

# Réduction de Modèle : Approches basiques diverses

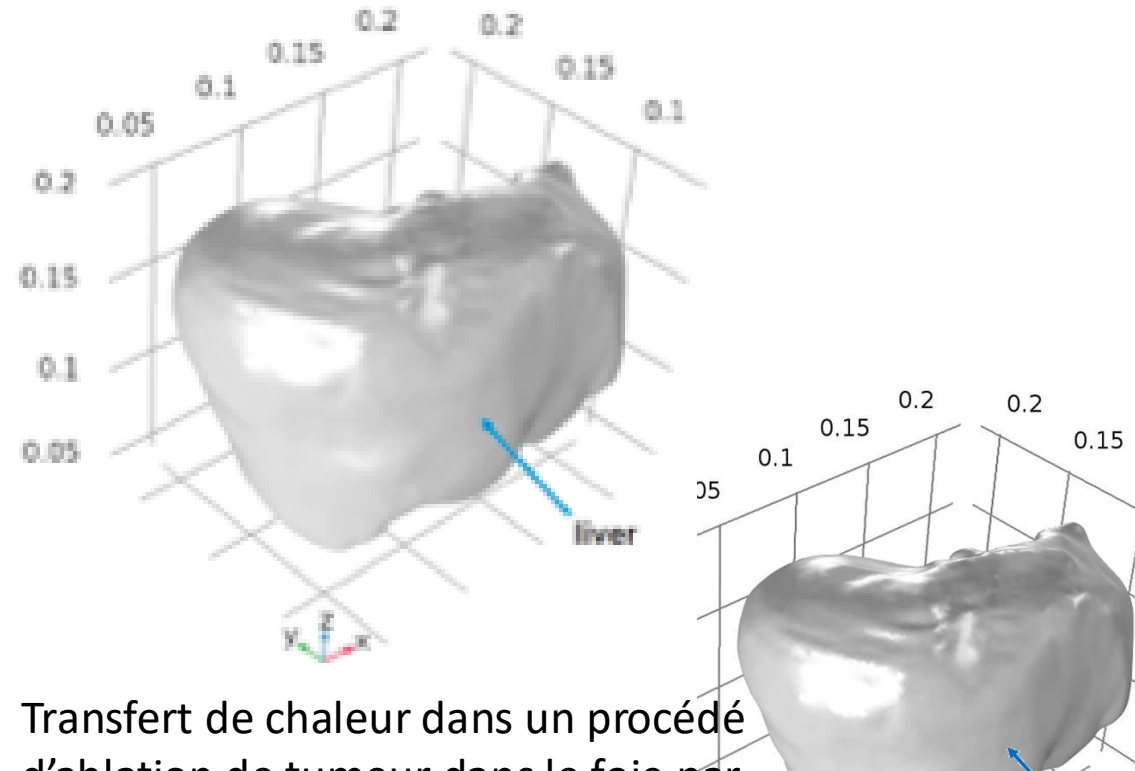
Réduction modale, POD, équilibrée et identification par sous-espace

<https://github.com/JLB-UB/MODRED>

Jean-Luc BATTAGLIA  
I2M, Université de Bordeaux

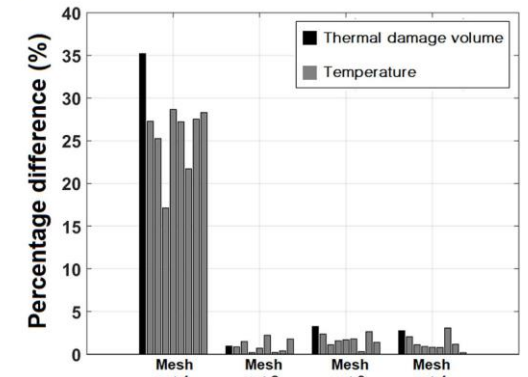
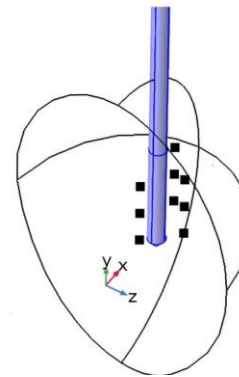
# Modèle

- Un modèle de connaissance s'appuie sur les équations de conservation (masse, quantité de mouvement et énergie) ainsi que sur les lois de transport (Fourier, Fick, ...)
- Les conditions aux limites et initiales complètent l'écriture.
- Les EDP obtenues sont généralement résolues par des méthodes de discrétisation temporelles et spatiales (différences finies, éléments finis, volumes finis)
- **L'ordre  $n$**  du modèle est relatif au nombre de degrés de liberté du maillage



Transfert de chaleur dans un procédé d'ablation de tumeur dans le foie par méthode thermique

<https://doi.org/10.1016/j.compbio.2019.01.003>



# Pourquoi réduire un modèle ?

- Réduire le coût de calcul
  - Adapter le modèle aux besoins (simulation temps réel, contrôleurs rapides, etc.)
  - Utilisation dans une procédure d'inversion de données expérimentales
- La réduction ne consiste pas à simplifier le modèle de connaissance en :
  - Simplifiant la géométrie
  - Dégradant le maillage
  - Rajoutant des hypothèses simplificatrices (accommodation thermique par exemple)
- Nous restreindrons la présente présentation aux **systèmes linéaires et stationnaires** (bien qu'en principe on verra que certaines méthodes se prêtent assez bien à des configurations non linéaires)

# Formulation du modèle (1/2)

On peut représenter le modèle des EDP sous la forme d'un système linéaire en espace d'état (modèle d'état):

$$\begin{cases} \dot{x}(t) = A x(t) + B u(t) \\ y(t) = C x(t) + D u(t) \end{cases}$$

- $x(t)$  : vecteur d'état de dimension  $n$  (températures aux nœuds du maillage)
- $u(t)$  : vecteur d'entrée de dimension  $p$  (température, flux, source)
- $y(t)$  : vecteur de sortie de dimension  $m$  (température observées)
- $A$  : matrice dynamique ( $n \times n$ )
- $B$  : matrice d'entrée ( $n \times p$ )
- $C$  : matrice de sortie ( $m \times n$ )
- $D$  : matrice de transmission directe ( $m \times p$ ) (souvent nulle)

# Formulation du modèle (2/2)

En prenant la transformée de Laplace ou Fourier du modèle d'état :

$$\begin{cases} \dot{x}(t) = A x(t) + B u(t) \\ y(t) = C x(t) + D u(t) \end{cases}$$

On aboutit au **modèle de transfert** ( $p = j \omega$ ) :

$$Y(p) = [C (p I_n - A)^{-1} B + D] U(p)$$

$$Y(p) = G(p)U(p)$$

# Fonctionnement classique des méthodes de réduction

- On **change de base** pour l'expression d'un champ en lien avec l'observable (revoir les **méthodes de séparation des variables** si vous avez des doutes)
- On établit **un lien** entre la nouvelle base et l'ancienne
- On détermine quels sont les  $r$  éléments « **dominants** » parmi les  $n$  du champ projeté sur la nouvelle base
- On ne **conserve** que ces modes et on ignore les autres (souvent assez satisfaisant selon les techniques).

# Réduction Modale (1/6)

## Principe

- Diagonalisation de la matrice dynamique
- Conservation des modes dominants (constante de temps, contribution énergétique, ...)



### Avantages

- Intuitif, **basé sur la physique**



### Limites

- Seulement pour systèmes linéaires ou faiblement non linéaires
- Moins satisfaisant avec des modes proches ou couplés

# Réduction Modale (2/6)

- **Étape 1** : Résolution du problème aux valeurs propres :

$$A V = \Lambda V$$

Avec :

- $V$  : matrice des vecteurs propres (modes **spatiaux** orthogonaux)
- $\Lambda$  : matrice diagonale des valeurs propres (**constante de temps**, pôles du système)

- **Étape 2** : Transformation modale :  $x(t) = V z(t)$

Système transformé :

$$\begin{aligned} \dot{z}(t) &= \Lambda z(t) + \Gamma u(t) \\ y(t) &= \Omega z(t) + D u(t) \end{aligned}$$

$$\text{avec : } \Gamma = V^{-1} B = V^T B \text{ et } \Omega = C V$$

**CODE**

```
[V,Lambda] = eig(A);
```

```
Gamma = V'*B;  
Omega = C*V;
```



# Réduction Modale (3/6)

- **Étape 3** : On partitionne en  $r \ll n$  modes dominants et  $n - r$  modes non-dominants :

$$\text{Équation des états : } \begin{bmatrix} \dot{z}_r \\ \dot{z}_s \end{bmatrix} = \begin{bmatrix} \Lambda_r & 0 \\ 0 & \Lambda_s \end{bmatrix} \begin{bmatrix} z_r \\ z_s \end{bmatrix} + \begin{bmatrix} \Gamma_r \\ \Gamma_s \end{bmatrix} u$$

$$\text{Equation de sortie : } y = [\Omega_r \quad \Omega_s] \begin{bmatrix} z_r \\ z_s \end{bmatrix} + D u$$

- Le critère de dominance peut être relatif à :
  - la valeur de  $\lambda_i$  (en conduction  $\tau_i = -1/\lambda_i$ )
  - La contribution du mode à la réponse à une excitation de type Heaviside :

$$D_i = \sum_{j=1}^p -\frac{\Gamma_{i,j} \Gamma_{i,j}^*}{2\lambda_i}$$

## CODE

```
LambdaR = diag(z(1:r),0);  
LambdaS = diag(z(r+1:n),0);  
GammaR = Gamma(1:r,:);  
GammaS = Gamma(r+1:n,:);  
OmegaR = Omega(:,1:r);  
OmegaS = Omega(:,r+1:n);
```

# Réduction Modale (4/6)

- Remarque : pour les très gros systèmes  $n \sim 10^4 - 10^6$  *eig.m* va prendre beaucoup trop de temps, il vaut mieux calculer les  $n_i \sim 10^2 - 10^3$  premiers modes par une méthode Lanczos/Arnoldi et effectuer la réduction  $\rightarrow$  *eigs.m*.

- Puis réduction aux seuls modes dominants :

$$\begin{aligned}\dot{z}_r &= \Lambda_r z_r + \Gamma_r u \\ y &= \Omega_r z_r + D u\end{aligned}$$

- Ce système peut être efficacement simulé avec une méthode de type Runge-Kutta (*ode15s.m*)

## CODE

```
[V,Lambda] = eigs(A,ni,flag2);
```

```
Gamma = -V\B;  
Omega = C*V;
```

```
LambdaR = diag(z(1:r),0);  
GammaR = Gamma(1:r,:);  
OmegaR = Omega(:,1:r);
```

# Réduction Modale (5/6)

- La mise de côté des modes non-dominants introduit un biais de modélisation
- Solution simple et rapide : correctif **quasi-statique** des modes non-dominants

$$\begin{bmatrix} \dot{z}_r \\ 0 \end{bmatrix} = \begin{bmatrix} \Lambda_r & 0 \\ 0 & \Lambda_s \end{bmatrix} \begin{bmatrix} z_r \\ z_s \end{bmatrix} + \begin{bmatrix} \Gamma_r \\ \Gamma_s \end{bmatrix} u \Rightarrow$$

$$y = \Omega_r z_r + (D - \Omega_s \Lambda_s^{-1} \Gamma_s) u$$

- La prise en compte de la solution en régime permanent ( $A^{-1}B$ ) peut aussi être prise en compte avant la réduction (attention cependant aux temps de calcul).

## CODE

```
[V,Lambda] = eigs(A,m,flag2);
```

```
Gamma = V'*B;
```

```
Omega = C*V;
```

```
LambdaR = diag(z(1:r),0);
```

```
LambdaS = diag(z(r+1:n),0);
```

```
GammaR = Gamma(1:r,:);
```

```
GammaS = Gamma(r+1:n,:);
```

```
OmegaR = Omega(:,1:r);
```

```
OmegaS = Omega(:,r+1:n);
```

```
iLambdaS = 1./LambdaS;
```

```
D = D-OmegaS*GammaS*iLambdaS;
```

# Réduction Modale (6/6)

Méthodes plus élaborées :

- **Amalgame modal** (combinaison linéaire des modes non dominants sur les modes dominants)
- Méthode de Litz : correction optimale par minimisation de l'écart entre réponse du modèle complet et celle du modèle réduit.
- ...

# Réduction POD (Proper Orthogonal Decomposition) (1/4)

## Principe

- Collecte de snapshots (instantanés de simulation ou mesures)
- Décomposition en valeurs singulières (SVD)
- Conservation des modes les plus énergétiques



## Avantages

- Peut être basée sur les données (pas besoin de connaître le système complet)
- Très efficace pour la simulation



## Limites

- Dépend des données choisies (risque de sous ou sur-ajustement)
- Peut manquer certains phénomènes.

# Rappel : Que fait une SVD ? (1/3)

- Elle permet de **décomposer une matrice rectangulaire quelconque** en un produit de trois matrices aux propriétés très utiles.
- Soit  $M \in \mathbb{R}^{m \times n}$  une matrice réelle (pas nécessairement carrée). La décomposition SVD s'écrit :

$$M = U \Sigma V^T$$

- Avec :
  - $U \in \mathbb{R}^{m \times m}$  : matrice carrée orthogonale  $U^T U = I$  (modes)
  - $\Sigma \in \mathbb{R}^{m \times n}$  : est une matrice **diagonale par blocs** avec des **valeurs singulières** sur la diagonale  $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_l > 0$  où  $l = \text{rang}(M)$
  - $V \in \mathbb{R}^{n \times n}$  : matrice carrée orthogonale (coefficients temporels).

# Rappel : Que fait une SVD ? (2/3)

La SVD exprime toute matrice comme une **transformation géométrique** en trois étapes :

- $V^T$  : rotation/changement de base de l'espace d'entrée
  - $\Sigma$  : mise à l'échelle (étirement/écrasement) le long d'axes orthogonaux
  - $U$  : rotation/changement de base de l'espace de sortie
- 
- La SVD peut être vue comme une généralisation du théorème spectral aux matrices non symétriques ou non carrées.

# Rappel : Que fait une SVD ? (3/3)

**Exemple** : soit une transformation linéaire de matrice  $M$  :

$$M = \begin{bmatrix} 3 & 1 \\ 0 & 2 \end{bmatrix}$$

On l'applique à un cercle unité  $\rightarrow$  on obtient une ellipse.

La SVD de  $M = U \Sigma V^T$  donne :

$$V^T = \begin{bmatrix} 0.88 & 0.47 \\ -0.47 & 0.88 \end{bmatrix}$$

qui correspond à une rotation d'environ  $-28^\circ$ ,

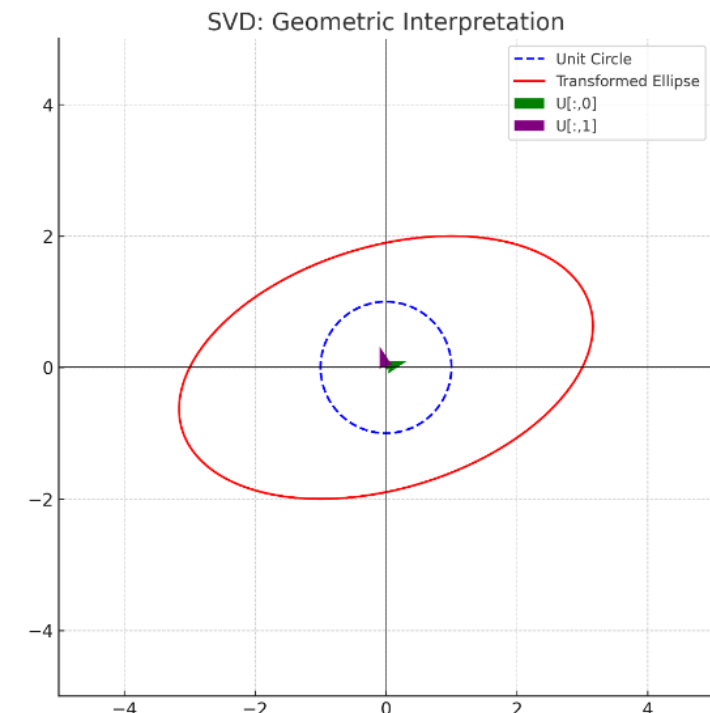
$$\Sigma = \begin{bmatrix} 3.26 & 0 \\ 0 & 1.84 \end{bmatrix}$$

C'est un étirement de 3,26 et 1,84 dans les deux directions principales. Et :

$$U = \begin{bmatrix} 0.96 & -0.29 \\ 0.29 & 0.96 \end{bmatrix}$$

On revient dans l'espace initial par une rotation d'environ  $17^\circ$

- Le cercle bleu pointillé représente l'**espace d'entrée** (unité).
- L'ellipse rouge est le **résultat de la transformation** par la matrice  $M$ .
- et ● sont les **vecteurs propres de la sortie** (colonnes de  $U$ ), orientés selon les directions principales de l'ellipse.





# Réduction POD (2/4)

## Étape 1 : Génération des snapshots

- On simule le système complet (ou on l'observe expérimentalement) pour collecter  $N$  snapshots (valeur initiale nulle avec  $u = 1$  par exemple) :

$$x(t_1), x(t_2), \dots, x(t_N)$$

Remarque : pour résoudre le modèle d'état il est conseillé d'utiliser une méthode robuste (Runge-Kutta par exemple)

- On forme la matrice de snapshots :

$$X = [x(t_1) \quad x(t_2) \cdots x(t_N)] \in \mathbb{R}^{n \times N}$$

```
t_final = 10; % temps final de
simulation (à ajuster)
numSnapshots = 1000; % nombre de
snapshots voulus
tspan = linspace(0, t_final,
numSnapshots);
% Définition de la dynamique avec
entrée constante u=1
odefun = @(t,x) A*x + B*1;
% Conditions initiales
x0 = zeros(n,1);
% Résolution avec ode15s
[~, Xsol] = ode15s(odefun, tspan,
x0);

% Transposition car ode15s renvoie
solution (temps x variables)
X = Xsol';
```

# Réduction POD (3/4)

**Étape 2** : On effectue une **décomposition en valeurs singulières** (SVD) de la matrice des snapshots (si  $X$  est large utiliser *svds.m* plutôt que *svd.m*):

$$X = U \Sigma V^T$$

On retient les  $r$  premiers modes (ceux associés aux plus grandes valeurs singulières  $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_r$ ) :

$$U_r = [u_1 \ u_2 \ \dots \ u_r] \in \mathbb{R}^{n \times r}$$

On projette l'état complet  $x(t)$  dans la base réduite :

$$x(t) = U_r z(t)$$

En injectant dans le système d'état :

$$U_r \dot{z}(t) = A U_r z(t) + B u(t)$$

```
% === SVD tronquée ===  
[U, S, ~] = svds(X, m,  
'largest');  
svals = diag(S);
```

```
% === Construction du  
modèle réduit ===  
V_r = U(:, 1:r);  
V_s = U(:, r+1:end);
```

# Réduction POD (4/4)

Multiplication à gauche par la transposée des modes :

$$U_r^T U \dot{z}(t) = U_r^T A U_r z(t) + U_r^T B u(t)$$

Comme  $U_r^T U_r = I_r$ , alors le modèle réduit est :

$$\begin{cases} \dot{z}(t) = A_r z(t) + B_r u(t) \\ y(t) = C_r z(t) + D u(t) \end{cases}$$

$$A_r = U_r^T A U_r$$

$$\text{Avec : } B_r = U_r^T B$$

$$C_r = C U_r$$

On peut appliquer la même correction statique que pour la méthode de réduction modale

$$A\_m = U' * A * U;$$

$$B\_m = U' * B;$$

$$C\_m = C * U;$$

$$A_r = A\_m(1:r, 1:r);$$

$$B_r = B\_m(1:r, :);$$

$$C_r = C\_m(:, 1:r);$$

$$A_s = A\_m(r+1:end, r+1:end);$$

$$B_s = B\_m(r+1:end, :);$$

$$C_s = C\_m(:, r+1:end);$$

if ~isempty(As)

$$D_r = D - C_s * (A_s \setminus B_s);$$

else

$$D_r = D;$$

end

# Premières conclusions Modal vs. POD

## Efficacité et pertinence

- **Réduction modale :**



- Très efficace si le système est linéaire, stable, et que les modes propres sont bien séparés et dominants.
- Facile à interpréter physiquement (chaque mode correspond à une vibration propre, une fréquence propre).
- Mais peut être moins adaptée aux systèmes fortement non-linéaires.

- **POD :**

- Très efficace pour des systèmes où on dispose de données représentatives (simulations, expériences).
- Peut capturer des comportements complexes, y compris non linéaires ou couplés, si les snapshots sont bien choisis.
- Optimise la représentation en capturant la majeure partie de l'énergie (variance) dans un petit nombre de modes.
- Peut donc réduire la dimension plus efficacement en pratique dans beaucoup de cas complexes.

# Réduction équilibrée (Balanced Truncation) (1/11)

## Principe

- Calcul des Grammiens de contrôlabilité et observabilité
- Transformation pour les équilibrer et tronquer les états les moins influents
-  Avantages
- Garantit une borne d'erreur
- Maintient stabilité et robustesse
-  Limites
- Calcul intensif pour grands systèmes
- Moins adapté aux systèmes non linéaires

# Réduction équilibrée (2/11)

La réduction équilibrée vise à identifier les **états les plus importants** pour le transfert entrée  $\rightarrow$  sortie. Elle utilise :

- **la contrôlabilité** (quelle quantité d'énergie il faut injecter pour atteindre un état)
- **l'observabilité** (quelle quantité d'information on peut extraire de l'état via les sorties).
- La méthode consiste à **équilibrer** le système afin que contrôlabilité et observabilité soient **égales et diagonales**, puis à tronquer les états associés aux plus faibles valeurs.

# Réduction équilibrée (3/11)

## Étape 1 : calcul des grammians de contrôlabilité et d'observabilité

Ce sont deux matrices symétriques définies comme les solutions des **équations de Lyapunov** :

- Grammien de contrôlabilité
$$AW_c + W_cA^T + BB^T = 0$$
- Grammien d'observabilité :
$$A^TW_o + W_oA + C^TC = 0$$

Ils mesurent respectivement :

- l'**effort nécessaire** pour contrôler les états,
- la **quantité d'information** sur chaque état qui se reflète dans la sortie.

```
% Compute the controllability  
and observability Gramians  
Wc = lyap(A, B * B');  
Wo = lyap(A', C' * C);
```

# Réduction équilibrée (4/11)

- **Etape 2 : diagonalisation conjointe : transformation équilibrée**

On cherche une base dans laquelle **les deux grammians deviennent égaux et diagonaux** :

$$T^{-1}W_c T^{-\top} = T^{\top}W_o T = \Sigma = \text{diag}(\sigma_1, \dots, \sigma_n)$$

- Les  $\sigma_i$  sont appelés **valeurs singulières de Hankel** : elles mesurent l'importance des états.
- Les états avec petites valeurs de  $\sigma_i$  sont **peu contrôlables et peu observables**.



# Réduction équilibrée (5/11)

- **Etape 3 : Calcul de  $T$**

On forme la matrice :

$$M = W_c W_o$$

- On effectue la SVD de  $M$  :

$$M = U \Sigma V^T$$

- On construit  $T$  :

$$T = U \Sigma U^T$$

- On calcule  $T^{-1}$

```
% Perform balancing transformation
[U, S, V] = svd(Wc * Wo);
S = sqrt(diag(S));
T = U * diag(S) * U';
T_inv = inv(T);
```

# Réduction équilibrée (6/11)

- Mise en forme équilibrée

$$\hat{x} = T^{-1}x$$

Le système équilibré est alors :

$$\begin{cases} \dot{\hat{x}} = \hat{A}\hat{x} + \hat{B}u \\ y = \hat{C}\hat{x} + Du \end{cases}$$

Où :

$$\hat{A} = T^{-1}AT$$

$$\hat{B} = T^{-1}B$$

$$\hat{C} = CT$$

$$A\_bal = T\_inv * A * T;$$

$$B\_bal = T\_inv * B;$$

$$C\_bal = C * T;$$

# Réduction équilibrée (7/11)

## Etape 3 : Réduction

- On tronque les états associés aux plus petites valeurs singulières (les moins contrôlables et observables).
- Le système réduit à  $r$  valeurs singulières est alors :

$$\begin{cases} \dot{\hat{x}}_r = \hat{A}_{11}\hat{x}_r + \hat{B}_1u \\ y = \hat{C}_1\hat{x}_r + Du \end{cases}$$

où  $\hat{A}_{11}$  correspond à la sous-matrice de  $\hat{A}$  conservant les  $r$  premiers modes.

- La somme des valeurs singulières « supprimées » donne une estimation de la taille maximale de l'erreur.

% Truncate the balanced model

```
Ar = A_bal(1:order, 1:order);
```

```
Br = B_bal(1:order, :);
```

```
Cr = C_bal(:, 1:order);
```

# Réduction équilibrée (8/11)

- En pratique il vaut mieux **Calculer la décomposition de Cholesky** pour trouver  $X_c$  et  $X_o$  tels que :

$$W_c \approx X_c X_c^T \quad W_o \approx X_o X_o^T$$

- On calcule alors la SVD de :

$$M = X_o^T X_c = U \Sigma V^T$$

- On obtient la matrice de transformation :

$$T = X_c V \Sigma^{-1/2}, \quad T^{-1} = \Sigma^{-1/2} U^T X_o^T$$

% Factorisation de Cholesky

Rc = chol(Wc, 'lower');

Ro = chol(Wo, 'lower');

% Calcul de la matrice M et SVD

M = Ro' \* Rc;

[U,Sigma,V] = svd(M);

% Transformation d'équilibrage

Sigma\_sqrt\_inv = diag(1 ./  
sqrt(diag(Sigma)));

T = Rc \* V \* Sigma\_sqrt\_inv;

T\_inv = Sigma\_sqrt\_inv \* U' \* Ro';

# Réduction équilibrée (9/11)

⚠ Coûteux pour les systèmes de grande dimension  
(nécessite la résolution de Lyapunov)

Solution : On remplace les Grammiens exacts par des **approximateurs de Grammiens** construits à partir de **simulations temporelles du système**.

Pour  $W_c$  : on simule le système en réponse à des impulsions ou des signaux d'entrée riches, puis on empile les **trajectoires d'états** (snapshots).

$$X_c = [x(t_1), x(t_2), \dots, x(t_{N_c})] \in \mathbb{R}^{n \times N_c}$$

Pour  $W_o$  : on simule le **système adjoint** en réponse à des impulsions de sortie (ou signaux riches), et on empile aussi les snapshots adjoints.

$$X_o = [z(t_1), z(t_2), \dots, z(t_{N_o})] \in \mathbb{R}^{n \times N_o}$$

où  $z(t)$  est l'état du système adjoint :  $\dot{z}(t) = A^T z(t) + C^T v(t)$

```
u_func = @(t) ones(m,1); % Impulsion
unité
x0 = zeros(n,1);
[~, x_snap] = ode15s(@(t,x) A*x +
B*u_func(t), tspan, x0);
X = x_snap'; % (n x N)
```

```
u_adj_func = @(t) ones(p,1);
x0_adj = zeros(n,1);
[~, x_adj_snap] = ode15s(@(t,x) A'*x +
C'*u_adj_func(t), tspan, x0_adj);
Y = x_adj_snap'; % (n x N)
```

# Réduction équilibrée (10/11)

On approxime alors :

$$W_c \approx X_c X_c^\top$$

$$W_o \approx X_o X_o^\top$$

Ces approximations sont de rang faible (car  $N_c, N_o \ll n$ )

On réalise une **décomposition en valeurs singulières (SVD)** sur la matrice croisée :

$$X_o^\top X_c = U \Sigma V^\top$$

avec  $\Sigma$  contenant les **valeurs singulières d'Hankel approximées**.

On construit la base équilibrée réduite :

$$T = X_c V \Sigma^{-1/2}$$

$$S = X_o U \Sigma^{-1/2}$$

et on définit l'état réduit :  $\hat{x} = T^\top x$

```
M = Y' * X; % (N x N)
[U, S, V] = svd(M, 'econ'); %
Économie mémoire
```

```
S_root_inv = diag(1 ./
sqrt(diag(S(1:r,1:r))));
Phi = X * V(:,1:r) * S_root_inv;
Psi = Y * U(:,1:r) * S_root_inv;
```

# Réduction équilibrée (11/11)

## Modèle réduit

- On projette les matrices d'état comme suit :

$$A_r = T^T A T$$

$$B_r = T^T B$$

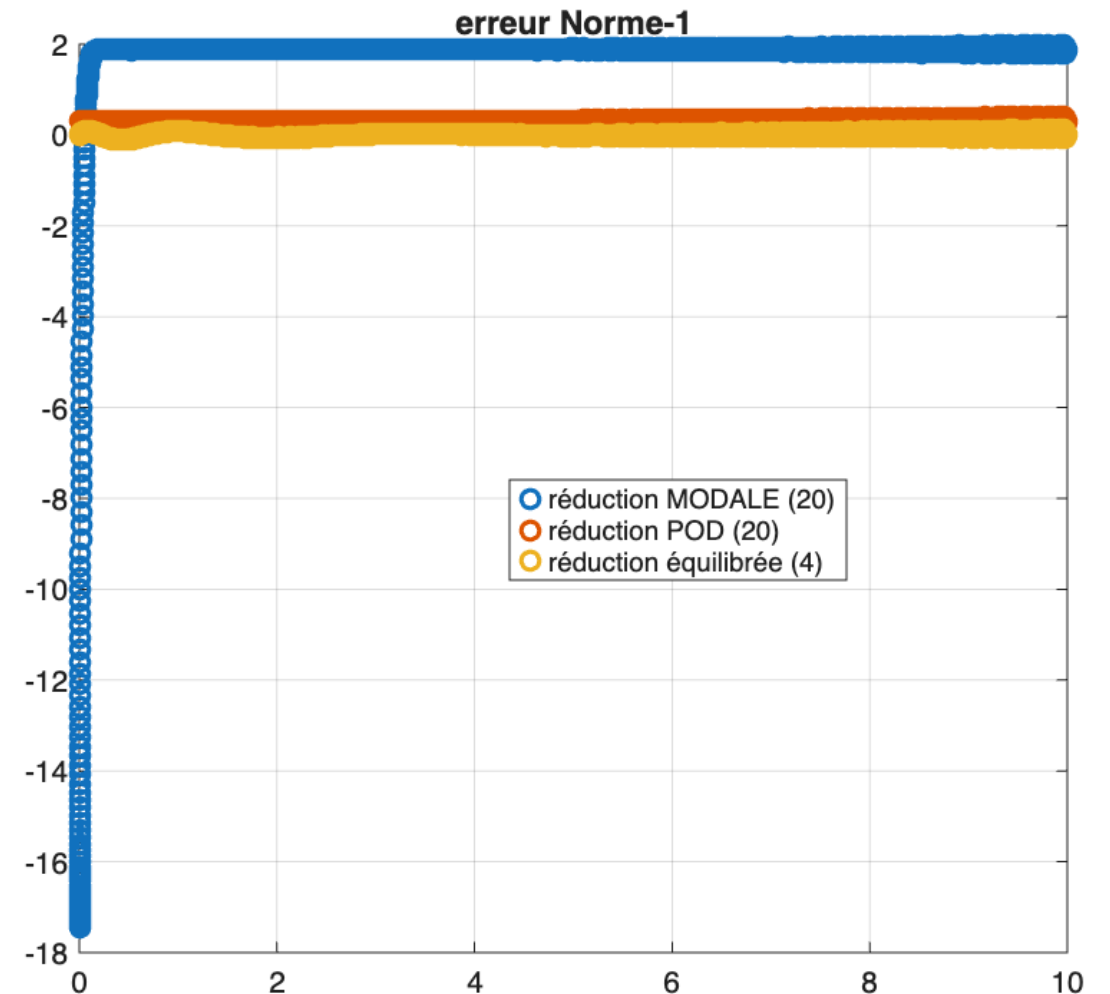
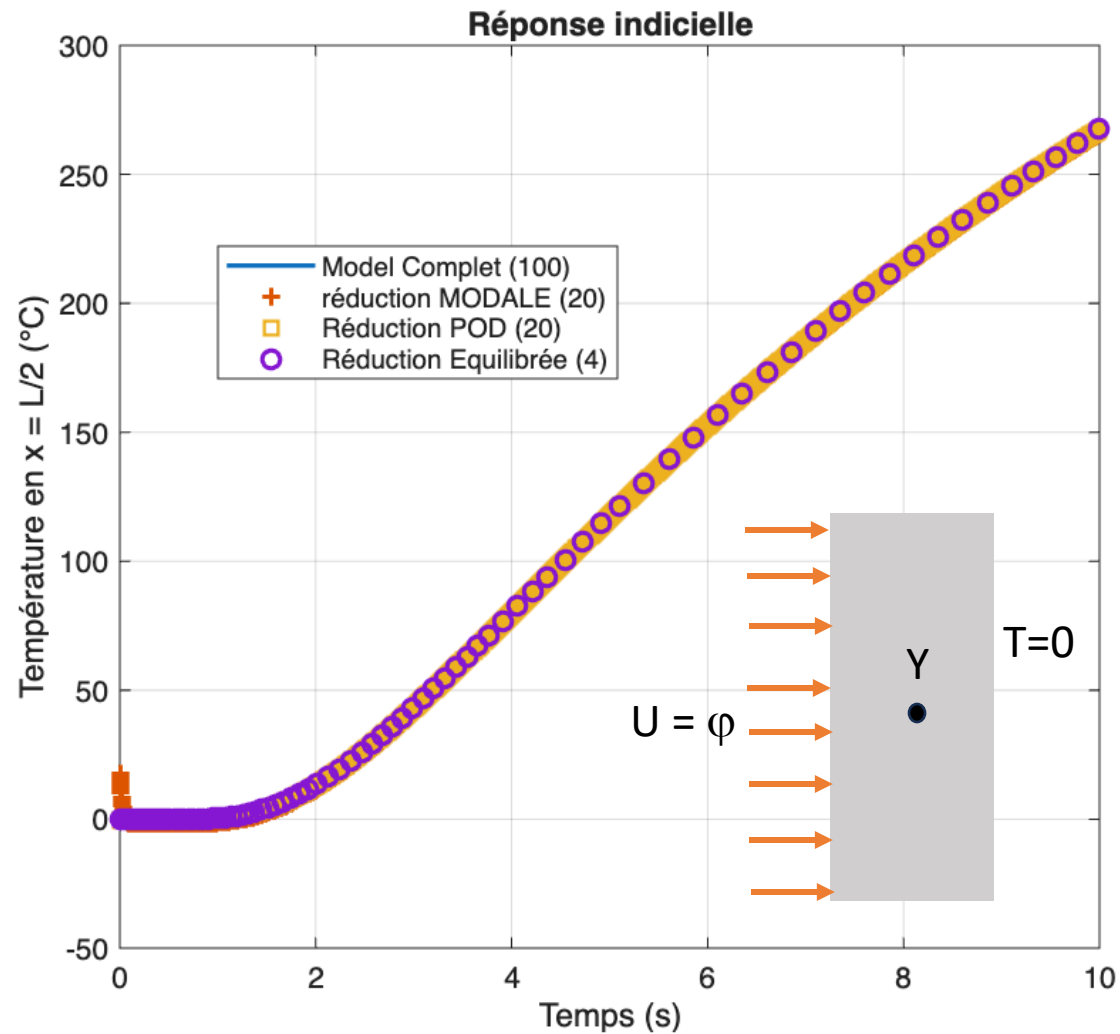
$$C_r = C T$$

$$D_r = D$$

On garde ensuite les  $r$  premiers modes selon les plus grandes valeurs singulières  $\Sigma$

$$\begin{aligned} A_r &= \Psi^T * A * \Phi; \\ B_r &= \Psi^T * B; \\ C_r &= C * \Phi; \\ D_r &= D; \end{aligned}$$



# Illustration





# Réduction par identification de sous-espace

## Principe

- Identification à partir des données d'entrée et de sortie
- Algorithmes N4SID, MOESP
-  Avantages
  - Très adapté aux systèmes expérimentaux
  - Permet l'obtention directe d'un modèle état-espace réduit
-  Limites
  - Sensible au bruit de mesure
  - Nécessite des données riches et persistantes

# Réduction par identification de sous-espace

## Contexte

- La méthode d'identification de sous-espace est principalement utilisée quand on dispose :
  - de données expérimentales (entrées et sorties mesurées)
  - et qu'on souhaite construire un modèle d'état de la forme :

$$\begin{cases} \hat{x}_{k+1} = A_r \hat{x}_k + B_r u_k \\ y_k = C_r \hat{x}_k + D_r u_k \end{cases}$$

- Elle est aussi exploitée pour **réduire un modèle** d'ordre élevé à un modèle d'ordre plus faible, en extrayant uniquement les modes significatifs à partir des données.

# Réduction par identification de sous-espace

## Principe général

- **Collecte des données :**  
On mesure des séquences d'entrée  $u_k$  et de sortie  $y_k$  sur une fenêtre temporelle.
- **Construction de matrices de Hankel :**  
On forme des matrices empilées contenant les données d'entrée et de sortie passées et futures. Ces matrices de Hankel capturent la dynamique du système.
- **Projection sur un sous-espace :**  
Par une méthode basée sur la décomposition en valeurs singulières (SVD), on extrait un sous-espace dominant qui représente l'espace d'états (observable et contrôlable).
- **Estimation des matrices d'état :**  
À partir du sous-espace identifié, on estime les matrices A,B,C,D du modèle d'état.
- **Réduction de l'ordre :**  
La SVD permet de choisir la dimension du modèle en conservant les valeurs singulières significatives, donc un modèle réduit.

# Réduction par identification de sous-espace

## Étape 1. Collecte des données

- On mesure les signaux :
  - Entrées  $u_0, u_1, \dots, u_N$
  - Sorties  $y_0, y_1, \dots, y_N$
  - $N$  la longueur des données
  - $i > r$  l'horizon d'observation (nombre de lignes dans la matrice Hankel)

## Étape 2. Construction des matrices de Hankel

- On construit des **matrices de Hankel** pour l'entrée et la sortie :

$$Y_p = \begin{bmatrix} y_1 & y_2 & \cdots & y_k \\ y_2 & y_3 & \cdots & y_{k+1} \\ \vdots & \vdots & \ddots & \vdots \\ y_i & y_{i+1} & \cdots & y_{i+k-1} \end{bmatrix} \quad U_p = \begin{bmatrix} u_1 & u_2 & \cdots & u_k \\ u_2 & u_3 & \cdots & u_{k+1} \\ \vdots & \vdots & \ddots & \vdots \\ u_i & u_{i+1} & \cdots & u_{i+k-1} \end{bmatrix}$$

avec  $j = N - 2i + 1$

```
[m, N] = size(u); % nb entrées, durée  
[p, Ny] = size(y);
```

```
j = N - 2*i + 1; % longueur des blocs
```

```
% Construction des matrices de Hankel  
Up = hankel_blocks(u, i, j);  
Yp = hankel_blocks(y, i, j);
```

```
Uf = hankel_blocks(u(:, i:N), i, j);  
Yf = hankel_blocks(y(:, i:N), i, j);
```

# Réduction par identification de sous-espace

## Étape 3. Factorisation en valeurs singulières (SVD)

- On forme la matrice de Hankel combinée entrée-sortie :

$$Z = \begin{bmatrix} U_p \\ Y_p \end{bmatrix}$$

- Puis on fait la SVD :

$$Z = U \Sigma V^T$$

- Les **valeurs singulières significatives** permettent de choisir l'ordre du modèle réduit (en regardant leur décroissance).

% Projection orthogonale :  
orthogonaliser Yf par  
rapport à Up  
Z = [Up; Yp]; % matrice de  
données passées

[~, ~, V] = svd(Z, 'econ');

# Réduction par identification de sous-espace


## Étape 4. Estimation de l'espace des états observés

- On garde les  $(m + p)i$  premiers vecteurs de droite de  $V$ , qui forment une base orthonormée de l'espace image de  $Z$ . Ces vecteurs décrivent l'espace généré par les données passées.

$$L = V((m + p)i)$$

- On construit le projecteur orthogonal :

$$\text{Proj} = I - LL^T$$

- qui permet de projeter n'importe quel vecteur orthogonalement à l'espace généré par  $Z$ .
-  Ce projecteur agit sur les colonnes (donc il est à droite dans les multiplications  $Y_f * \text{Proj}$ )

```
[~, ~, V] = svd(Z, 'econ');  
L = V(:, 1:(m + p) * i); % base  
pour projection  
Proj = eye(j) - L * L';
```

# Réduction par identification de sous-espace

On projette les sorties futures  $Y_f$  orthogonalement au passé :

$$Y_f^\perp = Y_f \cdot \text{Proj}$$

Ce qui revient à éliminer la partie de  $Y_f$  qui est corrélée au passé, pour ne garder que l'information nouvelle (due aux états du système).

- Puis on fait la SVD :

$$Y_f^\perp = U \Sigma V^\top$$

- On tronque  $U$ ,  $\Sigma$  et  $V$  à l'ordre  $r$
- On obtient la matrice d'observabilité tronquée :

$$\Gamma = U_r \Sigma_r^{1/2}$$

On obtient le vecteur des états :

$$X = \Sigma_r^{1/2} V_r^T$$

```
Yf_ortho = Yf * Proj;
```

```
[U, S, V] = svd(Yf_ortho, 'econ');
```

```
U1 = U(:,1:r);
```

```
S1 = S(1:r,1:r);
```

```
V1 = V(:,1:r);
```

```
Gamma = U1 * sqrt(S1);
```

```
X = sqrt(S1) * V1';
```

# Réduction par identification de sous-espace

## Identification des matrices du modèle réduit

- On cherche les matrices  $A_r$ ,  $B_r$ ,  $C_r$  et  $D_r$  du **modèle réduit** :

$$\begin{cases} \hat{x}_{k+1} = A_r \hat{x}_k + B_r u_k \\ y_k = C_r \hat{x}_k + D_r u_k \end{cases}$$

- On utilise la méthode des **moindres carrés** :
- Pour estimer  $A_r$  et  $B_r$  on utilise la relation entre les états décalés dans le temps :

$$X_+ = A_r X + B_r U$$

avec  $X_+$  l'état au pas suivant,  $X$  l'état actuel, et  $U$  l'entrée actuelle.

% Estimate the state-space matrices

X1 = X(:, 1:end-1);

X2 = X(:, 2:end);

U1 = u(:, i : i + j - 2);



# Réduction par identification de sous-espace

- En concaténant  $X$  et  $U$  dans une matrice  $H = \begin{bmatrix} X \\ U \end{bmatrix}$ , on pose :

$$X_+ = \begin{bmatrix} A_r & B_r \end{bmatrix} \begin{bmatrix} X \\ U \end{bmatrix} = \begin{bmatrix} A_r & B_r \end{bmatrix} H$$

- et on calcule :

$$\begin{bmatrix} A_r & B_r \end{bmatrix} = X_+ (H^T H)^{-1} H^T$$

```
% Estimate the state-space  
matrices from least square  
H = [X1; U1];  
AB = X2 * pinv(H);  
Ar = AB(:, 1:n);  
Br = AB(:, n+1:end);
```

# Réduction par identification de sous-espace

- Pour estimer  $C_r$  et  $D_r$ , on utilise la relation sortie :

$$Y = C_r X + D_r U$$

Soit :

$$Y = [C_r \quad D_r] \begin{bmatrix} X \\ U \end{bmatrix} = [C_r \quad D_r] H$$

- et on calcule :

$$[C_r \quad D_r] = Y (H^T H)^{-1} H^T$$

```
U1 = u(:, i : i + j - 1);  
CD = y(:, i : i + j - 1) * pinv([X;  
U1]);  
Cr = CD(:, 1:n);  
Dr = CD(:, n+1:end);
```

# Comparaison des méthodes

Méthode	Avantages	Limites
Réduction modale	Intuitive, physique	Linéaire uniquement
POD	Basée données, efficace	Risque de sur-ajustement
Réduction équilibrée	Erreur bornée, stabilité	Calcul coûteux
Sous-espace	Identification directe	Sensible au bruit