We would like to thank the reviewers for the feedback they provided. Following is our response.

Reviewer 1: The attached code will be open-sourced and available on GitHub in Python and MATLAB. Besides control and robotic applications, our method is relevant to machine learning as it can be applied without change to Koopman operators, which are increasingly gaining attention in deep learning. For example, recent work by Dogra et al. (2020) uses Koopman operators to optimize training of neural network methods; also work by Yeung et al. (2019) learns deep neural network models for Koopman operators of nonlinear dynamical systems. Imposing stability on Koopman operators that are based on basis functions learned via deep learning could lead to a scheme that combines the benefits of linear representations with the predictive power of neural networks in data-driven prediction and control.

Reviewer 2: The proposed algorithm can be readily applied to Koopman operators. We will include an experimental example of a robot learning how to push a planar block—an open problem in robot learning—illustrating both the need for stable operators and the performance when one closes the feedback loop. We will include animations of the results in Fig. 3 and Fig. 6 with our code release. The constraints on S, O, and C are enforced at every iterate. This is shown in the provided code. We will include the algorithmic steps and details about the projections in the revised manuscript and in the code repository. The 'best error frequency' metric credits all schemes that achieve the top performance such that curves may add up to more than 100% when two or more methods have obtained the same best error. The Franka control is the joint velocities.

Reviewer 3: With respect to the novelty of the algorithm, we use the characterization of the stable matrices to derive *new* gradient descent directions to compute matrices that are stable *and* minimize the reconstruction error with respect to data measurements. We will point out in the paper that the LS solution degrades over time and add supporting videos of the results in Fig. 3.

To our knowledge, this is the first work that tests the control performance of stable LDS. CG has been formulated but not evaluated for control tasks and is not straightforward that WLS can be implemented for such applications, as the results in our paper suggest. There is previous work by Ng et al. (2004) on rejecting unstable controllers in online learning, but the suggested method does not generate stable models and control. Instead it can only filter solutions, which would not be helpful if the implemented controller is always unstable. With respect to space complexity, the matrix operations and sizes in our algorithm require at most $O(n^2)$ memory. The CG and WLS algorithms use a quadratic programming solution with n^2 state dimension, which generates matrices of dimensions n^4 hence a space complexity of $O(n^4)$.

We attribute the difference in performance across the datasets to the type of motion represented in them. The UCLA and DynTex datasets contain periodic, stable motion that is more suitable for the algorithms examined; the UCSD dataset includes highway traffic with cars entering and exiting the frame, manifesting a discontinuous motion. Consequently, as the subspace dimension increases and the finer details are more prevalent, the ground truth lies further away from stable solutions leading to a higher possible reconstruction error. We will include results for higher dimensions. Note, however, that the subspace dimension cannot exceed the number of frames used for training. This is due to the SVD decomposition (used for dimensionality reduction) that is required for the WLS algorithm. Further, the computational time for CG and WLS is prohibitively high for online applications. We provide an example in the Table.

	SOC	CG	WLS
	`steam` (r = 80)		
$e(\hat{A})$	**6.38**	25.21	10.98
time (s)	**5.45**	1146.23	456.11

We will include results for the coffee cup sequence and also modify the transition to Section 2.1 (e.g., by mentioning that the literature on LDS has traditionally considered only no-input systems, which we will review first.)

Reviewer 4: Learning stable LDS will be most useful to systems that are known a priori to be stable. In practice, many systems have unforced dynamics that are naturally stable (asymptotically or in the sense of Lyapunov), such as mechanical systems, fluids, and electrical systems.

The novelty of this work lies in ensuring stability from the **first** iteration step of the optimization while achieving superior error performance and a lower space complexity. On the other hand, CG and WLS start from the LS solution and achieve stability only after converging. This difference can become crucial in online applications and time-sensitive tasks.

We will add references, among others, to stability preserving subspace identification methods (SPSIM) (i.e., Chui et al. (1996), Van Gestel et al. (2001), and Miller et al. (2013)). We omitted a comparison to such methods, given that WLS (2016) and CG (2007) compared and outperformed the seminal work in SPSIM by Lacy and Bernstein (2003) in terms of error, scalability, and execution time.

Reviewer 5: The product formulation of a matrix in SOC form bounds the maximum eigenvalues by constraining the B matrix to have norm at most one. The detailed proof of the expression can be found in the original paper. The proposed algorithm enforces stability on the learned matrix. It does not rely on any assumptions (contrary to WLS that assumes time continuity of data) and is applicable to any task that seeks a stable matrix that maps data from X to Y.