REPORT OF THE FIRST REFEREE – EM12134/O'CONNOR

This paper presents three methods for solving inverse problems DAL, SBI, and QRM. The paper is very well written, with a thorough introduction and a clear presentation of the methods and results. While I am generally in favor of this paper being accepted, I have some comments I would like to see addressed first.

1. In the introduction it is said that '...-chi(x,0) which constitutes the gradient or functional derivative of...'. This is true for how you have defined the Lagrangian, but is not necessarily the case in general. For example, a sign change of the adjoint variable in the Lagrangian would change the sign here so that the gradient would be + chi(x,0). Could this be rephrased to be more general until you define the Lagrangian.

   - Section I, edit: "Backward integration of the adjoint equations allows us to compute the gradient or functional derivative"

2. In the introduction it is said that DALs shortcomings are due to 'missing terms'. I'm not sure of this phrasing. Maybe it is the cost functional that is the issue, as you show in section 5.

   - We rewrote this paragraph, see our reply to item 3. Instead saying that DALs shortcomings are due to 'missing terms', we claim that DALs shortcomings can be alleviated by appending the adjoint with extra terms.
   - We have updated every instance of the phrase "missing terms" –¿ "additional terms" or "additional advective terms".

3. You also say that the issues you encounter with DAL would not be fixed by using a discrete adjoint. This is not demonstrated in the paper. I agree that the formulation would be the same with regards to the 'missing terms' and analysis of section 5. However, might the issues with DAL encountered in this paper potentially stem from an incorrect gradient being calculated due to discretization errors from the continuous adjoint formulation?

   - We rewrote this paragraph as follows: "In practice, researchers often customize the DAL algorithm to improve performance in specific cases. For example, [15] used the conjugate-gradient method with a discrete adjoint formulation to optimize initial conditions subject to norm constraints. For the PNS inverse problem, [9] combined Principle Orthogonal Decomposition (POD) with four-dimensional variational assimilation (4DVAR), an optimal control technique analogous to DAL. In both examples, researchers reduced the computational cost of each DAL iteration by reformulating the adjoint while simultaneously reducing the required number of DAL iterations by implementing Quasi-Newton minimization routines. For this investigation, we take a different approach. Instead of reformulating the adjoint or changing the way in which gradient information is used, we will later demonstrate that DAL's shortcomings in the context of Navier–Stokes inversion can be mitigated by appending the adjoint system with additional advective terms."

4. No verification of the adjoint method is given, so readers cannot be sure of how accurate the gradient is. Perhaps you could add a verification of the gradient in a specific direction returned by the adjoint for a short time horizon T where it should be reasonably accurate for a continuous formulation.

   - Evaluating the accuracy of gradients in high-dimensional spaces is challenging. In our 1D KdV-Burgers example problem, we can approximate the gradient of $\mathcal{J}_f^u$ using the `scipy.optimize.approx_fprime` function, which relies on the finite difference approximation. We must choose a point in the space of initial conditions to compare gradients. For this, we use the simple backward integration (SBI) initial guess which is described in section III as well as the appendix. We find that the adjoint-obtained gradient has a relative L_2 error $<$ 1e-3 and absolute L_2 error $<$ 2e-4. Their magnitudes have a ratio $\sim$ 1.00005 and the dot product (projection) of the normalized gradients $\sim$ 0.9999995.
   - For the 2D Navier–Stokes experiment, we again compute the gradient of $\mathcal{J}_f^{\mathbf{u}}$ using the `scipy.optimize.approx_fprime` method. To reduce the computational cost of this task, we changed the following parameters: $t_f = 5$, $Nx \times Nz = 32 \times 64$, $\Delta t = 0.005$, $\mathrm{Re} \equiv \nu^{-1} = 5,000$. We use the SBI initial guess as the point at which to compare gradients. In this case, the relative L_2 error $<$ 1e-2, absolute L_2

error < 1e-4. Their magnitudes have a ratio ∼ 0.999994 and the dot product (projection) of the normalized gradients ∼ 0.999995.

- In section III, we appended: "We verify the adjoint solver's accuracy by comparing its output to a gradient obtained via finite-difference."
- In section IVA, we appended: "We verify the accuracy of the adjoint solver by using the finite-difference approximation to reproduce the gradient at low-resolution."

5. At the end of section 2A it is said that adjoints require storing the forward solve in memory. Is this not also the case for SBI and QRM? So this is a potential area of improvement for all the methods presented?

- Yes this is true. We removed these two sentences because they are somewhat redundant. In the second paragraph of section III, we already state: "While computing $u(x,t)$, we store the solution vector at each timestep."

6. In section 3, figure 4, I am a bit confused by the DAL results. It appears that the LB and GD approaches converge rather poorly. This may be the case, but my main issue is that there are places where the cost functional increases rather than decreases. In most gradient descent algorithms (such as those in scipy.optimize.minimize) this can never happen. At each iteration a line search is done to ensure that the cost functional decreases at every iteration. Is that the case for your approach? If so why are there increases in figure 4? In my experience line searches are critical in order to ensure fast convergence.

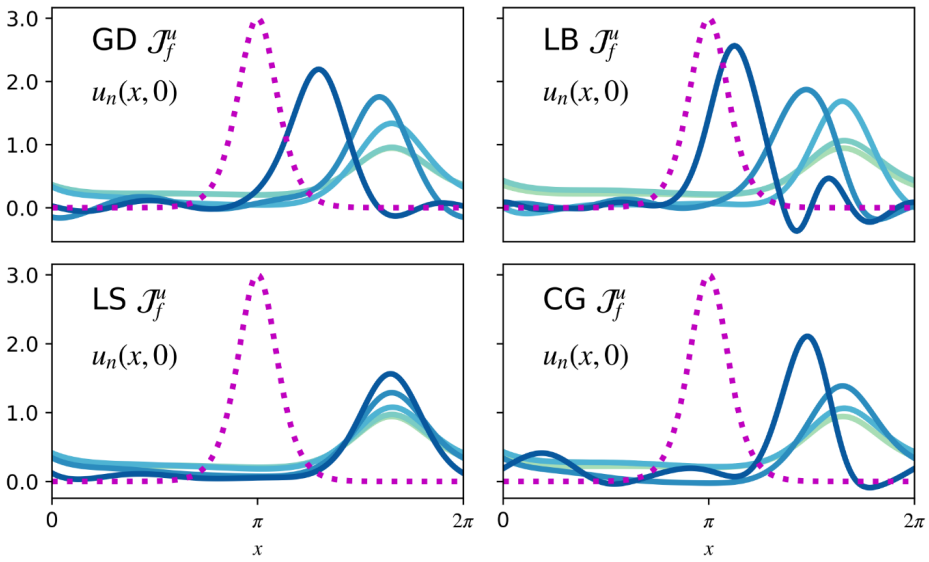

FIG. 1.

- Exact line searches require several evaluations of the objective functional. In our case, each evaluation necessitates its own simulation. We did experiment with using the `scipy.optimize.line_search` method, which has some compatible features: 1. This is an inexact line search. Each evaluation of the objective functional is accompanied by a gradient evaluation, such that we utilize the forward and backward solves of the adjoint loop. 2. `line_search` can be used with any descent direction. We use the (negative) gradient for simplicity.
- L-BFGS performs significantly better than every other `scipy.optimize` routine (using the default parameters), including `scipy.optimize.line_search`
- The intermittent increases or "spikes" in the final error ($\mathcal{J}_f^u$) are due to the scipy's L-BFGS method. This is a quasi-newton method which evaluates the gradient and the objective at several points prior to taking a descent step. In other words, we were showing the errors at intermediate steps in the optimization routine, because we want compare each method based on the number of solves (forward and backward).

- In the updated manuscript, we filtered out these intermittent spikes so that the final error ($\mathcal{J}_f^u$) decreases monotonically.
- In Figure 1 of this report, we plot the trial initial conditions for the KdVB inverse problem for four DAL cases: gradient descent (top left), L-BFGS (top right), line_search (bottom left), and conjugate-gradient (bottom right). This figure is not included in the manuscript, but the top two panels are identical to those in Figure 11. The primary issue we face with this class of inverse problems is that their gradients rapidly change direction in the space of initial conditions. Line searches do not appear to mitigate this problem.

7. In section 4A equation (16) there is a missing term in your adjoint equation of the form $-\mu \cdot (\nabla u)^T$ which is present in your reference [28]. Is this a typo?

- This was a typo. Thank you for your diligence.

8. Again, in figures 8 and 9 can you clarify why the DAL approaches have increases in the cost functional?

- We addressed this concern in our reply to item 6 and updated these figures.

9. Section 5 is very interesting. It appears to me that it can be interpreted from this that QRM and SBI are forms of DAL, but where the adjoint equation is regularized either through changing the sign of the diffusion term or through hyperdiffusion. Is this the case?

- The QRM backward integration equations are indeed regularized adjoint systems. We appended the following text to Section V: "As $\varepsilon \to 0$, QRM backward integration approaches the ill-posed nonlinear adjoint which minimizes $\mathcal{J}_0^u$. From this perspective, QRM is a regularized form of DAL."
- Referring to SBI as a "form of DAL" might invoke controversy from the optimal control community. Instead, we point out the resemblance between SBI and DAL in the following ways:

  1. Section II, Fig. 1, Caption: "Simple Backward Integration (SBI) is a hybrid method, which supplements the linear DAL system with two additional advective terms."

  2. Section IV, "SBI backward integration combines the linear adjoint which minimizes $\mathcal{J}_f^u$ with the additional terms appearing in the ill-posed nonlinear adjoint"

10. The cost functional given by (22) is also interesting as calculating it is ill-posed since it requires integrating the direct equations backwards in time. Does this explain why in the SBI and QRM approaches you get an approximate gradient, but no value of the cost functional which could otherwise be used for gradient descent algorithms?

- We do (briefly) point this out in Equation 3 and the accompanying text in the introduction.
- Section V illustrates the relationship between SBI/QRM and the inaccessible cost functional $\mathcal{J}_0^u$. Earlier, in Section II, we introduce SBI/QRM as methods which approximate the initial deviation $u'(x, 0)$ even though this is equivalent to approximating the gradient of $\mathcal{J}_0^u$. We believe that this particular ordering of topics makes the paper easier to interpret.

11. Section 5 also shows that the missing terms in the DAL approach are actually forcing terms to the adjoint equations stemming from a cost functional with an integral-in-time component. This links to my previous comment that 'missing terms' may not be the correct term. They seem more like extra terms that account for the different cost functional used.

- We have updated every instance of the phrase "missing terms" –¿ "additional terms" or "additional advective terms".

12. In general throughout the paper as all of the approaches are gradient based, could local minimums be found?

- The gradient-based algorithms we tested (L-BFGS, gradient-descent w/ Barzilai–Borwein step sizes, line search, conjugate-gradient) are all guaranteed to converge to a local minimum, eventually.
- We state this explicitly in Section VIA: "This gradient can be used to refine the trial initial condition toward a local extremum. However, the number of iterations required to approximate said extremum depends on the user's initial guess."
- Although local minima can be found in theory, in practice this is difficult because of the problems' ill-conditioning. We tried running the DAL cases for several thousand iterations (not shown in the manuscript). Despite this we were unable to confidently identify local extrema.

13. Again in figure 11, 12 and 13 there are increases in the cost functional with the DAL method.

- We addressed this concern in our reply to item 6 and updated these figures.

14. By no means an essential comment to address, but would you consider making the numerical code used for this paper available so that readers can easily try out these methods for themselves?

- Section III, we appended: "Our code for this paper is available at https://github.com/liamoconnor9/adjop"