DYNAMIC MODE DECOMPOSITION

BRYAN W. OAKLEY[1]

Contents

1. Dynamic Mode Decomposition

1.1. Introduction.

** The code for this project is in `DMDflow.m` **

In this project, we apply the Dynamic Mode Decomposition (DMD) algorithm to a nonlinear flow. We attempt to alter the algorithm by learning future $\widetilde{A}$ and show that the estimated flow field is accurate for a larger time interval. What follows is a presentation of three examples that highlight successes and failures of applying the DMD algorithm to nonlinear flows. When possible, we present methods to improve upon those failures. Throughout, we refer to various modifications of the DMD algorithm, so we begin by detailing said algorithms and notation in Section 1.2.

The first example is a spatially linear flow varying in time $\boldsymbol{u} = a(t)\,\boldsymbol{u}_0$ where $\boldsymbol{u}_0 = (-x, y)^\top$. We find that the DMD algorithm correctly discovers the dynamic mode $\boldsymbol{u}_0$ yet models the time dependence $a(t_{N+n})$ by an exponential fit $a(t_N)\lambda^n$. The exponential fit performs poorly and gives a large error when predicting future flow states. In this example, we can fix this issue by producing a better fit for the time dependence. We propose this method as an improved algorithm and refer to it as 'learnB DMD'.

[1]School of Mathematical and Statistical Sciences, Arizona State University
E-mail address: bwoakley@asu.edu.

The second example is an attempt to apply/improve the DMD algorithm to a fully nonlinear turbulence flow. Unlike in the previous example, this flow is complex enough to discover several dynamic modes along which the flow evolves approximately exponentially. We demonstrate that the learnB DMD algorithm could only produce marginal improvement over the DMD algorithm by showing that even if one predicts the energy in the dynamic modes exactly (referred to as 'exactB DMD'), then we cannot improve over DMD in a significant way. As a consequence, we see that the inability to reconstruct the turbulent flow lies with the dynamic modes themselves. The projection of the flow onto the dynamic modes loses too much information and the dynamic modes cannot reconstruct the flow to the desired accuracy.

In this second example, we also consider learning $\widetilde{A}$, as was the original project prompt. Similarly to the above, we may discuss the algorithms 'learnAtilde DMD' and 'exactAtilde DMD'. We present progress towards learning the entries and eigenvalues of $\widetilde{A}$. However, we argue that the learnAtilde DMD algorithm is not the correct way to proceed. We find that the exactAtilde DMD algorithm is outperformed by the original DMD algorithm. A heuristic explanation of this result is that by allowing $\widetilde{A}$ to change, we also allow the dynamic modes to change. Then, using the old dynamic modes (from the final snapshot $\boldsymbol{u}_N$) with the predicted eigenvalues of future $\widetilde{A}$ produces a poor reconstruction of the flow field. The future eigenvalues are fit to the future dynamic modes, so the reconstruction using the old dynamic modes performs poorly. The DMD algorithm itself fits eigenvalues to the old dynamic modes and uses them to reconstruct the flow field, so using any other eigenvalue is likely to be less accurate. If one wants to use future eigenvalues and future dynamic modes to reconstruct the flow field, then one must feed the estimated flow field back into the DMD algorithm to compute the high dimensional projection and the future dynamic modes. As a consequence, it is not necessary to learn $\widetilde{A}$. We present this algorithm and refer to it as 'feedback DMD', showing that it only provides a marginal improvement over the DMD algorithm.

The third example concerns a traveling scalar wave. We find that exactB DMD provides only a small improvement over the original DMD algorithm. As in the turbulence example, the fault lies with the dynamic modes, that they cannot reconstruct the flow to the desired accuracy. However, compared to the turbulence example, the accuracy of both exactB DMD and DMD algorithms applied to this traveling wave example are quite poor. A heuristic explanation for this reduction in accuracy is that the dynamic modes have little support in the parts of the domain where the traveling wave has yet to visit. As such, a linear combination of those modes (even if they have the correct time dependence as in exactB DMD) cannot reconstruct the traveling wave at future locations. One possible fix for this failure is to allow the dynamic modes themselves to move through space (by precomposing with the flow

field?). One may also consider an affine fit $\boldsymbol{u}_{k+1} \approx A\boldsymbol{u}_k + \boldsymbol{b}$ as opposed to the DMD algorithm's linear fit $\boldsymbol{u}_{k+1} \approx A\boldsymbol{u}_k$.

In conclusion, we argue that the original project prompt (the learnAtilde DMD algorithm) is not the correct way to proceed. Instead, we see better and more interpretable results with the learnB DMD algorithm. The learnB DMD algorithm has the benefit of being able to fix the time dependence issue with the DMD algorithm. As such, it is successful in the first example with the weakly nonlinear flow. However, the nonlinear flows in the second and third example have a more fundamental issue in that the dynamic modes themselves cannot reconstruct the flow field to the desired accuracy. Unfortunately, the learnB DMD algorithm does not address this issue.

1.2. **The Algorithms.**

- **Dynamic Mode Decomposition:** We use the following notation from KutzBrunton2016 DMD book. Say $\boldsymbol{u}_1 = \boldsymbol{u}(t_1), \boldsymbol{u}_2 = \boldsymbol{u}(t_2), \ldots, \boldsymbol{u}_N = \boldsymbol{u}(t_N)$ are snapshots of a velocity field (or a scalar field) at a sequence of times. Use the first $N-1$ snapshots as the columns of X_1 and the last $N-1$ snapshots as the columns of X_2.
  (1) Find the SVD of $X_1 = W\,\Sigma\,V^*$.
  (2) Say W_r, Σ_r, V_r are the truncations of W, Σ, V to the top r singular values. We have the r-truncated Koopman matrix $\widetilde{A} = W_r^* X_2 V_r \Sigma_r^{-1}$.
  (3) Compute the eigendecomposition $\widetilde{A} = Z\Lambda Z^{-1}$.
  (4) The r-truncated eigenvectors (dynamic modes) of A are then $\Phi = X_2 V_r \Sigma_r^{-1} Z$. Also note that the r-truncated eigenvalues of A are in Λ.
  (5) Find the coefficients $\boldsymbol{b}$ for the N^{th} snapshot: $\boldsymbol{b} = \Phi^\dagger \boldsymbol{u}_N$.
  (6) The future snapshots are then estimated as $\boldsymbol{u}_i^{\text{pred}} = \Phi\Lambda^i\boldsymbol{b}$.

- **learnB DMD:** Steps (1)-(4) are the same as the DMD algorithm. Proceed as follows.
  (5) For $k = 1, \ldots, N$, compute the coefficients $\boldsymbol{b}_k$ for the k^{th} snapshot: $\boldsymbol{b}_k = \Phi^\dagger \boldsymbol{u}_k$.
  (6) Use a learning algorithm to estimate the future entries of the sequence $\boldsymbol{b}_1, \ldots \boldsymbol{b}_N$, and call those predictions $\boldsymbol{b}_i^{\text{pred}}$.
  (7) The future snapshots are then estimated as $\boldsymbol{u}_i^{\text{pred}} = \Phi \boldsymbol{b}_i^{\text{pred}}$.

- **exactB DMD:** Assume we have access not only to snapshots $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_N$ but also future snapshots $\boldsymbol{u}_{N+1}, \ldots, \boldsymbol{u}_{N+P}$.
  ** Remark: This algorithm is not a proposed improvement to DMD since it cheats and has access to the unknown future snapshots. It is intended to

provide a lower bound on the error of learnB DMD and quantifies the error of the projection of the snapshots onto the dynamic modes. **
Steps (1)-(4) are the same as the DMD algorithm. Proceed as follows.

(5) For $k = 1, \ldots, N + P$, compute the coefficients $\boldsymbol{b}_k$ for the k^{th} snapshot: $\boldsymbol{b}_k = \Phi^\dagger \boldsymbol{u}_k$.
(6) The future snapshots are then estimated as $\boldsymbol{u}_i \approx \Phi \boldsymbol{b}_i$. This is an estimate in the sense that $\Phi \boldsymbol{b}_i$ is a projection of $\boldsymbol{u}_i$ onto the dynamic modes.

- **learnAtilde DMD:** Consider a sliding window on the snapshots, applying the DMD algorithm to each window. Say the window length is N and we consider M total number of windows. We will need access to snapshots $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_{N+M-1}$. The i^{th} window consists of snapshots $\boldsymbol{u}_i, \ldots, \boldsymbol{u}_{i+N-1}$.
  (1) Compute $\widetilde{A}_i$, which is the $\widetilde{A}$ associated to applying the DMD algorithm to the i^{th} window.
  (2) Use a learning algorithm to estimate the future entries of the sequence $\widetilde{A}_1, \ldots, \widetilde{A}_M$, and call those predictions $\widetilde{A}_i^{\text{pred}}$.
  (3) Compute the diagonal matrix Λ_i^{pred} of eigenvalues of $\widetilde{A}_i^{\text{pred}}$.
  (4) Apply the DMD algorithm to the final window (the M^{th} window) and acquire the dynamic modes Φ, the diagonal matrix Λ of eigenvalues of $\widetilde{A}$, and the coefficients $\boldsymbol{b}$ for the final snapshot $\boldsymbol{u}_{N+M-1}$.
  (5) The future snapshots are then estimated as $\boldsymbol{u}_i^{\text{pred}} = \Phi\, \Lambda_{i-1}^{\text{pred}} \cdots \Lambda_1^{\text{pred}} \Lambda\, \boldsymbol{b}$.

- **exactAtilde DMD:** As in learnAtilde DMD, consider a sliding window on the snapshots. This time, assume we have access to future snapshots.
  ** Remark: This algorithm is not a proposed improvement to DMD since it cheats and has access to the unknown future snapshots. It is intended to provide a lower bound on the error of learnAtilde DMD. **
  Instead of learning $\widetilde{A}_i^{\text{pred}}$ in step (2) of learnAtilde DMD, we may compute $\widetilde{A}_{M+i}$ explicitly. Replace $\widetilde{A}_i^{\text{pred}}$ with $\widetilde{A}_{M+i}$ and proceed with the steps of the learnAtilde algorithm to produce estimates $\boldsymbol{u}_i^{\text{pred}}$.

- **feedback DMD:** In this algorithm, we use the DMD algorithm to produce an estimate of the snapshot one time step in the future, then feed this guess back into the DMD algorithm.
  (1) Apply the DMD algorithm to the window of snapshots $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_N$ to produce the estimate of the next snapshot $\boldsymbol{u}_{N+1} \approx \boldsymbol{u}_1^{\text{pred}}$.
  (2) Apply the DMD algorithm to the window of snapshots $\boldsymbol{u}_2, \ldots, \boldsymbol{u}_N, \boldsymbol{u}_1^{\text{pred}}$ to produce the estimate of the next snapshot $\boldsymbol{u}_{N+2} \approx \boldsymbol{u}_2^{\text{pred}}$.
  $\vdots$

(P) Apply the DMD algorithm to the window of snapshots

$$\boldsymbol{u}_P, \ldots, \boldsymbol{u}_N, \boldsymbol{u}_1^{\text{pred}}, \ldots, \boldsymbol{u}_{P-1}^{\text{pred}}$$

to produce the estimate of the snapshot $\boldsymbol{u}_{N+P} \approx \boldsymbol{u}_P^{\text{pred}}$.

1.3. **Spatially linear flow varying in time.** Consider the spatially linear flow varying in time $\boldsymbol{u} = a(t)\,\boldsymbol{u}_0$ where $\boldsymbol{u}_0 = (-x, y)^\top$. For simplicity, we take $a(t) = 1 - \epsilon\, t$. For small ϵ, this example is weakly nonlinear and slowly varying in time. We use the value $\epsilon = 1/10$ in simulations. Letting $a_n = a(t_n)$, we see that every column of X_1 is a multiple of the vector $\boldsymbol{u}_0$. That is

$$X_1 = (a_1\boldsymbol{u}_0, a_2\boldsymbol{u}_0, \ldots, a_{N-1}\boldsymbol{u}_0)\,.$$

Then X_1 can be written in outer product form $X_1 = \boldsymbol{u}_0\,\boldsymbol{a}_1^\top$ where $\boldsymbol{a}_i = (a_i, a_2, \ldots a_{N+i-2})^\top$. Interpreting the singular value decomposition of X_1 as $W\Sigma V^* = \sum_{i=1}^{r} \sigma_i \boldsymbol{w}_i \boldsymbol{v}_i^\top$, we conclude that $\sigma_1 = \|\boldsymbol{u}_0\|\,\|\boldsymbol{a}_1\|$, $\boldsymbol{w}_1 = \boldsymbol{u}_0/\|\boldsymbol{u}_0\|$, $\boldsymbol{v}_1 = \boldsymbol{a}_1/\|\boldsymbol{a}_1\|$, and all the other singular values are zero. Since the other singular values are zero, we take $r = 1$. We see that the DMD algorithm only discovers one spatial mode $\boldsymbol{u}_0$. The r-truncated singular value decomposition of X_1 is $W_r\Sigma_r V_r^*$ where

$$\begin{aligned} W_r &= \frac{\boldsymbol{u}_0}{\|\boldsymbol{u}_0\|}\,, \\ \Sigma_r &= \|\boldsymbol{u}_0\|\,\|\boldsymbol{a}_1\|\,, \quad \text{and} \\ V_r &= \frac{\boldsymbol{a}_1}{\|\boldsymbol{a}_1\|}\,. \end{aligned}$$

Using that $X_2 = \boldsymbol{u}_0\,\boldsymbol{a}_2^\top$, we have

$$\begin{aligned} \widetilde{A} &= W_r^* X_2 V_r \Sigma_r^{-1} \\ &= \frac{\boldsymbol{u}_0^\top}{\|\boldsymbol{u}_0\|}\, X_2\, \frac{\boldsymbol{a}_1}{\|\boldsymbol{a}_1\|}\, \frac{1}{\|\boldsymbol{u}_0\|\,\|\boldsymbol{a}_1\|} \\ &= \frac{\boldsymbol{u}_0^\top}{\|\boldsymbol{u}_0\|}\, \boldsymbol{u}_0\,\boldsymbol{a}_2^\top\, \frac{\boldsymbol{a}_1}{\|\boldsymbol{a}_1\|}\, \frac{1}{\|\boldsymbol{u}_0\|\,\|\boldsymbol{a}_1\|} \\ &= \frac{\boldsymbol{a}_2 \cdot \boldsymbol{a}_1}{\|\boldsymbol{a}_1\|^2}\,. \end{aligned}$$

Remark: Consider a sliding window on the snapshots, applying the DMD algorithm to each window. Say $\widetilde{A}_i$ is the $\widetilde{A}$ on the i^{th} window. We see that the evolution of the scalar $\widetilde{A}_i = \boldsymbol{a}_{i+1} \cdot \boldsymbol{a}_i \,/\, \|\boldsymbol{a}_i\|^2$ is entirely determined by time dependence part $a(t)$ of the flow. We also notice that the spatial dependence $\boldsymbol{u}_0$ of the flow is contained in W while the time dependence of the flow is contained in V. That is, even

if we don't know the function $a(t)$ from the original flow, the components of V are $a_n = a(t_n)$ and we can use these to predict future $a(t_M)$.

Since $\widetilde{A}$ is a scalar, its eigendecomposition is $\widetilde{A} = Z\Lambda Z^{-1} = 1 \cdot \widetilde{A} \cdot 1$. In particular, the eigenvalue is $\lambda = \widetilde{A}$. The dynamic mode is

$$\begin{aligned}\Phi &= X_2 V_r \Sigma_r^{-1} Z \\ &= \boldsymbol{u}_0 \, \boldsymbol{a}_2^\top \, \frac{\boldsymbol{a}_1}{\|\boldsymbol{a}_1\|} \, \frac{1}{\|\boldsymbol{u}_0\| \, \|\boldsymbol{a}_1\|} \cdot 1 \\ &= \frac{\widetilde{A}}{\|\boldsymbol{u}_0\|} \, \boldsymbol{u}_0 \, .\end{aligned}$$

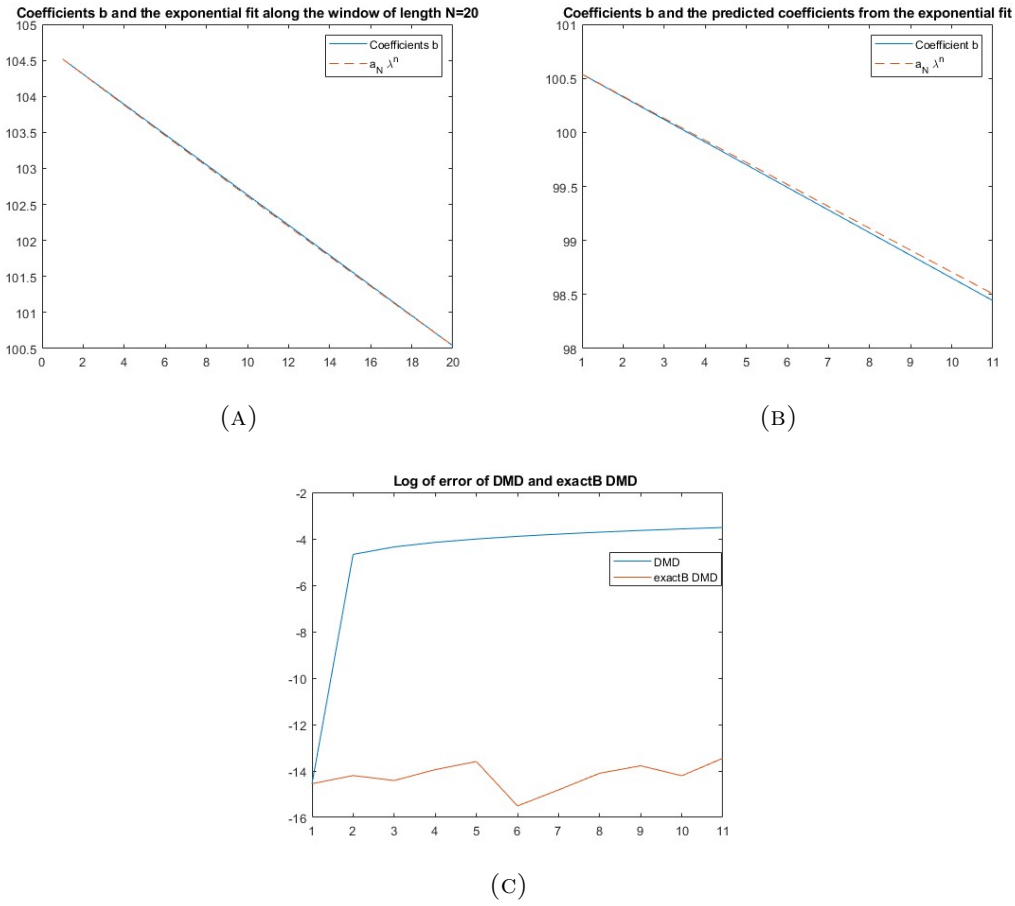
For the final snapshot $\boldsymbol{u}_N$, let the coefficient b_N be the amount of energy contained in the dynamic mode Φ, as determined by the equation $\Phi \, b_N = \boldsymbol{u}_N$. We see that that $b_N = \widetilde{A}^{-1} \, \|\boldsymbol{u}_0\| \, a_N$. The estimates of the future snapshots are then

$$\begin{aligned}\boldsymbol{u}_{N+n} &\approx \Phi \Lambda^n b_N \\ &= \frac{\widetilde{A}}{\|\boldsymbol{u}_0\|} \, \boldsymbol{u}_0 \widetilde{A}^n \, \widetilde{A}^{-1} \, \|\boldsymbol{u}_0\| \, a_N \\ &= a_N \, \widetilde{A}^n \, \boldsymbol{u}_0 \\ &= \widetilde{A}^n \, \boldsymbol{u}_N\end{aligned}$$

Success: The DMD algorithm correctly picks out the spatial mode. $\boldsymbol{u}_0$ is indeed the correct spatial dependence, and the flow evolves as a time varying multiple of that spatial dependence.

Failure: The DMD algorithm does a poor job at predicting the time dependence. In general, for the snapshot $\boldsymbol{u}_k$, let the coefficients $\boldsymbol{b}_k$ be the amount of energy contained in the dynamic modes Φ, as determined by the equation $\Phi \, \boldsymbol{b}_k = \boldsymbol{u}_k$. The DMD algorithm assumes the flow to be linear and so the flow would evolve exponentially along the dynamic modes. In this example, the DMD algorithm computes an exponential fit $b_{N+n} \approx a_N \, \lambda^n$. However, we know that b_k evolves linearly in time as $b_k = \widetilde{A}^{-1} \, \|\boldsymbol{u}_0\| \, (1 - \epsilon \, t_k)$.

As expected, this fit performs well over the window (see Fig. 1a) but poorly outside the window (see Fig. 1b). Figure 1c shows that the reconstruction from DMD using the exponential fit performs poorly (labeled as 'DMD' with error $\approx 10^{-4}$) while using the correct coefficients b_k performs well (labeled as 'exactB DMD' with error $\approx 10^{-14}$).

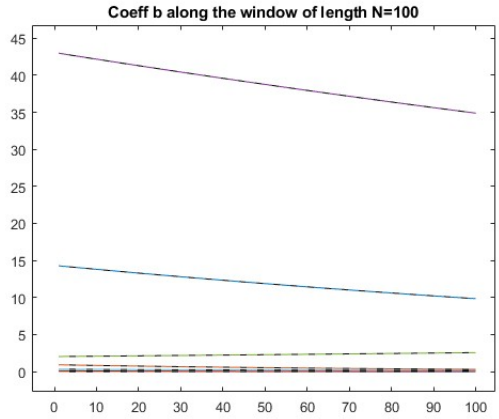


(A) (B)

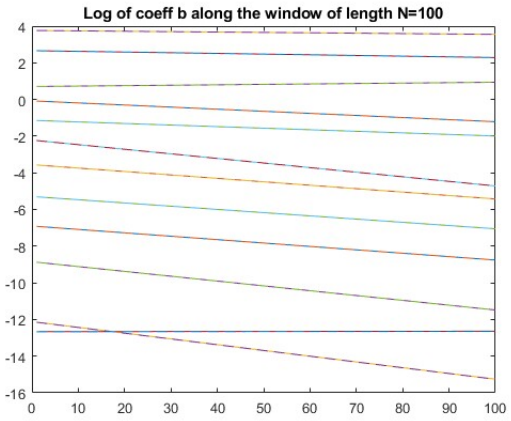
(C)

FIGURE 1. Subfigure (A) demonstrates that the exponential fit is tight over the window of 20 time steps. Subfigures (B) and (C) plot predictions 10 time steps forward. Subfigure (B) demonstrates the exponential fit begins to drift. Subfigure (C) compares the error in predictions from DMD and exactB DMD.

1.4. **Turbulence Data.** We now analyze the turbulence data (in Cases $--$ > Turb). In the figures below we take $N = 100, r = 24, \text{shift} = 50$. We plot below the coefficients $\boldsymbol{b}$ (in solid lines) with respect to the dynamic modes for each snapshot in the window of length 100. We also plot (in dashed lines) the exponential fit from DMD.

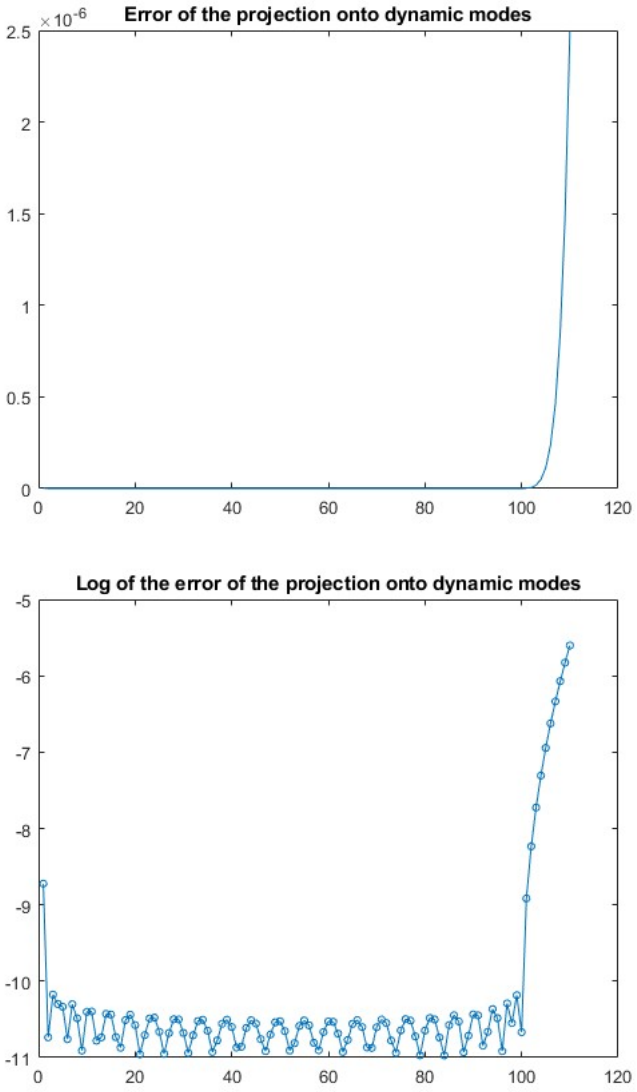


We also plot the logarithms of both quantities:

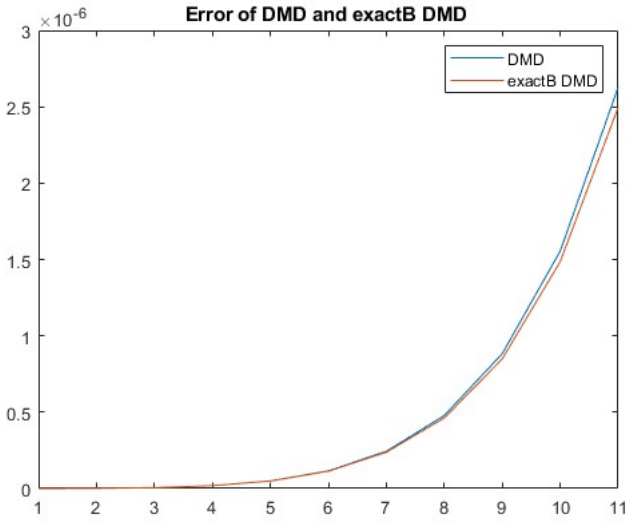


The evolution of the coefficients is captured well by the exponential fit.

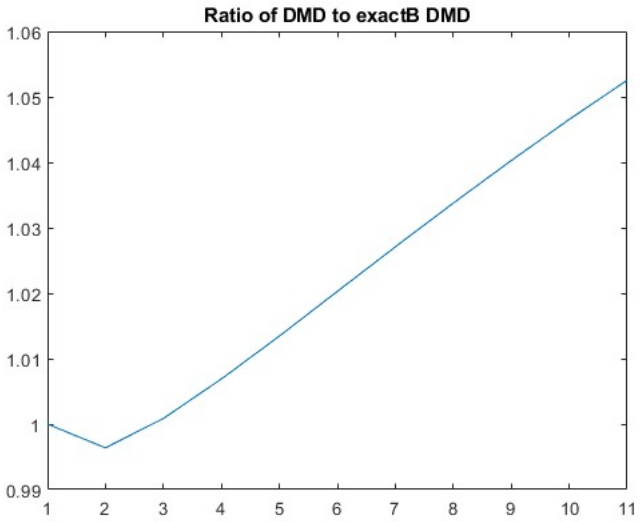
Unfortunately, it looks like our dynamic modes do a poor job of rebuilding future states. The following figure shows the error of exactB DMD compared to DNS. The figure demonstrates that the dynamic modes for this window do a great job (error $< 10^{-10}$) of rebuilding the flow states over the course of the window (from time steps 1 to 100). However, it does a poor job (error exponentially increasing) of rebuilding future flow states (from time steps 101 to 110)



Lastly, we see that exactB DMD is marginally better than DMD. We plot the error:



We quantify the difference by looking at the ratio of the error:



We see that exactB DMD is approximately a 5% improvement over DMD after predicting about 10 time steps.

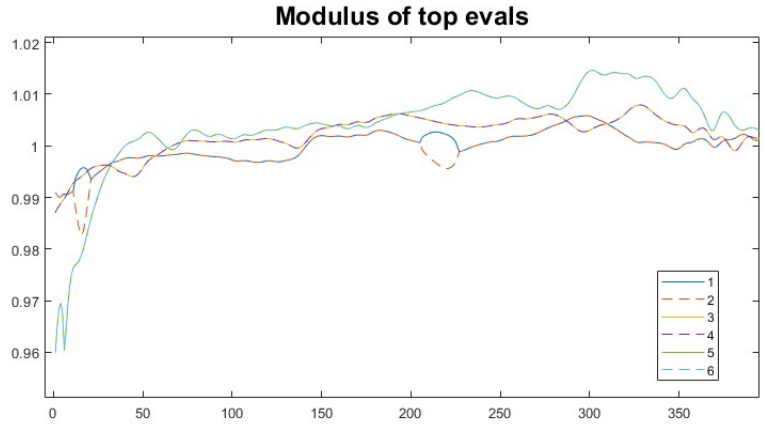
Unlike in the analytic example (spatially linear flow varying in time), this turbulence flow is complex enough to discover several dynamic modes along which the flow evolves approximately exponentially. We have demonstrated that the learnB DMD algorithm could only produce marginal improvement over the DMD algorithm by showing that even if one predicts the energy in the dynamic modes exactly (that

is, exactB DMD), then we cannot improve over DMD in a significant way (approximately a 5% improvement over 10 time steps). As a consequence, we see that the inability to reconstruct the turbulent flow lies with the dynamic modes themselves. The projection of the flow onto the dynamic modes loses too much information and the dynamic modes cannot reconstruct the flow to the desired accuracy.

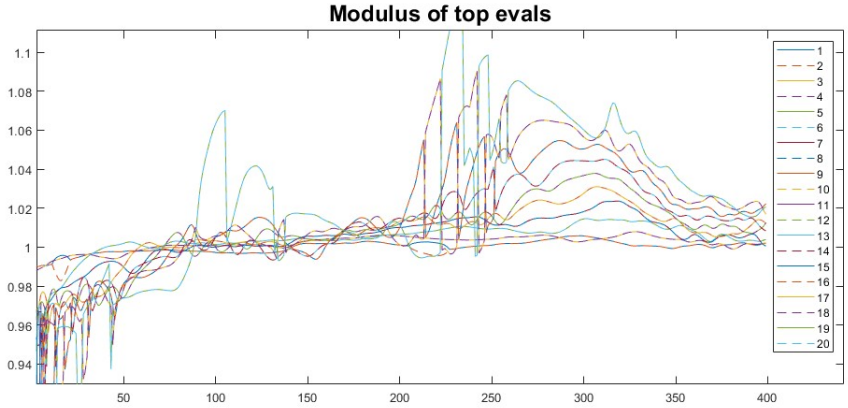
1.5. **Turbulence Data: considering the evolution of A tilde.** Running DMD with $N = 5$ and $r = 4$, we arrive at the following for $\widetilde{A}$:

```
Atilde =

   0.997026469804000  -0.006068058117810   0.002736220112216   0.000520158948316
   0.013726042108171   0.953139093917188  -0.031955060499594   0.022476727135306
  -0.000613409293573  -0.100000302047764   0.743475888854652   0.121963680727011
  -0.000156124895896  -0.011895903396211   0.226355965549448   0.589936362676404


Atilde =

   0.997123222986986   0.006837201455754   0.001864387449386   0.000293764106403
  -0.013262644485626   0.964773996767002   0.007232828210810  -0.014101902460045
  -0.000519624044833   0.084243709104181   0.822489235515536   0.052513299981363
  -0.000104413559051   0.008359467916365   0.192199694330889   0.674230317399426
```

Notice that entries $(2,1),(1,2),(3,2),(4,2),(2,3),(2,4)$ have changed sign, however otherwise the entries are only perturbed from one window to the next. We also notice that eigenvalues of $\widetilde{A}$ vary slowly, which are the quantities we wish to learn. Plotting the modulus of six leading eigenvalues over 400 windows, we have:

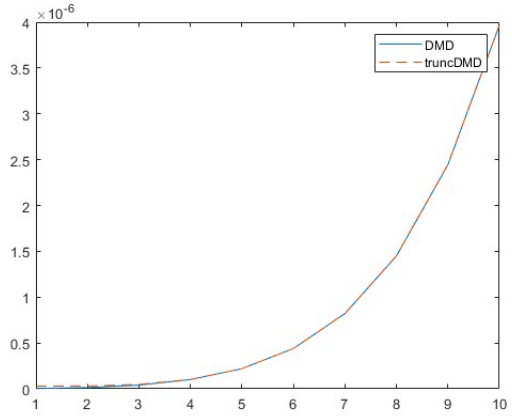


This behavior looks like it could be modeled well as a perturbation around $\lambda = 1$. We also plot the top 24 eigenvalues, showing that the evolution can be more erratic for the less dominant modes:

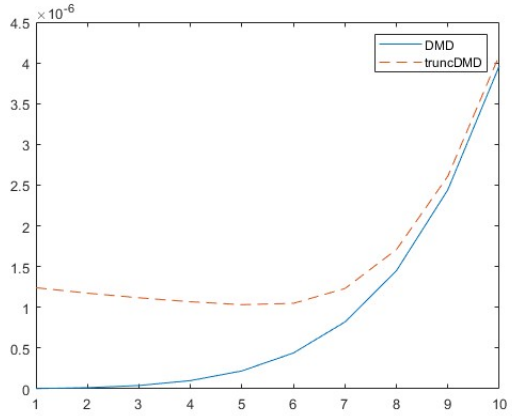


We see that predicting the evolution of the 24 eigenvalues may be more difficult than we thought. The evolution is quite intricate (see evalDanceScaled.avi). In fact, we would need to predict the evolution of most of the eigenvalues, not just the dominant ones, as the following demonstrates.

Below, we truncate $\widetilde{A}$ to the top 14 modes (referred to as 'truncDMD') and look at the error against the direct numerical simulation (DNS). We also plot the error of DMD against DNS.

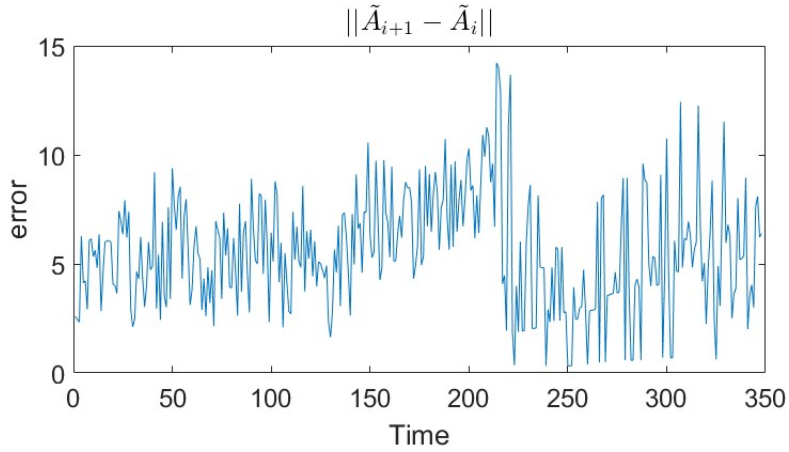


We see that truncDMD performs similarly to DMD. However, the situation is worse if we truncate to 12 modes:

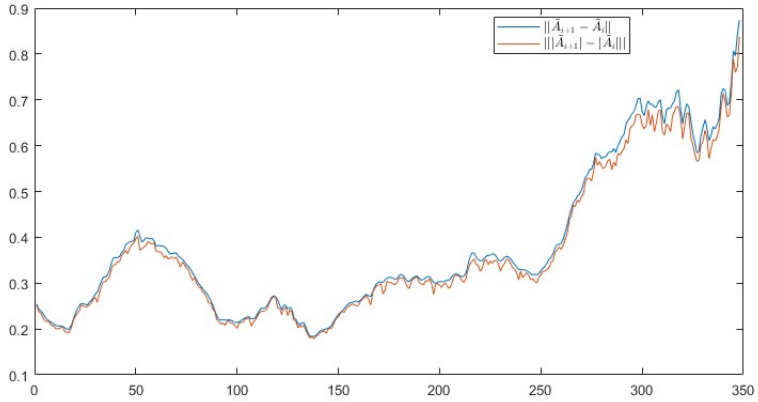


These error plots demonstrate that one must predict the evolution of many of the eigenvalues, not just the top eigenvalues, to have small error. Instead, come back and think about the evolution of the entries of $\widetilde{A}$. We noticed that some entries were switching sign. It turns out that this comes from V in $X_1 = W\Sigma V^*$. A singular direction of V may be multiplied by -1 and the corresponding singular direction of W may be multiplied by -1 when taking this singular value decomposition. This just has to do with how Matlab computes the eigenvectors. By forcing the vectors of V to be sampled from the right half of plane, this would eliminate the switching. The following demonstrates that doing so forces $\widetilde{A}$ to evolve more continuously.

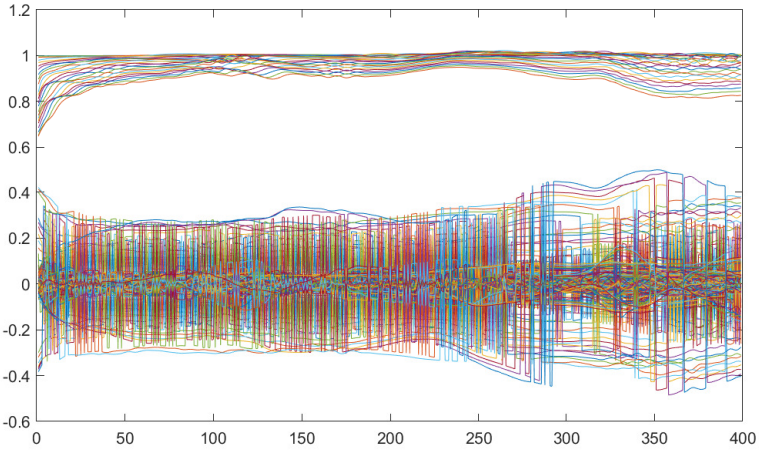
Without adjusting the V's, for our turbulence data, we have the following evolution of the Frobenius norm of the difference between subsequent $\widetilde{A}$.



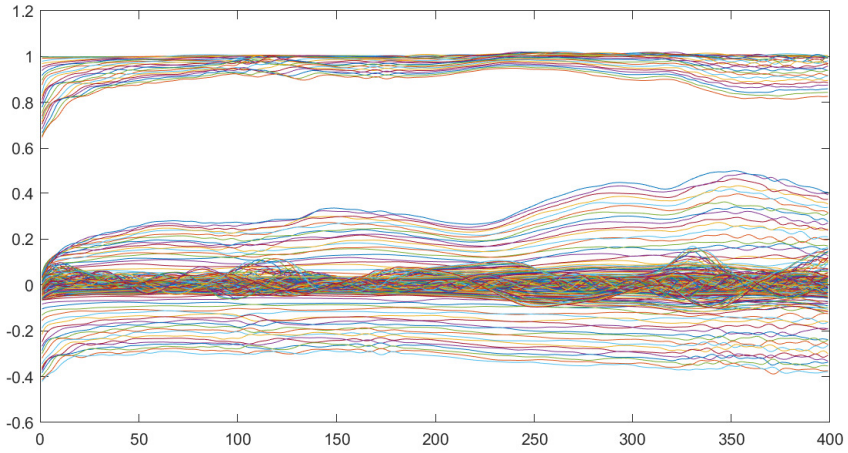
Now forcing the vectors in V to sample only from the right half of the plane (first entry > 0) forces the SVD to be more unique. The following figure shows that this adjustment accounts for almost all the difference in the sign change:



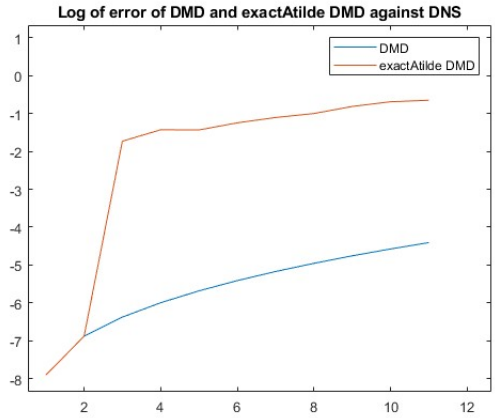
Finally, we can compare the evolution of the entries of $\widetilde{A}$ as a function of time. I plot below the entries of $\widetilde{A}$ as a function of time (for r=24, N=100, noWindows=399), where we do not adjust the vectors in V.



Now adjusting the vectors in V, we see the entries in $\widetilde{A}$ have a more smooth evolution.



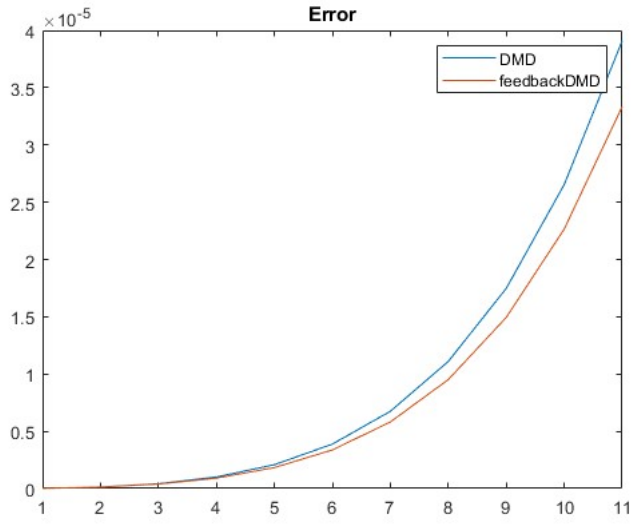
We see the entries of $\widetilde{A}$ have a more smooth evolution. It would now be possible to perform a learning algorithm and predict the next few time steps of $\widetilde{A}$. The question arises, how much better can we improve over the DMD algorithm if we can predict the evolution of the entries of $\widetilde{A}$? This will help us understand if our learning algorithm is accurate enough to reproduce $\widetilde{A}$, perhaps a first order correction is all that is needed? Unfortunately, it turns out that predicting $\widetilde{A}$ does not help us create an algorithm that performs better than DMD. We say that 'exactAtilde DMD' is the algorithm that uses the new eigenvalues from future $\widetilde{A}$ and the old dynamic modes from the window to reconstruct the fluid field. We see that this algorithm produces much more error than DMD even after a few time steps.



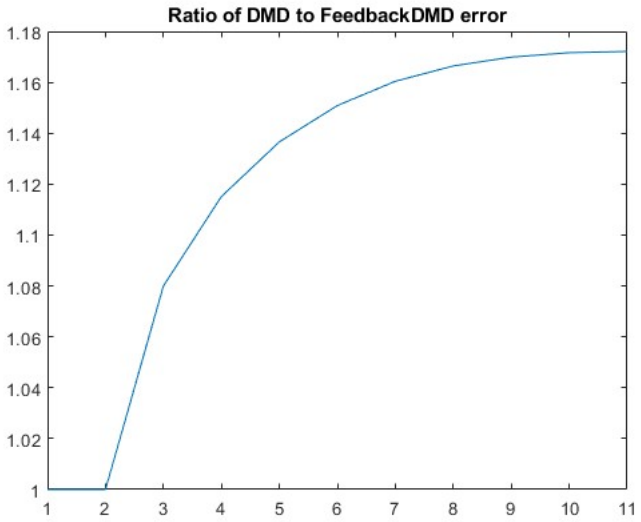
We have presented progress towards learning the entries and eigenvalues of $\widetilde{A}$. However, we argue that the learnAtilde DMD algorithm is not the correct way to

proceed. We find that the exactAtilde DMD algorithm is outperformed by the original DMD algorithm. A heuristic explanation of this result is that by allowing $\widetilde{A}$ to change, we also allow the dynamic modes to change. Then, using the old dynamic modes (from the final snapshot $\boldsymbol{u}_N$) with the predicted eigenvalues of future $\widetilde{A}$ produces a poor reconstruction of the flow field. The future eigenvalues are fit to the future dynamic modes, so the reconstruction using the old dynamic modes performs poorly. The DMD algorithm itself fits eigenvalues to the old dynamic modes and uses them to reconstruct the flow field, so using any other eigenvalue is likely to be less accurate. If one wants to use future eigenvalues and future dynamic modes to reconstruct the flow field, then one must feed the estimated flow field back into the DMD algorithm to compute the high dimensional projection and the future dynamic modes. As a consequence, it is not necessary to learn $\widetilde{A}$. We present this algorithm and refer to it as 'feedback DMD', showing that it only provides a marginal improvement over the DMD algorithm.

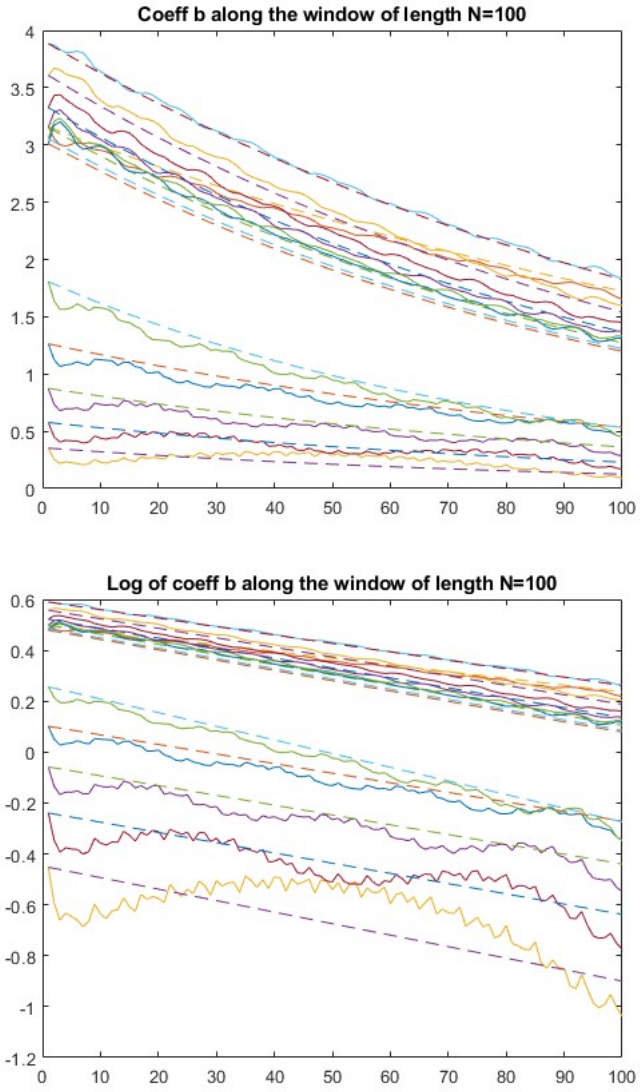
Indeed, suppose you wanted to learn future $\widetilde{A}$ and use the updated eigenvalues and dynamic modes associated to these $\widetilde{A}$. Step one is the same as usual: compute DMD on the snapshots $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_N$. Compute the dynamic modes from the equation $\Phi = X_2 V_r \Sigma_r^{-1} Z$. Use this to predict the flow field one step forward, call it $\boldsymbol{u}_1^{\text{pred}}$. Step two is where we differ: to compute the dynamic modes we must have access to the snapshot $\boldsymbol{u}_{N+1}$. That is, in order to compute the dynamic mode from the equation $\Phi = X_2 V_r \Sigma_r^{-1} Z$, we must have the matrix X_2. We have Σ from $X_1 = (\boldsymbol{u}_2, \ldots, \boldsymbol{u}_N)$ and even if we predict the future $\widetilde{A}$ so we have access to V_r and Z, then we still cannot compute Φ without having the matrix $X_2 = (\boldsymbol{u}_3, \ldots, \boldsymbol{u}_{N+1})$. Therefore, to compute the dynamic modes we must have access to the snapshot $\boldsymbol{u}_{N+1}$. We don't have that data, so the best approximation to $\boldsymbol{u}_{N+1}$ is $\boldsymbol{u}_1^{\text{pred}}$. If we have consigned to use $\boldsymbol{u}_1^{\text{pred}}$ in substitute of $\boldsymbol{u}_{N+1}$, then do we really need to use a learning algorithm to predict future $\widetilde{A}$? No, in fact, it would be more simple to just feed this prediction back into DMD by applying DMD to the snapshots $\boldsymbol{u}_2, \ldots, \boldsymbol{u}_N, \boldsymbol{u}_1^{\text{pred}}$, and as a consequence we will get an estimate for $\widetilde{A}$. In this way, continue to use DMD to predict the flow field one step forward and feeding that prediction back to compute another step of DMD. This is our proposed 'feedback DMD' algorithm. The following figure shows that feedbackDMD slightly outperforms DMD, comparing to the turbulence DNS.



We quantify the improvement by noting that the following figure shows that we see about a 16% improvement in feedbackDMD over original DMD after about 10 time steps.

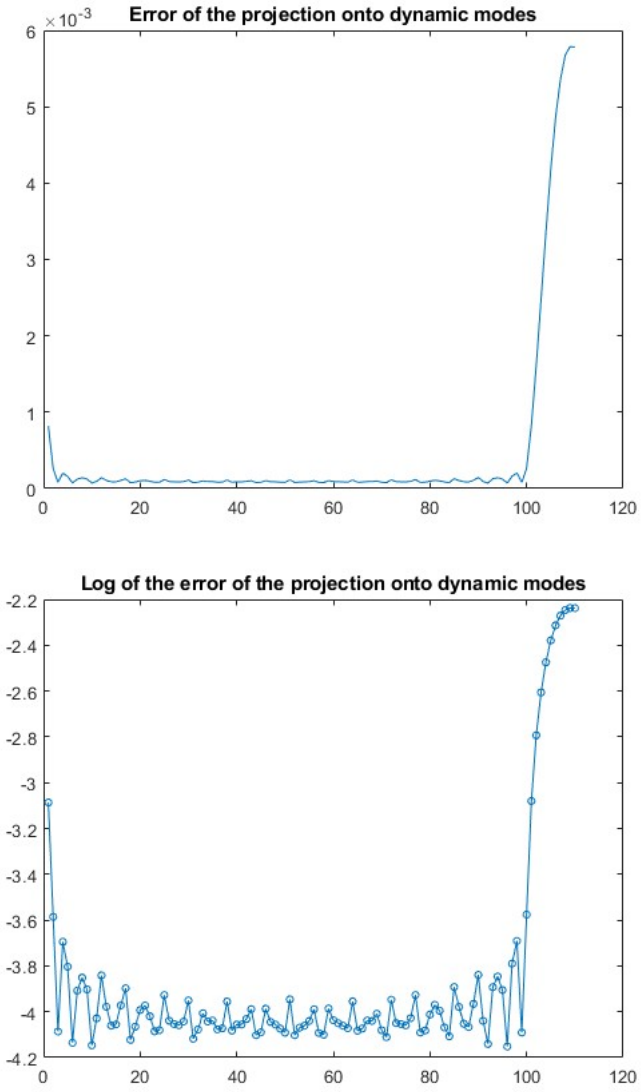


1.6. **Travelling Wave example.** In this section, we consider a scalar field given by $\theta(x,t) = \sin(r)e^{-r}$ where $r = \sqrt{(x-t)^2 + y^2}$. This example was chosen to mimic a traveling sine wave moving down the x-axis. We compute below the coefficients b with respect to the dynamic modes for each snapshot in the window of length 100. Take $N = 100, r = 24$.

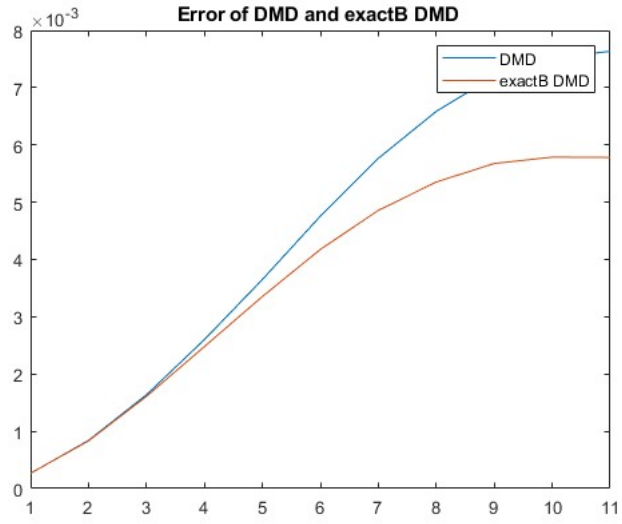


The coefficients seem to be oscillating above/below the exponential fit. The coefficients b are not modelled well by an exponential fit (with error $\approx 10^{-1}$ or 10^{-2}).

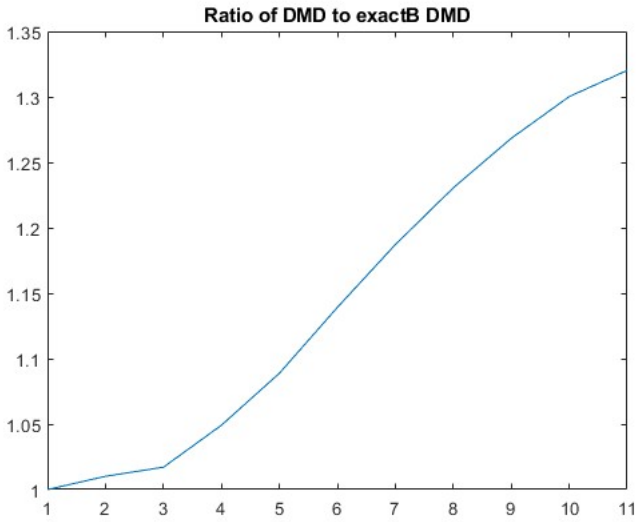
Unfortunately, it looks like our dynamic modes do a poor job of rebuilding snapshots. We proceed by analyzing exactB DMD. In the figure below, we see that the dynamic modes for this window do a decent job (error $\approx 10^{-4}$) of rebuilding the flow states over the course of the window (from time steps 1 to 100). However, it does a poor job of rebuilding future flow states (from time steps 101 to 110)



Lastly, we see that exactB DMD is marginally better than DMD. We plot the errors:



We quantify the difference by looking at the ratio of the error:

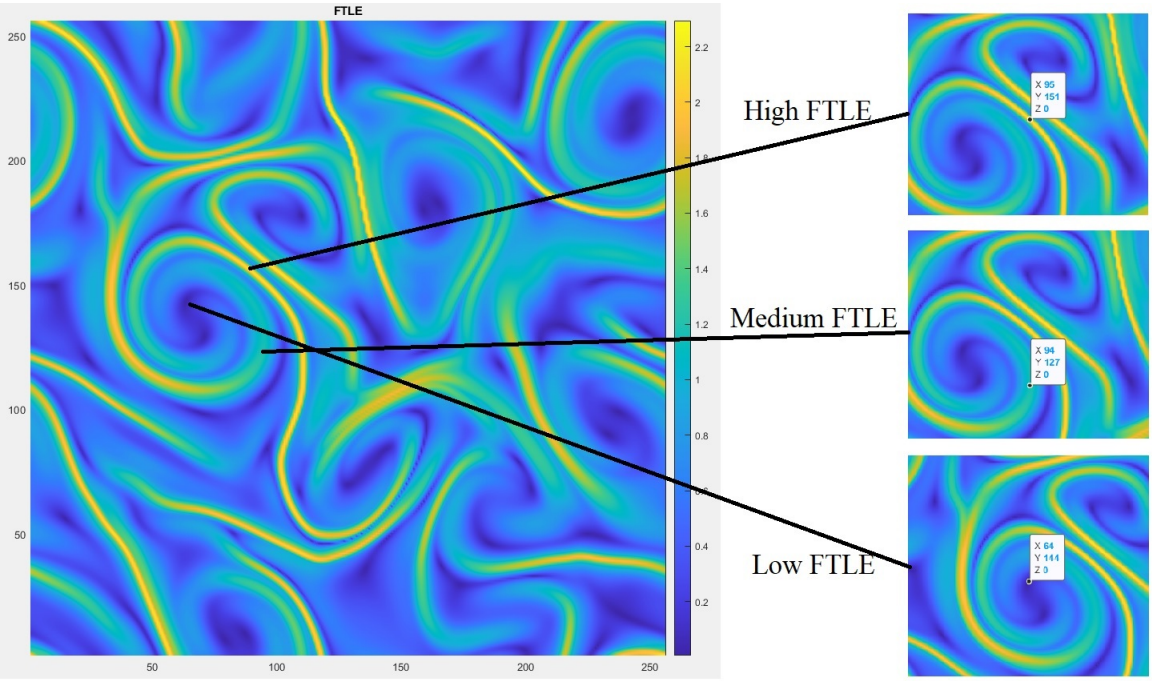


We see that exactB DMD is approximately a 30% improvement over DMD after predicting about 10 time steps.

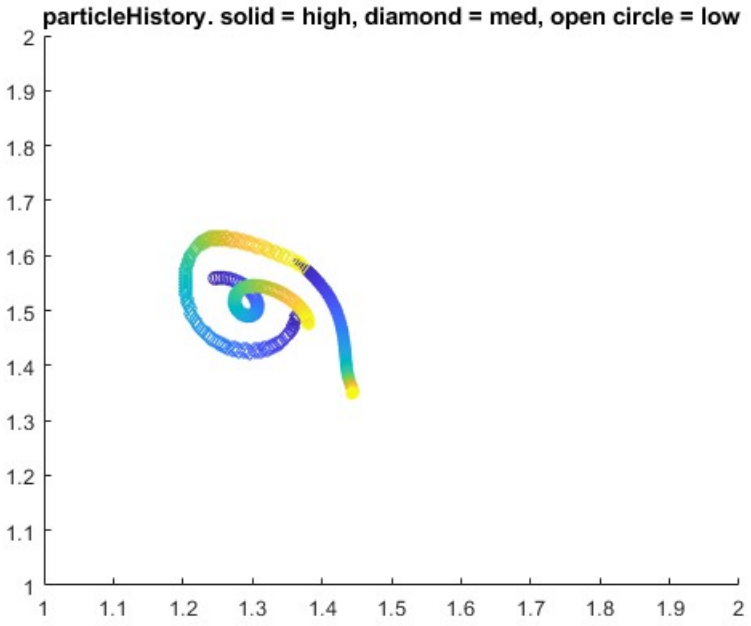
2. Finite Time Lyapunov Exponents

** The code for this project is in `FTLE_particle.m` **

In this project, we consider a turbulent flow and compute the Finite Time Lyapunov Exponents (over 2 time units) at each of the grid points. The domain is separated into zones: stagnant zones of low FTLE grid points, intermediate zones of medium FTLE grid points, and thin zones of high FTLE grid points. We choose a representative of each of these zones: high FTLE of 2.17 at $(95, 151)$, medium FTLE of 1.15 at $(94, 127)$, and low FTLE of 0.02 at $(64, 144)$. We plot our representatives on top of the FTLE surface:

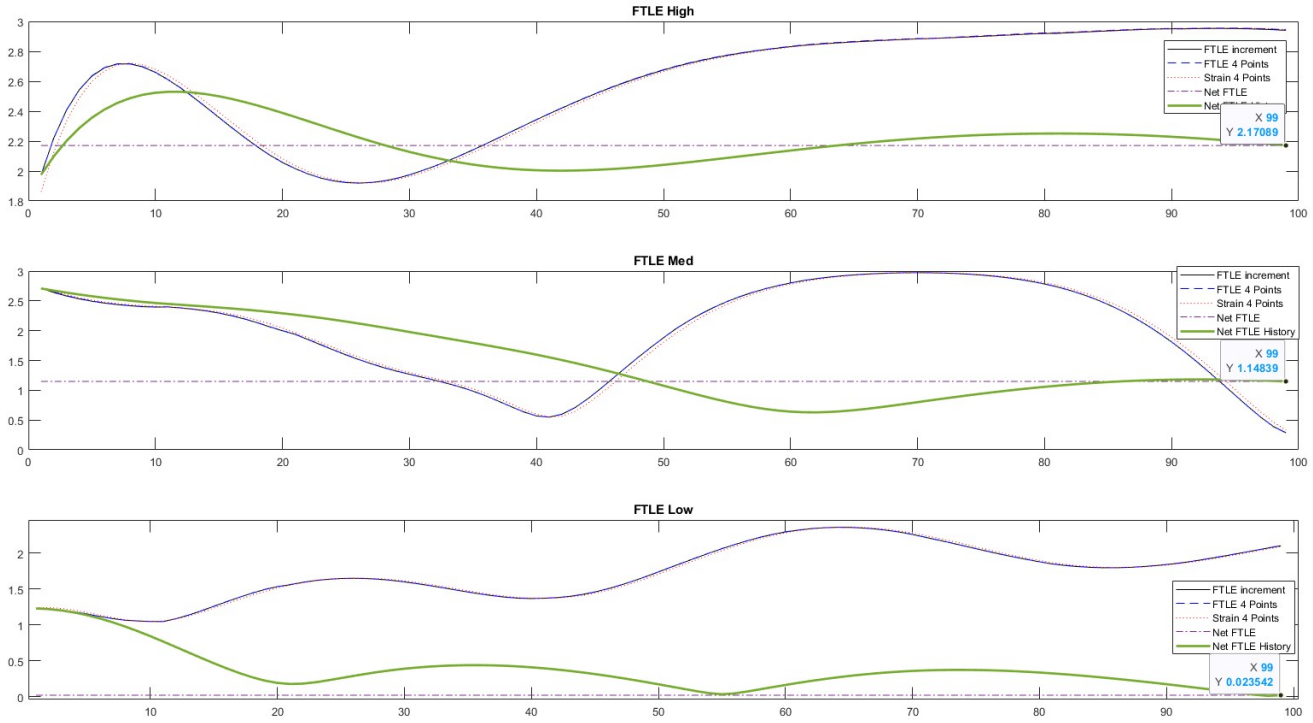


We also plot the particle paths of these three points over the 2 time units (blue is time $t = 0$ and yellow is time $t = 2$):



We wish to reconstruct the typical evolution for each of these representative trajectories by prescribing the instantaneous Lyapunov exponents. We plot 5 curves in the following figure:

- **FTLE increment:** We calculate the FTLE over one time step (a discrete approximation of the instantaneous Lyapunov exponents) at all grid points, then find the value of the FTLE at our particle's current position.
- **FTLE 4 points:** We calculate the FTLE over one time step using 4 nearby grid points. From the figure, we see that FTLE increment = FTLE 4 points.
- **Strain 4 points:** We calculate the strain using 4 nearby grid points. From the figure, we see that FTLE increment (and FTLE 4 points) $\approx$ Strain 4 points.
- **Net FTLE:** We have the FTLE over 2 time units (2.17 for High, 2.15 for Med and 0.02 for Low).
- **Net FTLE History:** Net FTLE History at time step k is the FTLE over the time interval $(0, k\Delta t)$.
  For $k = 1$ (that is $t = \Delta t = .02$), Net FTLE History = FTLE increment.
  For $k = 100$ (that is $t = 2$), Net FTLE History = Net FTLE.



The strain for High is sometimes below and sometimes above the FTLE of 2.17. The strain for Med is sometimes below and sometimes above the FTLE of 1.15. **Surprisingly**, the strain for Low is always much larger than the FTLE of 0.02. We conclude that strain alone is not enough to model the Low representative. How is it possible to have very small FTLE but large strain for the whole time?

Our result is a proposed answer to this question. The following figure plots the angle of the direction of largest stretching. We conjecture that the oscillation in the angle for Low is the reason why the FTLE is so small while the particle experiences high strain. The direction of stretching is oscillating in such a way that a circle of points around our particle is stretched and then unstretched (see the movies `HighStretch.avi`, `MedStretch.avi`, and `LowStretch.avi` for a visualization of the deformation of said circle). The net effect over the two time units is that a circle of points around our particle is essentially untouched.