Workshop on Molecular Graphics and Visual Analysis of Molecular Data (2018)
J. Byška, M. Krone, and B. Sommer (Editors)

An Accelerated Online PCA with $\mathcal{O}(1)$ Complexity for Learning Molecular Dynamics Data

Salaheddin Alakkari and John Dingliana

Graphics Vision and Visualisation Group (GV2), School of Computer Science and Statistics, Trinity College Dublin

Abstract

In this paper, we discuss the problem of decomposing complex and large Molecular Dynamics trajectory data into simple low-resolution representation using Principal Component Analysis (PCA). Since applying standard PCA for such large data is expensive in terms of space and time complexity, we propose a novel online PCA algorithm with $\mathcal{O}(1)$ complexity per new time-step. Our approach is able to approximate the full dimensional eigenspace per new time-step of MD simulation. Experimental results indicate that our technique provides an effective approximation to the original eigenspace computed using standard PCA in batch mode.

1. Introduction

Eigenvalue problems were first officially introduced by Hilbert in the 1920s (although they had been previously used under different terminology). Since then, they have been widely used for modelling many real-world problems and physical phenomena. Their importance is evidenced by the fact that no less than 20 Nobel prizes have gone to physicists whose significant achievements are related to eigenvalue problems [Tre11]. The Dirac and Schrödinger equations which are central theories in Quantum Physics are both eigenvalue problems that describe particle motion and properties over time. For instance, the quantum states that an electron in an atom can take (labeled as 1S, 2S, 2P etc) are actually time-dependent eigenfunctions [Smi10]. The main advantage that the eigenvalue model provides is that most variations in complex phenomena can be expressed by having merely a selection of a few eigenfunctions and a linear combination of such functions.

Before the computing era, finding eigenvectors and more generally finding eigenfunctions of a specific problem was done analytically, which is a challenging task in many cases. One of the best known techniques for analyzing eigenspace in time-varying data is Normal Mode Analysis (NMA) which approximates the behaviour of the studied system using a combination of harmonic motions. The problem with NMA is that it does not work well with systems of damped or driven oscillations. Principal Component Analysis (PCA) provides an unsupervised non-parametric scheme which finds the eigenspace of observational (or experimental) data automatically. Unlike NMA, PCA can work with any type of dynamics (including damped and driven oscillations) [DS10]. Despite the elegance of PCA, in its standard form, it has large space and time complexity with quadratic dependence on data size. This requires large memory and processing speed. Nowadays machines are shown to be more capable of handling such complexity thanks to larger available memory and faster CPU and GPU (Graphics Processing Unit) capabilities. However, for a wide range of problems where the dimensionality of the data is massive (due to the size and number of samples), extracting the principal components in the standard way becomes infeasible. In addition, the standard approach to PCA is a batch learning technique, meaning that the computation of the eigenspace cannot be done in streaming data scenarios. Many algorithms have been developed to find the most significant principal components incrementally with linear complexity dependence on data size. However most of these approaches are stochastic and are limited to extracting a small number of eigenvectors (principal components).

The use of PCA in the visualization of biomolecular simulations is long established in the literature. PCA is often employed in modelling and visualizing the fundamental modes of motion in complex biomolecule simulations, sometimes referred to as “Essential Dynamics” [ALB93, DJ14]. An example is shown in Figure 1, where the first fundamental mode of motion for a relatively simple molecular dataset, is visualized after it has been captured using PCA. Essential Dynamics can be used, for instance, to capture dynamical hinge-like opening and closing modes which are important to understand the functionality of biological nanomachines such as enzymes. Furthermore PCA has been used in the analysis of ion and water flow in MD simulations, where it is important to understand the distortion dynamics in the system and the transitions between conformers [DS10].

Despite the established importance of PCA to MD analysis and visualization, the inherent computational cost is prohibitive for large time-varying datasets that are often found in this field. Thus, the main contribution of this paper is to provide a novel accel-

DOI: 10.2312/molva.20181100

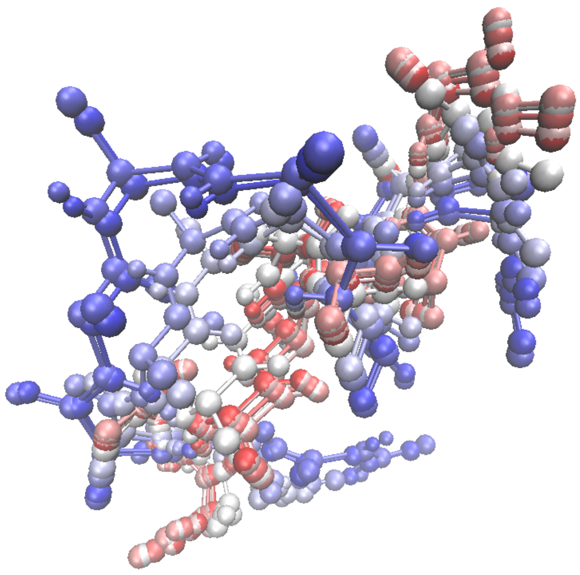

Figure 1: *Superposition of the first eigenvector motion of a molecular simulation dataset using our approach. The spherical nodes represent position of molecules, and colour encodes time: red represents the earliest time-steps and blue corresponds the last time-steps.*

erated variant of PCA that will enable the interactive modelling and visualization of such datasets with minimal loss of quality in the reduced model. In addition, such a solution allows visualization of streaming data, such as in the case of in-situ visualization [NMM*98, SMSS16], interactive simulation [SGS01] or when the full dataset cannot be cached in memory due to hardware limitations. Our technique provides lower computational complexity compared to other recent accelerated PCA algorithms. We test this algorithm on a number of Molecular Dynamics (MD) simulations. We compare the performance of our algorithm with the standard PCA applied in batch-mode and show that our approach provides a very close approximation to the standard approach with a much lower number of computations.

2. Concepts

The standard approach to PCA is as follows. Given data samples $X = [x_1 x_2 \cdots x_n] \in \mathbb{R}^{d\times n}$, where each sample is in column vector format, the covariance matrix is defined as

$$C = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})(x_i - \bar{x})^T, \tag{1}$$

where $\bar{x}$ is the sample mean. In the rest of this paper, we will assume that all samples are centered and hence there is no need to subtract the sample mean explicitly. After that, we can find the optimal low-dimensional bases that cover most of the data variance by extracting the most significant eigenvectors of the covariance matrix C. Eigenvectors are extracted by solving the following characteristic equation

$$(C - \lambda I)v = 0;\ v^T v = 1, \tag{2}$$

where $v \in \mathbb{R}^d$ is the eigenvector and λ is its corresponding eigenvalue. Eigenvalues describe the variance maintained by the corresponding eigenvectors. Hence, we are interested in the subset of eigenvectors that have the highest eigenvalues, $V = [v_1 v_2 \cdots v_p]$; $p \ll n$. Then we encode a given sample x using its p-dimensional projection values (referred to as *scores*) as follows

$$W = V^T x. \tag{3}$$

We can then reconstruct the sample as follows

$$x_{reconstructed} = VW. \tag{4}$$

One advantage of PCA is the low computational complexity when it comes to encoding and reconstructing samples.

2.1. Duality in PCA

Since in the case of $n \ll d$, C will be of rank $n-1$ and hence there are only $n-1$ eigenvectors that can be extracted from Eq. (2) and since C is of size $d \times d$, solving Eq. (2) becomes computationally expensive. We can find such eigenvectors from the dual eigenspace by computing the $n \times n$ matrix $X^T X$ and then solving the eigenvalue problem

$$\left(X^T X - (n-1)\lambda I\right) v_{dual} = 0 \tag{5}$$

$$\Rightarrow X^T X v_{dual} = (n-1)\lambda v_{dual};\ v_{dual}^T v_{dual} = 1. \tag{6}$$

Here, for simplicity, we assumed that the sample mean of X is the zero vector. After extracting the dual eigenvectors, one can note that by multiplying each side of Eq. (6) by X, we have

$$XX^T X v_{dual} = (n-1)\lambda X v_{dual}$$

$$\Rightarrow \frac{1}{n-1}XX^T(Xv_{dual}) = \lambda(Xv_{dual})$$

$$\Rightarrow C(Xv_{dual}) = \lambda(Xv_{dual})$$

$$\Rightarrow (C - \lambda I)(Xv_{dual}) = 0$$

which implies that

$$v = Xv_{dual}. \tag{7}$$

Thus, when $n \ll d$, we only need to extract the dual eigenvectors using Eq. (6) and then compute the real eigenvectors using Eq. (7). Only the first few eigenvectors $V_p = [v_1 v_2 \ldots v_p]$, $p \ll n \ll d$ will be chosen to represent the eigenspace, those with larger eigenvalues.

2.2. PCA for Modelling Dynamics in Time-Varying Data

Considering time-varying data $X = [x_{t_1}, x_{t_2}, \ldots, x_{t_n}] \in \mathbb{R}^{d\times n}$ where d is the total number of attributes. One can think of the time-varying attributes as a network of springs interconnecting particles of unknown masses and unknown settings. In such a case, a negative attribute value indicates a compressed spring and a positive attribute value indicates a stretched one. In this case, the time dependent behaviour of each spring (attribute) can be expressed as follows

$$\frac{\partial^2}{\partial t^2}A - \frac{1}{c^2}HA = 0, \tag{8}$$

where $A(x_0;t) \in \mathbb{R}^d$ is a vector function describing the state of each spring at time t and $x_0 \in \mathbb{R}^d$ are some initial conditions and H is the hessian matrix of all possible second order partial derivatives. By using separation of variables and considering the eigenmode assumption, one can write the solution as follows

$$A(x;t) = F(x)T(t) = Fe^{-iwt}. \tag{9}$$

According to [DPR91] the Hessian matrix can be approximated using the negative inverse of the covariance matrix $H \simeq -C^{-1}$. This is also related to Cramer-Rao lower bound (CRLB). By plugging these into eq. 8 we get

$$\left(C - \left(\frac{c}{w}\right)^2 I\right)A = 0, \tag{10}$$

which is precisely the PCA eigenvalue problem. The interesting part is that the frequency of each eigenvector provides many important physical aspects of the system such as the kinetic and potential energies which are key parts for understanding any MD simulation.

3. Review of Accelerated PCA algorithms

With regard to previous works in the area of PCA complexity optimization, the power iteration remains one of the most popular techniques for finding the top p eigenvectors [GVL12]. In the recent literature, Shamir proposed a stochastic PCA algorithm that is proven to converge faster than the power iteration method [Sha15]. Both techniques have a lower bound complexity of $\mathcal{O}\left(n\log\left(\frac{1}{\varepsilon}\right)\right)$ where ε is the precision of convergence. In addition, both techniques were experimentally tested to extract only a limited number of significant eigenvectors. Arora and De Sa et al. [ACLS12, ACS13, DSOR14] proposed stochastic techniques that are based on the gradient-descent learning rule. The slow convergence rate of the gradient-descent rule is one main limitation of these techniques.

Many algorithms have been developed to find eigenvectors incrementally per new number of time-steps. Such techniques are referred to as incremental PCA algorithms. The update schemes proposed by Krasulina [Kra69] and Oja [Oja82, OJE83] are the most popular incremental PCA techniques which are based on the Hebbian Learning rule. Given a new time-step x_{n+1} and a significant eigenvector v for previous samples, the general update rule according to Hebbian Learning is

$$v^{n+1} = v^n + \alpha\left\langle x_{n+1}, v^n\right\rangle x_{n+1};\; v^{n+1} = \frac{v^{n+1}}{\left\|v^{n+1}\right\|}, \tag{11}$$

where α is the learning rate. This process will keep updating until it converges to a stable state. The speed of convergence of this technique is a matter of ongoing research. Balsubramani et al. [BDF13] found that speed of convergence depends on the learning rate α. Another problem with this technique is that it does not consider change in weightings of previous time-steps. Mitiagkas et al. proposed an incremental PCA algorithm for streaming data with computational complexity of $\mathcal{O}(n\log(n))$ [MCJ13]. Skocaj et al. [SL03] proposed a spatio-temporal weighting scheme to incrementally adapt top eigenvectors. However, the proposed scheme suffers the problem of error propagation.

In terms of online PCA, Feng et al. [FXY12] proposed an online PCA scheme based on the Principal Component Pursuit (PCP) algorithm. The proposed method involves an optimization problem for each new time step of $\mathcal{O}\left(dp^2\right)$ arithmetic operations per new time-step where p is the number of distinct eigenvectors used and d is number of dimensions per time-step. The algorithm was tested on time-steps drawn from normal distribution with sparse corruption noise. Nie et al. [NKW16] compared the Gradient Descent (GD) and Memory Exponential Gradient (MEG) learning schemes when minimizing the PCA regret function in online settings. They found that both techniques reach the optimal lower bound of the regret function (as a function of time). However when expressing the regret function in terms of a "loss budget" factor, the MEG outperforms the GD algorithm. Karnin et al. [KL15, BGKL15] proposed an online PCA algorithm of $\mathcal{O}\left(np/\varepsilon^2\right)$ total time complexity. It is worth mentioning that many of these studies did not include any experimental results to demonstrate performance in practical settings. In addition, most of the recent studies require some regularization parameters for which performance and quality for a change in such values might significantly differ.

4. Our Algorithm

In this section, we will define our learning scheme. Our scheme adapts eigenvectors according to a new time-step with fixed arithmetic operations per new time-step. The main premise of our algorithm is based on the fact that an eigenvector is actually a weighted sum of the input samples. One can show this by rewriting the PCA eigenvalue equation as follows

$$\left(\frac{1}{n-1}\sum_{i=1}^{n} x_i x_i^T - \lambda I\right) v = 0$$

$$\Rightarrow v = \frac{1}{\lambda(n-1)}\sum_{i=1}^{n}\langle x_i, v\rangle x_i.$$

A first guess for an update formula given new time-step x_{n+1} would be

$$v^{t+1} = v^t + \left\langle x_{n+1}, v^t\right\rangle x_{n+1};\; v^{t+1} = \frac{v^{t+1}}{\left\|v^{t+1}\right\|}.$$

This is similar to Oja's update scheme [Oja82] as discusssed in Section 3. The problem with this formula is that it assumes the weightings of previous samples are fixed. As the eigenvector is updated for each new time-step, the weights of previous samples should also be adjusted according to their projections on the updated eigenvector. The change in weights will be proportional to the correlations between previous samples and the new time-step. In particular, the new eigenvector should be updated towards the direction where sample correlations are maximized. In other words, the eigenvector direction is led by time-steps that have higher correlations with other samples. In our algorithm we used the following update rule

$$v^{t+1} = v^t + \left(\sum_{j \in indices} \left\langle v^t, x_j \right\rangle \left\langle x_j, x_{n+1} \right\rangle^2 x_j \right) + \left\langle v^t, x_{n+1} \right\rangle \left(\sum_{j \in indices \cup \{n+1\}} \left\langle x_j, x_{n+1} \right\rangle \right)^2 x_{n+1}. \tag{12}$$

where $indices = \text{rand}(n, processing\_limit)$ is a sample population of *processing_limit* previous time-steps chosen at random. Unlike Oja's method, this is an online scheme that adapts weightings of previous samples based on the squared dot product with the new time-step. In this way, the change in weights will be higher for samples that have higher correlations with the new time-step. In addition, the new time-step is weighted based on the sum of all second order dot products multiplied by new time-step's score $\left\langle v^t, x_{n+1} \right\rangle$.

The full pseudo-code of our algorithm is shown in Algorithm 1. There are two parameters used in our algorithm: *space_limit* which specifies the maximal number of significant eigenvectors to compute and *processing_limit* which specifies the maximal number of dot products to compute per new time-step per eigenvector. As we mentioned earlier, our algorithm is capable of finding all eigenvectors of the data. In its full-dimensional mode, our algorithm starts with two time-steps with $\frac{x_2 - x_1}{\|x_2 - x_1\|}$ as the initial eigenvector and ends with the full-dimensional eigenspace of the data. Line 10 of the algorithm includes the general update rule. Line 11 is used to balance magnitudes of v^t and v^{t+1}. Line 13 performs Gram-Schmidt process to ensure that following update terms will be orthogonal to updated eigenvector. After finishing the loop, $\tilde{X}$ will constitute the least significant eigenvector since it will be perpendicular to all updated components.

Algorithm 1: Online PCA

```
1  for each new time-step $x_{n+1}$ do
2      $X = [X,\ x_{n+1}]$;
3      $\tilde{X} = X$;
4      if $n > processing\_limit$ then
5          $indices = \text{rand}(n, processing\_limit)$;
6      else
7          $indices = 1 : n$;
8      end
9      for $i = 1 : (\min(n, space\_limit) - 1)$ do
10         $\tilde{v} = v_i + \left(\sum_{j=indices} \langle v_i, \tilde{x}_j \rangle \langle \tilde{x}_j, \tilde{x}_{n+1} \rangle^2 \tilde{x}_j\right) + \langle v_i, \tilde{x}_{n+1} \rangle \left(\sum_{j=indices \cup \{n+1\}} \langle \tilde{x}_j, \tilde{x}_{n+1} \rangle\right)^2 \tilde{x}_{n+1}$;
11         $v_i = \tilde{v} + \langle \tilde{v}, v_i \rangle v_i$;
12         $v_i = \frac{v_i}{\|v_i\|}$;
13         $\tilde{X}_{indices \cup \{n+1\}} = \tilde{X}_{indices \cup \{n+1\}} - v_i \left(v_i^T \tilde{X}_{indices \cup \{n+1\}}\right)$;
14     end
15     $v_{\min(n, space\_limit)} = \sum_{j=indices \cup \{n+1\}} \tilde{x}_j$;
16     $n = n + 1$;
17 end
```

4.1. Edge Case Analysis

We show the soundness of our algorithm by assuming that all previous samples are included in the sample population. We study the following two edge cases:

1. x_{n+1} gives $\max \sum_{i=1}^n \langle x_i, x_{n+1} \rangle^2$.
   By theory, this is the case where x_{n+1} corresponds to the most significant eigenvector $x_{n+1} = v^t$. By plugging this into eq. 12, we get
   $$v^{t+1} = v^t + \left(\sum_{j=1}^n \left\langle v^t, x_j \right\rangle^3 x_j \right) + \gamma v^t, \tag{13}$$
   where γ is a scalar value. Now, we need to show that
   $$\sum_{j=1}^n \left\langle v^t, x_j \right\rangle^3 x_j = \alpha v^t, \tag{14}$$
   for a scalar value α. This can be demonstrated as follows
   $$\begin{aligned} \sum_{j=1}^n \left\langle v^t, x_j \right\rangle^3 x_j &= \sum_{j=1}^n x_j \left(x_j^T v^t \right)^3 \\ &= \sum_{j=1}^n x_j x_j^T v^t \left(v^t\right)^T x_j x_j^T v^t \\ &= \sum_{j=1}^n \lambda v^t \left(v^t\right)^T \lambda v^t \\ &= \sum_{j=1}^n \lambda^2 v^t \left(v^t\right)^T v^t \\ &= n \lambda^2 v^t. \end{aligned} \tag{15}$$
   Here we used the orthonormality condition and the the fact that $\mathbb{E}\left(xx^T v\right) = \lambda v$. Hence $v^{t+1} = \left(1 + n\lambda^2 + \gamma\right) v^t$ and since multiplying a vector with a scalar will not change its direction, the first significant eigenvector will not be changed according to Algorithm 1 and x_{n+1} will be eliminated int the first iteration by the Gram-Schmidt process in Line 13. As a result, the remaining eigenvectors will also not be changed.
2. x_{n+1} is orthogonal to all previous samples.
   This is the case where x_{n+1} corresponds to the least significant eigenvector. In such a case, neither one of the $n-1$ eigenvectors in the inner loop of Algorithm 1 will be changed, nor will x_{n+1} be eliminated by the Gram-Schmidt process in Line 13. Hence, x_{n+1} will be left to the newly formed nth eigenvector as in Line 15.

4.2. Computational Complexity Analysis

For convenience, we will denote $p = space\_limit$ and $k = processing\_limit$. By looking at Algorithm 1, one can note that for each new time-step the inner loop (Line 9) will have a maximum of p iterations. Each iteration performs $2(k+1)$ dot products of d multiplications. Since the two parameters are fixed throughout the execution, this requires $2(k+1) \times p = \mathcal{O}(p \times k) = \mathcal{O}(1)$ dot products per new time-step. This means that the number of dot products per new time-step is independent of the total number of samples. Table 1 shows the total time complexity of our algorithm (after processing all samples) compared to recent studies. Our algorithm has linear dependence to number of eigenvectors unlike [FXY13] which has quadratic dependence. Another important advantage of our approach is that it can be implemented in pipeline settings since the computation of v_i^{t+1} does not depend on v_{i+1}^t leading to a reduced cost of $\mathcal{O}(dk(n+p))$.

Table 1: *Time complexity comparison between our approach and recent accelerated PCA algorithms.*

Algorithm	Total time complexity
Our approach	$\mathcal{O}(ndpk)$, $k \ll p$
Robust Online PCA [FXY13]	$\mathcal{O}(ndp^2)$
OPCA [KL15]	$\mathcal{O}(ndp/\varepsilon^3)$, $\varepsilon \ll 1$
Incremental approaches [Sha15]	$\mathcal{O}(n(d+\frac{1}{\lambda^2})p\log(1/\varepsilon))$, λ is the eigengap

5. Experimental Results

In this section we evaluate the application of our technique to three Molecular Dynamics datasets, as listed in Table 2, that are chosen to represent low, medium and high-complexity datasets encountered in the field. We compare our results with *standard PCA* results, which we generated using the *pcacov* function in MATLAB [Mat17]. The choice of *processing_limit* value is driven by the required processing time per eigenvector update. Larger values lead to better quality at the expense of increased run-time. In scenarios where the data needs to be streamed, such as in-situ visualization, it is important to process the eigenvector before receiving the next time-step, and it is such highly demanding applications that our solution is particular catered to support. Thus, in the results shown in this paper, we chose a *processing_limit* of 20, which was proven by trial-and-error to provide high quality, whilst guaranteeing high processing rates.

Table 2: *Summary of each MD dataset.*

dataset name	No. of atoms	No. of time-steps	Publication
Alanin	66	100	VMD Development Team
gp45 clamps	10,602	2,000	[Oak16]
Ca2+ binding	50,805	5,000	[GBS*17]

Firstly, we compare the quality of our results with the standard PCA approach in terms of *explained variance* curves, which show the percentage of variance maintained up to a certain number of significant eigenvectors. The reason we use explained variance is because the PCA optimization problem aims to maximize variance using a minimal number of dimensions. Figure 2 shows the explained variance curve using each technique for each dataset. For the Alanin dataset both approaches cover 97% of the variability using 20 eigenvectors. For the gp45 data, almost 98% variance is maintained by 100 out of 2,000 eigenvectors. For Ca2+ data, our technique covers 87% of variance using 300 out of 5,000 eigenvectors while the standard approach was able to capture 91%. The scores of time-steps on the first three eigenvectors are shown in Figure 3. It is clear that our technique is quite consistent with the standard PCA. One can also note that lower significant eigenvectors have higher frequency with lower amplitude than significant

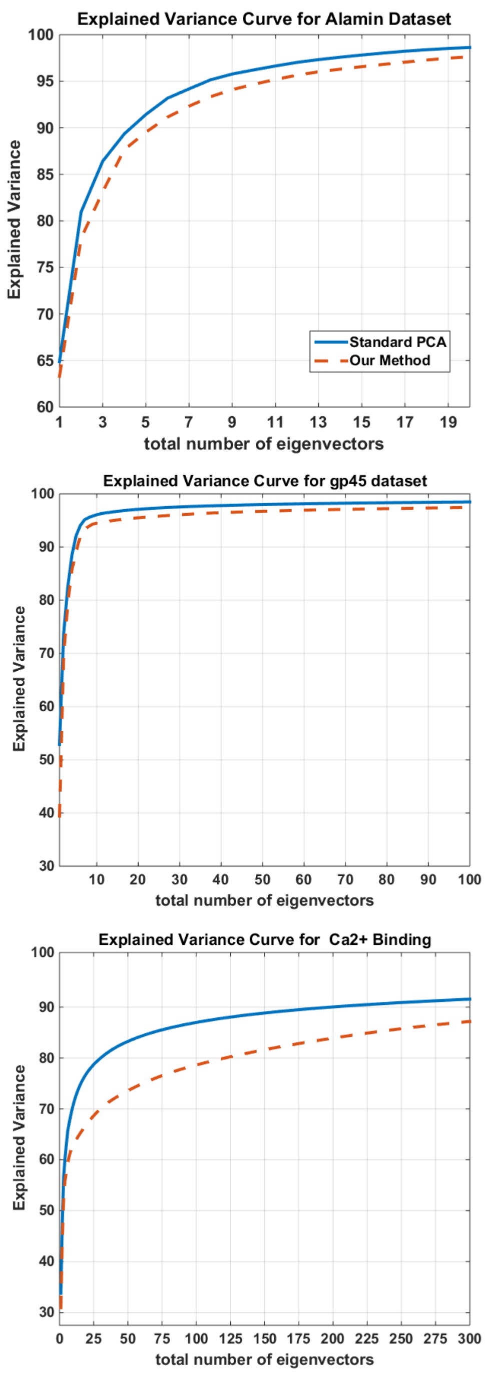

Figure 2: *Explained variance curves of standard PCA and our approach for each dataset.*

ones, which is consistent with our analysis and the theory of normal modes.

In addition, we compared PCA and our method by computing the Peak Signal to Noise Ratio (PSNR), which is a commonly used metric for measuring the quality of reconstructions of signals. Table

3 shows a comparison between the reconstructed frames in terms of mean PSNR value in relation to the original time-steps. Higher PSNR values indicated better reconstruction quality. For the online approach we found that the PSNR values exhibited an oscillating behaviour with many frames being approximated better than the standard approach, however the mean PSNR for the online approach was lower by 1-2 dB than standard PCA. The standard PCA, on the other hand, is more consistent, maintaining a relatively stable value for all frames. Figure 4 shows visualizations of some reconstructed samples of the three datasets using each technique in comparison with the original time-steps. It can be seen that the reconstructed samples using both techniques are very similar to the original ones. It should be noted however that a per-frame visual match is not really the goal, instead our objective is to ensure that the temporal behaviour (or animation) preserves the important behavioural features of the original data. However, the visual similarities seen in this figure along with the explained variance and PSNR results suggest that our technique provides very close results to the standard PCA alternative.

Table 3: *Mean PSNR values for each datast.*

	Alanin	gp45	Ca2+
our approach	15.4 dB	6.5 dB	7.5 dB
standard PCA	16.6 dB	7.8 dB	8.2 dB

Finally, we recorded the run-time measurements of the performance of both techniques on a MS Windows PC equipped with an Intel Xeon 3.5GHz CPU and 16GB RAM. The processing times for Alanin, gp45 and Ca2+ respectively were 0.01s, 6.5s and 2069s using standard PCA. In comparison, the online PCA measurements were 3.5ms, 12s and 165s where a single eigenvector update took 0.035ms, 6ms and 33ms respectively. Although the standard PCA is faster for smaller size datasets, as the number of samples grows (especially with samples of large ambient size) computing the eigenvectors using the standard approach becomes infeasible while the online approach remains efficient by maintaining linear time growth.

6. Conclusion

In this paper, we proposed an online PCA scheme for learning trajectories in MD simulations. Our method performs a fixed number of arithmetic operations per new time-step and hence has $\mathcal{O}(1)$ complexity. Our approach is a generalization of Oja's learning rule which considers the change in weightings for previous time-steps after each update operation. In comparison to the standard PCA, our technique provides an incremental low-computational learning platform with similar quality performance in terms of explained variance curves. Our technique serves as a robust tool for decomposition of complex time-dependent systems. This is a valuable component in the visualization and analysis of the behaviour of MD simulations.

7. Acknowledgements

This research has been conducted with the financial support of Science Foundation Ireland (SFI) under Grant Number 13/IA/1895.

References

[ACLS12] ARORA R., COTTER A., LIVESCU K., SREBRO N.: Stochastic optimization for PCA and PLS. In *Communication, Control, and Computing (Allerton), 2012 50th Annual Allerton Conference on* (2012), IEEE, pp. 861–868. 3

[ACS13] ARORA R., COTTER A., SREBRO N.: Stochastic optimization of PCA with capped MSG. In *Advances in Neural Information Processing Systems* (2013), pp. 1815–1823. 3

[ALB93] AMADEI A., LINSSEN A., BERENDSEN H. J.: Essential dynamics of proteins. *Proteins: Structure, Function, and Bioinformatics 17*, 4 (1993), 412–425. 1

[BDF13] BALSUBRAMANI A., DASGUPTA S., FREUND Y.: The fast convergence of incremental PCA. In *Advances in Neural Information Processing Systems* (2013), pp. 3174–3182. 3

[BGKL15] BOUTSIDIS C., GARBER D., KARNIN Z., LIBERTY E.: Online principal components analysis. In *Proceedings of the twenty-sixth annual ACM-SIAM symposium on Discrete algorithms* (2015), Society for Industrial and Applied Mathematics, pp. 887–901. 3

[DJ14] DAVID C. C., JACOBS D. J.: Principal component analysis: a method for determining the essential dynamics of proteins. In *Protein dynamics*. Springer, 2014, pp. 193–226. 1

[DPR91] DOVI V., PALADINO O., REVERBERI A.: Some remarks on the use of the inverse hessian matrix of the likelihood function in the estimation of statistical properties of parameters. *Applied Mathematics Letters 4*, 1 (1991), 87–90. 3

[DS10] DYKEMAN E. C., SANKEY O. F.: Normal mode analysis and applications in biological physics. *Journal of Physics: Condensed Matter 22*, 42 (2010), 423202. 1

[DSOR14] DE SA C., OLUKOTUN K., RÉ C.: Global convergence of stochastic gradient descent for some non-convex matrix problems. *arXiv preprint arXiv:1411.1134* (2014). 3

[FXY12] FENG J., XU H., YAN S.: Robust PCA in high-dimension: A deterministic approach. *arXiv preprint arXiv:1206.4628* (2012). 3

[FXY13] FENG J., XU H., YAN S.: Online robust PCA via stochastic optimization. In *Advances in Neural Information Processing Systems* (2013), pp. 404–412. 5

[GBS*17] GIORGIO V., BURCHELL V., SCHIAVONE M., BASSOT C., MINERVINI G., PETRONILLI V., ARGENTON F., FORTE M., TOSATTO S., LIPPE G., ET AL.: Ca2+ binding to f-atp synthase β subunit triggers the mitochondrial permeability transition. *EMBO reports 18*, 7 (2017), 1065–1076. 5

[GVL12] GOLUB G. H., VAN LOAN C. F.: *Matrix computations*, vol. 3. JHU Press, 2012. 3

[KL15] KARNIN Z., LIBERTY E.: Online PCA with spectral bounds. In *Conference on Learning Theory* (2015), pp. 1129–1140. 3, 5

[Kra69] KRASULINA T.: A method of stochastic approximation for the determination of the least eigenvalue of a symmetric matrix. *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki 9*, 6 (1969), 1383–1387. 3

[Mat17] THE MATHWORKS, INC.: *MATLAB and Statistics Toolbox Release 2017a*. Natick, Massachusetts, 2017. 5

[MCJ13] MITLIAGKAS I., CARAMANIS C., JAIN P.: Memory limited, streaming PCA. In *Advances in Neural Information Processing Systems* (2013), pp. 2886–2894. 3

[NKW16] NIE J., KOTLOWSKI W., WARMUTH M. K.: Online PCA with optimal regret. *Journal of Machine Learning Research 17*, 173 (2016), 1–49. 3

[NMM*98] NELMS B. E., MASER R. S., MACKAY J. F., LAGALLY M. G., PETRINI J. H.: In situ visualization of DNA double-strand break repair in human fibroblasts. *Science 280*, 5363 (1998), 590–592. 2

[Oak16] OAKLEY A. J.: Dynamics of open DNA sliding clamps. *PloS one 11*, 5 (2016), e0154899. 5

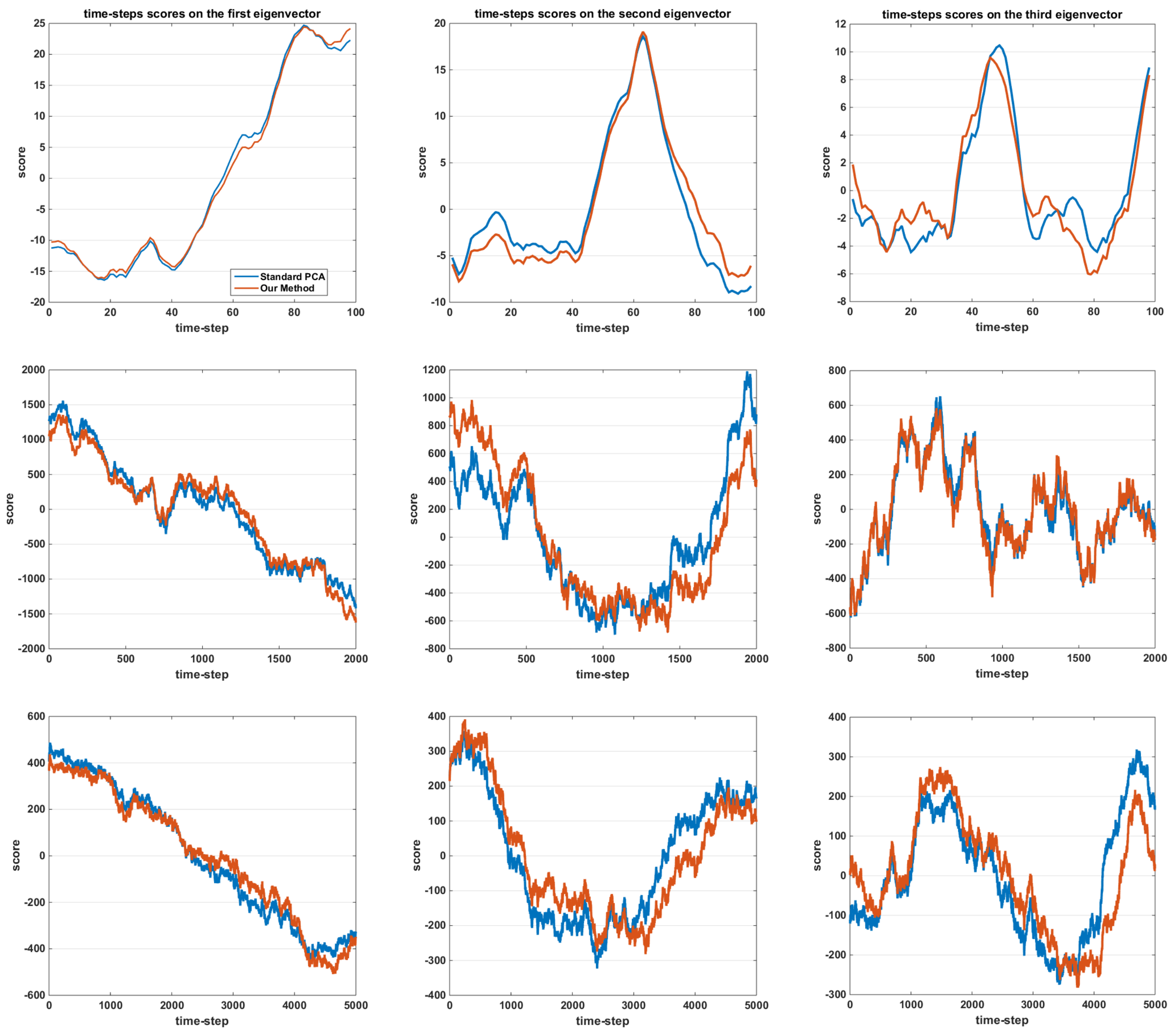

Figure 3: *Scores on the first three eigenvectors using each technique for Alanin dataset (top), gp45 clams (middle) and Ca2+ bindings (bottom). One can note how scores of both techniques are consistent.*

[Oja82] OJA E.: Simplified neuron model as a principal component analyzer. *Journal of mathematical biology 15*, 3 (1982), 267–273. 3

[OJE83] OJE E.: Subspace methods of pattern recognition. In *Pattern recognition and image processing series* (1983), vol. 6, Research Studies Press. 3

[SGS01] STONE J. E., GULLINGSRUD J., SCHULTEN K.: A system for interactive molecular dynamics simulation. In *Proceedings of the 2001 Symposium on Interactive 3D Graphics* (New York, NY, USA, 2001), I3D '01, ACM, pp. 191–194. URL: http://doi.acm.org/10.1145/364338.364398, doi:10.1145/364338.364398. 2

[Sha15] SHAMIR O.: A stochastic PCA and SVD algorithm with an exponential convergence rate. In *ICML* (2015), pp. 144–152. 3, 5

[SL03] SKOCAJ D., LEONARDIS A.: *Weighted and robust incremental method for subspace learning*. IEEE, 2003. 3

[Smi10] SMITH W. F.: *Waves and oscillations: a prelude to quantum mechanics*. Oxford University Press, 2010. 1

[SMSS16] STONE J. E., MESSMER P., SISNEROS R., SCHULTEN K.: High performance molecular visualization: In-situ and parallel rendering with EGL. In *2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)* (May 2016), pp. 1014–1023. doi:10.1109/IPDPSW.2016.127. 2

[Tre11] TREFETHEN N.: Favorite eigenvalue problems. *SIAM News 44*, 10 (2011). 1

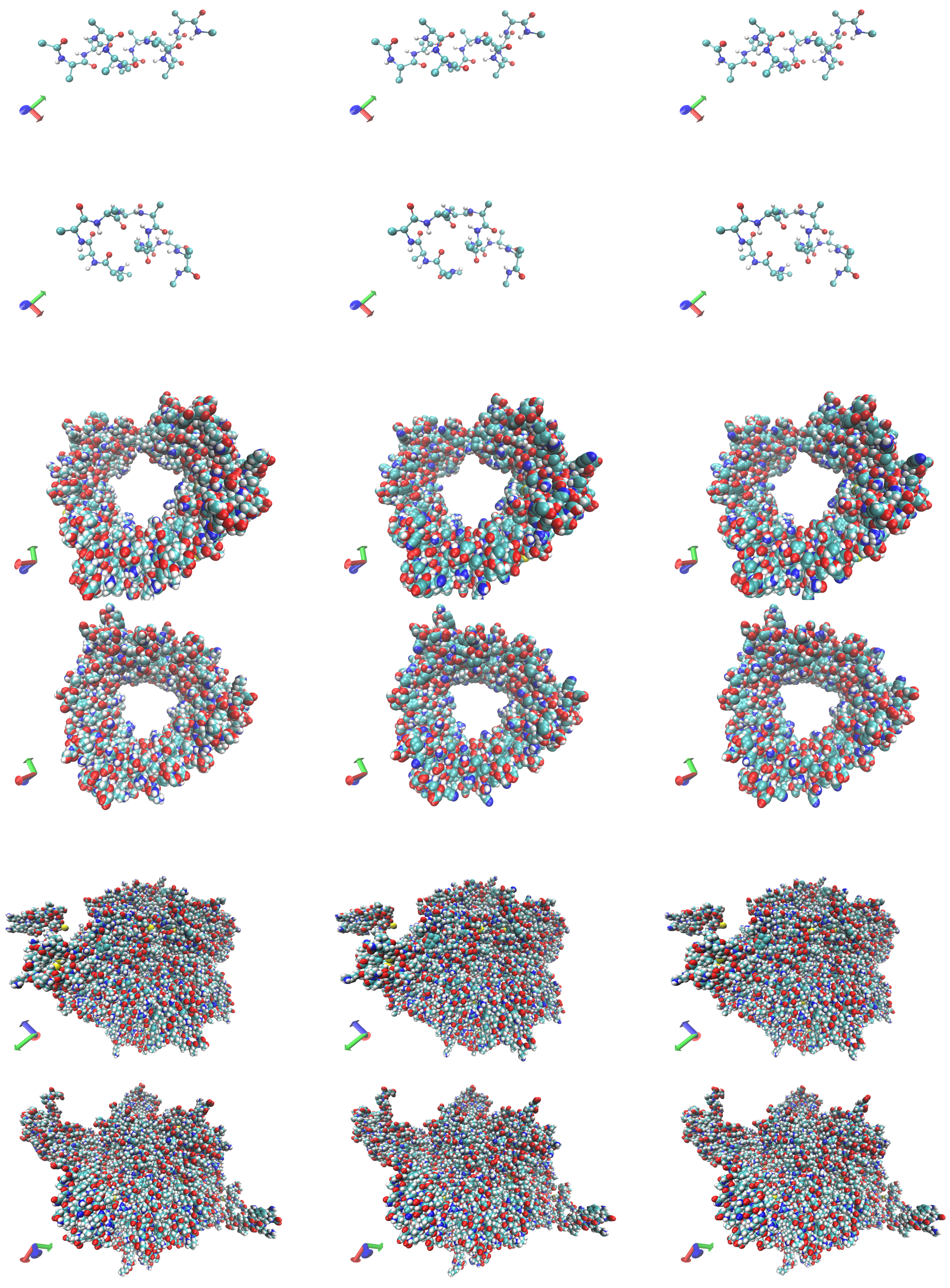

Figure 4: *Original frames from three MD datasets (left) compared to reconstructed frames using* standard PCA *(middle) and our technique (right). The top two rows are frames from the Alanin dataset, the third and fourth rows are frames from the gp45 bindings dataset and the bottom two rows are from the Ca2+ dataset. There are small differences in fine details however both PCA and our solution capture the overall behaviour of the system quite effectively.*