Reconstructing the flows from their reduced-order models

Di Labbio G†, Kadem L
Laboratory of Cardiovascular Fluid Dynamics, Concordia University, Montréal, QC, Canada, H3G 1M8

Overview

Here we describe some important details about the provided data, including both how they were produced and how to reconstruct the intraventricular flows from them. The data can be found at https://github.com/dilabbiog/ROMs--LV_flow_with_AR. The data is defined on a Cartesian grid, where we have adopted the convention that the lower-leftmost grid point corresponds to the origin of the coordinate system, i.e., $(x, y) = (0, 0)$. The grid points along the x and y axes are provided as column vectors in the files "`X.dat`" and "`Y.dat`" respectively. For node numbering, we take the convention of expressing the grid as a two-dimensional array, where element $(1, 1)$ corresponds to the upper-leftmost grid point; refer to the schematic in Fig. S1a.

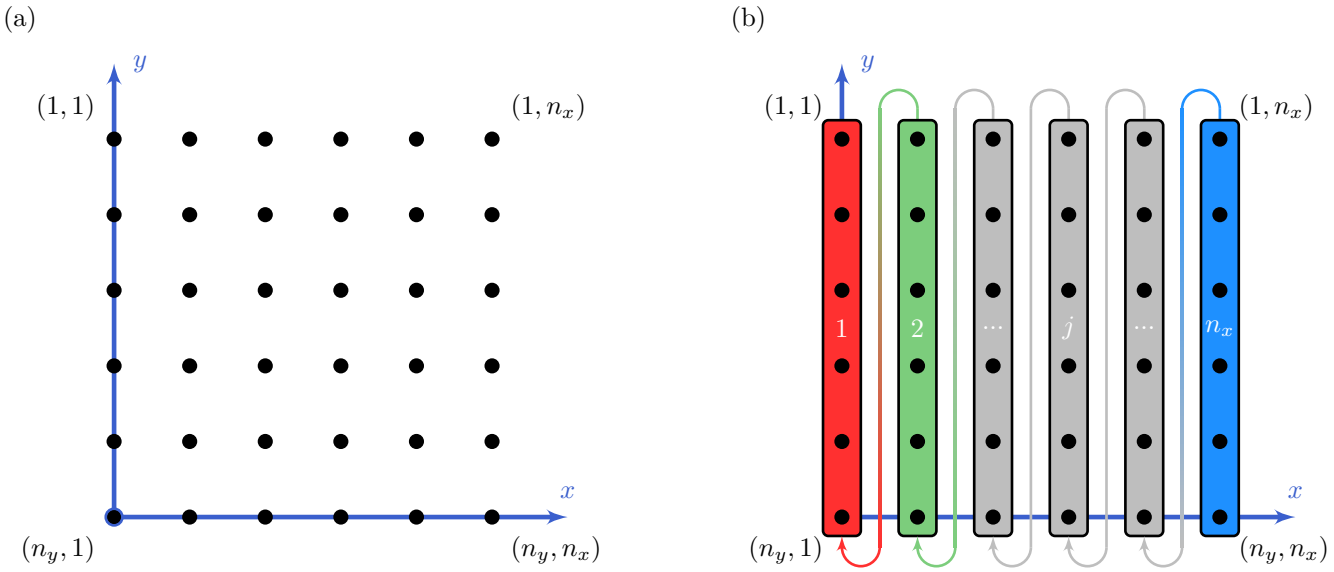

Figure S1: (a) Schematic of the Cartesian grid corresponding to the data, showing the origin of the coordinate system and the numbering of the grid points; n_x denotes the number of grid points along the x direction and n_y that along the y direction. (b) Illustration of the manner in which data is reorganized into a single column in the provided data files.

As briefly discussed in the main text, reconstruction of the flow fields will inevitably result in velocity vectors falling outside the ventricle boundaries. We therefore also provide the mask information for each case in the files "`Case_Mask.dat`" which can be applied to the flow after reconstruction. The columns in these files correspond to the mask at any given time for a total of $n = 343$ time steps. Recall that the 343 time steps were acquired at 400 Hz ($\Delta t = 2.5$ ms) and so the data corresponds to almost one complete cardiac cycle ($T = 0.8571$ s; i.e., 70 bpm) with a 2.1 ms undershoot. The mask has a value of unity inside the boundary of the left ventricle and is zero outside. Each column in the provided file contains the mask values at all grid points arranged in a column-under-column fashion as shown schematically in Fig. S1b. The grid for the provided datasets contains 75 points along the y direction and 82 along the x direction, corresponding to a resolution of 1.04 mm × 1.04 mm. Note that this is half the resolution used in the main text, however we found that using a simple bilinear or bicubic interpolation on the reconstructed datasets to recover the 0.52 mm × 0.52 mm resolution gives the same results to within 0.04% (i.e., the reconstructed data is rather smooth). We therefore decided to provide the subsampled data to reduce the overall file sizes at

† **Email address for correspondence:** g_dilabb@encs.concordia.ca

effectively no cost in accuracy. We would also like to note that the mitral inflow is always on the right side of the left ventricle in the datasets and the aortic outflow is therefore on the left. Code S1 shows a simple MATLAB script to read the mask and arrange the data into a three-dimensional array, with the third dimension representing the time step. Although provided for MATLAB, this code, as well as those provided in the two subsequent sections, can be used as pseudocode for other programming languages (we have added comments in the codes throughout this supplementary document). Should the reader have any questions for reconstructing the data, please contact the corresponding author.[†]

Code S1 – Reading the Mask File in MATLAB

```
>> fileID = fopen('Case_Mask.dat', 'rt');          % Open the file for reading.
>> format = repmat('%f', 1, 343);                  % Number format in the file.
>> mask = cell2mat(textscan(fileID, format));      % Read the file into a 2D array.
>> mask = reshape(mask, 75, 82, 343);              % Reshape the data into a 3D array.
>> fclose(fileID);                                 % Close the file.
>> clear fileID format;                            % Delete unnecessary variables.
```

The user should note that, as discussed in the text, the DMD models were produced by applying DMD with the snapshots ordered such that the flows begin with the **filling phase**. In the case of POD, although the snapshot ordering is not important, it was applied with the snapshots ordered such that the flows begin with the **ejection phase**. Nonetheless, in order to avoid confusion, for all the provided models the resulting temporal dynamics have been shifted so that the flows begin with the same instant of the cardiac cycle. That is to say that after reconstruction, snapshot 1 will correspond to $t^* = -0.04375$, snapshot 16 to $t^* = 0$, snapshot 166 to $t^* = 0.4375$ (the start of the filling phase) and snapshot 343 to $t^* = 0.95375$. The first 15 snapshots of the acquired data in fact belong to the filling phase, hence why $t^* = 0$ corresponds to snapshot 16. In the case of the DMD models, the first 15 snapshots may simply be moved to the end of the dataset since the temporal dynamics are well-behaved. In the case of the POD models, while this can still be done, there will be a slight discontinuity observed in the temporal dynamics (see Fig. 6 in the main text for example). Therefore, for the POD models, the user may want to instead consider extrapolating the temporal dynamics for these 15 snapshots so that the temporal dynamics are in better correspondence between the first and last snapshots.

Reconstruction from DMD Data

The healthy intraventricular flow as well as those in the presence of mild and moderate aortic valve regurgitation are described by reduced-order models constructed using dynamic mode decomposition. In order to generate the data, we have used the exact DMD method of Tu et al. (2014) and have summarized the algorithm in Code S2 below.

In order to simplify the reconstruction process, we provide data files of the dynamic modes and their corresponding temporal dynamics. The modes and dynamics are generally complex-valued and so separate data files were provided for their real and imaginary parts. The dynamic modes are contained in the files "`Case_Modes_Re.dat`" and "`Case_Modes_Im.dat`," where the real and imaginary parts are stored respectively. The modes are represented by the columns in the files, where each column contains the u component of velocity at all grid points followed by the v component at all grid points as in Eq. (S1). The values are again arranged column-under-column as shown in Fig. S1b.

$$\boldsymbol{\psi}_j = \begin{bmatrix} | \\ \boldsymbol{\psi}_j^{(u)} \\ | \\ \boldsymbol{\psi}_j^{(v)} \\ | \end{bmatrix} \tag{S1}$$

Code S2 – Exact Dynamic Mode Decomposition Algorithm for MATLAB

```
>> X = [reshape(VELu,[],n); reshape(VELv,[],n)];   % Construct the data matrix X.
>> [U,Sigma,V] = svd(X(:,1:n-1),0);                % Compute the (economy-sized) singular value
>>                                                 % decomposition of the matrix X_1^{n-1}.
>> Atilde = U'*X(:,2:n)*V/Sigma;                   % Compute the matrix Ã = U*X_2^n V Σ^{-1}.
>> [W,Lambda] = eig(Atilde, 'vector');             % Compute the eigenvalues λ_k and the
>>                                                 % corresponding eigenvectors w_k of Ã.
>> Psi = X(:,2:n)*V/Sigma*W/diag(Lambda);          % Compute the dynamic modes ψ_k
>>                                                 % from Ψ = X_2^n V Σ^{-1} W Λ^{-1}.
>> b = diag(Lambda)\(Psi\X(:,2));                  % Compute the amplitudes of the modes from
>>                                                 % b = Λ^{-1} Ψ^+ x_2.
>> % Alternatively, the above line may be written as b = diag(Lambda)\pinv(Psi)*X(:,2);
>> Tdyn = diag(b)*fliplr(vander(Lambda));          % Compute the temporal dynamics.
>> Gamma = log(Lambda)/dt;                         % Map the Ritz values using γ_k = ln(λ_k)/Δt.
```

Code S3 – Flow Reconstruction from DMD Data in MATLAB

```
>> fileID = fopen('Case_Modes_Re.dat', 'rt');      % Open the modes file (real) for reading.
>> format = repmat('%f', 1, N);                    % Number format (N = 49, 41, 61 or 47).
>> ModesRe = cell2mat(textscan(fileID, format));  % Read the file into a 2D array.
>> fclose(fileID);                                 % Close the file.
>> % Repeat the above 4 lines for the imaginary part of the modes and the temporal dynamics.
>> Modes = ModesRe + 1i*ModesIm;                   % Recombine the complex-valued modes.
>> Tdyn = (TdynRe + 1i*TdynIm).';                  % Recombine the complex-valued dynamics.
>> Flow = real(Modes*Tdyn);                        % Reconstruct the flow.
>> U = reshape(Flow(1:6150,:), 75, 82, 343);       % Reshape the u component of velocity.
>> V = reshape(Flow(6151:end,:), 75, 82, 343);     % Reshape the v component of velocity.
>> U = mask.*U; V = mask.*V;                       % Apply the mask (element-wise product).
>> clear fileID Flow format Modes ModesIm;         % Delete unnecessary variables.
>> clear ModesRe N Tdyn TdynIm TdynRe;             % Delete unnecessary variables.
```

Similarly, the temporal dynamics are contained in the files "`Case_Dynamics_Re.dat`" and "`Case_Dynamics_Im.dat`," where again the real and imaginary parts are stored respectively. Each column in the files represents the temporal dynamics for a given mode. Recall from the main text that 49, 41, 61 and 47 modes are respectively used for the 99.5% DMD models of the healthy, mild, moderate-1 and moderate-2 cases, which effectively correspond to the number of columns in the mode and temporal dynamics files. Once the data is read, flow reconstruction from the DMD models then only requires the application of the formula $\mathbf{X} = \mathbf{\Psi B T}$ followed by reshaping of the data, where $\mathbf{X}$ contains the reconstructed data in its columns, $\mathbf{\Psi}$ contains the dynamic modes in its columns, and the product $\mathbf{BT}$ contains the temporal dynamics in its rows. The MATLAB script to reconstruct the data is provided in Code S3 below. The user is encouraged to verify that the imaginary part of the reconstructed flow (line 8 in Code S3) is negligible prior to taking the real part.

Reconstruction from POD Data

The intraventricular flows subject to severe aortic valve regurgitation are described by reduced-order models constructed using proper orthogonal decomposition. In order to generate the data, we have used the snapshot POD method of Sirovich (1987) and have summarized the algorithm in Code S4 below.

Again, in order to simplify the reconstruction process, we provide data files of the proper orthogonal modes and their corresponding temporal dynamics. The proper orthogonal modes are contained in the "`Case_Modes.dat`" files.

Code S4 – Snapshot Proper Orthogonal Decomposition Algorithm for MATLAB

```
>> X = [reshape(VELu,[],n); reshape(VELv,[],n)];  % Construct the data matrix X.
>> C = X'*X;                                      % Compute the temporal correlation matrix
>>                                                % C = X^T X.
>> [Q,Lambda] = eig(C,'vector');                  % Compute the eigenvalues λ_k and
>>                                                % eigenvectors q_k of C.
>> [Lambda,ind] = sort(Lambda,'descend');         % Sort the eigenvalues and eigenvectors
>> Q = Q(:,ind);                                  % from largest to smallest.
>>                                                % corresponding eigenvectors w_k of Ã.
>> Phi = normc(X*Q);                              % Compute the normalized proper orthogonal
>>                                                % modes from Φ = XQΛ^(-1/2).
>> % Alternatively, the above line may be written as Phi = X*Q*diag(Lambda)^(-1/2);
>> B = Phi'*X;                                    % Compute the amplitudes of the modes from
>>                                                % B = Φ^T X.
```

The modes are represented by the columns in the files in the same way as for the DMD. The temporal dynamics are contained in the "`Case_Dynamics.dat`" files with each column representing the temporal dynamics for a given mode. Recall from the main text that 84 and 71 modes are respectively used for the 99.5% POD models of the severe-1 and severe-2 cases, which again correspond to the number of columns in the mode and temporal dynamics files. Once the data is read, flow reconstruction from the POD models then only requires the application of the formula $\mathbf{X} = \mathbf{\Phi B}$ followed by reshaping of the data, where $\mathbf{X}$ contains the reconstructed data in its columns, $\mathbf{\Phi}$ contains the proper orthogonal modes in its columns, and $\mathbf{B}$ contains the temporal dynamics in its rows. The MATLAB script to reconstruct the data is provided in Code S5 below.

Code S5 – Flow Reconstruction from POD Data in MATLAB

```
>> fileID = fopen('Case_Modes.dat', 'rt');        % Open the modes file for reading.
>> format = repmat('%f', 1, N);                   % Number format (N = 84 or 71).
>> Modes = cell2mat(textscan(fileID, format));    % Read the file into a 2D array.
>> fclose(fileID);                                % Close the file.
>> % Repeat the above 4 lines for the temporal dynamics.
>> Flow = Modes*(Tdyn.');                         % Reconstruct the flow.
>> U = reshape(Flow(1:6150,:), 75, 82, 343);      % Reshape the u component of velocity.
>> V = reshape(Flow(6151:end,:), 75, 82, 343);    % Reshape the v component of velocity.
>> U = mask.*U; V = mask.*V;                      % Apply the mask (element-wise product).
>> clear fileID Flow format Modes N Tdyn;         % Delete unnecessary variables.
```

Summary of Reconstruction Errors

In the main text, the errors associated with the reduced-order models were assessed using the circulation within the left ventricle as well as the total viscous energy loss; please refer to the main text for the details. The ability of the models to reconstruct these metrics compared to the original data are given below in Figs. S2a and S2b respectively.

Recall that the data being modeled resulted from the ensemble-averages of ten time-resolved acquisitions made for each simulated case. As a result, the viscous dissipation of the data being modeled in Fig. S2b (i.e., the open circles) is underestimated from its true value; please refer to both Sec. II and Appendix C in the main text for further detail. With regard to the error associated with global particle advection patterns, in the text we have advected $\sim 800,000$ virtual particles at the start of the ejection phase ($t^* = 0$) for 4 cardiac cycles and used the fraction of the initial particles remaining as the error metric. We display the result here in Fig. S3a, showing that up to 2 cardiac

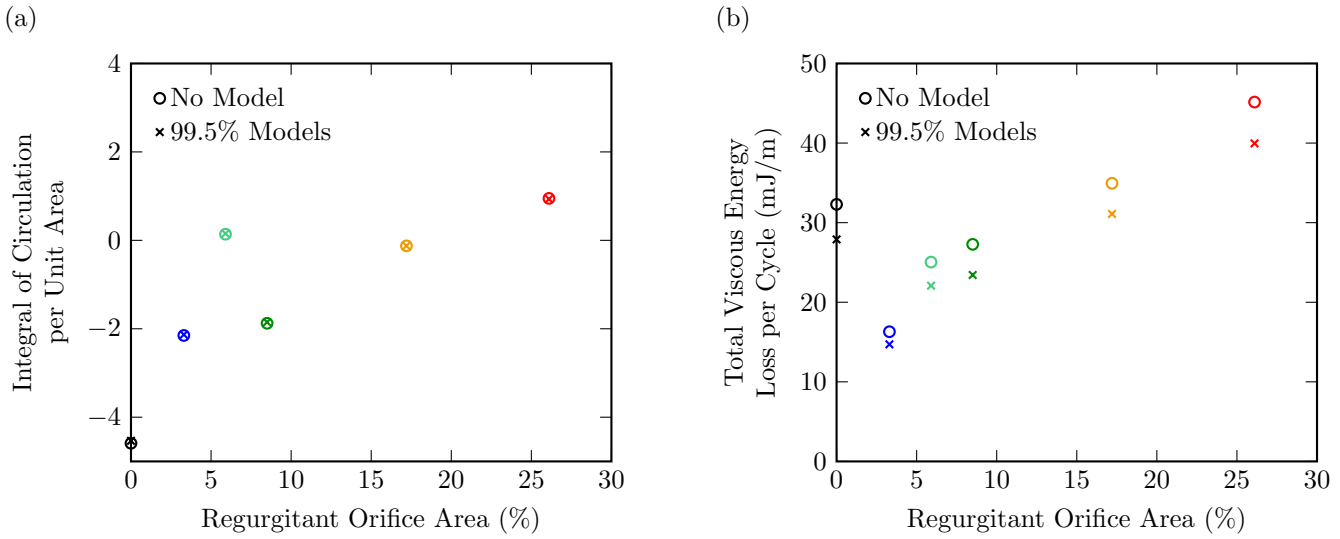

Figure S2: (a) The integral of the circulation per unit area (Eq. (2) in the main text) and (b) the total viscous energy loss per cardiac cycle (Eq. (3) in the main text) are shown for the intraventricular flows (open circles) and the corresponding reduced-order models (crosses).

cycles the errors fall below 5% for all models. The same is effectively true up to 4 cardiac cycles, with the exception of the healthy and moderate-2 cases (and briefly for the moderate-1 case in the third cycle). In order to evaluate the error in the velocity fields associated with flow reconstruction using the reduced-order models, we use the normalized

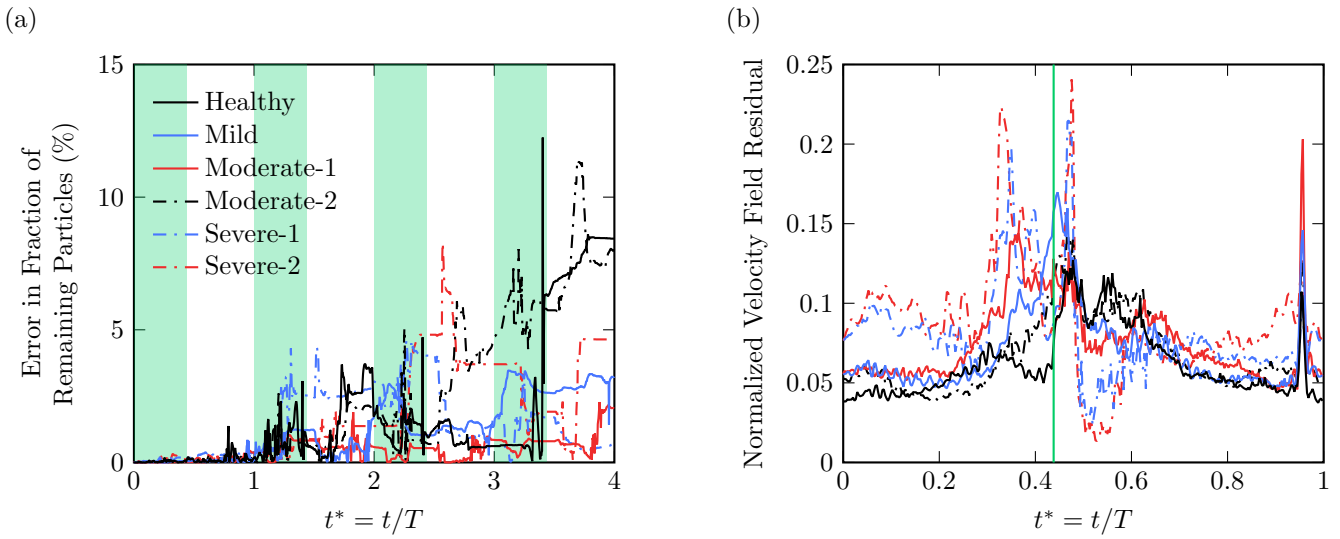

Figure S3: The error associated with the provided models is here evaluated for all cases with regard to two additional metrics. In (a), we show the error in the fraction of particles remaining in the left ventricle after releasing $\sim 800,000$ virtual particles at the start of the ejection phase ($t^* = 0$) for 4 cardiac cycles. The regions shaded in green correspond to ejection phases. In (b), we show the normalized residual of the velocity field at each time step, given by Eq. (S2). The vertical green line marks the beginning of the filling phase.

residual

$$e_k = \frac{\|\widetilde{\mathbf{x}}_k - \mathbf{x}_k\|}{\|\mathbf{x}_k\|}, \tag{S2}$$

which is similar to that used in computational fluid dynamics, where $\mathbf{x}_k$ denotes the true velocity state vector for snapshot k, $\widetilde{\mathbf{x}}_k$ denotes that produced by the model and $\|\cdot\|$ denotes the $\mathcal{L}_2$-norm. The normalized residual is shown in Fig. S3b for all cases using the 99.5% models. While the metric of Eq. (S2) is rather strict, the normalized residuals tend to fall below 0.1 for the majority of the cycle, including during the filling phase where the regurgitation occurs. The largest errors occur close to the end of the ejection phase, which is marked by a vertical green line.

References

Sirovich, L. (1987). Turbulence and the dynamics of coherent structures – Part I: Coherent structures. *Quarterly of Applied Mathematics*, *45*(3), 561-571.

Tu, J. H., Rowley, C. W., Luchtenburg, D. M., Brunton, S. L., & Kutz, J. N. (2014). On dynamic mode decomposition: Theory and application. *Journal of Computational Dynamics*, *1*(2), 391-421.