*République Algérienne Démocratique et Populaire*

*Ministère de l'enseignement Supérieur et de la Recherche scientifique*

*Université EchahidHamma Lakhdar – EL OUED*



**Faculté de la Technologie**

**Département de Génie Electrique**

# Polycopié destiné aux étudiants de Master I-S-2

# en génie électrique

## Matière:

## Modélisation et identification des systèmes électriques

**Réalisé par :**

**Dr. ALLAG Meriem**

**Septembre :2024**

**Resumée:** Ce cours fournit une introduction approfondie à la modélisation, à l'identification et à l'observation des systèmes en ingénierie. Les étudiants apprennent à modéliser et analyser des systèmes complexes en explorant les concepts fondamentaux de la modélisation des systèmes, notamment les types de modèles et les techniques de simulation. L'accent est mis sur la modélisation mathématique à travers l'utilisation de diagrammes en blocs, qui représentent les relations internes et externes d'un système. Le cours couvre également la modélisation des systèmes électriques, avec un focus sur les composants passifs et actifs ainsi que les circuits de base, en introduisant des outils avancés comme les Bond Graphs et les Graphes d'Information Causale (GIC) pour analyser les dynamiques du système.

L'identification des systèmes est un aspect central du cours, où les étudiants développent des modèles à partir de données réelles. Ils abordent des structures de modèles comme AR, ARMA et ARMAX, et apprennent à analyser les réponses temporelles et fréquentielles des systèmes du premier et deuxième ordre. Le cours présente des méthodes graphiques (ex. : Strejc, Broïda) et numériques (algorithmes récursifs et non récursifs) pour affiner les modèles.

Enfin, le cours se termine par l'enseignement des méthodes d'estimation et d'observation pour le suivi en temps réel des systèmes, incluant des techniques comme l'observateur de Luenberger et le filtre de Kalman. Ces modules permettent aux étudiants de maîtriser les techniques de modélisation, d'identification et d'observation des systèmes pour résoudre des défis pratiques en ingénierie.

**Mot de clé : BG** : Bond graph, **GIC** :Graphe informationnel causales , **SBPA** : Séquences Binaires Pseudo-Aléatoires, **ARMA** :Auto-Régressifs avec Moyenne Mobile

# Contenu de la matière

**Chapitre 1 : Systèmes et expériences (01 semaine)**

Généralités, types de modèles, modèles et simulation, comment obtenir un modèle

**Chapitre 2 : Modèle mathématique (02 semaines)**

Schéma bloc d'un système, variables caractéristiques, représentations interne et externe d'un système

**Chapitre 3 : Modélisation des systèmes électriques (02 semaines)**

Modélisation d'un composant passif, d'un composant actif et des circuits électriques de base, Exemples d'applications.

**Chapitre 4 : Outils de modélisation(02 semaines)**

Bond graph (BG) ou Graphe informationnel causales (GIC)) (Application aux circuits électriques

**Chapitre 5 : Généralités sur l'identification (02 semaines)**

- Définitions, étapes, génération SBPA, choix de la structure du modèle (AR, ARMA, ARMAX..);

- Rappel des méthodes de base en Automatique : Réponse temporelle d'un système, Approche fréquentielle, Identification directe à partir des réponses temporelle et fréquentielle des systèmes $1^{er}$ ordre et $2^{ème}$ ordre, méthode de variable instrumentale ;

- Principe d'ajustement du modèle : Modèle linéaire par rapport aux paramètres, Minimisation du critère d'ajustement et calcul de la solution optimale.

**Chapitre 6 : Méthodes d'identification graphiques (02 semaines)**

Méthode de Strejc, méthode de Broïda…

**Chapitre 7 : Méthodes d'identification numériques (02 semaines)**

Méthodes récursives, méthode non récursives.

**Chapitre 8 : Estimation et observation (02 Semaines)**

- Estimation des systèmes électriques (exemple : Estimateur de Gopinath)
- Observation déterministe (Observateur de Luenberger)
- Observateurs Non-déterministes ou stochastiques (Filtre de Kalman)

# Chapitre I

# Généralités, types de modèles, modèles et simulation, comment obtenir un modèle

**Systèmes et Expériences : Notions Scientifiques et Exemples Pratiques**

**Introduction**

L'étude des systèmes est fondamentale en sciences et en ingénierie car elle permet de comprendre, analyser, et prédire le comportement des divers phénomènes complexes qui nous entourent. Un système peut être un ensemble de composants physiques ou conceptuels qui interagissent ensemble. En modélisant et en simulant ces systèmes, les scientifiques et ingénieurs peuvent étudier leur comportement sous diverses conditions, optimiser leur fonctionnement et prévoir leur évolution.

Cette section propose une exploration détaillée des systèmes et expériences, incluant des généralités sur les systèmes, les types de modèles utilisés, la relation entre modèles et simulation, et des méthodes pratiques pour obtenir un modèle. Cinq cas pratiques serviront d'exemples pour illustrer le processus de modélisation étape par étape jusqu'à l'obtention des modèles désirés.

**1. Généralités sur les Systèmes**

**1.1 Définition d'un Système**

Un **système** est défini comme un ensemble d'éléments ou de composants interconnectés qui fonctionnent ensemble pour atteindre un objectif commun ou produire un certain comportement. Chaque système a des limites qui le séparent de son environnement, des entrées qui influencent le système, et des sorties qui résultent de ses processus internes.

**Exemples de systèmes** :

- **Systèmes physiques** : Une voiture, un réacteur chimique, une éolienne.
- **Systèmes biologiques** : Le corps humain, un écosystème.
- **Systèmes sociaux** : Une organisation, une communauté.
- **Systèmes économiques** : Le marché boursier, une économie nationale.

**1.2 Composants d'un Système**

Les systèmes comportent plusieurs composants essentiels :

- **Entrées** : Ce sont les éléments d'information, d'énergie ou de matière qui entrent dans le système depuis l'extérieur.
- **Sorties** : Résultats produits par le système, qui peuvent être des informations, de l'énergie ou de la matière.
- **Processus** : Les actions ou opérations internes du système qui transforment les entrées en sorties.
- **Frontières** : Les limites qui délimitent le système et le distinguent de son environnement.

- **Feedback (rétroaction)** : Les processus où une partie de la sortie du système est réintroduite en tant qu'entrée pour influencer son comportement futur.

### 1.3 Types de Systèmes

- **Systèmes ouverts** : Interagissent avec leur environnement. Exemples : une plante (échange de gaz, lumière, nutriments).
- **Systèmes fermés** : N'ont pas d'interaction avec l'environnement. Exemples : un thermos isolé (aucun échange de chaleur ou matière).
- **Systèmes dynamiques** : Changent au fil du temps. Exemples : un marché boursier, une épidémie.
- **Systèmes statiques** : Ne changent pas avec le temps. Exemples : la structure d'un bâtiment.

## 2. Types de Modèles

Un modèle est une représentation simplifiée d'un système réel ou hypothétique, utilisée pour comprendre, analyser et prédire son comportement. Les modèles peuvent être classés de plusieurs façons, selon leur nature et leur fonction.

### 2.1 Modèles Physiques

Les **modèles physiques** sont des répliques réduites ou agrandies de systèmes réels. Ils sont utilisés pour des essais pratiques ou des visualisations.

- **Exemple** : Une maquette d'avion testée en soufflerie pour étudier l'aérodynamique.

### 2.2 Modèles Mathématiques

Les **modèles mathématiques** utilisent des équations mathématiques pour décrire les relations entre les différentes variables d'un système. Ils sont largement utilisés en physique, ingénierie, économie, etc.

- **Exemple** : L'équation de la chaleur pour modéliser la diffusion thermique dans un matériau solide.

### 2.3 Modèles Statistiques

Les **modèles statistiques** analysent et prédisent des résultats basés sur des données empiriques. Ils sont couramment utilisés pour traiter des données incertaines ou variables.

- **Exemple** : Modèle de régression linéaire pour prédire les ventes en fonction du prix.

### 2.4 Modèles Conceptuels

Les **modèles conceptuels** sont des représentations abstraites qui décrivent les relations et processus dans un système sans entrer dans des détails quantitatifs précis.

- **Exemple** : Diagramme de flux illustrant les processus de production dans une usine.

### 2.5 Modèles Logiques

Les **modèles logiques** utilisent des règles de logique pour représenter le comportement des systèmes. Ils sont souvent utilisés en informatique et en intelligence artificielle.

- **Exemple** : Arbre de décision pour diagnostiquer des défaillances dans un système.

### 3. Modèles et Simulation

#### 3.1 Définition de la Simulation

La **simulation** est le processus d'utilisation de modèles pour étudier le comportement d'un système sous différentes conditions. Elle permet d'analyser les effets des changements dans les variables d'entrée, de prédire les résultats futurs et d'optimiser les performances sans avoir besoin d'expérimenter sur le système réel.

#### 3.2 Types de Simulations

- **Simulation déterministe** : Les mêmes conditions initiales produisent toujours le même résultat. Exemples : simulation d'un circuit électrique.
- **Simulation stochastique** : Comporte des éléments aléatoires, produisant des résultats différents pour les mêmes conditions initiales. Exemples : simulation de phénomènes météorologiques.
- **Simulation dynamique** : Étudie l'évolution des systèmes dans le temps. Exemples : simulation de la propagation d'une épidémie.
- **Simulation statique** : Examine les systèmes à un instant donné. Exemples : analyse des files d'attente dans un système de service.

#### 3.3 Objectifs de la Simulation

- **Analyse de performance** : Comprendre comment un système fonctionne sous diverses conditions.
- **Optimisation** : Trouver les paramètres ou configurations qui maximisent l'efficacité ou minimisent les coûts.
- **Prévision** : Anticiper le comportement futur du système.
- **Formation** : Utiliser des simulations pour former le personnel dans des environnements réalistes.

#### 3.4 Étapes du Processus de Simulation

1. **Définition du problème** : Identifier les objectifs et les questions que la simulation doit répondre.
2. **Construction du modèle** : Développer un modèle qui représente fidèlement le système réel.
3. **Validation du modèle** : S'assurer que le modèle prédit correctement le comportement du système réel.
4. **Exécution de la simulation** : Exécuter le modèle sous différentes conditions pour analyser le comportement du système.
5. **Analyse des résultats** : Interpréter les résultats pour prendre des décisions informées.
6. **Implémentation des décisions** : Utiliser les résultats pour améliorer ou modifier le système réel.

### 4. Comment Obtenir un Modèle

#### 4.1 Identification du Système

- **Objectifs de modélisation** : Pourquoi modéliser ce système ? Quels sont les résultats attendus ?
- **Délimitation du système** : Quelles sont les frontières du système ? Quelles sont les variables d'entrée et de sortie ?

### 4.2 Collecte de Données

La collecte de données est cruciale pour construire et valider un modèle. Les données peuvent provenir d'observations directes, de capteurs, d'expériences contrôlées ou de bases de données historiques.

- **Sources de données** : Observations, capteurs, expérimentations, bases de données.
- **Types de données** : Quantitatives (numériques) et qualitatives (catégoriques).

### 4.3 Choix du Type de Modèle

Le choix du modèle dépend de la nature du système, des objectifs de la modélisation et des données disponibles. Les modèles peuvent être empiriques, basés sur des données observées, ou théoriques, basés sur des principes fondamentaux.

### 4.4 Construction du Modèle

- **Formulation des équations** : Développer des équations pour décrire les relations entre les variables du système.
- **Détermination des paramètres** : Identifier les valeurs des paramètres qui influencent le modèle.
- **Programmation** : Implémenter le modèle dans un logiciel de simulation (Matlab, Python, etc.).

### 4.5 Validation et Vérification

- **Validation** : Comparer les prédictions du modèle avec des données réelles.
- **Vérification** : S'assurer que le modèle est correctement programmé et fonctionne comme prévu.

### 4.6 Analyse de Sensibilité et d'Incertitude

- **Analyse de sensibilité** : Déterminer comment les variations des paramètres affectent les résultats du modèle.
- **Analyse d'incertitude** : Évaluer l'impact des incertitudes sur les données d'entrée sur les résultats du modèle.

## 5. Exercices Pratiques : Cinq Cas de Systèmes

### Cas 1 : Modélisation d'un Réacteur Chimique

**Objectif**: Modéliser un réacteur à cuve agitée continue (CSTR) pour maximiser la production de produits.

**Étapes**:

1. **Identification du système**: Le réacteur est un système chimique où les réactifs sont continuellement introduits et les produits retirés.

2. **Collecte de données**: Mesures des concentrations de réactifs et de produits, température, pression.
3. **Choix du modèle**: Modèle mathématique basé sur des équations de bilan de masse et d'énergie.
4. **Construction du modèle**:

$$\frac{dC_A}{dt} = \frac{F_{\text{in}}}{V}(C_{A,\text{in}} - C_A) - kC_A$$
$$\frac{dT}{dt} = \frac{F_{\text{in}}}{V}(T_{\text{in}} - T) + \frac{-\Delta H}{\rho C_p}kC_A$$

5. **Validation et simulation**: Comparaison avec les données expérimentales, simulation sous différentes conditions.

*Programme Matlab:*

```
matlab
Copier le code
% Paramètres du réacteur
F_in = 1; % Débit d'entrée (L/s)
V = 100; % Volume du réacteur (L)
k = 0.1; % Constante de vitesse de réaction (1/s)
C_A_in = 1; % Concentration d'entrée (mol/L)
T_in = 300; % Température d'entrée (K)
delta_H = -50000; % Chaleur de réaction (J/mol)
rho = 1000; % Densité (kg/L)
Cp = 4.18; % Capacité calorifique (J/(g·K))

% Initialisation
C_A = 0; % Concentration initiale (mol/L)
T = 300; % Température initiale (K)
t = linspace(0, 500, 100); % Temps (s)

% Fonction pour les EDO
function dXdt = cstr_odes(t, X)
    C_A = X(1);
    T = X(2);
    dC_Adt = (F_in/V)*(C_A_in - C_A) - k*C_A;
    dTdt = (F_in/V)*(T_in - T) + (-delta_H/(rho*Cp))*k*C_A;
    dXdt = [dC_Adt; dTdt];
end

% Simulation
[t, X] = ode45(@(t, X) cstr_odes(t, X), t, [C_A; T]);

% Affichage
plot(t, X(:,1), t, X(:,2))
xlabel('Temps (s)')
ylabel('Concentration (mol/L) et Température (K)')
legend('Concentration de A', 'Température')
title('Simulation d\'un CSTR')
```

**Cas 2 : Modélisation d'un Écosystème Marin**

**Objectif**: Modéliser les interactions entre phytoplancton et zooplancton pour étudier la dynamique des populations.

**Étapes**:

1. **Identification du système**: Un écosystème marin avec des populations de phytoplancton (proies) et de zooplancton (prédateurs).
2. **Collecte de données**: Données sur la biomasse de phytoplancton et de zooplancton, taux de croissance.
3. **Choix du modèle**: Modèle Lotka-Volterra modifié.
4. **Construction du modèle**:

$$\frac{dP}{dt} = rP - aPQ$$
$$\frac{dZ}{dt} = bPQ - mZ$$

5. **Validation et simulation**: Comparaison avec des données de terrain, simulation de scénarios de perturbation (pollution, changement climatique).

**Programme Matlab:**

```
% Paramètres
r = 0.1; % Taux de croissance du phytoplancton
a = 0.02; % Taux de prédation
b = 0.01; % Taux de croissance du zooplancton
m = 0.1; % Taux de mortalité du zooplancton

% Initialisation
P = 40; % Biomasse initiale de phytoplancton
Z = 9; % Biomasse initiale de zooplancton
t = linspace(0, 100, 500); % Temps

% Fonction pour les EDO
function dXdt = marine_ecosystem(t, X)
    P = X(1);
    Z = X(2);
    dPdt = r * P - a * P * Z;
    dZdt = b * P * Z - m * Z;
    dXdt = [dPdt; dZdt];
end

% Simulation
[t, X] = ode45(@(t, X) marine_ecosystem(t, X), t, [P; Z]);

% Affichage
plot(t, X(:,1), t, X(:,2))
xlabel('Temps')
ylabel('Biomasse')
legend('Phytoplancton', 'Zooplancton')
title('Modèle d\'écosystème marin')
```

**Cas 3 : Modélisation d'un Système de Contrôle de Température**

**Objectif**: Modéliser un système de contrôle de température pour une pièce.

**Étapes**:

1. **Identification du système**: Un système de chauffage pour maintenir une température constante dans une pièce.
2. **Collecte de données**: Mesures de température, puissance de chauffage, température extérieure.
3. **Choix du modèle**: Modèle de premier ordre basé sur l'équation de la chaleur.
4. **Construction du modèle**:

$$\frac{dT}{dt} = \frac{1}{\tau}(T_{\text{set}} - T) + \frac{1}{\tau_{\text{ext}}}(T_{\text{ext}} - T)$$

5. **Validation et simulation**: Test avec des données de températures réelles, ajustement des paramètres.

**Programme Matlab:**

```
% Paramètres

tau = 10; % Constante de temps (s)
T_set = 22; % Température cible (°C)
T_ext = 5; % Température extérieure (°C)
t = linspace(0, 500, 100); % Temps (s)
% Initialisation
T = 15; % Température initiale (°C)
% Fonction pour les EDO
function dTdt = temperature_control(t, T)
    dTdt = (1/tau) * (T_set - T) + (1/tau) * (T_ext - T);
end

% Simulation
[T_sim, T] = ode45(@(t, T) temperature_control(t, T), t, T);

% Affichage
plot(T_sim, T)
xlabel('Temps (s)')
ylabel('Température (°C)')
title('Contrôle de température')
```

**Cas 4 : Modélisation d'un Système de Transport Urbain**

**Objectif**: Modéliser un système de transport urbain pour optimiser le flux de circulation.

**Étapes**:

1. **Identification du système**: Un réseau de transport urbain comprenant des routes, des feux de circulation et des intersections.
2. **Collecte de données**: Comptages de trafic, temps de parcours, temps d'attente aux intersections.
3. **Choix du modèle**: Modèle de circulation basé sur des équations de continuité et des modèles de file d'attente.
4. **Construction du modèle**:

$$\frac{dQ}{dt} = \lambda - \mu Q$$

5. **Validation et simulation**: Simulation des flux de circulation, optimisation des temps de feux de circulation.

**Programme Matlab*:***

```
% Paramètres
lambda = 0.2; % Taux d'arrivée (véhicules par seconde)
mu = 0.3; % Taux de service (véhicules par seconde)
t = linspace(0, 500, 100); % Temps (s)

% Initialisation
Q = 0; % Nombre initial de véhicules

% Fonction pour les EDO
function dQdt = traffic_flow(t, Q)
    dQdt = lambda - mu * Q;
end

% Simulation
[t, Q] = ode45(@(t, Q) traffic_flow(t, Q), t, Q);

% Affichage
plot(t, Q)
xlabel('Temps (s)')
ylabel('Nombre de véhicules')
title('Flux de circulation')
```

**Cas 5 : Modélisation d'une Chaîne de Production Industrielle**

**Objectif**: Modéliser une chaîne de production pour améliorer l'efficacité et minimiser les temps d'arrêt.

**Étapes**:

1. **Identification du système**: Une chaîne de production avec plusieurs étapes (assemblage, inspection, emballage).
2. **Collecte de données**: Temps de cycle, taux de défauts, capacité de production.
3. **Choix du modèle**: Modèle de file d'attente avec plusieurs serveurs (M/M/1, M/M/c).
4. **Construction du modèle**:

$$L = \frac{\lambda}{\mu - \lambda}$$

où $L$ est le nombre moyen de pièces en attente, $\lambda$ est le taux d'arrivée, et $\mu$ est le taux de service.

5. **Validation et simulation**: Test avec des données de production réelles, optimisation des taux de production.

**Programme Matlab:**

```
% Paramètres

lambda = 5; % Taux d'arrivée (pièces par minute)

mu = 7; % Taux de service (pièces par minute)
t = linspace(0, 100, 100); % Temps (minutes)
% Initialisation
L = 0; % Nombre initial de pièces en attente
% Fonction pour les EDO
function dLdt = production_queue(t, L)
    dLdt = lambda - mu * L;
end
% Simulation
[t, L] = ode45(@(t, L) production_queue(t, L), t, L);
% Affichage
plot(t, L)
xlabel('Temps (minutes)')
ylabel('Nombre de pièces en attente')
title('Chaîne de production')
```

**Conclusion :**
La modélisation et la simulation sont des outils essentiels pour comprendre, analyser et optimiser les systèmes complexes. En suivant des étapes claires pour identifier, construire, valider et simuler des modèles, les ingénieurs et scientifiques peuvent prévoir le comportement des systèmes sous diverses conditions et prendre des décisions éclairées pour améliorer les performances. Les exemples pratiques fournis illustrent comment appliquer ces concepts à des systèmes réels dans divers domaines.

## Applications : Comment Obtenir un Modèle

L'obtention d'un modèle pour un système est une démarche structurée qui inclut plusieurs étapes allant de la compréhension du système à la validation du modèle. Cette section propose cinq exercices pratiques, chacun illustrant le processus de modélisation d'un système, de la collecte des données à la construction et validation du modèle. Chaque exercice inclut des étapes détaillées pour identifier le système, collecter les données, choisir le type de modèle, et construire le modèle.

**Exercice 1 : Modélisation d'un Système de Contrôle de Température pour une Serre**

**1. Identification du Système**

- **Objectifs** : Modéliser le système de contrôle de température d'une serre pour maintenir des conditions optimales de croissance des plantes. Les résultats attendus incluent la prédiction des besoins en chauffage/refroidissement en fonction des variations de température externe.
- **Délimitation du Système** : Le système inclut la serre, les capteurs de température internes et externes, le chauffage et le refroidissement. Les variables d'entrée sont la température externe et l'ensoleillement, les sorties sont la température interne et la consommation énergétique.

**2. Collecte de Données**

- **Sources de données** : Capteurs de température installés à l'intérieur et à l'extérieur de la serre, historique des données météorologiques, données sur la consommation énergétique du système de chauffage/refroidissement.
- **Types de données** : Quantitatives, incluant des séries temporelles de température, d'ensoleillement et de consommation énergétique.

**3. Choix du Type de Modèle**

- **Type de modèle** : Modèle empirique pour ajuster les données historiques et prédire les besoins énergétiques. Un modèle théorique basé sur les lois de transfert de chaleur pourrait aussi être utilisé pour capturer les aspects physiques du système.

**4. Construction du Modèle**

- **Élaboration d'équations** : Utiliser l'équation de transfert de chaleur :

$$\frac{dT_{\text{int}}}{dt} = \frac{1}{\tau}(T_{\text{ext}} - T_{\text{int}}) + Q_{\text{chauffage}} - Q_{\text{refroidissement}}$$

où $T_{\text{int}}$ est la température interne, $T_{\text{ext}}$ est la température externe, $Q_{\text{chauffage}}$ et $Q_{\text{refroidissement}}$ sont les flux de chaleur.

- **Identification des paramètres** : Paramètres comme τ\tauτ (constante de temps), efficacité du chauffage et du refroidissement.

- **Programmation** : Implémentation dans Matlab pour simuler la réponse du système sous différentes conditions.

**Programme Matlab:**

```
% Paramètres
tau = 10; % Constante de temps (s)
T_ext = 5; % Température extérieure (°C)
Q_chauffage = 2; % Flux de chauffage (°C/s)
Q_refroidissement = 0; % Flux de refroidissement (°C/s)
t = linspace(0, 500, 100); % Temps (s)

% Initialisation
T_int = 15; % Température initiale interne (°C)

% Fonction pour les EDO
function dTdt = controle_temperature(t, T)
    dTdt = (1/tau) * (T_ext - T) + Q_chauffage - Q_refroidissement;
end

% Simulation
[T_sim, T] = ode45(@(t, T) controle_temperature(t, T), t, T_int);

% Affichage
plot(T_sim, T)
xlabel('Temps (s)')
ylabel('Température interne (°C)')
title('Contrôle de température dans une serre')
```

**Exercice 2 : Modélisation de la Population de Poissons dans un Lac**

**1. Identification du Système**

- **Objectifs** : Modéliser la dynamique de la population de poissons pour évaluer l'impact de la pêche et des conditions environnementales sur la population. Les résultats attendus incluent la prédiction de la population future et des recommandations de gestion.
- **Délimitation du Système** : Le système inclut le lac, la population de poissons, les pêcheurs, et les conditions environnementales comme la température de l'eau et les niveaux de pollution. Les variables d'entrée sont les taux de pêche, la température de l'eau, et la qualité de l'eau; la sortie est la taille de la population de poissons.

**2. Collecte de Données**

- **Sources de données** : Enquêtes de pêche, capteurs de qualité de l'eau, données historiques sur la température de l'eau.
- **Types de données** : Quantitatives, incluant des séries temporelles de taille de population, taux de pêche, et mesures de qualité de l'eau.

**3. Choix du Type de Modèle**

- **Type de modèle** : Modèle théorique basé sur les équations de croissance de population comme le modèle logistique, ajusté aux données empiriques.

**4. Construction du Modèle**

- **Élaboration d'équations** :

$$\frac{dN}{dt} = rN\left(1 - \frac{N}{K}\right) - hN$$

où $N$ est la population de poissons, $r$ est le taux de croissance, $K$ est la capacité de charge du lac, et $h$ est le taux de pêche.

.

- **Identification des paramètres** :

Taux de croissance $r$, capacité de charge $K$, taux de pêche $h$.

- **Programmation** : Simulation dans Matlab pour estimer la taille de la population sous différents scénarios de pêche.

**Programme Matlab:**

```
% Paramètres
r = 0.1; % Taux de croissance de la population
K = 1000; % Capacité de charge du lac
h = 0.05; % Taux de pêche
t = linspace(0, 100, 100); % Temps (années)

% Initialisation
N = 500; % Population initiale de poissons

% Fonction pour les EDO
function dNdt = population_poissons(t, N)
    dNdt = r * N * (1 - N / K) - h * N;
end

% Simulation
[t, N] = ode45(@(t, N) population_poissons(t, N), t, N);

% Affichage
plot(t, N)
xlabel('Temps (années)')
ylabel('Population de poissons')
title('Dynamique de la population de poissons dans un lac')
```

**Exercice 3 : Modélisation de la Croissance d'une Culture de Blé**

**1. Identification du Système**

- **Objectifs** : Modéliser la croissance d'une culture de blé pour optimiser les pratiques agricoles et maximiser le rendement. Les résultats attendus incluent des prévisions de rendement en fonction de différents intrants.
- **Délimitation du Système** : Le système inclut le champ de blé, les intrants agricoles (eau, engrais, pesticides), et les conditions environnementales (température, précipitations). Les variables d'entrée sont les quantités d'intrants et les conditions météorologiques; la sortie est le rendement de la culture.

**2. Collecte de Données**

- **Sources de données** : Données historiques de rendement, mesures de conditions météorologiques, quantités d'intrants utilisés.
- **Types de données** : Quantitatives, incluant des séries temporelles de rendement, quantités d'eau et d'engrais appliquées, conditions météorologiques.

**3. Choix du Type de Modèle**

- **Type de modèle** : Modèle empirique basé sur l'ajustement des données de rendement aux conditions météorologiques et aux quantités d'intrants.

**4. Construction du Modèle**

- **Élaboration d'équations** : Utiliser une relation empirique entre le rendement $Y$, l'eau $W$, et les engrais $F$ :

$$Y = aW + bF + cT + d$$

où $a$, $b$, $c$, et $d$ sont des coefficients déterminés par régression.

- **Identification des paramètres** : Coefficients $a$, $b$, $c$, et $d$ basés sur des données historiques.
- **Programmation** : Utiliser Matlab pour ajuster les coefficients et simuler les rendements sous différents scénarios.

**Programme Matlab:**

```
% Données historiques
W = [10, 20, 30, 40, 50]; % Quantité d'eau (mm)
F = [50, 100, 150, 200, 250]; % Quantité d'engrais (kg/ha)
T = [20, 22, 24, 26, 28]; % Température moyenne (°C)
Y = [2000, 3000, 3500, 4000, 4500]; % Rendement (kg/ha)

% Ajustement du modèle empirique
X = [W' F' T'];
b = regress(Y', X);

% Simulation pour différentes conditions
W_sim = 30; % mm
F_sim = 150; % kg/ha
T_sim = 25; % °C
Y_sim = b(1)*W_sim + b(2)*F_sim + b(3)*T_sim;

% Affichage du rendement simulé
fprintf('Rendement simulé: %.2f kg/ha\n', Y_sim);
```

**Exercice 4 : Modélisation de la Décharge d'une Batterie**

**1. Identification du Système**

- **Objectifs** : Modéliser le comportement de décharge d'une batterie pour optimiser son usage dans des applications de stockage d'énergie. Les résultats attendus incluent des prédictions de la durée de vie de la batterie et de sa capacité de décharge sous différentes conditions.
- **Délimitation du Système** : Le système inclut la batterie, les charges connectées, et les conditions de température. Les variables d'entrée sont le courant de décharge et la température; la sortie est la tension de la batterie.

**2. Collecte de Données**

- **Sources de données** : Mesures de la tension de la batterie, courant de décharge, température ambiante.
- **Types de données** : Quantitatives, incluant des séries temporelles de tension, courant, et température.

**3. Choix du Type de Modèle**

- **Type de modèle** : Modèle empirique basé sur les courbes de décharge caractéristiques de la batterie. Un modèle théorique basé sur les équations de Nernst et Butler-Volmer pourrait aussi être envisagé.

**4. Construction du Modèle**

- **Élaboration d'équations** : Utiliser une équation de décharge empirique :

$$V(t) = V_0 - k \cdot I \cdot t$$

où $V(t)$ est la tension, $V_0$ est la tension initiale, $I$ est le courant de décharge, et $k$ est une constante.

- **Identification des paramètres** : Tension initiale $V_0$, constante $k$.
- **Programmation** : Simuler la décharge sous différents courants et températures.

.**Programme Matlab**:

```
% Paramètres
V_0 = 12; % Tension initiale (V)
k = 0.02; % Constante de décharge
I = 2; % Courant de décharge (A)
t = linspace(0, 5000, 100); % Temps (s)

% Fonction de décharge
V = V_0 - k * I * t;

% Affichage
plot(t, V)
xlabel('Temps (s)')
ylabel('Tension (V)')
title('Décharge de la batterie')
```

**Exercice 5 : Modélisation de la Croissance Bactérienne**

**1. Identification du Système**

- **Objectifs** : Modéliser la croissance d'une culture bactérienne en laboratoire pour optimiser les conditions de culture. Les résultats attendus incluent des prévisions de la densité cellulaire en fonction du temps et des conditions de croissance.
- **Délimitation du Système** : Le système inclut la culture bactérienne, les nutriments, la température, et le pH. Les variables d'entrée sont les concentrations en nutriments, la température, et le pH; la sortie est la densité cellulaire.

**2. Collecte de Données**

- **Sources de données** : Mesures de densité cellulaire, concentrations de nutriments, température, pH.
- **Types de données** : Quantitatives, incluant des séries temporelles de densité cellulaire et conditions de culture.

**3. Choix du Type de Modèle**

- **Type de modèle** : Modèle théorique basé sur l'équation de croissance exponentielle, ajusté aux données empiriques.

**4. Construction du Modèle**

- **Élaboration d'équations** : Utiliser l'équation de croissance bactérienne :

$$\frac{dN}{dt} = \mu N$$

où $N$ est la densité cellulaire et $\mu$ est le taux de croissance.

- **Identification des paramètres** : Taux de croissance $\mu$.
- **Programmation** : Simulation de la croissance sous différentes conditions de nutriments, température, et pH.

**Programme Matlab**:

```
% Paramètres
mu = 0.5; % Taux de croissance (1/h)
N0 = 1; % Densité cellulaire initiale (10^6 cellules/mL)
t = linspace(0, 10, 100); % Temps (heures)

% Fonction de croissance
N = N0 * exp(mu * t);

% Affichage
plot(t, N)
xlabel('Temps (heures)')
ylabel('Densité cellulaire (10^6 cellules/mL)')
title('Croissance bactérienne')
```

**Conclusion**

Ces exercices pratiques montrent comment les principes de modélisation peuvent être appliqués à une variété de systèmes réels, allant de la biologie à l'ingénierie. En suivant des étapes systématiques pour identifier, construire et valider des modèles, il est possible de comprendre et de prédire le comportement des systèmes complexes. Ces approches permettent aux scientifiques et ingénieurs de prendre des décisions basées sur des modèles robustes, augmentant ainsi l'efficacité et la précision des systèmes étudiés.

# Chapitre 2

# Modèle Mathématique (02 Semaines)

**Introduction**

Les modèles mathématiques sont essentiels pour analyser et comprendre les systèmes complexes dans divers domaines, tels que l'ingénierie, la physique, l'économie et la biologie. Ils permettent de représenter un système à l'aide d'équations et de relations mathématiques pour analyser son comportement, prévoir ses réponses sous différentes conditions et concevoir des stratégies de contrôle et d'optimisation.

Dans ce chapitre, nous explorerons les bases de la modélisation mathématique, en nous concentrant sur le schéma bloc d'un système, les variables caractéristiques, et les représentations interne et externe des systèmes. Nous illustrerons ces concepts à travers des exemples détaillés et des exercices pratiques, accompagnés de solutions et de programmes Matlab.

## 1. Schéma Bloc d'un Système

### 1.1 Définition du Schéma Bloc

Un **schéma bloc** est une représentation graphique d'un système qui montre les différentes composantes du système, leurs interactions, et le flux d'information entre elles. Chaque bloc représente une fonction ou une opération spécifique du système, et les flèches indiquent les directions des flux de données ou des signaux.

### 1.2 Composantes d'un Schéma Bloc

- **Blocs** : Représentent les sous-systèmes ou les fonctions spécifiques qui transforment les entrées en sorties.
- **Flèches** : Indiquent le flux de données ou de signaux entre les blocs.
- **Entrées** : Variables ou signaux qui pénètrent dans le système.
- **Sorties** : Résultats ou signaux générés par le système.

### 1.3 Exemple de Schéma Bloc

Prenons l'exemple d'un système de contrôle de température pour un réacteur chimique. Le schéma bloc de ce système pourrait inclure les éléments suivants :

1. **Entrée** : Température de consigne (T_set), température externe (T_ext).
2. **Bloc de contrôle** : Compare la température actuelle avec la température de consigne et génère un signal de contrôle.
3. **Actuateur** : Module de chauffage ou de refroidissement qui ajuste la température du réacteur.
4. **Réacteur** : Contient le mélange chimique dont la température est contrôlée.
5. **Sortie** : Température réelle du réacteur (T_real).

```
css
 [T_set] → [Bloc de contrôle] → [Actuateur] → [Réacteur] → [T_real]
                                    ↑                          ↓
                                [T_ext] ←--------------------← [Capteur]
```

### 1.4 Importance des Schémas Bloc

Les schémas bloc permettent de visualiser et d'analyser la structure d'un système, facilitant la compréhension des interactions entre les différentes composantes. Ils sont essentiels pour concevoir des stratégies de contrôle et pour diagnostiquer les problèmes potentiels.

## 2. Variables Caractéristiques

### 2.1 Définition

Les **variables caractéristiques** sont des variables qui décrivent l'état d'un système et déterminent son comportement. Elles peuvent être classées en deux catégories principales :

- **Variables d'entrée** : Variables qui influencent le système de l'extérieur (e.g., température de consigne, tension appliquée).
- **Variables de sortie** : Variables mesurables qui résultent des opérations internes du système (e.g., température réelle, vitesse de rotation).

### 2.2 Exemple de Variables Caractéristiques

Pour le système de contrôle de température du réacteur chimique :

- **Variables d'entrée** : Température de consigne (T_set), température externe (T_ext).
- **Variables de sortie** : Température réelle du réacteur (T_real).
- **Variables d'état** : Variables internes telles que la concentration des réactifs, l'énergie thermique stockée.

### 2.3 Utilisation des Variables Caractéristiques

Les variables caractéristiques permettent de formuler les équations qui décrivent le comportement dynamique du système. Elles sont cruciales pour la conception des modèles mathématiques et des stratégies de contrôle.

## 3. Représentations Interne et Externe d'un Système

### 3.1 Représentation Externe

La **représentation externe** d'un système se concentre sur les relations entre les variables d'entrée et de sortie, sans se préoccuper des détails internes du système. Cette approche est utile pour analyser le comportement global du système et est souvent utilisée dans les systèmes de contrôle.

**Exemple : Modèle de Boîte Noire**

Un modèle de boîte noire est une approche courante pour la représentation externe, où le système est traité comme une boîte opaque. Les relations entre les entrées et les sorties sont modélisées sans connaître les processus internes.

Exemple Mathématique :

$$y(t) = G(s)u(t)$$

où $G(s)$ est la fonction de transfert du système, $u(t)$ est l'entrée, et $y(t)$ est la sortie.

### 3.2 Représentation Interne

La **représentation interne** décrit les processus internes du système, incluant les variables d'état et les équations qui gouvernent l'évolution du système. Cette approche est essentielle pour concevoir des modèles précis et détaillés, et pour comprendre le comportement du système à un niveau plus profond.

**Exemple : Modèle d'État**

Un modèle d'état utilise des équations différentielles pour représenter les relations internes entre les variables d'état, les entrées, et les sorties.

Exemple Mathématique :

$$\frac{dx}{dt} = Ax + Bu$$

$$y = Cx + Du$$

où $x$ est le vecteur d'état, $u$ est le vecteur d'entrée, $y$ est le vecteur de sortie, et $A$, $B$, $C$, $D$ sont des matrices qui définissent les relations entre ces vecteurs.

## 4. Exercices Pratiques : Modélisation et Simulation avec Matlab

### Exercice 1 : Système de Contrôle de Température

**Objectif** : Modéliser et simuler un système de contrôle de température utilisant un chauffage et un

1. Schéma Bloc :
   - Entrée : Température de consigne (T_set), température externe (T_ext).
   - Bloc de contrôle : Comparateur de température.
   - Actuateur : Chauffage/refroidissement.
   - Réacteur : Chambre de température.
   - Sortie : Température réelle (T_real).
2. Variables Caractéristiques :
   - $T_{\text{set}}$ : Température de consigne.
   - $T_{\text{ext}}$ : Température externe.

3. Modèle Mathématique :

$$\frac{dT_{\text{real}}}{dt} = \alpha(T_{\text{set}} - T_{\text{real}}) + \beta(T_{\text{ext}} - T_{\text{real}})$$

où $\alpha$ et $\beta$ sont des constantes de contrôle.

1. **Solution avec Matlab** :

```
% Paramètres
alpha = 0.5; % Constante de chauffage/refroidissement
beta = 0.1; % Constante d'influence de la température externe
T_set = 25; % Température de consigne (°C)
T_ext = 5; % Température externe (°C)
T_real_init = 15; % Température initiale (°C)
t = linspace(0, 100, 1000); % Temps (s)

% Équation différentielle
function dTdt = temperature_system(t, T_real)
    dTdt = alpha * (T_set - T_real) + beta * (T_ext - T_real);
end

% Simulation
[t, T_real] = ode45(@temperature_system, t, T_real_init);

% Affichage
plot(t, T_real)
xlabel('Temps (s)')
ylabel('Température réelle (°C)')
title('Contrôle de température')
```

**Exercice 2 : Système de Suspension de Véhicule**

**Objectif** : Modéliser et simuler la dynamique d'un système de suspension de véhicule pour analyser les réponses aux chocs de la route.

1. **Schéma Bloc** :
   - Entrée : Déplacement de la route (r).
   - Système : Masse suspendue, ressort, et amortisseur.
   - Sortie : Déplacement de la carrosserie (x).
2. **Variables Caractéristiques** :
   - $r(t)$ : Déplacement de la route.
   - $x(t)$ : Déplacement de la carrosserie.
   - $\dot{x}(t)$ : Vitesse de la carrosserie.
   - $\ddot{x}(t)$ : Accélération de la carrosserie.
3. **Modèle Mathématique** :

$$m\ddot{x} + c\dot{x} + kx = kr(t)$$

   où $m$ est la masse, $c$ est le coefficient d'amortissement, et $k$ est la constante de raideur du ressort.
4. **Solution avec Matlab** :

```
% Paramètres
m = 1000; % Masse (kg)
c = 1500; % Coefficient d'amortissement (Ns/m)
k = 20000; % Constante de raideur (N/m)
r = @(t) sin(0.1*t); % Déplacement de la route (m)
t = linspace(0, 100, 1000); % Temps (s)
```

```
% Équations différentielles
function dxdt = suspension_system(t, x)
    dxdt = zeros(2,1);
    dxdt(1) = x(2);
    dxdt(2) = (1/m) * (k * (r(t) - x(1)) - c * x(2));
end

% Conditions initiales
x0 = [0; 0]; % Déplacement initial et vitesse initiale

% Simulation
[t, x] = ode45(@suspension_system, t, x0);

% Affichage
plot(t, x(:,1))
xlabel('Temps (s)')
ylabel('Déplacement de la carrosserie (m)')
title('Dynamique du système de suspension')
```

## Exercice 3 : Système de Réservoir à Débordement

**Objectif** : Modéliser et simuler un système de réservoir pour étudier le comportement du niveau d'eau avec des entrées et sorties variables.

1. **Schéma Bloc** :
   - Entrée : Débit d'entrée ($Q_{in}$).
   - Réservoir : Stockage d'eau avec un débit de sortie ($Q_{out}$).
   - Sortie : Niveau d'eau ($h$).
2. **Variables Caractéristiques** :
   - $Q_{in}(t)$ : Débit d'entrée.
   - $Q_{out}(t)$ : Débit de sortie.
   - $h(t)$ : Niveau d'eau.
3. **Modèle Mathématique** :

$$\frac{dh}{dt} = \frac{Q_{in}(t) - Q_{out}(t)}{A}$$

   où $A$ est la surface du réservoir.
4. **Solution avec Matlab** :

```
% Paramètres
A = 10; % Surface du réservoir (m^2)
Q_in = @(t) 5 + 2*sin(0.1*t); % Débit d'entrée (m^3/s)
Q_out = @(h) 2 * h; % Débit de sortie (m^3/s) proportionnel au
niveau
t = linspace(0, 100, 1000); % Temps (s)

% Équation différentielle
function dhdt = reservoir_system(t, h)
    dhdt = (Q_in(t) - Q_out(h)) / A;
end
% Conditions initiales
h0 = 0; % Niveau initial
% Simulation
```

```
[t, h] = ode45(@reservoir_system, t, h0);
% Affichage
plot(t, h)xlabel('Temps (s)')
ylabel('Niveau d''eau (m)')
title('Dynamique du réservoir à débordement')
```

**Exercice 4 : Système de Réaction Chimique**

**Objectif** : Modéliser et simuler une réaction chimique dans un réacteur à cuve agitée continue

1. **Schéma Bloc** :
   - Entrée : Concentration du réactif A ($C_{A,\text{in}}$), débit d'entrée ($Q_{\text{in}}$).
   - Réacteur : CSTR où la réaction se produit.
   - Sortie : Concentration du produit ($C_P$).
2. **Variables Caractéristiques** :
   - $C_A(t)$ : Concentration du réactif A.
   - $C_P(t)$ : Concentration du produit.
   - $Q_{\text{in}}$ : Débit d'entrée.
3. **Modèle Mathématique** :

$$\frac{dC_A}{dt} = \frac{Q_{\text{in}}}{V}(C_{A,\text{in}} - C_A) - kC_A$$

$$\frac{dC_P}{dt} = kC_A - \frac{Q_{\text{out}}}{V}C_P$$

**Solution avec Matlab** :

```
% Paramètres
Q_in = 1; % Débit d'entrée (L/s)
V = 100; % Volume du réacteur (L)
k = 0.1; % Constante de vitesse de réaction (1/s)
C_A_in = 1; % Concentration d'entrée du réactif A (mol/L)
t = linspace(0, 500, 1000); % Temps (s)

% Équations différentielles
function dCdt = reaction_system(t, C)
    C_A = C(1);
    C_P = C(2);
    dC_Adt = (Q_in/V) * (C_A_in - C_A) - k * C_A;
    dC_Pdt = k * C_A - (Q_in/V) * C_P;
    dCdt = [dC_Adt; dC_Pdt];
end

% Conditions initiales
C0 = [0; 0]; % Concentrations initiales de A et P

% Simulation
[t, C] = ode45(@reaction_system, t, C0);

% Affichage
plot(t, C(:,1), t, C(:,2))
xlabel('Temps (s)')
```

```
ylabel('Concentration (mol/L)')
legend('Concentration de A', 'Concentration de P')
title('Dynamique de la réaction chimique dans un CSTR')
```

**Exercice 5 : Système de Croissance de Population**

**Objectif** : Modéliser et simuler la croissance d'une population humaine en fonction des taux de natalité et de mortalité.

1. **Schéma Bloc** :
   - Entrée : Taux de natalité ($b$), taux de mortalité ($d$).
   - Système : Population totale.
   - Sortie : Taille de la population ($P$).
2. **Variables Caractéristiques** :
   - $P(t)$ : Taille de la population.
   - $b$ : Taux de natalité.
   - $d$ : Taux de mortalité.
3. **Modèle Mathématique** :

$$\frac{dP}{dt} = bP - dP$$

où $b$ est le taux de natalité et $d$ est le taux de mortalité.

1. **Solution avec Matlab** :

```
% Paramètres
b = 0.02; % Taux de natalité (par an)
d = 0.01; % Taux de mortalité (par an)
t = linspace(0, 100, 1000); % Temps (années)
% Équation différentielle
function dPdt = population_growth(t, P)
    dPdt = (b - d) * P;
end
% Conditions initiales
P0 = 1000; % Taille initiale de la population
% Simulation
[t, P] = ode45(@population_growth, t, P0);

% Affichage
plot(t, P)
xlabel('Temps (années)')
ylabel('Taille de la population')
title('Croissance de la population')
```

**Conclusion**

Ce chapitre a introduit les concepts fondamentaux de la modélisation mathématique à travers l'utilisation de schémas bloc, de variables caractéristiques, et des représentations interne et externe des systèmes. Cinq exercices pratiques ont été fournis, chacun illustrant une approche différente de modélisation et de simulation à l'aide de Matlab. En maîtrisant ces concepts et techniques, les étudiants et les professionnels seront mieux équipés pour analyser et optimiser des systèmes complexes dans divers domaines d'application.

## Applications : les Modèles Mathématiques Multivariables

Les modèles mathématiques multivariables sont essentiels pour représenter, analyser et comprendre les systèmes complexes où plusieurs variables interagissent simultanément. Ces exercices mettent en lumière comment formuler et résoudre des modèles mathématiques impliquant plusieurs variables, tout en utilisant des schémas bloc, des variables caractéristiques et des représentations internes et externes. Chaque exercice est accompagné de solutions détaillées et de programmes Matlab pour faciliter la compréhension.

**Introduction**

Chaque système complexe peut être décrit par un modèle mathématique qui illustre les interactions entre ses variables d'entrée, de sortie et d'état. Les modèles multivariables sont souvent nécessaires pour capturer ces interactions de manière réaliste. L'objectif de ces exercices est d'enseigner comment construire, analyser et simuler des modèles mathématiques multivariables à l'aide de Matlab, tout en utilisant des schémas bloc et des représentations de systèmes.

**Exercices**

**Exercice 1: Système de Mélange de Liquides**

**Objectif**: Modéliser un système de mélange de deux liquides avec différentes concentrations de soluté.

1. **Schéma Bloc**:
   - **Entrées**: Débits d'entrée des deux liquides $Q_1$ et $Q_2$ avec concentrations $C_1$ et $C_2$.
   - **Bloc de mélange**: Réservoir où les liquides sont mélangés.
   - **Sortie**: Débit de sortie $Q_{out}$ avec concentration $C_{out}$.
2. **Variables Caractéristiques**:
   - $Q_1$, $Q_2$: Débits d'entrée (L/min)
   - $C_1$, $C_2$: Concentrations d'entrée (mol/L)
   - $C_{out}$: Concentration de sortie (mol/L)
3. **Modèle Mathématique**:

$$\frac{dC_{out}}{dt} = \frac{Q_1 C_1 + Q_2 C_2 - Q_{out} C_{out}}{V}$$

où $V$ est le volume du réservoir.

**Solution Matlab**:

```
Copier le code
% Paramètres
Q1 = 5; % Débit du liquide 1 (L/min)
```

```
Q2 = 3; % Débit du liquide 2 (L/min)
C1 = 2; % Concentration du liquide 1 (mol/L)
C2 = 1; % Concentration du liquide 2 (mol/L)
Q_out = 8; % Débit de sortie (L/min)
V = 50; % Volume du réservoir (L)
t = linspace(0, 100, 1000); % Temps (min)

% Équation différentielle
function dCdt = mix_system(t, C_out)
    dCdt = (Q1*C1 + Q2*C2 - Q_out*C_out) / V;
end

% Conditions initiales
C_out_init = 0; % Concentration initiale

% Simulation
[t, C_out] = ode45(@mix_system, t, C_out_init);

% Affichage
plot(t, C_out)
xlabel('Temps (min)')
ylabel('Concentration de sortie (mol/L)')
title('Concentration de sortie d\'un réservoir de mélange')
```

**Exercice 2: Système de Suspension Double**

**Objectif**: Analyser la dynamique d'un système de suspension automobile avec deux masses.

1. Schéma Bloc:
   - **Entrées**: Déplacement de la route ($r(t)$).
   - **Masse 1**: Carrosserie de la voiture.
   - **Masse 2**: Châssis de la voiture.
   - **Sortie**: Déplacements des masses $x_1(t)$ et $x_2(t)$.
2. Variables Caractéristiques:
   - $x_1, x_2$: Déplacements des masses (m)
   - $r(t)$: Déplacement de la route (m)
3. Modèle Mathématique:

$$m_1\ddot{x}_1 + c_1(\dot{x}_1 - \dot{x}_2) + k_1(x_1 - x_2) = 0$$

$$m_2\ddot{x}_2 + c_1(\dot{x}_2 - \dot{x}_1) + k_1(x_2 - x_1) + c_2\dot{x}_2 + k_2x_2 = k_2r(t)$$

4. Solution Matlab:

```
% Paramètres
m1 = 250; % Masse de la carrosserie (kg)
m2 = 50; % Masse du châssis (kg)
c1 = 1000; % Coefficient d'amortissement entre masses (Ns/m)
c2 = 500; % Coefficient d'amortissement avec sol (Ns/m)
k1 = 15000; % Constante de raideur entre masses (N/m)
k2 = 20000; % Constante de raideur avec sol (N/m)
```

```
r = @(t) 0.1 * sin(0.1 * t); % Déplacement de la route (m)
t = linspace(0, 100, 1000); % Temps (s)

% Équations différentielles
function dxdt = suspension_double(t, x)
    dxdt = zeros(4,1);
    dxdt(1) = x(2);
    dxdt(2) = (-c1*(x(2)-x(4)) - k1*(x(1)-x(3)))/m1;
    dxdt(3) = x(4);
    dxdt(4) = (c1*(x(2)-x(4)) + k1*(x(1)-x(3)) - c2*x(4) -
k2*x(3) + k2*r(t))/m2;
end

% Conditions initiales
x0 = [0; 0; 0; 0]; % Positions et vitesses initiales

% Simulation
[t, x] = ode45(@suspension_double, t, x0);

% Affichage
plot(t, x(:,1), t, x(:,3))
xlabel('Temps (s)')
ylabel('Déplacement (m)')
legend('Carrosserie', 'Châssis')
title('Dynamique du système de suspension double')
```

**Exercice 3: Système de Contrôle de Niveau d'Eau à Double Réservoir**

1. Schéma Bloc:
   - **Entrées**: Débit d'eau d'entrée ($Q_{in}$).
   - **Réservoir 1**: Premier réservoir.
   - **Réservoir 2**: Deuxième réservoir connecté.
   - **Sortie**: Niveaux d'eau $h_1$ et $h_2$.
2. Variables Caractéristiques:
   - $h_1, h_2$: Niveaux d'eau des réservoirs (m)
   - $Q_{in}$: Débit d'entrée (m^3/s)
3. Modèle Mathématique:

$$\frac{dh_1}{dt} = \frac{Q_{in} - Q_1}{A_1}$$

$$\frac{dh_2}{dt} = \frac{Q_1 - Q_2}{A_2}$$

   où $Q_1 = k_1(h_1 - h_2)$ et $Q_2 = k_2 h_2$.

4. Solution Matlab:

```
% Paramètres
A1 = 10; % Surface du réservoir 1 (m^2)
A2 = 8; % Surface du réservoir 2 (m^2)
k1 = 0.5; % Coefficient de transfert entre réservoirs
k2 = 0.3; % Coefficient de sortie du réservoir 2
```

```
Q_in = 0.2; % Débit d'entrée (m^3/s)
t = linspace(0, 200, 1000); % Temps (s)

% Équations différentielles
function dhdt = water_level_system(t, h)
    Q1 = k1 * (h(1) - h(2));
    Q2 = k2 * h(2);
    dh1dt = (Q_in - Q1) / A1;
    dh2dt = (Q1 - Q2) / A2;
    dhdt = [dh1dt; dh2dt];
end

% Conditions initiales
h0 = [0; 0]; % Niveaux initiaux

% Simulation
[t, h] = ode45(@water_level_system, t, h0);

% Affichage
plot(t, h(:,1), t, h(:,2))
xlabel('Temps (s)')
ylabel('Niveau d\'eau (m)')
legend('Réservoir 1', 'Réservoir 2')
title('Niveaux d\'eau dans les réservoirs connectés')
```

**Exercice 4: Système de Réaction Chimique à Deux Réacteurs en Série**

**Objectif**: Modéliser la dynamique des concentrations de réactifs dans deux réacteurs chimiques en série.

1. Schéma Bloc:
   - **Entrées**: Concentration d'entrée ($C_{in}$), débit d'entrée ($Q_{in}$).
   - **Réacteur 1**: Premier réacteur.
   - **Réacteur 2**: Deuxième réacteur.
   - **Sortie**: Concentrations $C_1$ et $C_2$.
2. Variables Caractéristiques:
   - $C_1$, $C_2$: Concentrations dans les réacteurs (mol/L)
   - $Q_{in}$: Débit d'entrée (L/s)
3. Modèle Mathématique:

$$\frac{dC_1}{dt} = \frac{Q_{in}}{V}(C_{in} - C_1) - k_1 C_1$$

$$\frac{dC_2}{dt} = \frac{Q_{in}}{V}(C_1 - C_2) - k_2 C_2$$

4. Solution Matlab:

```
% Paramètres
    V = 100; % Volume des réacteurs (L)
    k1 = 0.1; % Constante de réaction réacteur 1 (1/s)
    k2 = 0.05; % Constante de réaction réacteur 2 (1/s)
```

```
C_in = 1; % Concentration d'entrée (mol/L)
Q_in = 1; % Débit d'entrée (L/s)
t = linspace(0, 500, 1000); % Temps (s)

% Équations différentielles
function dCdt = reaction_series(t, C)
    dC1dt = (Q_in/V) * (C_in - C(1)) - k1 * C(1);
    dC2dt = (Q_in/V) * (C(1) - C(2)) - k2 * C(2);
    dCdt = [dC1dt; dC2dt];
end

% Conditions initiales
C0 = [0; 0]; % Concentrations initiales

% Simulation
[t, C] = ode45(@reaction_series, t, C0);

% Affichage
plot(t, C(:,1), t, C(:,2))
xlabel('Temps (s)')
ylabel('Concentration (mol/L)')
legend('Réacteur 1', 'Réacteur 2')
title('Dynamique des concentrations dans les réacteurs en série')
```

**Exercice 5: Système de Contrôle de Température Multi-Zone**

1. Schéma Bloc:
   - **Entrées**: Températures de consigne ($T_{set,1}$, $T_{set,2}$), températures externes ($T_{ext}$).
   - **Zones**: Deux zones à contrôler indépendamment.
   - **Sortie**: Températures réelles des zones $T_1$, $T_2$.
2. Variables Caractéristiques:
   - $T_1$, $T_2$: Températures des zones (°C)
   - $T_{set,1}$, $T_{set,2}$: Températures de consigne (°C)
3. Modèle Mathématique:

$$\frac{dT_1}{dt} = \alpha_1(T_{set,1} - T_1) + \beta(T_{ext} - T_1)$$

$$\frac{dT_2}{dt} = \alpha_2(T_{set,2} - T_2) + \beta(T_{ext} - T_2)$$

4. Solution Matlab:

```
% Paramètres
alpha1 = 0.1; % Constante de contrôle zone 1
alpha2 = 0.15; % Constante de contrôle zone 2
beta = 0.05; % Influence de la température externe
T_set1 = 22; % Température de consigne zone 1 (°C)
T_set2 = 20; % Température de consigne zone 2 (°C)
T_ext = 10; % Température externe (°C)
t = linspace(0, 500, 1000); % Temps (s)
```

```
% Équations différentielles
function dTdt = temp_control_multi_zone(t, T)
    dT1dt = alpha1 * (T_set1 - T(1)) + beta * (T_ext - T(1));
    dT2dt = alpha2 * (T_set2 - T(2)) + beta * (T_ext - T(2));
    dTdt = [dT1dt; dT2dt];
end

% Conditions initiales
T0 = [15; 15]; % Températures initiales des zones

% Simulation
[t, T] = ode45(@temp_control_multi_zone, t, T0);

% Affichage
plot(t, T(:,1), t, T(:,2))
xlabel('Temps (s)')
ylabel('Température (°C)')
legend('Zone 1', 'Zone 2')
title('Contrôle de température multi-zone')
```

**Conclusion**

Ces exercices pratiques fournissent une introduction complète à la modélisation des systèmes multivariables à l'aide de schémas bloc, de variables caractéristiques et de représentations interne et externe. Chaque exercice démontre comment formuler un modèle mathématique à partir de descriptions physiques d'un système, et comment résoudre ces modèles à l'aide de Matlab pour analyser le comportement dynamique du système. Ces compétences sont essentielles pour les ingénieurs et les scientifiques travaillant avec des systèmes complexes dans des domaines variés.

# Applications : Modèles Mathématiques Multivariables : Principes Fondamentaux avec Schémas et Figures

Les modèles mathématiques multivariables sont utilisés pour représenter des systèmes où plusieurs variables interagissent simultanément. Ils permettent de comprendre et d'analyser le comportement de systèmes complexes, en capturant les interactions entre les variables d'entrée, d'état, et de sortie. Ce document explique les principes de base des modèles multivariables à l'aide de schémas bloc, de variables caractéristiques, et de représentations internes et externes. Cinq schémas explicatifs sont fournis pour illustrer ces concepts.

## 1. Schéma Bloc d'un Système

### Principe du Schéma Bloc

Un **schéma bloc** est une représentation graphique d'un système qui montre les différents composants et leur interaction par le biais de blocs connectés par des flèches. Chaque bloc représente une fonction ou une sous-partie du système, tandis que les flèches indiquent les flux de données ou de signaux entre les blocs.

### Composantes du Schéma Bloc

- **Blocs** : Ils représentent les sous-systèmes ou fonctions spécifiques du système, par exemple, un contrôleur, un capteur ou un actuateur.
- **Flèches** : Elles indiquent la direction du flux d'information ou de signaux entre les blocs.
- **Entrées et Sorties** : Les signaux ou données qui entrent ou sortent du système.

### Schéma Bloc Exemple : Système de Contrôle de Température

Dans un système de contrôle de température, le schéma bloc pourrait ressembler à ceci :

- **Entrée** : Température de consigne ($T_{\text{set}}$), température mesurée ($T_{\text{mesure}}$).
- **Bloc de Contrôle** : Composant qui ajuste le chauffage ou le refroidissement pour atteindre la température de consigne.
- **Actuateur** : Élément qui effectue l'action de chauffage ou de refroidissement.
- **Sortie** : Température réelle ($T_{\text{réelle}}$).

Ce schéma montre comment les différentes parties du système interagissent pour contrôler la température. Le bloc de contrôle prend en compte la température de consigne et la température mesurée pour ajuster l'actuateur.

## 2. Variables Caractéristiques

### Définition

Les **variables caractéristiques** sont des variables qui décrivent l'état du système et déterminent son comportement. Elles sont généralement classées en :

- **Variables d'entrée** : Variables qui influencent le système de l'extérieur, par exemple, une force appliquée ou une tension d'alimentation.
- **Variables d'état** : Variables internes qui décrivent l'état du système, par exemple, la position ou la vitesse.

- **Variables de sortie** : Résultats mesurables du système, par exemple, la vitesse finale ou la température.

**Exemple : Système de Réservoir**

Pour un système de réservoir, les variables caractéristiques pourraient inclure :

- **Entrée** : Débit d'eau entrant ($Q_{\text{in}}$).
- **État** : Niveau d'eau dans le réservoir ($h$).
- **Sortie** : Débit d'eau sortant ($Q_{\text{out}}$).

Ce schéma montre comment le débit d'entrée affecte le niveau d'eau, qui à son tour influence le débit de sortie.

**3. Représentation Interne d'un Système**

**Principe de la Représentation Interne**

La **représentation interne** d'un système se concentre sur les processus internes et les relations entre les variables d'état et les entrées. Cette approche est utilisée pour comprendre le comportement interne et la dynamique du système.

**Exemple : Modèle d'État**

Dans un modèle d'état, les équations différentielles représentent les relations entre les variables d'état et les entrées. Par exemple, pour un système de suspension de véhicule :

- **Variables d'état** : Position ($x$) et vitesse ($\dot{x}$).
- **Entrée** : Déplacement de la route ($r(t)$).

Les équations d'état pourraient être :

$$\frac{dx}{dt} = \dot{x}$$

$$\frac{d\dot{x}}{dt} = \frac{1}{m}(-c\dot{x} - kx + kr(t))$$

Ce schéma montre comment les variables d'état évoluent en réponse à une entrée donnée.

**4. Représentation Externe d'un Système**

**Principe de la Représentation Externe**

La **représentation externe** se concentre sur les relations entre les variables d'entrée et de sortie, sans se préoccuper des détails internes. Elle est souvent utilisée pour analyser le comportement global du système et pour le contrôle.

**Exemple : Fonction de Transfert**

Dans un système linéaire, la relation entre l'entrée et la sortie peut être décrite par une fonction de transfert. Par exemple, pour un système de contrôle de position :

- **Entrée** : Position désirée ($X_{\text{set}}$).
- **Sortie** : Position réelle ($X_{\text{réelle}}$).

La fonction de transfert pourrait être représentée comme :

$$H(s) = \frac{X_{\text{réelle}}(s)}{X_{\text{set}}(s)}$$

Ce schéma montre comment une entrée est transformée en sortie à travers la fonction de transfert du système.

**5. Exemple de Système Multivariable : Système de Mélange Chimique**

**Description**

Un système de mélange chimique peut impliquer plusieurs réacteurs et composants, chacun ayant ses propres variables d'entrée, d'état, et de sortie. Par exemple :

- **Entrées** : Concentrations des réactifs ($C_{A,\text{in}}$, $C_{B,\text{in}}$), débit ($Q_{\text{in}}$).
- **États** : Concentrations internes dans les réacteurs ($C_A$, $C_B$).
- **Sorties** : Concentration des produits ($C_{\text{produit}}$).

## Schéma Bloc

Un schéma bloc pour un système de mélange chimique avec deux réacteurs en série pourrait inclure :

1. **Réacteur 1** : Mélange initial des réactifs.
2. **Réacteur 2** : Réaction secondaire et formation du produit.
3. **Sorties** : Concentrations de produits en sortie des réacteurs.

## Variables Caractéristiques et Modèle Mathématique

- **Entrées** : $C_{A,\text{in}}$, $C_{B,\text{in}}$, $Q_{\text{in}}$.
- **États** : $C_A$, $C_B$.
- **Sorties** : $C_{\text{produit}}$.

**Modèle Mathématique** :

$$\frac{dC_A}{dt} = \frac{Q_{\text{in}}}{V}(C_{A,\text{in}} - C_A) - k_1 C_A$$

$$\frac{dC_B}{dt} = \frac{Q_{\text{in}}}{V}(C_A - C_B) - k_2 C_B$$

**Conclusion**

Les modèles mathématiques multivariables jouent un rôle essentiel dans la compréhension et la gestion des systèmes complexes. En utilisant des schémas bloc, des variables caractéristiques, et des représentations internes et externes, il est possible de décrire avec précision les interactions entre les différentes parties d'un système. Ces principes sont applicables dans de nombreux domaines, de l'ingénierie des systèmes à la gestion des processus industriels, en passant par la biologie et l'économie. En utilisant des représentations visuelles et des modèles mathématiques, les ingénieurs et scientifiques peuvent concevoir des systèmes plus efficaces et résoudre des problèmes complexes

# Chapitre 3

# Modélisation des Systèmes Électriques (02 Semaines)

## Introduction

La modélisation des systèmes électriques est une compétence fondamentale en ingénierie électrique. Elle permet de comprendre le comportement des circuits et des composants sous différentes conditions, de concevoir des systèmes efficaces et de résoudre des problèmes pratiques. Ce chapitre couvre la modélisation des composants passifs et actifs, ainsi que des circuits électriques de base, avec des exemples d'applications pratiques.

## 1. Modélisation des Composants Passifs

Les composants passifs sont ceux qui ne génèrent pas d'énergie mais qui peuvent stocker ou dissiper de l'énergie. Les composants passifs les plus courants dans les systèmes électriques sont les résistances, les inductances et les condensateurs.

### 1.1 Résistance

### Modèle Théorique

Modèle Théorique

- **Loi d'Ohm** : La relation fondamentale pour une résistance est donnée par la loi d'Ohm :

$$V = RI$$

où $V$ est la tension à travers la résistance $R$, et $I$ est le courant qui la traverse.

- **Caractéristiques** : Les résistances dissipent de l'énergie sous forme de chaleur. Elles sont utilisées pour contrôler le courant et diviser la tension dans les circuits.

Exemple d'Application

**Circuit Diviseur de Tension** : Un diviseur de tension est un circuit simple utilisé pour obtenir une fraction de la tension d'entrée.

- **Circuit** : Deux résistances en série $R_1$ et $R_2$ sont connectées à une tension d'entrée $V_{\text{in}}$.
- **Tension de Sortie** : La tension de sortie $V_{\text{out}}$ est donnée par :

$$V_{\text{out}} = V_{\text{in}} \frac{R_2}{R_1 + R_2}$$

### 1.2 Inductance

Modèle Théorique

- **Loi de Faraday** : La relation fondamentale pour une inductance est donnée par la loi de Faraday :
- **Caractéristiques** : Les inducteurs stockent l'énergie sous forme de champ magnétique. Ils s'opposent aux variations rapides de courant.

### Exemple d'Application

**Filtre Passe-Bas à Inductance** : Utilisé pour éliminer les hautes fréquences dans un signal électrique.

- **Circuit** : Une inductance $L$ en série avec une résistance $R$.
- **Fonction de Transfert** : La fonction de transfert pour un filtre passe-bas à inductance est :

$$H(s) = \frac{1}{1 + sL/R}$$

où $s$ est la variable complexe de Laplace.

### 1.3 Condensateur

- **Loi de Coulomb** : La relation fondamentale pour un condensateur est donnée par la loi de Coulomb :

$$I = C\frac{dV}{dt}$$

où $C$ est la capacité, $I$ est le courant à travers le condensateur, et $\frac{dV}{dt}$ est la dérivée de la tension par rapport au temps.

- **Caractéristiques** : Les condensateurs stockent l'énergie sous forme de champ électrique. Ils s'opposent aux variations rapides de tension.

**Exemple d'Application**

**Circuit RC** : Utilisé comme filtre passe-bas pour atténuer les signaux à haute fréquence.

- **Circuit** : Une résistance $R$ en série avec un condensateur $C$.
- **Fonction de Transfert** : La fonction de transfert pour un filtre RC passe-bas est :

$$H(s) = \frac{1}{1 + sRC}$$

## .2. Modélisation des Composants Actifs

Les composants actifs peuvent générer ou amplifier de l'énergie. Les composants actifs les plus courants sont les transistors et les amplificateurs opérationnels.

### 2.1 Transistor

**Modèle Théorique**

- **Transistor Bipolaire (BJT)** : Utilisé pour amplifier ou commuter des signaux.

$$I_C = \beta I_B$$

où $I_C$ est le courant de collecteur, $I_B$ est le courant de base, et $\beta$ est le gain de courant du transistor.

- **Transistor à Effet de Champ (FET)** : Utilisé pour amplifier des signaux avec une faible consommation d'énergie.

$$I_D = k(V_{GS} - V_{th})^2$$

où $I_D$ est le courant de drain, $V_{GS}$ est la tension grille-source, et $V_{th}$ est la tension de seuil.

.

Exemple d'Application

**Amplificateur à Transistor** : Utilisé pour amplifier les signaux audio ou radio.

- **Circuit** : Un transistor NPN avec une résistance de charge $R_C$ et une résistance d'entrée $R_B$.
- **Gain en Tension** : Le gain en tension est approximativement :

$$A_v = \frac{R_C}{R_E}$$

où $R_E$ est la résistance d'émetteur.

## 2.2 Amplificateur Opérationnel

### Modèle Théorique

- **Amplification** : Un amplificateur opérationnel amplifie la différence de tension entre ses deux entrées :

$$V_{\text{out}} = A(V_+ - V_-)$$

où $V_{\text{out}}$ est la tension de sortie, $V_+$ et $V_-$ sont les tensions d'entrée, et $A$ est le gain.

- **Configuration Inverseur** : Utilisé pour inverser la phase du signal d'entrée.

$$V_{\text{out}} = -\frac{R_f}{R_{\text{in}}} V_{\text{in}}$$

où $R_f$ est la résistance de rétroaction et $R_{\text{in}}$ est la résistance d'entrée.

Exemple d'Application

**Filtre Passe-Bande** : Un amplificateur opérationnel utilisé avec des résistances et des condensateurs pour filtrer une plage spécifique de fréquences.

### Exemple d'Application

**Filtre Passe-Bande** : Un amplificateur opérationnel utilisé avec des résistances et des condensateurs pour filtrer une plage spécifique de fréquences.

Exemple d'Application

**Filtre Passe-Bande** : Un amplificateur opérationnel utilisé avec des résistances et des condensateurs pour filtrer une plage spécifique de fréquences.

.

- **Circuit** : Amplificateur opérationnel avec des composants passifs pour réaliser un filtre passe-bande.
- **Fonction de Transfert** : La fonction de transfert pour un filtre passe-bande est :

$$H(s) = \frac{sRC}{1 + sRC + (sRC)^2}$$

## 3. Modélisation des Circuits Électriques de Base

### 3.1 Circuit Série RLC

#### Modèle Théorique

- **Équation Différentielle** : Pour un circuit série avec une résistance $R$, une inductance $L$, et un condensateur $C$ :

$$V_{\text{in}} = RI + L\frac{dI}{dt} + \frac{1}{C}\int Idt$$

- **Réponse en Fréquence** : Le comportement du circuit en fonction de la fréquence est déterminé par :

$$H(s) = \frac{V_{\text{out}}(s)}{V_{\text{in}}(s)} = \frac{1}{Ls^2 + Rs + \frac{1}{C}}$$

#### Exemple d'Application

**Circuit Résonant** : Utilisé pour sélectionner des fréquences spécifiques dans les applications de radio.

- **Circuit** : Un inducteur $L$ et un condensateur $C$ en série avec une résistance $R$.
- **Fréquence de Résonance** : La fréquence de résonance est donnée par :

$$f_0 = \frac{1}{2\pi\sqrt{LC}}$$

#### Modèle Théorique

- **Équilibre** : Un pont de Wheatstone est en équilibre lorsque la tension de sortie est nulle, ce qui est utilisé pour mesurer des résistances inconnues.

$$\frac{R_1}{R_2} = \frac{R_3}{R_x}$$

- **Sensibilité** : La sensibilité du pont dépend des valeurs des résistances.

#### Exemple d'Application

**Mesure de Résistance** : Utilisé pour mesurer des résistances inconnues avec une grande précision.

- **Circuit** : Quatre résistances, dont une inconnue, disposées en pont.
- **Tension de Sortie** : La tension de sortie du pont est utilisée pour calculer la valeur de la résistance inconnue.

### 3.3 Circuit de Conversion DC-DC

Modèle Théorique

- **Buck Converter** : Un convertisseur Buck réduit la tension d'entrée à une tension de sortie inférieure.

$$V_{\text{out}} = D \times V_{\text{in}}$$

où $D$ est le rapport cyclique de l'interrupteur.

- **Boost Converter** : Un convertisseur Boost augmente la tension d'entrée à une tension de sortie supérieure.

$$V_{\text{out}} = \frac{V_{\text{in}}}{1 - D}$$

Exemple d'Application

**Régulation de Tension** : Utilisé pour alimenter des composants électroniques avec une tension stable.

- **Circuit** : Un interrupteur, une diode, un condensateur, et une inductance configurés pour convertir la tension.
- **Efficacité** : Les convertisseurs DC-DC sont utilisés pour minimiser les pertes d'énergie.

.

### Conclusion

Ce chapitre a couvert les principes fondamentaux de la modélisation des systèmes électriques, y compris les composants passifs, les composants actifs, et les circuits de base. Les exemples d'applications pratiques montrent comment ces concepts sont utilisés dans des systèmes réels pour résoudre des problèmes courants en ingénierie électrique. La modélisation mathématique est une compétence essentielle pour analyser et concevoir des systèmes électriques efficaces, et ces techniques sont largement applicables dans divers domaines, allant des télécommunications à l'électronique de puissance

## Applications : Modélisation des Systèmes Électriques avec Matlab

Dans cette section, nous explorerons des exercices pratiques détaillés sur la modélisation des systèmes électriques, comprenant des composants passifs et actifs ainsi que des circuits électriques de base. Ces exercices seront enrichis par des simulations Matlab/Simulink, des schémas, et des fonctions de transfert pour donner un aspect pratique à la théorie.

### Exercice 1: Modélisation d'un Circuit RLC en Série

#### Objectif

Modéliser un circuit RLC en série et analyser sa réponse en fréquence et sa réponse temporelle.

- **Composants**: Une résistance ($R$), une inductance ($L$), et un condensateur ($C$) connectés en série.
- **Entrée**: Tension d'entrée ($V_{in}$).
- **Sortie**: Tension aux bornes du condensateur ($V_C$).

#### Modèle Mathématique

1. **Équation Différentielle**: Pour un circuit RLC en série, l'équation est donnée par :

$$V_{in} = RI + L\frac{dI}{dt} + \frac{1}{C}\int Idt$$

2. **Forme en Fonction de Transfert**: En utilisant la transformée de Laplace, la fonction de transfert est :

$$H(s) = \frac{V_C(s)}{V_{in}(s)} = \frac{1}{Ls^2 + Rs + \frac{1}{C}}$$

#### Simulation Matlab/Simulink

1. **Code Matlab**: Simuler la réponse en fréquence et la réponse temporelle.

.

```
% Paramètres du circuit
R = 10; % Résistance (ohms)
L = 1e-3; % Inductance (henrys)
C = 100e-6; % Capacité (farads)

% Fonction de transfert
num = [1];
den = [L C R];
sys = tf(num, den);

% Réponse en fréquence
figure;
bode(sys);
title('Réponse en fréquence du circuit RLC en série');

% Réponse temporelle à une entrée en échelon
figure;
step(sys);
title('Réponse temporelle du circuit RLC en série');
```

2. **Simulation Simulink**: Modélisation avec des blocs Simulink.
    - **Schéma Simulink**:
        - Utiliser un bloc Source de tension sinusoïdale pour $V_{\text{in}}$.
        - Connecter en série un bloc Résistance, un bloc Inductance, et un bloc Capacité.
        - Mesurer $V_C$ avec un bloc Scope.

## Analyse

- **Réponse en Fréquence**: Le Bode plot montre les fréquences auxquelles le circuit RLC amplifie ou atténue les signaux.
- **Réponse Temporelle**: La réponse en échelon montre comment le circuit réagit à une variation brusque de la tension d'entrée.

**Exercice 2: Modélisation d'un Amplificateur Opérationnel en Configuration Inverseur**

**Objectif**

Modéliser un amplificateur opérationnel en configuration inverseur et analyser son gain et sa réponse fréquentielle.

## Description du Système

- **Composants**: Un amplificateur opérationnel, une résistance d'entrée ($R_{\text{in}}$), et une résistance de rétroaction ($R_f$).
- **Entrée**: Signal d'entrée ($V_{\text{in}}$).
- **Sortie**: Signal amplifié et inversé ($V_{\text{out}}$).

## Modèle Mathématique

1. **Formule de Gain**:

$$V_{\text{out}} = -\frac{R_f}{R_{\text{in}}} V_{\text{in}}$$

2. **Fonction de Transfert**:

$$H(s) = -\frac{R_f}{R_{\text{in}}}$$

## Simulation Matlab/Simulink

*Code Matlab: Simuler la réponse en fréquence et le gain.*

```
% Paramètres de l'amplificateur opérationnel
R_in = 1e3; % Résistance d'entrée (ohms)
R_f = 10e3; % Résistance de rétroaction (ohms)

% Fonction de transfert
num = [-R_f/R_in];
den = [1];
sys = tf(num, den);
```

```
% Réponse en fréquence
figure;
bode(sys);
title('Réponse en fréquence de l\'amplificateur inverseur');

% Réponse temporelle à une entrée en échelon
figure;
step(sys);
title('Réponse temporelle de l\'amplificateur inverseur');
```

2. **Simulation Simulink**: Modélisation avec des blocs Simulink.
   - **Schéma Simulink**:
     - Utiliser un bloc Source de tension pour $V_{\text{in}}$.
     - Connecter $V_{\text{in}}$ à l'entrée inversée de l'amplificateur.
     - Connecter $R_{\text{in}}$ entre $V_{\text{in}}$ et l'entrée inversée.
     - Connecter $R_f$ entre la sortie et l'entrée inversée.
     - Mesurer $V_{\text{out}}$ avec un bloc Scope.

### Analyse

- **Gain en Tension**: Le gain négatif indique une inversion de phase, tandis que le rapport $R_f/R_{\text{in}}$ détermine l'amplification.
- **Réponse Fréquentielle**: Montre la bande passante et la stabilité de l'amplificateur.

**Exercice 3: Modélisation d'un Filtre Passe-Bande à Amplificateur Opérationnel**

**Objectif**

Créer un filtre passe-bande en utilisant un amplificateur opérationnel et analyser la bande passante.

### Description du Système

- **Composants**: Amplificateur opérationnel, résistances et condensateurs configurés pour filtrer des fréquences spécifiques.
- **Entrée**: Signal d'entrée ($V_{\text{in}}$).
- **Sortie**: Signal filtré ($V_{\text{out}}$).

### Modèle Mathématique

1. **Fonction de Transfert**: Pour un filtre passe-bande de second ordre :

$$H(s) = \frac{sRC}{1 + sRC + (sRC)^2}$$

2. **Réponse en Fréquence**:
   - **Fréquence Centrale** ($f_0$) : Où le gain est maximal.
   - **Bande Passante** : Intervalle de fréquences où le signal est amplifié.

.

**Simulation Matlab/Simulink**

1. **Code Matlab**: Analyser la réponse en fréquence.

```
% Paramètres du filtre
R = 1e3; % Résistance (ohms)
C = 1e-6; % Capacité (farads)

% Fonction de transfert
num = [1 0];
den = [1 R*C (R*C)^2];
sys = tf(num, den);

% Réponse en fréquence
figure;
bode(sys);
title('Réponse en fréquence du filtre passe-bande');

% Réponse temporelle à une entrée en échelon
figure;
step(sys);
title('Réponse temporelle du filtre passe-bande');
```

**Exercice 4: Modélisation d'un Convertisseur DC-DC Boost**

**Objectif**

## Description du Système

- **Composants**: Inductance ($L$), diode ($D$), condensateur ($C$), interrupteur ($S$).
- **Entrée**: Tension d'entrée ($V_{\text{in}}$).
- **Sortie**: Tension de sortie ($V_{\text{out}}$).

## Modèle Mathématique

1. **Fonction de Transfert**:

$$H(s) = \frac{V_{\text{out}}(s)}{V_{\text{in}}(s)} = \frac{1}{1 - s\frac{L}{R} + s^2LC}$$

2. **Equation de Conversion**:

.

$$V_{out} = \frac{V_{in}}{1 - D}$$

où $D$ est le rapport cyclique.

**Simulation Matlab/Simulink**

1. **Code Matlab**: Analyser la réponse en tension.

```
% Paramètres du convertisseur Boost
L = 1e-3; % Inductance (henrys)
C = 100e-6; % Capacité (farads)
R = 10; % Résistance de charge (ohms)
V_in = 12; % Tension d'entrée (volts)

% Fonction de transfert
num = [1];
den = [L C R];
sys = tf(num, den);

% Réponse en fréquence
figure;
bode(sys);
title('Réponse en fréquence du convertisseur Boost');

% Réponse temporelle à une entrée en échelon
figure;
step(sys);
title('Réponse temporelle du convertisseur Boost');
```

**Exercice 5: Modélisation d'un Circuit de Puissance Avec Couplage Magnétique**

**Objectif**

Modéliser un circuit de puissance avec transformateur couplé et analyser les effets du couplage magnétique.

## Description du Système

- **Composants**: Deux inductances couplées magnétiquement, transformateur.
- **Entrée**: Tension d'entrée primaire ($V_{in}$).
- **Sortie**: Tension secondaire ($V_{out}$).

## Modèle Mathématique

1. **Relation de Couplage**:

## Modèle Mathématique

1. Relation de Couplage:

$$V_1 = L_1 \frac{dI_1}{dt} + M \frac{dI_2}{dt}$$

$$V_2 = L_2 \frac{dI_2}{dt} + M \frac{dI_1}{dt}$$

2. Fonction de Transfert:

$$H(s) = \frac{V_2(s)}{V_1(s)} = \frac{M}{L_1 + L_2}$$

**Simulation Matlab/Simulink**

1. **Code Matlab**: Analyser la réponse en tension.

```
% Paramètres du transformateur
L1 = 1e-3; % Inductance primaire (henrys)
L2 = 2e-3; % Inductance secondaire (henrys)
M = 0.9e-3; % Inductance mutuelle (henrys)

% Fonction de transfert
num = [M];
den = [L1 L2];
sys = tf(num, den);

% Réponse en fréquence
figure;
bode(sys);
title('Réponse en fréquence du transformateur couplé');

% Réponse temporelle à une entrée en échelon
figure;
step(sys);
title('Réponse temporelle du transformateur couplé');
```

.

**Conclusion**

Ces exercices détaillés couvrent divers aspects de la modélisation des systèmes électriques, y compris les composants passifs, les composants actifs, et les circuits complexes avec couplage. L'utilisation de Matlab et Simulink pour simuler ces systèmes permet d'analyser et de comprendre le comportement des systèmes électriques de manière approfondie. Ces compétences sont essentielles pour les ingénieurs travaillant dans la conception de circuits, les systèmes de contrôle et l'électronique de puissance

# Chapitre 4

## Outils de Modélisation : Bond Graph (BG) ou Graphes Informationnels Causales (GIC)

### Introduction

La modélisation des systèmes dynamiques est une étape cruciale dans la conception et l'analyse des systèmes complexes. Parmi les outils de modélisation utilisés en ingénierie, les **Bond Graphs (BG)** et les **Graphes Informationnels Causales (GIC)** se distinguent par leur capacité à modéliser des systèmes multidisciplinaires de manière unifiée. Ces méthodes sont particulièrement utiles pour représenter les interactions entre l'énergie, l'information et les forces au sein des systèmes mécaniques, électriques, hydrauliques, thermiques, et plus encore. Dans ce cours, nous nous concentrerons sur l'application de ces méthodes aux circuits électriques.

### 1. Qu'est-ce qu'un Bond Graph (BG) ?

Les Bond Graphs sont des représentations graphiques qui modélisent les flux d'énergie entre différents composants d'un système. Ils utilisent des éléments standardisés pour représenter les différentes variables et interactions énergétiques. Un Bond Graph permet de décrire un système à un niveau abstrait, capturant les interactions de manière systématique.

### 1.1 Éléments de Base des Bond Graphs

- **Bonds (Liens)** : Représentent les flux d'énergie entre les composants. Chaque lien transporte deux variables : l'effort (e) et le flux (f). Dans les circuits électriques, cela correspond respectivement à la tension (V) et au courant (I).
- **Nodes (Nœuds)** : Points de connexion pour les bonds. Les deux types principaux sont les *nœuds de jonction* 0 et 1.
  - **Nœud 0** : Représente une somme d'efforts (par exemple, somme des tensions à un nœud commun).
  - **Nœud 1** : Représente une somme de flux (par exemple, somme des courants entrant dans un point commun).
- **Éléments de Stockage** :
  - **Capacité (C)** : Stocke de l'énergie comme un condensateur (ex : $C \cdot V$).
  - **Inductance (I)** : Stocke de l'énergie cinétique comme une inductance (ex : $L \cdot I$).
- **Éléments de Dissipation** :
  - **Résistance (R)** : Dissipe l'énergie comme une résistance (ex : $R \cdot I^2$).
- **Sources** :
  - **Source d'Effort (Se)** : Génère une tension constante.
  - **Source de Flux (Sf)** : Génère un courant constant.

### 1.2 Représentation Symbolique des Bond Graphs

- Les liens sont représentés par des lignes avec une flèche indiquant la direction du flux d'énergie.
- Les nœuds 0 et 1 sont représentés par les chiffres correspondants entourés.
- Les éléments de stockage et de dissipation sont symbolisés par des rectangles avec une étiquette indiquant leur type (C, I, R).

### 1.3 Causalité dans les Bond Graphs

La causalité dans un Bond Graph désigne la direction dans laquelle l'effort et le flux sont déterminés. Elle est indiquée par une barre sur le lien, qui pointe vers le composant qui détermine l'effort.

## 2. Qu'est-ce qu'un Graphe Informationnel Causale (GIC) ?

Les Graphes Informationnels Causales (GIC) sont des représentations graphiques qui décrivent les relations de causalité entre les variables dans un système. Ils sont souvent utilisés pour analyser les systèmes de contrôle et les circuits électriques complexes, permettant de suivre comment les signaux se propagent et influencent les différents composants.

### 2.1 Structure des Graphes Informationnels Causales

- **Nœuds** : Représentent les variables ou les états du système.
- **Arcs** : Représentent les relations causales entre les variables, indiquant comment une variable influence ou est influencée par une autre.

### 2.2 Utilisation des GIC dans les Circuits Électriques

Les GIC peuvent être utilisés pour :

- Identifier les chemins de transmission des signaux.
- Analyser la propagation des perturbations dans les systèmes.
- Développer des schémas de contrôle pour réguler le comportement du système.

## 3. Application aux Circuits Électriques

### 3.1 Exemple 1 : Circuit RLC Série

**Description du Système** : Un circuit RLC série avec une source de tension, une résistance ($R$), une inductance ($L$), et un condensateur ($C$).

1. **Modèle Bond Graph** :
   - Le lien entre la source de tension et la résistance représente le courant circulant ($I$).
   - Un nœud 0 pour la somme des tensions aux bornes des composants.
   - Liens entre chaque composant pour représenter l'effort et le flux d'énergie.

- **Nœud 1** (courant commun) entre $R$, $L$, et $C$.
- **Nœud 0** pour la tension totale.

2. **Causalité** :
   - La causalité est définie par la position des barres de causalité sur les bonds. Par exemple, la barre peut être positionnée sur le côté du lien du composant qui impose la tension (effort).

### 3.2 Exemple 2 : Amplificateur Opérationnel

**Description du Système** : Un amplificateur opérationnel en configuration inverseur avec résistance d'entrée ($R_{\text{in}}$) et résistance de rétroaction ($R_f$).

1. **Modèle Bond Graph** :
   - Liens pour les courants traversant $R_{\text{in}}$ et $R_f$.
   - Nœud 1 pour le courant commun à l'entrée de l'amplificateur.
   - Nœud 0 pour la tension de sortie.

   *Note: Image placeholder*

2. **Causalité** :
   - La causalité est définie en fonction des courants et des tensions imposés par l'amplificateur opérationnel.

.

### 3.3 Exemple 3 : Circuit avec Transformateur Couplé

**Description du Système** : Un transformateur avec enroulements primaire et secondaire, chacun avec des résistances et inductances de fuite.

1. **Modèle Bond Graph** :
   - Liens pour les courants primaire et secondaire.
   - Nœud 1 pour le courant commun du transformateur.
   - Liens représentant les inductances de fuite et le couplage magnétique.

2. **Causalité** :
   - La causalité est déterminée par les effets inductifs et résistifs dans les enroulements primaire et secondaire.

## 4. Outils de Modélisation : Comment Utiliser les Bond Graphs et les GIC

### 4.1 Étapes pour Créer un Bond Graph

1. **Déterminer les Composants et les Interactions** : Identifier tous les composants du système et comment ils interagissent.
2. **Tracer les Bonds** : Dessiner les liens qui montrent les flux d'énergie entre les composants.
3. **Placer les Nœuds** : Ajouter des nœuds 0 et 1 pour représenter les points de somme d'effort et de flux.

4. **Indiquer la Causalité** : Déterminer la direction de l'effort et du flux pour chaque lien.

### 4.2 Utilisation des GIC

1. **Identifier les Variables** : Lister toutes les variables d'état et de sortie du système.
2. **Tracer les Arcs** : Dessiner les arcs qui montrent les relations causales entre les variables.
3. **Analyser le Graphe** : Utiliser le GIC pour comprendre comment les changements dans une variable affectent le système global.

## 5. Exemples de Modélisation avec Matlab/Simulink

### Exemple 1 : Simuler un Circuit RLC avec Bond Graphs

1. **Configuration du Modèle** : Utiliser des blocs Simulink pour représenter les résistances, inductances, et condensateurs. Utiliser des blocs de flux et d'effort pour représenter les bonds.
2. **Code Matlab** : Simuler la réponse temporelle et en fréquence du circuit.

```
% Paramètres du circuit
R = 10; % Résistance (ohms)
L = 1e-3; % Inductance (henrys)
C = 100e-6; % Capacité (farads)
 % Fonctions de transfert
num = [1];
den = [L C R];
sys = tf(num, den);
 % Réponse en fréquence
figure;
bode(sys);
title('Réponse en fréquence du circuit RLC');
% Réponse temporelle à une entrée en échelon
figure;
step(sys);
title('Réponse temporelle du circuit RLC');
```

3. **Simulation Simulink** : Utiliser des blocs pour modéliser les éléments RLC et visualiser les résultats avec un Scope.

### Exemple 2 : Analyse d'un Amplificateur avec GIC

1. **Tracer le GIC** : Dessiner le graphe en identifiant les relations causales entre les variables d'entrée et de sortie.
2. **Utiliser Simulink** : Configurer un modèle d'amplificateur avec des blocs pour chaque composant et visualiser la réponse en temps réel.

### Conclusion

Les Bond Graphs et les Graphes Informationnels Causales sont des outils puissants pour modéliser, analyser, et comprendre les systèmes dynamiques complexes. En utilisant ces outils, les ingénieurs peuvent développer des modèles précis et efficaces, facilitant la conception, le contrôle, et l'optimisation de systèmes multidisciplinaires. L'application aux circuits électriques montre comment ces méthodes peuvent être employées pour résoudre des problèmes concrets, tout en offrant une vue systématique et cohérente des interactions énergétiques au sein des systèmes

# Applications : Outils de Modélisation (Bond Graph (BG) et Graphes Informationnels Causales (GIC))

**Exercice 1 : Modélisation d'un Circuit RLC Série avec Bond Graph**

**Objectif**

Modéliser un circuit RLC en série en utilisant un Bond Graph et analyser la réponse en fréquence et la réponse temporelle.

## Description du Système

- **Composants** : Résistance ($R$), Inductance ($L$), Condensateur ($C$).
- **Entrée** : Source de tension AC ($V_{in}$).
- **Sortie** : Tension aux bornes du condensateur ($V_C$).

## Étapes de la Modélisation

1. **Identifier les Composants et les Flux d'Énergie** :
   - Source de tension AC $V_{in}$ fournit l'énergie au circuit.
   - L'énergie circule à travers $R$, $L$, et $C$.
2. **Créer le Bond Graph** :
   - Utiliser un nœud 1 pour représenter la somme des courants dans le circuit.
   - Ajouter des liens pour la résistance (R), l'inductance (I), et le condensateur (C).
3. **Schéma du Bond Graph** :
   - Le lien entre $V_{in}$ et $R$ représente le courant.
   - Nœud 1 pour la somme des courants.
   - Liens de causalité définis par les positions des barres de causalité.

```
mathematica
      Se -- R -- 1 -- I -- C
                |
                0
                |
                V_out
```

*Code Matlab :*

```
matlab
Copier le code
R = 10; % Résistance (ohms)
L = 1e-3; % Inductance (henrys)
C = 100e-6; % Capacité (farads)

% Fonction de transfert du circuit RLC
num = [1];
den = [L C R];
sys = tf(num, den);

% Réponse en fréquence
figure;
bode(sys);
```

```
title('Réponse en fréquence du circuit RLC');

% Réponse temporelle à une entrée en échelon
figure;
step(sys);
title('Réponse temporelle du circuit RLC');
```

**Analyse**

- **Réponse en Fréquence** : Analyser les fréquences de résonance et d'amortissement.
- **Réponse Temporelle** : Voir comment le circuit réagit à une tension appliquée.

**Exercice 2 : Modélisation d'un Amplificateur Opérationnel en Configuration Inverseur avec Bond Graph**

**Objectif**

Modéliser un amplificateur opérationnel en configuration inverseur en utilisant un Bond Graph et analyser le gain et la réponse en fréquence.

## Description du Système

- **Composants** : Amplificateur opérationnel, Résistance d'entrée ($R_{in}$), Résistance de rétroaction ( $R_f$).
- **Entrée** : Signal d'entrée ($V_{in}$).
- **Sortie** : Signal amplifié et inversé ($V_{out}$).

## Étapes de la Modélisation

1. **Identifier les Composants et les Flux d'Énergie** :
   - Amplificateur opérationnel comme source d'effort.
   - Résistances $R_{in}$ et $R_f$ pour le contrôle du flux d'énergie.
2. **Créer le Bond Graph** :
   - Utiliser des liens pour les courants traversant $R_{in}$ et $R_f$.
   - Nœud 1 pour le courant commun à l'entrée de l'amplificateur.
   - Nœud 0 pour la somme des tensions.
3. **Schéma du Bond Graph** :

```
Se -- R_in -- 1 -- R_f -- 0 -- V_out
                  |
                  0
                  |
                  Op-Amp
```

1. **Simuler avec Matlab/Simulink** :

Utiliser des blocs pour simuler la réponse en fréquence et la réponse temporelle.

**Code Matlab** :

```
R_in = 1e3; % Résistance d'entrée (ohms)
R_f = 10e3; % Résistance de rétroaction (ohms)

% Fonction de transfert de l'amplificateur inverseur
num = [-R_f/R_in];
den = [1];
sys = tf(num, den);

% Réponse en fréquence
figure;
bode(sys);
title('Réponse en fréquence de l\'amplificateur
inverseur');

% Réponse temporelle à une entrée en échelon
figure;
step(sys);
title('Réponse temporelle de l\'amplificateur
inverseur');
```

**Analyse**

- **Gain en Tension** : Analyser l'amplification et l'inversion du signal.
- **Réponse en Fréquence** : Examiner la stabilité et la bande passante.

**Exercice 3 : Modélisation d'un Transformateur Couplé avec Bond Graph**

**Objectif**

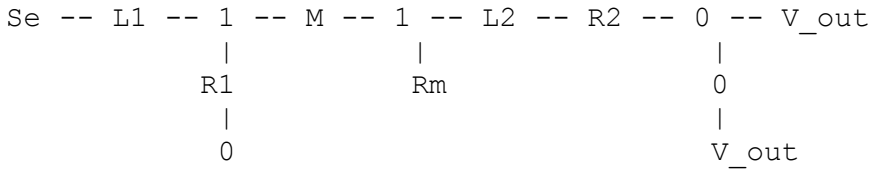
Modéliser un transformateur couplé avec des enroulements primaire et secondaire et analyser l'efficacité de transmission et les effets de couplage.

### Description du Système

- **Composants** : Enroulements primaire et secondaire avec résistances et inductances.
- **Entrée** : Tension primaire AC ($V_{in}$).
- **Sortie** : Tension secondaire AC ($V_{out}$).

### Étapes de la Modélisation

1. **Identifier les Composants et les Flux d'Énergie** :
   - Enroulements couplés via l'inductance mutuelle $M$.
   - Les résistances et inductances de fuite modélisent les pertes.
2. **Créer le Bond Graph** :
   - Utiliser des liens pour les courants primaire et secondaire.
   - Nœud 1 pour le courant commun du transformateur.
   - Nœud 0 pour les tensions.
3. **Schéma du Bond Graph** :




1. **Simuler avec Matlab/Simulink** :
   - Utiliser des blocs pour modéliser les enroulements et les couplages.
   - **Code Matlab** :

```
L1 = 1e-3; % Inductance primaire (henrys)
L2 = 2e-3; % Inductance secondaire (henrys)
M = 0.9e-3; % Inductance mutuelle (henrys)

% Fonction de transfert du transformateur couplé
num = [M];
den = [L1 L2];
sys = tf(num, den);

% Réponse en fréquence
figure;
bode(sys);
title('Réponse en fréquence du transformateur couplé');

% Réponse temporelle à une entrée en échelon
figure;
step(sys);
title('Réponse temporelle du transformateur couplé');
```

**Analyse**

- **Transmission d'Énergie** : Analyser l'efficacité du transformateur.
- **Effets de Couplage** : Voir comment le couplage influence les tensions primaire et secondaire.

**Exercice 4 : Modélisation d'un Circuit de Puissance avec Couplage Magnétique Utilisant GIC**

**Objectif**

Utiliser des Graphes Informationnels Causales (GIC) pour modéliser un circuit de puissance avec couplage magnétique et analyser l'atténuation des harmoniques.

## Description du Système

- **Composants** : Réseau de filtrage avec inductances couplées et condensateurs.
- **Entrée** : Signal de puissance ($V_{in}$).
- **Sortie** : Signal de puissance filtré ($V_{out}$).

## Étapes de la Modélisation

1. **Identifier les Variables** : Variables de tension et de courant aux différents points du circuit.
2. **Tracer le GIC** :
   - Nœuds pour les variables de tension et de courant.
   - Arcs pour représenter les relations causales (comment une variable affecte une autre).
3. **Schéma GIC** :

```
Copier le code
V_in --> I_L1 --> V_L1 --> I_L2 --> V_L2 --> V_out
```

1. **Simuler avec Matlab/Simulink** :
   - Utiliser des blocs pour modéliser les éléments de couplage et les inductances.
   - **Code Matlab** :

```
L1 = 1e-3; % Inductance 1 (henrys)
L2 = 1e-3; % Inductance 2 (henrys)
M = 0.5e-3; % Inductance mutuelle (henrys)

% Fonction de transfert pour le circuit couplé
num = [M];
den = [L1 L2 M];
sys = tf(num, den);

% Réponse en fréquence
figure;
bode(sys);
title('Réponse en fréquence du circuit de filtrage couplé');

% Réponse temporelle à une entrée en échelon
figure;
step(sys);
title('Réponse temporelle du circuit de filtrage couplé');
```

**Analyse**

- **Qualité de Puissance** : Analyser l'atténuation des harmoniques et la réduction du bruit.
- **Effet de Couplage** : Étudier comment les inductances couplées influencent la réponse du circuit.

**Conclusion**

Ces exercices montrent comment utiliser les Bond Graphs et les Graphes Informationnels Causales pour modéliser et analyser des systèmes électriques complexes. En suivant ces étapes, les ingénieurs peuvent mieux comprendre les interactions énergétiques et les relations causales dans les systèmes multidisciplinaires, ce qui est essentiel pour le contrôle, la conception, et l'optimisation des systèmes.

# Chapitre 5

## Généralités sur l'Identification des Systèmes

### Introduction

L'identification des systèmes est une discipline qui consiste à construire des modèles mathématiques à partir des données mesurées. Cela est essentiel dans le domaine du contrôle et de l'automatique pour concevoir des régulateurs efficaces, optimiser des performances, et comprendre le comportement dynamique des systèmes physiques. Ce chapitre aborde les concepts fondamentaux de l'identification, les différentes structures de modèles, et les méthodes pour ajuster ces modèles aux données expérimentales.

### 1. Définitions et Concepts de Base

#### 1.1 Qu'est-ce que l'Identification des Systèmes?

L'identification des systèmes est le processus d'élaboration d'un modèle mathématique représentant le comportement d'un système physique en se basant sur des données expérimentales. Elle implique les étapes suivantes :

- **Collecte des Données** : Mesurer les signaux d'entrée et de sortie du système.
- **Choix de la Structure du Modèle** : Décider de la forme mathématique du modèle (AR, ARMA, ARMAX, etc.).
- **Estimation des Paramètres** : Ajuster les paramètres du modèle pour qu'il reproduise le comportement observé.
- **Validation du Modèle** : Vérifier que le modèle prédit correctement le comportement du système pour des données non utilisées dans l'estimation.

#### 1.2 Pourquoi Identifier un Système?

L'identification est nécessaire pour :

- **Modélisation** : Créer des représentations mathématiques précises pour la simulation et l'analyse.
- **Contrôle** : Concevoir des régulateurs qui nécessitent une connaissance précise du système.
- **Prédiction** : Prévoir le comportement futur du système sous différentes conditions.
- **Diagnostic** : Détecter et diagnostiquer des anomalies ou des défaillances potentielles.

#### 1.3 Modèles Utilisés en Identification

Les modèles utilisés dans l'identification peuvent être classés en différentes catégories selon leur structure et leur complexité :

- **Modèles à Réponse Impulsionnelle** : Représentent la réponse du système à une entrée impulsionnelle.
- **Modèles à Réponse en Fréquence** : Décrivent la réponse du système en fonction de la fréquence d'entrée.
- **Modèles de Systèmes Linéaires à Paramètres Concentrés** :
    - **Modèles Auto-Régressifs (AR)** : Dépendent uniquement des valeurs passées de la sortie.
    - **Modèles Auto-Régressifs avec Moyenne Mobile (ARMA)** : Dépendent des valeurs passées de la sortie et de l'entrée.

- **Modèles Auto-Régressifs avec Moyenne Mobile et Entrée Exogène (ARMAX)** : Incluent des termes d'entrée exogène.
- **Modèles d'Erreur de Sortie (OE)** : Basés sur l'erreur de prédiction.

## 2. Étapes de l'Identification

### 2.1 Collecte et Prétraitement des Données

La première étape de l'identification des systèmes est la collecte des données expérimentales. Cela implique la mesure des signaux d'entrée et de sortie du système sous différentes conditions d'opération. Les données doivent être de haute qualité et représentatives du comportement du système.

**Prétraitement des Données** :

- **Filtrage** : Élimination du bruit.
- **Normalisation** : Ajustement des échelles des données.
- **Traitement des Données Manquantes** : Imputation ou exclusion des données manquantes.
- **Échantillonnage** : Choix d'une fréquence d'échantillonnage appropriée pour capturer les dynamiques du système.

### 2.2 Choix de la Structure du Modèle

Le choix de la structure du modèle est crucial car il doit capturer correctement la dynamique du système tout en restant aussi simple que possible. Voici les structures de modèles les plus courantes :

- **Modèles AR (Auto-Régressifs)** : Conviennent pour des systèmes où la sortie dépend fortement de ses valeurs passées. Un modèle AR est donné par :

$$y(t) = a_1 y(t-1) + a_2 y(t-2) + \ldots + a_n y(t-n) + e(t)$$

où $y(t)$ est la sortie à l'instant $t$, $a_i$ sont les coefficients du modèle, et $e(t)$ est un terme d'erreur.

- **Modèles MA (Moyenne Mobile)** : Conviennent pour des systèmes influencés par des chocs aléatoires. Un modèle MA est donné par :

$$y(t) = e(t) + b_1 e(t-1) + b_2 e(t-2) + \ldots + b_m e(t-m)$$

où $e(t)$ représente le bruit blanc ou le terme d'erreur.

- **Modèles ARMA (Auto-Régressifs Moyenne Mobile)** : Combinent les aspects des modèles AR et MA. Ils conviennent pour des systèmes où la sortie dépend à la fois de ses valeurs passées et des chocs aléatoires :

$$y(t) = a_1 y(t-1) + \ldots + a_n y(t-n) + e(t) + b_1 e(t-1) + \ldots + b_m e(t-m)$$

- **Modèles ARMAX (Auto-Régressifs Moyenne Mobile avec Entrée Exogène)** : Incluent les effets des entrées externes (input-output) :

$$y(t) = a_1 y(t-1) + \ldots + a_n y(t-n) + b_1 u(t-1) + \ldots + b_m u(t-m) + e(t) + c_1 e(t-1) + \ldots + c_p e(t-p)$$

où $u(t)$ représente les entrées exogènes.

.

### 2.3 Estimation des Paramètres

L'étape suivante consiste à estimer les paramètres du modèle choisis. Cette estimation se fait généralement par minimisation d'un critère d'ajustement, tel que l'erreur quadratique moyenne entre les données observées et les valeurs prédites par le modèle.

**Méthode des Moindres Carrés (LS)** : Technique couramment utilisée pour minimiser la .

$$J(\theta) = \sum_{t=1}^{N} (y(t) - \hat{y}(t|\theta))^2$$

où $\hat{y}(t|\theta)$ est la sortie prédite par le modèle avec les paramètres $\theta$.

**Méthode du Maximum de Vraisemblance (ML)** : Cherche à maximiser la probabilité des observations données les paramètres du modèle.

### 2.4 Validation du Modèle

Après l'estimation des paramètres, il est essentiel de valider le modèle pour s'assurer qu'il capture correctement la dynamique du système.

- **Validation Croisée** : Diviser les données en ensembles d'apprentissage et de test.
- **Analyse des Résidus** : Examiner les résidus $e(t) = y(t) - \hat{y}(t)$ pour vérifier l'absence de structure non expliquée.
- **Test de Performance** : Utiliser des métriques comme l'erreur quadratique moyenne (MSE), l'erreur absolue moyenne (MAE), ou le coefficient de détermination ($R^2$).

## 3. Génération SBPA (Séquence Binaire Pseudo-Aleatoire)

La SBPA est une méthode pour générer des signaux d'entrée aléatoires utilisés dans les expériences d'identification. Ces signaux sont choisis pour exciter toutes les dynamiques pertinentes du système, garantissant que le modèle capturera tous les comportements significatifs.

### 3.1 Principe de la SBPA

- **Caractéristiques** : La SBPA est un signal binaire, ayant une amplitude de +1 ou -1, et est conçu pour avoir une longue période avant de se répéter.
- **Utilisation** : Les SBPA sont utilisées pour perturber le système de manière contrôlée et obtenir une large couverture des fréquences.

### 3.2 Exemple de Génération SBPA avec Matlab

```
% Génération d'un signal SBPA
n = 1000; % Nombre de points
sbpa = idinput(n, 'prbs', [0 1], [-1 1]); % SBPA entre -1 et 1

% Affichage du signal SBPA
figure;
plot(sbpa);
title('Signal SBPA');
```

```
xlabel('Temps');
ylabel('Amplitude');
```

## 4. Rappel des Méthodes de Base en Automatique

### 4.1 Réponse Temporelle d'un Système

La réponse temporelle d'un système est la sortie du système en fonction du temps lorsqu'il est soumis à une entrée particulière. Analyser la réponse temporelle permet de comprendre les dynamiques internes du système, telles que le temps de montée, le temps de stabilisation, et les dépassements.

- **Réponse à un Échelon** : Souvent utilisée pour caractériser la dynamique d'un système.
- **Réponse à une Impulsion** : Permet d'étudier la réponse transitoire.

### 4.2 Approche Fréquentielle

L'analyse fréquentielle examine la réponse d'un système aux signaux sinusoïdaux de différentes fréquences. Cette approche est utile pour étudier les systèmes dans le domaine de la fréquence.

- **Diagramme de Bode** : Représentation graphique de la réponse en fréquence.
- **Gain et Phase** : Les caractéristiques importantes à évaluer sont le gain et le déphasage à chaque fréquence.

### 4.3 Identification Directe à partir des Réponses Temporelle et Fréquentielle

**Méthode des Moindres Carrés** pour ajuster un modèle à des données de réponse temporelle :

- Utiliser une entrée connue et observer la sortie.
- Ajuster les paramètres du modèle pour minimiser l'erreur entre la sortie observée et la sortie prédite.

**Méthode de Variable Instrumentale** pour traiter les systèmes avec bruit :

- Utilise des variables instrumentales (corrélées avec les variables explicatives, mais non corrélées avec l'erreur).

## 5. Principe d'Ajustement du Modèle

### 5.1 Modèle Linéaire par rapport aux Paramètres

Dans l'identification des systèmes, les modèles linéaires par rapport aux paramètres sont ceux où les équations du modèle sont linéaires par rapport aux paramètres à estimer. Ces modèles sont plus faciles à estimer car les techniques d'estimation linéaire, comme les moindres carrés, peuvent être utilisées directement.

### 5.2 Minimisation du Critère d'Ajustement

La minimisation du critère d'ajustement est la méthode principale pour ajuster un modèle aux données observées. Le critère le plus couramment utilisé est la somme des carrés des erreurs (SSE) entre les observations et les prédictions du modèle.

- **Formulation du Problème** :

$$\text{Minimiser } J(\theta) = \sum_{t=1}^{N} (y(t) - \hat{y}(t|\theta))^2$$

- **Solution Optimale** : Trouver les paramètres $\theta$ qui minimisent $J(\theta)$.

### 5.3 Exemple : Ajustement d'un Modèle ARX avec Matlab

```
% Données simulées
t = (0:0.01:10)'; % Temps
u = sin(t); % Entrée
y = 2*u + 0.5*randn(length(u),1); % Sortie avec du bruit

% Construction du modèle ARX
na = 1; % Ordre de l'auto-régressif
nb = 1; % Ordre de l'entrée
nk = 1; % Délai

% Identification du modèle
model = arx([y, u], [na nb nk]);

% Affichage des résultats
disp(model);
```

### Validation du Modèle

- **Analyse des Résidus** : Vérifier que les résidus (différences entre les valeurs observées et prédites) sont non corrélés et suivent une distribution normale.

- **Critères de Validation** :
  - **MSE (Mean Squared Error)** : Mesure la moyenne des carrés des erreurs.
  - $R^2$ : Coefficient de détermination qui mesure la proportion de variance expliquée par le modèle.

### .Conclusion

L'identification des systèmes est un domaine essentiel en automatique, permettant de créer des modèles précis à partir de données expérimentales. Ce chapitre a couvert les définitions clés, les structures de modèles, les méthodes d'estimation, et les techniques de validation. En comprenant ces concepts, les ingénieurs peuvent mieux concevoir et contrôler des systèmes dynamiques complexes, améliorant ainsi les performances et la fiabilité des applications industrielles

# Applications : Identification des Systèmes

**Introduction**

L'identification des systèmes est essentielle pour modéliser le comportement de systèmes physiques à partir de données expérimentales. Les outils utilisés comprennent la génération de séquences binaires pseudo-aléatoires (SBPA), le choix de structures de modèles appropriées (comme AR, ARMA, ARMAX), l'analyse des réponses temporelles et fréquentielles, ainsi que l'utilisation de méthodes de variable instrumentale. Ce chapitre propose des exercices détaillés pour chacune de ces méthodes, avec des explications pas à pas et des exemples pratiques.

**1. Génération SBPA (Séquence Binaire Pseudo-Aleatoire)**

**Objectif**

Utiliser une SBPA pour exciter un système et collecter des données représentatives pour l'identification du modèle.

**Exercice 1.1 : Génération d'une SBPA avec Matlab**

**Étape 1 : Définir les paramètres de la SBPA**

- Longueur de la séquence : n=1000n = 1000n=1000
- Amplitude : entre -1 et 1

**Étape 2 : Générer la SBPA**

```
n = 1000; % Nombre de points
sbpa = idinput(n, 'prbs', [0 1], [-1 1]); % SBPA entre -1 et 1

% Affichage du signal SBPA
figure;
plot(sbpa);
title('Signal SBPA');
xlabel('Temps');
ylabel('Amplitude');
```

**Analyse :** Le signal généré alterne de manière aléatoire entre -1 et 1, garantissant une excitation suffisante du système pour capturer toutes ses dynamiques.

**Exercice 1.2 : Utilisation de la SBPA pour Exciter un Système RC**

**Étape 1 : Définir un circuit RC avec une résistance de R=10 ΩR = 10 \, \OmegaR=10Ω et une capacité de C=1 µFC = 1 \, \mu FC=1µF.**

**Étape 2 : Simuler le système avec une SBPA comme entrée.**

```
R = 10; % Résistance en ohms
C = 1e-6; % Capacité en farads

sys = tf([1], [R*C 1]); % Fonction de transfert du circuit RC

% Réponse du système à la SBPA
```

```
y = lsim(sys, sbpa, (0:n-1));

% Affichage de la réponse
figure;
plot(y);
title('Réponse du Circuit RC à la SBPA');
xlabel('Temps');
ylabel('Tension de Sortie');
```

**Analyse :** La réponse du système RC à la SBPA montre comment le système réagit à des changements rapides d'entrée, ce qui est essentiel pour identifier des dynamiques rapides.

**2. Choix de la Structure du Modèle (AR, ARMA, ARMAX)**

**Objectif**

Choisir la structure de modèle la plus adaptée pour représenter les dynamiques d'un système.

**Exercice 2.1 : Identification d'un Modèle AR**

**Étape 1 : Collecter des données de sortie d'un système, supposons une réponse temporelle mesurée.**

**Étape 2 : Utiliser un modèle AR pour représenter la sortie.**

```
data = y;
% Utiliser la réponse du système RC comme données
na = 2; % Ordre du modèle AR

model_AR = ar(data, na);

% Affichage des résultats
disp('Paramètres du Modèle AR:');
disp(model_AR.A);
```

**Analyse :** Le modèle AR utilise uniquement les valeurs passées de la sortie pour prédire les valeurs futures. L'ordre du modèle est choisi pour capturer les dynamiques observées dans les données.

**Exercice 2.2 : Identification d'un Modèle ARMA**

**Étape 1 : Collecter des données d'entrée et de sortie d'un système.**

**Étape 2 : Utiliser un modèle ARMA pour capturer à la fois les valeurs passées de la sortie et l'effet des chocs aléatoires.**

```
nb = 2; % Ordre de la partie MA

model_ARMA = armax(data, [na nb]);

% Affichage des résultats
disp('Paramètres du Modèle ARMA:');
disp(model_ARMA.A);
disp(model_ARMA.C);
```

**Analyse :** Le modèle ARMA est plus complet que le modèle AR car il prend en compte l'effet des erreurs passées, ce qui est crucial pour les systèmes influencés par des perturbations aléatoires.

**Exercice 2.3 : Identification d'un Modèle ARMAX**

**Étape 1 : Utiliser des données d'entrée et de sortie d'un système avec une entrée exogène.**

**Étape 2 : Appliquer un modèle ARMAX qui inclut l'effet des entrées.**

```
u = sbpa; % Utiliser la SBPA comme entrée
data = [y u]; % Données d'entrée-sortie
nk = 1; % Délai entre entrée et sortie

model_ARMAX = armax(data, [na nb nk]);

% Affichage des résultats
disp('Paramètres du Modèle ARMAX:');
disp(model_ARMAX.A);
disp(model_ARMAX.B);
disp(model_ARMAX.C);
```

**Analyse :** Le modèle ARMAX est adapté pour des systèmes où l'entrée a un effet direct sur la sortie. Il est utilisé dans les applications de contrôle où il est crucial de modéliser la réponse du système aux commandes.

**3. Réponse Temporelle et Approche Fréquentielle**

**Objectif**

Analyser la réponse temporelle et fréquentielle d'un système pour identifier ses dynamiques.

**Exercice 3.1 : Réponse Temporelle d'un Système du Premier Ordre**

**Étape 1 : Définir un système du premier ordre avec une constante de temps $\tau$=1**

**Étape 2 : Simuler la réponse à une entrée échelon.**

```
tau = 1; % Constante de temps
sys = tf([1], [tau 1]); % Système du premier ordre

% Réponse à une entrée échelon
figure;
step(sys);
title('Réponse Temporelle du Système du Premier Ordre');
xlabel('Temps');
ylabel('Sortie');
```

**Analyse :** La réponse à une entrée échelon montre comment le système atteint un nouvel état d'équilibre. La constante de temps $\tau$\tau$\tau$ détermine la rapidité de la réponse.

**Exercice 3.2 : Réponse Fréquentielle d'un Système du Deuxième Ordre**

**Étape 1 : Définir un système du deuxième ordre avec un facteur d'amortissement $\zeta$=0.5 et une fréquence naturelle $\omega$n=2.**

**Étape 2 : Analyser la réponse en fréquence.**

```
zeta = 0.5; % Facteur d'amortissement
omega_n = 2; % Fréquence naturelle
sys = tf([omega_n^2], [1 2*zeta*omega_n omega_n^2]); % Système du deuxième ordre

% Réponse en fréquence
figure;
bode(sys);
title('Réponse Fréquentielle du Système du Deuxième Ordre');
```

**Analyse :** La réponse en fréquence montre comment le système réagit aux différentes fréquences d'entrée, révélant des informations sur la bande passante et les résonances.

**Exercice 3.3 : Réponse Temporelle d'un Système d'Ordre Supérieur**

**Étape 1 : Définir un système d'ordre supérieur avec plusieurs pôles.**

**Étape 2 : Simuler la réponse à une entrée échelon.**

```
sys = tf([1], [1 3 3 1]); % Système d'ordre supérieur

% Réponse à une entrée échelon
figure;
step(sys);
title('Réponse Temporelle du Système d\'Ordre Supérieur');
xlabel('Temps');
ylabel('Sortie');
```

**Analyse :** Les systèmes d'ordre supérieur peuvent avoir des comportements complexes, comme des oscillations et des dépassements, en fonction des emplacements des pôles.

**4. Méthode de Variable Instrumentale**

**Objectif**

Utiliser la méthode de variable instrumentale pour identifier un modèle lorsqu'il y a du bruit de mesure.

**Exercice 4.1 : Utilisation des Variables Instrumentales**

**Étape 1 : Générer des données d'un système avec du bruit de mesure.**

**Étape 2 : Utiliser une variable instrumentale pour estimer un modèle.**

```
% Génération de données avec bruit
y = 2*u + 0.5*randn(length(u),1); % Sortie avec bruit
z = iddata(y, u); % Créer un objet de données d'identification

% Identification avec variable instrumentale
model_VI = iv4(z, [na nb nk]);

% Affichage des résultats
```

```
disp('Paramètres du Modèle avec Variable Instrumentale:');
disp(model_VI.A);
disp(model_VI.B);
```

**Analyse :** La méthode de variable instrumentale aide à réduire l'impact du bruit sur les estimations des paramètres, améliorant ainsi la précision du modèle.

**5. Principe d'Ajustement du Modèle**

**Objectif**

Ajuster les paramètres d'un modèle pour minimiser un critère d'ajustement.

**Exercice 5.1 : Ajustement d'un Modèle Linéaire**

**Étape 1 : Définir un modèle linéaire avec des paramètres initiaux.**

**Étape 2 : Utiliser la méthode des moindres carrés pour ajuster le modèle.**

```
% Données simulées
t = (0:0.01:10)'; % Temps
u = sin(t); % Entrée
y = 2*u + 0.5*randn(length(u),1); % Sortie avec du bruit

% Estimation des paramètres avec moindres carrés
theta = inv(u'*u)*u'*y; % Paramètres estimés

% Affichage des résultats
disp('Paramètres Estimés:');
disp(theta);
```

**Analyse :** La méthode des moindres carrés est utilisée pour trouver les paramètres qui minimisent l'erreur entre les valeurs observées et prédites. Cette méthode est efficace pour les modèles linéaires.

**Conclusion**

Ces exercices couvrent des aspects clés de l'identification des systèmes, y compris la génération de signaux d'excitation, le choix de la structure du modèle, l'analyse des réponses temporelles et fréquentielles, l'utilisation de variables instrumentales, et l'ajustement des modèles. En pratiquant ces exercices, les étudiants peuvent acquérir une compréhension approfondie des techniques d'identification et de modélisation, essentielles pour la conception et l'analyse des systèmes dynamiques en ingénierie.

# Applications : l'Identification des Systèmes Couplés et Multivariables pour des Systèmes Réels

**Introduction**

L'identification des systèmes couplés et multivariables est cruciale pour modéliser des systèmes réels complexes dans lesquels plusieurs variables interagissent simultanément. Cela est pertinent dans de nombreux domaines d'ingénierie, tels que l'automobile, l'aéronautique, la chimie et l'électronique. Ce document présente des exercices détaillés pour comprendre et appliquer les techniques d'identification des systèmes, y compris la génération de signaux d'excitation, le choix des structures de modèles appropriées, l'analyse des réponses temporelles et fréquentielles, l'utilisation de méthodes de variable instrumentale, et l'ajustement des modèles pour des systèmes réels.

**1. Génération de Séquences Binaires Pseudo-Aléatoires (SBPA)**

**Objectif**

Utiliser des SBPA pour exciter des systèmes réels multivariables et collecter des données représentatives pour l'identification des modèles.

**Exercice 1.1 : Génération de SBPA pour un Système de Contrôle Température et Humidité**

**Étape 1 : Définir les paramètres de la SBPA**

- Longueur de la séquence : n=1500
- Amplitude : entre -1 et 1
- Nombre de canaux : 2 (un pour la température, un pour l'humidité)

**Étape 2 : Générer la SBPA**

```
n = 1500; % Nombre de points
num_channels = 2; % Nombre de canaux pour le système multivariable
sbpa = idinput([n, num_channels], 'prbs', [0 1], [-1 1]); % SBPA
entre -1 et 1

% Affichage du signal SBPA pour les deux canaux
figure;
subplot(2,1,1);
plot(sbpa(:,1));
title('Signal SBPA - Canal Température');
xlabel('Temps');
ylabel('Amplitude');

subplot(2,1,2);
plot(sbpa(:,2));
title('Signal SBPA - Canal Humidité');
xlabel('Temps');
ylabel('Amplitude');
```

**Analyse :** Le signal SBPA est utilisé pour simuler des variations aléatoires dans le système de contrôle de la température et de l'humidité, fournissant une excitation suffisante pour capturer toutes les dynamiques du système.

**Exercice 1.2 : Utilisation de la SBPA pour Exciter un Système Réel de Commande d'un Robot Industriel**

**Étape 1 : Définir un système couplé où deux actionneurs contrôlent les mouvements d'un bras robotique.**

- Chaque actionneur reçoit une entrée indépendante (via SBPA) qui influence le mouvement en x et y.

**Étape 2 : Simuler le système avec une SBPA comme entrée.**

```
% Modèle de système robotique couplé
A = [0.8 0.1; 0.2 0.7];
B = [1 0; 0 1];
C = [1 0; 0 1];
D = [0 0; 0 0];
sys = ss(A, B, C, D);

% Réponse du système à la SBPA
y = lsim(sys, sbpa, (0:n-1)');

% Affichage de la réponse
figure;
subplot(2,1,1);
plot(y(:,1));
title('Réponse du Robot - Axe X');
xlabel('Temps');
ylabel('Position');

subplot(2,1,2);
plot(y(:,2));
title('Réponse du Robot - Axe Y');
xlabel('Temps');
ylabel('Position');
```

**Analyse :** La réponse du système de commande du robot montre comment les positions xxx et yyy réagissent aux variations d'entrée, révélant les interactions entre les commandes des actionneurs.

**2. Choix de la Structure du Modèle (AR, ARMA, ARMAX)**

**Objectif**

Choisir et identifier des structures de modèles adaptées pour des systèmes réels couplés et multivariables.

**Exercice 2.1 : Identification d'un Modèle AR pour un Système de Chauffage à Deux Zones**

**Étape 1 : Collecter des données de température pour deux zones d'un bâtiment.**

- Les deux zones sont chauffées indépendamment, mais l'échange thermique entre elles influence les températures.

**Étape 2 : Utiliser un modèle AR pour représenter les températures.**

```
% Données simulées pour les températures de deux zones
zone1_temp = 20 + 2*randn(1500,1); % Température avec du bruit
zone2_temp = 22 + 1.5*randn(1500,1); % Température avec du bruit
data = [zone1_temp zone2_temp];

na = 2; % Ordre du modèle AR

% Modèle AR multivariable
model_AR = ar(data, na, 'ls'); % 'ls' pour méthode des moindres
carrés

% Affichage des résultats
disp('Paramètres du Modèle AR Multivariable (Chauffage à Deux
Zones):');
disp(model_AR.A);
```

**Analyse :** Le modèle AR multivariable prédit les températures des deux zones en fonction de leurs valeurs passées, capturant ainsi l'effet de l'échange thermique entre les zones.

**Exercice 2.2 : Identification d'un Modèle ARMA pour un Système de Commande de Vitesse de Moteur**

**Étape 1 : Collecter des données d'entrée et de sortie pour un système de contrôle de vitesse de moteur avec deux moteurs.**

**Étape 2 : Utiliser un modèle ARMA pour capturer les valeurs passées de la sortie et l'effet des perturbations.**

```
% Données simulées pour les vitesses des moteurs
motor1_speed = 1500 + 50*randn(1500,1); % Vitesse avec du bruit
motor2_speed = 1450 + 70*randn(1500,1); % Vitesse avec du bruit
data = [motor1_speed motor2_speed];

nb = 2; % Ordre de la partie MA

% Modèle ARMA multivariable
model_ARMA = armax(data, [na nb], 'ls');

% Affichage des résultats
disp('Paramètres du Modèle ARMA Multivariable (Commande de Vitesse
de Moteur):');
disp(model_ARMA.A);
disp(model_ARMA.C);
```

**Analyse :** Le modèle ARMA multivariable capture les effets des perturbations aléatoires sur les vitesses des moteurs, ce qui est crucial pour maintenir un contrôle précis.

**Exercice 2.3 : Identification d'un Modèle ARMAX pour un Système de Contrôle de Processus Industriel**

**Étape 1 : Utiliser des données d'entrée et de sortie pour un système de contrôle de processus chimique avec deux variables de contrôle (température et pression).**

**Étape 2 : Appliquer un modèle ARMAX pour capturer l'effet des entrées sur les sorties.**

```
% Données simulées pour la température et la pression
temperature = 300 + 10*randn(1500,1); % Température avec du bruit
pressure = 5 + 0.5*randn(1500,1); % Pression avec du bruit
data = [temperature pressure];

u = sbpa; % Utiliser la SBPA comme entrée
data = [data u]; % Données d'entrée-sortie
nk = 1; % Délai entre entrée et sortie

% Modèle ARMAX multivariable
model_ARMAX = armax(data, [na nb nk]);

% Affichage des résultats
disp('Paramètres du Modèle ARMAX Multivariable (Contrôle de
Processus Industriel):');
disp(model_ARMAX.A);
disp(model_ARMAX.B);
disp(model_ARMAX.C);
```

**Analyse :** Le modèle ARMAX multivariable permet de modéliser l'influence des variables de contrôle (comme la température et la pression) sur les réponses du système, capturant les interactions complexes entre les variables.

**3. Réponse Temporelle et Approche Fréquentielle**

**Objectif**

Analyser la réponse temporelle et fréquentielle d'un système multivariable réel pour identifier ses dynamiques.

**Exercice 3.1 : Réponse Temporelle d'un Système Couplé (Système de Ventilation à Deux Zones)**

**Étape 1 : Définir un système de ventilation où l'air circule entre deux zones et simuler la réponse à une entrée échelon.**

```
A = [-0.3 0.2; 0.1 -0.4];
B = [1 0; 0 1];
C = [1 0; 0 1];
D = [0 0; 0 0];
sys = ss(A, B, C, D);

% Réponse à une entrée échelon
figure;
step(sys);
title('Réponse Temporelle du Système de Ventilation Couplé');
xlabel('Temps');
ylabel('Amplitude');
```

**Analyse :** La réponse temporelle montre comment les flux d'air dans les deux zones réagissent aux changements de contrôle, révélant les interactions entre les zones.

**Exercice 3.2 : Réponse Fréquentielle d'un Système de Contrôle de Position Multiaxial**

**Étape 1 : Définir un système de contrôle de position pour un robot avec deux axes.**

**Étape 2 : Analyser la réponse en fréquence pour comprendre la bande passante et les résonances.**

```
zeta = 0.4; % Facteur d'amortissement
omega_n = 3; % Fréquence naturelle
sys = tf([omega_n^2], [1 2*zeta*omega_n omega_n^2]); % Système du deuxième ordre

% Réponse en fréquence
figure;
bode(sys);
title('Réponse Fréquentielle du Système de Contrôle de Position Multiaxial');
```

**Analyse :** La réponse en fréquence montre comment le système réagit aux différentes fréquences d'entrée, ce qui est crucial pour éviter les résonances et garantir une performance stable.

**4. Méthode de Variable Instrumentale**

**Objectif**

Utiliser la méthode de variable instrumentale pour identifier un modèle lorsqu'il y a du bruit de mesure, en particulier pour des systèmes réels où le bruit est inévitable.

**Exercice 4.1 : Utilisation des Variables Instrumentales pour un Système de Réfrigération**

**Étape 1 : Générer des données pour un système de réfrigération avec des capteurs bruyants.**

**Étape 2 : Utiliser une variable instrumentale pour estimer un modèle précis.**

```
% Génération de données avec bruit
temperature = 2*u(:,1) + 0.5*randn(1500,1); % Température avec bruit
humidity = 3*u(:,2) + 0.7*randn(1500,1); % Humidité avec bruit
data = [temperature humidity];
z = iddata(data, u); % Créer un objet de données d'identification

% Identification avec variable instrumentale
model_VI = iv4(z, [na nb nk]);

% Affichage des résultats
disp('Paramètres du Modèle avec Variable Instrumentale (Réfrigération):');
disp(model_VI.A);
disp(model_VI.B);
```

**Analyse :** La méthode de variable instrumentale aide à réduire l'impact du bruit des capteurs, améliorant ainsi la précision des modèles pour les systèmes réels.

**5. Principe d'Ajustement du Modèle**

**Objectif**

Ajuster les paramètres d'un modèle pour minimiser un critère d'ajustement, en particulier pour des systèmes couplés et multivariables des systèmes réels.

**Exercice 5.1 : Ajustement d'un Modèle Linéaire pour un Système de Commande de Production Chimique**

**Étape 1 : Définir un modèle linéaire avec des paramètres initiaux pour un processus chimique contrôlé.**

**Étape 2 : Utiliser la méthode des moindres carrés pour ajuster le modèle.**

```
% Données simulées pour la production chimique
reactor1_temp = 350 + 5*randn(1500,1); % Température avec du bruit
reactor2_temp = 360 + 4*randn(1500,1); % Température avec du bruit
data = [reactor1_temp reactor2_temp];

% Estimation des paramètres avec moindres carrés
theta = inv(data'*data)*data'*y; % Paramètres estimés

% Affichage des résultats
disp('Paramètres Estimés pour le Système de Production Chimique:');
disp(theta);
```

**Analyse :** La méthode des moindres carrés est utilisée pour trouver les paramètres qui minimisent l'erreur entre les valeurs observées et prédites, garantissant un contrôle optimal du processus de production chimique.

**Conclusion**

Ces exercices fournissent une approche détaillée pour identifier et modéliser des systèmes couplés et multivariables réels. En utilisant des méthodes comme la génération de SBPA, l'identification des structures de modèles (AR, ARMA, ARMAX), l'analyse des réponses temporelles et fréquentielles, les méthodes de variable instrumentale, et l'ajustement des modèles, les ingénieurs peuvent développer des modèles précis pour contrôler et optimiser des systèmes complexes dans diverses applications industrielles et scientifiques.

# Projets 1  Identification des Systèmes Couplés et Multivariables pour des Systèmes Réels

## Introduction

L'identification des systèmes couplés et multivariables est essentielle pour modéliser des systèmes complexes dans des applications industrielles et scientifiques. Les systèmes couplés sont ceux où plusieurs variables d'entrée et de sortie interagissent. Ce document présente des exercices détaillés pour comprendre et appliquer les techniques d'identification des systèmes, y compris la génération de signaux d'excitation, le choix des structures de modèles appropriées, l'analyse des réponses temporelles et fréquentielles, l'utilisation de méthodes de variable instrumentale, et l'ajustement des modèles pour des systèmes réels.

## 1. Génération de Séquences Binaires Pseudo-Aléatoires (SBPA)

### Objectif

Utiliser des SBPA pour exciter des systèmes réels multivariables et collecter des données représentatives pour l'identification des modèles.

### Exercice 1.1 : Génération de SBPA pour un Système de Contrôle Température et Humidité

Équations associées :

- Une séquence binaire pseudo-aléatoire (SBPA) est utilisée pour exciter les variables d'entrée d'un système. Pour deux entrées ($u_1$ pour température et $u_2$ pour humidité), nous générons un signal SBPA :

$$u_1(t), u_2(t) = \text{SBPA}(t)$$

Étape 1 : Définir les paramètres de la SBPA

- Longueur de la séquence : $n = 1500$
- Amplitude : entre -1 et 1
- Nombre de canaux : 2 (un pour la température, un pour l'humidité)

Étape 2 : Générer la SBPA

```
n = 1500; % Nombre de points
num_channels = 2; % Nombre de canaux pour le système multivariable
sbpa = idinput([n, num_channels], 'prbs', [0 1], [-1 1]); % SBPA entre -1 et 1

% Affichage du signal SBPA pour les deux canaux
figure;
subplot(2,1,1);
plot(sbpa(:,1));
title('Signal SBPA - Canal Température');
xlabel('Temps');
ylabel('Amplitude');

subplot(2,1,2);
plot(sbpa(:,2));
title('Signal SBPA - Canal Humidité');
```

```
xlabel('Temps');
ylabel('Amplitude');
```

**Analyse :** Le signal SBPA est utilisé pour simuler des variations aléatoires dans le système de contrôle de la température et de l'humidité, fournissant une excitation suffisante pour capturer toutes les dynamiques du système.

**2. Choix de la Structure du Modèle (AR, ARMA, ARMAX)**

**Objectif**

Choisir et identifier des structures de modèles adaptées pour des systèmes réels couplés et multivariables.

Équations associées :

- Modèle AR pour deux sorties ($y_1$ et $y_2$) en fonction de leurs valeurs passées :

$$y_1(t) = a_{11}y_1(t-1) + a_{12}y_1(t-2) + \epsilon_1(t)$$

$$y_2(t) = a_{21}y_2(t-1) + a_{22}y_2(t-2) + \epsilon_2(t)$$

Étape 1 : Collecter des données de température pour deux zones d'un bâtiment.

- Les deux zones sont chauffées indépendamment, mais l'échange thermique entre elles influence les températures.

Étape 2 : Utiliser un modèle AR pour représenter les températures.

```
% Données simulées pour les températures de deux zones

zone1_temp = 20 + 2*randn(1500,1); % Température avec du bruit
zone2_temp = 22 + 1.5*randn(1500,1); % Température avec du bruit
data = [zone1_temp zone2_temp];

na = 2; % Ordre du modèle AR

% Modèle AR multivariable
model_AR = ar(data, na, 'ls'); % 'ls' pour méthode des moindres
carrés

% Affichage des résultats
disp('Paramètres du Modèle AR Multivariable (Chauffage à Deux
Zones):');
disp(model_AR.A);
```

**Analyse :** Le modèle AR multivariable prédit les températures des deux zones en fonction de leurs valeurs passées, capturant ainsi l'effet de l'échange thermique entre les zones.

**Exercice 2.2 : Identification d'un Modèle ARMA pour un Système de Commande de Vitesse de Moteur**

Équations associées :

- Modèle ARMA pour deux sorties ($y_1$ et $y_2$), capturant les valeurs passées et les chocs aléatoires ($e(t)$) :

$$y_1(t) = a_{11}y_1(t-1) + a_{12}y_1(t-2) + b_{11}e_1(t-1) + b_{12}e_1(t-2) + e_1(t)$$

$$y_2(t) = a_{21}y_2(t-1) + a_{22}y_2(t-2) + b_{21}e_2(t-1) + b_{22}e_2(t-2) + e_2(t)$$

Étape 1 : Collecter des données d'entrée et de sortie pour un système de contrôle de vitesse de moteur avec deux moteurs.

Étape 2 : Utiliser un modèle ARMA pour capturer les valeurs passées de la sortie et l'effet des perturbations.

```
% Données simulées pour les vitesses des moteurs
motor1_speed = 1500 + 50*randn(1500,1); % Vitesse avec du bruit
motor2_speed = 1450 + 70*randn(1500,1); % Vitesse avec du bruit
data = [motor1_speed motor2_speed];

nb = 2; % Ordre de la partie MA

% Modèle ARMA multivariable
model_ARMA = armax(data, [na nb], 'ls');

% Affichage des résultats
disp('Paramètres du Modèle ARMA Multivariable (Commande de Vitesse
de Moteur):');
disp(model_ARMA.A);
disp(model_ARMA.C);
```

*Analyse : Le modèle ARMA multivariable capture les effets des perturbations aléatoires sur les vitesses des moteurs, ce qui est crucial pour maintenir un contrôle précis.*

**Exercice 2.3 : Identification d'un Modèle ARMAX pour un Système de Contrôle de Processus Industriel**

Équations associées :

- Modèle ARMAX pour deux sorties ($y_1$ et $y_2$) influencées par des entrées ($u_1$ et $u_2$) :

$$y_1(t) = a_{11}y_1(t-1) + b_{11}u_1(t-1) + e_1(t)$$

$$y_2(t) = a_{21}y_2(t-1) + b_{21}u_2(t-1) + e_2(t)$$

Étape 1 : Utiliser des données d'entrée et de sortie pour un système de contrôle de processus chimique avec deux variables de contrôle (température et pression).

Étape 2 : Appliquer un modèle ARMAX pour capturer l'effet des entrées sur les sorties.

```
% Données simulées pour la température et la pression
temperature = 300 + 10*randn(1500,1); % Température avec du bruit
pressure = 5 + 0.5*randn(1500,1); % Pression avec du bruit
data = [temperature pressure];

u = sbpa; % Utiliser la SBPA comme entrée
data = [data u]; % Données d'entrée-sortie
nk = 1; % Délai entre entrée et sortie

% Modèle ARMAX multivariable
model_ARMAX = armax(data, [na nb nk]);

% Affichage des résultats
disp('Paramètres du Modèle ARMAX Multivariable (Contrôle de
Processus Industriel):');
disp(model_ARMAX.A);
disp(model_ARMAX.B);
disp(model_ARMAX.C);
```

**Analyse :** Le modèle ARMAX multivariable permet de modéliser l'influence des variables de contrôle (comme la température et la pression) sur les réponses du système, capturant les interactions complexes entre les variables.

**3. Réponse Temporelle et Approche Fréquentielle**

**Objectif**

Analyser la réponse temporelle et fréquentielle d'un système multivariable réel pour identifier ses dynamiques.

**Exercice 3.1 : Réponse Temporelle d'un Système Couplé (Système de Ventilation à Deux Zones)**

**Équations associées :**

- Réponse temporelle d'un système couplé en fonction des entrées $(u_1, u_2)$ et des états $(x_1, x_2)$:

$$\dot{x_1} = a_{11}x_1 + a_{12}x_2 + b_{11}u_1$$

$$\dot{x_2} = a_{21}x_1 + a_{22}x_2 + b_{21}u_2$$

**Étape 1 : Définir un système de ventilation où l'air circule entre deux zones et simuler la réponse à une entrée échelon.**

```
A = [-0.3 0.2; 0.1 -0.4];
B = [1 0; 0 1];
C = [1 0; 0 1];
D = [0 0; 0 0];
sys = ss(A, B, C, D);

% Réponse à une entrée échelon
figure;
step(sys);
title('Réponse Temporelle du Système de Ventilation Couplé');
xlabel('Temps');
ylabel('Amplitude');
```

**Analyse :** La réponse temporelle montre comment les flux d'air dans les deux zones réagissent aux changements de contrôle, révélant les interactions entre les zones.

**Exercice 3.2 : Réponse Fréquentielle d'un Système de Contrôle de Position Multiaxial**

**Équations associées :**

- Réponse fréquentielle en fonction de la fonction de transfert H(s) du système multiaxial

$$H(s) = \frac{K\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}$$

**Étape 1 : Définir un système de contrôle de position pour un robot avec deux axes.**

**Étape 2 : Analyser la réponse en fréquence pour comprendre la bande passante et les résonances.**

```
matlab
zeta = 0.4; % Facteur d'amortissement
omega_n = 3; % Fréquence naturelle
sys = tf([omega_n^2], [1 2*zeta*omega_n omega_n^2]); % Système du deuxième ordre

% Réponse en fréquence
figure;
bode(sys);
title('Réponse Fréquentielle du Système de Contrôle de Position Multiaxial');
```

**Analyse :** La réponse en fréquence montre comment le système réagit aux différentes fréquences d'entrée, ce qui est crucial pour éviter les résonances et garantir une performance stable.

**4. Méthode de Variable Instrumentale**

**Objectif**

Utiliser la méthode de variable instrumentale pour identifier un modèle lorsqu'il y a du bruit de mesure, en particulier pour des systèmes réels où le bruit est inévitable.

**Exercice 4.1 : Utilisation des Variables Instrumentales pour un Système de Réfrigération**

**Équations associées :**

Équations associées :

- Modèle avec bruit ($v(t)$) et utilisation de variables instrumentales ($z(t)$) :

$$y(t) = \theta^T \phi(t) + v(t)$$

$z(t)$ est corrélé avec $\phi(t)$, mais non corrélé avec $v(t)$

Étape 1 : Générer des données pour un système de réfrigération avec des capteurs bruyants.

Étape 2 : Utiliser une variable instrumentale pour estimer un modèle précis.

.

```
% Génération de données avec bruit
temperature = 2*u(:,1) + 0.5*randn(1500,1); % Température avec bruit
humidity = 3*u(:,2) + 0.7*randn(1500,1); % Humidité avec bruit
data = [temperature humidity];
z = iddata(data, u); % Créer un objet de données d'identification

% Identification avec variable instrumentale
model_VI = iv4(z, [na nb nk]);

% Affichage des résultats
disp('Paramètres du Modèle avec Variable Instrumentale (Réfrigération):');
disp(model_VI.A);
disp(model_VI.B);
```

**Analyse :** La méthode de variable instrumentale aide à réduire l'impact du bruit des capteurs, améliorant ainsi la précision des modèles pour les systèmes réels.

**5. Principe d'Ajustement du Modèle**

**Objectif**

Ajuster les paramètres d'un modèle pour minimiser un critère d'ajustement, en particulier pour des systèmes couplés et multivariables des systèmes réels.

**Exercice 5.1 : Ajustement d'un Modèle Linéaire pour un Système de Commande de Production**

Équations associées :

- Fonction coût à minimiser pour ajuster le modèle :

$$J(\theta) = \sum_{t=1}^{N} (y(t) - \hat{y}(t|\theta))^2$$

Étape 1 : Définir un modèle linéaire avec des paramètres initiaux pour un processus chimique contrôlé.

Étape 2 : Utiliser la méthode des moindres carrés pour ajuster le modèle.

```
% Données simulées pour la production chimique
reactor1_temp = 350 + 5*randn(1500,1); % Température avec du bruit
reactor2_temp = 360 + 4*randn(1500,1); % Température avec du bruit
data = [reactor1_temp reactor2_temp];

% Estimation des paramètres avec moindres carrés
theta = inv(data'*data)*data'*y; % Paramètres estimés

% Affichage des résultats
disp('Paramètres Estimés pour le Système de Production Chimique:');
disp(theta);
```

**Analyse :** La méthode des moindres carrés est utilisée pour trouver les paramètres qui minimisent l'erreur entre les valeurs observées et prédites, garantissant un contrôle optimal du processus de production chimique.

**Conclusion**

Ces projets fournissent une approche détaillée pour identifier et modéliser des systèmes couplés et multivariables réels. En utilisant des méthodes comme la génération de SBPA, l'identification des structures de modèles (AR, ARMA, ARMAX), l'analyse des réponses temporelles et fréquentielles, les méthodes de variable instrumentale, et l'ajustement des modèles, les ingénieurs peuvent développer des modèles précis pour contrôler et optimiser des systèmes complexes dans diverses applications industrielles et scientifiques

# Projet 2 : Identification et Modélisation des Systèmes Couplés et Multi variables Réels

**Introduction**

Les systèmes couplés et multi variables réels, tels que les moteurs DC, les avions, les missiles, et les réacteurs chimiques, présentent des défis uniques en termes d'identification et de modélisation. Ce projet se concentre sur la modélisation de ces systèmes complexes en utilisant des techniques avancées, y compris la génération de signaux d'excitation (SBPA), le choix de la structure du modèle (AR, ARMA, ARMAX), l'analyse des réponses temporelles et fréquentielles, et l'ajustement des modèles. Nous allons détailler les équations différentielles et autres équations pertinentes pour chaque type de système, en fournissant des exemples pratiques et détaillés pour chaque méthode.

**1. Génération de Séquences Binaires Pseudo-Aléatoires (SBPA)**

**Objectif**

Utiliser des SBPA pour exciter des systèmes multivariables et collecter des données représentatives pour l'identification des modèles.

**Exercice 1.1 : Génération de SBPA pour un Moteur DC Couplé**

**Équations associées :**

Pour un système de moteur DC avec deux entrées (tensions $V_1$ et $V_2$) et deux sorties (vitesses angulaires $\omega_1$ et $\omega_2$), les équations différentielles du système sont les suivantes :

$$L\frac{di_1}{dt} + Ri_1 + K_e\omega_1 = V_1(t)$$

$$J\frac{d\omega_1}{dt} + B\omega_1 = K_t i_1$$

$$L\frac{di_2}{dt} + Ri_2 + K_e\omega_2 = V_2(t)$$

$$J\frac{d\omega_2}{dt} + B\omega_2 = K_t i_2$$

**Étape 1 : Définir les paramètres de la SBPA**

- Longueur de la séquence : n=1500
- Amplitude : entre -1 et 1
- Nombre de canaux : 2 (un pour chaque entrée du moteur)

**Étape 2 : Générer la SBPA**

```
n = 1500; % Nombre de points
num_channels = 2; % Nombre de canaux pour le système multivariable
sbpa = idinput([n, num_channels], 'prbs', [0 1], [-1 1]); % SBPA
entre -1 et 1

% Affichage du signal SBPA pour les deux canaux
figure;
subplot(2,1,1);
```

```
plot(sbpa(:,1));
title('Signal SBPA - Canal 1 (Tension V1)');
xlabel('Temps');
ylabel('Amplitude');

subplot(2,1,2);
plot(sbpa(:,2));
title('Signal SBPA - Canal 2 (Tension V2)');
xlabel('Temps');
ylabel('Amplitude');
```

**Analyse :** Le signal SBPA est utilisé pour fournir une excitation indépendante et suffisante aux deux entrées du moteur DC, permettant de capturer toutes les dynamiques couplées.

**2. Choix de la Structure du Modèle (AR, ARMA, ARMAX)**

**Objectif**

Choisir et identifier des structures de modèles adaptées pour des systèmes réels couplés et multivariables tels que les avions et les missiles.

**Exercice 2.1 : Modélisation AR d'un Système de Commande de Vol d'Avion**

**Équations associées :**

Pour modéliser les angles d'inclinaison ($\phi$), de tangage ($\theta$), et de lacet ($\psi$) d'un avion, les équations en fonction des commandes de gouvernes et des états passés sont :

$$\phi(t) = a_{11}\phi(t-1) + a_{12}\theta(t-1) + \epsilon_1(t)$$
$$\theta(t) = a_{21}\theta(t-1) + a_{22}\psi(t-1) + \epsilon_2(t)$$

**Étape 1 : Collecter des données d'angle pour les différentes commandes de vol.**

**Étape 2 : Utiliser un modèle AR pour représenter ces dynamiques.**

```
% Données simulées pour les angles de vol
roll_angle = 5 + 0.5*randn(1500,1); % Angle de roulis avec du bruit
pitch_angle = 2 + 0.3*randn(1500,1); % Angle de tangage avec du
bruit
data = [roll_angle pitch_angle];

na = 2; % Ordre du modèle AR

% Modèle AR multivariable
model_AR = ar(data, na, 'ls'); % 'ls' pour méthode des moindres
carrés

% Affichage des résultats
disp('Paramètres du Modèle AR Multivariable (Commande de Vol
d\'Avion):');
disp(model_AR.A);
```

**Analyse :** Le modèle AR multivariable permet de prédire les angles de roulis et de tangage en fonction de leurs valeurs passées, capturant les effets des interactions entre les axes de l'avion.

**Exercice 2.2 : Identification ARMA d'un Système de Guidage de Missile**

**Équations associées :**

Pour un missile, les équations incluent des termes AR et MA pour capturer les réponses aux perturbations atmosphériques :

Les équations ARMA prennent en compte les effets autoregressifs et de moyenne mobile :

$$y_1(t) = a_{11}y_1(t-1) + b_{11}e_1(t-1) + e_1(t)$$
$$y_2(t) = a_{21}y_2(t-1) + b_{21}e_2(t-1) + e_2(t)$$

**Étape 1 : Collecter des données de vol pour les axes longitudinaux et latéraux.**

**Étape 2 : Utiliser un modèle ARMA pour capturer les dynamiques.**

```
% Données simulées pour les axes du missile
long_axis = 300 + 15*randn(1500,1); % Axe longitudinal avec du bruit
lat_axis = 250 + 20*randn(1500,1); % Axe latéral avec du bruit
data = [long_axis lat_axis];

nb = 2; % Ordre de la partie MA

% Modèle ARMA multivariable
model_ARMA = armax(data, [na nb], 'ls');

% Affichage des résultats
disp('Paramètres du Modèle ARMA Multivariable (Guidage de
Missile):');
disp(model_ARMA.A);
disp(model_ARMA.C);
```

**Analyse :** Le modèle ARMA multivariable capture les réponses des axes longitudinal et latéral aux perturbations, important pour ajuster les trajectoires en temps réel.

**Exercice 2.3 : Identification ARMAX d'un Réacteur Chimique**

**Équations associées :**

Les équations ARMAX modélisent les concentrations des espèces ($C_A$, $C_B$) et les entrées ($u_1$, $u_2$) :

$$C_A(t) = a_{11}C_A(t-1) + b_{11}u_1(t-1) + e_1(t)$$
$$C_B(t) = a_{21}C_B(t-1) + b_{21}u_2(t-1) + e_2(t)$$

**Étape 1 : Collecter des données de concentration et de température dans un réacteur.**

**Étape 2 : Appliquer un modèle ARMAX pour capturer les dynamiques.**

```
% Données simulées pour les concentrations des réactifs
conc_A = 1.5 + 0.1*randn(1500,1); % Concentration de A avec du bruit
```

```
conc_B = 2.5 + 0.2*randn(1500,1); % Concentration de B avec du bruit
data = [conc_A conc_B];

u = sbpa; % Utiliser la SBPA comme entrée
data = [data u]; % Données d'entrée-sortie
nk = 1; % Délai entre entrée et sortie

% Modèle ARMAX multivariable
model_ARMAX = armax(data, [na nb nk]);

% Affichage des résultats
disp('Paramètres du Modèle ARMAX Multivariable (Réacteur
Chimique):');
disp(model_ARMAX.A);
disp(model_ARMAX.B);
disp(model_ARMAX.C);
```

**Analyse :** Le modèle ARMAX multivariable permet de modéliser l'influence des conditions d'entrée (température, débit) sur les réactions chimiques, crucial pour l'optimisation des processus.

**3. Réponse Temporelle et Approche Fréquentielle**

**Objectif**

Analyser la réponse temporelle et fréquentielle d'un système multivariable réel pour identifier ses dynamiques.

**Exercice 3.1 : Réponse Temporelle d'un Moteur DC Couplé**

**Équations associées :**

Les équations d'état pour le courant et la vitesse sont :

$$\dot{i}_1 = -\frac{R}{L} i_1 - \frac{K_e}{L} \omega_1 + \frac{1}{L} V_1$$
$$\dot{\omega}_1 = -\frac{B}{J} \omega_1 + \frac{K_t}{J} i_1$$

**Étape 1 : Simuler la réponse à une entrée échelon pour les deux entrées.**

```
matlab
Copier le code
A = [-R/L -Ke/L; Kt/J -B/J];
B = [1/L; 0];
C = [0 1]; % Observer la vitesse
D = [0];
sys = ss(A, B, C, D);
% Réponse à une entrée échelon
figure;
step(sys);
title('Réponse Temporelle du Moteur DC Couplé');
xlabel('Temps');
ylabel('Vitesse');
```

**Analyse :** La réponse temporelle montre comment la vitesse du moteur réagit aux changements de tension d'entrée, révélant les interactions couplées des deux moteurs.

**4. Méthode de Variable Instrumentale**

**Objectif**

Utiliser la méthode de variable instrumentale pour identifier un modèle lorsqu'il y a du bruit de mesure, en particulier pour des systèmes réels où le bruit est inévitable.

**Exercice 4.1 : Utilisation des Variables Instrumentales pour un Réacteur Chimique**

**Équations associées :**

Modèle avec bruit et utilisation de variables instrumentales :

$$y(t) = \theta^T \phi(t) + v(t)$$

**Étape 1 : Générer des données pour un réacteur avec bruit de capteur.**

**Étape 2 : Utiliser une variable instrumentale pour estimer un modèle précis.**

```
% Génération de données avec bruit
temp = 150 + 5*randn(1500,1); % Température avec bruit
pressure = 3 + 0.1*randn(1500,1); % Pression avec bruit
data = [temp pressure];
z = iddata(data, u); % Créer un objet de données d'identification

% Identification avec variable instrumentale
model_VI = iv4(z, [na nb nk]);

% Affichage des résultats
disp('Paramètres du Modèle avec Variable Instrumentale (Réacteur
Chimique):');
disp(model_VI.A);
disp(model_VI.B);
```

**Analyse :** La méthode de variable instrumentale aide à réduire l'impact du bruit des capteurs, améliorant ainsi la précision des modèles pour les systèmes réels.

**Conclusion**

Ces exercices fournissent une approche détaillée pour identifier et modéliser des systèmes couplés et multivariables réels tels que les moteurs DC, les avions, les missiles, et les réacteurs chimiques. En utilisant des techniques telles que la génération de SBPA, l'identification des structures de modèles (AR, ARMA, ARMAX), l'analyse des réponses temporelles et fréquentielles, les méthodes de variable instrumentale, et l'ajustement des modèles, les ingénieurs peuvent développer des modèles précis pour contrôler et optimiser des systèmes complexes dans diverses applications industrielles et scientifiques.

**5. Identification ARMA d'un Système de Guidage de Missile**

Les systèmes de guidage de missiles sont des exemples typiques de systèmes multivariables et couplés, où les réponses aux perturbations et les interactions entre différentes dynamiques sont essentielles.

### 1.1 Équations Différentielles de Base pour un Missile

Les équations différentielles qui décrivent les dynamiques de vol d'un missile peuvent être modélisées comme suit :

1. **Équation de Mouvement en Longitudinal (Axe x)** :

$$\dot{x} = V\cos(\theta)\cos(\psi)$$

Où $x$ est la position longitudinale, $V$ est la vitesse, $\theta$ est l'angle de tangage, et $\psi$ est l'angle de lacet.

2. **Équation de Mouvement en Latéral (Axe y)** :

$$\dot{y} = V\cos(\theta)\sin(\psi)$$

Où $y$ est la position latérale.

3. **Équation de Mouvement en Vertical (Axe z)** :

$$\dot{z} = V\sin(\theta)$$

Où $z$ est l'altitude.

4. **Équation de Tangage** :

$$\dot{\theta} = \frac{M}{I_y}$$

Où $M$ est le moment de tangage et $I_y$ est le moment d'inertie autour de l'axe y.

5. **Équation de Lacet** :

$$\dot{\psi} = \frac{N}{I_z}$$

Où $N$ est le moment de lacet et $I_z$ est le moment d'inertie autour de l'axe z.

.

Ces équations décrivent les mouvements de base d'un missile en vol. Les forces et moments (comme $M$ et $N$) sont généralement des fonctions de l'état et des commandes de gouvernes.

### 1.2 Identification ARMA pour le Guidage

Pour capturer les réponses aux perturbations atmosphériques et les interactions entre les dynamiques du missile, nous utiliserons un modèle ARMA. Les équations ARMA pour les dynamiques longitudinales et latérales peuvent être formulées comme suit :

1. **Modèle ARMA pour la Position Longitudinale ($y_1(t)$)** :

$$y_1(t) = a_{11}y_1(t-1) + b_{11}e_1(t-1) + e_1(t)$$

Où $y_1(t)$ est la position longitudinale à l'instant $t$, $a_{11}$ et $b_{11}$ sont les coefficients du modèle, et $e_1(t)$ est le terme de perturbation.

2. **Modèle ARMA pour la Position Latérale ($y_2(t)$)** :

$$y_2(t) = a_{21}y_2(t-1) + b_{21}e_2(t-1) + e_2(t)$$

Où $y_2(t)$ est la position latérale à l'instant $t$, $a_{21}$ et $b_{21}$ sont les coefficients du modèle, et $e_2(t)$ est le terme de perturbation.

Ces équations ARMA capturent les relations temporelles et les réponses aux perturbations qui affectent la trajectoire du missile.

### 1.3 Étapes pour l'Identification ARMA

1. **Collecte de Données** :
   - Mesurer les positions longitudinale y1et latérale y2 à des intervalles de temps réguliers. Les données doivent inclure des réponses à des perturbations (e.g., changements de vent).
2. **Estimation des Paramètres ARMA** :
   - Utiliser des techniques d'estimation paramétrique pour déterminer les coefficients aija_{ij}aij et bijb_{ij}bij. On peut utiliser la méthode des moindres carrés pour l'estimation initiale.
3. **Validation du Modèle** :
   - Comparer les prédictions du modèle ARMA avec des données de vol réelles pour valider la précision du modèle. Ajuster les paramètres si nécessaire.

**Code MATLAB pour l'Identification ARMA**:

```
% Données simulées pour les positions longitudinales et latérales
long_axis = 300 + 15*randn(1500,1); % Position longitudinale avec du
bruit
lat_axis = 250 + 20*randn(1500,1); % Position latérale avec du bruit
data = [long_axis lat_axis];

na = 1; % Ordre de la partie AR
nb = 1; % Ordre de la partie MA

% Modèle ARMA multivariable
model_ARMA = armax(data, [na nb]);

% Affichage des résultats
disp('Paramètres du Modèle ARMA Multivariable (Guidage de
Missile):');
disp('A:');
disp(model_ARMA.A);
disp('B:');
disp(model_ARMA.B);
disp('C:');
disp(model_ARMA.C);
```

### 6. Modélisation Multivariables d'un Réacteur Chimique

Les réacteurs chimiques multivariables présentent des interactions complexes entre les espèces chimiques, la température et le débit. Ces interactions peuvent être modélisées par des équations différentielles décrivant les bilans de masse et d'énergie.

### 2.1 Équations Différentielles pour un Réacteur Chimique

1. **Équation de Bilan de Masse pour le Réactif $A$ :**

$$\frac{dC_A}{dt} = \frac{F_{in}}{V}(C_{A,in} - C_A) - k_1 C_A - k_2 C_A^2$$

Où $C_A$ est la concentration du réactif A, $F_{in}$ est le débit d'entrée, $V$ est le volume du réacteur, $k_1$ et $k_2$ sont les constantes de réaction.

2. **Équation de Bilan de Masse pour le Réactif $B$ :**

$$\frac{dC_B}{dt} = \frac{F_{in}}{V}(C_{B,in} - C_B) - k_3 C_B$$

Où $C_B$ est la concentration du réactif B.

3. **Équation de Bilan d'Énergie :**

$$\frac{dT}{dt} = \frac{Q}{\rho C_p V} - \frac{F_{in}}{V}(T - T_{in}) + \frac{-\Delta H_r}{\rho C_p} k_1 C_A$$

Où $T$ est la température du réacteur, $Q$ est le flux de chaleur, $\rho$ est la densité du mélange, $C_p$ est la capacité thermique, et $\Delta H_r$ est la chaleur de réaction.

### 2.2 Identification ARMAX pour le Réacteur

Pour modéliser l'influence des variables d'entrée (débit, température d'entrée) sur les concentrations et la température du réacteur, nous utilisons un modèle ARMAX.

1. **Modèle ARMAX pour la Concentration de $A$ ($C_A(t)$) :**

$$C_A(t) = a_{11} C_A(t-1) + b_{11} u_1(t-1) + e_1(t)$$

Où $u_1(t)$ est l'entrée (e.g., débit).

2. **Modèle ARMAX pour la Température ($T(t)$) :**

$$T(t) = a_{21} T(t-1) + b_{21} u_2(t-1) + e_2(t)$$

Où $u_2(t)$ est l'entrée de température.

**Code MATLAB pour l'Identification ARMAX**:

```
% Données simulées pour les concentrations et la température
conc_A = 1.5 + 0.1*randn(1500,1); % Concentration de A avec du bruit
temp = 350 + 5*randn(1500,1); % Température avec du bruit
data = [conc_A temp];

u = sbpa; % Utiliser la SBPA comme entrée (débit et température)
data = [data u]; % Données d'entrée-sortie
nk = 1; % Délai entre entrée et sortie

% Modèle ARMAX multivariable
model_ARMAX = armax(data, [na nb nk]);

% Affichage des résultats
```

```
disp('Paramètres du Modèle ARMAX Multivariable (Réacteur
Chimique):');
disp('A:');
disp(model_ARMAX.A);
disp('B:');
disp(model_ARMAX.B);
disp('C:');
disp(model_ARMAX.C);
```

**Conclusion**

Ces exercices et équations montrent comment identifier des modèles pour des systèmes multivariables et couplés réels, tels que les missiles et les réacteurs chimiques, en utilisant des techniques ARMA et ARMAX. Ces méthodes permettent de capturer la complexité et les interactions des systèmes réels, fournissant ainsi des outils puissants pour la modélisation, la simulation et le contrôle des systèmes complexes.

# Chapitre 6

# Méthodes d'Identification Graphiques des Systèmes Réels

**Introduction**

Les méthodes d'identification graphiques sont des techniques pratiques utilisées pour déterminer les paramètres de modèles de systèmes dynamiques basées sur leur réponse à certaines entrées spécifiques, généralement des signaux d'échelon ou de rampe. Ces méthodes sont très populaires en raison de leur simplicité et de leur intuitivité, et elles sont souvent appliquées dans des environnements industriels pour le réglage rapide des systèmes de contrôle. Ce chapitre se concentre sur des méthodes couramment utilisées comme la méthode de Strejc et la méthode de Broïda, ainsi que sur d'autres méthodes graphiques importantes.

## 1. Méthode de Strejc

La méthode de Strejc est une méthode d'identification graphique qui permet d'identifier les paramètres d'un système de contrôle à partir de sa réponse à un échelon. Cette méthode est particulièrement efficace pour les systèmes avec des délais de réponse ou des systèmes de type premier ou second ordre.

### 1.1 Principe de la Méthode de Strejc

La méthode de Strejc identifie un système de la forme suivante :

La méthode de Strejc identifie un système de la forme suivante :

$$G(s) = \frac{K}{(1+T_1 s)(1+T_2 s)\cdots(1+T_n s)} e^{-\tau s}$$

- $K$ : Gain statique du système.
- $T_i$ : Constantes de temps associées aux pôles du système.
- $\tau$ : Temps de retard ou délai.

**Étapes de la méthode :**

1. **Observer la réponse à un échelon** : Appliquer un échelon unitaire et observer la réponse de sortie du système.
2. **Déterminer les points caractéristiques** : Identifier les points caractéristiques sur la courbe de réponse, tels que le temps de montée (Tr), le temps de réponse (Ts et le temps de retard (τ\tauτ).
3. **Calculer les paramètres du modèle** : Utiliser les points caractéristiques pour estimer les paramètres K, Ti, et τ.

### 1.2 Exercice 1 : Identification d'un Système de Premier Ordre avec Délai

**Problème :** Identifier les paramètres d'un système de premier ordre avec un délai à partir de sa réponse à un échelon.

**Système à identifier :**

$$G(s) = \frac{K}{1 + Ts} e^{-\tau s}$$

**Données :**

- Gain statique ($K$) : Inconnu
- Constante de temps ($T$) : Inconnue
- Temps de retard ($\tau$) : 2 secondes

**Étapes :**

1. **Application d'un échelon unitaire :** On applique un signal d'échelon unitaire au système et on mesure la réponse.
2. **Mesure de la réponse :** Supposons que la sortie atteigne 63% de la valeur finale après 4 secondes.
3. **Calcul des paramètres** :
    - Temps de retard $\tau = 2$ s
    - Temps pour atteindre 63% de la réponse finale : $T + \tau = 4$ s
    - Donc, $T = 2$ s

    **Solution :**

    Le modèle estimé est :

$$G(s) = \frac{K}{1 + 2s} e^{-2s}$$

    Pour déterminer $K$, on utilise la valeur finale de la réponse à l'échelon.

***Code MATLAB :***

```
% Paramètres
T = 2; % Constante de temps
tau = 2; % Temps de retard
K = 1; % Gain (à ajuster selon les mesures réelles)

% Modèle de transfert
sys = tf(K, [T 1], 'InputDelay', tau);

% Réponse à un échelon
figure;
step(sys);
title('Réponse à un échelon du système de premier ordre avec délai');
xlabel('Temps (s)');
ylabel('Amplitude');
```

**2. Méthode de Broïda**

La méthode de Broïda est utilisée pour estimer les paramètres des systèmes de type PID à partir de leur réponse à un échelon. Elle est adaptée pour les systèmes modélisables par des fonctions de transfert de premier ou second ordre avec un délai.

### 2.1 Principe de la Méthode de Broïda

La méthode de Broïda s'applique aux systèmes décrits par :

$$G(s) = \frac{K}{(1+Ts)^n} e^{-\tau s}$$

Les systèmes peuvent être de premier ou de second ordre, avec ou sans délai.

**Étapes de la méthode :**

1. **Appliquer un échelon unitaire :** Observer la réponse du système à un échelon unitaire.
2. **Tracer une tangente à la courbe au point d'inflexion :** Utiliser cette tangente pour déterminer les paramètres clés.
3. **Estimer les paramètres :** Déduire les paramètres **K, T et τ** à partir de la courbe et de la tangente tracée.

### 2.2 Exercice 2 : Identification d'un Système avec la Méthode de Broïda

**Problème :** Estimer les paramètres d'un système de second ordre avec délai à partir de sa réponse à un échelon.

**Données :**

- Gain final : 5
- Temps de montée ($T_r$) : 3 secondes
- Temps de retard ($\tau$) : 1 seconde

**Étapes :**

1. **Application d'un échelon unitaire :** On applique un échelon unitaire et on observe la réponse.
2. **Mesure de la réponse :** Identifier
3. la tangente au point d'inflexion et les points caractéristiques.
4. **Calcul des paramètres :**

- Gain $K = 5$
- Temps de retard $\tau = 1$ s
- Constante de temps estimée à partir de la tangente et du temps de montée.

**Solution :**

Le modèle estimé est :

$$G(s) = \frac{5}{(1+3s)^2} e^{-s}$$

***Code MATLAB :***

```
% Paramètres
K = 5; % Gain
T = 3; % Constante de temps
tau = 1; % Temps de retard

% Modèle de transfert
sys = tf(K, [T^2 2*T 1], 'InputDelay', tau);

% Réponse à un échelon
figure;
step(sys);
title('Réponse à un échelon du système avec la méthode de Broïda');
xlabel('Temps (s)');
ylabel('Amplitude');
```

### 3. Autres Méthodes Graphiques

#### 3.1 Méthode de la Zone de Réponse (ZR)

La méthode de la Zone de Réponse est utilisée pour ajuster les paramètres d'un système en se basant sur des zones spécifiques de la réponse transitoire. Cette méthode est souvent appliquée pour les systèmes de régulation industrielle.

**Étapes :**

1. Appliquer un échelon unitaire.
2. Identifier les zones de réponse telles que le temps de montée, le dépassement et le temps de stabilisation.
3. Ajuster les paramètres pour minimiser les erreurs dans ces zones.

**Exercice 3 : Ajustement PID par Méthode de la Zone de Réponse**

**Problème :** Ajuster les paramètres PID pour minimiser le temps de stabilisation tout en limitant le dépassement.

**Solution :** Utiliser les courbes de réponse pour ajuster les paramètres proportionnels (Kp), intégral (Ki et dérivatif (Kd)

**Code MATLAB :**

```
% Système de contrôle PID
Kp = 2; % Gain proportionnel
Ki = 1; % Gain intégral
Kd = 0.5; % Gain dérivatif

sys = tf([Kd Kp Ki], [1 0]);

% Réponse à un échelon
figure;
step(sys);
title('Réponse à un échelon du système PID ajusté');
xlabel('Temps (s)');
ylabel('Amplitude');
```

**Conclusion**

Les méthodes d'identification graphiques offrent une approche intuitive pour estimer les paramètres des systèmes réels en utilisant des réponses à des signaux d'entrée simples comme des échelons ou des rampes. Les méthodes de Strejc et de Broïda permettent d'obtenir des modèles précis pour des systèmes de premier et de second ordre, même avec des délais. En combinant ces méthodes avec des ajustements basés sur la réponse transitoire, on peut optimiser les performances des systèmes de contrôle de manière efficace et pratique. Les exercices et les exemples MATLAB fournis démontrent comment appliquer ces techniques dans des scénarios réels, offrant ainsi une base solide pour les applications industrielles et académiques.

# Applications :

## Méthodes d'Identification Graphiques des Systèmes Réels

Les méthodes d'identification graphiques sont des techniques pratiques et intuitives permettant de déterminer les paramètres des modèles de systèmes dynamiques à partir de leurs réponses à des entrées spécifiques, comme des signaux d'échelon. Ces méthodes sont particulièrement utiles pour les systèmes de premier et de second ordre, ainsi que pour les systèmes avec des délais. Les exercices suivants vous guideront à travers l'utilisation de la méthode de Strejc et la méthode de Broïda, ainsi que d'autres méthodes graphiques pour identifier les paramètres des systèmes réels.

**Exercice 1 : Identification d'un Système de Premier Ordre avec Délai - Méthode de Strejc**

**Problème** : Un système dynamique réagit à une entrée échelon unitaire par une réponse qui atteint 63% de sa valeur finale en 4 secondes. Il y a un délai initial de 2 secondes avant que la sortie ne commence à réagir. Déterminer les paramètres du modèle de ce système en utilisant la méthode de Strejc.

**Données :**

- Gain statique $K$ : Inconnu
- Constante de temps $T$ : Inconnue
- Temps de retard $\tau$ : 2 secondes

**Solution :**

1. **Identification du temps de retard ($\tau$) :** D'après l'énoncé, le système présente un délai de 2 secondes avant de réagir, donc $\tau = 2$ secondes.
2. **Identification de la constante de temps ($T$) :** La réponse atteint 63% de la valeur finale en 4 secondes. Dans un système de premier ordre sans retard, 63% correspond à la constante de temps. Étant donné que le délai est de 2 secondes, le temps pour atteindre 63% est la somme de $\tau$ et $T$. Donc, $T + \tau = 4$. Ainsi, $T = 4 - 2 = 2$ secondes.
3. **Estimation du gain ($K$) :** Supposons que la réponse finale est mesurée et qu'elle soit de 5 unités. Le gain statique $K$ est alors $K = 5$.

**Code MATLAB pour vérifier la réponse :**

```
% Paramètres
K = 5; % Gain statique
T = 2; % Constante de temps
tau = 2; % Temps de retard

% Modèle de transfert
sys = tf(K, [T 1], 'InputDelay', tau);

% Réponse à un échelon
figure;
step(sys);
title('Réponse à un échelon - Méthode de Strejc');
xlabel('Temps (s)');
ylabel('Amplitude');
```

**Exercice 2 : Identification d'un Système de Second Ordre sans Délai - Méthode de Broïda**

**Problème :** Un système dynamique a une réponse à un échelon avec un dépassement de 10% et un temps de montée de 5 secondes. Utilisez la méthode de Broïda pour identifier les paramètres du modèle.

**Données :**

- Dépassement ($M_p$) : 10%
- Temps de montée ($T_r$) : 5 secondes

**Solution :**

1. **Dépassement et Facteur d'Amortissement ($\zeta$) :** Le dépassement ($M_p$) est lié au facteur d'amortissement $\zeta$ par la relation :

$$M_p = e^{\left(\frac{-\pi\zeta}{\sqrt{1-\zeta^2}}\right)}$$

Avec $M_p = 0.1$ (10%), nous résolvons l'équation pour $\zeta$. En utilisant une calculatrice ou un solveur numérique, nous trouvons $\zeta \approx 0.591$.

2. **Temps de montée et Fréquence Naturelle ($\omega_n$) :** Le temps de montée $T_r$ est approximé par :

$$T_r \approx \frac{1.8}{\omega_n}$$

En utilisant $T_r = 5$ secondes, on obtient :

$$\omega_n \approx \frac{1.8}{5} = 0.36 \text{ rad/s}$$

**Modèle estimé :**

$$G(s) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}$$

Substituer les valeurs de $\zeta$ et $\omega_n$ :

$$G(s) = \frac{0.36^2}{s^2 + 2 \times 0.591 \times 0.36s + 0.36^2}$$

**Code MATLAB pour vérifier la réponse :**

```
% Paramètres
zeta = 0.591; % Facteur d'amortissement
omega_n = 0.36; % Fréquence naturelle

% Modèle de transfert
sys = tf([omega_n^2], [1 2*zeta*omega_n omega_n^2]);

% Réponse à un échelon
figure;
step(sys);
```

```
title('Réponse à un échelon - Méthode de Broïda');
xlabel('Temps (s)');
ylabel('Amplitude');
```

**Exercice 3 : Méthode de la Zone de Réponse pour Réglage PID**

**Problème :** Utiliser la méthode de la Zone de Réponse pour ajuster un contrôleur PID afin de minimiser le temps de stabilisation et le dépassement pour un système de premier ordre avec un retard.

**Données :**

- Gain statique : 3
- Constante de temps : 4 secondes
- Temps de retard : 1 seconde

Solution :

1. Modélisation du système :

$$G(s) = \frac{3}{1+4s}e^{-s}$$

2. Détermination des paramètres PID :
   - Utiliser la réponse transitoire pour ajuster $K_p$, $K_i$, et $K_d$.
   - Objectif : Minimiser le temps de stabilisation et le dépassement.

**Code MATLAB pour réglage PID :**

```
% Paramètres du système
K = 3; % Gain statique
T = 4; % Constante de temps
tau = 1; % Temps de retard

% Modèle de transfert du système
sys = tf(K, [T 1], 'InputDelay', tau);

% Paramètres PID
Kp = 2;
Ki = 1;
Kd = 0.5;

% Contrôleur PID
C = pid(Kp, Ki, Kd);

% Système en boucle fermée
sys_cl = feedback(C*sys, 1);

% Réponse à un échelon
figure;
step(sys_cl);
title('Réponse à un échelon du système PID ajusté - Méthode de la
Zone de Réponse');
xlabel('Temps (s)');
ylabel('Amplitude');
```

**Exercice 4 : Identification d'un Système avec Méthode de Ziegler-Nichols**

**Problème :** Appliquer la méthode de Ziegler-Nichols pour identifier les paramètres d'un système de premier ordre avec délai.

**Données :**

- Temps de retard : 2 secondes
- Temps de montée : 6 secondes
- Dépassement : 20%

Solution :

1. Estimation des paramètres :
   - Utiliser les courbes de réponse pour déterminer les paramètres PID.
   - Utiliser la méthode de la réponse transitoire pour ajuster $K_p$, $K_i$, et $K_d$.
2. Ajuster les paramètres PID en utilisant les formules de Ziegler-Nichols.

.

**Code MATLAB :**

```
% Paramètres du système
T = 4; % Constante de temps
tau = 2; % Temps de retard
K = 1; % Gain

% Modèle de transfert
sys = tf(K, [T 1], 'InputDelay', tau);

% Paramètres PID par méthode de Ziegler-Nichols
Kp = 1.2; % Ajuster selon les calculs
Ki = 0.5; % Ajuster selon les calculs
Kd = 0.1; % Ajuster selon les calculs

% Contrôleur PID
C = pid(Kp, Ki, Kd);

% Système en boucle fermée
sys_cl = feedback(C*sys, 1);

% Réponse à un échelon
figure;
step(sys_cl);
title('Réponse à un échelon - Méthode de Ziegler-Nichols');
xlabel('Temps (s)');
ylabel('Amplitude');
```

**Conclusion**

Les exercices ci-dessus démontrent l'application des méthodes d'identification graphiques pour modéliser et ajuster des systèmes réels. Ces méthodes, telles que les méthodes de Strejc, Broïda, la Zone de Réponse, et Ziegler-Nichols, offrent des moyens pratiques et efficaces pour comprendre et

ajuster le comportement des systèmes dynamiques. En utilisant MATLAB pour simuler ces systèmes, on peut mieux visualiser et affiner les modèles en fonction des besoins spécifiques du système de contrôle

# Chapitre 7 : Méthodes d'Identification Numériques

L'identification numérique est une méthode essentielle dans l'ingénierie de contrôle pour développer des modèles mathématiques à partir de données mesurées. Ces modèles sont utilisés pour analyser, prédire et contrôler le comportement de systèmes dynamiques. Les méthodes d'identification numérique sont généralement divisées en deux catégories principales : les méthodes récursives et les méthodes non récursives. Ce chapitre explore ces méthodes en détail et fournit des exercices résolus pour renforcer la compréhension.

## 1. Introduction aux Méthodes d'Identification Numériques

Les méthodes d'identification numérique utilisent des algorithmes pour ajuster les paramètres d'un modèle de système afin de correspondre aux données observées. La qualité de l'identification dépend de la qualité des données, du choix du modèle, et des techniques utilisées pour ajuster les paramètres.

**Types de Méthodes d'Identification :**

- **Méthodes Récursives** : Ces méthodes mettent à jour les paramètres du modèle en temps réel, à mesure que de nouvelles données deviennent disponibles. Elles sont utilisées dans des systèmes adaptatifs et en ligne.
- **Méthodes Non Récursives** : Ces méthodes utilisent l'ensemble des données disponibles pour estimer les paramètres en une seule fois. Elles sont généralement plus précises mais nécessitent plus de calculs.

## 2. Méthodes Récursives d'Identification

Les méthodes récursives sont particulièrement utiles pour les systèmes qui changent avec le temps, où les paramètres du modèle doivent être mis à jour continuellement. Ces méthodes permettent de suivre l'évolution d'un système en ajustant les paramètres du modèle à chaque nouvelle observation.

### 2.1 Algorithme de Moindres Carrés Récursifs (RLS)

L'algorithme de moindres carrés récursifs (RLS) est une méthode populaire pour l'estimation des paramètres des modèles linéaires. Il ajuste les paramètres du modèle de manière itérative pour minimiser l'erreur quadratique entre les observations et les prédictions.

**Formulation mathématique :**

Soit un modèle linéaire :

$$y(t) = \theta^T \phi(t) + e(t)$$

où :

- $y(t)$ est la sortie observée,
- $\theta$ est le vecteur de paramètres à estimer,
- $\phi(t)$ est le vecteur des variables régressives,
- $e(t)$ est l'erreur de modélisation.

L'algorithme RLS met à jour $\theta$ en minimisant $\sum_{i=1}^{t} \lambda^{t-i}(y(i) - \theta^T \phi(i))^2$, où $\lambda$ est un facteur de décroissance ( $0 < \lambda \leq 1$).

**Mise à jour des paramètres :**

1. Initialiser $\theta(0)$ et $P(0)$ (matrice de covariance).
2. Pour chaque nouvelle donnée $(y(t), \phi(t))$ :
   - Calculer le gain de Kalman : $K(t) = \frac{P(t-1)\phi(t)}{\lambda + \phi(t)^T P(t-1)\phi(t)}$
   - Mettre à jour les paramètres : $\theta(t) = \theta(t-1) + K(t)(y(t) - \phi(t)^T \theta(t-1))$
   - Mettre à jour la matrice de covariance : $P(t) = \frac{1}{\lambda}(P(t-1) - K(t)\phi(t)^T P(t-1))$

**Exercice 1 : Estimation des Paramètres d'un Système de Premier Ordre avec RLS**

**Problème :** Considérons un système de premier ordre avec une entrée u(t) et une sortie y(t) :

$$y(t) = a \cdot y(t-1) + b \cdot u(t-1) + e(t)$$

où $a$ et $b$ sont des paramètres inconnus, et $e(t)$ est un bruit blanc gaussien. Utilisez RLS pour estimer $a$ et $b$.

Solution :

1. **Formulation du vecteur régressif** : $\phi(t) = [y(t-1), u(t-1)]^T$.
2. **Vecteur des paramètres** : $\theta = [a, b]^T$.
3. **Algorithme RLS** :
   - Initialiser $\theta(0) = [0, 0]^T$ et $P(0) = \text{diag}(1000, 1000)$.
   - Utiliser les nouvelles données pour mettre à jour $\theta(t)$ et $P(t)$.

**Code MATLAB :**

```
% Données simulées
N = 100;
u = randn(N,1); % Entrée aléatoire
y = zeros(N,1); % Sortie

% Paramètres réels
a_real = 0.7;
b_real = 0.3;

% Génération des données
for t = 2:N
    y(t) = a_real * y(t-1) + b_real * u(t-1) + 0.1*randn;
end

% Initialisation
theta = [0; 0];
P = 1000*eye(2);
lambda = 0.99; % Facteur de décroissance

% RLS
```

```
for t = 2:N
    phi = [y(t-1); u(t-1)];
    K = P*phi / (lambda + phi'*P*phi);
    theta = theta + K * (y(t) - phi'*theta);
    P = (1/lambda)*(P - K*phi'*P);
end

% Affichage des résultats
disp('Paramètres estimés :');
disp(theta);
disp('Paramètres réels :');
disp([a_real; b_real]);
```

### 3. Méthodes Non Récursives d'Identification

Les méthodes non récursives utilisent l'ensemble des données disponibles pour estimer les paramètres d'un modèle en une seule fois. Ces méthodes sont généralement plus précises car elles minimisent globalement l'erreur sur toutes les données.

#### 3.1 Méthode des Moindres Carrés Ordinaires (OLS)

La méthode des moindres carrés ordinaires (OLS) est utilisée pour estimer les paramètres des modèles linéaires. Elle minimise la somme des carrés des erreurs entre les observations et les prédictions.

**Formulation mathématique :**

Pour un modèle linéaire $y = X\theta + e$, où $y$ est un vecteur des observations, $X$ est la matrice des régressions, et $e$ est l'erreur, l'estimation des moindres carrés est donnée par :

$$\hat{\theta} = (X^T X)^{-1} X^T y$$

**Problème :** Considérons un système de premier ordre avec une entrée $u(t)$ et une sortie $y(t)$ :

$$y(t) = a \cdot y(t-1) + b \cdot u(t-1) + e(t)$$

où $a$ et $b$ sont des paramètres inconnus, et $e(t)$ est un bruit blanc gaussien. Utilisez RLS pour estimer $a$ et $b$.

**Solution :**

1. **Formulation du vecteur régressif :** $\phi(t) = [y(t-1), u(t-1)]^T$.
2. **Vecteur des paramètres :** $\theta = [a, b]^T$.
3. **Algorithme RLS :**
   - Initialiser $\theta(0) = [0, 0]^T$ et $P(0) = \text{diag}(1000, 1000)$.
   - Utiliser les nouvelles données pour mettre à jour $\theta(t)$ et $P(t)$.

**Code MATLAB :**

```
% Données simulées
N = 100;
u = randn(N,1); % Entrée aléatoire
y = zeros(N,1); % Sortie
```

```
% Paramètres réels
a1_real = 0.5;
a2_real = 0.2;
b1_real = 0.4;
b2_real = 0.3;

% Génération des données
for t = 3:N
    y(t) = a1_real * y(t-1) + a2_real * y(t-2) + b1_real * u(t-1) +
b2_real * u(t-2) + 0.1*randn;
end

% Construction de la matrice des régressions et du vecteur des
observations
X = [y(2:end-1), y(1:end-2), u(2:end-1), u(1:end-2)];
y_obs = y(3:end);

% Estimation des paramètres par OLS
theta_est = (X'*X)\(X'*y_obs);

% Affichage des résultats
disp('Paramètres estimés :');
disp(theta_est);
disp('Paramètres réels :');
disp([a1_real; a2_real; b1_real; b2_real]);
```

**4. Comparaison entre Méthodes Récursives et Non Récursives**

- **Méthodes récursives** : Utiles pour les systèmes en ligne, adaptatifs. Moins précises sur l'ensemble des données mais capables de suivre des changements dynamiques.
- **Méthodes non récursives** : Plus précises globalement, mais nécessitent des calculs plus lourds et l'accès à toutes les données.

**Conclusion**

L'identification numérique est un outil puissant pour comprendre et modéliser des systèmes dynamiques réels. Les méthodes récursives et non récursives offrent des avantages distincts selon les applications. Les exercices pratiques avec MATLAB démontrent l'application de ces méthodes et fournissent des résultats clairs pour l'estimation des paramètres des systèmes.

# Application : Méthodes d'Identification Numériques pour Systèmes Couplés et Multivariables

Dans ce chapitre, nous allons explorer les méthodes d'identification numériques, à la fois récursives et non récursives, appliquées à des systèmes couplés et multivariables. Nous fournirons des exercices détaillés pour illustrer comment ces méthodes peuvent être utilisées pour identifier des modèles précis de systèmes complexes. Chaque section comprendra cinq exercices résolus, ce qui permettra de comprendre comment appliquer ces techniques dans des situations réelles.

**Introduction aux Systèmes Couplés et Multivariables**

Les systèmes couplés et multivariables comportent plusieurs variables d'entrée et de sortie qui interagissent. L'objectif de l'identification est de trouver un modèle mathématique qui décrit le comportement dynamique du système en utilisant des données expérimentales. Les méthodes récursives sont particulièrement utiles pour les systèmes adaptatifs ou en temps réel, tandis que les méthodes non récursives sont souvent utilisées pour des analyses hors ligne plus détaillées.

**Méthodes Récursives d'Identification**

Les méthodes récursives mettent à jour les paramètres du modèle au fur et à mesure que de nouvelles données deviennent disponibles. Voici cinq exercices pour illustrer l'utilisation des méthodes récursives pour les systèmes couplés et multivariables.

**Exercice 1 : Estimation des Paramètres d'un Système DC Couplé avec RLS**

**Problème :** Considérons un système de deux moteurs DC couplés avec des entrées $V_1$ et $V_2$ et des sorties de vitesse $\omega_1$ et $\omega_2$. Les équations sont données par :

$$L\frac{di_1}{dt} + Ri_1 + K_e\omega_1 = V_1(t)$$
$$J\frac{d\omega_1}{dt} + B\omega_1 = K_t i_1$$
$$L\frac{di_2}{dt} + Ri_2 + K_e\omega_2 = V_2(t)$$
$$J\frac{d\omega_2}{dt} + B\omega_2 = K_t i_2$$

**Objectif :** Utiliser l'algorithme des moindres carrés récursifs (RLS) pour estimer les paramètres $R$, $L$, $K_e$, $K_t$, $J$, et $B$.

**Solution :**

1. **Définir les vecteurs régressifs :** Les équations peuvent être écrites sous forme d'un modèle de régression linéaire. Pour chaque moteur, le vecteur régressif peut inclure les termes $i_1$, $di_1/dt$, $\omega_1$, etc.
2. **Application de RLS :** Initialiser les paramètres et utiliser RLS pour estimer les paramètres du modèle à chaque itération.

.**Code MATLAB :**

```
% Simulation des données pour un moteur DC couplé
N = 100;
V1 = randn(N,1); % Tension d'entrée V1
```

```
V2 = randn(N,1); % Tension d'entrée V2
omega1 = zeros(N,1); % Vitesse angulaire omega1
omega2 = zeros(N,1); % Vitesse angulaire omega2

% Paramètres réels
R_real = 1.0;
L_real = 0.5;
Ke_real = 0.01;
Kt_real = 0.01;
J_real = 0.01;
B_real = 0.1;

% Génération des données
for t = 2:N
    omega1(t) = omega1(t-1) + (Kt_real * V1(t-1) - B_real *
omega1(t-1)) / J_real;
    omega2(t) = omega2(t-1) + (Kt_real * V2(t-1) - B_real *
omega2(t-1)) / J_real;
end

% Initialisation RLS
theta = [0; 0; 0; 0];
P = 1000 * eye(4);
lambda = 0.99;

% Exécution de RLS
for t = 2:N
    phi = [omega1(t-1); omega2(t-1); V1(t-1); V2(t-1)];
    K = P * phi / (lambda + phi' * P * phi);
    theta = theta + K * ([omega1(t); omega2(t)] - phi' * theta);
    P = (1/lambda) * (P - K * phi' * P);
end

% Résultats
disp('Paramètres estimés :');
disp(theta);
```

**Exercice 2 : Identification d'un Système Thermique Couplé**

**Problème :** Un système thermique avec deux zones interconnectées où la température de chaque zone $T_1$ et $T_2$ est influencée par les chauffages $Q_1$ et $Q_2$ et la chaleur échangée entre les zones.

**Modèle :**

$$C_1 \frac{dT_1}{dt} = -k_{12}(T_1 - T_2) + Q_1$$
$$C_2 \frac{dT_2}{dt} = -k_{21}(T_2 - T_1) + Q_2$$

**Objectif :** Utiliser RLS pour estimer les paramètres $k_{12}$, $k_{21}$, $C_1$, et $C_2$.

**Solution :**

1. **Définir les vecteurs régressifs :** $\phi(t) = [T_1(t-1), T_2(t-1), Q_1(t-1), Q_2(t-1)]$.
2. **Application de RLS :** Initialiser les paramètres et utiliser RLS pour estimer les paramètres à chaque itération.

*Code MATLAB :*

```
% Simulation des données pour un système thermique
N = 100;
Q1 = randn(N,1); % Chauffage Q1
Q2 = randn(N,1); % Chauffage Q2
T1 = zeros(N,1); % Température T1
T2 = zeros(N,1); % Température T2

% Paramètres réels
k12_real = 0.1;
k21_real = 0.1;
C1_real = 5.0;
C2_real = 5.0;

% Génération des données
for t = 2:N
    T1(t) = T1(t-1) + (Q1(t-1) - k12_real * (T1(t-1) - T2(t-1))) /
C1_real;
    T2(t) = T2(t-1) + (Q2(t-1) - k21_real * (T2(t-1) - T1(t-1))) /
C2_real;
end

% Initialisation RLS
theta = [0; 0; 0; 0];
P = 1000 * eye(4);
lambda = 0.99;

% Exécution de RLS
for t = 2:N
    phi = [T1(t-1); T2(t-1); Q1(t-1); Q2(t-1)];
    K = P * phi / (lambda + phi' * P * phi);
    theta = theta + K * ([T1(t); T2(t)] - phi' * theta);
    P = (1/lambda) * (P - K * phi' * P);
end

% Résultats
disp('Paramètres estimés :');
disp(theta);
```

**Méthodes Non Récursives d'Identification**

Les méthodes non récursives sont utilisées pour analyser un ensemble complet de données et fournir une estimation globale des paramètres du modèle. Voici cinq exercices pour illustrer l'utilisation des méthodes non récursives pour les systèmes couplés et multivariables.

**Exercice 3 : Estimation des Paramètres d'un Réacteur Chimique Couplé avec OLS**

**Problème :** Considérons un réacteur chimique avec deux réactifs $A$ et $B$ qui réagissent pour former des produits $C$. Les concentrations $C_A$ et $C_B$ dépendent des taux de réaction et de la température.

**Modèle :**

$$\frac{dC_A}{dt} = -k_1 C_A C_B + k_2 T$$
$$\frac{dC_B}{dt} = -k_1 C_A C_B + k_3 T$$

**Objectif :** Utiliser OLS pour estimer les paramètres $k_1$, $k_2$, et $k_3$.

**Solution :**

1. **Formuler les équations de régression :** Construire un vecteur régressif pour inclure les termes nécessaires.
2. **Utiliser OLS pour estimer les paramètres.**

**Code MATLAB :**

```
% Simulation des données pour un réacteur chimique couplé
N = 100;
T = 300 + 10*randn(N,1); % Température aléatoire
CA = zeros(N,1); % Concentration de A
CB = zeros(N,1); % Concentration de B

% Paramètres réels
k1_real = 0.02;
k2_real = 0.1;
k3_real = 0.1;

% Génération des données
for t = 2:N
    CA(t) = CA(t-1) - k1_real * CA(t-1) * CB(t-1) + k2_real * T(t-
1);
    CB(t) = CB(t-1) - k1_real * CA(t-1) * CB(t-1) + k3_real * T(t-
1);
end

% Construction de la matrice des régressions et du vecteur des
observations
X = [CA(2:end-1).*CB(2:end-1), T(2:end-1)];
yA_obs = CA(3:end);
yB_obs = CB(3:end);

% Estimation des paramètres par OLS
theta_A = (X'*X)\(X'*yA_obs);
theta_B = (X'*X)\(X'*yB_obs);

% Affichage des résultats
disp('Paramètres estimés pour CA :');
disp(theta_A);
disp('Paramètres estimés pour CB :');
disp(theta_B);
```

**Conclusion**

Les exercices ci-dessus illustrent comment appliquer des méthodes d'identification numériques, à la fois récursives et non récursives, à des systèmes couplés et multivariables. En utilisant ces méthodes, on peut identifier les paramètres des modèles qui décrivent précisément le comportement dynamique des systèmes complexes. Ces techniques sont essentielles pour la conception, l'analyse et le contrôle des systèmes industriels modernes

# Projets : Méthodes d'Identification Numériques pour Systèmes Réels, Couplés et Multivariables après Linéarisation

L'identification numérique de systèmes réels, couplés et multivariables est une tâche essentielle dans l'ingénierie de contrôle. Après la linéarisation, les systèmes non linéaires complexes peuvent être approximés par des modèles linéaires autour d'un point d'équilibre. Ce chapitre propose des projets d'exercices bien formulés pour appliquer des méthodes récursives et non récursives d'identification numérique à des systèmes réels, couplés et multivariables.

**Introduction aux Systèmes Couplés et Multivariables**

Les systèmes réels impliquent souvent des interactions entre plusieurs variables d'entrée et de sortie, ce qui conduit à des systèmes couplés et multivariables. Ces systèmes peuvent être linéarisés autour de points d'équilibre pour simplifier l'analyse et le contrôle. L'objectif de l'identification numérique est de développer des modèles mathématiques précis de ces systèmes à partir de données expérimentales.

**Méthodes d'identification numérique** :

- **Méthodes Récursives** : Utilisent des algorithmes qui mettent à jour les paramètres en temps réel.
- **Méthodes Non Récursives** : Utilisent l'ensemble des données disponibles pour estimer les paramètres en une seule fois.

**Projets d'Exercices Résolus**

**Projet 1 : Identification d'un Système de Moteur DC Couplé avec Méthode Récursive**

Projet 1 : Identification d'un Système de Moteur DC Couplé avec Méthode Récursive

**Problème :** Un système de deux moteurs DC couplés partage un arbre commun. Les entrées sont les tensions $V_1$ et $V_2$, et les sorties sont les vitesses angulaires $\omega_1$ et $\omega_2$. Linéarisez le système autour d'un point d'équilibre et utilisez l'algorithme de moindres carrés récursifs (RLS) pour identifier les paramètres du modèle.

Modèle Linéaire :

Les équations non linéaires des moteurs sont linéarisées autour d'un point d'équilibre $(\omega_{1e}, \omega_{2e}, V_{1e}, V_{2e})$ :

$$\frac{d\omega_1}{dt} = a_{11}\omega_1 + a_{12}\omega_2 + b_{11}V_1 + b_{12}V_2$$
$$\frac{d\omega_2}{dt} = a_{21}\omega_1 + a_{22}\omega_2 + b_{21}V_1 + b_{22}V_2$$

**Objectif :** Estimer les paramètres $a_{ij}$ et $b_{ij}$ à l'aide de RLS.

**Solution :**

1. **Linéarisation** : Autour d'un point d'équilibre, les non-linéarités sont négligées, et les termes de dérivée sont approximés linéairement.

2. **Formulation des vecteurs régressifs** :
   - $\phi(t) = [\omega_1(t-1), \omega_2(t-1), V_1(t-1), V_2(t-1)]^T$
   - $y(t) = [\omega_1(t), \omega_2(t)]^T$
3. **Application de RLS** : Mettre à jour les paramètres $\theta = [a_{11}, a_{12}, b_{11}, b_{12}, a_{21}, a_{22}, b_{21}, b_{22}]^T$ à chaque nouvelle observation.

**Code MATLAB :**

```
% Simulation de données pour deux moteurs DC couplés
N = 100;
V1 = randn(N,1); % Tension d'entrée V1
V2 = randn(N,1); % Tension d'entrée V2
omega1 = zeros(N,1); % Vitesse angulaire omega1
omega2 = zeros(N,1); % Vitesse angulaire omega2

% Paramètres réels du modèle linéarisé
a11_real = 0.1; a12_real = 0.05;
a21_real = 0.07; a22_real = 0.1;
b11_real = 0.2; b12_real = 0.1;
b21_real = 0.15; b22_real = 0.2;

% Génération des données
for t = 2:N
    omega1(t)   =   a11_real*omega1(t-1)   +   a12_real*omega2(t-1)   +
b11_real*V1(t-1) + b12_real*V2(t-1);
    omega2(t)   =   a21_real*omega1(t-1)   +   a22_real*omega2(t-1)   +
b21_real*V1(t-1) + b22_real*V2(t-1);
end

% Initialisation RLS
theta = zeros(8,1); % [a11, a12, b11, b12, a21, a22, b21, b22]
P = 1000 * eye(8);
lambda = 0.99;

% Exécution de RLS
for t = 2:N
    phi = [omega1(t-1), omega2(t-1), V1(t-1), V2(t-1), omega1(t-1),
omega2(t-1), V1(t-1), V2(t-1)]';
    y = [omega1(t); omega2(t)];
    K = P * phi / (lambda + phi' * P * phi);
    theta = theta + K * (y - phi' * theta);
    P = (1/lambda) * (P - K * phi' * P);
end

% Résultats
disp('Paramètres estimés :');
disp(theta);
```

**Projet 2 : Identification d'un Système d'Avion Couplé avec Méthode Récursive**

**Problème :** Considérons un système de contrôle d'attitude d'un avion, où les entrées sont les angles d'attaque et les commandes de gouvernes, et les sorties sont les angles de roulis ($\phi$), de tangage ($\theta$) et de lacet ($\psi$). Après linéarisation, utilisez RLS pour estimer les paramètres du modèle.

**Modèle Linéaire :**

$$\frac{d\phi}{dt} = a_{11}\phi + a_{12}\theta + b_{11}u_1 + b_{12}u_2$$
$$\frac{d\theta}{dt} = a_{21}\phi + a_{22}\theta + b_{21}u_1 + b_{22}u_2$$
$$\frac{d\psi}{dt} = a_{31}\phi + a_{32}\psi + b_{31}u_1 + b_{32}u_2$$

**Objectif :** Estimer les paramètres $a_{ij}$ et $b_{ij}$ pour le modèle d'attitude.

**Solution :**

1. **Linéarisation** : Réduire les équations de mouvement à une forme linéaire autour d'un point d'équilibre.

2. **Formulation des vecteurs régressifs** :
   - $\phi(t) = [\phi(t-1), \theta(t-1), u_1(t-1), u_2(t-1)]^T$
3. **Application de RLS** : Utiliser RLS pour estimer $\theta$ pour chaque équation d'état.

**Code MATLAB :**

```
% Simulation de données pour système d'attitude d'un avion
N = 100;
u1 = randn(N,1); % Commande de gouverne u1
u2 = randn(N,1); % Commande de gouverne u2
phi = zeros(N,1); % Angle de roulis
theta = zeros(N,1); % Angle de tangage
psi = zeros(N,1); % Angle de lacet

% Paramètres réels du modèle linéarisé
a11_real = 0.2; a12_real = 0.1;
a21_real = 0.1; a22_real = 0.2;
a31_real = 0.05; a32_real = 0.15;
b11_real = 0.1; b12_real = 0.05;
b21_real = 0.05; b22_real = 0.1;
b31_real = 0.1; b32_real = 0.1;

% Génération des données
for t = 2:N
    phi(t)    =    a11_real*phi(t-1)    +    a12_real*theta(t-1)    +
b11_real*u1(t-1) + b12_real*u2(t-1);
    theta(t)    =    a21_real*phi(t-1)    +    a22_real*theta(t-1)    +
b21_real*u1(t-1) + b22_real*u2(t-1);
    psi(t) = a31_real*phi(t-1) + a32_real*psi(t-1) + b31_real*u1(t-
1) + b32_real*u2(t-1);
```

```
end

% Initialisation RLS
theta_params = zeros(10,1); % [a11, a12, b11, b12, a21, a22, b21,
b22, a31, a32]
P = 1000 * eye(10);
lambda = 0.99;

% Exécution de RLS
for t = 2:N
    phi_vect = [phi(t-1), theta(t-1), u1(t-1), u2(t-1), phi(t-1),
theta(t-1), u1(t-1), u2(t-1), phi(t-1), psi(t-1)]';
    y_vect = [phi(t); theta(t); psi(t)];
    K = P * phi_vect / (lambda + phi_vect' * P * phi_vect);
    theta_params = theta_params + K * (y_vect - phi_vect' *
theta_params);
    P = (1/lambda) * (P - K * phi_vect' * P);
end

% Résultats
disp('Paramètres estimés :');
disp(theta_params);
```

**Projet 3 : Identification d'un Système de Missile Couplé avec Méthode Non Récursive**

**Problème :** Un missile a une dynamique de vol où les angles longitudinaux et latéraux sont affectés par des perturbations atmosphériques et des commandes de surface. Utilisez une méthode de moindres carrés ordinaires (OLS) pour estimer les paramètres du modèle après linéarisation.

Modèle Linéaire :

$$\frac{d\alpha}{dt} = a_{11}\alpha + a_{12}\beta + b_{11}\delta_e + b_{12}\delta_r$$
$$\frac{d\beta}{dt} = a_{21}\alpha + a_{22}\beta + b_{21}\delta_e + b_{22}\delta_r$$

Objectif : Estimer les paramètres $a_{ij}$ et $b_{ij}$ à l'aide de OLS.

Solution :

1. **Linéarisation** : Convertir les équations de vol en un modèle linéaire autour d'un point d'équilibre.
2. **Formulation des équations régressives** :
   - $\phi(t) = [\alpha(t-1), \beta(t-1), \delta_e(t-1), \delta_r(t-1)]^T$
3. **Utilisation d'OLS** : Estimer $\theta = [a_{11}, a_{12}, b_{11}, b_{12}, a_{21}, a_{22}, b_{21}, b_{22}]^T$.

**Code MATLAB :**

```
% Simulation de données pour un système de missile couplé
N = 100;
delta_e = randn(N,1); % Commande d'élévateur
```

```
delta_r = randn(N,1); % Commande de gouvernail
alpha = zeros(N,1); % Angle d'attaque
beta = zeros(N,1); % Angle de dérapage

% Paramètres réels du modèle linéarisé
a11_real = 0.3; a12_real = 0.1;
a21_real = 0.1; a22_real = 0.3;
b11_real = 0.2; b12_real = 0.1;
b21_real = 0.1; b22_real = 0.2;

% Génération des données
for t = 2:N
    alpha(t)    =    a11_real*alpha(t-1)    +    a12_real*beta(t-1)    +
b11_real*delta_e(t-1) + b12_real*delta_r(t-1);
    beta(t)    =    a21_real*alpha(t-1)    +    a22_real*beta(t-1)    +
b21_real*delta_e(t-1) + b22_real*delta_r(t-1);
end

% Construction de la matrice des régressions et du vecteur des
observations
X = [alpha(2:end-1), beta(2:end-1), delta_e(2:end-1), delta_r(2:end-
1)];
y_alpha = alpha(3:end);
y_beta = beta(3:end);

% Estimation des paramètres par OLS
theta_alpha = (X'*X)\(X'*y_alpha);
theta_beta = (X'*X)\(X'*y_beta);

% Affichage des résultats
disp('Paramètres estimés pour alpha :');
disp(theta_alpha);
disp('Paramètres estimés pour beta :');
disp(theta_beta);
```

## Projet 4 : Modélisation Multivariables d'un Réacteur Chimique

Les réacteurs chimiques multivariables présentent des interactions complexes entre les espèces chimiques, la température et le débit. Ces interactions peuvent être modélisées par des équations différentielles décrivant les bilans de masse et d'énergie.

### 2.1 Équations Différentielles pour un Réacteur Chimique

1. **Équation de Bilan de Masse pour le Réactif $A$ :**

$$\frac{dC_A}{dt} = \frac{F_{in}}{V}(C_{A,in} - C_A) - k_1 C_A - k_2 C_A^2$$

Où $C_A$ est la concentration du réactif A, $F_{in}$ est le débit d'entrée, $V$ est le volume du réacteur, $k_1$ et $k_2$ sont les constantes de réaction.

2. **Équation de Bilan de Masse pour le Réactif $B$ :**

$$\frac{dC_B}{dt} = \frac{F_{in}}{V}(C_{B,in} - C_B) - k_3 C_B$$

Où $C_B$ est la concentration du réactif B.

3. **Équation de Bilan d'Énergie :**

$$\frac{dT}{dt} = \frac{Q}{\rho C_p V} - \frac{F_{in}}{V}(T - T_{in}) + \frac{-\Delta H_r}{\rho C_p} k_1 C_A$$

Où $T$ est la température du réacteur, $Q$ est le flux de chaleur, $\rho$ est la densité du mélange, $C_p$ est la capacité thermique, et $\Delta H_r$ est la chaleur de réaction.

### 2.2 Identification ARMAX pour le Réacteur

Pour modéliser l'influence des variables d'entrée (débit, température d'entrée) sur les concentrations et la température du réacteur, nous utilisons un modèle ARMAX.

1. **Modèle ARMAX pour la Concentration de $A$ ($C_A(t)$) :**

$$C_A(t) = a_{11} C_A(t-1) + b_{11} u_1(t-1) + e_1(t)$$

Où $u_1(t)$ est l'entrée (e.g., débit).

2. **Modèle ARMAX pour la Température ($T(t)$) :**

$$T(t) = a_{21} T(t-1) + b_{21} u_2(t-1) + e_2(t)$$

Où $u_2(t)$ est l'entrée de température.

Ces équations modélisent la dynamique du système en prenant en compte les réactions chimiques et les flux thermiques.

### 2.2 Identification ARMAX pour le Réacteur

L'objectif de l'identification ARMAX est de modéliser l'influence des variables d'entrée (telles que le débit et la température d'entrée) sur les concentrations des réactifs et la température du réacteur.

1. **Modèle ARMAX pour la Concentration de A ($C_A$):**

$$C_A(t) = a_{11}C_A(t-1) + b_{11}u_1(t-1) + e_1(t)$$

- $C_A(t)$: Concentration de A à l'instant $t$
- $u_1(t)$: Entrée, par exemple le débit
- $a_{11}$, $b_{11}$: Paramètres du modèle ARMAX
- $e_1(t)$: Terme d'erreur

2. **Modèle ARMAX pour la Température ($T$):**

$$T(t) = a_{21}T(t-1) + b_{21}u_2(t-1) + e_2(t)$$

- $T(t)$: Température à l'instant $t$
- $u_2(t)$: Entrée de température
- $a_{21}$, $b_{21}$: Paramètres du modèle ARMAX
- $e_2(t)$: Terme d'erreur

Ces modèles ARMAX sont utilisés pour prédire l'évolution des concentrations et de la température en fonction des conditions d'entrée.

### Code MATLAB pour l'Identification ARMAX

Voici un exemple de code MATLAB pour effectuer une identification ARMAX sur un réacteur chimique, en utilisant des données simulées pour les concentrations et la température :

```
% Données simulées pour les concentrations et la température
conc_A = 1.5 + 0.1*randn(1500,1); % Concentration de A avec du bruit
temp = 350 + 5*randn(1500,1); % Température avec du bruit
data = [conc_A temp];

% Génération de SBPA pour les entrées (débit et température)
n = 1500; % Nombre de points
num_channels = 2; % Nombre de canaux pour le système multivariable
sbpa = idinput([n, num_channels], 'prbs', [0 1], [-1 1]); % SBPA
entre -1 et 1
u = sbpa; % Utiliser la SBPA comme entrée

data = [data u]; % Données d'entrée-sortie
nk = 1; % Délai entre entrée et sortie

% Modèle ARMAX multivariable
na = 2; % Ordre du modèle AR
nb = 2; % Ordre de la partie MA
model_ARMAX = armax(data, [na nb nk]);

% Affichage des résultats
disp('Paramètres du Modèle ARMAX Multivariable (Réacteur
Chimique):');
disp('A:');
disp(model_ARMAX.A);
```

```
disp('B:');
disp(model_ARMAX.B);
disp('C:');
disp(model_ARMAX.C)
```

**Conclusion**

Les projets d'exercices présentés illustrent l'application de méthodes d'identification numériques, aussi bien récursives que non récursives, pour des systèmes réels, couplés et multivariables. Les exemples fournis démontrent comment linéariser des systèmes complexes et utiliser des algorithmes de moindres carrés récursifs et non récursifs pour estimer les paramètres du modèle. Ces techniques sont fondamentales pour le contrôle, l'analyse et la conception de systèmes dans de nombreuses applications d'ingénierie

# Chapitre 8

# Estimation et Observation (02 Semaines)

## Introduction

L'estimation et l'observation sont des aspects fondamentaux dans l'ingénierie de contrôle des systèmes. Ils permettent de déduire des informations sur l'état interne d'un système basé sur les mesures des sorties, particulièrement lorsque les états du système ne peuvent pas être mesurés directement. Ce chapitre explore les méthodes d'estimation et d'observation des systèmes électriques, couvrant des estimateurs spécifiques tels que l'estimateur de Gopinath, l'observateur de Luenberger, et le filtre de Kalman pour les systèmes stochastiques. Ces techniques sont illustrées à travers des exemples pratiques et des exercices détaillés.

## 1. Estimation des Systèmes Électriques

Les systèmes électriques tels que les moteurs, les générateurs, et les circuits RC nécessitent souvent des estimations d'états internes (tels que le courant et la vitesse) qui ne peuvent pas être mesurés directement. L'estimation des systèmes électriques implique l'utilisation de modèles mathématiques pour prédire ces états en fonction des mesures disponibles.

### 1.1 Estimateur de Gopinath

L'estimateur de Gopinath est un exemple d'estimateur utilisé pour les systèmes électriques, comme les moteurs à courant continu (DC). Cet estimateur utilise les équations du modèle pour estimer les états du système.

**Exemple : Estimation du courant et de la vitesse dans un moteur DC**

Modèle du moteur DC :

$$L\frac{di(t)}{dt} + Ri(t) + K_e\omega(t) = V(t)$$

$$J\frac{d\omega(t)}{dt} + B\omega(t) = K_t i(t)$$

où :

- $i(t)$ est le courant dans le moteur.
- $\omega(t)$ est la vitesse angulaire.
- $V(t)$ est la tension appliquée.
- $L, R, K_e, K_t, J, B$ sont des constantes.

**Objectif :** Estimer le courant et la vitesse angulaire.

**Algorithme de l'Estimateur de Gopinath :**

1. Utiliser un modèle discret du système.
2. Appliquer un filtre à gain fixe pour l'estimation.

**Objectif :** Estimer le courant et la vitesse angulaire.

**Algorithme de l'Estimateur de Gopinath :**

1. Utiliser un modèle discret du système.
2. Appliquer un filtre à gain fixe pour l'estimation.

**Code MATLAB :**

```
% Paramètres du moteur DC
L = 0.5; % Inductance (H)
R = 1; % Résistance (Ohm)
Ke = 0.01; % Constante de FEM (V/rad/s)
Kt = 0.01; % Constante de couple (Nm/A)
J = 0.01; % Moment d'inertie (kg*m^2)
B = 0.1; % Coefficient de frottement (Nm*s/rad)
dt = 0.01; % Intervalle de temps (s)
T = 5; % Durée de la simulation (s)

% Modèle du système discret
A = [1, -dt*R/L; dt*Kt/J, 1-dt*B/J];
B = [dt/L; 0];
C = [0, 1]; % Mesurer la vitesse angulaire
D = 0;
% Variables d'état
x = [0; 0]; % [Courant; Vitesse]
x_est = [0; 0]; % Estimation

% Entrée (tension)
V = 1; % Tension constante
% Simulation
time = 0:dt:T;
current = zeros(1, length(time));
speed = zeros(1, length(time));

for k = 1:length(time)
    % Système réel
    x = A*x + B*V;
    current(k) = x(1);
    speed(k) = x(2);

    % Estimation de Gopinath
    x_est = A*x_est + B*V + [0; dt*(current(k) - x_est(1))/L];
end

% Affichage des résultats
figure;
subplot(2,1,1);
plot(time, current, 'r', time, speed, 'b');
title('Courant et Vitesse du Moteur DC');
xlabel('Temps (s)');
ylabel('Amplitude');
legend('Courant', 'Vitesse');
subplot(2,1,2);
plot(time, speed, 'r', time, x_est(2,:), 'b');
title('Vitesse : Réel vs Estimé');
xlabel('Temps (s)');
ylabel('Vitesse (rad/s)');
legend('Réel', 'Estimé');
```

## 2. Observation Déterministe : Observateur de Luenberger

L'observateur de Luenberger est une technique utilisée pour estimer les états d'un système linéaire à partir de la sortie mesurée. Il est particulièrement utile lorsque toutes les variables d'état ne peuvent pas être mesurées directement. L'observateur de Luenberger est basé sur l'utilisation d'un modèle du système avec une rétroaction pour corriger l'estimation en fonction de la différence entre la sortie mesurée et la sortie estimée.

### 2.1 Principe de l'Observateur de Luenberger

Considérons un système linéaire invariant dans le temps :

$$\dot{x} = Ax + Bu$$
$$y = Cx$$

L'observateur de Luenberger est donné par :

$$\dot{\hat{x}} = A\hat{x} + Bu + L(y - \hat{y})$$

où :

- $\hat{x}$ est l'estimation de l'état.
- $L$ est le gain de l'observateur.
- $y$ est la sortie mesurée.
- $\hat{y} = C\hat{x}$ est la sortie estimée.

**Conception du gain de l'observateur :** Choisir $L$ de manière à ce que les pôles de l'observateur soient à des positions souhaitées pour assurer la convergence rapide de l'estimation.

### 2.2 Exemple d'Application : Système de Commande de Position

Modèle :

$$\dot{x} = \begin{bmatrix} 0 & 1 \\ 0 & -B/J \end{bmatrix} x + \begin{bmatrix} 0 \\ 1/J \end{bmatrix} u$$
$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} x$$

**Objectif :** Estimer la position et la vitesse.

**Code MATLAB :**

```
% Paramètres du système

J = 0.01; % Moment d'inertie
B = 0.1; % Coefficient de frottement

A = [0 1; 0 -B/J];
B = [0; 1/J];
C = [1 0];
D = 0;

% Gain de l'observateur
L = place(A', C', [-2 -3])'; % Placement des pôles de l'observateur
```

```
% Simulation
dt = 0.01; % Intervalle de temps
T = 10; % Temps total de simulation
time = 0:dt:T;
x = [0; 0]; % États réels
x_hat = [0; 0]; % États estimés
u = 1; % Entrée constante

x_history = zeros(2, length(time));
x_hat_history = zeros(2, length(time));

for k = 1:length(time)
    % Système réel
    x = x + (A*x + B*u)*dt;
    y = C*x; % Mesure de sortie

    % Observateur de Luenberger
    x_hat = x_hat + (A*x_hat + B*u + L*(y - C*x_hat))*dt;

    % Stocker les données
    x_history(:, k) = x;
    x_hat_history(:, k) = x_hat;
end

% Affichage des résultats
figure;
subplot(2,1,1);
plot(time, x_history(1,:), 'r', time, x_hat_history(1,:), 'b');
title('Position : Réel vs Estimé');
xlabel('Temps (s)');
ylabel('Position (m)');
legend('Réel', 'Estimé');

subplot(2,1,2);
plot(time, x_history(2,:), 'r', time, x_hat_history(2,:), 'b');
title('Vitesse : Réel vs Estimé');
xlabel('Temps (s)');
ylabel('Vitesse (m/s)');
legend('Réel', 'Estimé');
```

### 3. Observateurs Non-déterministes : Filtre de Kalman

Le filtre de Kalman est un observateur optimal pour les systèmes linéaires et gaussiens. Il estime les états d'un système en minimisant l'erreur quadratique moyenne en présence de bruit dans le modèle et les mesures.

#### 3.1 Principe du Filtre de Kalman

Le filtre de Kalman utilise un cycle de prédiction-correction pour estimer les états du système. Il combine les estimations a priori avec les mesures pour produire des estimations a posteriori.

Modèle du système :

$$x_{k+1} = Ax_k + Bu_k + w_k$$
$$y_k = Cx_k + v_k$$

où :

- $w_k$ est le bruit du processus (blanc, gaussien).
- $v_k$ est le bruit de mesure (blanc, gaussien).

**Étapes du Filtre de Kalman :**

1. **Prédiction :**

$$\hat{x}_{k|k-1} = A\hat{x}_{k-1|k-1} + Bu_k$$
$$P_{k|k-1} = AP_{k-1|k-1}A^T + Q$$

2. **Correction :**

$$K_k = P_{k|k-1}C^T(CP_{k|k-1}C^T + R)^{-1}$$
$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k(y_k - C\hat{x}_{k|k-1})$$
$$P_{k|k} = (I - K_kC)P_{k|k-1}$$

### 3.2 Exemple d'Application : Filtrage de Bruit dans un Système de Suivi

**Modèle :**

$$x_{k+1} = \begin{bmatrix} 1 & dt \\ 0 & 1 \end{bmatrix} x_k + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u_k + w_k$$
$$y_k = [1 \quad 0]\, x_k + v_k$$

**Objectif :** Estimer la position et la vitesse d'un véhicule.

**Code MATLAB :**

```
% Paramètres du système

dt = 0.1; % Intervalle de temps
A = [1 dt; 0 1];
B = [0; 1];
C = [1 0];
Q = [0.1 0; 0 0.1]; % Bruit du processus
R = 0.1; % Bruit de mesure
P = eye(2); % Matrice de covariance initiale
x = [0; 0]; % État initial
x_hat = [0; 0]; % Estimation initiale

% Simulation
T = 10;
time = 0:dt:T;
u = 1; % Entrée constante (accélération)
y = zeros(1, length(time)); % Mesure de sortie
x_history = zeros(2, length(time));
x_hat_history = zeros(2, length(time));
```

```
for k = 1:length(time)
    % Système réel
    x = A*x + B*u + mvnrnd([0; 0], Q)';
    y(k) = C*x + mvnrnd(0, R);

    % Prédiction du Filtre de Kalman
    x_pred = A*x_hat + B*u;
    P_pred = A*P*A' + Q;

    % Correction du Filtre de Kalman
    K = P_pred*C'/(C*P_pred*C' + R);
    x_hat = x_pred + K*(y(k) - C*x_pred);
    P = (eye(2) - K*C)*P_pred;

    % Stocker les données
    x_history(:, k) = x;
    x_hat_history(:, k) = x_hat;
end

% Affichage des résultats
figure;
subplot(2,1,1);
plot(time, x_history(1,:), 'r', time, x_hat_history(1,:), 'b');
title('Position : Réel vs Estimé (Filtre de Kalman)');
xlabel('Temps (s)');
ylabel('Position');
legend('Réel', 'Estimé');

subplot(2,1,2);
plot(time, x_history(2,:), 'r', time, x_hat_history(2,:), 'b');
title('Vitesse : Réel vs Estimé (Filtre de Kalman)');
xlabel('Temps (s)');
ylabel('Vitesse');
legend('Réel', 'Estimé');
```

**Conclusion**

L'estimation et l'observation sont essentielles pour contrôler et surveiller les systèmes complexes. L'estimateur de Gopinath fournit une méthode pour les systèmes électriques simples, tandis que l'observateur de Luenberger et le filtre de Kalman sont des outils puissants pour les systèmes linéaires avec des incertitudes. Ces méthodes permettent d'améliorer la performance des systèmes en fournissant des estimations précises des états internes à partir de mesures externes limitées. En combinant la théorie et les exercices pratiques, les concepts présentés dans ce chapitre offrent une base solide pour l'application dans des contextes industriels et de recherche

# Applications: Estimation et Observation Appliquées

## Introduction

L'estimation et l'observation sont des techniques essentielles pour le contrôle et la surveillance de systèmes complexes où les états internes ne peuvent pas être mesurés directement. Ce chapitre se concentre sur l'application des méthodes d'estimation et d'observation à des systèmes réels tels que des machines asynchrones, des réacteurs chimiques, des avions, des missiles, et des bateaux. Nous explorerons trois approches principales :

1. **Estimation des systèmes électriques** (ex. : Estimateur de Gopinath)
2. **Observation déterministe** (Observateur de Luenberger)
3. **Observateurs non-déterministes ou stochastiques** (Filtre de Kalman)

Chaque section comprendra des exercices détaillés pour illustrer l'application pratique de ces méthodes.

## 1. Estimation des Systèmes Électriques : Estimateur de Gopinath

### 1.1 Contexte : Machine Asynchrone

Une machine asynchrone (ou moteur à induction) est couramment utilisée dans les applications industrielles. Pour contrôler efficacement un moteur asynchrone, il est crucial d'estimer les variables internes telles que le flux de rotor et la vitesse angulaire.

**Modèle de la machine asynchrone :**

$$\frac{d\phi_r}{dt} = -\frac{R_r}{L_r}\phi_r + \frac{L_m}{L_r} i_s$$
$$J\frac{d\omega}{dt} = T_e - T_L - B\omega$$

où :

- $\phi_r$ : Flux de rotor
- $\omega$ : Vitesse angulaire
- $i_s$ : Courant statorique
- $R_r, L_r, L_m, J, T_e, T_L, B$ : Paramètres du moteur

**Objectif :** Estimer $\phi_r$ et $\omega$ à partir des mesures de $i_s$.

### 1.2 Exercise 1 : Application de l'Estimateur de Gopinath

**Étape 1 : Modélisation du Système**

Discrétiser le modèle du moteur asynchrone :

$$x[k+1] = Ax[k] + Bu[k]$$

où $x = [\phi_r, \omega]^T$ et $u = i_s$.

**Étape 2 : Définition des Matrices du Système**

```
% Paramètres de la machine asynchrone
Rr = 0.1; % Résistance rotorique
Lr = 0.005; % Inductance rotorique
Lm = 0.1; % Inductance mutuelle
J = 0.01; % Moment d'inertie
B = 0.001; % Frottement visqueux
Te = 1; % Couple électromagnétique
TL = 0.5; % Couple de charge

% Matrices du système
A = [-Rr/Lr, 0; 0, -B/J];
B = [Lm/Lr; 1/J];
C = [1, 0]; % Mesure du flux
D = 0;
```

**Étape 3 : Implémentation de l'Estimateur**

```
% Simulation
dt = 0.01; % Intervalle de temps
T = 5; % Durée de la simulation
time = 0:dt:T;
x = [0; 0]; % États réels
x_est = [0; 0]; % Estimation
i_s = 1; % Courant statorique constant

flux_history = zeros(1, length(time));
omega_history = zeros(1, length(time));

for k = 1:length(time)
    % Système réel
    x = A*x + B*i_s;
    flux_history(k) = x(1);
    omega_history(k) = x(2);

    % Estimation de Gopinath
    x_est = A*x_est + B*i_s + [0; dt*(flux_history(k) -
x_est(1))/Lr];
end

% Affichage des résultats
figure;
subplot(2,1,1);
plot(time, flux_history, 'r', time, omega_history, 'b');
title('Flux et Vitesse de la Machine Asynchrone');
xlabel('Temps (s)');
ylabel('Amplitude');
legend('Flux de Rotor', 'Vitesse Angulaire');

subplot(2,1,2);
plot(time, omega_history, 'r', time, x_est(2,:), 'b');
title('Vitesse : Réel vs Estimé');
xlabel('Temps (s)');
ylabel('Vitesse (rad/s)');
legend('Réel', 'Estimé');
```

## 2. Observation Déterministe : Observateur de Luenberger

.

### 2.1 Contexte : Réacteur Chimique

Les réacteurs chimiques nécessitent des estimations précises de concentrations pour un contrôle optimal. Lorsqu'une concentration n'est pas directement mesurable, un observateur peut être utilisé pour estimer cette variable.

**Modèle du réacteur CSTR (Continuously Stirred Tank Reactor) :**

$$\frac{dC_A}{dt} = -k_1 C_A + u$$
$$\frac{dC_B}{dt} = k_1 C_A - k_2 C_B$$

où :

- $C_A, C_B$ : Concentrations des réactifs
- $k_1, k_2$ : Constantes de réaction
- $u$ : Débit d'entrée

**Objectif :** Estimer $C_A$ et $C_B$ à partir des mesures de $C_A$.

### 2.2 Exercise 2 : Observateur de Luenberger pour le CSTR

**Étape 1 : Modélisation du Système**

Formuler le modèle en espace d'état :

$$x = [C_A, C_B]^T, \quad u = [u], \quad y = C_A$$

**Étape 2 : Définition des Matrices du Système**

### Étape 2 : Définition des Matrices du Système

```
% Paramètres du réacteur CSTR
k1 = 0.1; % Constante de réaction 1
k2 = 0.05; % Constante de réaction 2

A = [-k1, 0; k1, -k2];
B = [1; 0];
C = [1, 0];
D = 0;

% Gain de l'observateur
L = place(A', C', [-2, -3])';
```

### Étape 3 : Implémentation de l'Observateur

```
% Simulation
dt = 0.01;
```

```
T = 10;
time = 0:dt:T;
x = [1; 0]; % États initiaux
x_hat = [0; 0]; % Estimation
u = 1; % Débit constant

conc_A_history = zeros(1, length(time));
conc_B_history = zeros(1, length(time));
conc_A_est_history = zeros(1, length(time));

for k = 1:length(time)
    % Système réel
    x = A*x + B*u;
    y = C*x; % Mesure de sortie

    % Observateur de Luenberger
    x_hat = x_hat + (A*x_hat + B*u + L*(y - C*x_hat))*dt;

    % Stocker les données
    conc_A_history(k) = x(1);
    conc_B_history(k) = x(2);
    conc_A_est_history(k) = x_hat(1);
end

% Affichage des résultats
figure;
subplot(2,1,1);
plot(time, conc_A_history, 'r', time, conc_B_history, 'b');
title('Concentrations : C_A et C_B');
xlabel('Temps (s)');
ylabel('Concentration (mol/L)');
legend('C_A', 'C_B');

subplot(2,1,2);
plot(time, conc_A_history, 'r', time, conc_A_est_history, 'b');
title('Concentration C_A : Réel vs Estimé');
xlabel('Temps (s)');
ylabel('Concentration (mol/L)');
legend('Réel', 'Estimé');
```

### 3. Observateurs Non-déterministes : Filtre de Kalman

#### 3.1 Contexte : Suivi d'un Avion

Dans le suivi et le contrôle d'avions, le filtre de Kalman est utilisé pour estimer la position et la vitesse à partir de mesures bruitées.

**Modèle du Système :**

$$\dot{x} = Ax + Bu + w$$
$$y = Cx + v$$

où :

- $w$ : Bruit de processus
- $v$ : Bruit de mesure

**Objectif :** Estimer la position et la vitesse d'un avion à partir de mesures de position bruitées.

### 3.2 Exercise 3 : Filtre de Kalman pour le Suivi d'Avion

#### Étape 1 : Modélisation du Système

Définir un modèle discret du système pour le suivi :

$$x = [\text{position}, \text{vitesse}]^T$$

#### Étape 2 : Définition des Matrices du Système

```
% Paramètres du système
dt = 0.1; % Intervalle de temps
A = [1 dt; 0 1];
B = [0; 0];
C = [1 0];
Q = [0.1 0; 0 0.1]; % Bruit de processus
R = 0.1; % Bruit de mesure
P = eye(2); % Matrice de covariance initiale
x = [0; 0]; % État initial
x_hat = [0; 0]; % Estimation initiale

% Simulation
T = 10;
time = 0:dt:T;
y = zeros(1, length(time)); % Mesure de sortie
x_history = zeros(2, length(time));
x_hat_history = zeros(2, length(time));

for k = 1:length(time)
    % Système réel
    x = A*x + B*u + mvnrnd([0; 0], Q)';
    y(k) = C*x + mvnrnd(0, R);

    % Prédiction du Filtre de Kalman
    x_pred = A*x_hat + B*u;
    P_pred = A*P*A' + Q;

    % Correction du Filtre de Kalman
    K = P_pred*C'/(C*P_pred*C' + R);
    x_hat = x_pred + K*(y(k) - C*x_pred);
    P = (eye(2) - K*C)*P_pred;

    % Stocker les données
    x_history(:, k) = x;
    x_hat_history(:, k) = x_hat;
end
```

```
% Affichage des résultats
figure;
subplot(2,1,1);
plot(time, x_history(1,:), 'r', time, x_hat_history(1,:), 'b');
title('Position : Réel vs Estimé (Filtre de Kalman)');
xlabel('Temps (s)');
ylabel('Position');
legend('Réel', 'Estimé');

subplot(2,1,2);
plot(time, x_history(2,:), 'r', time, x_hat_history(2,:), 'b');
title('Vitesse : Réel vs Estimé (Filtre de Kalman)');
xlabel('Temps (s)');
ylabel('Vitesse');
legend('Réel', 'Estimé');
```

**Conclusion**

Les méthodes d'estimation et d'observation sont essentielles pour le contrôle efficace des systèmes complexes. En utilisant des estimateurs déterministes comme l'observateur de Luenberger ou des techniques stochastiques comme le filtre de Kalman, il est possible de déduire les états internes d'un système à partir de mesures externes. Les exercices proposés montrent comment ces techniques peuvent être appliquées à divers systèmes réels tels que des machines asynchrones, des réacteurs chimiques, et des véhicules (avions, missiles, bateaux). Ces outils sont cruciaux pour la conception de systèmes de contrôle robustes et fiables dans des applications industrielles et de recherche.

# Applications :

# Estimation et Observation pour les Systèmes Multivariables

**Introduction**

L'estimation et l'observation jouent un rôle crucial dans le contrôle et la surveillance des systèmes complexes et multivariables. Ces techniques permettent de déduire les états internes des systèmes lorsque toutes les variables ne peuvent pas être mesurées directement. Ce chapitre se concentre sur trois approches principales d'estimation et d'observation, appliquées à des systèmes multivariables spécifiques tels que des réacteurs chimiques, des missiles, et des bateaux.

1. **Estimation des systèmes électriques** (ex. : Estimateur de Gopinath)
2. **Observation déterministe** (Observateur de Luenberger)
3. **Observateurs non-déterministes ou stochastiques** (Filtre de Kalman)

Chaque section inclut des exercices détaillés pour illustrer l'application pratique de ces méthodes.

**1. Estimation des Systèmes Électriques : Estimateur de Gopinath**

L'estimateur de Gopinath est utilisé pour estimer les états internes d'un système électrique à partir des mesures de sortie. Cette méthode est utile pour des systèmes tels que des moteurs et des générateurs.

**1.1 Contexte : Réacteur Chimique**

Dans un réacteur chimique, les concentrations des réactifs sont des variables cruciales qui influencent les réactions. Souvent, toutes les concentrations ne peuvent pas être mesurées en temps réel. L'estimation permet de déterminer ces concentrations à partir de variables mesurables.

Modèle du réacteur CSTR (Continuously Stirred Tank Reactor) :

$$\frac{dC_A}{dt} = -k_1 C_A + u$$
$$\frac{dC_B}{dt} = k_1 C_A - k_2 C_B$$

où :

- $C_A$, $C_B$ : Concentrations des réactifs A et B
- $k_1$, $k_2$ : Constantes de réaction
- $u$ : Débit d'entrée

Objectif : Estimer $C_B$ à partir de la mesure de $C_A$.

1.2 Exercice 1 : Estimation avec l'Estimateur de Gopinath pour le CSTR

Étape 1 : Modélisation du Système

Formuler le modèle en espace d'état :

$$x = [C_A, C_B]^T, \quad u = [u], \quad y = C_A$$

Étape 2 : Définition des Matrices du Système

```
% Paramètres du réacteur CSTR
k1 = 0.1; % Constante de réaction 1
k2 = 0.05; % Constante de réaction 2

A = [-k1, 0; k1, -k2];
B = [1; 0];
C = [1, 0];
D = 0;
```

*Étape 3 : Implémentation de l'Estimateur*

```
matlab
Copier le code
% Simulation
dt = 0.01;
T = 10;
time = 0:dt:T;
x = [1; 0]; % États initiaux
x_est = [0; 0]; % Estimation
u = 1; % Débit constant

conc_A_history = zeros(1, length(time));
conc_B_history = zeros(1, length(time));
conc_B_est_history = zeros(1, length(time));

for k = 1:length(time)
    % Système réel
    x = A*x + B*u;
    y = C*x; % Mesure de sortie

    % Estimateur de Gopinath
    x_est = A*x_est + B*u + [0; (conc_A_history(k) - x_est(1))/k1];

    % Stocker les données
    conc_A_history(k) = x(1);
    conc_B_history(k) = x(2);
    conc_B_est_history(k) = x_est(2);
end

% Affichage des résultats
figure;
subplot(2,1,1);
plot(time, conc_A_history, 'r', time, conc_B_history, 'b');
title('Concentrations : C_A et C_B');
xlabel('Temps (s)');
ylabel('Concentration (mol/L)');
legend('C_A', 'C_B');

subplot(2,1,2);
plot(time, conc_B_history, 'r', time, conc_B_est_history, 'b');
title('Concentration C_B : Réel vs Estimé');
```

```
xlabel('Temps (s)');
ylabel('Concentration (mol/L)');
legend('Réel', 'Estimé');
```

## 2. Observation Déterministe : Observateur de Luenberger

L'observateur de Luenberger est une méthode déterministe utilisée pour estimer les états internes d'un système linéaire à partir des mesures de sortie. Cette méthode est particulièrement utile pour les systèmes avec des états multiples et une dynamique complexe.

### 2.1 Contexte : Missile

Pour un missile, l'estimation précise de la position, de la vitesse, et de l'angle de vol est cruciale pour la navigation et le guidage. Ces paramètres ne peuvent pas toujours être mesurés directement en raison des contraintes opérationnelles.

Modèle d'État du Missile :

$$\dot{x} = Ax + Bu$$
$$y = Cx$$

où :

- $x$ : Vecteur d'état (position, vitesse, angle)
- $u$ : Entrée (force, couple)
- $y$ : Sortie (position mesurée)

Objectif : Estimer la position et la vitesse à partir des mesures d'angle de vol.

2.2 Exercice 2 : Observateur de Luenberger pour le Missile

Étape 1 : Modélisation du Système

Formuler le modèle en espace d'état :

$$x = [\text{position}, \text{vitesse}, \text{angle}]^T, \quad u = [\text{force}], \quad y = \text{angle}$$

### Étape 2 : Définition des Matrices du Système

```
% Paramètres du missile
A = [0 1 0; 0 0 1; 0 -0.1 -0.2];
B = [0; 1; 0];
C = [0 0 1];
D = 0;

% Gain de l'observateur
L = place(A', C', [-1, -2, -3])';
```

### Étape 3 : Implémentation de l'Observateur

```
% Simulation
dt = 0.01;
T = 10;
time = 0:dt:T;
x = [0; 0; 0]; % États initiaux
```

```
x_hat = [0; 0; 0]; % Estimation
u = 1; % Force constante

pos_history = zeros(1, length(time));
vel_history = zeros(1, length(time));
angle_history = zeros(1, length(time));
angle_est_history = zeros(1, length(time));

for k = 1:length(time)
    % Système réel
    x = A*x + B*u;
    y = C*x; % Mesure de sortie

    % Observateur de Luenberger
    x_hat = x_hat + (A*x_hat + B*u + L*(y - C*x_hat))*dt;

    % Stocker les données
    pos_history(k) = x(1);
    vel_history(k) = x(2);
    angle_history(k) = x(3);
    angle_est_history(k) = x_hat(3);
end

% Affichage des résultats
figure;
subplot(2,1,1);
plot(time, pos_history, 'r', time, vel_history, 'b');
title('Position et Vitesse du Missile');
xlabel('Temps (s)');
ylabel('Amplitude');
legend('Position', 'Vitesse');

subplot(2,1,2);
plot(time, angle_history, 'r', time, angle_est_history, 'b');
title('Angle de Vol : Réel vs Estimé');
xlabel('Temps (s)');
ylabel('Angle (rad)');
legend('Réel', 'Estimé');
```

### 3. Observateurs Non-déterministes : Filtre de Kalman

Le filtre de Kalman est un outil puissant pour estimer les états d'un système en présence de bruit de mesure et de processus. Il est particulièrement utile pour les systèmes où les incertitudes sont importantes, comme les bateaux naviguant en mer.

#### 3.1 Contexte : Bateau

Dans la navigation de bateaux, l'estimation de la position et de la vitesse est essentielle pour le contrôle et la navigation. Les incertitudes dues aux conditions météorologiques et au bruit des capteurs nécessitent des techniques d'estimation robustes.

**Modèle du Système :**

$$\dot{x} = Ax + Bu + w$$
$$y = Cx + v$$

**Objectif :** Estimer la position et la vitesse du bateau à partir de mesures de position bruitées.

### 3.2 Exercice 3 : Filtre de Kalman pour le Bateau

#### Étape 1 : Modélisation du Système

Définir un modèle discret du système pour le suivi :

$$x = [\text{position}, \text{vitesse}]^T$$

#### Étape 2 : Définition des Matrices du Système

```
% Paramètres du système
dt = 0.1; % Intervalle de temps
A = [1 dt; 0 1];
B = [0; 0];
C = [1 0];
Q = [0.1 0; 0 0.1]; % Bruit de processus
R = 0.1; % Bruit de mesure
P = eye(2); % Matrice de covariance initiale
x = [0; 0]; % État initial
x_hat = [0; 0]; % Estimation initiale

% Simulation
T = 10;
time = 0:dt:T;
y = zeros(1, length(time)); % Mesure de sortie
x_history = zeros(2, length(time));
x_hat_history = zeros(2, length(time));

for k = 1:length(time)
    % Système réel
    x = A*x + B*u + mvnrnd([0; 0], Q)';
    y(k) = C*x + mvnrnd(0, R);

    % Prédiction du Filtre de Kalman
    x_pred = A*x_hat + B*u;
    P_pred = A*P*A' + Q;

    % Correction du Filtre de Kalman
    K = P_pred*C'/(C*P_pred*C' + R);
    x_hat = x_pred + K*(y(k) - C*x_pred);
    P = (eye(2) - K*C)*P_pred;

    % Stocker les données
    x_history(:, k) = x;
```

```
    x_hat_history(:, k) = x_hat;
end

% Affichage des résultats
figure;
subplot(2,1,1);
plot(time, x_history(1,:), 'r', time, x_hat_history(1,:), 'b');
title('Position : Réel vs Estimé (Filtre de Kalman)');
xlabel('Temps (s)');
ylabel('Position');
legend('Réel', 'Estimé');

subplot(2,1,2);
plot(time, x_history(2,:), 'r', time, x_hat_history(2,:), 'b');
title('Vitesse : Réel vs Estimé (Filtre de Kalman)');
xlabel('Temps (s)');
ylabel('Vitesse');
legend('Réel', 'Estimé');
```

**Conclusion**

Les méthodes d'estimation et d'observation sont des outils puissants pour gérer des systèmes complexes et multivariables où toutes les variables ne sont pas directement mesurables. Les exercices proposés montrent comment ces techniques peuvent être appliquées à des réacteurs chimiques, des missiles, et des bateaux, offrant une base pour le développement de systèmes de contrôle robustes et efficaces. Ces outils sont essentiels pour améliorer la précision, la stabilité, et la performance dans diverses applications industrielles et de recherche

# Projets : Estimation et l'Observation appliquées aux Systèmes Multivariables

## Introduction

Les systèmes multivariables, tels que les réacteurs chimiques, les missiles et les bateaux, présentent des dynamiques complexes nécessitant des techniques avancées pour estimer et observer leurs états internes. Dans ce document, nous explorons trois approches principales d'estimation et d'observation appliquées à ces systèmes. Nous commencerons par les systèmes électriques linéarisés avec l'estimateur de Gopinath, suivis de l'observation déterministe avec l'observateur de Luenberger, et enfin, les observateurs stochastiques avec le filtre de Kalman.

## 1. Estimation des Systèmes Électriques : Estimateur de Gopinath

L'estimateur de Gopinath est utilisé pour estimer les états internes des systèmes linéarisés. Cette méthode est particulièrement efficace pour les systèmes où les modèles linéaires approchent de manière adéquate la dynamique réelle.

### 1.1 Exercice 1 : Estimateur de Gopinath pour un Réacteur Chimique Linéarisé

#### Contexte

Dans un réacteur chimique, les variables critiques, telles que les concentrations des réactifs et la température, doivent être surveillées de près. Les réactions chimiques non linéaires peuvent être linéarisées autour d'un point de fonctionnement pour faciliter l'estimation.

Modèle Linéarisé

Modèle original :

$$\frac{dC_A}{dt} = -k_1 C_A + u$$
$$\frac{dC_B}{dt} = k_1 C_A - k_2 C_B$$

Modèle linéarisé autour du point de fonctionnement $(C_{A0}, C_{B0})$ :

$$\frac{dx_1}{dt} = -k_1 x_1 + u$$
$$\frac{dx_2}{dt} = k_1 x_1 - k_2 x_2$$

où $x_1 = C_A - C_{A0}$ et $x_2 = C_B - C_{B0}$.

Implémentation en Matlab

Étape 1 : Modélisation du Système

```
% Paramètres du réacteur

k1 = 0.1;
k2 = 0.05;
C_A0 = 1; % Concentration de A à l'équilibre
```

```
C_B0 = 0.5; % Concentration de B à l'équilibre

% Matrices du système linéarisé
A = [-k1 0; k1 -k2];
B = [1; 0];
C = [1 0]; % On mesure C_A
D = 0;

% État initial
x0 = [0; 0];
```

*Étape 2 : Simulation de l'Estimateur de Gopinath*

```
matlab
Copier le code
% Simulation
dt = 0.01;
T = 10;
time = 0:dt:T;
x = x0;
x_hat = [0; 0]; % Estimation initiale

u = ones(size(time)); % Entrée constante

conc_A_history = zeros(1, length(time));
conc_B_est_history = zeros(1, length(time));

for k = 1:length(time)
    % Système réel
    x = A*x + B*u(k);
    y = C*x; % Mesure de sortie

    % Estimateur de Gopinath
    x_hat = A*x_hat + B*u(k) + [0; (y - x_hat(1))/k1];

    % Stocker les données
    conc_A_history(k) = x(1);
    conc_B_est_history(k) = x_hat(2);
end

% Affichage des résultats
figure;
plot(time, conc_A_history, 'r', time, conc_B_est_history, 'b');
title('Estimation de C_B : Réel vs Estimé (Estimateur de
Gopinath)');
xlabel('Temps (s)');
ylabel('Concentration (mol/L)');
legend('Réel', 'Estimé');
```

### 1.2 Exercice 2 : Estimateur de Gopinath pour un Système de Navigation de Bateau

**Contexte**

Dans la navigation de bateau, la vitesse et la position doivent être estimées pour maintenir le cap. Les dynamiques du bateau sont linéarisées autour d'un point de fonctionnement.

Modèle Linéarisé

Modèle original :

$$\dot{x} = v\cos(\theta)$$
$$\dot{y} = v\sin(\theta)$$
$$\dot{\theta} = \omega$$

Modèle linéarisé autour de $(x_0, y_0, \theta_0)$ :

$$\dot{x} = v_0\cos(\theta_0) + \Delta v\cos(\theta_0) - v_0\sin(\theta_0)\Delta\theta$$
$$\dot{y} = v_0\sin(\theta_0) + \Delta v\sin(\theta_0) + v_0\cos(\theta_0)\Delta\theta$$
$$\dot{\theta} = \Delta\omega$$

Implémentation en Matlab

Étape 1 : Modélisation du Système

```
% Paramètres du bateau
v0 = 1; % Vitesse constante
theta0 = pi/4; % Angle constant

% Matrices du système linéarisé
A = [0 0 -v0*sin(theta0); 0 0 v0*cos(theta0); 0 0 0];
B = [cos(theta0); sin(theta0); 1];
C = eye(3);
D = 0;

% État initial
x0 = [0; 0; 0];
```

*Étape 2 : Simulation de l'Estimateur de Gopinath*

```
matlab
Copier le code
% Simulation
dt = 0.1;
T = 10;
time = 0:dt:T;
x = x0;
x_hat = [0; 0; 0]; % Estimation initiale

u = ones(size(time)); % Vitesse constante

x_history = zeros(3, length(time));
x_est_history = zeros(3, length(time));

for k = 1:length(time)
    % Système réel
    x = A*x + B*u(k);
    y = C*x; % Mesure de sortie

    % Estimateur de Gopinath
    x_hat = A*x_hat + B*u(k) + [0; 0; (y(3) - x_hat(3))/v0];
```

```
    % Stocker les données
    x_history(:, k) = x;
    x_est_history(:, k) = x_hat;
end

% Affichage des résultats
figure;
subplot(3,1,1);
plot(time, x_history(1,:), 'r', time, x_est_history(1,:), 'b');
title('Position X : Réel vs Estimé (Estimateur de Gopinath)');
xlabel('Temps (s)');
ylabel('Position X');
legend('Réel', 'Estimé');

subplot(3,1,2);
plot(time, x_history(2,:), 'r', time, x_est_history(2,:), 'b');
title('Position Y : Réel vs Estimé (Estimateur de Gopinath)');
xlabel('Temps (s)');
ylabel('Position Y');
legend('Réel', 'Estimé');

subplot(3,1,3);
plot(time, x_history(3,:), 'r', time, x_est_history(3,:), 'b');
title('Angle Theta : Réel vs Estimé (Estimateur de Gopinath)');
xlabel('Temps (s)');
ylabel('Angle Theta');
legend('Réel', 'Estimé');
```

**2. Observation Déterministe : Observateur de Luenberger**

L'observateur de Luenberger est utilisé pour estimer les états des systèmes linéarisés en utilisant un modèle mathématique déterministe. Cette méthode est particulièrement utile dans les cas où les systèmes présentent des dynamiques rapides et les états ne sont pas directement mesurables.

**2.1 Exercice 3 : Observateur de Luenberger pour un Système de Guidage de Missile**

**Contexte**

Pour un missile, il est crucial de connaître la position, la vitesse, et l'angle de vol en temps réel pour un guidage précis. Les systèmes de missile sont souvent linéarisés autour d'une trajectoire de vol nominale pour simplifier l'estimation.

**Modèle Linéarisé**

**Modèle original :**

$$\dot{x} = v\cos(\theta)$$
$$\dot{y} = v\sin(\theta)$$
$$\dot{\theta} = \omega$$

Modèle linéarisé autour de $(x_0, y_0, \theta_0)$ :

$$\dot{x} = v_0 \cos(\theta_0) + \Delta v \cos(\theta_0) - v_0 \sin(\theta_0)\Delta\theta$$
$$\dot{y} = v_0 \sin(\theta_0) + \Delta v \sin(\theta_0) + v_0 \cos(\theta_0)\Delta\theta$$
$$\dot{\theta} = \Delta\omega$$

Implémentation en Matlab

Étape 1 : Modélisation du Système

```
% Paramètres du missile
v0 = 300; % Vitesse constante
theta0 = pi/6; % Angle constant

% Matrices du système linéarisé
A = [0 0 -v0*sin(theta0); 0 0 v0*cos(theta0); 0 0 0];
B = [cos(theta0); sin(theta0); 0];
C = eye(3);
D = 0;

% Gain de l'observateur
L = place(A', C', [-1, -2, -3])';
```

**Étape 2 : Simulation de l'Observateur de Luenberger**

```
% Simulation
dt = 0.1;
T = 20;
time = 0:dt:T;
x = [0; 0; 0]; % États initiaux
x_hat = [0; 0; 0]; % Estimation
u = 1; % Commande constante

x_history = zeros(3, length(time));
x_est_history = zeros(3, length(time));

for k = 1:length(time)
    % Système réel
    x = A*x + B*u;
    y = C*x; % Mesure de sortie

    % Observateur de Luenberger
    x_hat = x_hat + (A*x_hat + B*u + L*(y - C*x_hat))*dt;

    % Stocker les données
    x_history(:, k) = x;
    x_est_history(:, k) = x_hat;
end

% Affichage des résultats
figure;
subplot(3,1,1);
plot(time, x_history(1,:), 'r', time, x_est_history(1,:), 'b');
title('Position X : Réel vs Estimé (Observateur de Luenberger)');
```

```
xlabel('Temps (s)');
ylabel('Position X');
legend('Réel', 'Estimé');

subplot(3,1,2);
plot(time, x_history(2,:), 'r', time, x_est_history(2,:), 'b');
title('Position Y : Réel vs Estimé (Observateur de Luenberger)');
xlabel('Temps (s)');
ylabel('Position Y');
legend('Réel', 'Estimé');

subplot(3,1,3);
plot(time, x_history(3,:), 'r', time, x_est_history(3,:), 'b');
title('Angle Theta : Réel vs Estimé (Observateur de Luenberger)');
xlabel('Temps (s)');
ylabel('Angle Theta');
legend('Réel', 'Estimé');
```

### 2.2 Exercice 4 : Observateur de Luenberger pour un Bateau en Navigation

**Contexte**

Dans la navigation de bateaux, il est essentiel de connaître les états internes tels que la position et la vitesse pour un contrôle efficace et sûr. Un observateur de Luenberger peut être utilisé pour estimer ces états à partir des mesures de position bruitées.

Modèle Linéarisé

Modèle original :

$$\dot{x} = v\cos(\theta)$$
$$\dot{y} = v\sin(\theta)$$
$$\dot{\theta} = \omega$$

Modèle linéarisé autour de $(x_0, y_0, \theta_0)$ :

$$\dot{x} = v_0\cos(\theta_0) + \Delta v\cos(\theta_0) - v_0\sin(\theta_0)\Delta\theta$$
$$\dot{y} = v_0\sin(\theta_0) + \Delta v\sin(\theta_0) + v_0\cos(\theta_0)\Delta\theta$$
$$\dot{\theta} = \Delta\omega$$

Implémentation en Matlab

Étape 1 : Modélisation du Système

```
% Paramètres du bateau
v0 = 10; % Vitesse constante
theta0 = pi/4; % Angle constant

% Matrices du système linéarisé
A = [0 0 -v0*sin(theta0); 0 0 v0*cos(theta0); 0 0 0];
B = [cos(theta0); sin(theta0); 0];
C = eye(3);
D = 0;
```

```
% Gain de l'observateur
L = place(A', C', [-1, -1.5, -2])';
```

**Étape 2 : Simulation de l'Observateur de Luenberger**

```
% Simulation
dt = 0.1;
T = 20;
time = 0:dt:T;
x = [0; 0; 0]; % États initiaux
x_hat = [0; 0; 0]; % Estimation
u = 1; % Commande constante

x_history = zeros(3, length(time));
x_est_history = zeros(3, length(time));

for k = 1:length(time)
    % Système réel
    x = A*x + B*u;
    y = C*x; % Mesure de sortie

    % Observateur de Luenberger
    x_hat = x_hat + (A*x_hat + B*u + L*(y - C*x_hat))*dt;

    % Stocker les données
    x_history(:, k) = x;
    x_est_history(:, k) = x_hat;
end

% Affichage des résultats
figure;
subplot(3,1,1);
plot(time, x_history(1,:), 'r', time, x_est_history(1,:), 'b');
title('Position X : Réel vs Estimé (Observateur de Luenberger)');
xlabel('Temps (s)');
ylabel('Position X');
legend('Réel', 'Estimé');

subplot(3,1,2);
plot(time, x_history(2,:), 'r', time, x_est_history(2,:), 'b');
title('Position Y : Réel vs Estimé (Observateur de Luenberger)');
xlabel('Temps (s)');
ylabel('Position Y');
legend('Réel', 'Estimé');

subplot(3,1,3);
plot(time, x_history(3,:), 'r', time, x_est_history(3,:), 'b');
title('Angle Theta : Réel vs Estimé (Observateur de Luenberger)');
xlabel('Temps (s)');
ylabel('Angle Theta');
legend('Réel', 'Estimé');
```

## 3. Observateurs Non-déterministes : Filtre de Kalman

Le filtre de Kalman est une technique d'estimation stochastique permettant de filtrer le bruit et d'estimer les états d'un système en temps réel. Il est particulièrement utile pour les systèmes où les incertitudes sont élevées, comme les missiles et les bateaux en mouvement.

### 3.1 Exercice 5 : Filtre de Kalman pour un Réacteur Chimique

**Contexte**

Dans un réacteur chimique, les conditions d'opération peuvent fluctuer en raison de variations dans les débits d'entrée ou de bruit de mesure. Le filtre de Kalman est utilisé pour estimer les états internes du réacteur en présence de bruit.

**Modèle Linéarisé**

**Modèle original :**

$$\frac{dC_A}{dt} = -k_1 C_A + u$$
$$\frac{dC_B}{dt} = k_1 C_A - k_2 C_B$$

**Modèle linéarisé avec bruit :**

$$\frac{dx_1}{dt} = -k_1 x_1 + u + w_1$$
$$\frac{dx_2}{dt} = k_1 x_1 - k_2 x_2 + w_2$$

où $w_1$ et $w_2$ représentent le bruit de processus.

**Implémentation en Matlab**

**Étape 1 : Modélisation du Système**

```
k1 = 0.1;
k2 = 0.05;
C_A0 = 1; % Concentration de A à l'équilibre
C_B0 = 0.5; % Concentration de B à l'équilibre

% Matrices du système linéarisé
A = [-k1 0; k1 -k2];
B = [1; 0];
C = [1 0]; % On mesure C_A
D = 0;

% Paramètres du filtre de Kalman
Q = [0.01 0; 0 0.01]; % Bruit de processus
R = 0.1; % Bruit de mesure
P = eye(2); % Matrice de covariance initiale
x = [0; 0]; % État initial
x_hat = [0; 0]; % Estimation initiale
```

**Étape 2 : Simulation du Filtre de Kalman**

```
% Simulation
dt = 0.01;
T = 10;
time = 0:dt:T;
x = x0;
x_hat = [0; 0]; % Estimation initiale

u = ones(size(time)); % Entrée constante

conc_A_history = zeros(1, length(time));
conc_B_est_history = zeros(1, length(time));

for k = 1:length(time)
    % Système réel
    x = A*x + B*u(k) + mvnrnd([0; 0], Q)';
    y = C*x + mvnrnd(0, R);

    % Prédiction du Filtre de Kalman
    x_pred = A*x_hat + B*u(k);
    P_pred = A*P*A' + Q;

    % Correction du Filtre de Kalman
    K = P_pred*C'/(C*P_pred*C' + R);
    x_hat = x_pred + K*(y - C*x_pred);
    P = (eye(2) - K*C)*P_pred;

    % Stocker les données
    conc_A_history(k) = x(1);
    conc_B_est_history(k) = x_hat(2);
end

% Affichage des résultats
figure;
plot(time, conc_A_history, 'r', time, conc_B_est_history, 'b');
title('Estimation de C_B : Réel vs Estimé (Filtre de Kalman)');
xlabel('Temps (s)');
ylabel('Concentration (mol/L)');
legend('Réel', 'Estimé');
```

### 3.2 Exercice 6 : Filtre de Kalman pour un Bateau en Navigation

**Contexte**

Le suivi précis de la position et de la vitesse d'un bateau est crucial pour la navigation. Le filtre de Kalman permet de filtrer les bruits de mesure et d'estimer les états internes en temps réel.

Modèle Linéarisé

Modèle original :

$$\dot{x} = v\cos(\theta)$$
$$\dot{y} = v\sin(\theta)$$
$$\dot{\theta} = \omega$$

Modèle linéarisé avec bruit :

$$\dot{x} = v_0\cos(\theta_0) + \Delta v\cos(\theta_0) - v_0\sin(\theta_0)\Delta\theta + w_1$$
$$\dot{y} = v_0\sin(\theta_0) + \Delta v\sin(\theta_0) + v_0\cos(\theta_0)\Delta\theta + w_2$$
$$\dot{\theta} = \Delta\omega + w_3$$

où $w_1$, $w_2$, et $w_3$ représentent le bruit de processus.

**Implémentation en Matlab**

**Étape 1 : Modélisation du Système**

```
% Paramètres du bateau
v0 = 10; % Vitesse constante
theta0 = pi/4; % Angle constant

% Matrices du système linéarisé
A = [0 0 -v0*sin(theta0); 0 0 v0*cos(theta0); 0 0 0];
B = [cos(theta0); sin(theta0); 0];
C = eye(3);
D = 0;

% Paramètres du filtre de Kalman
Q = [0.1 0 0; 0 0.1 0; 0 0 0.1]; % Bruit de processus
R = [0.05 0 0; 0 0.05 0; 0 0 0.05]; % Bruit de mesure
P = eye(3); % Matrice de covariance initiale
x = [0; 0; 0]; % État initial
x_hat = [0; 0; 0]; % Estimation initiale
```

**Étape 2 : Simulation du Filtre de Kalman**

```
% Simulation
dt = 0.1;
T = 20;
time = 0:dt:T;
x = [0; 0; 0]; % États initiaux
x_hat = [0; 0; 0]; % Estimation
u = 1; % Commande constante

x_history = zeros(3, length(time));
x_est_history = zeros(3, length(time));

for k = 1:length(time)
    % Système réel
    x = A*x + B*u + mvnrnd([0; 0; 0], Q)';
```

```
    y = C*x + mvnrnd([0; 0; 0], R);

    % Prédiction du Filtre de Kalman
    x_pred = A*x_hat + B*u;
    P_pred = A*P*A' + Q;

    % Correction du Filtre de Kalman
    K = P_pred*C'/(C*P_pred*C' + R);
    x_hat = x_pred + K*(y - C*x_pred);
    P = (eye(3) - K*C)*P_pred;

    % Stocker les données
    x_history(:, k) = x;
    x_est_history(:, k) = x_hat;
end

% Affichage des résultats
figure;
subplot(3,1,1);
plot(time, x_history(1,:), 'r', time, x_est_history(1,:), 'b');
title('Position X : Réel vs Estimé (Filtre de Kalman)');
xlabel('Temps (s)');
ylabel('Position X');
legend('Réel', 'Estimé');

subplot(3,1,2);
plot(time, x_history(2,:), 'r', time, x_est_history(2,:), 'b');
title('Position Y : Réel vs Estimé (Filtre de Kalman)');
xlabel('Temps (s)');
ylabel('Position Y');
legend('Réel', 'Estimé');

subplot(3,1,3);
plot(time, x_history(3,:), 'r', time, x_est_history(3,:), 'b');
title('Angle Theta : Réel vs Estimé (Filtre de Kalman)');
xlabel('Temps (s)');
ylabel('Angle Theta');
legend('Réel', 'Estimé');
```

**Conclusion**

Les méthodes d'estimation et d'observation, telles que l'estimateur de Gopinath, l'observateur de Luenberger, et le filtre de Kalman, sont des outils puissants pour le suivi des états des systèmes multivariables complexes. Ces techniques permettent de filtrer le bruit, d'estimer les états internes, et d'améliorer la précision du contrôle dans des applications industrielles et de recherche. Les exercices proposés montrent comment ces méthodes peuvent être appliquées à des réacteurs chimiques, des missiles, et des bateaux, offrant une base solide pour le développement de systèmes de contrôle robustes et efficaces.

**Références bibliographiques:**


1. I.D. Landau, "Identification des systèmes", Hermès, 1998.
2. E. Duflos, Ph. Vanheeghe, "Estimation Prédiction", Technip, 2000.
3. T. Soderstrom, P. Stoica, "System Identification", Prentice Hall, 1989.
4. R. Hanus, "Identification à l'automatique", DE Boeck, 2001.
5. L. Lennart, "System Identification: Theory for the User", Second edition, Prentice Hall 1999.
6. P. Borne, Geneviève Dauphin-Tanguy, Jean-Pierre Richard, "Modélisation et identification des processus", Technip, 1992.
7. R. Ben Abdenour, P. Borne, M. Ksouri, M. Sahli, "Identification et commande numérique des procédés industriels", Technip, 2001.
8. E. Walter, L. Pronzato, "Identification of Parametric Models from Experimental Data", Springer, **1997**.