University of Pittsburgh

ME 2061

Reduced Order Modeling for Engineers

Project

Reduced order modelling of incompressible flow over a two-dimensional cylinder

Author:

Shamsulhaq Basir

December 06, 2019

Contents

1 Introduction

Reduced-order modeling is the technique of reducing the computational complexity of the mathematical models by reducing their associated state-space dimensions. It is the optimal choice where numerical simulation is either infeasible or prohibitively expensive. In the field of fluid dynamics, the Navier-Stokes equations can accurately describe any given fluid system. However, due to the non-linearity and the complexity of these equations, their direct application to engineering design is rather difficult. Therefore, data from different fluid configurations are sampled and used to develop models that describe their interactions. The objective of this project is to build a model that describes the flow over a two-dimensional cylinder.

2 Mathematical formulation

The fluid system considered in this project is governed by the incompressible Navier-Stokes equations.

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = -\frac{\partial p}{\partial x} + \frac{1}{Re}(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}), \tag{1}$$

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} = -\frac{\partial p}{\partial y} + \frac{1}{Re}(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}), \tag{2}$$

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0. \tag{3}$$

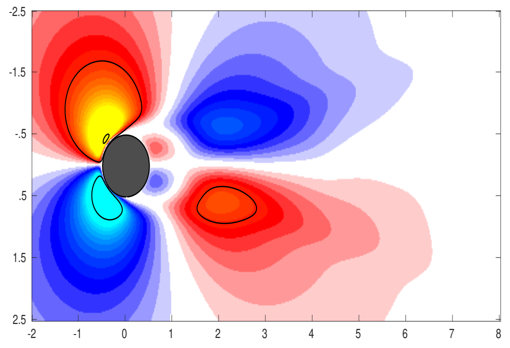
where $\boldsymbol{u}(x, y, t) = u(x, y, t)\boldsymbol{i} + v(x, y, t)\boldsymbol{j}$ is the velocity field, $p(x, y, t)$ is the pressure field and $Re = \rho V_\infty D/\mu$ is the Reynolds number, where ρ is the density , V_∞ is the free-stream velocity and D is the cylinder diameter.Equations (1) and (2) are conservation of x-momentum and y-momentum, respectively, and equation (3) is the conservation of mass. The above equations are in nondimensional form and all the quantities that appear in equation (1)-(3) are non-dimensional. Spectral element method is used to solve the above equations at Re = 100.

3 Proper Orthogonal Decomposition

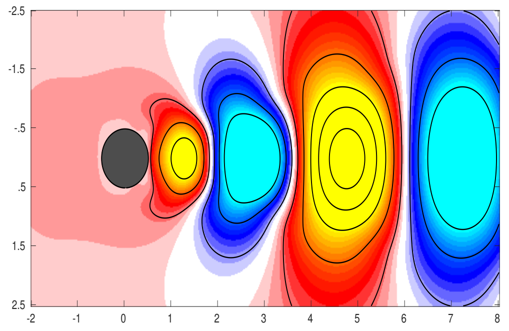
In POD reduced-order modeling we represent the full-dimensional solution by a set of basis vectors in a reduced dimension sub-space with the help of Singular Value Decomposition (SVD).

3.1 POD modes

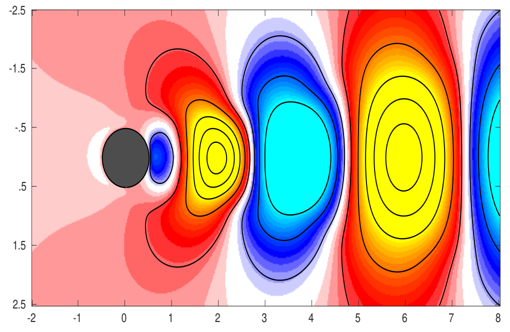
POD modes or eigenfunctions are obtained by solving an optimization problem. They are representative of the flow organization. The v components of the first 6 dominant modes are shown in figure (1).



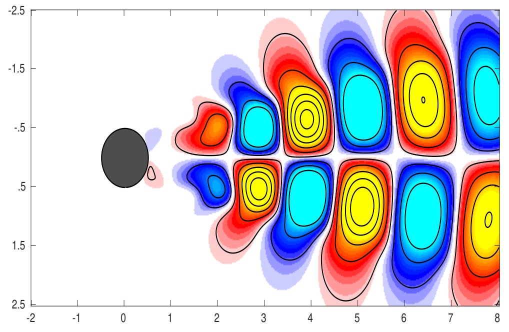
(a) mode 1



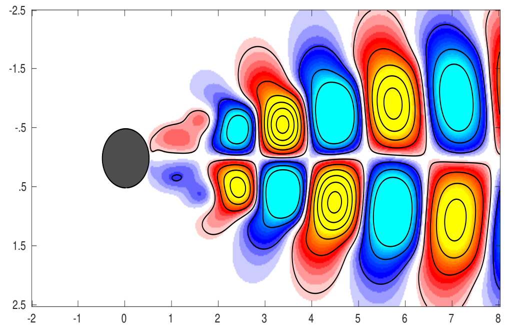
(b) mode 2



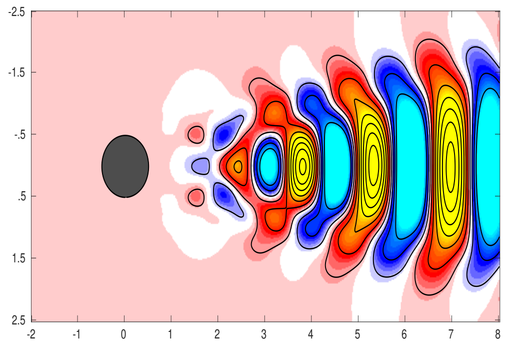
(c) mode 3



(d) mode 4



(e) mode 5



(f) mode 6

Figure 1: The first 6 dominant v component of the POD modes in (a) -(f)

3.2 Eigenvalues of the covariance matrix

Due to the size of the data matrix, the direct application of Singular Value Decomposition (SVD) on it would be computationally expensive. Therefore, a covariance matrix is obtained that would provide the singular values and the basis.
$A = \begin{bmatrix} U \\ V \end{bmatrix}$, $A = U\Sigma V^T$, $\tilde{C} = A^T A$, $\tilde{C}V = V\Sigma^2$. Where A is the data matrix and$\tilde{C}$ is the covariance matrix. The columns of U and V are called the left-singular vectors and right-singular vectors of A respectively. Figure (2) shows the first 20 eigenvalues of .

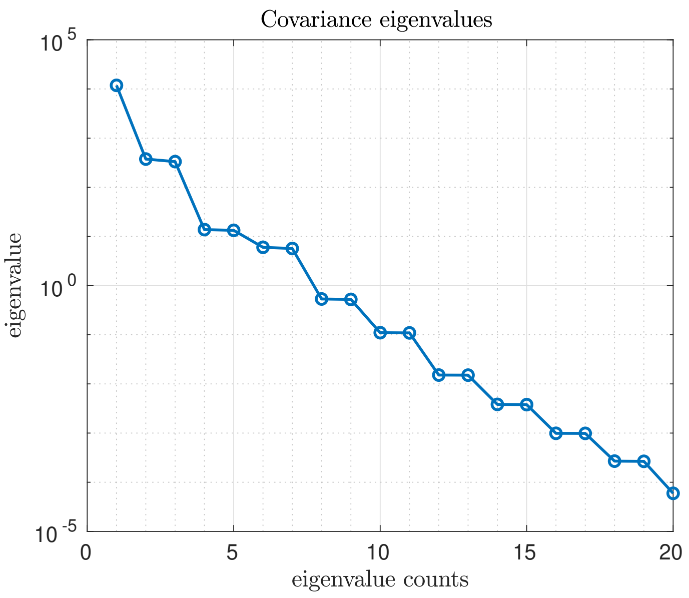
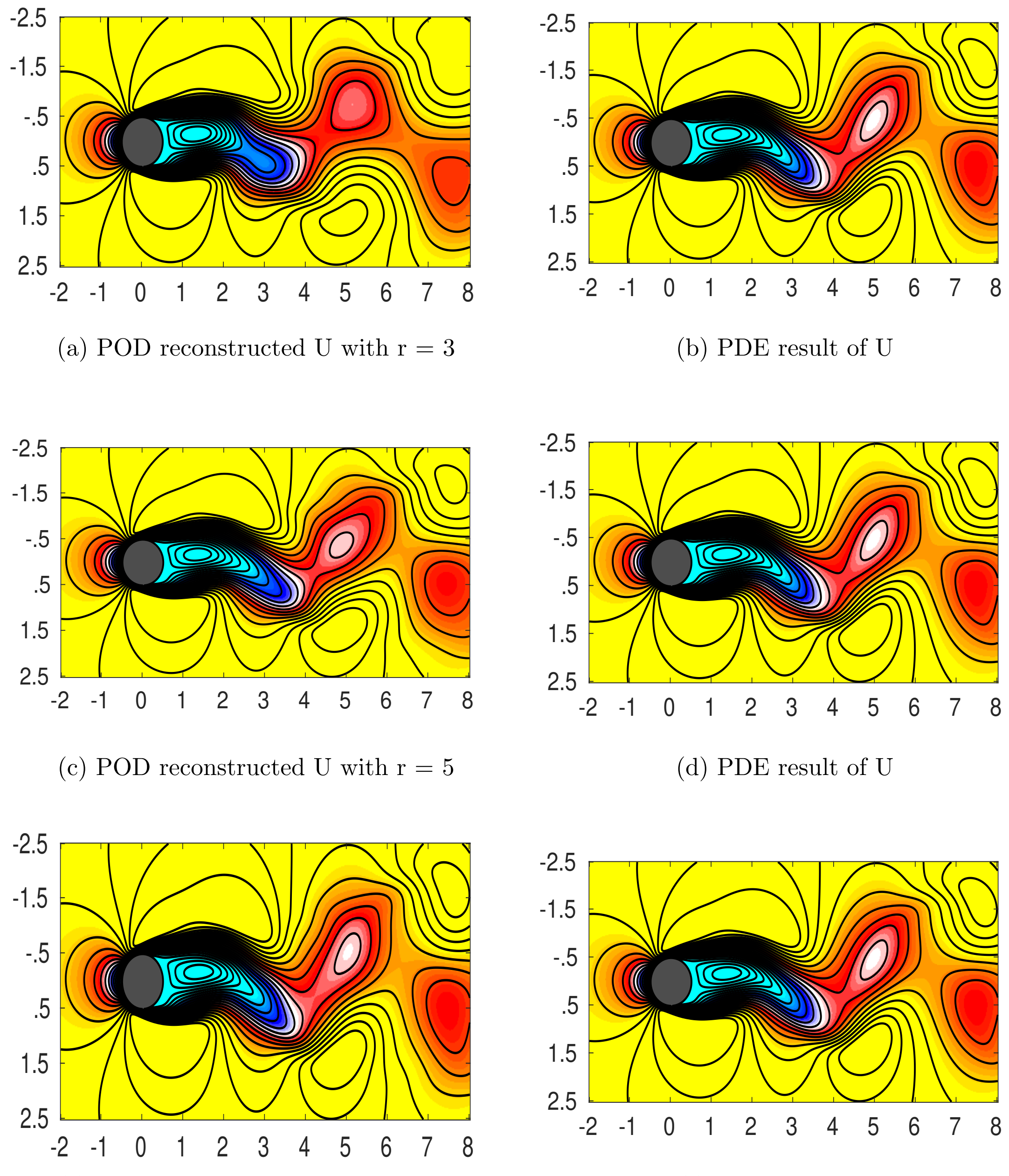


Figure 2: The first 20 eigenvalues of the covariance matrix

3.3 Reconstruction of the Navier-Stokes Solution

The solution of the Navier-Stokes solution at $t = 20$ is reconstructed with POD modes. we consider $\boldsymbol{u}(x, y, t) = \sum_{i=1}^{r} y_i^{POD}(t)\Phi_i(x, y)$ and $y_i^{POD}(t)$ is obtained for t = 20.
Figure (3) shows the comparison between the u components of the velocity obtained by the direct solution of the NS and the reduced order model with 3, 5, and 7 modes at t = 20.



(a) POD reconstructed U with r = 3

(b) PDE result of U

(c) POD reconstructed U with r = 5

(d) PDE result of U

(e) POD reconstructed U with r = 7

(f) PDE result of U

Figure 3: u component of the velocity at t = 20

3.4 Galerkin Projection of Navier-Stokes equations

Given an orthogonal basis from POD, it is possible to obtain a ROM of a dynamical system through Galerkin projection. The state vector $\boldsymbol{u}$ is approximated by a finite sum of POD modes, and this expansion is substituted directly into the dynamical system. The resulting dynamical system typically has a much smaller dimension r compared with the dimension n of the state-space. $\boldsymbol{u}(x, y, t)$ represents a continuous vector field of velocity components of the fluid in space and time:

$$\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} = -\nabla p + \frac{1}{Re}\nabla^2 \boldsymbol{u}, \tag{4}$$

$$\nabla \cdot \boldsymbol{u} = 0. \tag{5}$$

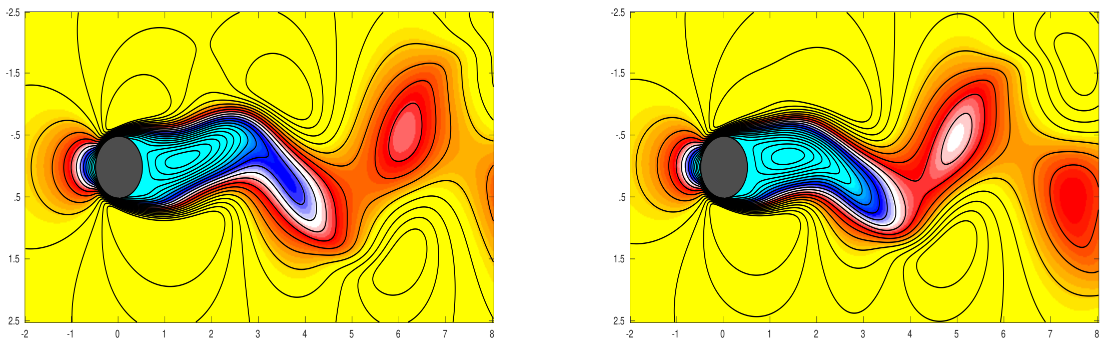
Let $\boldsymbol{u}(x, y, t) = \sum_{i=1}^{r} y_i^{POD}(t)\Phi_i(x, y)$. It is possible to write equation (4) and (5) as a dynamical system. Equation (6) is the ROM model obtained after substituting $\boldsymbol{u(x, y, t)}$ into equation (4) and (5).

$$\dot{y}_k = \boldsymbol{L_{ik}} y_i + \boldsymbol{N_{ijk}} y_i y_j, \tag{6}$$

where $\boldsymbol{L_{ik}} = \langle \boldsymbol{\Phi}_k, \Delta\boldsymbol{\Phi}_i \rangle$, $\boldsymbol{N_{ijk}} = \langle \boldsymbol{\Phi}_k, (\boldsymbol{\Phi_i} \cdot \nabla)\boldsymbol{\Phi}_j \rangle$ are linear and bi-linear operators on mode coefficients, and y_i , y_j are the POD time coefficients. Inner product of the pressure gradient with the basis are zero after taking the integration by part since basis are divergence-free and pressure are constant at the far boundaries.

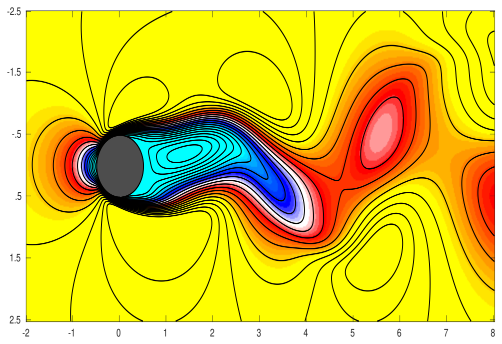
3.5 Reconstruction of the solution with ROM

For the reconstruction of the Navier-Stokes solution, the ordinary differential equation (6) is solved with an initial condition. *parDiff* is defined to compute the partial derivatives of $\boldsymbol{\Phi(x)}$ and *ROM_Coefficient* is defined to compute the linear and the non-linear operators in equation (6). *ode*45 is used to solve equation (6). Due to a subtle bug in the code, time coefficients were decaying which is not what expected. Time coefficients must capture the oscillation of the solution. Therefore, Figure (5)-(6) shows the U and V components of the velocity the from the direct solution of the PDE.

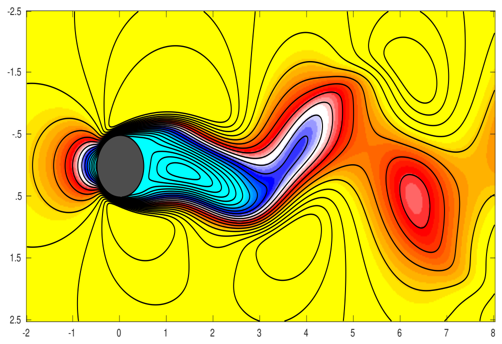


(a) t = 10 (b) t = 20

Figure 4: U component of the velocity using the direct solution of the NS

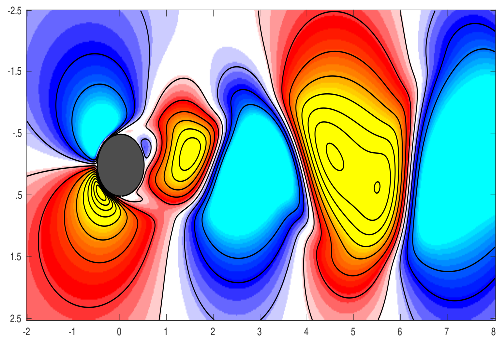


(a) t = 30

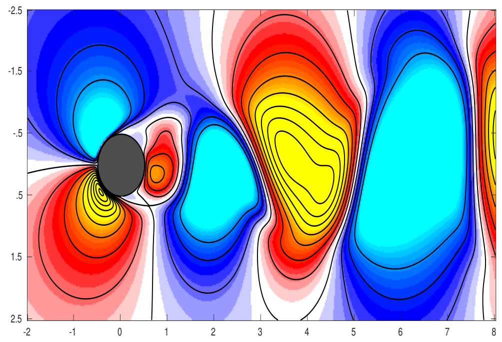


(b) t = 40

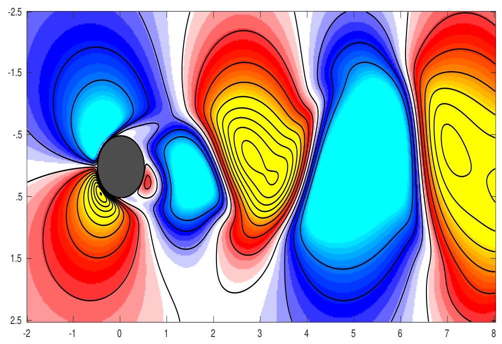
Figure 5: U component of the velocity using the direct solution of the NS



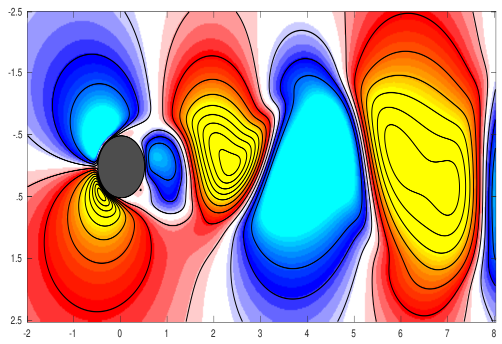
(a) t = 10



(b) t = 20



(c) t = 30



(d) t = 40

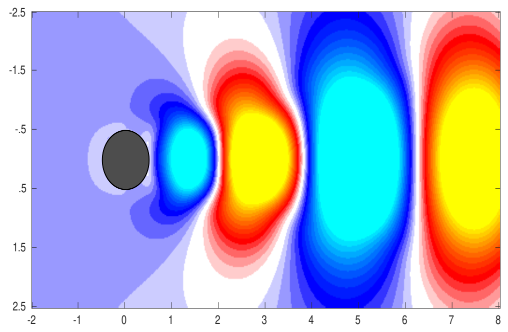
Figure 6: V component of the velocity using the direct solution of the NS

4 Dynamic Mode Decomposition

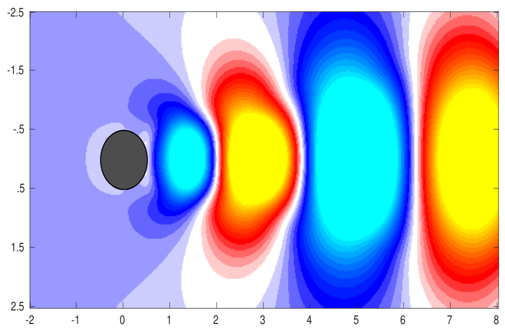
The DMD method is a decomposition technique that provides a spatiotemporal decomposition of the collected data into a set of dynamic basis that describes the flow configuration associated with a given temporal frequency, either with a growth rate or decay rate.

4.1 Computation of modes

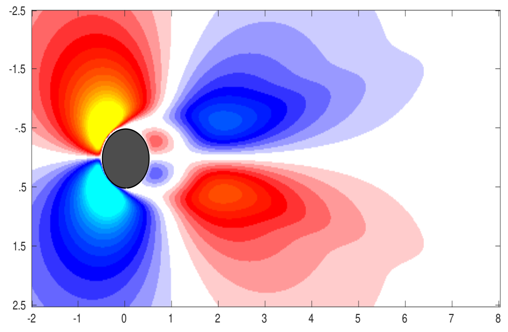
Using the DMD algorithm, we have found the first dominant 8 modes as shown in figure 4(a)-(d) and figure 5(a)-(d)



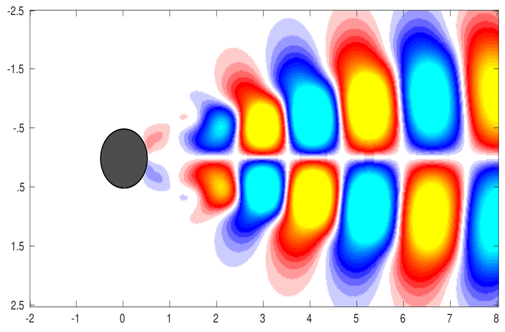
(a) mode 1



(b) mode 2

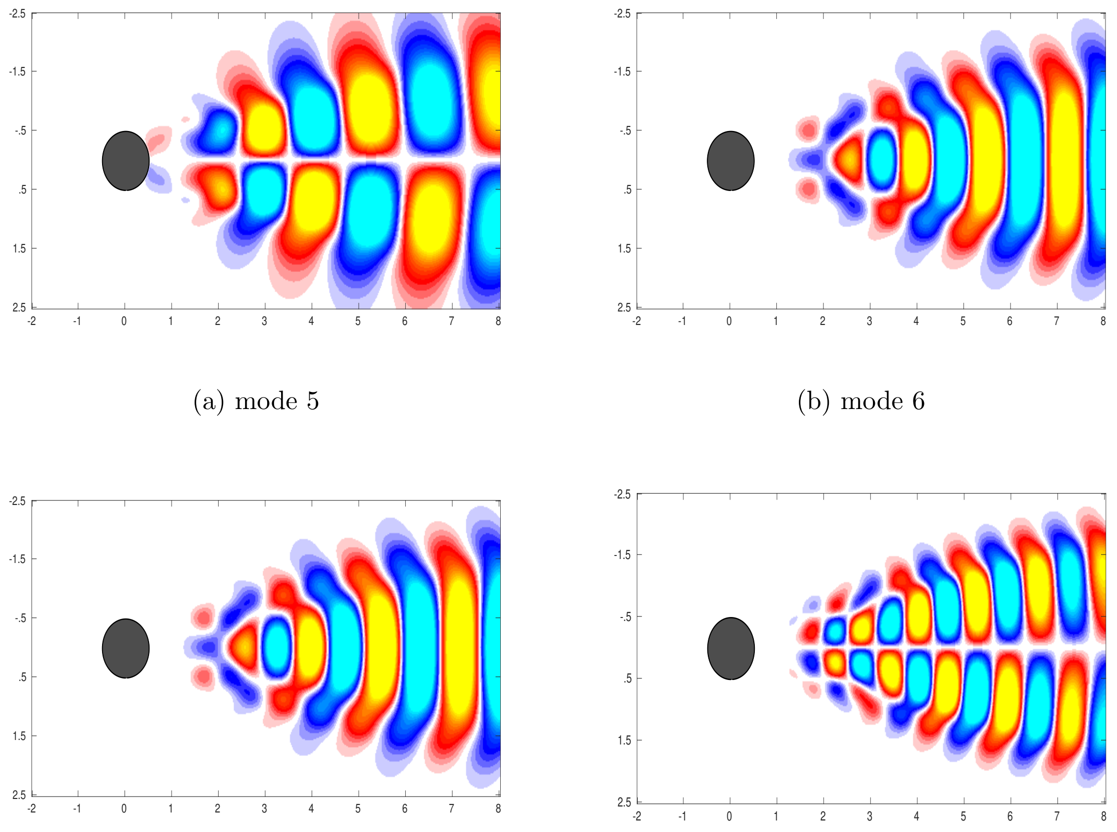


(c) mode 3



(d) mode 4

Figure 7: The first 4 of the 8 dominant DMD modes (a) -(d)



(a) mode 5 (b) mode 6

(c) mode 7 (d) mode 8

Figure 8: The last 4 of the 8 dominant DMD modes (a) -(d)

4.2 Computation of the eigenvalues

eigenvalues associated with the DMD modes are shown in a complex plane in figure (9).

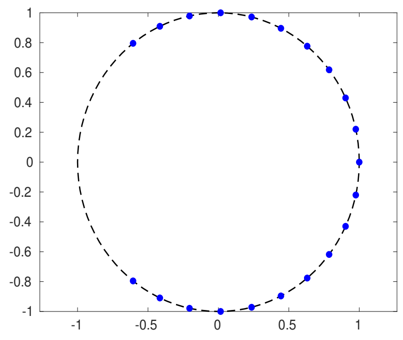
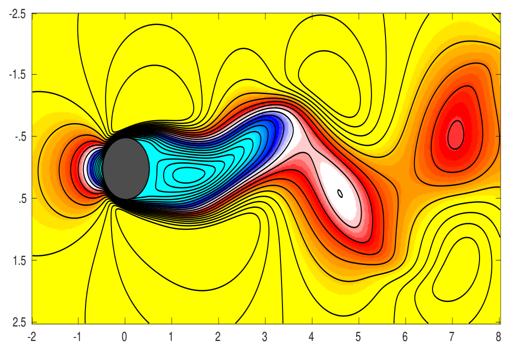
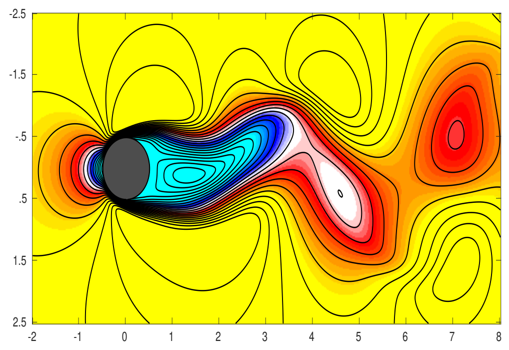


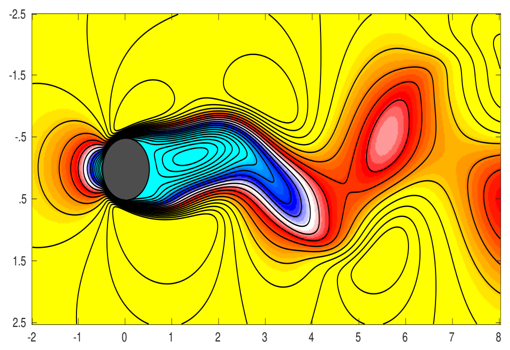
Figure 9: eigenvalues DMD

4.3 Reconstruction and forecasting

Finally, DMD modes are used to reconstruct the direct solution of the NS at different time levels. Figure(10) shows the comparison between he horizontal component of the velocity obtained from the direct solution of equation(1)-(3) and the DMD method for t = 45 and 49.


(a) DMD at t = 45


(b) PDE at t = 45


(c) DMD at t = 49

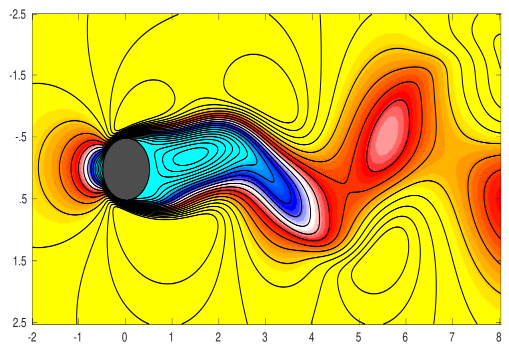

(d) PDE at t = 49

Figure 10: Horizontal velocity component using DMD method (a), (c) and the direct solution of the NS equations (b),(d).

Figure(10) compares DMD results with the direct solution of the NS equations. It can be seen that DMD method captures all the flow structures with great accuracy.

```
%% Project
%% Shamsulhaq Basir

clear all, close all, clc
LW = 'linewidth'; FS = 'fontsize'; IN = 'interpret'; LT = 'latex';
FW = 'fontweight';B  = 'bold';
set(0,'defaulttextinterpreter','latex')
load CYLINDER.mat


%% ------------- Animating the u velocity ------------
C1 = linspace(-.2,1.2,20);
C2 = linspace(-5,5,20);
figure(1)
for i=1:1:250
    plotCylinder(reshape(U(:,i),ny,nx),C1);
    colorbar
    title('U Velocity')
    hold off
    drawnow
end
%% ------------- Animating the y-gradeint of the u velocity ------------


figure(2)
for i=1:1:250
    [Ux,Uy] = gradient(reshape(U(:,i),ny,nx),dx,dy);
    [Vx,Vy] = gradient(reshape(V(:,i),ny,nx),dx,dy);
    VORT = Uy-Vx;

    plotCylinder(reshape(VORT,ny,nx),C2); hold on
    title('Vorticity')
    set(gca,'FontSize',15)
    drawnow
    hold off
end


%%
A = [U;V];
h = dx*dy;
%% covariance matrix
C = A'*A*h;
%% POD modes
[Vc Sc VcT] = svd(C);
Sm = sqrt(Sc);
Phi = A*Vc*inv(Sm);

%% : V components of the first 6 dominant modes
Urow = size(U,1);
C1 = linspace(-.2,1.2,20);
C2 = linspace(-5,5,20);

for i=1:1:6
    %     subplot(3,2,i)
    figure
    plotCylinder(reshape(Phi(Urow+1:end,i),ny,nx),C1);
    %     title(['V component of POD mode [',num2str(i,'%1d'),']']);
    set(gca,FS,12)
end

%% : 25 eigenvalues of the covariance matrix
figure()
s = diag(Sc);
s = s(1:20);
semilogy(s,'-o',LW,1.6)
title("Covariance eigenvalues ");
xlabel("eigenvalue counts",FW,B,FS,14);
ylabel("eigenvalue",FW,B,FS,14);
grid minor
grid on
set(gca,'fontsize',12);
%% cumulative
figure()
set(gca,FS,12)
plot(cumsum(s)/sum(s)*100,'-ob',LW,1.6)
pctg = cumsum(s)/sum(s)*100;

for i=1:size(pctg)
    if(pctg(i) >= 99.99)
        fprintf("Mode # %3d \n",i);
        break;
    end
end

%% 2 :
dt = 0.2;
t = 20;
nsnap = t/(dt) +1;

for r =[3 5 7]
    close all;
    y = Phi(:,1:r)'*A(:,nsnap)*h;
    u = Phi(1:Urow,1:r)*y;
    figure()
    plotCylinder(reshape(u(1:Urow,1),ny,nx),C1);
    set(gca,'fontsize',14);
    figure()
    plotCylinder(reshape(U(:,nsnap),ny,nx),C1);
    set(gca,'fontsize',14);
end

%% - ROM Model
Tf = 50;
dt = 0.2;
t = 0:dt:Tf;
for r = [6]
    Ub   = Phi(:,1:r); % <Ub,Ub> = Ub(:,1:r)'*Ub(:,1:r)*dx*dy = I
    y0   = A(:,1)'*Ub.*(dx*dy);y0  = y0';
    [N,D]= ROM_Coefficient(Phi(:,1:r),nx,ny,dx,dy);

    [t,y] = ode45(@(t,y)dydt(y,N,D,r),t,y0);
    rom = Ub*y';
end

for i=1:1:250
    % U
    plotCylinder(reshape(rom(1:nx*ny,i),ny,nx),C1);
    colorbar
    title('$U_{rom}$ Velocity')
    hold off
    drawnow
end



%% DMD :  we use up to t = 40 to develop a DMD model
nsnap = 40/dt+1;
X1 = A(:,1:nsnap-1);
X2 = A(:,2:nsnap);
%% SVD and rank -r truncation
r = 21;
[U_dmd S_dmd V_dmd] = svd(X1,'econ');
Ur = U_dmd(:,1:r);
Sr = S_dmd(1:r,1:r);
Vr = V_dmd(:,1:r);

%% Build Atilde and DMD Modes
Atilde = Ur'*X2*Vr/Sr;
[W, eigs] = eig(Atilde);
Phi = X2*Vr/Sr*W; % DMD Modes

%% DMD Spectra
lambda = diag(eigs);                          % discrete-time DMD eigenvalues
omega = log(lambda)/dt;                       % the continuous-time DMD eigenvalues
[omega,I]=sort(omega,'descend','ComparisonMethod','real'); % sort the omegas
Phi= Phi(:,I(1:r));                           % sort the DMD Modes accordingly

%% 8 DMD Modes : 7 modes are requested but for the sake of subplots we plot 8
close all;

for i=1:1:8
    figure();
    phi_re = real(Phi(:,i));
    plotCylinder(reshape(phi_re(Urow+1:end),ny,nx),C1);
    set(gca,'fontsize',12);
end


%% EigenValues
figure
theta = (0:1:100)*2*pi/100;
plot(cos(theta),sin(theta),'k--',LW,1.2) % plot unit circle
hold on
h1 = scatter(real(lambda),imag(lambda),'ob','filled');
set(gca,FS,12)
axis equal
```

```
%% Compute DMD Solution
x1 = X1(:,1);
b = Phi\x1;
t = 0:dt:50;
time_dynamics = zeros(r,length(t));
for iter = 1:length(time_dynamics)
    time_dynamics (:,iter) = (b.*exp(omega*t(iter)));
end
X_dmd = Phi * time_dynamics;


%% DMD forcast
close all
t = [45 49];
for i = 1:2
    figure()
    phi_r = real(X_dmd(:,t(i)/dt+1));
    plotCylinder(reshape(phi_r(1:Urow),ny,nx),C1);
    set(gca,'fontsize',12);
    %     title(['POD forcast at Time = ',num2str(t(i)),' (s)'])
    figure()
    plotCylinder(reshape(U(1:Urow,t(i)/dt+1),ny,nx),C1);
    %     title(['PDE at Time = ',num2str(t(i)),' (s)'])
    set(gca,'fontsize',12);

end

function [N,D] = ROM_Coefficient(Phi,nx,ny,dx,dy)
% - function for computing the Linear and non-linear coefficeint of the
% - ROM equation
Ub   = Phi;
r = size(Phi,2);
[Ubx, Uby ] = parDiff(Ub,nx,ny,dx,dy);
D = (-Ubx'*Ubx.*dx - Uby'*Uby.*dy)./100;

u = Phi(1:nx*ny,1:r);
v = Phi(nx*ny+1:end,1:r);

ux = Ubx(1:nx*ny,1:r);
uy = Uby(1:nx*ny,1:r);

vx = Ubx(nx*ny+1:end,1:r);
vy = Uby(nx*ny+1:end,1:r);

N = cell(r,1);
for k = 1:r
    for i = 1:r
        for j = 1:r
            N{k}(j,i) = -sum(sum(dx.*u(:,k).*(u(:,i).*ux(:,j)+v(:,i).*uy(:,j))
+...
                dy.*v(:,k).*(u(:,i).*vx(:,j)+v(:,i).*vy(:,j))));
        end
    end
end
```

```
end


function [Ubx, Uby ]= parDiff(Ub,nx,ny,dx,dy)
% -- function for taking partial x and partial y derivatives

nt  = size(Ub,2);
np = nx*ny;
% allocate arrays
Ubx = 0.*Ub;
Uby = 0.*Ub;

for i = 1:nt
    % separate the u and v components
    Ub_u = Ub(1:np,i) ;
    Ub_v = Ub(np+1:end,i);
    % take the derivative
    [Ub_ux,Ub_uy]= gradient(reshape(Ub_u,ny,nx),dx,dy);

    Ubux = reshape(Ub_ux,ny*nx,1);
    Ubuy = reshape(Ub_uy,ny*nx,1);

    [Ub_vx,Ub_vy]= gradient(reshape(Ub_v,ny,nx),dx,dy);
    Ubvx = reshape(Ub_vx,ny*nx,1);
    Ubvy = reshape(Ub_vy,ny*nx,1);

    Ubx(1:np,i)      = Ubux;
    Ubx(np+1:end,i)  = Ubvx;

    Uby(1:np,i)      = Ubuy ;
    Uby(np+1:end,i)  = Ubvy;
end


end


function dy = dydt(y,Q,D,r)
% -- ODE

dy = zeros(r,1);

for i = 1:r
    dy(i) = y'*Q{i}*y +D(i,:)*y;
end

end
```