San Diego State University

Department of Mathematics and Statistics
Math 638
Continuous Dynamical Systems

SAN DIEGO STATE UNIVERSITY

Final Project - First Draft:

Dynamic Mode Decomposition vs. Data Assimilation:
Modeling the Lorentz System

Lourdes Coria,
Horacio Lopez,
Matteo Polimeno

Professor:
Dr. Christopher Curtis

April 17^{th}, 2018

Contents

1 Abstract

The focus of this project are two methods of analyzing and interpreting data dynamics:

1. *Dynamic mode decomposition* (DMD), as an example of equation-free method;

2. *Data-assimilation* method, as an example of hybrid 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 system, and ultimately analyze and compare the accuracy of the results from each method.

2 DMD

2.1 Introduction 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 needs to 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) and it does not require underlying governing equations. Instead, it uses snapshots of measurements to predict and control a system's dynamical behavior.
The DMD method aims to exploit low-dimensionality in a given set of data without having to rely on a set of governing equations. The method gives a decomposition of experimental data into a set of dynamic modes that are collected from the snapshots in a given time interval.

2.2 DMD Methodology

The methodology of the DMD method uses measurement data from numerical simulations or experimental results to extract important dynamic characteristics. Here is the mathematical background of the DMD method.

The data collection process is of utmost importance since the method is data-based driven. Let the following two parameters be defined,

$$N = \frac{\text{number of spatial points saved}}{\text{time snapshot}} \tag{1}$$

$$M = \text{number of snapshots taken} \tag{2}$$

It is crucial to collect data at regularly-spaced time intervals. Let the interval between data collection times be Δt. Then,

$$\Delta t = t_{m+1} - t_m, \tag{3}$$

where the data collection starts at t_1 and ends at t_M. Let $\mathbf{x}$ be the vector of data collection points with length N. Then the data can be arranged into an $N \times M$ matrix,

$$\mathbf{X} = [U(\mathbf{x}, t_1)\ U(\mathbf{x}, t_2)\ U(\mathbf{x}, t_3)\ \cdots\ U(\mathbf{x}, t_M)] \tag{4}$$

The objective of dynamic mode decomposition is to mine the data matrix $\mathbf{X}$ for important dynamical behavior such as unstable growth modes, resonance, and spectral properties.

The following matrix is defined to continue the building of the DMD method,

$$\mathbf{X}_j^k = [U(\mathbf{x}, t_j)\ U(\mathbf{x}, t_{j+1})\ U(\mathbf{x}, t_{j+2})\ \cdots\ U(\mathbf{x}, t_k)] \tag{5}$$

This matrix includes the columns j to k of the original data matrix $\mathbf{X}$. Now that the parameters and matrix have been defined the DMD method uses them to approximate the modes of the Koopman operator[2].

2.2.1 The Koopman Operator and its use in the DMD method

The calculation of the Koopman operator is the core of the DMD methodology. The Koopman operator is a linear, infinite dimensional operator. It represents nonlinear, infinite dimensional dynamics without the need to linearize. It can be thought of as using experimental data to compute the low dimensional modes (the eigenvalues and eigenvectors) of a linear model that approximates the underlying dynamics. This is possible even if the dynamics are nonlinear. Assume the model is linear, then the decomposition results in the frequencies and growth rates associated with each mode [3]. When DMD is applied to the linear model it recovers the leading eigenvalues and eigenvectors.

Now in order to construct the appropriate Koopman operator consider the following. Let $\mathbf{A}$ be the Koopman operator which is linear and time-independent. Now the fundamental assumption that connects the state of a linear dynamical system to the next is, [2, 3],

$$\mathbf{x}_{j+1} = \mathbf{A}\mathbf{x}_j \tag{6}$$

Here j is the specific data collection time and $\mathbf{A}$ is the linear operator the maps the data from time t_j to t_{j+1}. The vector $\mathbf{x}_j$ is an N-dimensional vector of the data points that are collected at time j. Recall that even though the underlying dynamics that may have constructed $\mathbf{x}_j$, the mapping over Δ remains linear.

In order to construct the Koopman operator that best represents the collected data consider the matrix $\mathbf{X}_1^{M-1}$ of the form,

$$\mathbf{X}_1^{M-1} = [\mathbf{x}_1\ \mathbf{x}_2\ \mathbf{x}_3\ \cdots \mathbf{x}_{M-1}] \tag{7}$$

Using equation (6) the matrix can be reduced to,

$$\mathbf{X}_1^{M-1} = \left[\mathbf{x}_1\ \mathbf{A}\mathbf{x}_1\ \mathbf{A}^2\mathbf{x}_1\ \cdots \mathbf{A}^{M-2}\mathbf{x}_1\right] \tag{8}$$

The columns of the matrix $\mathbf{X}_1^{M-1}$ are elements of the Krylov space. The matrix fits the first M-1 data collection points using the Koopman operator $\mathbf{A}$. The last data point, $\mathbf{X}_M$ is represented as,

$$X_M = \sum_{m=1}^{M-1} b_m \mathbf{x}_m + r \tag{9}$$

Here b_m are the coefficients of the Krylov space vectors, $\mathbf{r}$ is the error that lies orthogonal to the Krylov space.

Now the goal of the dimensionality reduction method is to take advantage of any low dimensional structures in the data. To do this the singular-value decomposition (SVD) if (8) is calculated,

$$\mathbf{X}_1^{M-1} = \mathbf{U}\Sigma\mathbf{V}^*$$

Where $\mathbf{U} \in \mathbb{C}^{N\times K}$, $\Sigma \in \mathbb{C}^{K\times K}$, $\mathbf{V} \in \mathbb{C}^{M-1\times K}$, and * represents the conjugate transpose. K represents the reduced SVD's approximation to the rank of the matrix $\mathbf{X}_1^{M-1}$. The data matrix has to be able to be approximated by a low rank matrix to work. DMD takes advantage of the low dimensional structure to predict the future state of the system.

The next step is to generalize (6) to its matrix form,

$$\mathbf{A}\mathbf{X}_1^{M-1} = \mathbf{X}_2^M \tag{10}$$

Using (9) on the right hand side of (10) results in,

$$\mathbf{X}_2^M = \mathbf{X}_1^{M-1}\mathbf{S} + \mathbf{r}g_{M-1}^*. \tag{11}$$

Here $\mathbf{S}$ is

$$\mathbf{S} = \begin{bmatrix} 0 & ... & & 0 & b_1 \\ 1 & \ddots & & 0 & b_2 \\ 0 & \ddots & \ddots & & \vdots \\ \vdots & \ddots & \ddots & 0 & b_{M-2} \\ 0 & ... & 0 & 1 & b_{M-1} \end{bmatrix} \tag{12}$$

.

The eigenvalues of $\mathbf{S}$ approximate some of the eigenvalues of $\mathbf{A}$(Koopman operator). Schmid [4] demonstrated that instead of computing the matrix $\mathbf{S}$ directly, the lower-rank matrix can be computed instead. That is,

$$\tilde{\mathbf{S}} = \mathbf{U}^* \mathbf{X}_2^M \mathbf{V} \Sigma^{-1} \tag{13}$$

Note that $\tilde{\mathbf{S}}$ is related to $\mathbf{S}$ via a similarity transformation. Going back to $\tilde{\mathbf{S}}$, consider the eigenvalue problem associated with it,

$$\tilde{\mathbf{S}} \mathbf{y}_k = \mu_k \mathbf{y}_k \qquad k = 1, 2, 3, \cdots, K \tag{14}$$

Here K represents the rank of the approximation that is being made, μ_k represents the eigenvalues that capture the time dynamics of $\mathbf{A}$ as a time step is taken. In order to construct the DMD modes the eigenvalues and eigenvectors from above can transformed to the original eigenvalues and eigenvectors of $\mathbf{S}$, resulting in,

$$\psi_k = \mathbf{U} \mathbf{y}_k \tag{15}$$

Now that calculations of the low-rank approximations of the eigenvalues and eigenvectors are available the projected future solution can be determined for all time. [2].

Therefore the apporximate solution which will be denoted $\mathbf{x}_{\mathrm{DMD}(t)}$ is given by,

$$\mathbf{x}_{\mathrm{DMD}}(t) = \sum_{k=1}^{K} b_k(0) \psi_k \exp(\omega t) = \boldsymbol{\psi} \mathrm{diag}(\exp(\omega t) \mathbf{b} \tag{16}$$

Here $\omega_k = \frac{\ln(\mu_k)}{\Delta t}$, $b_k(0)$ is the initial amplitude of each mode, $\boldsymbol{\psi}$ is the matrix whose columns are the eigenvectors ψ_k, $\mathrm{diag}(\omega t)$ is a diagonal matrix whose entries are the eigenvalues $\exp(\omega_k t)$, and $\mathbf{b}$ is a vector of the coefficients b_k.In

order to calculate $b_k(0)$ consider the initial snapshot of data $\mathbf{x}_1$ at time = 0. Then equation (16) gives $\mathbf{x}_1\boldsymbol{\psi}\mathbf{b}$. To solve for $\mathbf{b}$ use,

$$\mathbf{b} = \boldsymbol{\psi}^{+}\mathbf{x}_1 \tag{17}$$

Here $\boldsymbol{\psi}^{+}$ is the Moore-Penrose pseudo-inverse which gives the best solution $\mathbf{b}$ in the least squares sense [2].

2.3 Limitations

The DMD method has limitations. One of them is that sampling must be done in equally spaced time intervals. Another limitation is that if data matrix is full rank and the data has no suitable low dimensional structure, then the DMD method fails. The DMD method does not work on chaotic systems. There are modified methods of DMD such as DMD with control that try to improve and address the limitations of the basic DMD method.

2.4 Implementation

From [2], the Lorentz equations are given by

$$x' = \sigma(y - x) \tag{18}$$
$$y' = rx - y - xz \tag{19}$$
$$z' = xy - bz. \tag{20}$$

These equations serve as model for convective-driven atmospheric motion [2, 5].

In order to implement the DMD algorithm for the chaotic Lorenz equation, we numerically solve the equation 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, will be 0.01 seconds. The data matrix will consist of data taken from the x solution of the Lorenz system and will be set up as a Hankel matrix [6] which has the form:

$$\mathbf{H} = \begin{bmatrix} x(t_1) & x(t_2) & x(t_3) & ... & x(t_p) \\ x(t_2) & x(t_3) & x(t_4) & ... & x(t_{p+1}) \\ x(t_3) & x(t_4) & x(t_5) & ... & x(t_{p+2}) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x(t_q) & x(t_{q+1}) & x(t_{q+2}) & ... & x(t_m) \end{bmatrix} \tag{21}$$

.

The DMD method is then implemented using the above approximation to the linear Koopman operator 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 want to be solved for in the prediction.

2.5 Results and Discussion

After implementing the DMD algorithm to the Hankel matrix consisting of the x solution elements, we see through the graph that not only does it look like DMD replicated the existing data accurately in Figure 1, but the norm of the error vector is in the magnitude of $1e-12$. So, the approximated linear operator does a good job recreating the data. But, when it came to predicting future states of the function for the system, as seen in Figure 2, it does not work with chaotic systems.

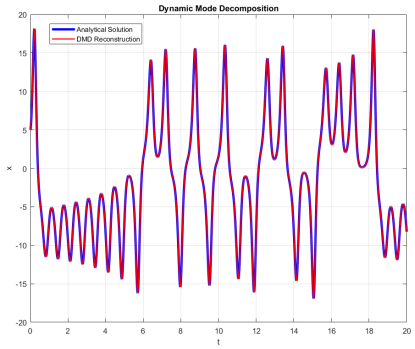

Figure 1: DMD data reconstruction vs. analytical solution for t = [0, 20]

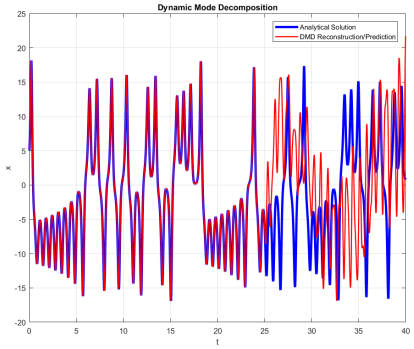

Figure 2: DMD data reconstruction/prediction vs. analytical solution for t = [0, 40]

DMD applied to the Van der Pol equation and proof that DMD does work for non-chaotic systems will be in final draft. Explanation of why it does not work for chaotic systems will also be in final draft.

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 (17) − (19).

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. We will use a fourth-order Runge-Kutta ODE-solver for our simulation.
Standard parameters are given by [2] as $\sigma = 10$, $b = 8/3$ and $r = 28$. The initial condition vector is $\mathbf{x}(0) = [5\ 5\ 5]$ 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.
Initial conditions will be perturbed and noise added to the randomly-generated data. 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 (17), (18) and (19).

3.2 Implementation

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

$$\mathbf{x}(0) = \mathbf{x}_0 + \sigma_2 \mathbf{q}(0, 1), \tag{22}$$

where $\mathbf{q}(0, 1)$ 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

$$\mathbf{y}(t_n) = \mathbf{x}(t_n) + \sigma_3 \mathbf{q}(0, 1) \tag{23}$$

where $\mathbf{y}(t_n)$ represents the observations at a time point t_n, $\mathbf{q}(0, 1)$ is the error variance in the data collected and σ_3 is chosen to be 1, following [2].
The data-assimilated prediction (see [2], chapter 21, p. 528) of the correct state is given by

$$\bar{\mathbf{x}}_{k+1} = \mathbf{x}_{0_{k+1}} + \mathbf{K}_{k+1}(\mathbf{y}_{k+1} - \mathbf{x}_{0_{k+1}}), \tag{24}$$

where $\mathbf{K}_{k+1} = \mathbf{P}_{k+1}(\mathbf{P}_{k+1} + \mathbf{R})^{-1}$ is the so-called Kalman gain matrix, as derived in [2]. $\mathbf{R}$ is the noise covariance matrix, while $\mathbf{P}_{k+1}$ measures the error variance between the true solution and the prediction at t_{k+1}.
From [2], we write the covariance evolution as

$$\mathbf{P}_{k+1} = \mathbf{J}(\mathbf{f})\mathbf{P}_k\mathbf{J}(\mathbf{f})^T, \tag{25}$$

where

$$\mathbf{J}(\mathbf{f}) = \begin{bmatrix} -\sigma & \sigma & 0 \\ r - z & -1 & -x \\ y & x & -b \end{bmatrix} \tag{26}$$

is the Jacobian for the Lorentz' system and $\mathbf{P}_k$ measures the error in estimating the initial state of the system at a time t_k.

Therefore, we can outline the data assimilation algorithm in the following steps:
(i) Determine and incorporate the errors for both the initial conditions and the data measurements to design the matrices $\mathbf{P}_k$ and $\mathbf{R}$, respectively.
(ii) Compute the Jacobian at t_n and use it to compute $\mathbf{P}_{k+1}$.
(iii) Compute the Kalman gain $\mathbf{K}_{k+1}$.
(iv) Compute the new state of the system.
(v) Use the new state of the system to project to a future time when new data is collected again.

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

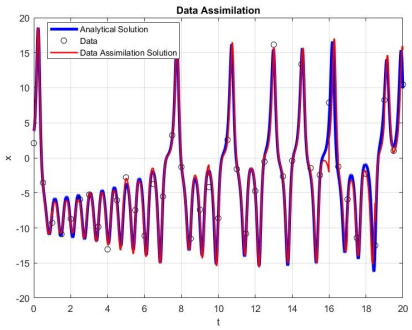

Figure 3: Data Assimilation solution vs. true dynamics

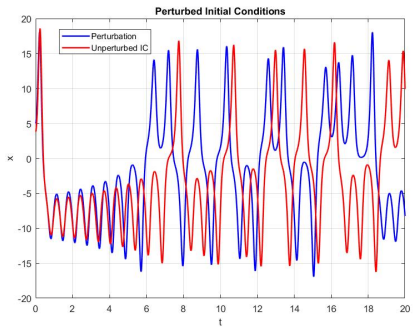

Figure 4: True dynamics vs Perturbed initial conditions

3.3 Results and Discussion

Our results are displayed in the figures above.
From figure 3, we can readily see that the data assimilation method (red line) does an excellent job in reproducing the true dynamics as it is almost indistinguishable from the true solution (blue line) and matches the data (black circles) almost perfectly. There is, however, some divergence in the predictions of the method at around $t \approx 16$ and $t = 18$. 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. This is an aspect that might be the object of further analysis towards the final version of this paper. Nevertheless, the numerical results obtained in this study are in agreement with those derived in [2].
In contrast, figure 4 shows a direct numerical simulation of the perturbed initial conditions. It can readily be seen how the direct simulation (red line) completely fails to predict the true dynamics (blue line) after $t \approx 6$. This result shows the longer accuracy that the data assimilation method has, if 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 paper.

4 Comparison and Final Remarks

To compare both systems we will test their accuracy at both reconstructing the underlying solution through data with noise added to it and at their potential for predicting future states of the system.

As can be seen in Figure 6 and Figure 7, both do a decent job at approximating the underlying solution by clearing out noise.

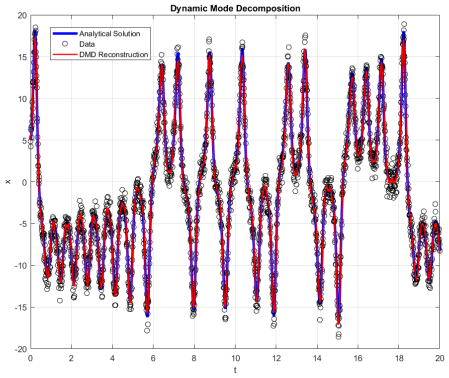

Figure 5: DMD: underlying system reconstruction vs. analytical solution for t = [0, 20]

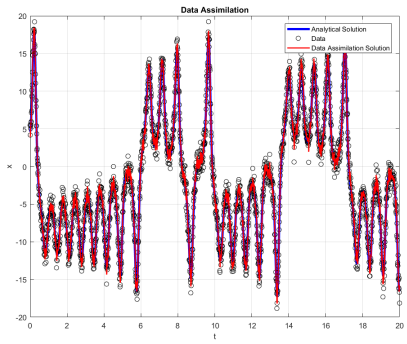

Figure 6: Data assimilation: underlying system reconstruction vs. analytical solution for t = [0, 20]

For DMD we took advantage of the ability to select the dynamics we want to keep through eigenvalue selection and truncation at the time of taking the SVD of the data matrix. This not only reduced the dimensionality of the data matrix, but it cleared out high frequency dynamics generated through the random noise added to the data. If they would have been left in, then the system would have striven to reconstruct the data along with the dynamics in the noise. The norm of the error vector between underlying system and DMD reconstruction was more or less 17 with small decimal variations.

For data assimilation, more noisy data was added to the system in order to create a larger data set. Data assimilation was able to approximate the system with an error vector norm of around 25 with, again, small variations depending on the noise added.

Future state prediction of both systems is difficult, and in the case of DMD impossible, for a chaotic system. Due to the nature of chaos, regardless of how close data assimilation approximates an initial condition close to the underlying system, after a certain number of time steps without any more data added both solutions will develop completely different paths.

The final paper will include analysis of the Van der Pol equation and which method works best at determining future states of the underlying system through noisy data.

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.