San Diego State University

Department of Mathematics and Statistics
Math 638
Continuous Dynamical Systems

SAN DIEGO STATE UNIVERSITY

Final Project:

Modeling the Lorentz System and
the Van der Pol Equation:
Dynamic Mode Decomposition
and
Data Assimilation

Lourdes Coria,
Horacio Lopez,
Matteo Polimeno

Professor:
Dr. Christopher Curtis

May 10^{th}, 2018

Contents

1 Abstract

This project focuses on two different techniques of analyzing and interpreting data dynamics:

1. *Dynamic mode decomposition* (DMD), as an example of an equation-free method;
2. *Data-assimilation* method, as an example of a method which makes use of both measurements collected and the governing equations of the system.

Using both techniques, the purpose is to reproduce the dynamics of the Lorentz and Van der Pol equation, and ultimately analyze and compare the accuracy of the results from each method.

2 Dynamic Mode Decomposition

2.1 Introduction to DMD

Model-based algorithms are used to understand the behavior of complex dynamical systems. However, such models might not work properly when the governing equations for the system fail to work or the equations are not known. Therefore, an alternative approach based on experimental data can be used to understand, copy, and control the dynamics of a given system.

The data-based algorithm discussed in the following section is named dynamic mode decomposition (DMD). This method does not make use of underlying governing equation, but rather snapshots of measurements to predict and control a system's dynamical behavior. It gives a decomposition of experimental data into a set of dynamic modes that are collected from the snapshots in a given time interval.

The DMD method is applied on data matrices. Therefore, the data collection process is of utmost importance, as the method is data-driven. Let $\mathbf{X}$ and $\mathbf{X}'$ be the data matrices whose columns are taken from a sequential time-series $\{x_k\}_{k=1}^{m}$. Where $\mathbf{X}$ is the original data matrix and $\mathbf{X}'$ is the time-shifted snapshot matrix of X,

$$\mathbf{X} = \begin{bmatrix} | & | & & | \\ x_1 & x_2 & \dots & x_{m-1} \\ | & | & & | \end{bmatrix} \qquad \mathbf{X}' = \begin{bmatrix} | & | & & | \\ x_2 & x_3 & \dots & x_m \\ | & | & & | \end{bmatrix} \tag{2.1}$$

The definition of DMD is an approximate eigen-decomposition of the operator $\mathbf{A} = \mathbf{X}'\mathbf{X}^{\dagger}$. Here $\mathbf{X}^{\dagger}$ is the Moore-Penrose inverse of $\mathbf{X}$. The operator is the least squares/minimum-norm solution to $\mathbf{AX} = \mathbf{X}'$. This makes $\mathbf{A}$ the best-fit operator that relates $\mathbf{X}$ and $\mathbf{X}'$. The following definition describes how to

derive the DMD of the data matrices $\mathbf{X}$ and $\mathbf{X}'$.

Definition 1. Let $\mathbf{X}$, $\mathbf{X}' \in \mathbb{R}^{n\times m}$, and the operator $\mathbf{A}$ be defined as,

$$\mathbf{A} = \mathbf{X}'\mathbf{X}^{\dagger} \tag{2.2}$$

Here $\mathbf{X}^{\dagger}$ is the Moore-Penrose inverse of $\mathbf{X}$. Let the Singular mode decomposition (SVD) of $\mathbf{X}$ be,

$$\mathbf{X} = [\mathbf{\Phi} \ ... \ ] \begin{bmatrix} \mathbf{\Sigma} & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{W}^* \\ \vdots \end{bmatrix} \tag{2.3}$$

where $\mathbf{\Phi} \in \mathbb{R}^{n\times q}$,$\mathbf{\Sigma} \in \mathbb{R}^{q\times q}$, and $\mathbf{W} \in \mathbb{R}^{m\times q}$ and $\mathbf{W}^*$ is the conjugate transpose of $\mathbf{W}$. Therefore,

$$\mathbf{A} = \mathbf{X}'\mathbf{W}\mathbf{\Sigma}^{-1}\mathbf{\Phi}^* \tag{2.4}$$

with its projection onto the column space $\mathbf{\Phi}^*$ given by,

$$\tilde{\mathbf{A}} = \mathbf{\Phi}^*\mathbf{A}\mathbf{\Phi} \tag{2.5}$$

The eigen-decomposition of $\tilde{\mathbf{A}}$ is given by,

$$\tilde{\mathbf{A}}\tilde{\mathbf{V}} = \tilde{\mathbf{V}}\mathbf{\Lambda} \tag{2.6}$$

The DMD eigenvalues are given by the diagonal matrix $\mathbf{\Lambda}$. The eigenvalues lying on the diagonal are the DMD eigenvalues. The corresponding DMD modes are found in the columns of $\mathbf{\Phi}\tilde{\mathbf{V}}$ and denoted as,

$$\mathbf{\Psi} = \mathbf{\Phi}\tilde{\mathbf{V}} \tag{2.7}$$

The pair $(\mathbf{\Psi}, \mathbf{\Lambda})$ compose the dynamic mode decomposition of the data matrices $\mathbf{X}$ and $\mathbf{X}'$ [7].
Based on the definition above, it is deduced that the pair $(\mathbf{\Psi}, \mathbf{\Lambda})$ are related to the operator $\mathbf{A}$ from $\mathbf{AX} = \mathbf{X}'$. To prove that $\mathbf{\Psi}$ and $\mathbf{\Lambda}$ are related to $\mathbf{A}$, we establish and prove the following Theorem.

Theorem 1. If the columns of $\mathbf{X}'$ are spanned by $\mathbf{X}$ the dynamic modes and eigenvalues defined in Definition 1 are the eigenvalues and eigenvectors of the operator $\mathbf{A}$ defined as $\mathbf{A} = \mathbf{X}'\mathbf{W}\mathbf{\Sigma}^{-1}\mathbf{\Phi}^*$.
Proof: Using the definition above, substituting

$$\tilde{\mathbf{A}} = \mathbf{\Phi}^*\mathbf{A}\mathbf{\Phi}$$

into

$$\tilde{\mathbf{A}}\tilde{\mathbf{V}} = \tilde{\mathbf{V}}\mathbf{\Lambda}$$

results in

$$\mathbf{\Phi}^*\mathbf{A}\mathbf{\Phi}\tilde{\mathbf{V}} = \tilde{\mathbf{V}}\mathbf{\Lambda}$$
$$\mathbf{\Phi}\mathbf{\Phi}^*\mathbf{A}\mathbf{\Phi}\tilde{\mathbf{V}} = \mathbf{\Phi}\tilde{\mathbf{V}}\mathbf{\Lambda}$$

Recall,

$$\mathbf{\Psi} = \mathbf{\Phi}\tilde{\mathbf{V}}$$

thus,

$$\mathbf{\Phi}\mathbf{\Phi}^*\mathbf{A}\mathbf{\Phi}\tilde{\mathbf{V}} = \mathbf{\Phi}\tilde{\mathbf{V}}\mathbf{\Lambda}$$
$$\mathbf{\Phi}\mathbf{\Phi}^*\mathbf{A}\mathbf{\Psi} = \mathbf{\Psi}\mathbf{\Lambda} \tag{2.8}$$

In this case $\mathbf{\Phi}^*\mathbf{\Phi}$ and $\mathbf{W}^*\mathbf{W}$ are equal to the identity matrix but $\mathbf{\Phi}^*\mathbf{\Phi}$ and $\mathbf{W}^*\mathbf{W}$ are not the identity matrix. They are eigenvectors and eigenvalues of the projection of $\mathbf{A}$ onto the columns of $\mathbf{\Phi}$. The columns of $\mathbf{\Phi}$ are orthonormal thus,

$$\mathbf{X}' = \mathbf{\Phi}^*\mathbf{\Phi}\mathbf{X}' + \mathbf{R}'$$

such that $\Phi^*\mathbf{R}' = 0$. Plugging $\mathbf{X}' = \mathbf{\Phi}^*\mathbf{\Phi}\mathbf{X}' + \mathbf{R}'$ into $\mathbf{A} = \mathbf{X}'\mathbf{W}\mathbf{\Sigma}^{-1}\mathbf{\Phi}^*$ yields,

$$\mathbf{A} = \mathbf{X}'\mathbf{W}\mathbf{\Sigma}^{-1}\mathbf{\Phi}^*$$
$$\mathbf{A} = \left(\mathbf{\Phi}^*\mathbf{\Phi}\mathbf{X}' + \mathbf{R}'\right)\mathbf{W}\mathbf{\Sigma}^{-1}\mathbf{\Phi}^*$$
$$\mathbf{A} = \mathbf{\Phi}^*\mathbf{\Phi}\mathbf{X}'\mathbf{W}\mathbf{\Sigma}^{-1}\mathbf{\Phi}^* + \mathbf{R}'\mathbf{W}\mathbf{\Sigma}^{-1}\mathbf{\Phi}^*$$

$$\mathbf{A} = \mathbf{\Phi}^*\mathbf{\Phi}\mathbf{A} + \mathbf{R}'\mathbf{W}\mathbf{\Sigma}^{-1}\mathbf{\Phi}^* \tag{2.9}$$

Plugging this $\mathbf{A}$ into $\mathbf{\Phi}\mathbf{\Phi}^*\mathbf{A}\mathbf{\Psi} = \mathbf{\Psi}\mathbf{\Lambda}$ and recalling that $\mathbf{\Psi} = \mathbf{\Phi}\tilde{\mathbf{V}}$ results in,

$$\left(\mathbf{A} - \mathbf{R}'\mathbf{W}\mathbf{\Sigma}^{-1}\mathbf{\Phi}^*\right)\mathbf{\Psi} = \mathbf{\Psi}\mathbf{\Lambda}$$

$$\mathbf{A}\mathbf{\Psi} - \mathbf{\Psi}\mathbf{\Lambda} = \mathbf{R}'\mathbf{W}\mathbf{\Sigma}\tilde{\mathbf{V}} \tag{2.10}$$

If the columns of $\mathbf{X}'$ are spanned by $\mathbf{X}$ then $\mathbf{R}' = 0$ the dynamic modes and eigenvalues are the eigenvalues and eigenvectors of the operator $\mathbf{A}$.

The core of DMD yields an approximate decomposition of the best-fit linear operator relating the two data matrices $\mathbf{X}$ and $\mathbf{X}'$. Computing the DMD of a system can be done using Definition 1 and Theorem 1. Nevertheless, this is not the ideal way to calculate the DMD. There are more practical ways to calculating the DMD of a system that still make use of Definition 1 and Theorem 1 [7].

2.2 SVD-Based Algorithm to Compute DMD

The algorithm that has become the dominating method to compute the dynamic mode decomposition is the SVD-based algorithm. Let $\mathbf{X}$ and $\mathbf{X}'$ be the data matrices whose columns are taken from a sequential time-series $\{x_k\}_{k=1}^{m}$. Where $\mathbf{X}$ is the original data matrix and $\mathbf{X}'$ is the time-shifted snapshot matrix of X. The rest of the SVD-based algorithm follows the steps from Definition 1 with the only difference being that the $\tilde{\mathbf{A}}$ is calculated directly. The computation of $\tilde{\mathbf{A}}$ is given by plugging equation (2.4) into equation (2.5)

$$\begin{aligned}\tilde{\mathbf{A}} &= \mathbf{\Phi}^* \mathbf{A} \mathbf{\Phi} \\ &= \mathbf{\Phi}^* (\mathbf{X}' \mathbf{W} \mathbf{\Sigma}^{-1} \mathbf{\Phi}^*) \mathbf{\Phi} \\ &= \mathbf{\Phi}^* \mathbf{X}' \mathbf{W} \mathbf{\Sigma}^{-1} \end{aligned} \tag{2.11}$$

With the calculation of $\tilde{\mathbf{A}}$ The rest of the algorithm follows Definition 1 [7]. The reason the SVD-based algorithm should be the way to compute DMD is because there are cases in applications where the state dimension is much larger than the number of taken snapshots. The use of the SVD-based algorithm to compute DMD is key to computational efficiency.

2.3 Limitations

One of the limitations of the DMD method consists on the fact that sampling must be done in equally spaced time intervals. Moreover, the DMD method fails if the data matrix is full-ranked and the data has no suitable low-dimensional structure. Modified methods, such as DMD with control [3], try to address such limitations in order to improve results.

2.4 Implementation: Lorentz

From [2], the Lorentz equations are given by

$$x' = \sigma(y - x) \tag{2.12}$$
$$y' = rx - y - xz \tag{2.13}$$
$$z' = xy - bz. \tag{2.14}$$

In order to implement the DMD algorithm for the chaotic Lorenz equation, we solve the system numerically using a fourth-order Runge-Kutta solver with parameters $\sigma = 10$, $b = 8/3$ and $r = 28$. The initial condition vector is $\mathbf{x}(0) = [5\ 5\ 5]$. The time step, Δt, is 0.01 time units. The data matrix consists

of data taken from the x-solution of the Lorenz system and it is set up as a Hankel matrix [6] which has the form:

$$\mathbf{H} = \begin{bmatrix} x(t_1) & x(t_2) & x(t_3) & ... & x(t_p) \\ x(t_2) & x(t_3) & x(t_4) & ... & x(t_{p+1}) \\ x(t_3) & x(t_4) & x(t_5) & ... & x(t_{p+2}) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x(t_q) & x(t_{q+1}) & x(t_{q+2}) & ... & x(t_m) \end{bmatrix} \tag{2.15}$$

The SVD-based DMD method is then implemented using the above approximation to the linear operator, $\tilde{\mathbf{A}}$, and is used to both recreate the data matrix and predict future time states by extraction of the vector

$$x_m = [x(t_m)x(t_{m+1})...x(t_{m+M})]$$

where M dictates how many time steps to solve for in the prediction.

2.4.1 Lorentz Results: Reconstructing Data

After implementing the DMD algorithm to the Hankel matrix consisting of the x-solution elements, we see that the norm of the error vector is in the magnitude of 10^{-12} between the original data and the replicated data. So, the approximated linear operator does a good job recreating the data. The main reason this data reconstruction is important even though it is uninteresting, since that data is already given, is that before the implementation of the Hankel matrix approach, the DMD algorithm would fail at this stage. Moreover, in future parts of this report, we will attempt to reconstruct the underlying dynamics of noisy data while filtering out as much noise as possible.

2.4.2 Lorentz Results: Prediction Future States

The DMD does not work in predicting the future states of the function for chaotic systems, as can be seen in Figure 2. The data matrix gave values for the Lorentz system up to $t = 25$, and we can see that the prediction fails immediately after the data stops.

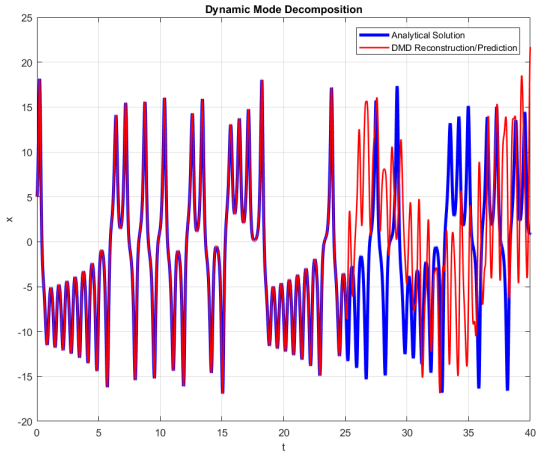

Figure 1: Future State Prediction

A consensus between papers was not found for why DMD fails at predicting future states for chaotic systems. However, in a recently-published paper, it is found that in order for DMD to work in chaotic systems, HAVOK analysis is used to identify the intermittently forced linear system representation of chaos [6]. A description of HAVOK analysis is not included in this report .

2.4.3 Lorentz Results: Noise Reduction

The DMD algorithm is now used to measure how well it recreates hidden dynamics from noisy data. Noise is added to each data step with a Guassian distribution with $\sigma = 1$. We select the dynamics we want to keep through eigenvalue selection and truncation at the time of taking the SVD of the data matrix [9].

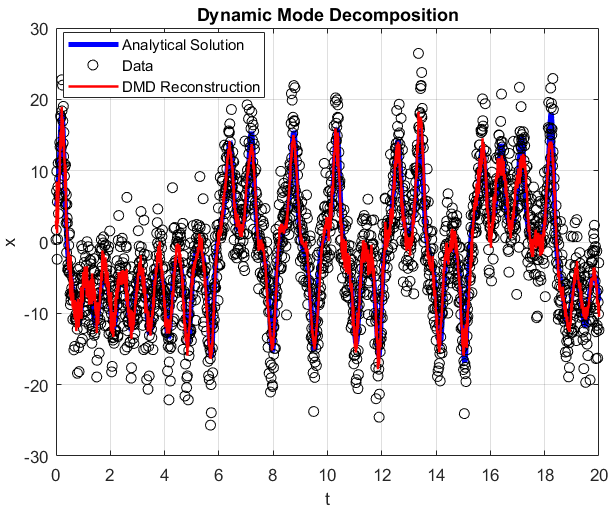

Figure 2: Noise Elimination

As can be seen from figure 2, implementing DMD with truncated-SVD clears out high-frequency dynamics generated by the random noise. The noisy data and original system error vector norm was originally of a magnitude around 179. The norm of the error vector between the underlying system and the DMD reconstruction is roughly 78, with small decimal variations.

2.5 Implementation: Van der Pol

From [2], the Van der Pol equation is given by

$$x'' - \mu(1 - x^2)x' + x = 0 \tag{2.16}$$

We drive the system past Hopf bifurcation with parameter: $\mu = 3$. Our initial condition vector is: $\mathbf{x}(0) = [1\ 1]$. The time-step and Hankel matrix setup is the same as the Lorentz setup in the previous sections.

2.5.1 Van der Pol Results: Reconstructing Data

After implementing the DMD algorithm to the Van der Pol Hankel matrix consisting of the x solution elements, we see that the norm of the error vector is in the magnitude of $7 \cdot 10^{-11}$ between the original data and the replicated data. Thus, the approximated linear operator does a good job in recreating the data in this scenario as well.

2.5.2 Van der Pol Results: Prediction Future States

In this non-chaotic scenario, DMD predicts future system values fairly accurately. The data again stops at $t = 25$, and after that, the DMD algorithm continues extending the underlying system. The norm of the error vector is in the magnitude of 0.11.

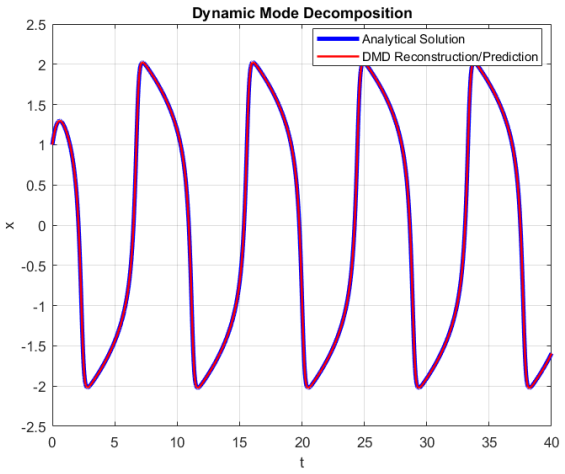

Figure 3: Future State Prediction

This extension of the underlying dynamics approximated with the linear operator does a good job in predicting future states, as it manages to extend the underlying system up to 1500 time steps more. This is a much different result than what was obtained from a chaotic system and shows the true effectiveness of DMD in a non-chaotic nonlinear system.

2.5.3 Van der Pol Results: Noise Reduction

As can be seen from figure 4 and 5, not only does implementing DMD with truncated SVD clears out high-frequency dynamics generated by the random noise, but it also allows for prediction/extension of the underlying system. The noisy data and original system error vector norm was originally of a magnitude around 4.5. The norm of the error vector between underlying system and DMD reconstruction is roughly 1.5, with small decimal variations. As expected, this underlying system reconstruction/prediction will eventually fail due to the noisy dynamics still present in the approximation of the linear operator (leftover from the truncated SVD).

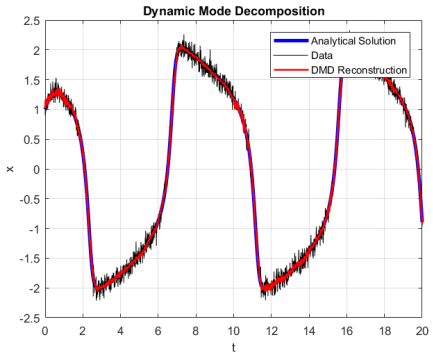

Figure 4: Van der Pol Noise Reduction with $\sigma = .1$.

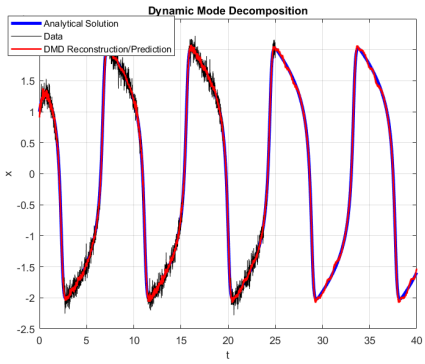

Figure 5: Van der Pol Noise Reduction and future state prediction with $\sigma = .1$.

3 Data Assimilation

The DMD method does not require any governing equation to extract useful information about the dynamics of the system under consideration.
By contrast, the method of *data assimilation* is a hybrid technique that makes use of both data collected in time about the system and a set of equations that govern the dynamics. As both measurements and simulations are influenced by noise fluctuations, the method of *data assimilation* combines the two in order to greatly improve the predictive power of the model.
It is beyond the scope of this paper to discuss the theory of the *data assimilation* method in details, thus we refer to [2] for a more thorough derivation of the algorithm. Instead, we will illustrate the effect of such method by implementing it to model the Lorentz system (2.12) – (2.14).

3.1 Numerical set-up

In order to run our data assimilation algorithm, we need to perform a "perfect" simulation of the system, i.e. without perturbation on the initial state or the observations. Such simulation will serve as the *truth* that our algorithm will try to reproduce. Matlab's built-in function *ODE45* is used to solve the system.

3.1.1 Lorentz Equation

For the Lorentz equation, parameters and initial conditions remain from the DMD section, and the time t goes from 0 to 20 units. It is worth noting that the choice of the parameter r to be as large makes the system highly sensitive to initial conditions [2, 5]. We will limit our analysis to the dynamics of the x-component of the system.
The initial conditions are perturbed and noisy data is randomly-generated every $t = 0.5$ units of time. Both errors will be taken to be Gaussian-distributed random variables with mean 0 and variance 1. There will be no error added to the evolution equations $(2.12 - 2.14)$.

3.1.2 Van der Pol Equation

For the Van der Pol Equation, the parameter μ is given to be equals 3. The initial condition vector is $\mathbf{x}(0) = [1\ 1]$ and the time t goes from 0 to 20 units. The initial conditions are perturbed and noisy data is randomly-generated every $t = 0.5$ units of time. Both errors will be taken to be Gaussian-distributed random variables with mean 0 and variance 1. There will be no error added to equation (2.16).

3.2 Implementation

3.2.1 Lorentz

From [2] and [5] we know the role that sensitivity to initial conditions plays in compromising the ability of the model to correctly predict the future state of the system and our goal is to mitigate this effect by the use of the data assimilation algorithm.
For the perturbed initial conditions, following the notation in [2], we have

$$\mathbf{x}(0) = \mathbf{x}_0 + \sigma_2 \mathbf{q}, \tag{3.1}$$

where $\mathbf{q}$ is the error variance in the initial conditions and σ_2 is chosen to be 1, following [2].
For a M number of observations at a time point t_n it holds

$$\mathbf{y}(t_n) = \mathbf{x}(t_n) + \sigma_3 \mathbf{q} \tag{3.2}$$

where $\mathbf{y}(t_n)$ represents the observations at a time point t_n, $\mathbf{q}$ is the error variance in the data collected and σ_3 is chosen to be 1.
Following the notation in [2], we define the Kalman gain as

$$K = \frac{\sigma_2}{\sigma_2 + \sigma_3} \tag{3.3}$$

and the data assimilation prediction as

$$\bar{x}_{k+1} = x_{k+1} + K_{k+1}(y_{k+1} - x_{0_{k+1}}), \tag{3.4}$$

where x_{k+1} indicates the state of the system at time t_{k+1}, y_{k+1} is the data collected at t_{k+1} and $x_{0_{k+1}}$ is the forecast of the state of the system at t_{k+1} [2]. While, for a complete derivation of the data assimilation algorithm, we refer to [2], a brief outline of the algorithm can be given as follows:

1. Solve the system exactly (this is the true dynamics that the data assimilation algorithm will try to recreate)
2. Perturb the initial conditions and add noise to the data
3. Create a vector to store the data assimilation solution
4. Solve the system with perturbed initial conditions for 0.5 units of time
5. Use perturbed solution as initial condition for future state (i.e. next 0.5 units of time)
6. Define Kalman gain as in (3.3)
7. Compute data-assimilated prediction as in (3.4)
8. Repeat (4-7) and store data-assimilation solution

In the next section we discuss the results obtained by implementing the aforementioned algorithm for the Lorentz system.

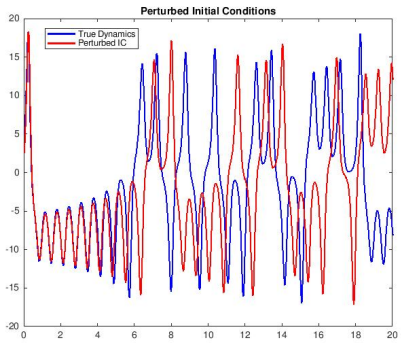

Figure 6: Unperturbed solution vs. perturbation of initial conditions with error variance $\sigma_2 = 1$

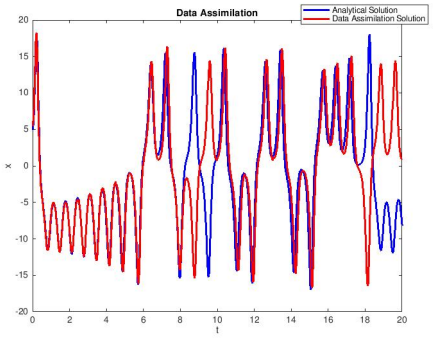

Figure 7: Data-assimilated solution with error variance $\sigma_2 = 1$ and $\sigma_3 = 4$

3.3 Results and Discussion

Our results are displayed in Figure 6 and Figure 7. Figure 6 shows how perturbing the initial conditions affects the dynamics of the system, as the model is not able to reproduce the true dynamics after $t = 5$. From figure 7, it can readily be seen that the data assimilation method (red line) extends the capabilities of the model, as it reproduces the true dynamics up to $t = 8$. However, the two solutions diverge between $t = 8$ and $t = 10$ and again after $t = 18$, indicating that the method, as implemented, is not very accurate for a meaningful-enough time span.
It is unclear at this stage if such divergence is a consequence of some inherent instability of the method itself, or if there were rather some features of the algorithm that were incorrectly implemented. Most likely, the definition of the Kalman gain (3.3) from [2] needs to be modified, as it is given as a simple scalar quantity.
Nevertheless, this result shows the longer accuracy that the data assimilation method compared to the direct simulation of the system. Moreover, it shows the impact of filtering perturbations, especially for systems with high sensitivity to initial conditions (i.e. chaotic) as the one studied in this section.

3.4 Van der Pol

The data assimilation algorithm follows the procedure used in section (3.2.1) in this paper. The parameter μ for the Van der Pol equation is chosen to be 3. The error variances for the initial conditions and the data are chosen to be Gaussian-distributed random variables with, respectively, $\sigma_2 = 0.5$ and $\sigma_3 = 0.1$. Only the dynamics of the x-variable is object of study in the present paper.
The results obtained are displayed in Figures 8 and 9.

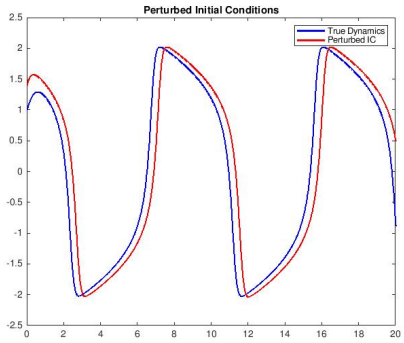

Figure 8: Perturbed initial conditions vs. true dynamics for $\sigma_2 = 0.5$

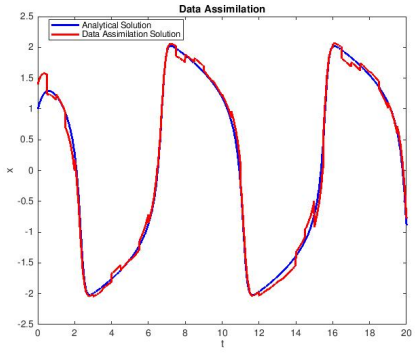

Figure 9: Data assimilation solution vs. true dynamics, for $\sigma_2 = 0.5$, $\sigma_3 = 0.1$

From figure 8, it can be seen that perturbing the initial state of the system has a small, but nonetheless significant, impact on the dynamics, as the solution is shifted upwards and to the right. On the other hand, from Figure 9, it can readily be seen that, in this case, the data assimilation method is not quite accurate in the reproduction of the true dynamics of the system, as it even fails to reproduce the smoothness of the original function.
Once again, a different definition of the Kalman gain other than (3.3) from [2] should probably be considered in future studies.

4 Final Remarks

Our original course of action was to compare both Dynamic Mode Decomposition and the data assimilation method found in [2], but the results are vastly different. Although better data assimilation techniques exist that would have made the comparison feasible, the one implemented in [2] and this paper left much to be desired. On the other hand, the DMD algorithm, although not useful in chaotic systems, performs splendidly when applied to a non-chaotic nonlinear system. So, we have instead shown an overview and uses of both methods when applied to both chaotic and non-chaotic ODEs.

5 Bibliography

References

[1] Grewal, M.S., Andrews, A.P. *Kalman Filtering: Theory and Practice Using MATLAB*. Wiley, 2008.

[2] J.Nathan Kutz. *Data-Driven Modeling & Scientific Computation Methods for Complex Systems and Big Data*. Oxford University Press, 2013.

[3] Joshua L. Proctor, Steven L. Brunton, J. Nathan Kutz. *Dynamic mode decomposition with control*. September 2014.

[4] Peter J. Schmit. *Dynamic Mode Decomposition of Numerical and Experimental Data*. Journal of Fluid Mechanics, vol. 656, 2010, pp. 5–28.

[5] Steven H. Strogatz. *Nonlinear Dynamics and Chaos, with applications to Physics, Biology, Chemistry and Engineering*. Westview Press, 2015

[6] Steven L. Brunton, Bingni W. Brunton, Joshua L. Proctor, Eurika Kaiser, J. Nathan Kutz, *Chaos as an intermittently forced linear system*. Nature Communications vol. 8, 2017

[7] Jonathan H. Tu *Dynamic Mode Decomposition: Theory and Applications*. Diss. Princeton University, 2013.

[8] J. Julier, Simon, K. Uhlmann, Jeffrey. *A New Extension of the Kalman Filter to Nonlinear Systems*. SPIE 3068, Signal Processing, Sensor Fusion, and Target Recognition VI, 28 July 1997.

[9] Xin Liu, Zaiwen Wen and Yin Zhang *Limited Memory Block Krylov Subspace Optimization for Computing Dominant Singular Value Decompositions*. SIAM Journal on Scientific Computing, 35-3 (2013), A1641-A1668.