

RANDOMIZED SKETCHING ALGORITHMS FOR LOW-MEMORY DYNAMIC OPTIMIZATION*

RAMCHANDRAN MUTHUKUMAR[†], DREW P. KOURI[‡], AND MADELEINE UDELL[†]

Abstract. This paper develops a novel limited-memory method to solve dynamic optimization problems. The memory requirements for such problems often present a major obstacle, particularly for problems with PDE constraints such as optimal flow control, full waveform inversion, and optical tomography. In these problems, PDE constraints uniquely determine the state of a physical system for a given control; the goal is to find the value of the control that minimizes an objective. While the control is often low dimensional, the state is typically more expensive to store.

This paper suggests using randomized matrix approximation to compress the state as it is generated and shows how to use the compressed state to reliably solve the original dynamic optimization problem. Concretely, the compressed state is used to compute approximate gradients and to apply the Hessian to vectors. The approximation error in these quantities is controlled by the target rank of the sketch. This approximate first- and second-order information can readily be used in any optimization algorithm. As an example, we develop a sketched trust-region method that adaptively chooses the target rank using *a posteriori* error information and provably converges to a stationary point of the original problem. Numerical experiments with the sketched trust-region method show promising performance on challenging problems such as the optimal control of an advection-reaction-diffusion equation and the optimal control of fluid flow past a cylinder.

Key words. PDE-constrained optimization; matrix approximation; randomized algorithm; single-pass algorithm; sketching; adaptivity; trust-region method; flow control; Navier–Stokes equations; adjoint equation

AMS subject classifications. 49M37, 49L20, 68W20, 90C30, 90C39, 93C20

1. Introduction. In this paper, we introduce novel low-memory methods to solve discrete-time dynamic optimization problems based on randomized matrix sketching. Such problems arise in many practical applications including full waveform inversion [21, 28, 33], optimal flow control [13, 23], financial engineering [17] and optical tomography [2, 18] to name a few. Let M be the dimension of the state space and m be the dimension of the control space. For many practical applications $M \gg m$. We consider the discrete-time dynamic optimization problem

$$(1.1) \quad \begin{aligned} & \underset{\mathbf{u}_n \in \mathbb{R}^M, \mathbf{z}_n \in \mathbb{R}^m}{\text{minimize}} && \sum_{n=1}^N f_n(\mathbf{u}_{n-1}, \mathbf{u}_n, \mathbf{z}_n) \\ & \text{subject to} && c_n(\mathbf{u}_{n-1}, \mathbf{u}_n, \mathbf{z}_n) = \mathbf{0}, \quad n = 1, \dots, N, \end{aligned}$$

where $\mathbf{z}_n \in \mathbb{R}^m$, $\mathbf{u}_n \in \mathbb{R}^M$ are the control actions and system states at the n^{th} time step respectively, $\mathbf{u}_0 \in \mathbb{R}^M$ is the provided initial state of the system, $f_n : \mathbb{R}^M \times \mathbb{R}^M \times \mathbb{R}^m \rightarrow \mathbb{R}$ is a “cost” or “objective” associated with the n^{th} state and control, and $c_n : \mathbb{R}^M \times \mathbb{R}^M \times \mathbb{R}^m \rightarrow \mathbb{R}^M$ is a constraint function that advances the state from \mathbf{u}_{n-1} into \mathbf{u}_n . One major application of Problem (1.1) is to optimize (a discretized version of) a continuous-time dynamical system. In this case, the form of c_n presented above corresponds to single-step time integration schemes. Other time stepping methods can also be handled with the approach described here. Additionally, our approach can handle dynamic optimization problems with static controls including, e.g., initial conditions, material parameters, and shape or topological designs. However, for simplicity we focus on problems of the form (1.1).

1.1. Memory versus computation: trade-offs. Memory limits often constrain numerical algorithms for (1.1). For example, suppose the objective and constraints are twice differentiable. To solve (1.1) using a traditional sequential quadratic programming algorithm, we must store the entire state trajectory $\{\mathbf{u}_n\}$, the Lagrange multipliers associated with each constraint function in (1.1), and the control trajectory $\{\mathbf{z}_n\}$: in total, a memory requirement of $N(2M + m)$ floating point numbers. For example, discretizations of full waveform inversion problems for petroleum exploration regularly result in state vectors of size $M = 64$ billion

*Submitted to the editors [DATE].

[†]Department of Operations Research and Information Engineering, Cornell University, Ithaca, NY 14853 (rm949@cornell.edu, udell@cornell.edu).

[‡]Optimization and Uncertainty Quantification, Sandia National Laboratories, Albuquerque, NM 87185 (dpkouri@sandia.gov).

42 with the number of time steps exceeding $N = 400,000$ [22]. In view of the onerous memory requirements of
 43 straightforward algorithms, algorithm designers must make hard choices to reduce the fidelity of the model
 44 or to repeat computation.

45 One can reduce the storage and computational complexity — at the cost of accuracy — using coarse
 46 spatial and temporal grids to model the problem. A more ambitious approach than coarsening is to solve
 47 (1.1) using a reduced-order model (ROM) [1, 8, 16]. However, ROMs are often tailored for specific dynamical
 48 systems and demand significant domain expertise. Moreover, ROMs can be difficult to implement in practice,
 49 requiring significant and often invasive modification of the simulation software. Naïvely implemented, ROMs
 50 are also a poor fit for optimization. For example, proper orthogonal decomposition ROMs are constructed
 51 using snapshots of the state trajectory $\{\mathbf{u}_n\}$, which depend on the current control trajectory $\{\mathbf{z}_n\}$. Therefore,
 52 as the control changes during optimization, the approximation quality of the ROM degrades. Adaptive ROM
 53 generation for optimization is an active research topic [9, 36].

54 An alternative approach substitutes computation for memory. Suppose the dynamic constraint in (1.1)
 55 uniquely determines the state given the control, and form the equivalent reduced optimization problem by
 56 eliminating the state “nuisance variable”. The optimization variable in this approach is simply the control
 57 $\{\mathbf{z}_n\}$: Nm floating point numbers. However, evaluating the objective function requires solving the dynamic
 58 constraint. Worse, evaluating the gradient of the objective function requires the solution of the backward-
 59 in-time adjoint equation [15]: to solve it, we must traverse the state trajectory backward, from the end to
 60 the beginning. Unfortunately, the state must generally be computed forward in time.

61 Checkpointing methods perform this backward pass without storing the full state [3, 10, 25, 32]. Instead,
 62 they store judiciously chosen snapshots of the state variables \mathbf{u}_n in memory or to hard disk. The state is
 63 then recomputed from these checkpoints to solve the adjoint equation. This procedure results in lower
 64 memory requirements, but drastically increases the cost of computing gradient information. For example,
 65 if we can store at most k state vectors in memory (i.e., kM floating point numbers) and we solve the
 66 dynamic optimization problem (1.1) using the checkpointing strategy described in [10] with k checkpoints,
 67 then Proposition 1 of [10] guarantees that the minimum number of additional state time steps required to
 68 perform the backward pass of the adjoint equation is

$$69 \quad w(N, k) := \tau N - \beta(k + 1, \tau - 1) \quad \text{where} \quad \beta(s, t) := \binom{s + t}{s}$$

70 and τ is the unique integer satisfying $\beta(k, \tau - 1) < N \leq \beta(k, \tau)$. This cost is compounded when higher-order
 71 derivatives are required.

72 **1.2. Randomized sketching for dynamic optimization.** In contrast to checkpointing methods,
 73 our sketching methods can achieve $\mathcal{O}(N)$ computation with $\mathcal{O}(N + M)$ storage, where the constant hidden
 74 by the big-O notation depends on the rank of the state matrix. Indeed, our methods solve the state equation
 75 only once at each iterate. The sketching method is simple and easy to integrate into existing codes: 1)
 76 compute the sketch while solving the state equation by forming a random projection, 2) reconstruct the
 77 approximate state via simple linear algebra, and 3) use the low-rank approximation in place of the state
 78 throughout the remainder of the computation; for example, to solve the adjoint equation and compute
 79 an approximate gradient. Under standard assumptions, we can quantify the effect of these approximate
 80 gradients on the quality of the approximate solution to the dynamic optimization problem (1.1). We also
 81 develop a trust-region algorithm to solve (1.1) that ensures convergence by adaptively choosing the rank.

82 **1.3. Outline.** We first introduce notation and describe the problem formulation. We then introduce a
 83 sketching methods for matrix approximation and analyze the error committed when solving (1.1) with a fixed-
 84 rank sketch. Subsequently, we introduce an adaptive-rank trust-region algorithm and discuss its convergence.
 85 We verify our assumptions for a class of optimal control problems constrained by linear parabolic partial
 86 differential equations (PDE). We provide numerical results for this class of problems as well as for a class of
 87 flow control problems for which the assumptions have not been verified.

88 **2. Problem formulation.** To begin, we introduce notation for the dynamic optimization problem.
 89 We consider the control vectors \mathbf{z}_n and the state vectors \mathbf{u}_n to be column vectors and collect the control and

90 state trajectories into the stacked column vectors

$$91 \quad \mathbf{Z} = \begin{bmatrix} \mathbf{z}_1 \\ \vdots \\ \mathbf{z}_N \end{bmatrix}, \quad \mathbf{z}_n \in \mathbb{R}^m \quad \forall n = 1, \dots, N, \quad \text{and} \quad \mathbf{U} = \begin{bmatrix} \mathbf{u}_1 \\ \vdots \\ \mathbf{u}_N \end{bmatrix}, \quad \mathbf{u}_n \in \mathbb{R}^M \quad \forall n = 1, \dots, N.$$

92 We denote the feasible sets of control and state vectors by $\mathfrak{Z} := \mathbb{R}^{mN}$ and $\mathfrak{U} := \mathbb{R}^{MN}$. Moreover, we consider
93 the family of coordinate projections $p_n : \mathbb{R}^M \times \mathbb{R}^m \rightarrow \mathbb{R}^M \times \mathbb{R}^m \times \mathbb{R}^m$ defined by

$$94 \quad p_1(\mathbf{U}, \mathbf{Z}) := (\mathbf{u}_0, \mathbf{u}_1, \mathbf{z}_1) \quad \text{and} \quad p_n(\mathbf{U}, \mathbf{Z}) := (\mathbf{u}_{n-1}, \mathbf{u}_n, \mathbf{z}_n), \quad n = 2, \dots, N,$$

95 where the initial state \mathbf{u}_0 is given. Other choices of the projection mappings $\{p_n\}$ result in different orderings
96 of the trajectory. These model, e.g., delays in the dynamics or different time stepping schemes. Throughout
97 the paper, all norms $\|\cdot\|$ are Euclidean (for matrices, Frobenius) unless stated otherwise. For later results,
98 we will require the weighted norms $\|\mathbf{v}\|_{\mathbf{A}}^2 = \mathbf{v}^\top \mathbf{A} \mathbf{v}$ for $\mathbf{v} \in \mathbb{R}^\ell$ where $\mathbf{A} \in \mathbb{R}^{\ell \times \ell}$ is a symmetric positive
99 definite matrix. In addition, we denote the singular values of a matrix $\mathbf{B} \in \mathbb{R}^{M \times N}$ by $\sigma_{\min}(\mathbf{B}) = \sigma_1(\mathbf{B}) \leq$
100 $\dots \leq \sigma_{\min(M,N)}(\mathbf{B}) = \sigma_{\max}(\mathbf{B})$.

101 Using this notation, we can represent the dynamic constraint and objective as the functions

$$102 \quad c(\mathbf{U}, \mathbf{Z}) := \begin{bmatrix} c_1 \circ p_1 \\ \vdots \\ c_N \circ p_N \end{bmatrix}(\mathbf{U}, \mathbf{Z}) \quad \text{and} \quad f(\mathbf{U}, \mathbf{Z}) := \sum_{n=1}^N f_n \circ p_n(\mathbf{U}, \mathbf{Z}),$$

103 where $c : \mathfrak{U} \times \mathfrak{Z} \rightarrow \mathfrak{U}$ and $f : \mathfrak{U} \times \mathfrak{Z} \rightarrow \mathbb{R}$ and we can rewrite the dynamic optimization problem (1.1) as

$$104 \quad (2.1) \quad \begin{aligned} & \underset{\mathbf{U} \in \mathfrak{U}, \mathbf{Z} \in \mathfrak{Z}}{\text{minimize}} && f(\mathbf{U}, \mathbf{Z}) \\ & \text{subject to} && c(\mathbf{U}, \mathbf{Z}) = \mathbf{0}. \end{aligned}$$

105 **2.1. Assumptions and the reduced problem.** Throughout this paper, we will assume that f and c
106 are continuously differentiable on $\mathfrak{U} \times \mathfrak{Z}$. In general, we denote by \mathbf{d}_i the partial derivative of a function with
107 respect to its i^{th} argument. We assume that the state Jacobian of the constraint, $\mathbf{d}_1 c(\mathbf{U}, \mathbf{Z})$ has a bounded
108 inverse for all controls $\mathbf{Z} \in \mathfrak{Z}$ and that there exists a control-to-state map $S : \mathfrak{Z} \rightarrow \mathfrak{U}$ such that for any control
109 $\mathbf{Z} \in \mathfrak{Z}$, $\bar{\mathbf{U}} := S(\mathbf{Z})$ is the unique state trajectory that satisfies the dynamic constraint,

$$110 \quad c(\bar{\mathbf{U}}, \mathbf{Z}) = 0.$$

111 Note that the unique state trajectory $\bar{\mathbf{U}} = S(\mathbf{Z})$ has the form

$$112 \quad S(\mathbf{Z}) := \begin{bmatrix} S_1(\mathbf{u}_0, \mathbf{z}_1) \\ S_2(S_1(\mathbf{u}_0, \mathbf{z}_1), \mathbf{z}_2) \\ \vdots \\ S_N(S_{N-1}(\dots, \mathbf{z}_{N-1}), \mathbf{z}_N) \end{bmatrix}$$

113 where $\bar{\mathbf{u}}_n = S_n(\bar{\mathbf{u}}_{n-1}, \mathbf{z}_n) \in \mathbb{R}^M$ denotes the unique solution to

$$114 \quad c_n(\bar{\mathbf{u}}_{n-1}, \bar{\mathbf{u}}_n, \mathbf{z}_n) = 0 \quad \forall n = 1, \dots, N.$$

115 Under these assumptions, the Implicit Function Theorem (cf. [15, Th. 1.41]) ensures that the operators S_n
116 and S are continuously differentiable. In addition, if c has continuous ℓ^{th} -order derivatives for $\ell \in \mathbb{N}$, then
117 S_n and S are ℓ^{th} -order continuously differentiable. Using the control-to-state map S , we can reformulate
118 (2.1) as the reduced dynamic optimization problem

$$119 \quad (2.2) \quad \underset{\mathbf{Z} \in \mathfrak{Z}}{\text{minimize}} \quad \{F(\mathbf{Z}) := f(S(\mathbf{Z}), \mathbf{Z})\}.$$

120 Our goal is to solve the reduced dynamic optimization problem (2.2) efficiently. This reduced formulation is
121 helpful when the problem size, and therefore the memory required to store the state, is large.

122 **2.2. Gradient computation and adjoints.** We focus on derivative-based optimization approaches
 123 to solving the dynamic optimization problem (2.2). These require computing first-order and (if possible)
 124 second-order derivative information. To compute the gradient of the reduced objective function F , we employ
 125 the adjoint approach [15], which results from an application of the chain rule to the implicitly defined reduced
 126 objective function F . In particular, the variation of F in the direction $\mathbf{V} \in \mathfrak{Z}$ is given by

$$127 \quad \langle \nabla F(\mathbf{Z}), \mathbf{V} \rangle_{\mathfrak{Z}} = \langle \mathbf{d}_1 f(S(\mathbf{Z}), \mathbf{Z}), S'(\mathbf{Z})\mathbf{V} \rangle_{\mathfrak{U}} + \langle \mathbf{d}_2 f(S(\mathbf{Z}), \mathbf{Z}), \mathbf{V} \rangle_{\mathfrak{Z}},$$

$$128 \quad = \langle S'(\mathbf{Z})^* \mathbf{d}_1 f(S(\mathbf{Z}), \mathbf{Z}) + \mathbf{d}_2 f(S(\mathbf{Z}), \mathbf{Z}), \mathbf{V} \rangle_{\mathfrak{Z}},$$

130 where $S'(\mathbf{Z})$ denotes the derivative of the control-to-state map S at \mathbf{Z} and $S'(\mathbf{Z})^*$ its adjoint. Here, $\langle \cdot, \cdot \rangle_{\mathfrak{Z}}$ and
 131 $\langle \cdot, \cdot \rangle_{\mathfrak{U}}$ denote inner products on \mathfrak{Z} and \mathfrak{U} , respectively. The Implicit Function Theorem ensure that $S'(\mathbf{Z})\mathbf{V}$
 132 satisfies the linear system of equations

$$133 \quad (2.3) \quad \mathbf{d}_1 c(S(\mathbf{Z}), \mathbf{Z})S'(\mathbf{Z})\mathbf{V} + \mathbf{d}_2 c(S(\mathbf{Z}), \mathbf{Z})\mathbf{V} = 0.$$

134 By the assumption that the state Jacobian of the constraint, $\mathbf{d}_1 c(S(\mathbf{Z}), \mathbf{Z})$, has a bounded inverse for all
 135 control $\mathbf{Z} \in \mathfrak{Z}$, we have that (2.3) has a unique solution given by

$$136 \quad S'(\mathbf{Z}) = -(\mathbf{d}_1 c(S(\mathbf{Z}), \mathbf{Z}))^{-1} \mathbf{d}_2 c(S(\mathbf{Z}), \mathbf{Z}).$$

137 Therefore, the adjoint of the derivative of the control-to-state map is given by

$$138 \quad S'(\mathbf{Z})^* = -(\mathbf{d}_2 c(S(\mathbf{Z}), \mathbf{Z}))^* (\mathbf{d}_1 c(S(\mathbf{Z}), \mathbf{Z}))^{-*}.$$

139 Substituting this expression into (2.2) yields the gradient

$$140 \quad \nabla F(\mathbf{Z}) = (\mathbf{d}_2 c(S(\mathbf{Z}), \mathbf{Z}))^* \bar{\boldsymbol{\Lambda}} + \mathbf{d}_2 f(S(\mathbf{Z}), \mathbf{Z}),$$

141 where *the adjoint*, $\bar{\boldsymbol{\Lambda}} = \boldsymbol{\Lambda}(\mathbf{Z}) \in \mathfrak{U}$, is the unique trajectory that solves the *adjoint equation*:

$$142 \quad (2.4) \quad (\mathbf{d}_1 c(S(\mathbf{Z}), \mathbf{Z}))^* \bar{\boldsymbol{\Lambda}} = -\mathbf{d}_1 f(S(\mathbf{Z}), \mathbf{Z}).$$

143 This discussion gives rise to [Algorithm 2.1](#) for computing gradients of the reduced objective function F .

Algorithm 2.1 Compute gradient using adjoints.

Input: Control \mathbf{Z}

Output: Gradient of reduced objective function $\nabla F(\mathbf{Z})$

- 1: **function** GRADIENT(\mathbf{Z})
 - 2: Solve the **state equation**, $c(\mathbf{U}, \mathbf{Z}) = \mathbf{0}$ and denote the solution $\bar{\mathbf{U}}$
 - 3: Solve the **adjoint equation**, $(\mathbf{d}_1 c(\bar{\mathbf{U}}, \mathbf{Z}))^* \boldsymbol{\Lambda} = -\mathbf{d}_1 f(\bar{\mathbf{U}}, \mathbf{Z})$ and denote the solution $\bar{\boldsymbol{\Lambda}}$
 - 4: Compute the gradient as $\nabla F(\mathbf{Z}) = \mathbf{d}_2 f(\bar{\mathbf{U}}, \mathbf{Z}) + (\mathbf{d}_2 c(\bar{\mathbf{U}}, \mathbf{Z}))^* \bar{\boldsymbol{\Lambda}}$
 - 5: **return** $\nabla F(\mathbf{Z})$
-

144 [Algorithm 2.1](#) hides the dynamic nature of the state and adjoint computations. In fact, we compute $\bar{\mathbf{U}}$
 145 forward in time starting from \mathbf{u}_1 to \mathbf{u}_N . In contrast, the adjoint equation is computed backward in time.
 146 To see this, express the adjoint equation in terms of the N components f_n and c_n :

$$147 \quad \mathbf{0} = \mathbf{d}_1 f(S(\mathbf{Z}), \mathbf{Z}) + (\mathbf{d}_1 c(S(\mathbf{Z}), \mathbf{Z}))^* \bar{\boldsymbol{\Lambda}} = \sum_{n=1}^N \mathbf{d}_1 (f_n \circ p_n)(S(\mathbf{Z}), \mathbf{Z}) + (\mathbf{d}_1 (c_n \circ p_n)(S(\mathbf{Z}), \mathbf{Z}))^* \bar{\boldsymbol{\Lambda}}_n.$$

149 We can calculate partial derivatives of $c_n \circ p_n$ and $f_n \circ p_n$ using the chain rule. To this end, we have that

$$150 \quad \mathbf{d}_1 (c_n \circ p_n)(\mathbf{U}, \mathbf{Z}) = \begin{bmatrix} \mathbf{0} \\ \vdots \\ \nabla_1 c_n(\mathbf{u}_{n-1}, \mathbf{u}_n, \mathbf{z}_n) \\ \nabla_2 c_n(\mathbf{u}_{n-1}, \mathbf{u}_n, \mathbf{z}_n) \\ \vdots \\ \mathbf{0} \end{bmatrix} \quad \text{and} \quad \mathbf{d}_1 (f_n \circ p_n)(\mathbf{U}, \mathbf{Z}) = \begin{bmatrix} \mathbf{0} \\ \vdots \\ \nabla_1 f_n(\mathbf{u}_{n-1}, \mathbf{u}_n, \mathbf{z}_n) \\ \nabla_2 f_n(\mathbf{u}_{n-1}, \mathbf{u}_n, \mathbf{z}_n) \\ \vdots \\ \mathbf{0} \end{bmatrix}.$$

151 Hence, the adjoint equation reduces to the following system of equations for $n = 1, \dots, N$,

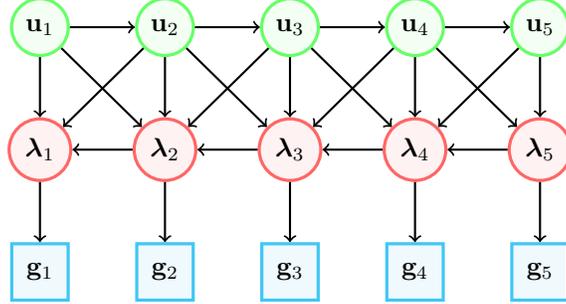
$$\begin{aligned}
 152 \quad & (d_2 c_N(\bar{\mathbf{u}}_{N-1}, \bar{\mathbf{u}}_N, \mathbf{z}_N))^* \boldsymbol{\lambda}_N = -d_2 f_N(\bar{\mathbf{u}}_{N-1}, \bar{\mathbf{u}}_N, \mathbf{z}_N), \\
 153 \quad & (d_2 c_n(\bar{\mathbf{u}}_{n-1}, \bar{\mathbf{u}}_n, \mathbf{z}_n))^* \boldsymbol{\lambda}_n = -d_2 f_n(\bar{\mathbf{u}}_{n-1}, \bar{\mathbf{u}}_n, \mathbf{z}_n) - d_1 f_{n+1}(\bar{\mathbf{u}}_n, \bar{\mathbf{u}}_{n+1}, \mathbf{z}_{n+1}) \\
 154 \quad & \quad \quad \quad - d_1 c_{n+1}(\bar{\mathbf{u}}_n, \bar{\mathbf{u}}_{n+1}, \mathbf{z}_{n+1})^* \boldsymbol{\lambda}_{n+1},
 \end{aligned}$$

156 where $\bar{\mathbf{u}}_n = S_n(\bar{\mathbf{u}}_{n-1}, \mathbf{z}_n)$ for $n = 1, \dots, N$. Here, information required for the solve flows backward in
 157 time from $\bar{\boldsymbol{\lambda}}_N$ to $\bar{\boldsymbol{\lambda}}_1$: in general, computing $\bar{\boldsymbol{\lambda}}_n$ requires the state vectors $\bar{\mathbf{u}}_{n-1}$, $\bar{\mathbf{u}}_n$ and $\bar{\mathbf{u}}_{n+1}$. The most
 158 straightforward computational approach is to solve the state equation and store the full state trajectory
 159 before computing the adjoint. The adjoint vectors $\bar{\boldsymbol{\lambda}}_n$ are used to form the gradient vector g_n at the n^{th}
 160 time step as

$$161 \quad g_n = d_3 f_n(\mathbf{u}_{n-1}, \mathbf{u}_n, \mathbf{z}_n) + (d_3 c_n(\mathbf{u}_{n-1}, \mathbf{u}_n, \mathbf{z}_n))^* \boldsymbol{\lambda}_n.$$

162 The data dependency of the gradient computation is shown in [Figure 1](#) for $N = 5$. The dependencies of the green nodes (*state*) flow forward, while the dependencies of the red nodes (*adjoint*) flow backward. The
 163 target nodes of the computation are the gradient vectors g_n . Both the state $\bar{\mathbf{U}}$ and adjoint $\bar{\boldsymbol{\Lambda}}$ are intermediate

Figure 1: Data dependency between computations.



164 variables used to compute the gradient $\nabla F(\mathbf{Z})$, and both require MN storage. The control requires only
 165 mN storage, which is often much smaller in practical applications, i.e., $M \gg m$.
 166

167 **3. Low-memory matrix approximation.** Our method forms a low-memory approximation to the
 168 state matrix in order to solve the dynamic optimization problem without storing or recomputing the state
 169 matrix. In this section, we describe this approximation in detail. Given a fixed storage budget, in a single
 170 pass column-by-column over the matrix, the method collects information about the matrix from which the
 171 matrix can be accurately reconstructed. This information is called a *sketch* of the matrix. The approach
 172 we adopt in this paper uses a random projection of the matrix to compute the sketch. This approach has
 173 been studied extensively in the numerical analysis and theoretical computer science communities, and many
 174 different methods are available [5, 6, 12, 24, 29, 30, 31, 34, 35]. When the state space itself has tensor
 175 product structure, a tensor sketch that respects that structure can further reduce memory requirements [26].

176 Consider a matrix $\mathbf{A} \in \mathbb{R}^{M \times N}$ and a target rank parameter r . Each of these methods produces a
 177 low-rank matrix approximation $\hat{\mathbf{A}}$ that is (in expectation) not much farther from \mathbf{A} than the best rank- r
 178 approximation, using $\mathcal{O}(r(M + N))$ storage. For concreteness (and for use in our numerical experiments),
 179 we describe the method developed in [31].

180 Define the sketch parameters $r \leq k \leq s$. The quality of the approximation, and also the storage required
 181 for the sketch, increases with these parameters. In this paper, we choose $k := 2r + 1$ and $s := 2k + 1$, and
 182 adjust the target rank parameter r to obtain satisfactory performance. To define the sketch, fix four random
 183 linear dimension reduction maps (DRMs) with iid standard normal entries:

$$\begin{aligned}
 184 \quad & \Upsilon \in \mathbb{R}^{k \times M} \quad \text{and} \quad \Omega \in \mathbb{R}^{k \times N}; \\
 & \Phi \in \mathbb{R}^{s \times M} \quad \text{and} \quad \Psi \in \mathbb{R}^{s \times N}.
 \end{aligned}$$

185 Note that other random ensembles work similarly; see [31]. The sketch of the target matrix \mathbf{A} consists of

$$\begin{aligned}
 186 \quad & \mathbf{X} := \mathbf{\Upsilon} \mathbf{A} \in \mathbb{R}^{k \times N}, & \text{the range sketch;} \\
 187 \quad & \mathbf{Y} := \mathbf{A} \mathbf{\Omega}^* \in \mathbb{R}^{M \times k}, & \text{the co-range sketch;} \\
 188 \quad & \mathbf{Z} := \mathbf{\Phi} \mathbf{A} \mathbf{\Psi}^* \in \mathbb{R}^{s \times s}, & \text{the core sketch.}
 \end{aligned}$$

190 Roughly speaking, the range sketch \mathbf{X} captures the row space (top left singular vectors) of \mathbf{A} ; the co-range
 191 sketch \mathbf{Y} captures the column space (top right singular vectors); and the core sketch \mathbf{Z} captures their
 192 interactions (singular values). Linearity of the sketch allows us to compute it without storing the full matrix
 193 \mathbf{A} . Suppose $\mathbf{A} = [\mathbf{a}_1 \cdots \mathbf{a}_N]$ is presented column by column. Then, we can compute the range sketch
 194 $\mathbf{X} = \mathbf{X}^{(N)}$ by the recursion

$$195 \quad (3.1) \quad \mathbf{X}^{(0)} = \mathbf{0}, \quad \mathbf{X}^{(i)} = \mathbf{X}^{(i-1)} + \mathbf{\Upsilon} \mathbf{a}_i \mathbf{e}_i^\top, \quad i = 1, \dots, N,$$

196 where \mathbf{e}_i is the i^{th} unit vector, and similarly for the co-range sketch \mathbf{Y} and core sketch \mathbf{Z} .

197 *Sketch object.* We use $\{\mathbf{A}\}_r$ to denote an object of the sketch class, which contains the sketch parameters
 198 k, s , the dimension reduction maps $\mathbf{\Upsilon}, \mathbf{\Omega}, \mathbf{\Phi}, \mathbf{\Psi}$, and the range, co-range, and core sketches $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$.

199 *Storage.* The sketch matrices \mathbf{X}, \mathbf{Y} , and \mathbf{Z} can be stored using $k(M + N) + s^2$ floating point numbers.
 200 Hence the memory required to store a sketch object with target rank parameter r is $\mathcal{O}(r(M + N) + r^2)$. When
 201 storage is limited, the DRMs can be regenerated on the fly from a random seed or generated from a random
 202 ensemble with lower storage requirements [26, 27], so we omit the DRMs from our storage calculation.

203 **3.1. Reconstruction.** We can reconstruct a low-rank approximation from the sketch. To reconstruct,
 204 compute the thin QR factorizations of \mathbf{X}^* and \mathbf{Y} ,

$$\begin{aligned}
 205 \quad (3.2) \quad & \mathbf{X}^* =: \mathbf{P} \mathbf{R}_1 \quad \text{where } \mathbf{P} \in \mathbb{R}^{N \times k}; \\
 & \mathbf{Y} =: \mathbf{Q} \mathbf{R}_2 \quad \text{where } \mathbf{Q} \in \mathbb{R}^{M \times k}.
 \end{aligned}$$

206 Use the core sketch \mathbf{Z} to compute a core approximation by solving two small least-squares problems

$$207 \quad (3.3) \quad \mathbf{C} := (\mathbf{\Phi} \mathbf{Q})^\dagger \mathbf{Z} ((\mathbf{\Psi} \mathbf{P})^\dagger)^* \in \mathbb{R}^{k \times k}.$$

208 Then compute a rank- k approximation of the target matrix \mathbf{A} as

$$209 \quad (3.4) \quad \{\{\mathbf{A}\}\}_r := \mathbf{Q} \mathbf{C} \mathbf{P}^*.$$

210 This approximation can be truncated to rank r by replacing $\mathbf{C} \in \mathbb{R}^{k \times k}$ by its best rank- r approximation.

211 For use in the dynamic optimization problem, after reconstruction we store the low-rank factors of the
 212 approximation, $\mathbf{Q} \in \mathbb{R}^{M \times k}$ and $\mathbf{W} = \mathbf{C} \mathbf{P}^* \in \mathbb{R}^{k \times N}$, in the sketch object $\{\mathbf{A}\}_r$. (To reduce storage further,
 213 one can overwrite \mathbf{X} and \mathbf{Y} with \mathbf{Q} and \mathbf{W}). From these, we can reconstruct the j^{th} column (the state at the
 214 j^{th} time step) as needed, via $\{\{\mathbf{A}\}\}_r[:, j] = \mathbf{Q} \mathbf{W}[:, j]$. Each of these operations uses storage proportional to
 215 $k(M + N)$, so the total storage complexity to approximate $\mathbf{A} \in \mathbb{R}^{M \times N}$ (in factored form) is $\mathcal{O}(k(M + N))$.
 216 We summarize the sketch class and its methods in [Algorithm A.1](#) in [Appendix A](#).

217 Sketching provides a tractable way to control the relative error in approximation by varying the target
 218 rank parameter r , since the error tends to zero as r increases. We state the reconstruction error bound estab-
 219 lished in [31] and a useful lemma that shows any fixed error tolerance can be achieved by an approximation
 220 with sufficiently large r . For use in these results, the j^{th} tail energy of a matrix \mathbf{A} is defined by

$$221 \quad \tau_j(\mathbf{A}) := \min_{\text{Rank}(\mathbf{B}) < j} \|\mathbf{A} - \mathbf{B}\| = \left(\sum_{i \geq j} \sigma_i^2(\mathbf{A}) \right)^{\frac{1}{2}}.$$

222

223 **THEOREM 3.1** (Reconstruction Error [31]). *Let $\mathbf{A} \in \mathbb{R}^{M \times N}$ be a matrix and let r be the target rank*
 224 *parameter. Choose sketch parameters $k = 2r + 1$ and $s = 2k + 1$. Compute range, co-range and core sketches*
 225 *($\mathbf{X}, \mathbf{Y}, \mathbf{Z}$) according to (3.1). The low-rank approximation $\{\{\mathbf{A}\}\}_r$ computed in (3.2)-(3.4) satisfies*

$$226 \quad \mathbf{E} \|\mathbf{A} - \{\{\mathbf{A}\}\}_r\| \leq \sqrt{6} \cdot \tau_{r+1}(\mathbf{A}).$$

227 This result shows that the rank- k approximation to \mathbf{A} computed by sketching is only a constant factor
 228 farther from \mathbf{A} than the best rank- r approximation. It is also possible to obtain an unbiased estimate of the
 229 error in approximation, $\|\mathbf{A} - \{\{\mathbf{A}\}\}_r\|$, in one streaming pass over the target matrix \mathbf{A} ; see [31] for details.
 230 In the following sections, we state several bounds on the expected error of various quantities. Notice that
 231 these all yield high probability bounds using the following result.

232 **LEMMA 3.2.** *For any $\mathbf{A} \in \mathbb{R}^{M \times N}$ and for all $\delta, \epsilon > 0$, there exists a rank $r(\delta, \epsilon)$ such that for any*
 233 *$r \geq r(\delta, \epsilon)$, the sketching approximation error is bounded by ϵ with probability δ :*

$$234 \quad \text{Prob}(\|\mathbf{A} - \{\{\mathbf{A}\}\}_r\| \geq \epsilon) \leq \delta.$$

235 *Proof.* Observe that for any target matrix $\mathbf{A} \in \mathbb{R}^{M \times N}$, $\tau_1(\mathbf{A}) \geq \dots \geq \tau_{\min\{M, N\}}(\mathbf{A}) = 0$. Therefore
 236 for any δ and ϵ , there is some rank r such that $\mathbf{E} \|\mathbf{A} - \{\{\mathbf{A}\}\}_r\| \leq \delta\epsilon$. Combining this fact with Markov's
 237 inequality yields the desired bound. \square

238 **4. Randomized sketching for dynamic optimization.** This section presents our limited-memory
 239 algorithm to solve the dynamic optimization problem (2.2). Any first-order optimization method relies on
 240 the gradient of the objective function, so we begin with a discussion of how to compute a limited-memory
 241 approximate gradient in Subsection 4.1. We also discuss how the same approach can be extended to apply
 242 the Hessian to a vector using limited memory. This enables usage of second-order methods. We next
 243 quantify the error in the approximate gradient. To quantify this error, we rely on regularity assumptions
 244 detailed in Subsection 4.2. This analysis undergirds our results on the optimization algorithms presented
 245 in the next two subsections. In Subsection 4.3, we present our first approach that considers computing the
 246 gradient using a fixed-rank sketch. This method has the advantage that it uses a fixed storage budget.
 247 However, for this method to work well, the state corresponding to any control must be well approximated
 248 by a fixed-rank sketch whose rank is known in advance. In Subsection 4.4, we present our second approach,
 249 an adaptive method that updates the sketch rank to control the error in the gradient. We obtain a provably
 250 convergent optimization method by using this adaptive approach to compute the gradient within a trust-
 251 region algorithm. Unlike the fixed-rank method this approach does not require a rank estimate *a priori*.
 252 However, this approach has the disadvantage that the storage budget required is dictated by the progress of
 253 the optimization algorithm and is not known *a priori*.

254 **4.1. Computing first- and second-order information with limited memory.** To compute the
 255 gradient and to apply the Hessian with limited memory, we can sketch the state while solving the state
 256 equation, $c(\mathbf{U}, \mathbf{Z}) = \mathbf{0}$. Upon solving the adjoint equation, we reconstruct from the sketch to compute an
 257 approximate state. This allows us to compute an approximate gradient based on the approximate state.

258 **4.1.1. Solving the state equation.** Fix the target rank parameter r . To set notation, denote
 259 by $\text{mat}(\mathbf{U}, M, N)$ the state vectors at different time steps \mathbf{u}_n collected into a matrix, $\text{mat}(\mathbf{U}, M, N) :=$
 260 $[\mathbf{u}_1 \cdots \mathbf{u}_N] \in \mathbb{R}^{M \times N}$. With some abuse of notation, we define the approximate state to be the low-rank
 261 approximation reconstructed via sketching the state matrix $\{\{\mathbf{U}\}\}_r := \{\{\text{mat}(\mathbf{U}, M, N)\}\}_r$. Since the state
 262 matrix is computed forward in time starting from \mathbf{u}_1 to \mathbf{u}_N , we can simultaneously update the sketch
 263 matrices \mathbf{X}, \mathbf{Y} and \mathbf{Z} using the COLUMNUPDATE! function of the sketch class Algorithm A.1. The reduced
 264 objective function can be simultaneously exactly evaluated in this procedure. This method is presented as
 265 Algorithm 4.1. Here the notation FUNCTION!() denotes a method that can modify its arguments or associated
 266 class. In the context of the approximate state, we shall refer to $c(\mathbf{U}, \mathbf{Z})$ as the state residual.

267 **4.1.2. Computing an approximate gradient from the sketched state.** Recall that variables
 268 with bars denote exact solutions to the state or adjoint equation, while variables with hats are approximate
 269 solutions: fixing the control \mathbf{Z} , the true adjoint $\bar{\Lambda}$ solves the adjoint equation (2.4) at the true state $\bar{\mathbf{U}}$, while
 270 the approximate adjoint $\hat{\Lambda}_r$ solves the adjoint equation (2.4) at the sketched state $\hat{\mathbf{U}}_r = \{\{\bar{\mathbf{U}}\}\}_r$:

$$271 \quad (d_1 c(\hat{\mathbf{U}}_r, \mathbf{Z}))^* \hat{\Lambda}_r = -d_1 f(\hat{\mathbf{U}}_r, \mathbf{Z}).$$

272 By analogy with the state residual $c(\mathbf{U}, \mathbf{Z})$, define the adjoint residual at $(\Lambda, \mathbf{U}, \mathbf{Z}) \in \mathfrak{U} \times \mathfrak{U} \times \mathfrak{Z}$ as

$$273 \quad h(\Lambda, \mathbf{U}, \mathbf{Z}) := d_1 f(\mathbf{U}, \mathbf{Z}) + (d_1 c(\mathbf{U}, \mathbf{Z}))^* \Lambda.$$

Algorithm 4.1 Solve state equation and compute exact objective function value.

Input: A control iterate $\mathbf{Z} \in \mathbb{R}^{m \times N}$, sketch object $\{\mathbf{U}\}_r$ for state and sketch rank parameter $r \leq \min\{M, N\}$

Output: Updated sketch object $\{\mathbf{U}\}_r$ and reduced objective function value $F(\mathbf{Z})$

Storage: $\mathcal{O}(r(M + N) + mN)$

```

1: function SOLVESTATE! ( $\{\mathbf{U}\}_r, \mathbf{Z}$ )
2:    $(\mathbf{u}_{\text{old}}, F) \leftarrow (\mathbf{u}_0, 0)$ 
3:   for  $n \leftarrow 1$  to  $N$  do
4:     Solve  $c_n(\mathbf{u}_{\text{old}}, \mathbf{u}_{\text{new}}, \mathbf{z}_n) = 0$  for  $\mathbf{u}_{\text{new}}$            Solve  $n^{\text{th}}$  state equation
5:      $F \leftarrow F + f_n(\mathbf{u}_{\text{old}}, \mathbf{u}_{\text{new}}, \mathbf{z}_n)$            Update objective function value
6:      $\{\mathbf{U}\}_r.$ COLUMNUPDATE! ( $\mathbf{u}_{\text{new}}, n$ )           Update sketch with  $n^{\text{th}}$  column of state
7:      $\mathbf{u}_{\text{old}} \leftarrow \mathbf{u}_{\text{new}}$ 
8:   return  $F$ 

```

274 The adjoint residual evaluated at arguments $(\mathbf{\Lambda}, \mathbf{U}, \mathbf{Z})$ is zero when $\mathbf{\Lambda}$ solves the adjoint equation (2.4) for
275 any control \mathbf{Z} and state \mathbf{U} . Consider in particular the special cases of this equality using the true state
276 $h(\bar{\mathbf{\Lambda}}, \bar{\mathbf{U}}, \mathbf{Z}) = 0$ and using the sketched state $h(\hat{\mathbf{\Lambda}}_r, \hat{\mathbf{U}}_r, \mathbf{Z}) = 0$. For arbitrary $\mathbf{\Lambda}$, \mathbf{U} and \mathbf{Z} , we define the map
277 $g : \mathfrak{U} \times \mathfrak{U} \times \mathfrak{Z} \rightarrow \mathfrak{Z}$ as

$$278 \quad g(\mathbf{\Lambda}, \mathbf{U}, \mathbf{Z}) := \mathbf{d}_2 f(\mathbf{U}, \mathbf{Z}) + (\mathbf{d}_2 c(\mathbf{U}, \mathbf{Z}))^* \mathbf{\Lambda}.$$

279 This function computes the gradient at \mathbf{Z} when we use the true state $\bar{\mathbf{U}}$ and true adjoint $\bar{\mathbf{\Lambda}}$:

$$280 \quad g(\bar{\mathbf{\Lambda}}, \bar{\mathbf{U}}, \mathbf{Z}) = \mathbf{d}_2 f(\bar{\mathbf{U}}, \mathbf{Z}) + (\mathbf{d}_2 c(\bar{\mathbf{U}}, \mathbf{Z}))^* \bar{\mathbf{\Lambda}} = \nabla F(\mathbf{Z}).$$

281 On the other hand, the function g can approximate the gradient using the sketched variables as

$$282 \quad g_r(\mathbf{Z}) := g(\hat{\mathbf{\Lambda}}_r, \hat{\mathbf{U}}_r, \mathbf{Z}).$$

283 **Algorithm 4.2** describes a backward-in-time procedure for computing a limited-memory approximate gra-
284 dient $g_r(\mathbf{Z})$ from the sketched state $\hat{\mathbf{U}}_r$. One can also use the sketching method to compute second-order
285 information with limited storage. Matrix-free second-order optimization methods such as Krylov-Newton
286 methods require only the application of the Hessian to a vector: for an arbitrary vector $\mathbf{V} \in \mathfrak{Z}$, we must com-
287 pute $\nabla^2 F(\mathbf{Z})\mathbf{V}$. Using the chain rule, we can apply the Hessian to \mathbf{V} by first computing the *state sensitivity*
288 $\bar{\mathbf{W}} := S'(\mathbf{Z})\mathbf{V} \in \mathfrak{U}$ (i.e., the solution to (2.3)) and then the *adjoint sensitivity* $\bar{\mathbf{P}} := \Lambda'(\mathbf{Z})\mathbf{V} \in \mathfrak{U}$. The state
289 sensitivity is computed forward in time, while the adjoint sensitivity is computed backward in time. We can
290 control the storage footprint for these operations by sketching the state, adjoint, and state sensitivity. **Algo-**
291 **rithms A.3 to A.6** detail the steps required to apply the Hessian with storage $\mathcal{O}((r_1 + r_2 + r_3)(M + N) + mN)$
292 for rank parameters $r_1, r_2, r_3 \leq \min\{M, N\}$.

293 **4.2. Regularity assumptions.** Throughout the remainder of the paper, we make the following regu-
294 larity assumptions. These assumptions allow us to develop provable guarantees on the optimization error in
295 the algorithms presented in the next two subsections. These conditions are adapted from [36].

296 **ASSUMPTION 1.** Assume that the following conditions hold for the dynamic optimization problem (2.2).

- 297 1. The set of states corresponding to controls in an open and bounded set $\mathfrak{Z}_0 \subseteq \mathfrak{Z}$ is bounded: there
298 exists $\mathfrak{U}_0 \subset \mathfrak{U}$ open and bounded such that $\{\mathbf{U} \in \mathfrak{U} \mid \exists \mathbf{Z} \in \mathfrak{Z}_0, c(\mathbf{U}, \mathbf{Z}) = \mathbf{0}\} \subseteq \mathfrak{U}_0$.
- 299 2. There exist singular value thresholds $0 < \sigma_0 \leq \sigma_1 < \infty$ such that for any $\mathbf{U} \in \mathfrak{U}_0$ and $\mathbf{Z} \in \mathfrak{Z}_0$, the
300 state Jacobian matrix $\mathbf{d}_1 c(\mathbf{U}, \mathbf{Z})$ satisfies $\sigma_0 \leq \sigma_{\min}(\mathbf{d}_1 c(\mathbf{U}, \mathbf{Z})) \leq \sigma_{\max}(\mathbf{d}_1 c(\mathbf{U}, \mathbf{Z})) \leq \sigma_1$.
- 301 3. The following functions are Lipschitz continuous on $\mathfrak{U}_0 \times \mathfrak{Z}_0$ with respect to their first arguments:
302 (a) The state Jacobian of the constraint, $\mathbf{d}_1 c(\mathbf{U}, \mathbf{Z})$;
303 (b) The control Jacobian of the constraint, $\mathbf{d}_2 c(\mathbf{U}, \mathbf{Z})$;
304 (c) The state gradient of the objective function, $\mathbf{d}_1 f(\mathbf{U}, \mathbf{Z})$;
305 (d) The control gradient of the objective function, $\mathbf{d}_2 f(\mathbf{U}, \mathbf{Z})$.

Algorithm 4.2 Compute gradient from sketched state.

Input: A control iterate $\mathbf{Z} \in \mathbb{R}^{m \times N}$ and sketch object $\{\mathbf{U}\}_r$ for state

Output: Approximate gradient $g = g_r(\mathbf{Z}) \approx \nabla F(\mathbf{Z})$

Storage: $\mathcal{O}(r(M + N) + mN)$

- 1: **function** GRADIENT($\{\mathbf{U}\}_r, \mathbf{Z}$)
- 2: $(\mathbf{u}_{\text{curr}}, \mathbf{u}_{\text{next}}) \leftarrow (\{\mathbf{U}\}_r.\text{COLUMN}(N - 1), \{\mathbf{U}\}_r.\text{COLUMN}(N))$
- 3: Solve the adjoint equation at index N for $\boldsymbol{\lambda}_{\text{next}}$,

$$(\mathbf{d}_2 c_N(\mathbf{u}_{\text{curr}}, \mathbf{u}_{\text{next}}, \mathbf{z}_N))^* \boldsymbol{\lambda}_{\text{next}} = \mathbf{d}_2 f_N(\mathbf{u}_{\text{curr}}, \mathbf{u}_{\text{next}}, \mathbf{z}_N)$$

- 4: Compute gradient at index N ,

$$g_N \leftarrow \mathbf{d}_3 f_N(\mathbf{u}_{\text{curr}}, \mathbf{u}_{\text{next}}, \mathbf{z}_N) + (\mathbf{d}_3 c_N(\mathbf{u}_{\text{curr}}, \mathbf{u}_{\text{next}}, \mathbf{z}_N))^* \boldsymbol{\lambda}_{\text{next}}$$

- 5: **for** $n \leftarrow N - 1$ to 1 **do**
- 6: **if** $n = 1$ **then**
- 7: $\mathbf{u}_{\text{prev}} \leftarrow \mathbf{u}_0$
- 8: **else**
- 9: $\mathbf{u}_{\text{prev}} \leftarrow \{\mathbf{U}\}_r.\text{COLUMN}(n - 1)$
- 10: Solve the adjoint equation at index n for $\boldsymbol{\lambda}_{\text{curr}}$,

$$\begin{aligned} (\mathbf{d}_2 c_n(\mathbf{u}_{\text{prev}}, \mathbf{u}_{\text{curr}}, \mathbf{z}_n))^* \boldsymbol{\lambda}_{\text{curr}} &= \mathbf{d}_2 f_n(\mathbf{u}_{\text{prev}}, \mathbf{u}_{\text{curr}}, \mathbf{z}_n) + \mathbf{d}_1 f_{n+1}(\mathbf{u}_{\text{curr}}, \mathbf{u}_{\text{next}}, \mathbf{z}_{n+1}) \\ &\quad - (\mathbf{d}_1 c_{n+1}(\mathbf{u}_{\text{curr}}, \mathbf{u}_{\text{next}}, \mathbf{z}_{n+1}))^* \boldsymbol{\lambda}_{\text{next}} \end{aligned}$$

- 11: Compute gradient at index n ,

$$g_n \leftarrow \mathbf{d}_3 f_n(\mathbf{u}_{\text{prev}}, \mathbf{u}_{\text{curr}}, \mathbf{z}_n) + (\mathbf{d}_3 c_n(\mathbf{u}_{\text{prev}}, \mathbf{u}_{\text{curr}}, \mathbf{z}_n))^* \boldsymbol{\lambda}_{\text{curr}}$$

- 12: $(\mathbf{u}_{\text{next}}, \mathbf{u}_{\text{curr}}, \boldsymbol{\lambda}_{\text{next}}) \leftarrow (\mathbf{u}_{\text{curr}}, \mathbf{u}_{\text{prev}}, \boldsymbol{\lambda}_{\text{curr}})$
 - 13: **return** $g = [g_1, \dots, g_N]$
-

306 These assumptions are often satisfied in applications. For example, we show in [Section 5](#) that they hold for
307 optimal control problems with parabolic PDE constraints.

308 **4.3. A fixed-rank approach.** A natural limited-memory algorithm to solve the dynamic optimization
309 problem (2.2) is to fix the sketch rank parameter r used to compute the gradient *a priori*. [Algorithm 4.3](#)
310 shows the steps involved in this method. The resulting approximate gradient can be used inside any first-
311 order optimization method to (approximately) solve the dynamic optimization problem (2.2). In this section,
312 we analyze the error of the resulting method and prove a useful stopping criterion under [Assumption 1](#).

Algorithm 4.3 Fixed-rank algorithm for approximate gradient.

Input: A control iterate $\mathbf{Z} \in \mathbb{R}^{m \times N}$ and rank parameter $r \leq \min\{M, N\}$.

Output: Approximate gradient $g_r(\mathbf{Z})$

Storage: $\mathcal{O}(r(M + N) + mN)$

- 1: **function** FIXEDRANKGRADIENT(\mathbf{Z})
 - 2: $\{\mathbf{U}\}_r \leftarrow \text{SKETCH}(M, N, \text{rank} = r)$ Initialize sketch object for state
 - 3: $F \leftarrow \text{SOLVESTATE!}(\{\mathbf{U}\}_r, \mathbf{Z})$ Solve state equation
 - 4: $\{\mathbf{U}\}_r.\text{RECONSTRUCT!}()$ Reconstruct low rank factors
 - 5: $g \leftarrow \text{Gradient}(\{\mathbf{U}\}_r, \mathbf{Z})$ Compute gradient
 - 6: **return** g
-

313 PROPOSITION 4.1. *Suppose [Assumption 1](#) holds for a bounded control set \mathfrak{Z}_0 . Then there exists $\kappa_0, \kappa_1 >$*

314 0 such that the error in the state satisfies

$$315 \quad (4.1) \quad \kappa_0 \|\mathbf{U} - \bar{\mathbf{U}}\| \leq \|c(\mathbf{U}, \mathbf{Z})\| \leq \kappa_1 \|\mathbf{U} - \bar{\mathbf{U}}\| \quad \forall \mathbf{U} \in \mathfrak{U}_0, \mathbf{Z} \in \mathfrak{Z}_0,$$

316 where $\mathfrak{U}_0 \subseteq \mathfrak{U}$ is defined in [Assumption 1.1](#). Furthermore, the error in the adjoint is controlled by the adjoint
317 residual together with the state residual: for some $\kappa_2 > 0$ and $\kappa_3 > 0$,

$$318 \quad (4.2) \quad \|\boldsymbol{\Lambda} - \bar{\boldsymbol{\Lambda}}\| \leq \kappa_2 \|c(\mathbf{U}, \mathbf{Z})\| + \kappa_3 \|h(\boldsymbol{\Lambda}, \mathbf{U}, \mathbf{Z})\| \quad \forall \mathbf{U}, \boldsymbol{\Lambda} \in \mathfrak{U}_0, \quad \forall \mathbf{Z} \in \mathfrak{Z}_0.$$

319 Hence, the error in the gradient is controlled by the adjoint and state residuals: for some $\kappa_4 > 0$ and $\kappa_5 > 0$,

$$320 \quad (4.3) \quad \|g(\boldsymbol{\Lambda}, \mathbf{U}, \mathbf{Z}) - g(\bar{\boldsymbol{\Lambda}}, \bar{\mathbf{U}}, \mathbf{Z})\| = \|g(\boldsymbol{\Lambda}, \mathbf{U}, \mathbf{Z}) - \nabla F(\mathbf{Z})\| \leq \kappa_4 \|c(\mathbf{U}, \mathbf{Z})\| + \kappa_5 \|h(\boldsymbol{\Lambda}, \mathbf{U}, \mathbf{Z})\|.$$

321 *Remark 4.2.* All constants in [Proposition 4.1](#) depend only on finite quantities defined by [Assumption 1](#).

322 *Proof.* The proof of this result is similar to the proofs of Propositions A.1-2 in [\[36\]](#). To bound the error
323 in the state, recall that the state residual is zero when evaluated at the true state, $c(\bar{\mathbf{U}}, \mathbf{Z}) = 0$. Therefore,

$$324 \quad c(\mathbf{U}, \mathbf{Z}) = c(\mathbf{U}, \mathbf{Z}) - c(\bar{\mathbf{U}}, \mathbf{Z}) = \int_0^1 d_1 c(\bar{\mathbf{U}} + t(\mathbf{U} - \bar{\mathbf{U}}), \mathbf{Z}) \cdot (\mathbf{U} - \bar{\mathbf{U}}) dt.$$

325 The error bound (4.1) then follows from [Assumption 1.2](#) using $\kappa_0 = \sigma_0$ and $\kappa_1 = \sigma_1$. Similarly we show the
326 bound on the adjoint error using the adjoint residual,

$$327 \quad h(\boldsymbol{\Lambda}, \bar{\mathbf{U}}, \mathbf{Z}) = h(\boldsymbol{\Lambda}, \bar{\mathbf{U}}, \mathbf{Z}) - h(\bar{\boldsymbol{\Lambda}}, \bar{\mathbf{U}}, \mathbf{Z}) = (d_1 c(\bar{\mathbf{U}}, \mathbf{Z}))^* (\boldsymbol{\Lambda} - \bar{\boldsymbol{\Lambda}}),$$

328 together with the Cauchy-Schwarz inequality and [Assumption 1.2](#) to see that

$$329 \quad \sigma_0 \|\boldsymbol{\Lambda} - \bar{\boldsymbol{\Lambda}}\| \leq \|h(\boldsymbol{\Lambda}, \bar{\mathbf{U}}, \mathbf{Z})\| \leq \sigma_1 \|\boldsymbol{\Lambda} - \bar{\boldsymbol{\Lambda}}\|.$$

330 We now bound the adjoint residual as

$$331 \quad \|h(\boldsymbol{\Lambda}, \bar{\mathbf{U}}, \mathbf{Z})\| \leq \|h(\boldsymbol{\Lambda}, \mathbf{U}, \mathbf{Z})\| + \|h(\boldsymbol{\Lambda}, \bar{\mathbf{U}}, \mathbf{Z}) - h(\boldsymbol{\Lambda}, \mathbf{U}, \mathbf{Z})\|, \\ 332 \quad \leq \|h(\boldsymbol{\Lambda}, \mathbf{U}, \mathbf{Z})\| + \|d_1 c(\mathbf{U}, \mathbf{Z}) - d_1 c(\bar{\mathbf{U}}, \mathbf{Z})\| \|\boldsymbol{\Lambda}\| + \|d_1 f(\mathbf{U}, \mathbf{Z}) - d_1 f(\bar{\mathbf{U}}, \mathbf{Z})\|.$$

334 The bound (4.2) follows from the Lipschitz continuity of $d_1 c$ and $d_1 f$, the boundedness of $\mathfrak{U}_0 \times \mathfrak{Z}_0$, and (4.1).
335 The proof of (4.3) is identical to the proof of Proposition A.2 in [\[36\]](#). In particular, (4.3) follows from (4.1),
336 (4.2) and the assumed Lipschitz continuity of $d_2 c$ and $d_2 f$. \square

337 **COROLLARY 4.3.** *Suppose [Assumption 1](#) holds for a bounded control set \mathfrak{Z}_0 . Fix a control $\mathbf{Z} \in \mathfrak{Z}_0$ and*
338 *rank parameter r . Suppose the approximate state $\hat{\mathbf{U}}_r = \{\{\bar{\mathbf{U}}\}\}_r$ is in \mathfrak{U}_0 almost surely. Then the state residual*
339 *is bounded by the tail energy of the true state $\bar{\mathbf{U}}$ on average:*

$$340 \quad \mathbf{E} \|c(\hat{\mathbf{U}}_r, \mathbf{Z})\| \leq 2\kappa_1 \tau_{r+1}(\text{mat}(\bar{\mathbf{U}}, M, N)).$$

341 *Now recall the approximate adjoint $\hat{\boldsymbol{\Lambda}}_r$ solves the adjoint equation (2.4) at the approximate state $\hat{\mathbf{U}}_r$. Suppose*
342 *that $\hat{\boldsymbol{\Lambda}}_r \in \mathfrak{U}_0$ almost surely. Then the error in the adjoints satisfies*

$$343 \quad (4.4) \quad \mathbf{E} \|\hat{\boldsymbol{\Lambda}}_r - \bar{\boldsymbol{\Lambda}}\| \leq 2\kappa_1 \kappa_2 \tau_{r+1}(\text{mat}(\bar{\mathbf{U}}, M, N)).$$

344 *Finally, the approximate gradient $g_r(\mathbf{Z}) = g(\hat{\boldsymbol{\Lambda}}_r, \hat{\mathbf{U}}_r, \mathbf{Z})$ satisfies the error bound*

$$345 \quad (4.5) \quad \mathbf{E} \|g_r(\mathbf{Z}) - \nabla F(\mathbf{Z})\| \leq 2\kappa_1 \kappa_4 \tau_{r+1}(\text{mat}(\bar{\mathbf{U}}, M, N)).$$

346 *Proof.* This result is a direct consequence of [Proposition 4.1](#) and [Theorem 3.1](#). \square

347 [Corollary 4.3](#) suggests that we should choose the fixed rank parameter r so that the tail energy,
348 $\tau_{r+1}(\text{mat}(\bar{\mathbf{U}}, M, N))$, is small. However, it can be difficult to choose a good fixed rank parameter in advance,
349 since the tail energy of the true state $\bar{\mathbf{U}}$ depends on the control variable \mathbf{Z} . Under stronger assumptions on
350 the reduced objective F , we can bound the distance from a given control to the optimum as a function of
351 the approximate gradient and the state residual. Both of these are easy to compute, and hence this result
352 can be used as a stopping criterion.

353 THEOREM 4.4. *Instate the assumptions of Corollary 4.3 for control $\mathbf{Z} \in \mathfrak{Z}_0$ and rank parameter r .*
 354 *Additionally assume that the reduced objective function F is strongly convex on \mathfrak{Z}_0 with parameter $\alpha > 0$.*
 355 *Let $\mathbf{Z}^* \in \mathfrak{Z}_0$ denote the solution to the reduced dynamic optimization problem (2.2). Then*

$$356 \quad (4.6) \quad \alpha \|\mathbf{Z} - \mathbf{Z}^*\| \leq \kappa_4 \|c(\mathbf{U}_r, \mathbf{Z})\| + \|g_r(\mathbf{Z})\|.$$

357 *Proof.* Using the strong convexity of F and the optimality of \mathbf{Z}^* , the error in control is bounded above
 358 by the gradient of the reduced objective function F as

$$359 \quad \alpha \|\mathbf{Z} - \mathbf{Z}^*\|^2 \leq \langle \nabla F(\mathbf{Z}) - \nabla F(\mathbf{Z}^*), \mathbf{Z} - \mathbf{Z}^* \rangle_{\mathfrak{Z}} = \langle \nabla F(\mathbf{Z}), \mathbf{Z} - \mathbf{Z}^* \rangle_{\mathfrak{Z}}.$$

360 Applying the Cauchy-Schwarz inequality and employing (4.3) ensures that

$$361 \quad \alpha \|\mathbf{Z} - \mathbf{Z}^*\| \leq \|\nabla F(\mathbf{Z}) - g_r(\mathbf{Z}) + g_r(\mathbf{Z})\| \leq \|\nabla F(\mathbf{Z}) - g_r(\mathbf{Z})\| + \|g_r(\mathbf{Z})\| \leq \kappa_4 \|c(\mathbf{U}_r, \mathbf{Z})\| + \|g_r(\mathbf{Z})\|. \quad \square$$

362 To use Theorem 4.4, run any optimization method using the approximate gradient $g_r(\mathbf{Z})$. Suppose the
 363 method terminates after k iterations at control $\mathbf{Z}^{(k)}$ so that $\|g_r(\mathbf{Z}^{(k)})\| \leq \epsilon$. Theorem 4.4 shows that the
 364 error in our optimal control is controlled by the state residual:

$$365 \quad \alpha \|\mathbf{Z}^{(k)} - \mathbf{Z}^*\| \leq \kappa_4 \|c(\mathbf{U}_r, \mathbf{Z})\| + \epsilon.$$

366
 367 **4.4. An adaptive rank approach.** In this section, we introduce an optimization algorithm, the
 368 *sketched trust-region method*, that dynamically adjusts the sketching rank parameter used to compute the
 369 approximate gradient. The rank is chosen to guarantee convergence to a stationary point of the dynamic
 370 optimization problem (2.2). This algorithm relies on the trust-region framework [7], which converges despite
 371 inexact first- and second-order information [14, 19, 20]. Unlike the fixed-rank method described in the
 372 previous section, the sketched trust-region method is a complete limited-memory optimization recipe.

373 Let us describe the standard trust-region method and the conditions required for convergence in the
 374 context of the dynamic optimization problem (2.2). Let $\mathbf{Z}^{(k)}$ be the control at the k^{th} iteration, with
 375 corresponding reduced objective function value $f^{(k)} := F(\mathbf{Z}^{(k)})$. The trust-region method approximates the
 376 reduced objective function centered around $\mathbf{Z}^{(k)}$, $\boldsymbol{\nu} \mapsto F(\mathbf{Z}^{(k)} + \boldsymbol{\nu})$, by a quadratic model

$$377 \quad m^{(k)}(\boldsymbol{\nu}) := f^{(k)} + \langle g^{(k)}, \boldsymbol{\nu} \rangle_{\mathfrak{Z}} + \frac{1}{2} \langle H^{(k)} \boldsymbol{\nu}, \boldsymbol{\nu} \rangle_{\mathfrak{Z}}.$$

378 To find the next iterate, the trust-region method computes a step $\bar{\boldsymbol{\nu}}$ which approximately¹ solves the trust-
 379 region subproblem constrained by the trust-region radius $\Delta^{(k)}$:

$$380 \quad (4.8) \quad \begin{aligned} & \text{minimize} && m^{(k)}(\boldsymbol{\nu}) \\ & \text{subject to} && \|\boldsymbol{\nu}\| \leq \Delta^{(k)}. \end{aligned}$$

381 This step is accepted so long as the actual decrease in the objective function value is large enough relative
 382 to the predicted decrease according to the model $m^{(k)}$. If the step is accepted and the actual reduction
 383 exceeds a specified threshold, the trust-region radius $\Delta^{(k)}$ is increased. If the step is rejected we decrease
 384 the trust-region radius.

385 To ensure global convergence of the trust-region method, the model used to form the trust-region sub-
 386 problem must satisfy Assumption 2 [19].

387 ASSUMPTION 2 (Trust-region model).

¹ Formally, the algorithm computes a step $\bar{\boldsymbol{\nu}}$ that satisfies the fraction of Cauchy decrease condition [7],

$$(4.7) \quad m^{(k)}(\mathbf{0}) - m^{(k)}(\bar{\boldsymbol{\nu}}) \geq \kappa_{\text{fcd}} \|g^{(k)}\| \min \left\{ \Delta^{(k)}, \frac{\|g^{(k)}\|}{1 + \|H^{(k)}\|} \right\},$$

for some $\kappa_{\text{fcd}} \geq 0$ independent of k . This condition is easy to achieve using, e.g., the Dogleg or truncated Conjugate Gradient method to compute $\bar{\boldsymbol{\nu}}$.

388 1. The approximate gradient $g^{(k)}$ is close to the true gradient $\nabla F(\mathbf{Z}^{(k)})$ in that it satisfies

$$389 \quad (4.9) \quad \|g^{(k)} - \nabla F(\mathbf{Z}^{(k)})\| \leq \theta \min \left\{ \|g^{(k)}\|, \Delta^{(k)} \right\},$$

390 for some fixed $\theta > 0$ independent of k .

391 2. The approximate Hessians $H^{(k)}$ are bounded independent of k : there exists $\tau_1 > 0$ such that

$$392 \quad \|H^{(k)}\| \leq \tau_1 < \infty \quad \forall k = 1, 2, \dots$$

393 We will show below how to ensure the first requirement with an approximate gradient $g^{(k)} := g_r(\mathbf{Z}^{(k)})$
 394 computed using the sketched state $\widehat{\mathbf{U}}_r$ with a sufficiently large rank parameter r . The second requirement is
 395 easily ensured by setting $H^{(k)}$ to be the identity, while we expect (and observe) faster convergence in practice
 396 when $H^{(k)}$ is the approximate Hessian. See [Algorithm A.5](#), which shows how to apply the approximate
 397 Hessian. Convergence is guaranteed regardless of the rank chosen for the Hessian approximation. We
 398 suggest fixing this parameter to be the same as the rank parameter for the approximate gradient.

399 **4.4.1. Choosing the rank to guarantee convergence.** The sketched trust-region method sets $g^{(k)} =$
 400 $g_r(\mathbf{Z}^{(k)})$ for some rank r . [Algorithm 4.4](#) ensures that r is chosen large enough that this approximate gradient
 401 satisfies the error bound (4.9), as proved in the following lemma. The function $\mu : \mathbb{N} \times [0, \infty) \rightarrow \mathbb{N}$ on line 9
 402 of [Algorithm 4.4](#) dictates how the rank r is increased and therefore is increasing in its first argument and
 403 decreasing in its second. A simple choice would be $\mu(r, \tau) = 2r$ or $\mu(r, \tau) = r + 1$.

404 **LEMMA 4.5.** *Instate [Assumption 1](#). Compute the gradient approximation $g^{(k)}$ using the Adaptive Rank*
 405 *[Algorithm 4.4](#). Then $g^{(k)}$ satisfies the gradient error bound (4.9) with $\theta = \kappa_4 \kappa_{\text{grad}}$.*

406 *Proof.* The adaptive-rank algorithm controls the error in the gradient approximation by increasing the
 407 target rank parameter until the constraint residual satisfies

$$408 \quad (4.10) \quad \|c(\widehat{\mathbf{U}}_r, \mathbf{Z}^{(k)})\| \leq \kappa_{\text{grad}} \min \left\{ \|g_r(\mathbf{Z}^{(k)})\|, \Delta^{(k)} \right\}.$$

409 A rank that satisfies (4.10) necessarily exists since the residual norm $\|c(\widehat{\mathbf{U}}_r, \mathbf{Z}^{(k)})\| \rightarrow 0$ as $r \rightarrow \min\{M, N\}$.
 410 Therefore, [Proposition 4.1](#) provides a bound on the error in the gradient approximation,

$$411 \quad \left\| g_r(\mathbf{Z}^{(k)}) - \nabla F(\mathbf{Z}^{(k)}) \right\| \leq \kappa_4 \left\| c(\widehat{\mathbf{U}}_r, \mathbf{Z}^{(k)}) \right\| \leq \kappa_4 \kappa_{\text{grad}} \min \left\{ \left\| g_r(\mathbf{Z}^{(k)}) \right\|, \Delta^{(k)} \right\}. \quad \square$$

412 [Algorithm A.2](#) presents a function `ResidualNorm` that computes the Frobenius norm of the state residual.

Algorithm 4.4 Adaptive-rank algorithm for approximate gradient.

Input: A control iterate $\mathbf{Z} \in \mathbb{R}^{m \times N}$, initial rank estimate r , sketch object for state $\{\mathbf{U}\}_r$, trust-region radius
 $\Delta > 0$, state residual tolerance $\kappa_{\text{grad}} > 0$, and rank update function $\mu : \mathbb{N} \times [0, \infty) \rightarrow \mathbb{N}$.

Output: Approximate gradient $g_r(\mathbf{Z}) \approx \nabla F(\mathbf{Z})$ for rank parameter r such that the bound (4.10) is satisfied.

Storage: $\mathcal{O}(r(M + N) + mN)$ for some rank parameter $r \leq \min\{M, N\}$.

```

1: function ADAPTIVERANKGRADIENT( $\mathbf{Z}$ ,  $r$ ,  $\{\mathbf{U}\}_r$ )
2:   repeat
3:      $\{\mathbf{U}\}_r$ .RECONSTRUCT!()           Reconstruct low-rank factors
4:      $\text{rnorm} \leftarrow \text{RESIDUALNORM}(\{\mathbf{U}\}_r, \mathbf{Z})$    Compute norm of constraint residual
5:      $g \leftarrow \text{Gradient}(\{\mathbf{U}\}_r, \mathbf{Z})$            Compute gradient
6:      $\text{rtol} \leftarrow \kappa_{\text{grad}} \cdot \min\{\|g\|, \Delta\}$ .   Compute residual tolerance
7:     if  $\text{rnorm} \leq \text{rtol}$  then           Gradient approximation satisfies (4.9)
8:       return  $g$ 
9:      $r \leftarrow \mu(r, \text{rtol})$            Increase rank parameter
10:     $\{\mathbf{U}\}_r \leftarrow \text{INITIALIZE!}(M, N, \text{rank} = r)$    Initialize sketch object for state
11:     $F \leftarrow \text{SOLVESTATE!}(\{\mathbf{U}\}_r, \mathbf{Z})$ .       Solve state equation
12:  until  $r > \min\{M, N\}$ 
13:  return  $g$ 

```

413 **4.4.2. Sketched trust-region algorithm.** We present the resulting sketched trust-region algorithm
414 as [Algorithm 4.5](#). To start the optimization we use an initial trust-region radius $\Delta^{(0)}$, initial rank param-
415 eter r_0 and the initial control $\mathbf{Z}^{(0)}$. The trust-region hyper-parameters of [Algorithm 4.5](#) are the ratio of
416 reduction thresholds $0 < \eta_1 < \eta_2 < 1$ and the trust-region radius update parameter $\gamma \in (0, 1)$. The func-
417 tion `SOLVETRUSUBPROBLEM` computes the step $\bar{\nu}$ that approximately solves the trust-region subproblem (4.8)
418 and satisfies the fraction of Cauchy decrease condition (4.7). Internally, `SOLVETRUSUBPROBLEM` may use the
419 function `APPLYFIXEDRANKHESSIAN` (see [Algorithm A.5](#)) to apply the fixed-rank Hessian approximation. To
420 validate the trust-region step, we compare the actual and predicted reductions,

$$421 \text{ared}^{(k)} := F(\mathbf{Z}^{(k)}) - F(\mathbf{Z}^{(k)} + \bar{\nu}) \quad \text{and} \quad \text{pred}^{(k)} := m^{(k)}(\mathbf{0}) - m^{(k)}(\bar{\nu}).$$

423 We accept the step if their ratio is greater than the threshold η_1 . The predicted reduction can be readily
424 computed as the model $m^{(k)}$ is known. The actual reduction requires us to solve the state and evaluate the
425 reduced objective function at the control candidate $\mathbf{Z}^{(k)} + \bar{\nu}$. The sketched trust-region method sketches
426 the state at $\mathbf{Z}^{(k)} + \bar{\nu}$ while computing the actual reduction so that (if the step is accepted) the approximate
gradient can be computed using the sketch without solving the state equation again. The following theorem

Algorithm 4.5 Sketched trust-region algorithm

Input: Initial control $\mathbf{Z}^{(0)}$, trust-region radius $\Delta^{(0)}$, target rank parameter r_0 ,
and trust-region hyper parameter set $P = \{\eta_1, \eta_2, \gamma\}$

Output: Control iterate $\mathbf{Z}^{(K)}$ such that the stopping criterion is satisfied

```

1: function SKETCHEDTRUSTREGION( $\mathbf{Z}^{(0)}$ ,  $\Delta^{(0)}$ ,  $r_0$ ,  $P$ )
2:    $\{\mathbf{U}\}_r \leftarrow \text{INITIALIZE!}(M, N, \text{rank} = r_0)$            Initialize state sketch
3:    $F \leftarrow \text{SOLVESTATE!}(\{\mathbf{U}\}_r, \mathbf{Z}^{(k)})$              Sketch state and evaluate objective
4:    $r \leftarrow r_0$                                        Set sketch rank
5:   while “Not Converged” do
6:      $(g^{(k)}, r) \leftarrow \text{ADAPTIVERANKGRADIENT}(\mathbf{Z}^{(k)}, r, \{\mathbf{U}\}_r)$    Approximate gradient
7:      $(\bar{\nu}, \text{pred}^{(k)}) \leftarrow \text{SOLVETRUSUBPROBLEM}(g^{(k)}, \Delta^{(k)})$    Compute trial step
8:      $\{\mathbf{U}\}_r \leftarrow \text{INITIALIZE!}(M, N, \text{rank} = r)$            Reinitialize state sketch
9:      $f^{(k+1)} \leftarrow \text{SOLVESTATE!}(\{\mathbf{U}\}_r, \mathbf{Z}^{(k)} + \bar{\nu})$    Compute new objective function value
10:     $\rho^{(k)} = (f^{(k)} - f^{(k+1)})/\text{pred}^{(k)}$                  Compute ratio of reduction
11:    if  $\rho^{(k)} \geq \eta_1$  then                                  Validate step using ratio of reduction
12:       $\mathbf{Z}^{(k+1)} = \mathbf{Z}^{(k)} + \bar{\nu}$ 
13:    else
14:       $\mathbf{Z}^{(k+1)} = \mathbf{Z}^{(k)}$ 
15:      if  $\rho^{(k)} \geq \eta_2$  then                                  Update trust-region radius
16:         $\Delta^{(k+1)} \in [\Delta^{(k)}, \infty)$ 
17:      else if  $\rho^{(k)} \leq \eta_1$  then
18:         $\Delta^{(k+1)} \in (0, \gamma\|\bar{\nu}\|]$ 
19:      else
20:         $\Delta^{(k+1)} \in [\gamma\|\bar{\nu}\|, \Delta^{(k)}]$ 

```

427 shows that the sequence of iterates $\{\mathbf{Z}^{(k)}\}$ generated by [Algorithm 4.5](#) converges to a stationary point of the
428 reduced objective function F .
429

430 **THEOREM 4.6** (Convergence of the sketched trust-region algorithm). *Instate [Assumption 1](#), and further*
431 *suppose that the reduced objective function F is bounded below and twice continuously differentiable with*
432 *locally uniformly bounded Hessian: for any bounded convex set $\mathfrak{Z}_0 \subset \mathfrak{Z}$, there exists $\tau_0 > 0$ such that*

$$433 \|\nabla^2 F(\mathbf{Z})\| \leq \tau_0 < \infty \quad \forall \mathbf{Z} \in \mathfrak{Z}_0.$$

434 *Suppose that the iterates $\mathbf{Z}^{(k)}$ generated by [Algorithm 4.5](#) lie in the open, bounded and convex set $\mathfrak{Z}_0 \subset \mathfrak{Z}$ for*
435 *all k . Then, the sequence of iterates $\{\mathbf{Z}^{(k)}\}$ satisfies*

$$436 \liminf_{k \rightarrow \infty} \|g_r^{(k)}\| = \liminf_{k \rightarrow \infty} \|\nabla F(\mathbf{Z}^{(k)})\| = 0.$$

437 *Proof.* Notice that the trust-region model used by [Algorithm 4.5](#) satisfies [Assumption 2](#). Therefore, the
 438 proof of this result follows from the convergence analysis for the inexact trust-region method in [\[19\]](#) with
 439 only a slight modification to account for the local assumptions associated with \mathfrak{J}_0 . \square

440 *Remark 4.7.* The assumption that F is twice continuously differentiable can be relaxed to the require-
 441 ment that F is continuously differentiable with Lipschitz continuous gradient. In this case, the proof of
 442 [Theorem 4.6](#) is virtually identical to the proof in [\[19\]](#); however, the proofs of Lemmas A.2 and A.3 in [\[19\]](#)
 443 must be updated accordingly.

444 **5. Optimal control of linear parabolic PDEs.** In this section, we introduce a class of linear
 445 parabolic optimal control problems and discuss how to discretize them to obtain a problem of the form
 446 [\(2.1\)](#) that satisfies [Assumption 1](#) and the inexact gradient condition [\(4.9\)](#). Let $\Omega \subset \mathbb{R}^d$ be an open, con-
 447 nected, and bounded set and let $\Gamma \subseteq \Omega \cup \partial\Omega$ where $\partial\Omega$ denotes the boundary of Ω . The set Γ is the spatial
 448 support of the control function and permits both boundary and volumetric controls. The state is supported
 449 on the space-time cylinder $\Omega_T := (0, T) \times \Omega$ and the control is supported on $\Gamma_T := (0, T) \times \Gamma$ for $T > 0$.
 450 We denote by $H^1(\Omega)$ the usual Sobolev space of $L^2(\Omega)$ -functions with weak derivatives in $L^2(\Omega)$ and let
 451 $V \subseteq H^1(\Omega)$ be a separable Hilbert space such that V is continuously and densely embedded into $L^2(\Omega)$
 452 (typically, $V = H^1(\Omega)$ or $V = H_0^1(\Omega)$). Furthermore, we assume that Γ is sufficiently regular so that

$$453 \quad v \mapsto \int_{\Gamma} gv \, dx \in V^* \quad \forall g \in L^2(\Gamma),$$

454 where V^* denotes the topological dual space of V .

455 Let $\mathcal{L}(t) : V \rightarrow V^*$ denote a second-order linear elliptic partial differential operator for $t \in [0, T]$. For
 456 example, $\mathcal{L}(t)$ could represent the weak form of the advection-reaction-diffusion operator

$$457 \quad (5.1) \quad u \mapsto \{-\nabla \cdot (A(t, \cdot) \nabla u) + b(t, \cdot) \cdot \nabla u + c(t, \cdot) u\},$$

458 where $A : \Omega_T \rightarrow \mathbb{R}^{d \times d}$ is the diffusivity tensor, $b : \Omega_T \rightarrow \mathbb{R}^d$ is an advection field, and $c : \Omega_T \rightarrow \mathbb{R}$ is a reaction
 459 coefficient. Here, ∇ refers to the derivative with respect to x . To guarantee the existence of solutions, we
 460 assume that the linear operator \mathcal{L} is uniformly bounded and uniformly coercive, which we define below.

461 **DEFINITION 5.1.** *The operator \mathcal{L} is uniformly bounded if for some $\varepsilon_0 > 0$ independent of $t \in [0, T]$,*

$$462 \quad \langle \mathcal{L}(t)u, v \rangle_{V^*, V} \leq \varepsilon_0 \|u\|_V \|v\|_V \quad \forall u, v \in V,$$

463 *for almost all (a.a.) $t \in [0, T]$. Moreover, \mathcal{L} is uniformly coercive if for some $\varepsilon_1 > 0$ independent of $t \in [0, T]$,*

$$464 \quad \varepsilon_1 \|v\|_V^2 \leq \langle \mathcal{L}(t)v, v \rangle_{V^*, V} \quad \forall v \in V,$$

465 *for a.a. $t \in [0, T]$. Here, $\langle \cdot, \cdot \rangle_{V^*, V}$ denotes the duality pairing between V^* and V .*

466 In addition to uniformly bounded and coercive, we assume that $t \mapsto \langle \mathcal{L}(t)u, v \rangle_{V^*, V}$ is measurable for all
 467 $u, v \in V$, $\beta \in L^2(0, T; V^*)$ is a forcing term, and $u_0 \in L^2(\Omega)$ is the initial state. We consider the optimal
 468 control problem

$$469 \quad (5.2a) \quad \text{minimize } \left\{ \frac{1}{2} \int_0^T \int_{\Omega} (u - w)^2 \, dx dt + \frac{\alpha}{2} \int_0^T \int_{\Gamma} z^2 \, dx dt \right\}$$

$$470 \quad \text{subject to } \quad u \in W(0, T), \quad z \in L^2(\Gamma_T)$$

$$471 \quad (5.2b) \quad \begin{cases} \int_{\Omega} \frac{\partial u}{\partial t}(t, \cdot) v \, dx + \langle [\mathcal{L}(t)u](t, \cdot), v \rangle_{V^*, V} = \langle \beta(t), v \rangle_{V^*, V} + \int_{\Gamma} z(t, \cdot) v \, dx & \text{a.a. } t \in [0, T], \forall v \in V \\ u(0, x) = u_0(x) & \text{a.a. } x \in \Omega, \end{cases}$$

473 where $\alpha > 0$ is the control penalty parameter, $w \in L^2(\Omega_T)$ is the desired state, and

$$474 \quad W(0, T) := \{v : \Omega_T \rightarrow \mathbb{R} \mid v \in L^2(0, T; V) \text{ and } \partial v / \partial t \in L^2(0, T; V^*)\}$$

475 is the solution space for [\(5.2b\)](#). In fact, under the stated assumptions, [\(5.2b\)](#) has a unique solution in $W(0, T)$
 476 for any $z \in L^2(\Gamma_T)$ (cf. [\[15, Th. 1.35\]](#)).

477 **5.1. Discretization.** To obtain a finite-dimensional approximation of (5.2), we discretize the PDE
478 (5.2b) using Galerkin finite elements in space and implicit Euler in time. The subsequent results also hold
479 for other time discretizations including Crank-Nicolson or explicit Euler. We partition the time interval
480 $[0, T]$ into N subintervals (t_{n-1}, t_n) with $0 = t_0 < t_1 < \dots < t_{N-1} < t_N = T$ and denote the finite-element
481 approximation space for the state by $V_M \subset V$ where M is the dimension of V_M . We further denote by
482 $Z_m \subset L^2(\Gamma)$ the control approximation space where m is the dimension of Z_m . Using these spaces, we
483 can write the discretized state equation as: for fixed $z_{m,n} \in Z_m$ with $n = 1, \dots, N$, find $u_{M,n} \in V_M$ with
484 $n = 1, \dots, N$ such that $u_{M,0} = \tilde{u}_0$ where $\tilde{u}_0 \in V_M$ is an approximation of u_0 and

$$485 \int_{\Omega} u_{M,n} v \, dx + \delta t_n \langle \mathcal{L}(t_n) u_{M,n}, v \rangle_{V^*, V} = \int_{\Omega} u_{M,n-1} v \, dx + \delta t_n \langle \beta(t_n), v \rangle_{V^*, V} + \delta t_n \int_{\Gamma} z_{m,n} v \, dx \quad \forall v \in V_M,$$

487 where $\delta t_n := t_n - t_{n-1}$. Given bases $\{\phi_i\}_{i=1}^M$ and $\{\psi_i\}_{i=1}^m$ of V_M and Z_m , respectively, we can rewrite the
488 discretized PDE as the linear system of equations: given $\mathbf{z}_n \in \mathbb{R}^m$ for $n = 1, \dots, N$, find $\mathbf{u}_n \in \mathbb{R}^M$ such that

$$489 (5.3) \quad c_n(\mathbf{u}_{n-1}, \mathbf{u}_n, \mathbf{z}_n) = (\mathbf{M} + \delta t_n \mathbf{K}_n) \mathbf{u}_n - \mathbf{M} \mathbf{u}_{n-1} - \delta t_n \mathbf{b}_n - \delta t_n \mathbf{B} \mathbf{z}_n = 0, \quad n = 1, \dots, N,$$

490 where

$$491 [\mathbf{M}]_{ij} := \int_{\Omega} \phi_j \phi_i \, dx, \quad [\mathbf{K}_n]_{ij} := \langle \mathcal{L}(t_n) \phi_j, \phi_i \rangle_{V^*, V}, \quad [\mathbf{B}]_{ij} := \int_{\Gamma} \psi_j \phi_i \, dx, \quad \text{and} \quad [\mathbf{b}_n]_j := \langle \beta(t_n), \phi_j \rangle_{V^*, V}.$$

492 With this notation, the discretized version of (5.2a) is

$$493 (5.4) \quad \underset{\mathbf{U} \in \mathfrak{U}, \mathbf{Z} \in \mathfrak{Z}}{\text{minimize}} \quad \frac{1}{2} \sum_{n=1}^N \delta t_n \{ (\mathbf{u}_n - \mathbf{w}_n)^\top \mathbf{M} (\mathbf{u}_n - \mathbf{w}_n) + \alpha \mathbf{z}_n^\top \mathbf{R} \mathbf{z}_n \} \quad \text{subject to} \quad (5.3),$$

494 where we have approximated the temporal integral in the objective function using the right endpoint rule,
495 $\mathbf{w}_n \in \mathbb{R}^M$ are the coefficients associated with an approximation of $w(t_n, \cdot)$ in V_M , and

$$496 [\mathbf{R}]_{ij} = \int_{\Gamma} \psi_j \psi_i \, dx.$$

497 The assumptions on \mathcal{L} and the choice of discretization ensure that $(\mathbf{M} + \delta t_n \mathbf{K}_n)$ is invertible for all $n =$
498 $1, \dots, N$ and therefore [Assumption 1.2](#) is satisfied. In addition, since the dynamic constraint in (5.4) is
499 linear in the state and control variables, and the objective function in (5.4) is quadratic, [Assumption 1.3](#) is
500 satisfied. Finally, since the matrices $(\mathbf{M} + \delta t_n \mathbf{K}_n)$ are invertible and the constraint is linear, the dynamic
501 constraint has a unique solution that depends linearly on the control $\mathbf{Z} \in \mathfrak{Z}$. Therefore, [Assumption 1.1](#)
502 holds for any bounded set of controls. To verify (4.9), we employ stability estimates for (5.3).

503 **5.2. Stability estimates.** The linearity of (5.2b) and the uniform coercivity of \mathcal{L} provide numerous
504 convenient properties associated with the discretized PDE (5.3). In this section, we use these properties to
505 ensure that the required assumptions for our sketching algorithm are satisfied. We first have the following
506 error bound associated with the discretized state equation (5.3).

507 **THEOREM 5.2.** *Let $\bar{\mathbf{U}} \in \mathfrak{U}$ denote the solution to (5.3) for fixed control $\mathbf{Z} \in \mathfrak{Z}$ and let $\mathbf{U} \in \mathfrak{U}$ be arbitrary.*
508 *Then, the following bound holds*

$$509 \|\mathbf{u}_n - \bar{\mathbf{u}}_n\|_{\mathbf{M}} \leq (1 + \delta t_n \omega \varepsilon_1)^{-1} \left\{ \|\mathbf{u}_0 - \bar{\mathbf{u}}_0\|_{\mathbf{M}} + \sum_{i=1}^n \|c_i(\mathbf{u}_{i-1}, \mathbf{u}_i, \mathbf{z}_i)\|_{\mathbf{M}^{-1}} \right\}, \quad n = 1, \dots, N,$$

510 where $\bar{\mathbf{u}}_n$ and \mathbf{u}_n are the n^{th} subvectors in $\bar{\mathbf{U}}$ and \mathbf{U} , respectively, and $\omega > 0$ is the embedding constant
511 associated with $V \hookrightarrow L^2(\Omega)$.

512 *Proof.* First, we write $c_n(\mathbf{u}_{n-1}, \mathbf{u}_n, \mathbf{z}_n)^\top (\mathbf{u}_n - \bar{\mathbf{u}}_n)$ in the form of (5.1). To this end, let $\bar{u}_{M,n}, u_{M,n} \in V_M$
513 denote the functions

$$514 \bar{u}_{M,n} = \sum_{i=1}^M [\bar{\mathbf{u}}_n]_i \phi_i \quad \text{and} \quad u_{M,n} = \sum_{i=1}^M [\mathbf{u}_n]_i \phi_i,$$

515 respectively, and $\eta_n = (u_{M,n} - \bar{u}_{M,n})$. Then we have that $c_n(\mathbf{u}_{n-1}, \mathbf{u}_n, \mathbf{z}_n)^\top (\mathbf{u}_n - \bar{\mathbf{u}}_n)$ is equal to

$$516 \quad \int_{\Omega} u_{M,n} \eta_n \, dx + \delta t_n \langle \mathcal{L}(t_n) u_{M,n}, \eta_n \rangle_{V^*, V} - \int_{\Omega} u_{M,n-1} \eta_n \, dx - \delta t_n \langle \beta(t_n), \eta_n \rangle_{V^*, V} - \delta t_n \int_{\Gamma} z_{m,n} \eta_n \, dx$$

$$517 \quad = \int_{\Omega} \eta_n^2 \, dx + \delta t_n \langle \mathcal{L}(t_n) \eta_n, \eta_n \rangle_{V^*, V} - \int_{\Omega} \eta_{n-1} \eta_n \, dx,$$

519 since $\bar{u}_{M,n}$ solves (5.3). The Cauchy-Schwarz inequality, the continuous embedding of V into $L^2(\Omega)$, and the
520 uniform coercivity of \mathcal{L} then ensure that

$$521 \quad (1 + \delta t_n \omega \varepsilon_1) \|\mathbf{u}_n - \bar{\mathbf{u}}_n\|_{\mathbf{M}} \leq \|c_n(\mathbf{u}_{n-1}, \mathbf{u}_n, \mathbf{z}_n)\|_{\mathbf{M}^{-1}} + \|\mathbf{u}_{n-1} - \bar{\mathbf{u}}_{n-1}\|_{\mathbf{M}}.$$

522 Repeated application of this inequality yields the desired bound. \square

523 **Theorem 5.2** ensures that the lower bound in (4.1) holds for all $\mathbf{Z} \in \mathfrak{Z}$ and $\mathbf{U} \in \mathfrak{U}$ (rather than only on
524 some bounded sets \mathfrak{Z}_0 and \mathfrak{U}_0). The upper bound in (4.1) follows due to the linearity of c_n and holds again
525 for all $\mathbf{Z} \in \mathfrak{Z}$ and $\mathbf{U} \in \mathfrak{U}$. Moreover, if \mathbf{U}_r is the sketched state, then **Theorem 5.2** yields

$$526 \quad \|\mathbf{u}_{r,n} - \bar{\mathbf{u}}_n\|_{\mathbf{M}} \leq (1 + \delta t_n \omega \varepsilon_1)^{-1} \sum_{i=1}^n \|c_i(\mathbf{u}_{r,i-1}, \mathbf{u}_{r,i}, \mathbf{z}_i)\|_{\mathbf{M}^{-1}}.$$

527 Next, we demonstrate that the adjoint error bound (4.2) holds globally as well. To this end, we write the
528 adjoint equation associated with the discretized problem (5.4):

$$529 \quad (5.5a) \quad (\mathbf{M} + \delta t_N \mathbf{K}_N)^* \boldsymbol{\lambda}_N = -\delta t_N \mathbf{M}(\mathbf{u}_N - \mathbf{w}_N),$$

$$530 \quad (5.5b) \quad (\mathbf{M} + \delta t_n \mathbf{K}_n)^* \boldsymbol{\lambda}_n = \mathbf{M} \boldsymbol{\lambda}_{n+1} - \delta t_n \mathbf{M}(\mathbf{u}_n - \mathbf{w}_n) \quad \text{for } n = N-1, \dots, 1.$$

532 As before, we denote the solution to (5.5) with \mathbf{u}_n replaced by $\bar{\mathbf{u}}_n$ for $n = 1, \dots, N$ by $\bar{\boldsymbol{\Lambda}}$. We have the
533 following useful stability estimate associated with (5.5) that bounds the error in the adjoint.

534 **THEOREM 5.3.** *Let $\bar{\boldsymbol{\Lambda}} \in \mathfrak{U}$ be the solution to the adjoint equation (5.5) associated with $\bar{\mathbf{U}} \in \mathfrak{U}$ and let*
535 *$\boldsymbol{\Lambda}, \mathbf{U} \in \mathfrak{U}$ be arbitrary. Then the following bound holds*

$$536 \quad \|\boldsymbol{\lambda}_n - \bar{\boldsymbol{\lambda}}_n\|_{\mathbf{M}} \leq (1 + \delta t_n \omega \varepsilon_1)^{-1} \left\{ \sum_{j=n}^N \|(\mathbf{M} + \delta t_j \mathbf{K}_j)^* \boldsymbol{\lambda}_j - \mathbf{M} \boldsymbol{\lambda}_{j+1} + \delta t_j \mathbf{M}(\mathbf{u}_j - \mathbf{w}_j)\|_{\mathbf{M}^{-1}} \right.$$

$$537 \quad \left. + \|\boldsymbol{\lambda}_N - \bar{\boldsymbol{\lambda}}_N\|_{\mathbf{M}} + \delta t_n \|\mathbf{u}_n - \bar{\mathbf{u}}_n\|_{\mathbf{M}} \right\}, \quad n = 1, \dots, N,$$

539 where $\bar{\boldsymbol{\lambda}}_n$ and $\boldsymbol{\lambda}_n$ denote the n^{th} subvectors of $\bar{\boldsymbol{\Lambda}}$ and $\boldsymbol{\Lambda}$, respectively and $\omega > 0$ is the embedding constant
540 associated with $V \hookrightarrow L^2(\Omega)$.

541 *Proof.* Using similar notation as in the proof of **Theorem 5.2**, we can write adjoint residual evaluated at
542 $\boldsymbol{\Lambda}$ and \mathbf{U} as

$$543 \quad (5.7) \quad \int_{\Omega} \lambda_{M,n} v \, dx + \delta t_n \langle \mathcal{L}(t_n)^* \lambda_{M,n}, v \rangle_{V^*, V} - \int_{\Omega} \lambda_{M,n+1} v \, dx + \delta t_n \int_{\Omega} (u_{M,n} - w_{M,n}) v \, dx,$$

544 for $v \in V_M$ where $w_{M,n} \in V_M$ is the appropriate approximation of w . Evaluating this residual at $\bar{\boldsymbol{\Lambda}}$ and $\bar{\mathbf{U}}$
545 returns zero. Let $e_n = (\lambda_{M,n} - \bar{\lambda}_{M,n})$ and $\eta_n = (u_{M,n} - \bar{u}_{M,n})$. With this notation, (5.7) is equal to

$$546 \quad \int_{\Omega} e_n v \, dx + \delta t_n \langle \mathcal{L}(t_n)^* e_n, v \rangle_{V^*, V} - \int_{\Omega} e_{n+1} v \, dx + \delta t_n \int_{\Omega} \eta_n v \, dx.$$

547 Set $v = e_n$. Then, applying the Cauchy-Schwarz inequality, and using the uniform coercivity of \mathcal{L} and the
548 continuous embedding of V into $L^2(\Omega)$ yields

$$549 \quad (1 + \delta t_n \omega \varepsilon_1) \|\boldsymbol{\lambda}_n - \bar{\boldsymbol{\lambda}}_n\|_{\mathbf{M}} \leq \|(\mathbf{M} + \delta t_n \mathbf{K}_n)^* \boldsymbol{\lambda}_n - \mathbf{M} \boldsymbol{\lambda}_{n+1} + \delta t_n \mathbf{M}(\mathbf{u}_n - \mathbf{w}_n)\|_{\mathbf{M}^{-1}}$$

$$550 \quad + \|\boldsymbol{\lambda}_{n+1} - \bar{\boldsymbol{\lambda}}_{n+1}\|_{\mathbf{M}} + \delta t_n \|\mathbf{u}_n - \bar{\mathbf{u}}_n\|_{\mathbf{M}}.$$

552 Repeated application of this bound proves the desired result. \square

553 By [Theorems 5.2](#) and [5.3](#), we see that the adjoint error bound [\(4.2\)](#) holds globally. In particular, let Λ_r
 554 denote the solution to [\(5.5\)](#) associated with the sketched state \mathbf{U}_r . Then [Theorems 5.2](#) and [5.3](#) ensure that

$$555 \quad \|\lambda_{r,n} - \bar{\lambda}_n\|_{\mathbf{M}} \leq (1 + \delta t_n \omega \varepsilon_1)^{-1} \delta t_n \|\mathbf{u}_{r,n} - \bar{\mathbf{u}}_n\|_{\mathbf{M}} \leq (1 + \delta t_n \omega \varepsilon_1)^{-2} \delta t_n \sum_{i=1}^n \|c_i(\mathbf{u}_{r,i-1}, \mathbf{u}_{r,i}, \mathbf{z}_i)\|_{\mathbf{M}^{-1}}.$$

556 Hence [Algorithm 4.4](#) ensures that the inexact gradient condition [\(4.9\)](#) is satisfied.

558 **6. Numerical examples.** We demonstrate the effectiveness of the sketched trust-region algorithm on
 559 two PDE-constrained optimization problems. We present one example, the optimal control of an advection-
 560 reaction-diffusion equation, that satisfies the assumptions of the previous section, and therefore is guaranteed
 561 to converge. We also present results on optimal flow control. This application is governed by the Navier-
 562 Stokes equations for which it is difficult to verify the assumptions of our theory, and so our algorithm does
 563 not necessarily admit guarantees for this problem. Nevertheless, we show remarkably good performance for
 564 this application.

565 In the numerics, we compute the function `ResidualNorm` using a domain-specific weighted norm (instead
 566 of the Frobenius norm) that respects the natural problem scaling. The guarantees of the method still hold:
 567 since all norms are equivalent in finite-dimensional vector spaces, we can ensure the gradient error bound
 568 [\(4.9\)](#) holds (with a different value of the parameter θ) using any norm to measure the state residual. In
 569 addition, for both examples we set $\kappa_{\text{grad}} = 1$, $\Delta^{(0)} = 10$, $\eta_1 = 0.05$, $\eta_2 = 0.9$, and $\gamma = 0.25$ in [Algorithm 4.5](#).
 570 We terminate the algorithm if the norm of the gradient is smaller than a prescribed tolerance `gtol` or when
 571 it exceeds a set maximum number of iterations `maxit`.

572 **6.1. Optimal control of an advection-reaction-diffusion equation.** For this example, our goal
 573 is to control the linear parabolic PDE [\(5.2b\)](#) where $\Omega = (0, 0.6) \times (0, 0.2)$, $\Gamma = \Omega$, and \mathcal{L} is given by [\(5.1\)](#)
 574 with time-independent coefficients. In particular, the forcing term β is the characteristic function of the
 575 intersection of the ball of radius 0.07 centered at $(0.1, 0.1)^\top$ with Ω , the diffusivity coefficient $A = 0.1\mathbf{I}$
 576 where $\mathbf{I} \in \mathbb{R}^{d \times d}$ is the identity matrix, the reaction coefficient is $c \equiv 1$, and the advection field is given
 577 by $b(x) = (7.5 - 2.5x_1, 2.5x_2)^\top$. We further supply [\(5.2b\)](#) with zero initial concentration $u_0 = 0$ and pure
 578 Neumann boundary conditions (i.e., $V = H^1(\Omega)$). Note that \mathcal{L} is constant in time and is uniformly coercive
 579 since $\nabla \cdot b \equiv 0$ in Ω and $b \cdot n \geq 0$ on $\partial\Omega$ where n denotes the outward normal vector. Moreover, we set the
 580 target state $w \equiv 1$. Our optimization problem is then given by [\(5.2\)](#) with $\alpha = 10^{-4}$. We discretized [\(5.2b\)](#)
 581 in space using Q1 finite elements on a uniform mesh of 60×20 quadrilateral elements. In time, we discretize
 582 using Implicit Euler with 500 equal time steps. This discretization results in $1,281 \times 500 = 640,500$ degrees
 583 of freedom. Moreover, the maximum possible rank of the state matrix is 500. [Figure 2](#) depicts the tail
 584 energy and sketching error averaged over 20 realizations for the uncontrolled and optimal states. Both the
 585 tail energy and sketching error decay exponentially fast until saturating below $\mathcal{O}(10^{-12})$.

586 We solved this problem using a Newton-based trust-region algorithm with fixed sketch rank and using
 587 [Algorithm 4.5](#) with the rank update function $\mu(r, \tau) = \max\{r + 2, \lceil (b - \log \tau)/a \rceil\}$ where $a > 0$ and $b \in \mathbb{R}$
 588 are computed by fitting a linear model of the logarithm of the average sketching error as a function of the
 589 rank for the uncontrolled state. For this problem, $a = 2.6125$, $b = 2.4841$, `gtol` = 10^{-7} , and `maxit` = 20.
 590 The final objective value, the iteration count, the number of function evaluations, the number of gradient
 591 evaluations, the cumulative truncated conjugate gradient (CG) iteration count, and the compression factor
 592 ζ defined to be

$$593 \quad \zeta := \frac{\text{full storage}}{\text{reduced storage}} = \frac{640,500}{k(1,281 + 500) + s^2}$$

594 where $k = 2r + 1$, $s = 2k + 1$ for each rank parameter r from 1 to 5 are displayed in [Table 1](#). Notice that
 595 with rank 1 the algorithm did not converge, whereas the optimal objective function value is achieved up
 596 to 6 digits with rank 2. This is likely due to inaccuracies in the gradient. For this problem, the rank-2
 597 sketch requires roughly three times more CG iterations (which dominate the computational work) than the
 598 full-storage algorithm; however, using the rank-2 sketch reduces the required memory by a factor of 70.96.

599 The iteration history of [Algorithm 4.5](#) is listed in the top section of [Table 2](#). For comparison, we have
 600 also listed the iteration history for the full-storage algorithm in the bottom section of [Table 2](#). We notice
 601 that the sketched trust-region algorithm performs comparably to the full-storage algorithm, but reduces the

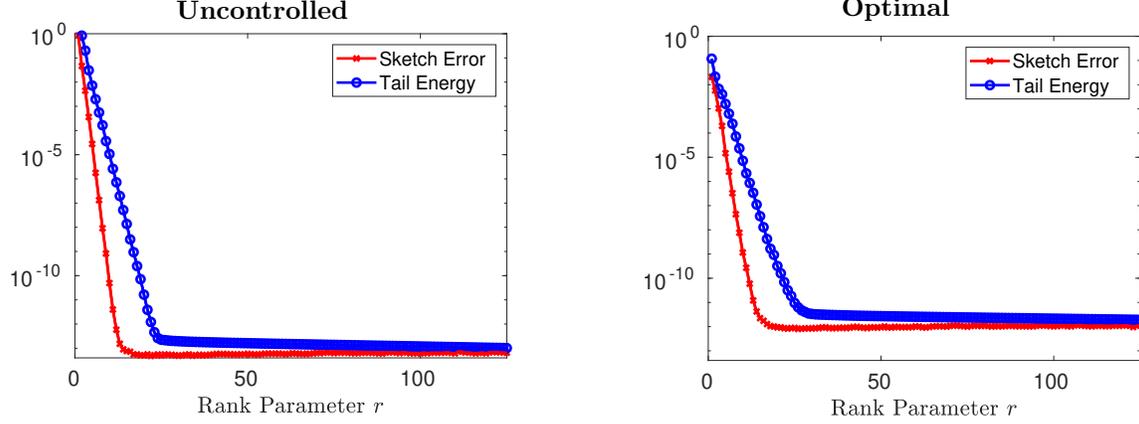


Figure 2: The sketching error averaged over 20 realizations and the tail energy for the uncontrolled state (left) and the optimal state (right) of the advection-reaction-diffusion example. Recall that the rank of the sketch is $k = 2r + 1$.

602 required memory by a factor of $\zeta = 23.14$ at the final iteration. It may be possible to further reduce the
603 memory burden by tuning the parameter κ_{grad} or by employing a different rank update function μ .

rank	objective	iteration	nfval	ngrad	iterCG	compression ζ
*1	5.544040e-4	20	21	11	196	118.79
2	5.528490e-4	5	6	6	151	70.96
3	5.528490e-4	4	5	5	78	50.46
4	5.528490e-4	4	5	5	67	38.08
5	5.528490e-4	4	5	5	59	31.83
Adaptive	5.528490e-4	4	5	5	65	23.14
Full	5.528490e-4	4	5	5	53	1.00

Table 1: Algorithmic performance summary for the advection-reaction-diffusion example for fixed rank, adaptive rank and full storage: **objective** is the final objective function value, **iteration** is the total number of iterations, **nfval** is the number of function evaluations, **ngrad** is the number of gradient evaluations, **iterCG** is the total number of truncated CG iterations, and **compression ζ** is the compression factor. *The rank 1 experiment terminated because it exceeded the maximum number of iterations.

603

604 **6.2. Optimal control of flow past a cylinder.** For this example, we follow the problem set up in
605 [13] and consider fluid flow past a cylinder. The cylinder impedes the flow; our goal is to rotate the cylinder
606 to improve the flow rate. Formally, we let cylinder $C \subset \mathbb{R}^2$ denote the closed ball of radius $R = 0.5$ centered
607 at the origin $x_0 = (0, 0)^\top$, define the domain $D = (-15, 45) \times (-15, 15)$ and let $\Gamma_{\text{out}} = \{45\} \times (-15, 15)$
608 denote the outflow boundary. We consider the optimal flow control problem

$$609 \quad (6.1) \quad \underset{z \in L^2(0, T)}{\text{minimize}} \int_0^T \left\{ \int_{\partial C} \left(\frac{1}{\text{Re}} \frac{\partial \mathbf{v}}{\partial n} - pn \right) \cdot (z\tau - \mathbf{v}_\infty) dx + \frac{\alpha}{2} z(t)^2 \right\} dt, \quad \alpha > 0,$$

610 where the velocity and pressure pair $(\mathbf{v}, p) : [0, T] \times D \rightarrow \mathbb{R}^2 \times \mathbb{R}$ solves the Navier-Stokes equations

$$611 \quad (6.2a) \quad \frac{\partial \mathbf{v}}{\partial t} - \frac{1}{\text{Re}} \Delta \mathbf{v} + (\mathbf{v} \cdot \nabla) \mathbf{v} + \nabla p = 0 \quad \text{in } (0, T) \times D \setminus C$$

$$612 \quad (6.2b) \quad \nabla \cdot \mathbf{v} = 0 \quad \text{in } (0, T) \times D \setminus C$$

$$613 \quad (6.2c) \quad \frac{1}{\text{Re}} \frac{\partial \mathbf{v}}{\partial n} - pn = 0 \quad \text{on } (0, T) \times \Gamma_{\text{out}}$$

$$614 \quad (6.2d) \quad \mathbf{v} = \mathbf{v}_\infty \quad \text{on } (0, T) \times \partial D \setminus \Gamma_{\text{out}}$$

$$615 \quad (6.2e) \quad \mathbf{v} = z\tau \quad \text{on } (0, T) \times \partial C$$

	iter	value	gnorm	snorm	delta	iterCG	rank	rnorm
Adaptive	0	5.446e-2	5.990e-3	---	1.000e+1	---	1	2.707e-4
	1	1.375e-2	2.205e-3	1.000e+1	2.500e+1	1	1	1.990e-4
	2	1.475e-3	1.408e-4	2.499e+1	6.250e+1	5	5	6.700e-7
	3	5.531e-4	6.431e-7	4.077e+1	1.563e+1	27	7	3.893e-9
	4	5.528e-4	5.039e-9	1.059e+0	3.906e+2	32	7	1.508e-9
Full	0	5.446e-2	5.989e-3	---	1.000e+1	---	---	---
	1	1.375e-2	2.201e-3	1.000e+1	2.500e+1	1	---	---
	2	1.472e-3	1.401e-4	2.500e+1	6.250e+1	5	---	---
	3	5.538e-4	1.361e-6	4.051e+1	1.563e+1	19	---	---
	4	5.528e-4	8.416e-9	2.178e+0	3.906e+2	28	---	---

Table 2: The iteration histories for the adaptive rank (top) and full storage (bottom) algorithms: `iter` is the iteration number, `value` is the objective function value, `gnorm` is the norm of the gradient, `snorm` is the norm of the step, `delta` is the trust-region radius, `iterCG` is the number of truncated CG iterations, `rank` is the rank of the sketch, and `rnorm` is the maximum residual norm associated with the sketched state. In the adaptive rank algorithm, the rank is updated using the rank update function $\mu(r, \tau)$ if the maximum residual norm exceeds the prescribed tolerance.

617 with appropriately specified initial conditions on \mathbf{v} and p . In (6.2), n is the outward normal vector on Γ_{out} ,
618 τ is the tangent vector

$$619 \quad \tau(x) = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} (x - x_0), \quad x \in \partial C,$$

620 and Re is the Reynold’s number. Problem (6.1) minimizes the power required to overcome the drag on C .
621 See [13, p. 87] for a comprehensive physical interpretation of this problem. The control action z defined in
622 (6.2e) is the angular velocity of the cylinder. The vector $\mathbf{v}_\infty := (1, 0)^\top$ is the freestream velocity profile
623 and the boundary condition (6.2c) is stress-free. Similar to [13], we generate initial conditions for (6.2) by
624 simulating (6.2) on the time interval $(-T_0, 0)$ for some $T_0 > 0$ starting with $\mathbf{v}(-T_0, \cdot)$ and $p(-T_0, \cdot)$ set to
625 the potential flow around C [4]. The first row of Figure 3 depicts the computed initial velocity \mathbf{v}_0 .

626 We discretized (6.2) in time with Implicit Euler and approximated the temporal integral in (6.1) with the
627 right end-point rule. Moreover, we discretized (6.2) in space using Q2–Q1 finite elements on the quadrilateral
628 mesh depicted in Figure 4. The mesh contains 2,672 elements and 2,762 vertices. For our results, we set the
629 time step $\delta t_n = 0.025$, $T_0 = 80$, $T = 20$, and $\text{Re} = 200$. We refer to [11, 13, 15] and the references therein
630 for partial verification of Assumption 1 for various flow control problems.

631 The second row of Figure 3 depicts the optimal vorticity at the final time $t = 20$ (left) and the velocity
632 field near the cylinder (right), while in the final row of Figure 3, we plot the computed optimal control.
633 As seen in Figure 3, the optimal control effectively eliminates the vortex shedding seen in the first row of
634 Figure 3 for the initial velocity. In Figure 5, we plot the sketching error averaged over 20 realizations and the
635 tail energy (ranks 1 through 200) for the uncontrolled state (left) and the optimal state (right). We see that
636 the decay in the sketching error and tail energy is roughly exponential, suggesting that our method should
637 only require modest storage.

638 We solved the discretized optimization problem using a Newton-based trust-region algorithm with fixed
639 sketch ranks $\{8, 16, 32, 64\}$ and using Algorithm 4.5 with the rank update function $\mu(r, \tau) = 2r$. The
640 performance of the fixed rank, adaptive rank, and full storage experiments is summarized in Table 3. For
641 each experiment, we set $\text{gtol} = 10^{-5}$ and $\text{maxit} = 40$. The only fixed-rank experiment to converge was
642 rank 64. However, the rank-32 experiment produced an objective function value that was within 6 digits
643 of the optimal value. The rank-32 experiment likely did not converge due to inaccuracies in the gradient.
644 For the adaptive algorithm, we started with the initial rank set to 8. The behavior of the rank updates as
645 well as the required gradient inexactness tolerances and computed residual norms are plotted in Figure 6.
646 Algorithm 4.5 required comparable computation as the full-storage approach, but reduced the memory by
647 a factor of $\zeta = 5.88$ at the final iteration. The memory burden could be further reduced by tuning κ_{grad} or
648 by choosing a less aggressive rank update function μ .

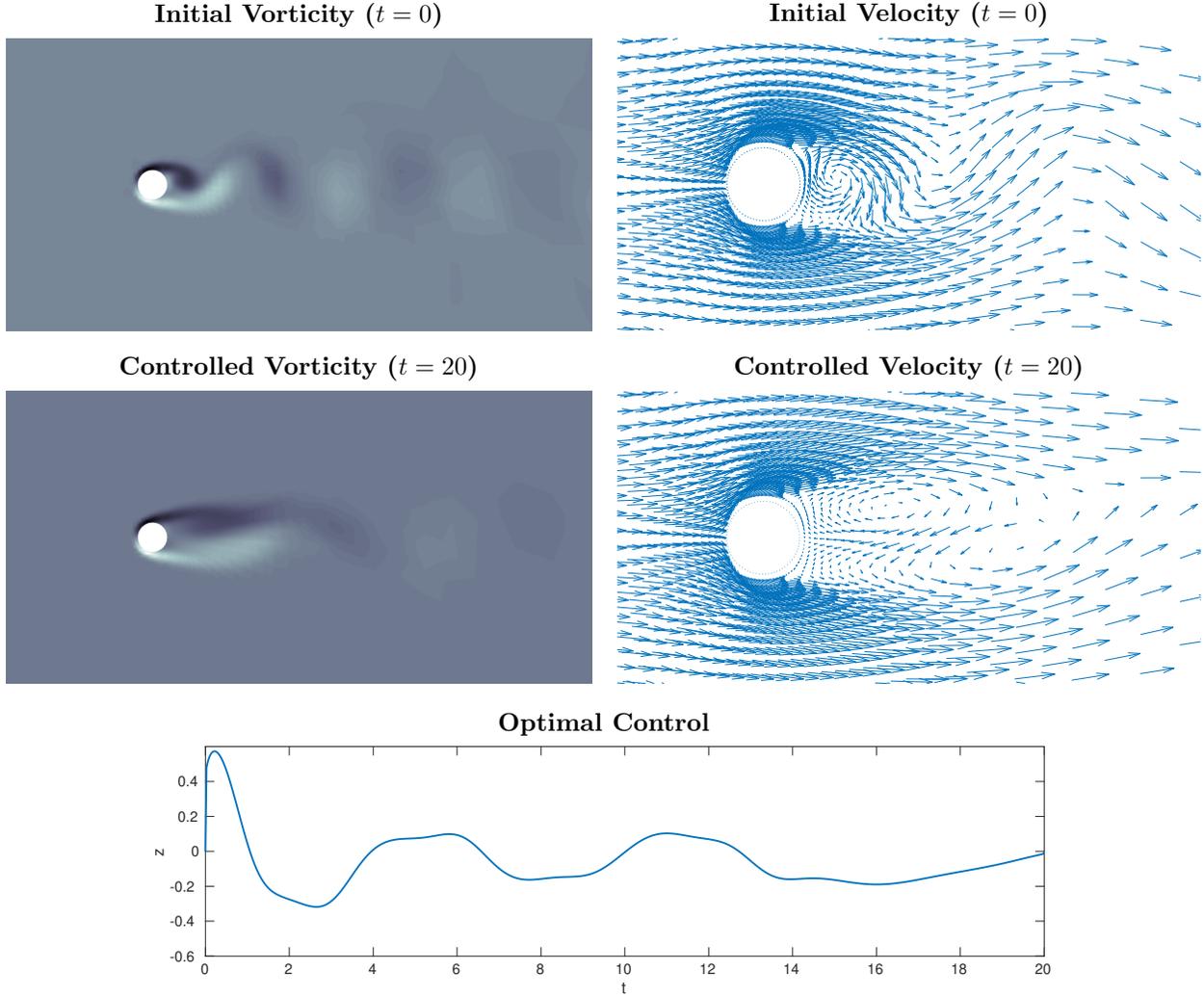


Figure 3: The initial velocity (first row) and the final, controlled velocity (second row). These images include the magnitude of vorticity ($\nabla \times v$) on the subdomain $[-5, 15] \times [-5, 5]$ (left) and velocity v on the subdomain $[-2, 6] \times [-2, 2]$ (right). The bottom row depicts the computed optimal control.

rank	objective	iteration	nfval	ngrad	iterCG	compression ζ
* 8	18.35919	40	41	15	136	45.44
*16	18.20003	40	41	33	897	23.35
*32	18.19779	40	41	31	236	11.80
64	18.19779	29	41	34	110	5.88
Adaptive	18.19779	23	24	24	121	5.88
Full	18.19779	29	30	24	107	---

Table 3: Algorithmic performance summary for the flow control example for fixed rank, adaptive rank and full storage: **objective** is the final objective function value, **iteration** is the total number of iterations, **nfval** is the number of function evaluations, **ngrad** is the number of gradient evaluations, **iterCG** is the total number of truncated CG iterations, and **compression ζ** is the compression factor. *The rank 8, 16, and 32 experiments terminated because they exceeded the maximum number of iterations.

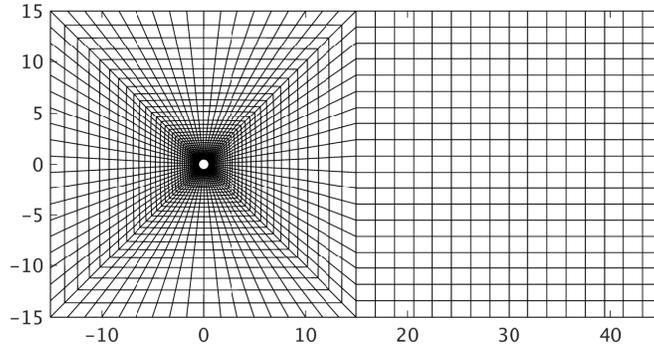


Figure 4: Quadrilateral mesh with 2672 elements and 2762 vertices.

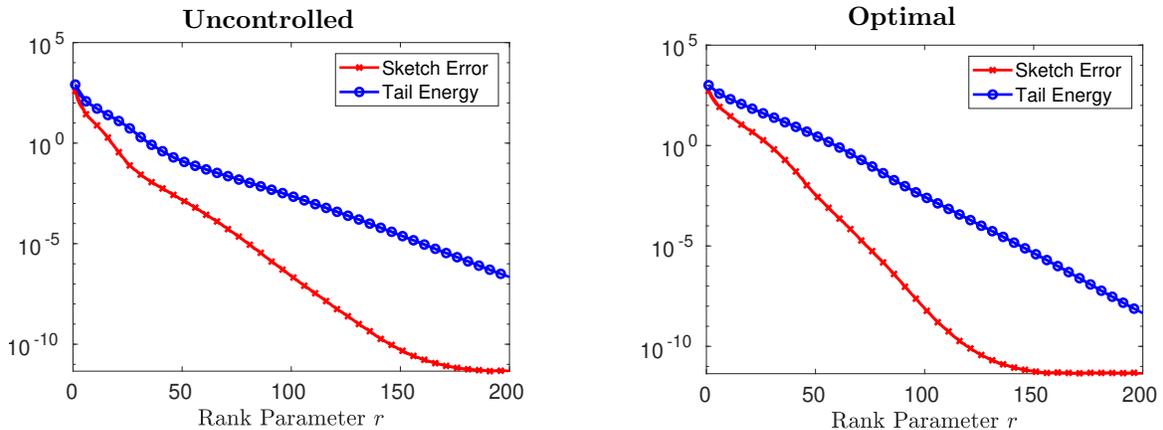


Figure 5: The sketching error averaged over 20 realizations and the tail energy for the uncontrolled state (left) and the optimal state (right) for the flow control example. Recall that the rank of the sketch is $k = 2r + 1$.

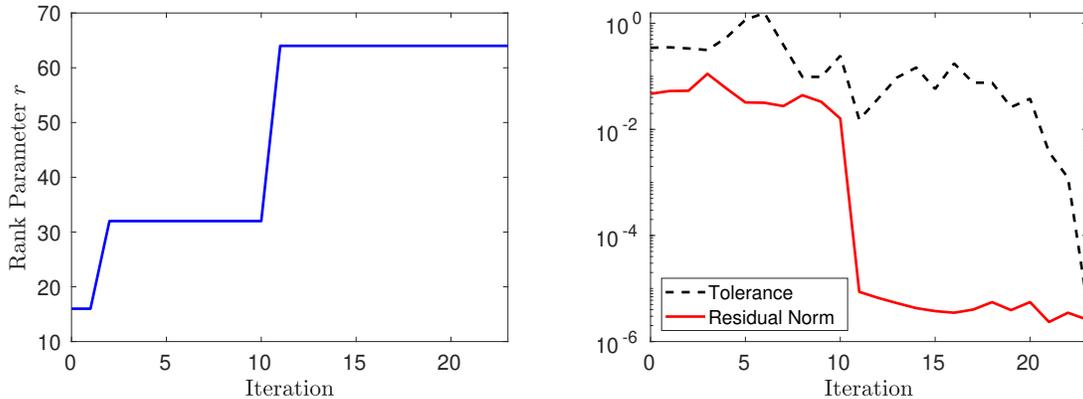


Figure 6: Inexact gradient behavior of Algorithm 4.5 applied to the flow control problem. Left: The sketch rank as a function of iteration. Right: The required gradient inexactness tolerance and computed residual norm as functions of iteration.

649 **Acknowledgments.** MU and RM were supported in part by DARPA Award FA8750-17-2-0101. DPK
 650 and RM (in part) were supported by the Laboratory Directed Research and Development program at Sandia

651 National Laboratories. Sandia National Laboratories is a multimission laboratory managed and operated by
 652 National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell
 653 International, Inc., for the U.S. Department of Energy’s National Nuclear Security Administration under
 654 contract DE-NA0003525. This paper describes objective technical results and analysis. Any subjective
 655 views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S.
 656 Department of Energy or the United States Government.

657

REFERENCES

- 658 [1] A. C. ANTOULAS, *Approximation of large-scale dynamical systems*, vol. 6 of Advances in Design and Control, Society for
 659 Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2005, <https://doi.org/10.1137/1.9780898718713>.
- 660 [2] S. R. ARRIDGE AND J. C. SCHOTLAND, *Optical tomography: forward and inverse problems*, Inverse Problems, 25 (2009),
 661 p. 123010, <https://doi.org/10.1088/0266-5611/25/12/123010>.
- 662 [3] G. AUPY, J. HERRMANN, P. HOVLAND, AND Y. ROBERT, *Optimal multistage algorithm for adjoint computation*, SIAM
 663 Journal on Scientific Computing, 38 (2016), pp. C232–C255, <https://doi.org/10.1137/1.9780898718713>.
- 664 [4] G. BATCHELOR, *An Introduction to Fluid Dynamics*, Cambridge Mathematical Library, Cambridge University Press, 2000,
 665 <https://doi.org/10.1017/CBO9780511800955>.
- 666 [5] C. BOUTSIDIS, D. P. WOODRUFF, AND P. ZHONG, *Optimal principal component analysis in distributed and streaming*
 667 *models*, in Proceedings of the Forty-Eighth Annual ACM Symposium on Theory of Computing, ACM, 2016, pp. 236–
 668 249, <https://doi.org/10.1137/1.9780898718713>.
- 669 [6] K. L. CLARKSON AND D. P. WOODRUFF, *Numerical linear algebra in the streaming model*, in Proceedings of the Forty-First
 670 ACM Symposium on Theory of Computing, Bethesda, 2009, <https://doi.org/10.1145/1536414.1536445>.
- 671 [7] A. CONN, N. GOULD, AND P. TOINT, *Trust Region Methods*, Society for Industrial and Applied Mathematics, 2000,
 672 <https://doi.org/10.1137/1.9780898719857>.
- 673 [8] L. DEDÈ, *Reduced basis method and a posteriori error estimation for parametrized linear-quadratic optimal control*
 674 *problems*, SIAM Journal on Scientific Computing, 32 (2010), pp. 997–1019, <https://doi.org/10.1137/090760453>,
 675 <http://link.aip.org/link/?SCE/32/997/1>.
- 676 [9] M. FAHL AND E. SACHS, *Reduced order modelling approaches to PDE-constrained optimization based on proper orthogonal*
 677 *decomposition*, in Large-Scale PDE-Constrained Optimization, L. T. Biegler, O. Ghattas, M. Heinkenschloss, and
 678 B. van Bloemen Waanders, eds., Lecture Notes in Computational Science and Engineering, Vol. 30, Heidelberg, 2003,
 679 Springer-Verlag, https://doi.org/10.1007/978-3-642-55508-4_16.
- 680 [10] A. GRIEWANK AND A. WALTHER, *Algorithm 799: Revolve: An implementation of checkpointing for the reverse or adjoint*
 681 *mode of computational differentiation*, ACM Trans. Math. Softw., 26 (2000), pp. 19–45, [https://doi.org/10.1145/](https://doi.org/10.1145/347837.347846)
 682 [347837.347846](https://doi.org/10.1145/347837.347846).
- 683 [11] M. GUNZBURGER, *Perspectives in Flow Control and Optimization*, Society for Industrial and Applied Mathematics, 2002,
 684 <https://doi.org/10.1137/1.9780898718720>.
- 685 [12] N. HALKO, P. G. MARTINSSON, AND J. A. TROPP, *Finding structure with randomness: probabilistic algorithms for*
 686 *constructing approximate matrix decompositions*, SIAM Rev., 53 (2011), pp. 217–288, [https://doi.org/10.1137/](https://doi.org/10.1137/090771806)
 687 [090771806](https://doi.org/10.1137/090771806).
- 688 [13] J.-W. HE, R. GLOWINSKI, R. METCALFE, A. NORDLANDER, AND J. PERIAUX, *Active Control and Drag Optimization for*
 689 *Flow Past a Circular Cylinder: I. Oscillatory Cylinder Rotation*, Journal of Computational Physics, 163 (2000),
 690 pp. 83 – 117, <https://doi.org/10.1006/jcph.2000.6556>.
- 691 [14] M. HEINKENSCHLOSS AND L. VICENTE, *Analysis of inexact trust-region sqp algorithms*, SIAM Journal on Optimization,
 692 12 (2002), pp. 283–302, <https://doi.org/10.1137/S1052623499361543>.
- 693 [15] M. HINZE, R. PINNAU, M. ULBRICH, AND S. ULBRICH, *Optimization with PDE Constraints*, Mathematical Modelling:
 694 Theory and Applications, Springer Netherlands, 2008, <https://doi.org/10.1007/978-1-4020-8839-1>.
- 695 [16] A. A. JALALI, C. S. SIMS, AND P. FAMOURI, *Reduced order systems*, vol. 343 of Lecture Notes in Control and Information
 696 Sciences, Springer-Verlag, Berlin, 2006, <https://doi.org/10.1007/11597018>.
- 697 [17] C. KAEBE, J. H. MARUHN, AND E. W. SACHS, *Adjoint-based monte carlo calibration of financial market models*, Fi-
 698 nance and Stochastics, 13 (2009), pp. 351–379, <https://doi.org/10.1007/s00780-009-0097-9>, [https://doi.org/10.1007/](https://doi.org/10.1007/s00780-009-0097-9)
 699 [s00780-009-0097-9](https://doi.org/10.1007/s00780-009-0097-9).
- 700 [18] A. D. KLOSE AND A. H. HIELSCHER, *Optical tomography using the time-independent equation of radiative transfer—part*
 701 *2: inverse model*, Journal of Quantitative Spectroscopy and Radiative Transfer, 72 (2002), pp. 715 – 732, [https://doi.org/10.1016/S0022-4073\(01\)00151-0](https://doi.org/10.1016/S0022-4073(01)00151-0).
 702 [https://doi.org/10.1016/S0022-4073\(01\)00151-0](https://doi.org/10.1016/S0022-4073(01)00151-0).
- 703 [19] D. KOURI, M. HEINKENSCHLOSS, D. RIDZAL, AND B. VAN BLOEMEN WAANDERS, *A trust-region algorithm with adaptive*
 704 *stochastic collocation for PDE optimization under uncertainty*, SIAM Journal on Scientific Computing, 35 (2013),
 705 pp. A1847–A1879, <https://doi.org/10.1137/120892362>.
- 706 [20] D. P. KOURI AND D. RIDZAL, *Inexact Trust-Region Methods for PDE-Constrained Optimization*, Springer New York, New
 707 York, NY, 2018, pp. 83–121, https://doi.org/10.1007/978-1-4939-8636-1_3.
- 708 [21] J. R. KREBS, J. E. ANDERSON, D. HINKLEY, R. NEELAMANI, S. LEE, A. BAUMSTEIN, AND M.-D. LACASSE, *Fast full-*
 709 *wavefield seismic inversion using encoded sources*, Geophysics, 74 (2009), pp. WCC177–WCC188, [https://doi.org/](https://doi.org/10.1190/1.3230502)
 710 [10.1190/1.3230502](https://doi.org/10.1190/1.3230502).
- 711 [22] M.-D. LACASSE, L. WHITE, H. DENLI, AND L. QIU, *Full-Wavefield Inversion: An Extreme-Scale PDE-Constrained Opti-*
 712 *mization Problem*, Springer New York, New York, NY, 2018, pp. 205–255, <https://doi.org/10.1007/978-1-4939-8636-1>.

- 713 6.
714 [23] C. LEE, J. KIM, AND H. CHOI, *Suboptimal control of turbulent channel flow for drag reduction*, Journal of Fluid Mechanics,
715 358 (1998), p. 245–258, <https://doi.org/10.1017/S002211209700815X>.
716 [24] M. W. MAHONEY, *Randomized algorithms for matrices and data*, Found. Trends Mach. Learning, 3 (2011), pp. 123–224,
717 <https://doi.org/10.1561/22000000035>.
718 [25] P. STUMM AND A. WALTHER, *New algorithms for optimal online checkpointing*, SIAM Journal on Scientific Computing,
719 32 (2010), pp. 836–854, <https://doi.org/10.1137/080742439>.
720 [26] Y. SUN, Y. GUO, C. LUO, J. A. TROPP, AND M. UDELL, *Low-rank tucker approximation of a tensor from streaming data*,
721 arXiv preprint arXiv:1904.10951, (2019), <https://arxiv.org/abs/1904.10951>.
722 [27] Y. SUN, Y. GUO, J. A. TROPP, AND M. UDELL, *Tensor random projection for low memory dimension reduction*,
723 in NeurIPS Workshop on Relational Representation Learning, 2018, [https://r2learning.github.io/assets/papers/](https://r2learning.github.io/assets/papers/CameraReadySubmission%2041.pdf)
724 [CameraReadySubmission%2041.pdf](https://r2learning.github.io/assets/papers/CameraReadySubmission%2041.pdf).
725 [28] A. TARANTOLA, *Linearized inversion of seismic reflection data*, Geophysical Prospecting, 32 (1984), pp. 998–1015, <https://doi.org/10.1111/j.1365-2478.1984.tb00751.x>.
726 [29] J. A. TROPP, A. YURTSEVER, M. UDELL, AND V. CEVHER, *Fixed-rank approximation of a positive-semidefinite matrix*
727 *from streaming data*, in Adv. Neural Information Processing Systems 30 (NIPS), Long Beach, Dec. 2017.
728 [30] J. A. TROPP, A. YURTSEVER, M. UDELL, AND V. CEVHER, *Practical sketching algorithms for low-rank matrix approxi-*
729 *mation*, SIAM J. Matrix Anal. Appl., 38 (2017), pp. 1454–1485, <https://doi.org/10.1137/17M1111590>.
730 [31] J. A. TROPP, A. YURTSEVER, M. UDELL, AND V. CEVHER, *Streaming low-rank matrix approximation with an application*
731 *to scientific simulation*, SIAM Journal on Scientific Computing, (2019), <https://arxiv.org/abs/1902.08651>.
732 [32] Q. WANG, P. MOIN, AND G. IACCARINO, *Minimal repetition dynamic checkpointing algorithm for unsteady adjoint calcu-*
733 *lation*, SIAM Journal on Scientific Computing, 31 (2009), pp. 2549–2567, <https://doi.org/10.1137/080727890>.
734 [33] M. WARNER AND L. GUASCH, *Adaptive waveform inversion: Theory*, GEOPHYSICS, 81 (2016), pp. R429–R445, <https://doi.org/10.1190/geo2015-0387.1>.
735 [34] D. P. WOODRUFF, *Sketching as a tool for numerical linear algebra*, Found. Trends Theor. Comput. Sci., 10 (2014),
736 pp. iv+157, <https://doi.org/10.1561/04000000060>.
737 [35] F. WOOLFE, E. LIBERTY, V. ROKHLIN, AND M. TYGERT, *A fast randomized algorithm for the approximation of matrices*,
738 Appl. Comput. Harmon. Anal., 25 (2008), pp. 335–366, <https://doi.org/10.1016/j.acha.2007.12.002>.
739 [36] M. J. ZAHR, K. T. CARLBERG, AND D. P. KOURI, *An efficient, globally convergent method for optimization under*
740 *uncertainty using adaptive model reduction and sparse grids*, arXiv e-prints, (2018), [https://arxiv.org/abs/1811.](https://arxiv.org/abs/1811.00177)
741 [00177](https://arxiv.org/abs/1811.00177).
742
743

744 **Appendix A. Sketching Routines.** In this appendix, we provide pseudo-code for the sketching
745 algorithms described throughout the paper. In particular, we first present the abstract sketch class and then
746 describe the methods required to apply the sketch-based approximation of the Hessian to a vector.

Algorithm A.1 Sketch class and methods

```

1: class SKETCH
2:   member variables  $k, s$                                 sketch parameters
3:   member variables  $\Upsilon, \Omega, \Phi, \Psi$            random test matrices
4:   member variables  $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$                  sketch matrices
5:   member variables  $\mathbf{Q}, \mathbf{W}$                          low rank factors
6:   member variables  $\text{rec}$                              reconstruction flag
7:   function SKETCH( $M, N, \text{rank} = r$ )                 Sketch class constructor
8:     Initialize!( $M, N, \text{rank} = r$ )
9:   function INITIALIZE!( $M, N, \text{rank} = r$ )           Sketch class initializer
10:    reconstructed  $\leftarrow$  FALSE.
11:     $k \leftarrow 2r + 1, s \leftarrow 2k + 1$ 
12:     $\Upsilon \leftarrow \text{randn}(k, M), \Omega \leftarrow \text{randn}(k, N)$    Test matrix for range and co-range
13:     $\Phi \leftarrow \text{randn}(s, M), \Psi \leftarrow \text{randn}(s, N)$    Test matrices for core
14:     $\mathbf{X} \leftarrow \text{zeros}(k, N), \mathbf{Y} \leftarrow \text{zeros}(M, k)$    Approximation sketch of zero matrix
15:     $\mathbf{Z} \leftarrow \text{zeros}(s, s)$ 
16:     $\mathbf{Q} \leftarrow \text{zeros}(M, k), \mathbf{W} \leftarrow \text{zeros}(k, N)$    Low rank factors
17:   function COLUMNUPDATE!( $\mathbf{h}, n, \theta = 1.0, \eta = 1.0$ )   Update sketch matrices
18:     $\mathbf{X} \leftarrow \theta \mathbf{X} + \eta(\Upsilon \mathbf{h}) \mathbf{e}_n^*$ 
19:     $\mathbf{Y} \leftarrow \theta \mathbf{Y} + \eta \mathbf{h}(\Omega \mathbf{e}_n)^*$ 
20:     $\mathbf{Z} \leftarrow \theta \mathbf{Z} + \eta(\Phi \mathbf{h})(\Psi \mathbf{e}_n)^*$ 
21:   function RECONSTRUCT!( )                             Reconstruct low rank factors
22:     $(\mathbf{Q}, \mathbf{R}_2) \leftarrow \text{qr}(\mathbf{Y}, 0)$ 
23:     $(\mathbf{P}, \mathbf{R}_1) \leftarrow \text{qr}(\mathbf{X}^*, 0)$ 
24:     $\mathbf{C} \leftarrow ((\Phi \mathbf{Q}) \setminus \mathbf{Z}) / ((\Psi \mathbf{P})^*)$ 
25:     $\mathbf{W} \leftarrow \mathbf{C} \mathbf{P}^*$ 
26:    reconstructed  $\leftarrow$  TRUE
27:   function COLUMN(j)                                   Reconstruct a single column
28:     if reconstructed then
29:       return  $\mathbf{Q} \cdot \mathbf{W}[:, j]$ 

```

Algorithm A.2 Compute residual norm for control \mathbf{Z} .

Input: A control iterate $\mathbf{Z} \in \mathbb{R}^{m \times N}$ and sketch object $\{\mathbf{U}\}_r$ for state $\mathbf{U} \in \mathbb{R}^{MN \times 1}$

Output: Residual norm : $\text{rnorm} = \|c(\widehat{\mathbf{U}}, \mathbf{Z})\|^2$

Storage: $\mathcal{O}(r(M + N))$

```

1: function RESIDUALNORM( $\{\mathbf{U}\}_r, \mathbf{Z}$ )
2:    $(\mathbf{u}_{\text{old}}, \text{rnorm}) \leftarrow (\{\mathbf{U}\}_r.\text{COLUMN}(N), 0)$ 
3:   for  $n \leftarrow N$  to 1 do
4:      $\mathbf{u}_{\text{new}} \leftarrow \{\mathbf{U}\}_r.\text{COLUMN}(n - 1)$ 
5:      $\text{rnorm} \leftarrow \text{rnorm} + \|c_n(\mathbf{u}_{\text{new}}, \mathbf{u}_{\text{old}}, \mathbf{z}_n)\|^2$ 
6:      $\mathbf{u}_{\text{old}} \leftarrow \mathbf{u}_{\text{new}}$ 
7:   return rnorm

```

Algorithm A.3 Solve adjoint equation.

Input: Control $\mathbf{Z} \in \mathbb{R}^{m \times N}$ and sketch objects:

$\{\mathbf{U}\}_r$ for state $\mathbf{U} \in \mathbb{R}^{M \times N}$
 $\{\mathbf{\Lambda}\}_r$ for adjoint $\mathbf{\Lambda} \in \mathbb{R}^{M \times N}$

Output: Updated adjoint sketch object $\{\mathbf{\Lambda}\}_r$

Storage: $\mathcal{O}((r_1 + r_2)(M + N))$ for adjoint rank parameter $r_2 \leq \min\{M, N\}$

1: **function** SOLVEADJOINT! ($\{\mathbf{\Lambda}\}_r, \{\mathbf{U}\}_r, \mathbf{Z}$)
2: $(\mathbf{u}_{\text{curr}}, \mathbf{u}_{\text{next}}) \leftarrow (\{\mathbf{U}\}_r.\text{COLUMN}(N - 1), \{\mathbf{U}\}_r.\text{COLUMN}(N))$
3: Solve the adjoint equation at index N for $\boldsymbol{\lambda}_{\text{next}}$,

$$\mathbf{d}_2 c_N(\mathbf{u}_{\text{curr}}, \mathbf{u}_{\text{next}}, \mathbf{z}_N) \boldsymbol{\lambda}_{\text{next}} = \mathbf{d}_2 f_N(\mathbf{u}_{\text{curr}}, \mathbf{u}_{\text{next}}, \mathbf{z}_N)$$

4: $\{\mathbf{\Lambda}\}_r.\text{COLUMNUPDATE!}(\boldsymbol{\lambda}_{\text{next}}, N)$
5: **for** $n = N - 1$ to 1 **do**
6: **if** $n = 1$ **then**
7: $\mathbf{u}_{\text{prev}} \leftarrow \mathbf{u}_0$
8: **else**
9: $\mathbf{u}_{\text{prev}} \leftarrow \{\mathbf{U}\}_r.\text{COLUMN}(n - 1)$
10: Solve the adjoint equation at index n for $\boldsymbol{\lambda}_{\text{curr}}$,

$$\begin{aligned} \mathbf{d}_2 c_n(\mathbf{u}_{\text{prev}}, \mathbf{u}_{\text{curr}}, \mathbf{z}_n) \boldsymbol{\lambda}_{\text{curr}} &= \mathbf{d}_2 f_n(\mathbf{u}_{\text{prev}}, \mathbf{u}_{\text{curr}}, \mathbf{z}_n) + \mathbf{d}_1 f_{n+1}(\mathbf{u}_{\text{curr}}, \mathbf{u}_{\text{next}}, \mathbf{z}_{n+1}) \\ &\quad - \mathbf{d}_1 c_{n+1}(\mathbf{u}_{\text{curr}}, \mathbf{u}_{\text{next}}, \mathbf{z}_{n+1}) \boldsymbol{\lambda}_{\text{next}} \end{aligned}$$

11: $\{\mathbf{\Lambda}\}_r.\text{COLUMNUPDATE!}(\boldsymbol{\lambda}_{\text{curr}}, n)$
12: $(\mathbf{u}_{\text{next}}, \mathbf{u}_{\text{curr}}, \boldsymbol{\lambda}_{\text{next}}) \leftarrow (\mathbf{u}_{\text{curr}}, \mathbf{u}_{\text{prev}}, \boldsymbol{\lambda}_{\text{curr}})$

Algorithm A.4 Solve state sensitivity equation.

Input: Control $\mathbf{Z} \in \mathbb{R}^{m \times N}$, direction vector $\mathbf{V} \in \mathbb{R}^{m \times N}$, and sketch objects:

$\{\mathbf{U}\}_r$ for state $\mathbf{U} \in \mathbb{R}^{M \times N}$
 $\{\mathbf{W}\}_r$ for state sensitivity $\mathbf{W} \in \mathbb{R}^{M \times N}$

Output: Updated state sensitivity sketch object $\{\mathbf{W}\}_r$

Storage: $\mathcal{O}((r_1 + r_3)(M + N))$ for state sensitivity rank parameter $r_3 \leq \min\{M, N\}$

1: **function** SOLVESTATESENSITIVITY! ($\{\mathbf{W}\}_r, \{\mathbf{U}\}_r, \mathbf{Z}, \mathbf{V}$)
2: **for** $n = 1$ to N **do**
3: **if** $n = 1$ **then**
4: $(\mathbf{u}_{\text{prev}}, \mathbf{u}_{\text{curr}}, \mathbf{w}_{\text{prev}}) \leftarrow (\mathbf{u}_0, \{\mathbf{U}\}_r.\text{COLUMN}(1), \mathbf{0})$
5: **else**
6: $(\mathbf{u}_{\text{prev}}, \mathbf{u}_{\text{curr}}, \mathbf{w}_{\text{prev}}) \leftarrow (\mathbf{u}_{\text{curr}}, \{\mathbf{U}\}_r.\text{COLUMN}(n), \mathbf{w}_{\text{curr}})$
7: Solve the state sensitivity equation at index n for \mathbf{w}_{curr}

$$\mathbf{d}_2 c_n(\mathbf{u}_{\text{prev}}, \mathbf{u}_{\text{curr}}, \mathbf{z}_n) \mathbf{w}_{\text{curr}} = \mathbf{d}_3 c_n(\mathbf{u}_{\text{prev}}, \mathbf{u}_{\text{curr}}, \mathbf{z}_n) \mathbf{v}_n - \mathbf{d}_1 c_n(\mathbf{u}_{\text{prev}}, \mathbf{u}_{\text{curr}}, \mathbf{z}_n) \mathbf{w}_{\text{prev}}$$

8: $\{\mathbf{W}\}_r.\text{COLUMNUPDATE!}(\mathbf{w}_{\text{curr}}, n)$

Algorithm A.5 Apply fixed-rank Hessian approximation to a vector.

Input: Control $\mathbf{Z} \in \mathbb{R}^{m \times N}$, sketch object for state $\{\mathbf{U}\}_r$, direction $\mathbf{V} \in \mathbb{R}^{m \times N}$,
and rank parameters $r_2, r_3 \leq \min\{M, N\}$

Output: Application of approximate Hessian to vector \mathbf{V} , $\tilde{\mathbf{H}} \approx \nabla^2 f(\mathbf{Z})\mathbf{V}$

Storage: $\mathcal{O}((r_1 + r_2 + r_3)(M + N))$

```

1: function APPLYFIXEDRANKHESSIAN( $\mathbf{Z}$ ,  $\{\mathbf{U}\}_r$ ,  $\mathbf{V}$ ,  $r_2$ ,  $r_3$ )
2:    $\{\mathbf{A}\}_r \leftarrow \text{SKETCH}(M, N, \text{rank} = r_2)$            Initialize adjoint sketch object
3:   SOLVEADJOINT!( $\{\mathbf{A}\}_r, \{\mathbf{U}\}_r, \mathbf{Z}$ )                 Solve adjoint equation
4:    $\{\mathbf{A}\}_r.\text{RECONSTRUCT!}()$                            Get low-rank factors for adjoint
5:    $\{\mathbf{W}\}_r \leftarrow \text{SKETCH}(M, N, \text{rank} = r_3)$        Initialize state sensitivity sketch object
6:   SOLVESTATESENSITIVITY!( $\{\mathbf{W}\}_r, \{\mathbf{U}\}_r, \mathbf{Z}, \mathbf{V}$ )   Solve state sensitivity equation
7:    $\{\mathbf{W}\}_r.\text{RECONSTRUCT!}()$                            Get low-rank factors for state sensitivity
8:    $\tilde{\mathbf{H}} \leftarrow \text{ApplyHessian}(\{\mathbf{W}\}_r, \{\mathbf{A}\}_r, \{\mathbf{U}\}_r, \mathbf{Z}, \mathbf{V})$  Apply Hessian to  $\mathbf{V}$ 
9:   return  $\tilde{\mathbf{H}}$ 

```

Algorithm A.6 Apply Hessian to a vector using sketching.

Input: Control $\mathbf{Z} \in \mathbb{R}^{m \times N}$, direction vector $\mathbf{V} \in \mathbb{R}^{m \times N}$, and sketch objects:

$\{\mathbf{U}\}_r$ for state $\mathbf{U} \in \mathbb{R}^{M \times N}$
 $\{\mathbf{A}\}_r$ for adjoint $\mathbf{A} \in \mathbb{R}^{M \times N}$
 $\{\mathbf{W}\}_r$ for state sensitivity $\mathbf{W} \in \mathbb{R}^{M \times N}$

Output: Application of approximate Hessian to vector \mathbf{V} , $\mathbf{H} \approx \nabla^2 f(\mathbf{Z})\mathbf{V}$

Storage: $\mathcal{O}((r_1 + r_2 + r_3)(M + N))$

```

1: function APPLYHESSIAN( $\{\mathbf{W}\}_r, \{\mathbf{A}\}_r, \{\mathbf{U}\}_r, \mathbf{Z}^{(k)}, \mathbf{V}$ )
2:   ( $\mathbf{u}_{\text{curr}}, \mathbf{u}_{\text{next}}, \boldsymbol{\lambda}_{\text{next}}$ )  $\leftarrow$  ( $\{\mathbf{U}\}_r.\text{COLUMN}(N-1), \{\mathbf{U}\}_r.\text{COLUMN}(N), \{\mathbf{A}\}_r.\text{COLUMN}(N)$ )
3:   ( $\mathbf{w}_{\text{curr}}, \mathbf{w}_{\text{next}}$ )  $\leftarrow$  ( $\{\mathbf{W}\}_r.\text{COLUMN}(N-1), \{\mathbf{W}\}_r.\text{COLUMN}(N)$ )
4:   Solve the adjoint sensitivity equation at index  $N$  for  $\mathbf{p}_{\text{next}}$ ,

   ( $\mathbf{d}_2 c_N(\mathbf{u}_{\text{curr}}, \mathbf{u}_{\text{next}}, \mathbf{z}_N)$ ) $^*$  $\mathbf{p}_{\text{next}} = -\mathbf{d}_{2,3} L_N(\mathbf{u}_{\text{curr}}, \mathbf{u}_{\text{next}}, \mathbf{z}_N, \boldsymbol{\lambda}_{\text{next}})\mathbf{v}_N$ 
   +  $\mathbf{d}_{2,2} L_N(\mathbf{u}_{\text{curr}}, \mathbf{u}_{\text{next}}, \mathbf{z}_N, \boldsymbol{\lambda}_{\text{next}})\mathbf{w}_{\text{next}} + \mathbf{d}_{2,1} L_N(\mathbf{u}_{\text{curr}}, \mathbf{u}_{\text{next}}, \mathbf{z}_N, \boldsymbol{\lambda}_{\text{next}})\mathbf{w}_{\text{curr}}$ 

5:   Apply Hessian of Lagrangian at index  $N$ ,

    $\mathbf{h}_N = \mathbf{d}_{3,3} L_N(\mathbf{u}_{\text{curr}}, \mathbf{u}_{\text{next}}, \mathbf{z}_N, \boldsymbol{\lambda}_{\text{next}})\mathbf{v}_n - \mathbf{d}_{3,1} L_N(\mathbf{u}_{\text{curr}}, \mathbf{u}_{\text{next}}, \mathbf{z}_N, \boldsymbol{\lambda}_{\text{next}})\mathbf{w}_{\text{curr}}$ 
   -  $\mathbf{d}_{3,2} L_N(\mathbf{u}_{\text{curr}}, \mathbf{u}_{\text{next}}, \mathbf{z}_N, \boldsymbol{\lambda}_{\text{next}})\mathbf{w}_{\text{next}} + (\mathbf{d}_3 c_N(\mathbf{u}_{\text{curr}}, \mathbf{u}_{\text{next}}, \mathbf{z}_N))$  $^*$  $\mathbf{p}_{\text{next}}$ 

6:   for  $n = N - 1$  to 1 do
7:     Solve the adjoint sensitivity equation at index  $n$  for  $\mathbf{p}_{\text{curr}}$ ,

   ( $\mathbf{d}_2 c_n(\mathbf{u}_{\text{prev}}, \mathbf{u}_{\text{curr}}, \mathbf{z}_n)$ ) $^*$  $\mathbf{p}_{\text{curr}} = -(\mathbf{d}_1 c_{n+1}(\mathbf{u}_{\text{curr}}, \mathbf{u}_{\text{next}}, \mathbf{z}_n))$  $^*$  $\mathbf{p}_{\text{next}}$ 
   -  $\mathbf{d}_{2,3} L_n(\mathbf{u}_{\text{prev}}, \mathbf{u}_{\text{curr}}, \mathbf{z}_n, \boldsymbol{\lambda}_{\text{curr}})\mathbf{v}_n - \mathbf{d}_{1,3} L_{n+1}(\mathbf{u}_{\text{curr}}, \mathbf{u}_{\text{next}}, \mathbf{z}_n, \boldsymbol{\lambda}_{\text{next}})\mathbf{v}_{n+1}$ 
   +  $\mathbf{d}_{2,2} L_n(\mathbf{u}_{\text{prev}}, \mathbf{u}_{\text{curr}}, \mathbf{z}_n, \boldsymbol{\lambda}_{\text{curr}})\mathbf{w}_{\text{curr}} + \mathbf{d}_{1,2} L_{n+1}(\mathbf{u}_{\text{curr}}, \mathbf{u}_{\text{next}}, \mathbf{z}_n, \boldsymbol{\lambda}_{\text{next}})\mathbf{w}_{\text{next}}$ 
   +  $\mathbf{d}_{2,1} L_n(\mathbf{u}_{\text{prev}}, \mathbf{u}_{\text{curr}}, \mathbf{z}_n, \boldsymbol{\lambda}_{\text{curr}})\mathbf{w}_{\text{prev}} + \mathbf{d}_{1,1} L_{n+1}(\mathbf{u}_{\text{curr}}, \mathbf{u}_{\text{next}}, \mathbf{z}_n, \boldsymbol{\lambda}_{\text{next}})\mathbf{w}_{\text{curr}}$ .

8:     Apply Hessian of Lagrangian at index  $n$ 

    $\mathbf{h}_n \leftarrow \mathbf{d}_{3,3} L_n(\mathbf{u}_{\text{prev}}, \mathbf{u}_{\text{curr}}, \mathbf{z}_n, \boldsymbol{\lambda}_{\text{curr}})\mathbf{v}_n - \mathbf{d}_{3,1} L_n(\mathbf{u}_{\text{prev}}, \mathbf{u}_{\text{curr}}, \mathbf{z}_n, \boldsymbol{\lambda}_{\text{curr}})\mathbf{w}_{\text{prev}}$ 
   -  $\mathbf{d}_{3,2} L_n(\mathbf{u}_{\text{prev}}, \mathbf{u}_{\text{curr}}, \mathbf{z}_n, \boldsymbol{\lambda}_{\text{curr}})\mathbf{w}_{\text{curr}} + (\mathbf{d}_3 c_n(\mathbf{u}_{\text{prev}}, \mathbf{u}_{\text{curr}}, \mathbf{z}_n))$  $^*$  $\mathbf{p}_{\text{curr}}$ 

9:     ( $\mathbf{u}_{\text{next}}, \mathbf{u}_{\text{curr}}, \boldsymbol{\lambda}_{\text{next}}$ )  $\leftarrow$  ( $\mathbf{u}_{\text{curr}}, \mathbf{u}_{\text{prev}}, \boldsymbol{\lambda}_{\text{curr}}$ )
10:    ( $\mathbf{w}_{\text{next}}, \mathbf{w}_{\text{curr}}, \mathbf{p}_{\text{next}}$ )  $\leftarrow$  ( $\mathbf{w}_{\text{curr}}, \mathbf{w}_{\text{prev}}, \mathbf{p}_{\text{curr}}$ )
11:   return  $\mathbf{H} = [\mathbf{h}_1, \dots, \mathbf{h}_N]$ 

```
