Manual para implementar DMD a estructuras de flujo

Juan Sebastián Hincapié

Pre-procesamiento: matriz de Snapshots

Antes de empezar con la programación pura y dura de la descomposición modal, es esencial que el programador responda a la pregunta ¿Qué propiedad del fluido o del fenómeno se desea descomponer en modos? Ya que se puede descomponer una gran variedad de propiedades tales como la temperatura, la presión, las componenetes de la velocidad, la viscosidad del fluido, vorticidad, etc.

Teniendo claro con qué propiedad se va a trabajar, se procede a organizar los datos obtenidos de la simulación CFD. El campo de dicha propiedad a lo largo del tiempo se organiza en tantas matrices $\mathbf{x}(r, t_k)$ como paso temporales se quieran incluir en la descomposición modal. Por lo tanto, cada matriz agrupa el campo de la propiedad en un paso temporal $t_k$específico de la simulación. Además, al interior de la matriz $\mathbf{x}(r, t_k)$ el valor nodal de la propiedad se ordena según la posición del nodo al que corresponde, donde $r_{i,j}$ representa la vorticidad en el nodo (i, j).

$$\mathbf{x}(\mathbf{r}, t_k) = \begin{bmatrix} x(r_{1,1}, t_k) & x(r_{1,2}, t_k) & \cdots & x(r_{1,p}, t_k) \\ x(r_{2,1}, t_k) & x(r_{2,2}, t_k) & \cdots & x(r_{2,p}, t_k) \\ \vdots & \vdots & \ddots & \vdots \\ x(r_{q,1}, t_k) & x(r_{q,2}, t_k) & \cdots & x(r_{q,p}, t_k) \end{bmatrix}$$

Cada una de las matrices $\mathbf{x}(r, t_k)$ se transforma en un vector "alto" y "delgado" de la siguiente manera:

$$\mathbf{x}_k = \begin{bmatrix} x(r_{1,1}, t_k) \\ x(r_{1,2}, t_k) \\ \vdots \\ x(r_{2,1}, t_k) \\ \vdots \\ x(r_{q,p}, t_k) \end{bmatrix}$$

Esta transformación se puede llevar a cabo usando el comando **reshape** de Matlab. Por ejemplo:

```
Data = [1 2 3;
        4 5 6;
        7 8 9];

i = 1; %Número de columnas
j = 9; %Número de filas
disp('Matriz Data con la nueva forma alta y delgada:')
```

```
Matriz Data con la nueva forma alta y delgada:
```

```
snap = reshape(Data, j, i)
```

```
snap = 9×1
     1
     4
     7
     2
     5
     8
     3
     6
     9
```

Cada uno de los vectores $\mathbf{x}_k$ se agrupa en lo que se conoce como la matriz de **snapshots** $\mathbf{X}$:

$$\mathbf{X} = \begin{bmatrix} | & | & & | \\ \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_m \\ | & | & & | \end{bmatrix}$$

El libro Dynamic Mode Decomposition publicado por J. Nathan Kutz, Steven L. Brunton, Bingini W. Brunton y Joshua L. Proctor brinda el campo de velocidad, vorticidad y de presión para la simulación del flujo alrededor de un cilindro con Reynold 100. Estos datos ya están agrupados en matrices snapshots, así que no es necesario hacer el procedimiento antes mencionado. Los detalles de la simulación se pueden encontrar en el capítulo 2 del libro.

Se debe descargar el archivo **DATA.zip** desde la página http://dmdbook.com/. Al descomprimirla se generan una serie de carpetas. Dentro de la carpeta llamada **Fluids** se encuentran archivos de extensión **.mat**. Para este estudio, únicamente se trabjará con **CYLINDER_ALL.mat**. Se puede hacer una copia de este archivo en una carpeta aparte.

Empezamos extrayendo la información del campo de vorticidad del archivo **CYLINDER_ALL.mat**. Esta información se puede llamar con el nombre de la llave, que para este caso es "VORTALL". Es buena prática trabajar/programar con la copia de los datos y no con la fuente misma.

```
load("CYLINDER_ALL.mat", "VORTALL")
Avort = VORTALL(:,1:150);
```

Factorización SVD

Luego se usa el método SVD (Singular Value Decomposition, por sus siglas en inglés) que se encarga de factorizar la matriz de snapshots **Avort** de dimensiones 89351x150 en tres matrices:

- **U:** matriz ortonormal. Cuando se usar Full SVD tiene dimensiones 89351x89351; pero cuando se usa Reduced SVD tiene dimensiones 89351x150.
- Σ**:** matriz de valores singular. Para Full SVD tiene dimensiones 89351x150; pero cuando se usar Reduced SVD tiene dimensiones 150x150.
- **V:** matriz ortonormal. Esta matriz es de dimensión 150x150 tanto en Full como en Reduced SVD.

La factorización es de la sigueintes forma:

$$\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^{\mathrm{T}}$$

Nota: ¿Hay alguna ventaja en usar un método u otro? Sí y enorme, en términos de memoria computacional requerida para el cáculo de SVD, pues el método Reduced usa matrices de menor tamañao que en el método Full. Por otro lado, se puede llegar a pensar que que el método RSVD es menos preciso que FSVD, dado que usa menos información; pero no es así. Ambos están al mismo nivel de precisión. Un ejemplo a continuación para ilustrar los anterior:

```
%Ejemplo
disp('La matriz a factorizar es:')
```

```
La matriz a factorizar es:
```

```
A = [3,1;1,2;6,5]
```

```
A = 3×2
     3     1
     1     2
     6     5
```

```
disp('Factorización completa:')
```

```
Factorización completa:
```

```
[U, S, V] = svd(A)
```

```
U = 3×3
   -0.3454    0.7514   -0.5623
   -0.2360   -0.6494   -0.7229
   -0.9083   -0.1170    0.4016
S = 3×2
    8.5967         0
         0    1.4482
         0         0
V = 2×2
   -0.7819    0.6234
   -0.6234   -0.7819
```

```
disp('Factorización reducida. Notar la diferencia de tamaño entre U y S')
```

```
Factorización reducida. Notar la diferencia de tamaño entre U y S
```

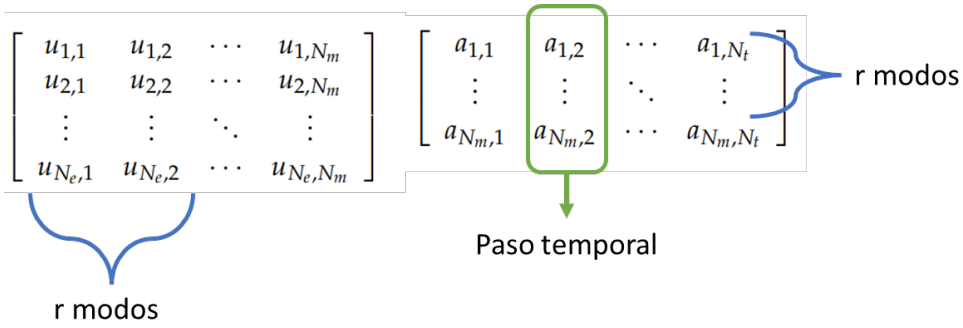
```
[U, S, V] = svd(A, "econ")
```

```
U = 3×2
   -0.3454    0.7514
   -0.2360   -0.6494
   -0.9083   -0.1170
S = 2×2
    8.5967         0
         0    1.4482
V = 2×2
   -0.7819    0.6234
   -0.6234   -0.7819
```

Continuando con el procedimiento, aplicamos la factorización **SVD** a la matriz *Avort* utilizando el comando *svd* con la opción *'econ'* para calcular la factorizar con la forma reducida. Además, se calcula el productor entre las matrices *S* y *V*:

```
[U, S, V] = svd(Avort, "econ");
alpha = S*V';
```

Esta es la base para del método que sigue a continuación: el POD (Descomposición Ortogonal Adecuada, por sus siglas en inglés). Este método numérico permite "descomponer" el flujo en estructuras coeherentes o modos a partir de los datos de simulación. Lo que se debe hace es truncar la factorización SVD, es decir, usar unos cuantos valores singulares, no todos, para representar el flujo en un tiempo específico:



La primera matriz es *U* y la segunda matriz, llamada matriz de coeficientes, es *alpha* o el producto entre *S* y *V*. El número de columnas de *U* es igual al número de modos que tiene el sistema, y el número de filas es igual al número de valores nodales. Por otro lado, en *alpha*, el número de columnas es igual al número de pasos temporales y el número filas es igual al número de modos que tiene el sistema. Por tanto, para representar el flujo con **r** modos, se debe selecionar **r columnas** de *U* y **r filas** de *alpha*. Adicionalmente, se seleciona la columna de *alpha* que representa el paso temporal deseado.

Contenidos energético

En este punto puede surgir naturalmente una pregunta ¿Cuántos modos utililizar para la representación de la estructura? o coloquilamente ¿Cuál sería el número mágico de modos? Y la respuesta es simple: depende de la aplicación. Sí, depende de qué tanta información se puede perder, y eso lo define la aplicación. Para tener más cirterios de selección se puede hacer dos gráficas complementarias entre sí:

- La primera representa la energía (o información) contenida en cada uno de los modos del sistema. El valor de *n* representa el número de modos totales

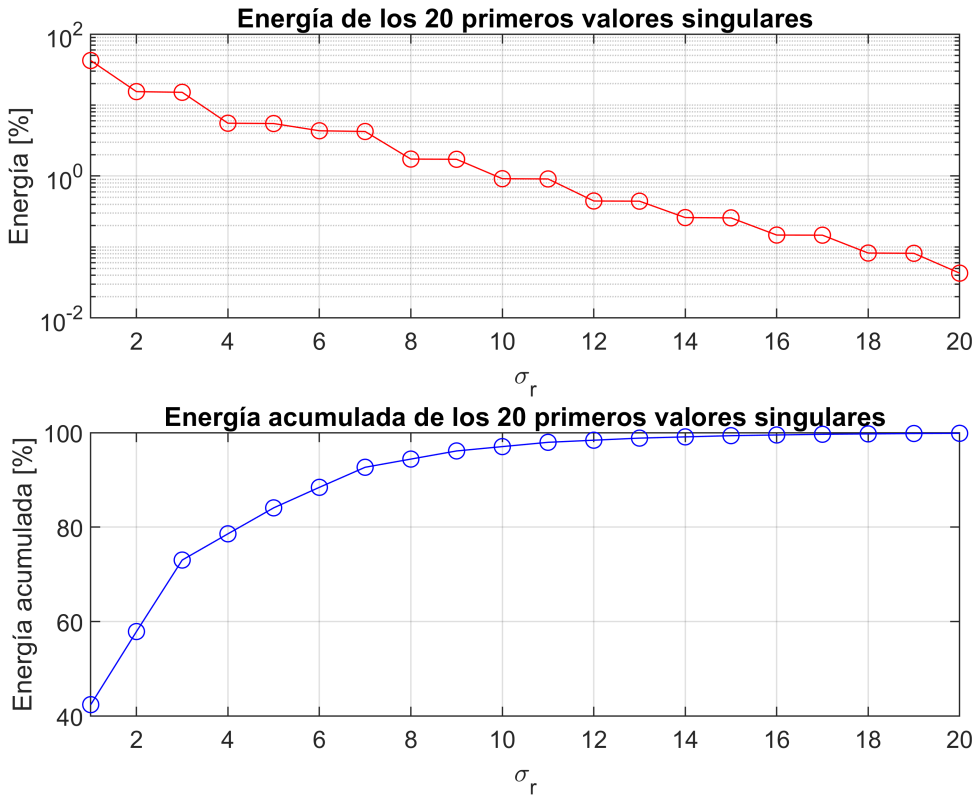
$$E_i = \frac{\sigma_i}{\sum_{i=1}^{n} \sigma_i}$$

- La segunda grafica representa la energía acumualda a medida que se añaden más modos. El valor de *r* representa el número de modos usado en el truncamiento del SVD::

$$E_i = \frac{\sum_{i=1}^{r} \sigma_i}{\sum_{i=1}^{n} \sigma_i}$$

A continuación se presenta le código empleado para construir las dos gráficas. Se usaron 20 modos en el truncamiento.

```
sDiagonal = diag(S);           %Construyo un vector con la diagonal de S
sSuma = sum(sDiagonal);        %Sumo el vector y guardo el valor
sSumatoria = cumsum(sDiagonal);%Suma acumulativa

singularEnerg = sDiagonal/sSuma *100;       %Energía de cada modo en porcentaje
singularEnergAcum = sSumatoria/sSuma *100;  %Energía en porcentaje
r = 20;                                     %Número de valores singulares a usar


fig1 = figure;
subplot(2,1,1)
semilogy(singularEnerg(1:r), "-or")
title(sprintf("Energía de los %d primeros valores singulares",r))
xlabel("\sigma_r")
ylabel("Energía [%]")
xlim([1 r])
grid on

subplot(2,1,2)
plot(singularEnergAcum(1:r), "-ob")
title(sprintf("Energía acumulada de los %d primeros valores singulares",r))
xlabel("\sigma_r")
ylabel("Energía acumulada [%]")
xlim([1 r])
grid on
hold off
```



Podemos observar de la primera gráfica que el contenido energético de los modos va decreciendo; esto se debe a que en el método numérico SVD, los valores singulares están ordenados de mayor a menor en la matriz *S*. A medida que vamos utilizando más modos, mayor información del sistema estaremos representando en la reconstrucción del flujo, como se puede ver en la gráfica de energía acumulada. Con esta información, se puede escoger el número de modos apropaidos para representar con fidelidad la estructura; pero sin hacer muy pesado computacionalmente la reconstrucción.

Modos estables

Otro pregunta que hay que responderse, y que no suele surgir naturalmente, antes de ejecutar el POD es ¿Qué modos del sistemas son estables? Esta pregunta es imporatan responderla porque, básicamente los modos estables son los que deberían usarse para la reconstrucción del flujo y aquellos que son inestables no deben ni "mirarse" porque arruinan la reconstrucción.

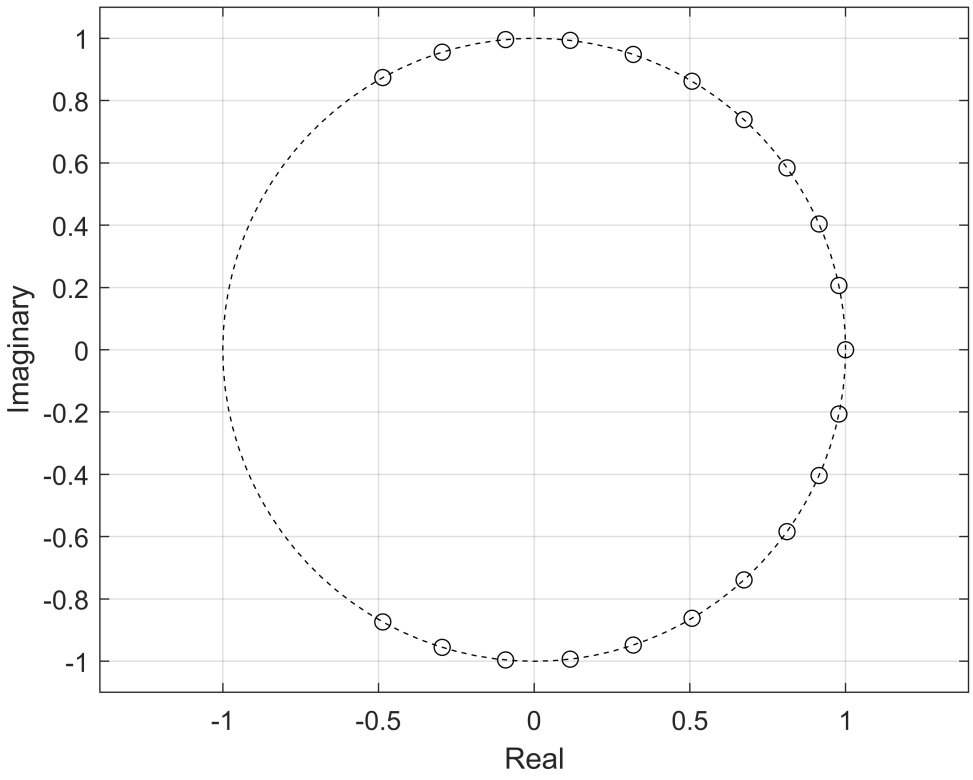
Y bien ¿Cómo saber qué modos son estables? Primero se debe definir la martiz $\widetilde{\mathbf{A}}$:

$$\widetilde{\mathbf{A}} \approx \mathbf{U}\mathbf{X}^T\mathbf{V}\mathbf{\Sigma}^{-1}$$

Donde $\mathbf{X}$ es la matriz de Snapshots. Luego se calculan los autovalores de la matriz $\widetilde{\mathbf{A}}$ y se grafican en el plano complejo. Si los autovalores se encuentran sobre o dentro de la circunferencia del círculo unitario, significa entonces que son estables y por tanto que pueden ser usados para la resconstrucción del flujo. Los modos que estén por fuerza del circulo unitario son inestables.

```
Avortex1 = VORTALL(:,1:end-1);
```

```
Avortex2 = VORTALL(:,2:end);

r = 21;
U = U(:, 1:r);
S = S(1:r,1:r);
V = V(:, 1:r);
Awierd = U'*Avortex2*V*inv(S);

[W,eigs] = eig(Awierd);
Phi = Avortex2*V*inv(S)*W;

fig2 = figure;
theta = (0:1:100) *2*pi/100;
plot(cos(theta),sin(theta),"k--") % plot unit circle
hold on, grid on
scatter(real (diag (eigs)),imag (diag(eigs)),"ok")
axis ([-1.1 1.1 -1.1 1.1]);
axis equal;
xlabel("Real")
ylabel("Imaginary")
hold off
```

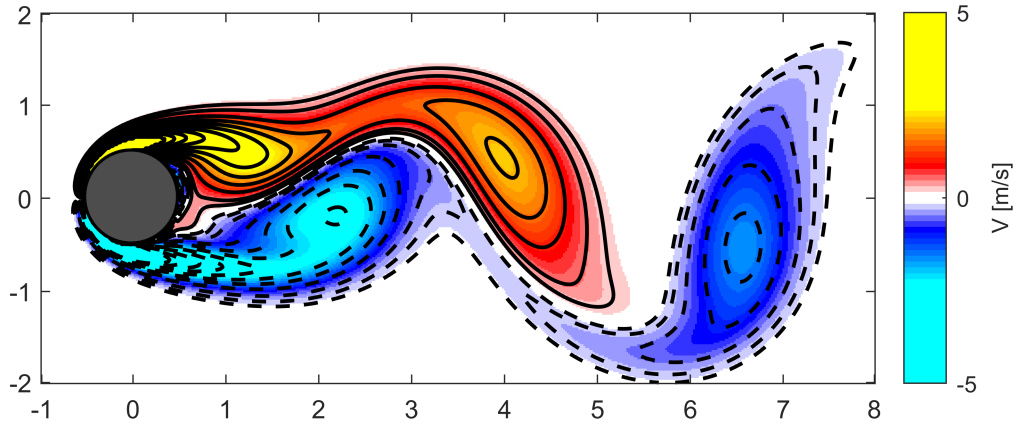


Reconstrucción del sistema

Primero echémosle un vistazo al sistema original. En libro de donde se sacaron los tomas indican que la matriz de Snapshot se organizó con tomando datos de la simulación cada $10\Delta t$, donde el $\Delta t = 0.02$ para satsifacer

número de CFL. Además, usaron un dominio espacial con 449 nodos en dirección X y 199 nodos en dirección Y.

```
delta_t = 0.02;                      %Paso temporal
t_analisis = 15;                     %Tiempo que se desea graficar
t_index = t_analisis/(10*delta_t);   %índice dentro de la matriz de Snapshots

nx = 199;
ny = 449;
prueba1 = reshape(Avort(:,t_index),nx,ny);

fig3 = figure;
plotVortex(prueba1)
```

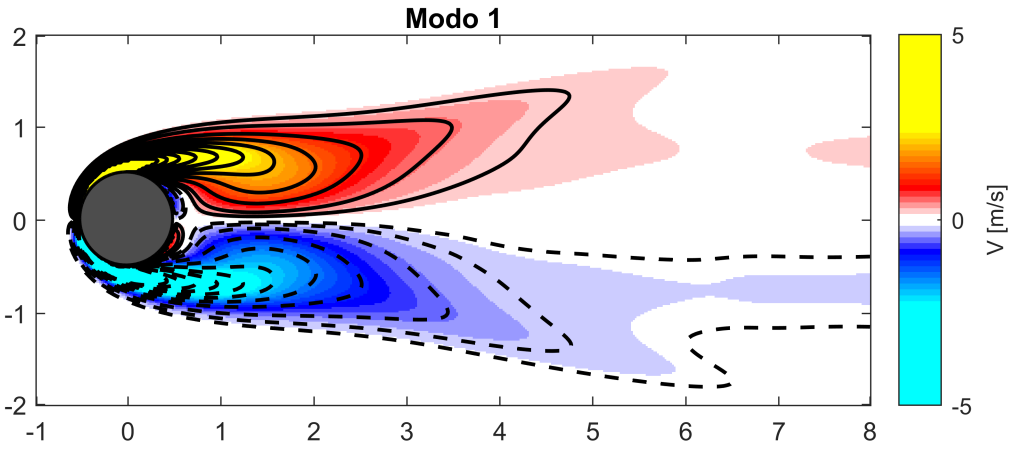


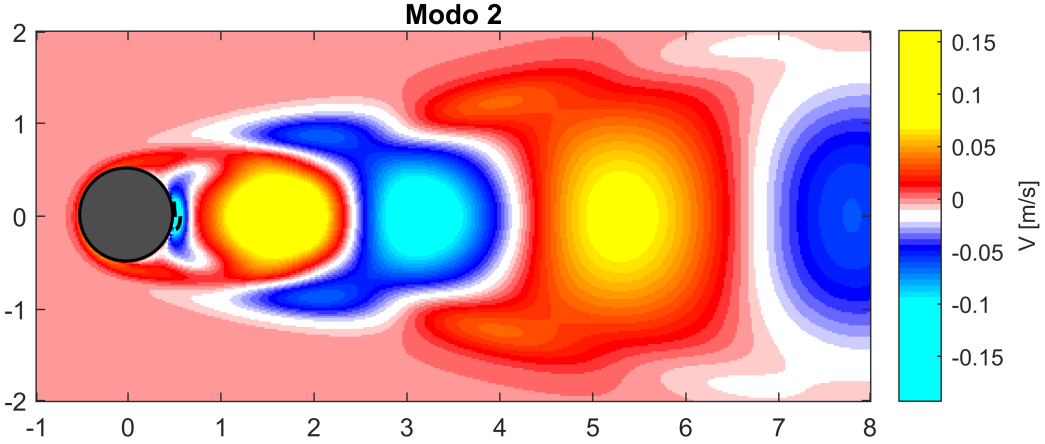

```
ans =
  Figure (4) with properties:

      Number: 4
        Name: ''
       Color: [1 1 1]
    Position: [500 300 600 260]
       Units: 'pixels'

  Show all properties
```

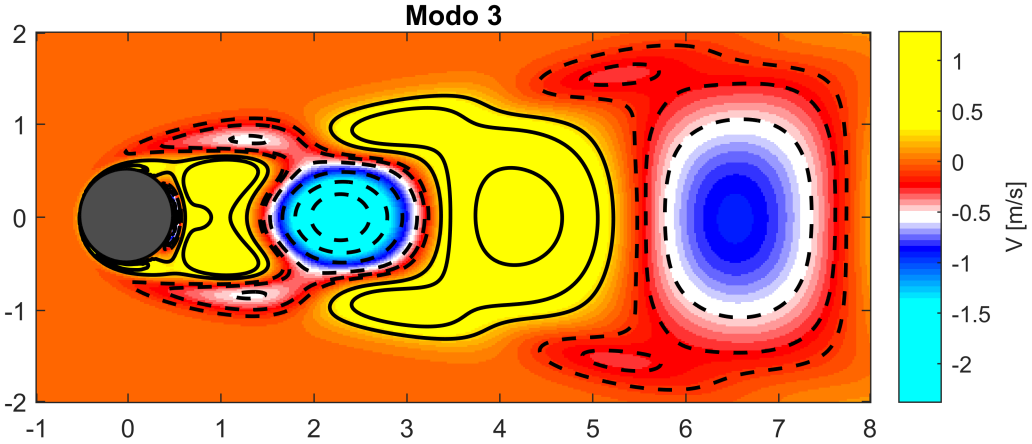
Con esta figura se puede visualizar el fenómeno de la **calle de vórtices de Von Kármán**, patrón de vórtices que se repite debido a la separación no estacionaria de la capa de fluido al pasar sobre cuerpos sumergidos, en este caso, de un cilindro de longitud infinita.

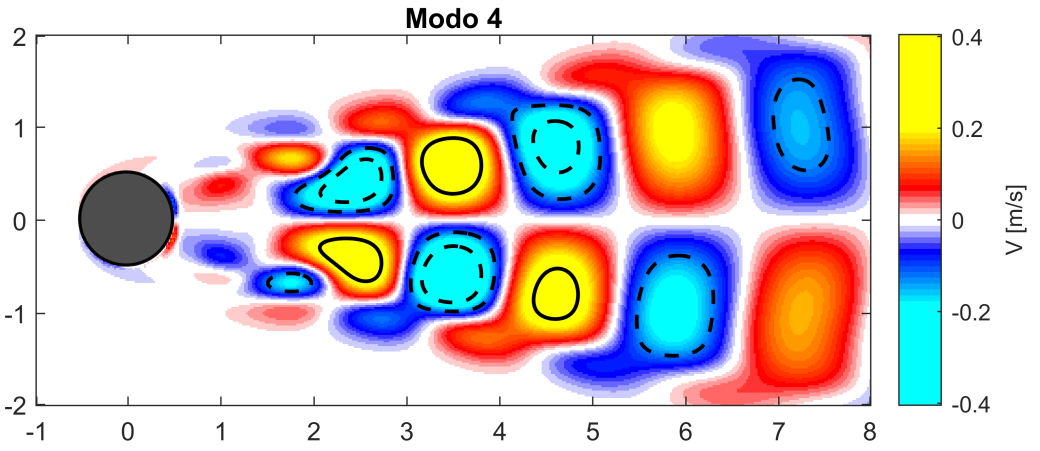
Ahora bien, conociendo el contenido energético de los modos y habiendo identificado cuáles son estables, ya se puede pasar a la reconstrucción del sistema. Si aún no se tiene claro cuántos modos usar, se podrían hacer varios experimentos y validar por inspección visual qué tan bien se está reconstruyendo el flujo. Grafiquemos los primeros 8 modos por separado:

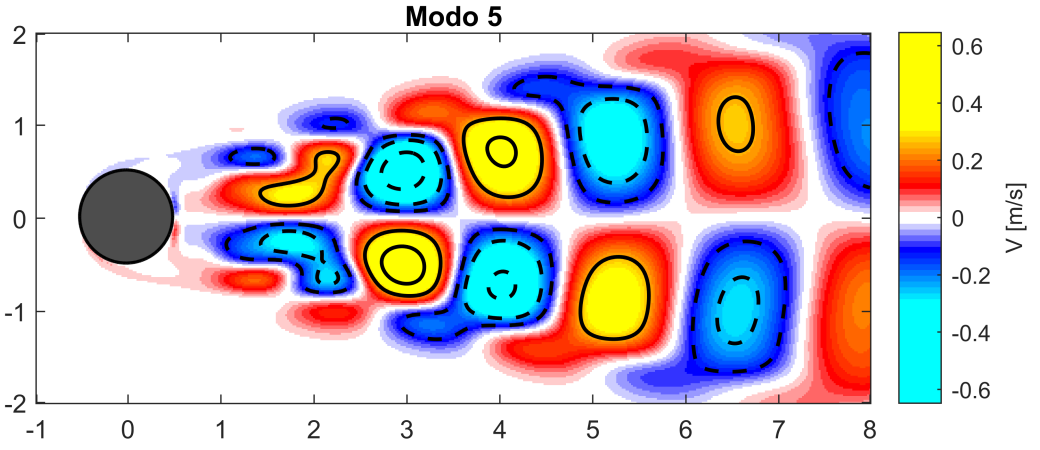
```
%Estudiamos los modos por separado
r = [1:1:8];
modes(r, U, alpha, t_index, nx, ny)
```

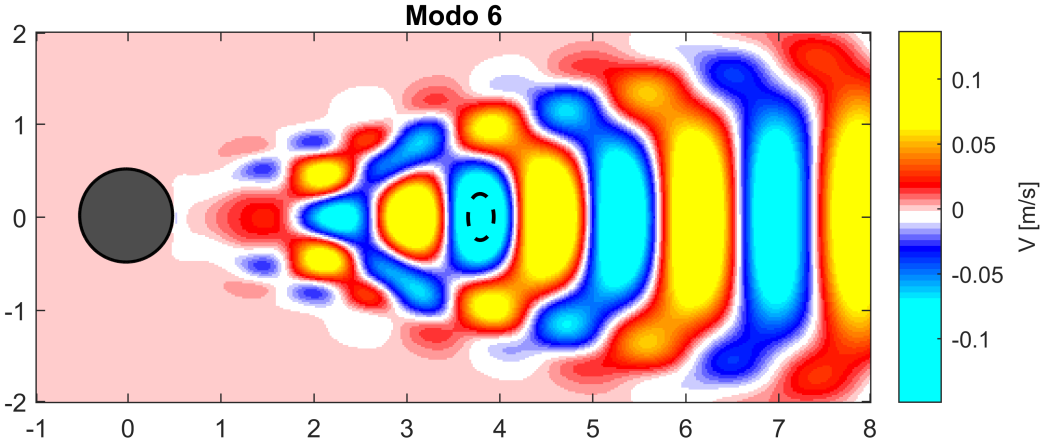

Modo 1
2
1
0
-1
-2
-1
0
1
2
3
4
5
6
7
8
5
0
-5
V [m/s]

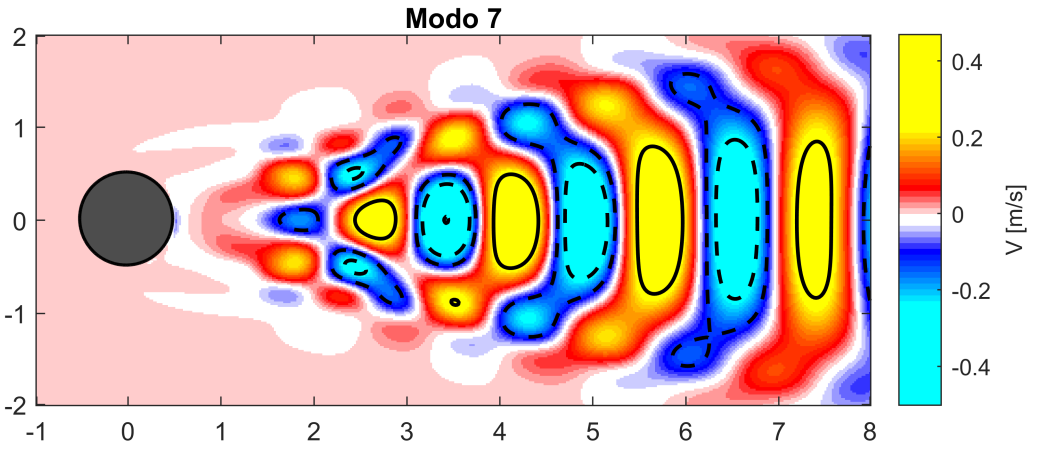

Modo 2
2
1
0
-1
-2
-1
0
1
2
3
4
5
6
7
8
0.15
0.1
0.05
0
-0.05
-0.1
-0.15
V [m/s]

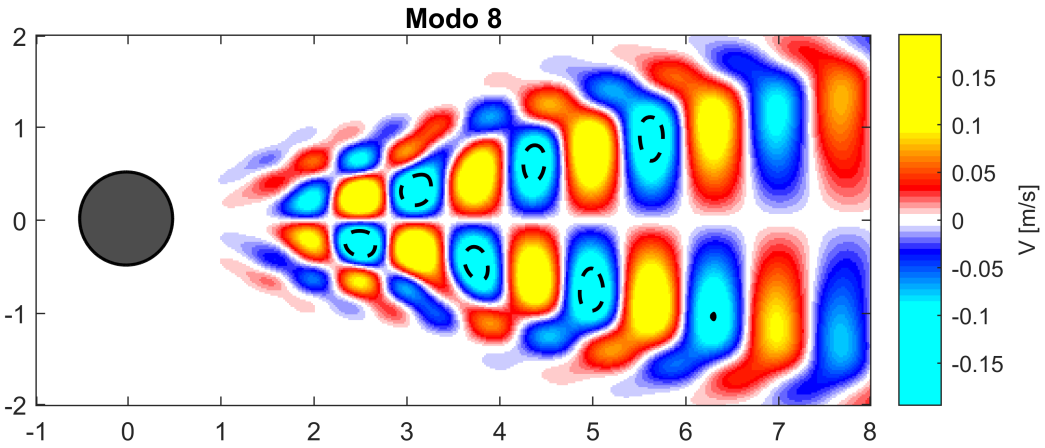

Modo 3
2
1
0
-1
-2
-1
0
1
2
3
4
5
6
7
8
1
0.5
0
-0.5
-1
-1.5
-2
V [m/s]


Modo 4
2
1
0
-1
-2
-1
0
1
2
3
4
5
6
7
8
0.4
0.2
0
-0.2
-0.4
V [m/s]


Modo 5
2
1
0
-1
-2
-1
0
1
2
3
4
5
6
7
8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
V [m/s]


Modo 6
2
1
0
-1
-2
-1
0
1
2
3
4
5
6
7
8
0.1
0.05
0
-0.05
-0.1
V [m/s]



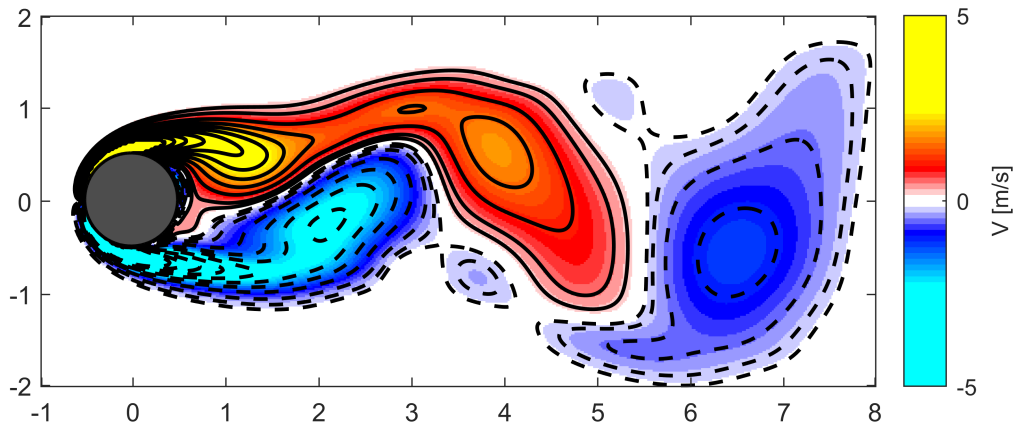


¿Qué representa cada uno de estos modos?

- El modo 1 contiene las características principales de flujo, sin el componente oscilatorio.
- El modo 2 representa el desprendimiento de vórtices. Crece asintóticamente.
- Modo 2 y 3 son complementarios y representan el desprendimiento de los vórtices. Contienen oscilación en el eje horizontal.
- Modo 4 y 5 son complementarios y representan el desprendimiento y la oscilación en el eje vertical y horizontal.

A modo de ejemplo, usamos los 5 primeros modos para reconstruir el sistemas:

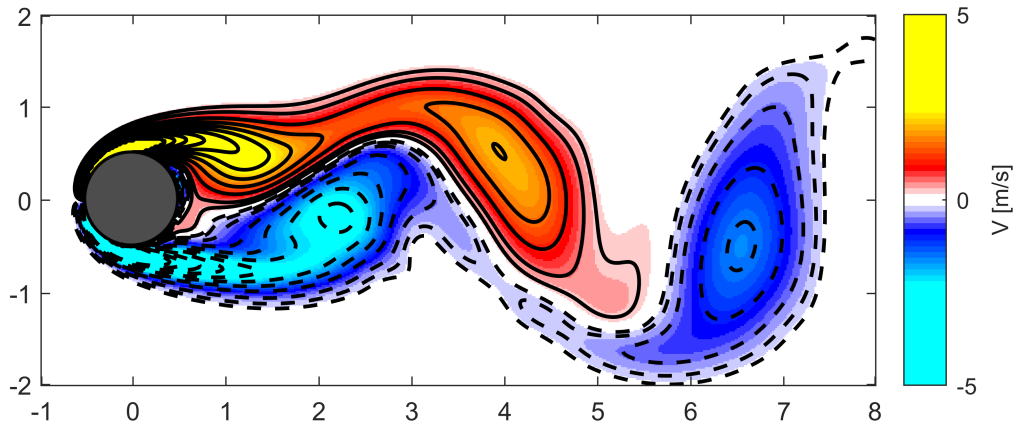
```
r = 5;
%Contruyo mis matrices usando r modos
U_red = U(:, 1:r);
alpha_red = alpha(1:r,:);

Avort_red = U_red*alpha_red(:,t_index);
Avort_red5 = reshape(Avort_red,nx,ny);
fig5 = figure;
plotVortex(reshape(Avort_red,nx,ny))
```




```
ans =
  Figure (15) with properties:

      Number: 15
        Name: ''
       Color: [1 1 1]
    Position: [500 300 600 260]
       Units: 'pixels'

  Show all properties
```

Ahora reconsutryamos el sistema usando 8 modos:

```
r = 8;
%Contruyo mis matrices usando r modos
U_red = U(:, 1:r);
alpha_red = alpha(1:r,:);

Avort_red = U_red*alpha_red(:,t_index);
Avort_red8 = reshape(Avort_red,nx,ny);
fig6 = figure;
plotVortex(reshape(Avort_red,nx,ny))
```




```
ans =
```

```
Figure (17) with properties:

    Number: 17
      Name: ''
     Color: [1 1 1]
  Position: [500 300 600 260]
     Units: 'pixels'

Show all properties
```

Cuantificar pérdida de información

A medida que vamos añadiendo modos en la combinación lineal, mejor es la aproximación. Esto se pudo comprobar por medio de una inspección visual, revisando simplemente los "detalles" de la gráfica. Sin embargo, en ciencia es importante cuantificar el error.Esto se puede hacer por medio de la siguiente ecuación:

$$\left(\sum_{j=1}^{cols} \sum_{i=1}^{rows} |\mathbf{a_{ij}} - \mathbf{\tilde{a}_{ij}}|^2 \right)^{(1/2)}$$

```
%Error entre la matriz orginal y la matriz con 5 modos
error5 = norm(prueba1-Avort_red5, "fro")
```

```
error5 = 44.6585
```

```
%Error entre la matriz orginal y la matriz con 8 modos
error8 = norm(prueba1-Avort_red8, "fro")
```

```
error8 = 14.2826
```

El error calculado por medio de la norma L_2 disminuye a medida que aumenta el número de modos.

```
function f1 = plotVortex(Vortex)

    load CCcool.mat
    f1 = figure;
    vortmin = -5;
    vortmax = 5;

    Vortex(Vortex>vortmax) = vortmax;
    Vortex(Vortex<vortmin) = vortmin;

    imagesc(Vortex);
    colormap(CC);

    xticks([1 50 100 150 200 250 300 350 400 449])
    xticklabels({'-1','0','1','2','3','4','5','6','7','8'})
    yticks([1 50 100 150 199])
    yticklabels({'2','1','0','-1','-2'})
    set(gcf,'Position',[500 300 600 260])
    axis equal
    hold on
```

```
    contour(Vortex , [vortmin-0.5:0.5:-0.5 -0.250 -0.125], "--k", "LineWidth",1.4)
    contour(Vortex , [0.250 0.5:0.5:vortmax], "-k", "LineWidth",1.4)
    cb = colorbar;
    ylabel(cb,'V [m/s]');

    theta = [0:0.1:359.9]';
    x = ones(size(theta,2));
    y = ones(size(theta,2));
    x = 49+25*sind(theta);  %Son esos centro porque la imagen está escalada
    y = 99+25*cosd(theta);
    fill(x,y,[.3 .3 .3])  % place cylinder
    plot(x,y,'k','LineWidth',1.2) % cylinder boundary
    hold off
end

function [] = modes(r, U, alpha, t_index, nx, ny)
    numb = size(r,2);
    for i=1:numb
        figure(4+i);
        U_red = U(:,r(i));
        alpha_red = alpha(r(i),t_index);
        Avort_red = U_red*alpha_red;
        plotVortex(reshape(Avort_red,nx,ny));
        title(sprintf("Modo %d",r(i)))
    end
end
```