraph Instrument.

James R. Fienup received his AB from Holy Cross College and his MS and PhD degrees in applied physics from Stanford University, where he was a National Science Foundation graduate fellow. After performing research at ERIM, he became the Robert E. Hopkins Professor of Optics at the University of Rochester. He is a fellow of SPIE and OSA, a member of the National Academy of Engineering, and a recipient of SPIE's Rudolf Kingslake Medal and Prize.