Hybrid dynamical system Identification: geometry, sparsity, and nonlinearities

van Luong Le

Ecole doctorale IAEM Lorraine
Département de formation doctorale en automatique
UFR ESSTIN

Identification de Systèmes Dynamiques Hybrides : géométrie, parcimonie, et non-linéarités

—

Hybrid Dynamical System Identification: geometry, sparsity, and nonlinearities

THÈSE

présentée et soutenue publiquement le 04/10/2013
pour l'obtention du

Doctorat de l'Université de Lorraine
spécialité automatique

par

Van Luong Le

Centre de Recherche en Automatique de Nancy, UMR CNRS 7039, Université de Lorraine

Remerciements

Tout au long de ces trois années de thèse, de nombreuses personnes m'ont aidé d'une manière ou d'une autre. Même si je ne peux les citer toutes, je voudrais les remercier.

Je tiens à remercier plus particulièrement M. Gérard Bloch, mon directeur de thèse, et M. Fabien Lauer, mon encadrant de thèse, de m'avoir accepté en thèse. Je les remercie pour leur direction, leur compétences, leur sympathie, leurs encouragements et leur assistance constante durant toutes ces trois années. Sincèrement, grâce à eux j'ai pu apprendre plusieurs choses importantes pour mon développement personnel. Surtout, sans eux, je n'aurais pu arriver jusqu'au bout de ma thèse. J'ai pris un grand plaisir à travailler avec eux.

Je voudrais remercier les rapporteurs de cette thèse, M. Stéphane Lecœuche, Professeur à l'Ecole des Mines de Douai, et M. Alain Rakotomamonjy, Professeur à l'Université de Rouen, pour l'intérêt qu'ils ont porté à mon travail. Je remercie également les autres membres de mon jury : M. Laurent Bako, Maître de Conférences à l'Ecole Centrale de Lyon, M. Gilles Millérioux, Professeur à l'Université de Lorraine, et Mr. Henrik Ohlsson, Assistant Professor à l'Université de Linköping en Suède, d'avoir accepté de participer à mon jury en qualité d'examinateurs.

Je suis très reconnaissant à Mme Carole Courrier, Mme Karine Jacquot, M. Georges Billant, Mme Soraya Khelifa et tous les personnels techniques et administratifs du CRAN et de l'Université de Lorraine qui m'ont facilité la vie dans les démarches doctorales.

J'adresse de sincères remerciements à M. Hugues Garnier, M. David Brie, M. Brice Vincent, M. Hervé Ortega, M. Sylvain Kubler et tous les personnels du département R&T de l'IUT Nancy-Brabois qui m'ont aidé à effectuer mes services d'enseignement pendant ces trois années.

Je remercie sincèrement les personnels du groupe CID du CRAN et du groupe ABC du LORIA de m'avoir adopté pendant trois ans. Plus particulièrement, je voudrais remercier M. Yann Guermeur, Mlle Fabienne Thomarat et Mlle Mariem Halimi pour des encouragements et des conversations joyeuses.

Merci à mes amis vietnamiens en France qui ont partagé avec moi des souvenirs inoubliables et m'ont soutenu pendant ma soutenance de thèse.

Je consacre mes dernières pensées à mes familles au Vietnam et en France, et surtout mes parents, pour leur soutien au cours de ces trois années sans lequel je n'en serais pas là aujourd'hui.

A mes parents

Résumé et contributions

Contexte

En automatique, l'obtention d'un modèle du système est le cœur et la première étape des procédures comme la synthèse d'une commande, la détection des défaillances, la prédiction... L'approche classique utilise les connaissances physiques pour décrire le comportement du système. Cette méthodologie est très utile pour des systèmes assez simples. Pour des systèmes plus complexes, les lois physiques ne sont pas suffisantes et le temps de calcul du modèle peut devenir inacceptable. Dans ce cas, une autre approche, nommée identification de systèmes, qui construit le modèle à partir de mesures expérimentales d'entrée et de sortie, est utilisée.

Au cours des dernières années, une classe de systèmes complexes, les systèmes hybrides, qui impliquent l'interaction des comportements continus et discrets, a attiré l'attention de la communauté de recherche en automatique.

- Dans la conception de systèmes technologiques, les concepteurs de plusieurs disciplines comme l'électronique, la mécanique, l'informatique... ont besoin de coopérer. Travailler en groupes parallèles nécessite une compréhension et des modèles communs.
- Dans les systèmes contrôlés en réseau, le comportement des sous-systèmes est influencé ou contrôlé par des événements transmis sur le réseau. Les systèmes contrôlés en réseau peuvent âtre considérés comme des systèmes hybrides complexes.
- Il existe de nombreux processus physiques impliquant différents comportements dynamiques : systèmes de mouvement avec des modèles de frottement statique et cinétique (phénomène *stick-slip*), jeux et zones mortes dans les engrenages, transitions de mode pour des diodes... Ces processus peuvent âtre bien sûr mieux décrits par un modèle hybride que par un modèle dynamique unique.

La modélisation de ces systèmes est un problème intéressant et stimulant, qui a été étudié dans la littérature, mais pose encore des questions ouvertes.

Objectif

La thèse considère une classe de systèmes hybrides qui, pour une sortie y, peuvent âtre écrits en temps discret sous la forme de modèles autorégressifs avec entrée exogène (ARX) et à commutation :

$$y_k = f_{q_k}(\boldsymbol{x}_k) + v_k, \tag{1}$$

où $\boldsymbol{x}_k = [y_{k-1} \ \dots \ y_{k-n_a}, \ u_{k-1} \ \dots \ u_{k-n_b}]^\top$ est l'état continu (ou vecteur de régression) de dimension d contenant les dernières sorties y_{k-i} et entrées u_{k-i}, $q_k \in \{1, ..., s\}$ est l'état discret (ou mode) déterminant lequel des sous-systèmes $\{f_j\}_{j=1}^s$ est actif à l'instant k, et v_k est un terme de bruit additif.

Comme la plupart des travaux sur l'identification de systèmes hybrides dans la littérature, nous nous concentrons uniquement sur la modélisation de deux classes de systèmes hybrides: les systèmes lisses par morceaux et les systèmes à commutation arbitraire.

- **Systèmes lisses par morceaux**

  L'état discret q_k dépend de l'état continu $\boldsymbol{x}_k$:

$$q_k = \begin{cases} 1 & \text{si } \boldsymbol{x}_k \in \Re_1 \\ \vdots & \\ s & \text{si } \boldsymbol{x}_k \in \Re_s \end{cases} \tag{2}$$

  où les régions $\Re_j, j \in \{1, \dots, s\}$, forment une partition de l'espace de régression. Dans chaque région, le sous-modèle $f_j(\boldsymbol{x})$ doit âtre une fonction lisse. Les régions sont souvent des polyèdres convexes [103, 39, 77, 12]. Cependant, elles peuvent âtre plus complexes telles que présentées dans [58]. Une sous-classe populaire des systèmes lisses par morceaux considérée dans la littérature est la classe des systèmes affines par morceaux.

- **Systèmes à commutation arbitraire**

  L'état discret est indépendant de l'état continu et peut âtre choisi arbitrairement dans un ensemble fini de modes, c'est-à-dire $q_k \in \{1, 2, \dots, s\}$.

La thèse porte sur l'identification de systèmes avec ces modèles hybrides et cherche à résoudre le problème suivant.

Soit un ensemble de couples de données $\{(\boldsymbol{x}_k, y_k)\}_{k=1}^N$, généré par un système hybride (1), estimer :

- le nombre de sous-modèles s,
- les sous-modèles $\{f_j\}_{j=1}^s$,
- la séquence de commutation $\{q_k\}_{k=1}^N$.

Approches

Boîte noire

La thèse considère les systèmes dynamiques hybrides comme des boîtes noires dont le fonctionnement est analysé ou déterminé à partir de leurs entrées et sorties. Cette approche de modélisation nécessite un minimum de connaissances sur les systèmes. Elle s'applique dans les nombreux cas où les phénomènes physiques mis en jeu sont inconnus ou trop complexes pour conduire à un modèle exploitable.

Parcimonie

Le terme "parcimonie" (*sparsity*) évoque des choses peu nombreuses et dispersées. Plus formellement, un vecteur parcimonieux, ou creux, est un vecteur dont la plupart des éléments sont nuls. La notion apparaît dans les domaines des mathématiques appliquées, de l'informatique et du traitement du signal, y compris l'acquisition compressée (*compressive* *sensing*) [30, 41, 33]. Elle a attiré l'attention de nombreux chercheurs et de grands progrès théoriques et pratiques ont été réalisés au cours des dernières années. L'idée de parcimonie a été appliquée récemment à l'identification de systèmes. En particulier, elle montre un fort potentiel pour les systèmes hybrides, où elle n'a été que très récemment étudiée.

Classification

La classification a pour but de grouper les données en sous-ensembles cohérents par rapport aux hypothèses de travail. Dans l'identification de systèmes hybrides, elle est une tâche essentielle pour déterminer la séquence de commutation. Dans l'étape d'apprentissage, elle permet de trouver les points de données associés à un sous-modèle de régression. Dans l'étape de prédiction, elle permet de déterminer le mode du système pour calculer la prédiction de sa sortie.

Optimisation

En identification de systèmes, le modèle est souvent déterminé par optimisation de certains critères. Si l'optimisation convexe fournit une solution globale, l'optimisation non-convexe ne garantit qu'une solution locale. Il existe de nombreux solveurs disponibles pour ces deux problèmes. L'optimisation convexe est plus appréciée grâce à sa solution globale. Cependant, l'optimisation non-convexe doit également âtre étudiée, en particulier dans le cas de l'identification de systèmes hybrides qui est un problème non-convexe par nature. Elle doit âtre vue comme un problème difficile plutôt qu'insoluble, dont la résolution est gourmande en temps de calcul.

Chapitre 1 - Introduction

Le premier chapitre présente les systèmes dynamiques hybrides et l'identification de systèmes. Des modèles spécifiques et quelques définitions de base utilisées en identification ainsi que dans les chapitres suivants sont présentés. Les problèmes étudiés dans la thèse sont mis en évidence : l'identification de systèmes hybrides à partir de données expérimentales avec des modèles ARX affines à commutation ou affines par morceaux et des modèles hybrides avec des sous-modèles non-linéaires.

Une section est également dédiée aux techniques de régression non-linéaire de l'apprentissage statistique (*machine learning*) et introduit les concepts de surapprentissage et de régularisation ainsi que les méthodes à noyaux. Ces approches sont intéressantes car elles permettent d'approcher des non-linéarités arbitraires sans connaissances a priori sur la forme de la fonction recherchée.

Les concepts liés à la parcimonie, utilisés tout au long de la thèse, sont également introduits dans ce chapitre.

Chapitre 2 - Etat de l'art

Les principales méthodes de la littérature pour l'identification de systèmes hybrides linéaires, où $f_j(\boldsymbol{x}) = \boldsymbol{\theta}_j^\top \boldsymbol{x}$, sont passées en revue dans ce chapitre. Les grands principes sur lesquels reposent la plupart des méthodes sont synthétisés.

Approche algébrique. L'idée principale de l'approche algébrique est d'employer la contrainte de découplage hybride (HDC), une relation toujours vérifiée quelle que soit la séquence de commutation. Sous les hypothèses d'un nombre de modes s connu et de données sans bruit, la HDC exprime que le produit des erreurs de s sous-modèles est nul. La HDC est ensuite développée sous forme polynomiale avec un vecteur de coefficients $\boldsymbol{h}$. Les N équations polynomiales obtenues découplent les deux sous-problèmes de l'estimation des paramètres $\boldsymbol{\theta}_j$ et de la détermination de la séquence de commutation $\{q_k\}$. En particulier, la résolution de ces équations en $\boldsymbol{h}$ permet de retrouver les paramètres du système sans devoir évaluer le mode q_k.

Approche basée sur la classification. Dans cette approche, qui s'applique aux systèmes affines par morceaux, le sous-problème de l'estimation de la séquence de commutation $\{q_k\}$ est traité comme un problème de classification non supervisée. Le principe de base est de commencer par classer les données. Si les points de données sont bien classés, le problème d'estimation des sous-modèles devient facile. Dans l'hypothèse où le système est localement linéaire, les régions voisines partagent le mâme vecteur de paramètres. Les points de données représentés par des vecteurs de paramètres locaux similaires peuvent alors âtre regroupés. Chaque sous-modèle est ensuite estimé à partir des données d'un des groupes.

Approche à erreur bornée. L'idée principale est d'obtenir un modèle hybride avec une erreur maximale prédéfinie sur l'ensemble des données. Cette approche tente donc de trouver un nombre *minimum* de sous-modèles tel que, pour chaque point de données, au moins l'un de ces sous-modèles a une erreur inférieure à un seuil donné δ. Il s'agit d'un changement de perspective par rapport à la plupart des autres méthodes qui minimisent l'erreur pour un nombre de modes fixé.

Approche basée sur la parcimonie et la relaxation convexe. Dans le cadre de l'identification de systèmes hybrides, un sous-modèle peut être estimé de telle sorte qu'il produise un vecteur creux dont les composantes nulles correspondent aux points de données de ce sous-modèle et les composantes non-nulles aux points de données des autres sous-modèles. Ce principe conduit à minimiser la norme ℓ_0 d'un vecteur, c'est-à-dire le nombre de ses composantes non-nulles. La minimisation de la norme ℓ_0 étant NP difficile en général, une relaxation convexe basée sur la norme ℓ_1 est généralement utilisée.

L'approche basée sur l'ordre temporel. Cette approche exploite l'ordre temporel des points de données pour détecter les commutations entre les modes. S'il n'y a pas de commutation dans une période de temps donnée, le sous-modèle ne varie pas. A partir de l'ensemble des données, l'approche détermine un sous-modèle bien ajusté aux données et à variation limitée et utilise un autre sous-modèle uniquement lorsque la variation est trop forte.

Les méthodes développées à partir de ces approches pour les systèmes à commutation et les systèmes affines par morceaux sont ensuite détaillées dans ce chapitre. Pour chaque type de système, elles sont classées en fonction de l'hypothèse sur le nombre de modes, fixé ou non.

Chapitre 3 - Approche géométrique

Ce chapitre propose une nouvelle approche pour l'identification de systèmes hybrides linéaires basée sur les propriétés géométriques des systèmes hybrides dans l'espace des paramètres. Plus précisément, un point de donnée détermine un hyper-plan dans l'espace des paramètres et les hyper-plans d'un mode se croisent au vecteur de paramètres du sous-modèle correspondant. Nous considérons les projections orthogonales d'un point sur les hyper-plans d'un mode et montrons qu'elles se situent sur une hyper-sphère. Les données sont alors représentées par des hyper-sphères correspondant aux modes. Nous montrons ensuite comment ces hypersphères peuvent âtre séparées par l'Analyse en Composantes Principales (PCA) et proposons une condition sous laquelle cette séparation est optimale pour les systèmes à deux modes. Enfin, la régression classique est appliquée pour estimer les paramètres des sous-modèles à partir des ensembles de données classées. Une procédure simple inspirée de l'approche à erreur bornée est également proposée pour étendre la méthode à l'identification des systèmes à commutation à plus de deux modes et les expériences montrent que l'algorithme final peut estimer avec précision à la fois les paramètres et le nombre de modes tout en étant simple à appliquer et beaucoup plus robuste au bruit que les autres méthodes.

Chapitre 4 - Minimisation sélective de la norme ℓ_1 pour l'optimisation de la parcimonie

La première partie de ce chapitre traite du problème du calcul de la solution la plus parcimonieuse de systèmes d'équations linéaires sous-déterminés, comme étudié dans la littérature de l'acquisition compressée (*compressive sensing*). Ce problème apparaît comme central dans les approches basées sur la parcimonie pour l'identification de systèmes hybrides. Plus précisément, nous nous concentrons sur la relaxation convexe du problème basée sur la norme ℓ_1 où les conditions sur la parcimonie de la solution pour l'équivalence ont été étudiées. Dans ce cadre, l'algorithme itératif de repondération de Candès [25] est typiquement utilisé pour améliorer la parcimonie de la solution de l'optimisation de la norme ℓ_1. Nous proposons un nouvel algorithme pour améliorer les conditions sous lesquelles la relaxation fournit la solution la plus creuse. Nous prouvons la convergence du nouvel algorithme en un nombre fini d'itérations et établissons des conditions suffisantes pour la convergence vers la solution la plus creuse. Les expériences montrent que l'algorithme proposé améliore considérablement les approches précédentes pour l'acquisition compressée.

La deuxième partie de ce chapitre montre comment ces résultats peuvent âtre utilisés pour l'identification de systèmes hybrides linéaires. En particulier, nous suivons l'approche basée sur l'optimisation de la parcimonie du vecteur d'erreur [5]. Les sous-modèles du système hybride sont estimés un à un avec le nouvel algorithme qui permet d'assouplir les conditions d'optimalité.

Chapitre 5 - Identification de systèmes hybrides non linéaires

La plupart des approches d'identification considèrent que les systèmes hybrides commutent entre des dynamiques linéaires. Les dynamiques non linéaires sont traitées par seulement quelques méthodes [58, 59, 7, 38], développées dans le cadre de l'optimisation continue [60] ou de l'optimisation parcimonieuse [5, 38]. Le chapitre se concentre sur ces deux cadres en considérant des sous-modèles à noyaux pour approcher des non-linéarités arbitraires.

Dans le premier cadre, afin de maintenir l'efficacité pour de grands jeux de données, une étape de prétraitement est nécessaire pour fixer la taille des sous-modèles et limiter le nombre de variables d'optimisation. Quelques approches construisant des sous-modèles à noyaux de taille réduite sont alors proposées pour l'identification de systèmes hybrides non linéaires. Elles sont comparées dans des expériences numériques, qui montrent que le traitement proposé permet d'obtenir simultanément la classification des points de données et une approximation des comportements non linéaires de manière efficace et précise.

Dans le second cadre, les sous-modèles sont estimés un à un en maximisant la parcimonie du vecteur d'erreur correspondant. Nous étendons cette approche de plusieurs façons. Tout d'abord, nous relaxons la condition de parcimonie en introduisant la parcimonie robuste, qui peut âtre optimisée par la minimisation d'une norme ℓ_1 modifiée ou, de façon équivalente, de la fonction de perte ε-insensible. Ensuite, nous montrons que, selon le choix du terme de régularisation, la méthode est équivalente à différentes formes de régression à vecteurs support [85], une méthode populaire en apprentissage statistique. Plus précisément, les sous-modèles peuvent âtre estimés en résolvant itérativement un problème classique de régression à vecteurs support, où la parcimonie des vecteurs support correspond à celle du vecteur d'erreur dans le cadre de l'identification de systèmes hybrides. Cela permet de transférer les résultats théoriques et algorithmiques du domaine de l'apprentissage statistique à l'identification de systèmes hybrides. En particulier, des techniques d'optimisation efficaces sur de grands jeux de données sont obtenues.

Chapitre 6 - Identification de systèmes lisses par morceaux

Alors que les précédents chapitres ont considéré les systèmes à commutation arbitraire, ce chapitre se concentre sur les systèmes lisses par morceaux, dont le mode dépend du vecteur de régression. Dans ce cas, l'identification équivaut à un problème de régression lisse par morceaux. Les méthodes d'identification de systèmes à commutation arbitraire de la littérature peuvent âtre appliquées avec plus ou moins d'efficacité, et seulement pour des systèmes affines par morceaux. Ce chapitre propose une nouvelle méthode qui approche les systèmes lisses par morceaux par un modèle unique, mais flexible, appartenant à une classe de fonctions lisses construites comme des combinaisons linéaires de noyaux. Bien que contraint à âtre globalement lisse pour éviter le surapprentissage en cas de données bruitées, le modèle estimé peut avoir de très grandes dérivées à certains endroits pour approcher la discontinuité de la fonction cible. Ceci est obtenu en introduisant de nouveaux termes de régularisation pénalisant les dérivées de manière locale et des algorithmes d'apprentissage sous forme de programmes d'optimisation convexe. Le modèle lisse obtenu peut ensuite âtre transformé en un modèle lisse par morceaux. Cette approche est appliquée sur des exemples d'identification de systèmes dynamiques hybrides et de reconstruction d'image.

Conclusions et perspectives

La thèse considère le problème d'identification de systèmes hybrides sous différents aspects : géométrie, parcimonie et non-linéarité. Les méthodes proposées ont des avantages notables et ont prouvé leur efficacité par rapport aux autres méthodes sur des exemples numériques.

Contributions

Les principaux résultats de la thèse peuvent âtre résumés et reliés aux publications de l'auteur ainsi.

- *Optimisation convexe.* De nombreuses méthodes de la littérature pour l'identification de systèmes hybrides peuvent âtre considérées comme des algorithmes d'optimisation appliquées à une formulation du problème non-convexe, et donc souffrent généralement de la présence de minima locaux et d'une forte sensibilité à l'initialisation. Les nouvelles méthodes proposées dans [J2, C1, C2, C3, C4] reformulent le problème de l'identification de systèmes hybrides comme un programme d'optimisation convexe pour lequel une solution globale peut âtre calculée de manière efficace.

- *Réglage aisé des hyperparamètres.* De nombreuses méthodes d'identification de systèmes hybrides bien connues comprennent un ou plusieurs hyperparamètres qui peuvent âtre difficiles à régler, en particulier lorsque ces hyperparamètres sont sensibles, dépendants ou impossibles à interpréter à haut niveau. Dans cette thèse, de nouvelles méthodes d'identification de systèmes hybrides sans ou avec seulement peu de paramètres facilement réglables sont proposées [J2, C2, C3].

- *Application à de grands jeux de données.* Dans de nombreuses applications, de grands jeux de données doivent âtre traités pour bénéficier d'un maximum d'information. Cela nécessite des méthodes efficaces en terme de temps de calcul. C'est également vrai pour l'identification de systèmes hybrides, et devient particulièrement critique pour les systèmes hybrides non linéaires. C'est une des principales limites de nombreuses méthodes. Dans cette thèse, certaines méthodes rapides sont proposées pour traiter des ensembles de données volumineux [J1, C3].

- *Parcimonie.* Cette thèse propose une nouvelle méthode itérative [J2] basée sur la minimisation d'une norme ℓ_1 pour l'obtention de solutions parcimonieuses. L'algorithme proposé offre trois avantages majeurs par rapport à celui de [25] : i) il converge en un nombre fini d'itérations, ii) des garanties théoriques de convergence vers la solution la plus parcimonieuse peuvent âtre obtenues, et iii) des expériences ont montré qu'il permet de retrouver la solution la plus parcimonieuse dans des cas plus difficiles. Les caractéristiques de cet algorithme nous permettent d'améliorer le cadre de l'optimisation parcimonieuse [5] pour l'identification de systèmes hybrides.

Perspectives

- *Approche géométrique.* Pour les systèmes à commutation à deux modes, la classification des données à partir de la direction d'inertie principale est affectée par la structure des deux hypersphères et la distribution des points sur les deux hypersphères. La condition sur les données proposée pour l'optimalité de la méthode est invérifiable en pratique. Des conditions vérifiables devraient âtre étudiées.

  Par ailleurs, bien qu'un algorithme itératif inspiré de l'approche à erreur bornée ait été proposé pour traiter le cas à plus de deux modes, une étude théorique de cet algorithme est nécessaire. En outre, la séparation directe de l'ensemble des hypersphères qui, dans ce cas, ne peut pas être bien traitée par la PCA est une voie de recherche intéressante.

- *Minimisation sélective de la norme ℓ_1 pour l'optimisation de la parcimonie.* Les travaux à venir se concentreront sur l'analyse de la condition d'obtention de la solution la plus parcimonieuse proposée et la comparaison avec les résultats classiques obtenus pour la méthode de poursuite de base (*basis pursuit*) correspondant à la première itération de l'algorithme proposé.

  En ce qui concerne l'identification de systèmes hybrides, bien que la méthode s'est avérée robuste au bruit dans les expériences, d'autres investigations devraient âtre menées afin d'étendre les résultats théoriques aux données bruitées.

  Dans une perspective plus large, d'autres applications de l'algorithme générique de renforcement de parcimonie pourraient âtre envisagées, comme suggéré par la grande variété de problèmes formulés comme des problèmes d'optimisation parcimonieuse dans divers domaines, voir, par exemple, [25].

- *Identification de systémes hybrides non linéaires.* Dans le cadre de l'optimisation parcimonieuse, l'algorithme modifié a introduit un nouveau paramètre ν, qui peut âtre interprété comme la fraction de données considérées comme des valeurs aberrantes pour le sous-modèle. La relation précise entre ce paramètre et la fraction des données générées par chaque mode, qui est également impliquée dans les conditions de parcimonie, est encore inconnue. En particulier, la caractérisation de l'influence de l'algorithme itératif de repondération sur le choix de ν reste une question ouverte.

  Une autre direction de recherche porte sur le calcul des chemins de solutions complets (*full solution paths*) en ce qui concerne la constante de régularisation λ et les autres hyper-paramètres. Le but est d'obtenir des modèles pour toutes les valeurs possibles des hyper-paramètres avec un coût de calcul faible. Pour cette question, nous pouvons profiter de l'équivalence avec la régression à vecteurs support et des nombreux résultats sur ce sujet.

- *Identification de systèmes lisses par morceaux.* Résoudre directement les problèmes d'optimisation sous contraintes proposés peut devenir prohibitif pour les ensembles de données très volumineux. Les travaux futurs pourraient envisager des algorithmes plus rapides pour l'optimisation sans contrainte d'une fonction de coût lissée.

  Une autre direction de recherche avec des conséquences pratiques concerne la dérivation du chemin de solution complet pour la constante de régularisation λ.

Bibliographie de l'auteur

Articles de journal

[J1] V.L. Le, G. Bloch, and F. Lauer. Reduced-Size Kernel Models for Nonlinear Hybrid System Identification. *IEEE Transactions on Neural Networks*, 22(12):2398–2405, 2011.

[J2] V.L. Le, F. Lauer and G. Bloch. Selective ℓ_1 minimization for sparse recovery. *IEEE Transactions on Automatic Control, 2013* (accepté provisoirement).

Articles de conférence

[C1] F. Lauer, V.L. Le and G. Bloch. Learning smooth models of nonsmooth functions via convex optimization. In Proc. of *22nd IEEE Int. Workshop on Machine Learning for Signal Processing (MLSP 2012)*, Santander, Spain, September 23-26, 2012.

[C2] V.L. Le, F. Lauer and G. Bloch. Identification of linear hybrid systems: a geometric approach. In Proc. of *American Control Conference (ACC 2013)*, Washington DC, USA, June 17-19, 2013.

[C3] V.L. Le, F. Lauer, L. Bako and G. Bloch. Learning nonlinear hybrid systems: from sparse optimization to support vector regression. In Proc. of *16th ACM Int. Conf. on Hybrid Systems: Computation and Control (HSCC 2013)*, Philadelphia, USA, April 08-11, pp. 33-42, 2013.

[C4] L. Bako, V.L. Le, F. Lauer and G. Bloch. Identification of MIMO switched state-space models. In Proc. of *American Control Conference (ACC 2013)*, Washington DC, USA, June 17-19, 2013.

Contents

Summary and contributions

Context

In automatic control, an accurate model of a system has a very important role. Determining a model is always the first step for the further purposes as controller design, fault detection or prediction, etc... For simple systems, a model can be built by using physical and mathematical knowledge to describe behaviors or properties of simple subsystems as blocks, then, combine these blocks to obtain a model of the whole system. This methodology is difficult to apply to compact and complex systems which cannot be unpacked or analyzed in details. Instead of that, another useful methodology called system identification is considered. System identification aims at building models or estimating unknown parameters of dynamical systems from experimental data.

In recent years, a class of systems attracting research attention is the class of hybrid systems which involve the interaction of continuous and discrete behaviors. These systems are easily found all around us. Modeling these systems is naturally posed as an interesting and challenging problem.

Approaches to hybrid system identification

Black box

This thesis considers dynamical hybrid systems as black boxes whose operation is determined or analyzed through the system inputs and outputs. This approach to modeling, also known as agnostic learning, requires the least amount of knowledge about the systems. Thus, it is useful in many cases of identification of complex systems where the first principle laws are impossible to be implemented in real time or where involved physical phenomena are unknown.

Sparsity

The term "sparsity" is used to describe something scattered. More specifically, a sparse vector is a vector whose entries are mostly zeros and very few are non-zeros. This term appears in the areas of applied mathematics, computer science, and signal processing, including Compressive Sensing [30, 41, 33, 23]. Sparsity attracts attention of many researchers with great advancement about theory as well as practical tools. In recent years, it has been considered for system identification to obtain the sparsity in models. Especially, in the context of hybrid system, the application of sparsity shows a high potential and has only recently been studied.

Classification

Classification aims at grouping the data into coherent subsets according to working hypotheses. In hybrid system identification, it is an essential task to determine the system mode. In the learning step, it helps to find data points of a submodel for its estimation. In the prediction step, it helps to determine the mode to calculate the output prediction.

Optimization

In system identification, a model is determined by optimizing certain criteria. While convex optimization problems lead to global solutions, non-convex optimization problems normally lead to local solutions. There exists many available solvers for both problems. Convex optimization is more appreciated thanks to its global solution. However, non-convex optimization with its specific advantages also needs to be investigated, especially for the hybrid system identification problem with its non-convex nature. Non-convex optimization is a difficult problem rather than an intractable one although it requires much computing time to be solved with an arbitrary accuracy.

Thesis outline

Chapter 1 - Introduction

This chapter introduces hybrid dynamical systems and system identification. Specific models and some basic definitions used in identification as well as in the following chapters are presented. The problems investigated in the thesis are highlighted, namely, hybrid system identification from experimental data for switched affine or piecewise affine ARX models as well as hybrid models with arbitrary nonlinear submodels. A section is also dedicated to nonlinear regression techniques borrowed from machine learning and introduces the concepts of overfitting, regularization and the kernel methods. The concepts related to sparsity which are used throughout the thesis are also introduced.

Chapter 2 - A state of the art

The main methods in the literature for hybrid system identification, particularly for linear hybrid systems, are reviewed in this chapter. After the presentation of the different points of view (algebraic, bounded-error, clustering, sparsity and convex relaxation, and time order), the methods are considered for switched systems and for piecewise systems. For each type of systems, the methods are classified with respect to the assumption on the number of modes, which can be fixed or not.

Chapter 3 - Geometric approach

This chapter proposes a new approach for the identification of linear hybrid systems based on the geometric properties of hybrid systems in the parameter space. More precisely, the data are mapped in that space such that each submodel is represented by a hypersphere. Then, we show how the hyperspheres can be easily separated by Principal Component Analysis (PCA) and derive a condition under which this separation is optimal for systems with two modes. Finally, classical regression is applied to estimate the system parameters from the classified data set. A simple procedure is also proposed to extend the method to the identification of switched systems with more than two modes. Experiments show that the final algorithm can accurately estimate both the parameters and the number of modes while being simple to apply and far more robust to noise than other methods.

Chapter 4 - Selective ℓ_1 minimization for sparsity optimization

The first part of the chapter deals with the recovery of sparse solutions of underdetermined systems of linear equations, as studied in the compressive sensing literature. More precisely, we focus on the convex relaxation of the problem, where the ℓ_1-norm of the variables is minimized, and propose a new iteratively reweighted scheme in order to improve the conditions under which this relaxation provides the sparsest solution. We prove the convergence of the new scheme and derive sufficient conditions for the convergence towards the sparsest solution. Experiments show that the new scheme significantly improves the previous approaches for compressive sensing.

Then, the second part of the chapter shows how these results can be used in linear hybrid system identification. In particular, we follow the approach of [5] where the new scheme allows us to relax the conditions on the data.

Chapter 5 - Nonlinear hybrid system identification

Most of the approaches proposed to solve the hybrid system identification problem consider only hybrid systems switching between linear dynamics. The nonlinear dynamics are dealt with by only a few methods [58, 59, 7, 38] which are developed in the frameworks of continuous optimization [60] or sparse optimization [5, 38]. This chapter focuses on these frameworks. To be able to approximate arbitrary nonlinearities, kernel submodels are considered.

Within the first framework, in order to maintain efficiency for large data sets, a preprocessing step is required to fix the submodel sizes and limit the number of optimization variables. In this chapter, some approaches to deal with this issue and build sparse kernel submodels are reviewed and proposed. These are compared in numerical experiments, which show that the overall method achieves the simultaneous classification of data points and approximation of the nonlinear behaviors in an efficient and accurate manner.

Within the second framework, the submodels are iteratively estimated one by one by maximizing the sparsity of the corresponding error vector. We extend this approach in several ways. First, we relax the sparsity condition by introducing robust sparsity, which can be optimized through the minimization of a modified ℓ_1-norm or, equivalently, of the ε-insensitive loss function. Then, we show that, depending on the choice of regularizer, the method is equivalent to different forms of support vector regression, i.e., a popular class of kernel methods. More precisely, the submodels can be estimated by iteratively solving a classical support vector regression problem, in which the sparsity of support vectors relates to the sparsity of the error vector in the considered hybrid system identification framework. This allows us to extend theoretical results as well as efficient optimization algorithms from the field of machine learning to the hybrid system framework.

Chapter 6: Piecewise smooth system identification

While the previous chapters considered arbitrarily switched systems, this chapter focuses on piecewise smooth systems, in which the mode depends on the regression vector. In this case the identification amounts to a piecewise smooth regression problem. This chapter proposes an approach where the model belongs to a class of smooth functions built as kernel expansions. Though constrained to be globally smooth, the trained model can have very large derivatives at particular locations to approximate the nonsmoothness of the target function. This is obtained through the definition of new regularization terms which penalize the derivatives in a location-dependent manner and training algorithms in the form of convex optimization programs. Examples of application to hybrid dynamical system identification and image reconstruction are provided.

Contributions

The main results of the thesis can be summarized and related to the author's publications as follows.

Convex optimization

Many methods in the literature for hybrid system identification can be seen as optimization algorithms applied to a nonconvex problem formulation, and thus typically suffer from the presence of local minima and the sensitivity to the initialization. On the other hand, the new methods proposed in [J2, C1, C2, C3, C4] reformulate the problem of hybrid system identification as a convex optimization program for which a global solution can be computed efficiently.

Easy tuning of the hyperparameters

Many well-known hybrid system identification methods include one or several hyperparameters that can be difficult to tune, especially when these parameters are sensitive, dependent or impossible to be interpreted at a high level. In this thesis, new methods for hybrid system identification with no or only a few easily tunable parameters are proposed [J2, C2, C3].

Application to large data sets

In many applications, we need to process large data sets to benefit from the maximal amount of information, and this requires fast computing methods. This is also true in hybrid system identification, and becomes especially critical for nonlinear hybrid systems. This is one of the main limitations of many methods. In this thesis, some computationally efficient methods are proposed to handle large data sets [J1, C3].

Sparsity

This thesis proposes a new iterative method [J2] based on ℓ_1-norm minimization for the recovery of sparse solutions of underdetermined linear systems. The proposed scheme offers three major advantages when compared with the one of [25]: i) it converges in a finite number of iterations, ii) theoretical guarantees of convergence towards the sparsest solution can be obtained, and iii) experiments showed that it allows the sparsest solution to be recovered in a larger range of sparsity level. Then, the advantages of this new sparsity enhancing scheme allow us to improve the framework of [5] for hybrid system identification.

Author's bibliography

Journal papers

[J1] V.L. Le, G. Bloch, and F. Lauer. Reduced-Size Kernel Models for Nonlinear Hybrid System Identification. *IEEE Transactions on Neural Networks*, 22(12):2398–2405, 2011.

[J2] V.L. Le, F. Lauer and G. Bloch. Selective ℓ_1 minimization for sparse recovery. *IEEE Transactions on Automatic Control, 2013* (provisionally accepted).

Conference papers

[C1] F. Lauer, V.L. Le and G. Bloch. Learning smooth models of nonsmooth functions via convex optimization. In Proc. of *22nd IEEE Int. Workshop on Machine Learning for Signal Processing (MLSP 2012)*, Santander, Spain, September 23-26, 2012.

[C2] V.L. Le, F. Lauer and G. Bloch. Identification of linear hybrid systems: a geometric approach. In Proc. of *American Control Conference (ACC 2013)*, Washington DC, USA, June 17-19, 2013.

[C3] V.L. Le, F. Lauer, L. Bako and G. Bloch. Learning nonlinear hybrid systems: from sparse optimization to support vector regression. In Proc. of *16th ACM Int. Conf. on Hybrid Systems: Computation and Control (HSCC 2013)*, Philadelphia, USA, April 8-11, pp. 33-42, 2013.

[C4] L. Bako, V.L. Le, F. Lauer and G. Bloch. Identification of MIMO switched state-space models. In Proc. of *American Control Conference (ACC 2013)*, Washington DC, USA, June 17-19, 2013.

Notations

Typography

Scalars are written in default and lowercase letters (e.g. a), unless otherwise specified.
Vectors are column vectors written in boldface and lowercase letters (e.g. $\boldsymbol{a}$), unless otherwise specified. For a vector $\boldsymbol{a}$, the ith entry is denoted a_i.
Matrices are written in boldface and uppercase letters (e.g. $\boldsymbol{A}$), unless otherwise specified. For a matrix $\boldsymbol{A}$, the jth column is denoted $\boldsymbol{A}_j$ and the entry in the ith row and the jth column is denoted A_{ij} or a_{ij}.

General notations

$\mathbf{1}_n$	: vector in $\mathbb{R}^n$ with all elements equal to 1
$\mathbf{0}_n$	: vector in $\mathbb{R}^n$ with all elements equal to 0
$\boldsymbol{a}^\top$	: transpose of $\boldsymbol{a}$
$\boldsymbol{a}^*$	: optimal estimate of $\boldsymbol{a}$
$\hat{\boldsymbol{a}}$	: estimate of $\boldsymbol{a}$
$\lvert a\rvert$	: absolute value of a
$\mathrm{diag}(\boldsymbol{a})$	: diagonal matrix with the components of $\boldsymbol{a}$ on the diagonal
$\boldsymbol{I}_n$	: identity matrix in $\mathbb{R}^{n\times n}$ with $(\boldsymbol{I}_n)_{ii} = 1$ and all other components equal to 0
$\lvert S\rvert$	: cardinality of the set S
$\mathrm{Trace}(\boldsymbol{A})$	: trace of matrix $\boldsymbol{A}$
$\delta_{i,j}$	: Kronecker delta which is 1 if $i = j$ and is 0 if $i \neq j$
$\langle \boldsymbol{x}, \boldsymbol{z}\rangle_{\mathcal{F}}$	: inner product between $\boldsymbol{x}$ and $\boldsymbol{z}$ in $\mathcal{F}$
$\odot$	: Hadamard (entrywise) product
$\otimes$	: Kronecker product

Norms

$\lVert\boldsymbol{a}\rVert_p$	: ℓ_p-norm of $\boldsymbol{a}$, $\lVert\boldsymbol{a}\rVert_p = (\lvert a_1\rvert^p + \cdots + \lvert a_d\rvert^p)^{1/p}$ with $\boldsymbol{a} \in \mathbb{R}^d$
$\lVert\boldsymbol{a}\rVert_1$	: ℓ_1-norm of $\boldsymbol{a}$, $\lVert\boldsymbol{a}\rVert_1 = \lvert a_1\rvert + \cdots + \lvert a_d\rvert$ with $\boldsymbol{a} \in \mathbb{R}^d$
$\lVert\boldsymbol{a}\rVert_2$	: (Euclidean) ℓ_2-norm of $\boldsymbol{a}$, $\lVert\boldsymbol{a}\rVert_2 = \sqrt{\lvert a_1\rvert^2 + \cdots + \lvert a_d\rvert^2}$ with $\boldsymbol{a} \in \mathbb{R}^d$
$\lVert\boldsymbol{a}\rVert_\infty$	: ℓ_∞-norm of $\boldsymbol{a}$, $\lVert\boldsymbol{a}\rVert_\infty = \max_i\{\lvert a_i\rvert\}$
$\lVert\boldsymbol{a}\rVert_0$	: ℓ_0-peudo norm of $\boldsymbol{a}$, the number of its non-zero entries, $\lVert\boldsymbol{a}\rVert_0 = \lvert\{i : a_i \neq 0\}\rvert$
$\lVert\boldsymbol{A}\rVert_F$	: Frobenius norm of a matrix $\boldsymbol{A}$, $\lVert\boldsymbol{A}\rVert_F = \sqrt{\sum_{i,j} A_{ij}^2}$
$\lVert\boldsymbol{A}\rVert_{max}$	: max-norm of a matrix $\boldsymbol{A}$, $\lVert\boldsymbol{A}\rVert_{max} = \max_{i,j} \lvert A_{ij}\rvert$

Specific notations

$\boldsymbol{\alpha}$	: parameter vector of SISO nonlinear systems
d	: number of parameters of SISO linear systems
$\mathcal{D}_0$	: data set of system input-output pairs
$\mathcal{D}$	: data set of regressor-output pairs
$\mathcal{D}_j$	: data subset of regressor-output pairs generated by jth mode
$\mathcal{D}_{\boldsymbol{x}}$	: data set of regression vectors
e_k	: prediction error at discrete time k of single output systems
f	: function, model to be estimated
f_j	: submodel j
k	: discrete-time or point index
$\kappa(\cdot,\cdot)$	: kernel function
$\boldsymbol{K}$	: kernel or Gram matrix $\boldsymbol{K} = \{\kappa(\boldsymbol{x}_i, \boldsymbol{x}_k)\}$
$\ell(\cdot,\cdot)$	: loss function
λ	: regularization parameter
n_a	: number of lagged outputs in the regression vector
n_b	: number of lagged inputs in the regression vector
N	: number of (training) data points
N_t	: number of (test) data points
N_v	: number of (validation) data points
$\boldsymbol{\phi}$	: nonlinear mapping $\boldsymbol{\phi} : \boldsymbol{x} \mapsto \boldsymbol{\phi}(\boldsymbol{x})$
q	: mode index (discrete state)
q_k	: active mode at discrete-time k
$\mathcal{R}$	: regularizer
$\Re_j$	: domains of the partitioned regression space
s	: number of modes (submodels)
$\boldsymbol{\theta}$	: parameter vector of SISO linear systems
u_k	: input at discrete time k of single input systems
v_k	: noise term at discrete time k of single output systems
$\boldsymbol{x}$	: regression vector
$\boldsymbol{X}$	: observation matrix composed of regression vectors $\boldsymbol{x}_k$ as columns
$\mathcal{X}$	: regression domain
y_k	: output at discrete time k of single output systems
$\boldsymbol{y}$	: target vector defined as $\boldsymbol{y} = [y_1\ y_2\ \dots\ y_N]^\top$
$\mathcal{Y}$	: output domain

Abbreviations

ARX	: AutoRegressive with eXogenous input
FVS	: Feature Vector Selection
KPCR	: Kernel Principal Component Regression
MCS	: Multilevel Coordinate Search
ME	: Minimum-of-Errors (estimator)
PE	: Product-of-Errors (estimator)

Abbreviations (cont.)

PWA	:	PieceWise Affine
PWC	:	PieceWise Constant
PWQ	:	PieceWise Quadratic
PWS	:	PieceWise Smooth
PWARX	:	PieceWise ARX
QP	:	Quadratic Program
RKPCR	:	Reduced Kernel Principal Component Regression
SARX	:	Switched ARX
SOCP	:	Second-Order Cone Program
SVM	:	Support Vector Machine
SVR	:	Support Vector Regression
s.t.	:	subject to
w.r.t.	:	with respect to

Chapter 1

Introduction

THIS chapter introduces hybrid dynamical systems and system identification. Specific models and some basic definitions used in identification as well as in the following chapters are presented. The problems investigated in the thesis are highlighted, namely, hybrid system identification from experimental data for switched affine or piecewise affine models as well as hybrid models with arbitrary nonlinear submodels. A section is also dedicated to nonlinear regression techniques borrowed from machine learning and introduces the concepts of overfitting, regularization and the popular kernel methods. The concepts related to sparsity which are used throughout the thesis are also introduced.

1.1 Hybrid dynamical systems

We are in the digital age where digitized data are spread and treated over networks of equipments such as computers, sensors, actuators, etc... Besides the mechanic-electric part, the systems now include a digital part and associate interdependent continuous and discrete components. The term "hybrid" is used to name such systems. More specifically, hybrid dynamical systems, also called hybrid systems for short, are dynamical systems that exhibit both continuous and discrete dynamical behaviors. The vector field defining the evolution of the continuous state depends on the discrete state. Hybrid systems provide a suitable framework for modeling systems in a wide range of engineering applications [20, 62].

There are many physical processes involving different dynamical behaviors. Obviously, these processes can be better described by a hybrid model than by a single dynamical model. In mechanical engineering, continuous motions may be interrupted by collisions. Other examples are backslash in gears or motion systems with friction models that distinguish between stick and slip modes. In electrical circuits, continuous phenomena, such as the charging of capacitors, are interrupted by switches or diodes.

Many control systems can also be considered as hybrid systems, such as chemical processes where the continuous evolution of chemical reactions is controlled by valves and pumps, thermal process where a thermostat controlling the temperature switches heating or cooling choices on or off. A more elaborate application is gear shift control in automatic transmissions of cars, where both continuous (throttle position, car velocity) and discrete (gear ratio) variables are involved. More generally, hybrid systems are natural models for computer-controlled systems since they involve a physical process and a computer.

The communication between systems become more and more important and widespread. In the networked control systems, the behavior of systems is influenced or controlled by events communicated over the network. Such systems can be considered as complex hybrid systems.

Nowadays, multiple disciplines like electronics, mechanics, computer science must cooperate for the overall design of technological systems. Working in parallel groups requires a common understanding and common models. Thus, hybrid system theory and models play an essential role as a foundation for the cross disciplinary design.

1.2 Hybrid system models

In recent years, hybrid systems have been more and more studied in the automatic control community. Depending on the mechanism for switching between the modes, there are many model subclasses of hybrid systems proposed in the literature: (extended) Linear Complementarity (LC) systems [46, 94, 31], Mixed Logical Dynamical (MLD) systems [13], Max-Min-Plus-Scaling (MMPS) systems [32], PieceWise Affine (PWA) systems [87] and arbitrarily switched systems [73]. Note that among the first four subclasses, it is possible to transform one into the other under (rather mild) additional assumptions [47]. As most works on hybrid system identification in the literature, we focus only on the modeling of the piecewise smooth systems, which include PWA systems, and the arbitrarily switched systems.

In this thesis, we consider discrete-time models of hybrid systems with a single output, written in the input-output form

$$y_k = f_{q_k}(\boldsymbol{x_k}), \tag{1.1}$$

where the output y_k depends on the continuous state $\boldsymbol{x}_k$ and the discrete state (or mode) q_k.

Two cases can be distinguished according to the dependence of the discrete state on the continuous state.

- **PieceWise Smooth (PWS) systems**

  The discrete state q_k depends on the continuous state $\boldsymbol{x}_k$, i.e.,

$$q_k = \begin{cases} 1 & \text{if } \boldsymbol{x}_k \in \Re_1 \\ \dots \\ s & \text{if } \boldsymbol{x}_k \in \Re_s \end{cases} \tag{1.2}$$

  where $\Re_j, j \in \{1, \dots, s\}$, are regions that form a partition of the regression space. In each region, the submodel $f_{q_k}(\boldsymbol{x}_k)$ must be a smooth function. The regions can be convex polyhedra as considered in many works [103, 39, 77, 12]. However, they can be more complex as presented in [58]. A popular subclass of piecewise smooth systems considered in the literature is the PWA system class where the subsystems f_j are affine.

- **Arbitrarily switched systems**

  The discrete state is independent of the continuous state and can be switched arbitrarily in a finite set of modes, i.e., $q_k \in \{1, 2, \dots, s\}$.

Hybrid systems can also be distinguished according to the continuous dynamics.

- **Linear hybrid systems**

A discrete-time linear Single-Input Single-Output (SISO) hybrid system can be modeled in the input-output form with an AutoRegressive with eXogenous input (ARX) model.

Formally, we focus on models written as

$$y_k = \boldsymbol{\theta}_{q_k}^\top \boldsymbol{x}_k + v_k, \tag{1.3}$$

where $y_k \in \mathbb{R}$ is the output at discrete time k, $\boldsymbol{x}_k = [y_{k-1}, \dots, y_{k-n_a},\ u_{k-1}, \dots, u_{k-n_b}]^\top \in \mathbb{R}^{n_a+n_b}$ is the regression vector, with the past outputs y_{k-i} and inputs u_{k-i} and the model orders n_a and n_b, $q_k \in \{1, \dots, s\}$ is the discrete state, s is the number of submodels, $\boldsymbol{\theta}_j \in \mathbb{R}^{n_a+n_b}$, $j = 1, \dots, s$, are the vectors of parameters defining each submodel and $v_k \in \mathbb{R}$ is a noise term. In the case of affine systems, the regression vector $\boldsymbol{x}_k$ is simply replaced by $\tilde{\boldsymbol{x}}_k = [\boldsymbol{x}_k^\top, 1]^\top$.

- **Nonlinear hybrid systems**

  For nonlinear hybrid systems, subsystems should be modeled by nonlinear functions. Particularly, a SISO nonlinear hybrid system is expressed by a set of s smooth functions $\{f_j\}_{j=1}^s$ as

  $$y_k = f_{q_k}(\boldsymbol{x}_k) + v_k, \tag{1.4}$$

  where $q_k, \boldsymbol{x}_k$ and v_k are defined as in (1.3) and $\{f_j\}_{j=1}^s$ are s nonlinear submodels.

The classes of hybrid systems in ARX form are summarized in Table 1.1, as in [58].

Table 1.1: Nomenclature of the hybrid models in ARX form

Classes	**abbr.**	**models** f_j	**discrete state** q	**domains** $\Re_j$	**Chapter discussed**
PieceWise Affine	PWARX	affine	function of $\boldsymbol{x}$	polyhedral	2, 6
PieceWise Smooth	PWSARX	smooth	function of $\boldsymbol{x}$	polyhedral	6
Nonlinearly PieceWise Affine	NPWARX	affine	function of $\boldsymbol{x}$	arbitrary	
Nonlinearly PieceWise Smooth	NPWSARX	smooth	function of $\boldsymbol{x}$	arbitrary	
Switched Affine	SARX	affine	arbitrary		2, 3, 4
Switched Nonlinear	SNARX	nonlinear	arbitrary		5

1.3 System identification

In automatic control, an accurate model of a system has a very important role. Determining a model is always the first step for the further purposes as controller design, fault detection or prediction, etc... For simple systems, a model can be built by using physical and mathematical knowledge to describe behaviors or properties of simple subsystems as blocks, then, combine these blocks to obtain a model of the whole system. This methodology is difficult to apply to compact and complex systems which cannot be unpacked or analyzed in details. Instead, another methodology, called system identification, is considered. System identification aims at building models of dynamical systems from experimental data.

We note that a mathematical model is never an exact description of a real-life system but only an approximation of some aspects considered as identification goals. In such a context, the measured data are assumed to be well generated for these goals.

A system identification procedure usually contains the steps of Procedure 1 [61].

Procedure 1 Identification procedure

1. Recording a data set $\mathcal{D}_0 = \{(u_k, y_k)\}_{k=1}^N$ via designed experiments.
2. Choosing class of models or the model structure.
3. Determining the "best" model in the class, guided by the data with a criterion of fit.
4. Validating the obtained model via a test procedure.

In this thesis, only the ARX model for SISO systems in discrete time with the regression vector

$$\boldsymbol{x}_k = [y_{k-1}, \dots, y_{k-n_a},\ u_{k-1}, \dots, u_{k-n_b}]^\top, \tag{1.5}$$

is considered. Thus, the data set is given directly in form of pairs of regressors and outputs,

$$\mathcal{D} = \{(\boldsymbol{x}_k, y_k)\}_{k=1}^N. \tag{1.6}$$

Other models typically considered in system identification are AutoRegressive Moving Average model with eXogenous inputs model (ARMAX), Output Error model (OE), etc... but they have not been used for hybrid system identification so far. Since the identification problem of hybrid system is already very complex, as in the majority of works, we consider the problem as a regression problem and assume that the data (1.6) are independent and identically distributed.

Loss function. An important concept involved in Procedure 1 is the one of a loss function. A loss function,

$$\ell : \mathbb{R}^2 \longrightarrow \mathbb{R}^+, \tag{1.7}$$

such that $\ell(y, y) = 0$ is typically used to evaluate a model and computed from the prediction error (one-step-ahead prediction error for dynamical systems) which is the difference between the system output y and the model prediction $\hat{y}$,

$$e = y - \hat{y}. \tag{1.8}$$

The most common loss functions used in regression are the squared loss function

$$\ell(y, \hat{y}) = (y - \hat{y})^2, \tag{1.9}$$

the absolute loss function

$$\ell(y, \hat{y}) = |y - \hat{y}|, \tag{1.10}$$

and the robust Hampel's loss function, given for instance in [28] (page 138) as

$$\ell(y, \hat{y}) = \begin{cases} \delta_v^2/\pi \left(1 - cos(\pi(y - \hat{y})/\delta_v)\right), & \text{if } |y - \hat{y}| \leq \delta_v, \\ 2\delta_v^2/\pi & \text{otherwise}, \end{cases} \tag{1.11}$$

where δ_v is a tunable parameter defining a threshold for outliers.

1.3.1 Linear system identification

The model class frequently chosen in Procedure 1 is the family of linear ARX models,

$$y_k = f(\boldsymbol{x}_k, \boldsymbol{\theta}) = \boldsymbol{\theta}^\top \boldsymbol{x}_k,$$

where $\boldsymbol{\theta} \in \mathbb{R}^{n_a+n_b}$ is the parameter vector and $\boldsymbol{x}_k$ is the regression vector defined in (1.5). The problem of linear system identification can be stated as follows.

Problem 1. *Given a data set $\mathcal{D}_0 = \{(u_k, y_k)\}_{k=1}^N$, estimate the model orders n_a and n_b and the model parameter vector $\boldsymbol{\theta}$.*

The identification of linear systems has been vastly studied in the literature for decades [86, 61]. In the simple case where the orders n_a and n_b are given, there are many methods to estimate the parameter vector $\boldsymbol{\theta}$. The prediction error method [61] which solves the optimization problem

$$\min_{\boldsymbol{\theta}} \sum_{k=1}^{N} \ell\left(y_k, \boldsymbol{\theta}^\top \boldsymbol{x}_k\right), \tag{1.12}$$

is perhaps the most popular.

1.3.2 Nonlinear system identification

For nonlinear systems, there are two types of model classes to choose from in Step 2 of Procedure 1: parametric models and non-parametric models. The nonlinearity can introduce additional difficulties for the optimization or lead to the overfitting/underfitting problem. In this section, besides dealing with parametric and non-parametric models, we also present the regularization approach for controling the model complexity and a popular class of non-parametric models. These concepts are essential for nonlinear system identification.

Parametric models

Parametric models are models with a fixed structure and a finite number of unknown parameters to be determined from the data. A well-posed parametric model class rapidly gives a good model. However, the restricted flexibility of models in such a limited class can easily lead to the underfitting problem: the model obtained by learning from the training data may badly approximate the nonlinearity of the system.

Non-parametric models

Non-parametric models are models with both the structure and the parameters of the model to be estimated from the data. Recent approaches developed in machine learning consider non-parametric models which can sufficiently well approximate any function. An example for a non-parametric model class is the linearly parameterized function class of function expansions [83],

$$\mathcal{H} = \left\{ f : f = \sum_{i=1}^{M} \alpha_i g_i(\cdot),\ \alpha_i \in \mathbb{R}, M \in \mathbb{N} \right\}, \tag{1.13}$$

where α_i are function weights and g_i are referred to as basis functions. The typical ones are Radial Basis Functions (RBF), wavelet functions, kernel functions...

One issue which must be considered with care is overfitting/underfitting. The typical function classes provide sufficient flexibility for the model to yield a perfect fit of the data. Thus, if we were to minimize the error on a data set, the model would learn the noise as well as the target function, i.e., overfit the training data. Overfitting can be detected, if the error on an independent test data set is significantly higher than the error on the training data set whereby both data sets have to be within the same range of the response variables to prevent additional biases due to extrapolation. In contrast to overfitting, underfitting appears when the function classes are not flexible enough to fit the data. Underfitting shows high errors for both data sets.

Most approaches to nonlinear modeling include a regularization scheme to control the complexity (or flexibility) of the model and avoid overfitting, as described next.

Regularization in system identification

Formally, the regularization approach to nonlinear modeling can be stated as follows. Given a training set $\mathcal{D}$ of N pairs $(\boldsymbol{x}_k, y_k) \in (\mathcal{X} \subset \mathbb{R}^d) \times (\mathcal{Y} \subset \mathbb{R})$, $k = 1, \dots, N$, the general goal is to learn a function f in some function class $\mathcal{H}$ such that it minimizes a regularized functional representing a trade-off between the fit to the data and some regularity conditions of f:

$$\min_{f \in \mathcal{H}} \sum_{k=1}^{N} \ell(y_k, f(\boldsymbol{x}_k)) + \lambda \mathcal{R}(f), \tag{1.14}$$

where the data term is defined through a loss function (1.7), $\mathcal{R}(f)$ is a general regularization term measuring the complexity of f and $\lambda \geq 0$ tunes the trade-off between the two terms.

If the class of functions $\mathcal{H}$ consists of function expansions as defined in (1.13), the regularizer only acts on the parameters $\boldsymbol{\alpha}$ of the model,

$$\mathcal{R}(f) = \mathcal{R}_{\boldsymbol{\alpha}}(\boldsymbol{\alpha}), \tag{1.15}$$

which is often computed as a vector norm of $\boldsymbol{\alpha}$.

Though searching for f within a specific function class $\mathcal{H}$ can be related in some cases to a particular choice of structure for the nonlinear model f, this can also be more general. In particular, by assuming that f is an expansion over some functional basis as (1.13), a single function $f \in \mathcal{H}$ can have multiple representations (and parametrizations) depending on the choice of the basis. In addition, we will see below that $\mathcal{H}$ can be an infinite dimensional function space with the universal approximation capacity while still allowing for learning from a finite set of data. As a practical consequence, arbitrary nonlinearities can be learned without introducing an approximation error due to an arbitrary choice of structure for f.

Reproducing kernel Hilbert space

Historically, the subject of Reproducing kernel Hilbert space was developed by Nachman Aronszajn in 1950 [2] in functional analysis. A reproducing kernel Hilbert space (RKHS) (see e.g. [37] for an overview) is a Hilbert space of functions $(\mathcal{H}, \langle \cdot, \cdot \rangle_{\mathcal{H}})$ which admits a real-valued positive definite function κ on $\mathcal{X}^2$ with the reproducing property: $\forall \boldsymbol{x} \in \mathcal{X}$, $\forall f \in \mathcal{H}$, $\langle f, \kappa(\boldsymbol{x}, \cdot) \rangle_{\mathcal{H}} = f(\boldsymbol{x})$, and in particular,

$$\langle \kappa(\boldsymbol{x}, \cdot), \kappa(\boldsymbol{x}', \cdot) \rangle_{\mathcal{H}} = \kappa(\boldsymbol{x}, \boldsymbol{x}').$$

κ is the reproducing kernel of $\mathcal{H}$ and the class of functions $\mathcal{H}$ can be written as

$$\mathcal{H} = \left\{ f : f = \sum_{i=1}^{\infty} \alpha_i \kappa(\boldsymbol{x}_i, \cdot),\ \alpha_i \in \mathbb{R}, \boldsymbol{x}_i \in \mathcal{X}, \|f\|_{\mathcal{H}} < +\infty \right\} \tag{1.16}$$

with the norm in $\mathcal{H}$ induced by the inner product and defined as

$$\|f\|_{\mathcal{H}}^2 = \langle f, f \rangle_{\mathcal{H}} = \sum_{i=1}^{\infty} \sum_{j=1}^{\infty} \alpha_i \alpha_j \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j). \tag{1.17}$$

Typical examples of kernel functions include the Gaussian Radial Basis Function (Gaussian RBF) kernel, $\kappa(\boldsymbol{x}, \boldsymbol{x}') = \exp(-\|\boldsymbol{x} - \boldsymbol{x}'\|_2^2/2\sigma^2)$, and the polynomial kernel, $\kappa(\boldsymbol{x}, \boldsymbol{x}') = (\boldsymbol{x}^\top \boldsymbol{x}' + 1)^\gamma$.

When learning in an RKHS, a natural choice for $\mathcal{R}(f)$ is based on the RKHS norm:

$$\mathcal{R}(f) = \frac{1}{2}\|f\|_{\mathcal{H}}^2. \tag{1.18}$$

In addition, with (1.18), the generalized representer theorem in [79] provides an explicit structure for the solution to (1.14). This theorem is recalled below, where $\mathcal{D}_x$ denotes the set of all points $\boldsymbol{x}_k$ in the training set $\mathcal{D}$.

Theorem 1 (Generalized Representer Theorem, [79])**.** *The solution f^* to* (1.14)*, with $\mathcal{H}$ defined as in* (1.16)*, $\mathcal{R}(f) = g(\|f\|_{\mathcal{H}})$ and a monotonically increasing function $g : \mathbb{R}^+ \to \mathbb{R}^+$, is a kernel expansion over the training set, i.e., f^* is in the span of $\{\kappa(\boldsymbol{x}_k, \cdot) : \boldsymbol{x}_k \in \mathcal{D}_x\}$.*

This result shows that minimizing any regularized functional of the form (1.14) over an RKHS leads to a finite linear combination of kernel functions computed at the training points:

$$f(\boldsymbol{x}) = \sum_{k=1}^{N} \alpha_k \kappa(\boldsymbol{x}_k, \boldsymbol{x}). \tag{1.19}$$

Note that a semiparametric version of Theorem 1 is also provided in [79] to allow for a bias term in the model. This is obtained by considering a model $\tilde{f} = f + b$, with $f \in \mathcal{H}$ and $b \in \mathbb{R}$, regularized only in f.

1.3.3 Hybrid system identification

The choice of model classes for hybrid systems is more complicated than in the classical cases of linear and nonlinear systems. The model can be a combination of various submodel types. Moreover, the switching between subsystems causes a mixed data set which cannot be used directly to identify each subsystem separately. Therefore, if the switching sequence is unknown, the identification of hybrid systems becomes an NP-hard problem [73] which comprises simultaneously the classification task, i.e., grouping the data points into subsets corresponding to the modes, and the submodel estimation task.

We now formalize the hybrid system identification problems via the ARX form for different cases. For switched systems, the identification problem can be seen as follows.

Problem 2. *[73] Given a data set $\mathcal{D}_0$ of N input-output pairs (u_k, y_k), $k = 1, ..., N$, generated by a switched system, estimate the model orders n_a and n_b, the number of submodels s, the submodels $\{f_j\}_{j=1}^s$ and the discrete state q_k for each input-output pair.*

Commonly, with some prior knowledge about the system, the model orders are fixed. Then Problem 2 becomes simpler as follows.

Problem 3. *Given a data set $\mathcal{D} = \{(\boldsymbol{x}_k, y_k)\}_{k=1}^N$ generated by a switched system, estimate the number of submodels s, the submodels $\{f_j\}_{j=1}^s$ and the switching sequence $\{q_k\}_{k=1}^N$.*

For linear switched systems with model form (1.3), we have the following problem.

Problem 4. *Given a data set $\mathcal{D} = \{(\boldsymbol{x}_k, y_k)\}_{k=1}^N$ generated by a linear switched system, estimate the number of submodels s, the submodel parameter vectors $\{\boldsymbol{\theta}_j\}_{j=1}^s$ and the switching sequence $\{q_k\}_{k=1}^N$.*

When we consider piecewise smooth systems with knowledge about model orders, the identification problem is reformulated as follows.

Problem 5. *Given a data set* $\mathcal{D} = \{(\boldsymbol{x}_k, y_k)\}_{k=1}^N$ *generated by a piecewise smooth system, estimate the number of submodels* s*, the submodels* $\{f_j\}_{j=1}^s$ *and the regions* $\Re_j$*,* $j = 1, \dots, s$*, or the switching boundaries in the regression space so that the discrete state* $q_k = j$ *if* $\boldsymbol{x}_k \in \Re_j$*.*

The existing appoaches for these problems are presented in Chapter 2. Some new approaches to Problem 3 are proposed in the next chapters. They can be extended to Problem 5 by adding a step for estimating the regions $\Re_j$ as described in Section 2.3. On the contrary, Chapter 6 will deal directly with Problem 5.

1.3.4 Model validation

Validating the obtained model is the important final step of Procedure 1. The model needs to satisfy the specific criteria for validation before it is employed as a basis for further purposes such as prediction, controller design or fault detection. For each model type, one or all of the following criteria are used in this step.

For linear or nonlinear parametric models

In the estimation theory framework, the obtained linear or parametric models are typically evaluated with the Normalized Parametric Error (NPE), defined as

$$NPE = \frac{\|\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}\|_2}{\|\boldsymbol{\theta}\|_2}, \tag{1.20}$$

where $\boldsymbol{\theta}$ and $\hat{\boldsymbol{\theta}}$ are the true and estimated parameter vectors.

For non-parametric models

For non-parametric models in the context of learning theory, the dimension of the parameter vector is not fixed and can differ from the one of the true parameter vector of the system. In addition, the parameters are meaningless and measuring the parametric error as for parametric models is irrelevant. Thus, we cannot use NPE to evaluate the model obtained.

In the statistical learning framework, given a pair of random variables $(\boldsymbol{X}, Y) \in \mathcal{X} \times \mathcal{Y}$ of unknown probability distribution P, we try to find a model $f(\boldsymbol{X})$ minimizing the expected risk

$$R(f) = E_{\boldsymbol{X},Y}[\ell(Y, f(\boldsymbol{X}))] = \int_{\mathcal{X}\times\mathcal{Y}} \ell(y, f(\boldsymbol{x})) p_{\boldsymbol{X},Y}(\boldsymbol{x}, y) d\boldsymbol{x} dy,$$

where ℓ is a loss function (1.7) and $p_{\boldsymbol{X},Y}(\boldsymbol{x}, y)$ is the probability density function. Since P is unknown, it is impossible to calculate this risk. Therefore, the model quality is measured by an estimate of the risk defined as an average error on a test set $\mathcal{D}_t = \{(\boldsymbol{x}_k, y_k)\}_{k=1}^{N_t}$ which is independent of the training data set $\mathcal{D}$ and contains data identically sampled from P,

$$Test\ error = \frac{1}{N_t} \sum_{k=1}^{N_t} \ell\left(y_k, f(\boldsymbol{x}_k)\right).$$

This estimate converges to the expected risk $R(f)$ when the number of test samples goes to infinity, i.e., $Test\ error \overset{N_t \to \infty}{\longrightarrow} R(f)$. With the squared loss (1.9), this yields to the common Mean Squared Error (MSE) criterion:

$$MSE = \frac{1}{N_t} \sum_{k=1}^{N_t} \left(f(\boldsymbol{x}_k) - y_k \right)^2 . \tag{1.21}$$

Another error criterion used in practice is FIT which is calculated with normalized errors as

$$FIT = \frac{1}{N_t} \sum_{k=1}^{N_t} \frac{\left(f(\boldsymbol{x}_k) - y_k \right)^2}{\left(y_k - \bar{y} \right)^2} \tag{1.22}$$

where $\bar{y}$ is the mean of the output test data, i.e., $\bar{y} = \frac{1}{N_t} \sum_{k=1}^{N_t} y_k$.

Note that the validation criteria for non-parametric models can also be applied to parametric models.

For hybrid models

For hybrid system identification, if submodels are linear or parametric, the criterion NPE (1.20) can be used with, instead of $\boldsymbol{\theta}$, the parameter matrix $\boldsymbol{\Theta} = [\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_s]$, which consists of submodel parameter vectors as columns. In this case, the matrix Frobenius norm replaces the ℓ_2-norm, i.e.,

$$NPE = \frac{\|\hat{\boldsymbol{\Theta}} - \boldsymbol{\Theta}\|_F}{\|\boldsymbol{\Theta}\|_F}. \tag{1.23}$$

In other cases, we can use MSE (1.21) and FIT (1.22).

Another important criterion to evaluate hybrid models is the Classification Error rate (CE),

$$CE = \frac{1}{N} \sum_{k=1}^{N} \mathbb{I}[\hat{q}_k \neq q_k] \tag{1.24}$$

where

$$\mathbb{I}[\hat{q}_k \neq q_k] = \begin{cases} 1 & \text{if } \hat{q}_k \neq q_k \\ 0 & \text{if } \hat{q}_k = q_k, \end{cases} \tag{1.25}$$

which measures the quality of the estimation of the switching sequence $\{q_k\}_{k=1}^{N}$. This criterion allows us to analyze the ability of the identification methods to distinguish between the modes.

Model selection

The quality of the model obtained by the estimation/learning step is investigated via the above criteria. Firstly, it is checked on the training data. If the result is not satisfactory, the model certainly cannot be accepted. We go back to step 2 of Procedure 1.

If the model provides a good result on the training data, it still needs to pass a test on the test data set. This step aims at ensuring a good operation of the model in working conditions. The test data set should excite the system at these conditions. Again, when the test result is not acceptable, the model is not validated and we go back to step 2 of Procedure 1.

In step 2 of Procedure 1, we can tune the hyperparameters of the models or the identification algorithm. These parameters are often chosen by the user. This can sometimes add difficulties if these parameters are sensitive, dependent or impossible to be interpreted at high level. Instead of hand tuning, a procedure is used with a validation data set which is independent of the training and test data sets. The best choice of hyperparameters is the one that leads to the best criteria on the validation data set.

If the validation data set is not available, we can use the k-fold cross validation. This technique divides the training data set into k parts. Each part is used once as a validation set while the other $k-1$ parts are used as training set. The error average over k different validation sets provides an estimate of the expected risk that can be used to tune the hyperparameters.

Finally, if no adjustment of the hyperparameters yields a good model on the test data set, the problem can be that either information for model estimation is missing or the model class is not flexible enough.

1.4 Sparsity

The term "sparsity" is used to describe something scattered. More formally, a sparse vector is a vector whose entries are mostly zeros and very few are non-zeros. This term appears in areas of applied mathematics, computer science, and signal processing, including Compressive Sensing [30, 41, 33, 23]. The sparsity level of a vector is measured by its ℓ_0 pseudo-norm which is a function counting the number of non-zero entries, defined as:

$$\|\boldsymbol{a}\|_0 = |\{i : a_i \neq 0\}|. \tag{1.26}$$

1.4.1 Sparse models

A model is sparse if its parameter vector is sparse. Getting sparse models is attractive because of the following advantages.

- In linear or parametric estimation, searching for a sparse model amounts to selecting variables since the obtained model does not depend on the variables corresponding to zero parameters.
- Non-parametric models with a parameter vector of high dimension may have enough flexibility to approach any system behavior. Thus, they lead to overfitting in case of noisy data. In this respect, forcing a model to be sparse reduces its flexibility, which limits the overfitting.
- A sparse model needs less memory and it is more computationally efficient for calculating $f(\boldsymbol{x})$.

1.4.2 Obtaining sparsity

Obtaining sparsity in a problem corresponds to finding a solution with many zero elements. This is usually done by the following approaches.

- Thank to a preprocessing step, we can select important data or variables to obtain the sparsity. All parameters which are not linked with them, are set to zero before searching for the solution.
- Getting a sparse solution simultaneously in the learning/estimation process via an optimization program. Consider the generic learning problem written as

$$\min_{\boldsymbol{a}\in\mathcal{C}} \|\boldsymbol{a}\|_0 + \mathcal{J}(\boldsymbol{a}), \tag{1.27}$$

where $\mathcal{J}(\boldsymbol{a})$ is a data fitting term and $\mathcal{C}$ is the set of feasible points for the problem with constraints. This is in general an NP-hard problem [24] because of its combinatorial nature in which all possibilities for the selection of non-zero a_i must be verified to find an optimal solution. The current trend to solve such a problem is to consider a convex relaxation of (1.27) which can be efficiently optimized, for instance with methods described in [19]. In particular, the ℓ_1 convex relaxation of the zero pseudo-norm [34] can be used as follows

$$\min_{\boldsymbol{a}\in\mathcal{C}} \|\boldsymbol{a}\|_1 + \mathcal{J}(\boldsymbol{a}). \tag{1.28}$$

A particular case of this generic problem where $\mathcal{J}(\boldsymbol{a}) = 0$ and $\mathcal{C}$ is the set of solutions of a linear system of equations has been considered in many applications of compressive sensing, signal and image processing, see, e.g., [25]. To improve the sparsity of the solution, the iteratively reweighted method [25] is usually employed:

$$\min_{\boldsymbol{a}\in\mathcal{C}} \|\boldsymbol{W}^{(i)}\boldsymbol{a}\|_1 = \sum_k \left| w_{kk}^{(i)} a_k \right|, \tag{1.29}$$

where $\boldsymbol{W}^{(i)}$ is a weighting diagonal matrix with elements $w_{kk}^{(i)} > 0$ which are updated at each iteration i by

$$w_{kk}^{(i+1)} = (|a_k^{(i)}| + \epsilon)^{-1}, \tag{1.30}$$

where $a_k^{(i)}$ is the optimal solution of (1.29) at the i^{th} iteration, with $w_{kk}^{(0)} = 1$, and ϵ is a small constant.

1.5 Conclusions

In this chapter, hybrid dynamical systems were introduced with their identification problems. The concepts related to sparsity, specifically the iteratively reweighted technique [25], were also introduced. These will be used throughout the thesis and more particularly in Chapter 4. Nonlinear regression techniques borrowed from machine learning and the concepts of overfitting and regularization were reviewed. They will serve for the identification of nonlinear hybrid systems in Chapter 5 and for the piecewise smooth regression method proposed in Chapter 6.

In the next chapter, we review the existing methods for the identification of linear hybrid systems. This problem will be the main subject of Chapter 3.

Chapter 2

A state of the art

ABSTRACT. *The main methods in the literature for hybrid system identification, particularly for linear hybrid systems, are reviewed in this chapter. After the presentation of the different points of view (algebraic, bounded-error, clustering, sparsity and convex relaxation, and time order), the methods are considered for switched systems and for piecewise systems. For each type of systems, the methods are classified with respect to the assumption on the number of modes, which can be fixed or not.*

CHAPTER 1 introduced the models and problems associated to the identification of hybrid systems from data. The identification of hybrid systems has attracted researchers' attention since the last two decades with different approaches. In this chapter, we review various identification methods for *linear* hybrid systems based on the input-output model (1.3):

$$y_k = f_{q_k}(\boldsymbol{x}_k) + v_k = \boldsymbol{\theta}_{q_k}^\top \boldsymbol{x}_k + v_k, \tag{2.1}$$

where $y_k \in \mathbb{R}$ is the output at discrete time k, $\boldsymbol{x}_k = [y_{k-1}, \dots, y_{k-n_a},\ u_{k-1}, \dots, u_{k-n_b}]^\top \in \mathbb{R}^d$ is the regression vector of dimension $d = n_a + n_b$ with the past outputs y_{k-i} and inputs u_{k-i} and the model orders n_a and n_b, $q_k \in \{1, \dots, s\}$ is the discrete state or mode, s is the number of submodels, $\boldsymbol{\theta}_j \in \mathbb{R}^d$, $j = 1, \dots, s$, are the parameter vectors defining each submodel and $v_k \in \mathbb{R}$ is a noise term. In the case of affine subsystems, the regression vector $\boldsymbol{x}_k$ is simply replaced by $\tilde{\boldsymbol{x}}_k = [\boldsymbol{x}_k^\top, 1]^\top$. In most of the methods, n_a and n_b are supposed to be known and Problem 4 of Chapter 1 is considered:

Problem 4. *Given a data set $\mathcal{D} = \{(\boldsymbol{x}_k, y_k)\}_{k=1}^N$ generated by a linear switched system, estimate the number of submodels s, the submodel parameter vectors $\boldsymbol{\theta}_j$, $j = 1, \dots, s$, and the switching sequence $\{q_k\}_{k=1}^N$.*

This chapter continues with the main points of view for Problem 4 in Sect. 2.1. These points of view have been considered in many studies over the two recent decades. The methods are classified in two groups: for SARX systems in Sect. 2.2 and PWARX systems in Sect. 2.3. We further distinguish between methods that work with a free or fixed number of modes.

2.1 Points of view

Problem 4 is an NP-hard problem in general [73]. Indeed, the number of submodels s, the submodel parameter vectors $\boldsymbol{\theta}_j$, $j = 1, \dots, s$, and the switching sequence $\{q_k\}_{k=1}^N$ mutually influence the estimation of each others. Thus, we must simultaneously identify them. This makes the identification of hybrid systems much more difficult than that of "classical" linear or nonlinear systems. From the early 2000s, several approaches which reduce the difficulty of Problem 4 with some assumptions have been proposed: algebraic, bounded-error, clustering, sparsity and convex relaxation, and time order based approaches. They are based on different aspects of the problem which we review here.

By fixing the number of modes, Problem 4 can be naturally considered as a mixed integer optimization problem which is inferred from the classical prediction error approach [61] in the context of hybrid systems:

$$\begin{aligned} \min_{\{\boldsymbol{\theta}_j\},\{\beta_{jk}\}} \ & \frac{1}{N}\sum_{k=1}^{N}\sum_{j=1}^{s}\beta_{jk}\ell(y_k,\boldsymbol{\theta}_j^\top\boldsymbol{x}_k) \\ \text{s.t.} \ & \beta_{jk}\in\{0,1\},\quad k=1,\dots,N,\ j=1,\dots,s, \\ & \sum_{j=1}^{s}\beta_{jk}=1,\ k=1,\dots,N, \end{aligned} \tag{2.2}$$

where $\beta_{jk}=1$ if and only if the point $\boldsymbol{x}_k$ is assigned to mode j and ℓ is a loss function defined by (1.7). The switching sequence $\{q_k\}_{k=1}^N$ is reconstructed as follows:

$$q_k = j \iff \beta_{jk}=1,\quad k=1,\dots,N.$$

In the optimization program (2.2), all submodels and the switching sequence are estimated at the same time. However, this program is computationally intractable with even moderate data sets. Thus, some works tried to reformulate it or to find a suboptimal solution for Problem 4 from different viewpoints or with different assumptions.

Algebraic approach. The main idea of the algebraic approach [103] is to employ the so-called Hybrid Decoupling Constraint (HDC), a relation which always holds whatever the switching sequence is. Under the assumptions of known number of modes s and noiseless data, the HDC is expressed as

$$\forall k,\ \prod_{j=1}^{s}\left(\boldsymbol{\theta}_j^\top\boldsymbol{x}_k - y_k\right)=0. \tag{2.3}$$

These polynomial equations decouple the two subproblems of estimating the parameters $\boldsymbol{\theta}_j$ and the switching sequence $\{q_k\}$. In particular, solving these equations for $\boldsymbol{\theta}_j$ allows one to recover the system parameters without having to estimate the mode q_k.

Clustering approach. The main principle of the clustering procedure [39] for PWARX systems is to classify the training data before estimating the parameter vectors. If the data points are well classified, the problem of estimating submodels becomes easy. Under the assumption that a PWARX system is locally linear, neighboring regions share the same parameter vector. From this property, if data points are represented by similar local parameter vectors, they are grouped together. Then each submodel is estimated from the data in one of the groups.

In this approach, the subproblem of estimating the switching sequence $\{q_k\}$ is treated as an unsupervised data classification problem.

Bounded-error approach. The main idea of the bounded-error approach [12] is to obtain a hybrid system model having a predefined maximal error on the data. Therefore, it tries to find a *minimum* number of submodels such that for each data point, at least one of those submodels satisfies a given error threshold δ, i.e.,

$$\forall k, \exists j \in \{1, \ldots, \hat{s}\}, \; |y_k - \hat{\boldsymbol{\theta}}_j^\top \boldsymbol{x}_k| \leq \delta, \tag{2.4}$$

where $\hat{\boldsymbol{\theta}}_j$ is the estimated parameter vector of a submodel. This is a change of perspective compared with most of the other methods which minimize the error for a given number of modes.

Sparsity and convex relaxation approach [5, 72, 69]**.** In the context of hybrid system identification, a submodel can be estimated such that it produces a sparse vector where the zero entries correspond to data points associated with that submodel and the nonzero entries to data points of the others. This principle leads to minimize the ℓ_0-norm of this vector which is the number of its nonzero entries. Since minimizing an ℓ_0-norm is NP-hard in general, as described in Sect. 1.4.2, an ℓ_1-norm convex relaxation is usually used.

Time order based approach. This approach [72, 70] exploits the time information of the collected data points to detect the switchings between modes. If there is no switching in a given time period, the submodel does not vary. Therefore, this approach finds a submodel that fits the data points while limiting the variation of this submodel for as long as possible and only uses a new submodel when the variation is too high.

From the main ideas presented above, the different approaches for the identification of linear hybrid systems are presented now in more details for SARX systems before focusing on PWARX systems.

2.2 SARX system identification

SARX systems form a basic class of hybrid systems. As shown in Chapter 1, investigations of this class can be easily extended to other classes (see Sect. 1.2). Therefore, there are many works focusing on this class, which are reviewed below by distinguishing between methods that fix the number of modes and those that fix the model error instead.

2.2.1 Fixed number of modes

To reduce the difficulty of the NP-hard Problem 4, a class of methods fixes a priori the number of modes and estimates the parameter vectors and the switching sequence to minimize the prediction error of the model. Therefore, the number of modes is either known from prior knowledge or tuned as a hyperparameter to build a suitable model.

Algebraic methods (ALG)

The algebraic approach [103] was proposed by Vidal *et al.* in 2003 with the assumptions of noiseless data and fixed number of modes. From that to now, as a main trend, a family of methods is developed based on this approach. The main principle is to employ the Hybrid Decoupling Constraint (HDC) (2.3). This HDC can be rewritten as

$$\forall k, \; \prod_{j=1}^{s} \left(\boldsymbol{\beta}_j^\top \boldsymbol{z}_k \right) = 0, \tag{2.5}$$

where $\boldsymbol{\beta}_j = [1\ \boldsymbol{\theta}_j^\top]^\top \in \mathbb{R}^{d+1}$ and $\boldsymbol{z}_k = [-y_k\ \boldsymbol{x}_k^\top]^\top \in \mathbb{R}^{d+1}$. Thus we have a set of N equations corresponding to the N data points in order to determine the model parameter vectors $\boldsymbol{\beta}_j$.

By considering the elements of the vector $\boldsymbol{z} = [z_1, \dots, z_{d+1}]^\top$ as variables, the HDC is rewritten in the form of an homogeneous polynomial of $d+1$ variables with total degree s,

$$P_s(\boldsymbol{z}) = \prod_{j=1}^{s} \left(\boldsymbol{\beta}_j^\top \boldsymbol{z}\right) = \boldsymbol{h}^\top \boldsymbol{\nu}_s(\boldsymbol{z}) = 0, \tag{2.6}$$

where $\boldsymbol{h}$ is the coefficient vector of the polynomial and $\boldsymbol{\nu}_s(\boldsymbol{z})$ is the Veronese map of degree s which is a vector including all possible monomials of total degree s, *i.e.*, $z_1^{n_1}...z_i^{n_i}...z_{d+1}^{n_{d+1}}$ with $0 \le n_i \le s$, $i = 1, \dots, d+1$, and $\sum_i n_i = s$. The total number of independent monomials is $M_s(d+1) = \binom{s+d}{s}$.

For example, in a case where $s = 2$ and $d = 2$, we have

$$\begin{aligned} P_2(\boldsymbol{z}) = \prod_{j=1}^{2} \left(\boldsymbol{\beta}_j^\top \boldsymbol{z}\right) = \left(\boldsymbol{\beta}_1^\top \boldsymbol{z}\right)\left(\boldsymbol{\beta}_2^\top \boldsymbol{z}\right) &= (\beta_{11}z_1 + \beta_{12}z_2 + \beta_{13}z_3)(\beta_{21}z_1 + \beta_{22}z_2 + \beta_{23}z_3) \\ &= \beta_{11}\beta_{21}z_1^2 + (\beta_{11}\beta_{22} + \beta_{12}\beta_{21})z_1z_2 + (\beta_{11}\beta_{23} + \beta_{13}\beta_{21})z_1z_3 \\ &\quad + \beta_{12}\beta_{22}z_2^2 + (\beta_{12}\beta_{23} + \beta_{13}\beta_{22})z_2z_3 + \beta_{13}\beta_{23}z_3^2. \end{aligned}$$

Then,

$$\begin{aligned} &\boldsymbol{\nu}_2(\boldsymbol{z}) = [z_1^2,\ z_1z_2,\ z_1z_3,\ z_2^2,\ z_2z_3,\ z_3^2]^\top, \\ &\boldsymbol{h} = [\beta_{11}\beta_{21},\ (\beta_{11}\beta_{22} + \beta_{12}\beta_{21}),\ (\beta_{11}\beta_{23} + \beta_{13}\beta_{21}),\ \beta_{12}\beta_{22},\ (\beta_{12}\beta_{23} + \beta_{13}\beta_{22}),\ \beta_{13}\beta_{23}]^\top \in \mathbb{R}^6. \end{aligned}$$

The vector $\boldsymbol{h} \in \mathbb{R}^{M_s(d+1)}$ is determined by solving the linear system of equations

$$\begin{bmatrix} \boldsymbol{\nu}_s(\boldsymbol{z}_1)^\top \\ \boldsymbol{\nu}_s(\boldsymbol{z}_2)^\top \\ \vdots \\ \boldsymbol{\nu}_s(\boldsymbol{z}_N)^\top \end{bmatrix} \boldsymbol{h} = \boldsymbol{0}. \tag{2.7}$$

Finally, the model parameter vectors $\boldsymbol{\beta}_j$ can be calculated from the partial derivative of $P_s(\boldsymbol{z})$ with respect to $\boldsymbol{z}$,

$$DP_s(\boldsymbol{z}) = \frac{\partial P_s(\boldsymbol{z})}{\partial \boldsymbol{z}} = \sum_{j=1}^{s} \prod_{l \neq j} \left(\boldsymbol{\beta}_l^\top \boldsymbol{z}\right) \boldsymbol{\beta}_j,$$

as follows:

$$\boldsymbol{\beta}_j = \left.\frac{DP_s(\boldsymbol{z})}{\boldsymbol{a}_d^\top DP_s(\boldsymbol{z})}\right|_{\boldsymbol{z}=\boldsymbol{z}_j^*}, \tag{2.8}$$

where $\boldsymbol{a}_d = [1, 0, \dots, 0]^\top \in \mathbb{R}^{d+1}$ (recall that $\beta_{j1} = 1$) and $\boldsymbol{z}_j^*$ is a point belonging to the mode j.

For example, in the case where $s = 2$, we have

$$DP_2(\boldsymbol{z}) = (\boldsymbol{\beta}_2^\top \boldsymbol{z})\boldsymbol{\beta}_1 + (\boldsymbol{\beta}_1^\top \boldsymbol{z})\boldsymbol{\beta}_2.$$

Then, for a point $\boldsymbol{z}_1^*$ of the 1st mode, $\boldsymbol{\beta}_1^\top \boldsymbol{z}_1^* = 0$ and

$$\boldsymbol{\beta}_1 = \frac{DP_2(\boldsymbol{z}_1^*)}{\boldsymbol{a}_d^\top DP_2(\boldsymbol{z}_1^*)} = \frac{(\boldsymbol{\beta}_2^\top \boldsymbol{z}_1^*)\boldsymbol{\beta}_1}{\boldsymbol{\beta}_2^\top \boldsymbol{z}_1^*}.$$

To determine s parameter vectors by (2.8), one needs s points $\boldsymbol{z}_j^*$ of s modes. In [103] the s points are chosen at the intersections of a line and the hyperplanes $\{\boldsymbol{z} : \boldsymbol{\beta}_j^\top \boldsymbol{z} = 0\}$, $j = 1, \dots, s$. In other words, these are roots of $P_s(\boldsymbol{z}) = 0$ in the form $\boldsymbol{z} = \boldsymbol{z}_0 + \mu \boldsymbol{v}$ where $\boldsymbol{z}_0$ and $\boldsymbol{v}$ are predefined and $\mu \in \mathbb{R}$ is a variable to be determined. Thus, this leads to a polynomial equation of degree s in μ. To ensure that this equation has s real roots, it is necessary that $\boldsymbol{z}_0 \neq 0$, $\boldsymbol{v}$ is linearly independent of $\boldsymbol{z}_0$ and $P_s(\boldsymbol{v}) \neq 0$.

The switching sequence $\{q_k\}_{k=1}^N$ is estimated by

$$\hat{q}_k = \arg \min_{j=1,\dots,s} \left(y_k - \boldsymbol{\theta}_j^\top \boldsymbol{x}_k \right)^2 = \arg \min_{j=1,\dots,s} \left(\boldsymbol{\beta}_j^\top \boldsymbol{z}_k \right)^2 \tag{2.9}$$

As previously mentioned, the above procedure is only proposed under the assumptions of known number and orders of subsystems. The work [98] investigated the method for cases of unknown model orders and known number of modes and proposed a method to solve the problem for a number of modes greater than 4 which is a limitation of [103] due to the polynomial equation of degree s solved to find a point $\boldsymbol{z}_j^*$ of each mode. Then, the work [63] completed the algebraic approach for both unknown model orders and unknown number of modes. Finally, the work [10] extended the algebraic approach to MIMO SARX systems.

The algebraic approach provides the exact solution in the noiseless case where the HDC (2.3) is valid. The algorithm complexity is $\mathcal{O}\left((M_s(d+1))^3\right)$ which is the computational complexity of solving the linear system (2.7) of $M_s(d+1)$ linearly independent equations. It depends only on the number of modes s and of regressors d and does not depend on the number of data points N. Moreover, there is no optimization problem to solve in this method.

In the noisy case, this approach is sensitive to noise and can provide poor results [51]. This sensitivity is improved with moments-based convex optimization in [71].

Inspired by the recursive parameter estimation for a single ARX system [61] (only one mode), an exponentially convergent recursive algebraic procedure to identify SARX systems is proposed in [101, 45, 99]. The recursive algorithm is applied to the hybrid decoupling constraint HDC (2.6) to find $\boldsymbol{h}$. We briefly recall the recursive identification procedure for an ARX system in the input-output form

$$y_k = \boldsymbol{\theta}^\top \boldsymbol{x}_k, \tag{2.10}$$

where $\boldsymbol{x}_k \in \mathbb{R}^d$ is the regression vector at the time k and $\boldsymbol{\theta} \in \mathbb{R}^d$ is the parameter vector. It is easy to write (2.10) as

$$\boldsymbol{\beta}^\top \boldsymbol{z}_k = 0, \tag{2.11}$$

with $\boldsymbol{\beta} = [1, \boldsymbol{\theta}^\top]^\top$ and $\boldsymbol{z}_k = [-y_k, \boldsymbol{x}_k^\top]^\top$.

By applying the equation error identifier in [61] to recursively estimate $\boldsymbol{\beta}$, we have

$$\hat{\boldsymbol{\beta}}_{k+1} = \left(\boldsymbol{I}_{d+1} - \frac{\mu \boldsymbol{\Pi}_{d+1} \boldsymbol{z}_k \boldsymbol{z}_k^\top}{1 + \mu \|\boldsymbol{\Pi}_{d+1} \boldsymbol{z}_k\|_2^2} \right) \hat{\boldsymbol{\beta}}_k, \tag{2.12}$$

where $\hat{\boldsymbol{\beta}}_k$ is the estimate of $\boldsymbol{\beta}$ at the kth iteration, $\mu > 0$ is a tuning parameter and $\boldsymbol{\Pi}_{d+1} = \begin{bmatrix} 0 & \boldsymbol{0}_d^\top \\ \boldsymbol{0}_d & \boldsymbol{I}_d \end{bmatrix}$.

Returning to SARX systems, the homogeneous polynomial of degree s in $d+1$ variables (2.6),

$$P_s(\boldsymbol{z}_k) = \boldsymbol{h}^\top \boldsymbol{\nu}_s(\boldsymbol{z}_k) = 0,$$

is considered instead of (2.11). Thus, by applying the above recursive scheme, a sequence $\{\boldsymbol{h}_k\}$ of the model parameters is determined to converge to $\boldsymbol{h}$. This allows to obtain, via the derivatives of $\boldsymbol{h}^\top \boldsymbol{\nu}_s(\boldsymbol{z})$, the sequence $\{\boldsymbol{\beta}_{jk}\}$ of submodel parameters converging to $\boldsymbol{\beta}_j$.

Continuous optimization (CO)

By reformulating the mixed-integer optimization program (2.2), the authors of [60] proposed a continuous optimization framework for hybrid system identification with two estimators called Minimum-of-Errors (ME) estimator and Product-of-Errors (PE) estimator. They are generally stated as follows.

Minimum-of-errors estimator. An optimal solution of problem (2.2) is a global solution of the optimization program

$$\min_{\{\boldsymbol{\theta}_j\}_{j=1}^s} J^{ME} = \frac{1}{N}\sum_{k=1}^{N}\left(\min_{j=1,\dots,s} \ell\left(y_k, \boldsymbol{\theta}_j^\top \boldsymbol{x}_k\right)\right), \tag{2.13}$$

where ℓ is a loss function (1.7). In this program, the assignment of data points to the best submodel is integrated by $\min\limits_{j=1,\dots,s} \ell(y_k, \boldsymbol{\theta}_j^\top \boldsymbol{x}_k)$.

Product-of-errors estimator. The PE estimator is obtained by solving the smooth optimization program

$$\min_{\{\boldsymbol{\theta}_j\}_{j=1}^s} J^{PE} = \frac{1}{N}\sum_{k=1}^{N}\prod_{j=1}^{s} \ell\left(y_k, \boldsymbol{\theta}_j^\top \boldsymbol{x}_k\right). \tag{2.14}$$

We note that the ME and PE estimators have the same global minimum for noiseless data, i.e., $J^{ME} = J^{PE} = 0$. These estimators are expressed via non-convex optimization programs, with any suitable loss function. Nevertheless, compared with the mixed-integer optimization problem (2.2), since there is no integer variables in the optimization programs, they allow to efficiently deal with large-scale problems while obtaining accurate results and robustness to outliers. There are many solvers for nonlinear optimization programs but they cannot guarantee to obtain a global solution. For example, the Multilevel Coordinate Search (MCS) algorithm [49] is used in [60][1].

The continuous optimization framework requires a hyperparameter, if any, which is the parameter of the chosen loss function to deal with particular noise assumptions (for example δ_v in the Hampel loss function (1.11)). This method needs to solve a nonconvex optimization problem in which the number of variables is sd. The computation time depends on the chosen global optimization solver and is typically exponential in the number of variables. Thus the method becomes slow for systems with many modes or parameters, but remains efficient w.r.t. the number of data.

It is worth noting that the program (2.14) with the squared loss function (1.9) minimizes the same cost function as the algebraic method in the noisy case [63].

In [66], a special case of the program (2.13) with squared loss function (1.9) is found. The author proposed also a meta-heuristic optimization algorithm, so-called particle swarm optimization to avoid being trapped in suboptimal solutions though without any guarantee.

In [56], the same cost function as in (2.14) with the squared loss function (1.9) is considered. In addition, the authors propose a continuous cost function with the harmonic

[1]The software is freely available as Matlab code at `http://www.loria.fr/~lauer/software.html`

mean of the squared errors,

$$J^{HM} = \sum_{k=1}^{N} \frac{1}{\sum_{j=1}^{s} \frac{1}{\ell(y_k, \theta_j^\top \boldsymbol{x}_k)}},$$

where ℓ is the squared loss function (1.9). Both of them are solved with an iterative optimization procedure. After the initialization, the parameter vectors are iteratively refined. Only one parameter vector is considered while other parameter vectors are fixed and the prediction errors of the corresponding submodels play the role of the weights. The procedure stops when the parameter vectors converge, potentially in a local minimum.

Alternating optimization methods (ALT)

In [90, 57], s parameter vectors of ARX submodels are initialized randomly. Then, each data point is classified into the mode with the minimal prediction error, i.e.,

$$\hat{q}_k = \arg \min_{j=1,\dots,s} \left(y_k - \boldsymbol{\theta}_j^\top \boldsymbol{x}_k \right)^2 . \tag{2.15}$$

The s parameter vectors of submodels are updated on the classified subsets. Data classification and parameter refinement are repeated until the parameter vectors converge. This procedure is simple but does not guarantee to obtain a global solution. While [90] extends this procedure for PWARX systems, the random initializations and the estimation of the probability of drawing an initialization leading to a satisfactory solution are carefully investigated in [57].

Only one hyperparameter is required: the number of restarts of the procedure to avoid local minima. These methods use the least squares technique to estimate submodels. Therefore, they have a low computational complexity.

Another procedure based on the recursive least squares technique for linear system identification is proposed in [6]. After the initialization of the submodels, the first data point is classified to a mode as in (2.15). This single data point is used to update the parameter vector of the selected submodel with the recurrence equation (2.12). Then, the procedure continues with the next data point and so on until all data points are used.

Bayesian method

The Bayesian procedure was introduced by Juloski *et al.* in [52]. In this method, the submodel parameter vectors $\boldsymbol{\theta}_j$ and the noise v_k are considered as random variables with probability density functions (pdfs) fixed a priori. At the beginning (iteration $k = 0$), all pdfs of the parameter vectors $p_{\boldsymbol{\theta}_j}(\boldsymbol{\theta}; 0)$ are randomly initialized while the pdf of the noise $p_v(.)$ is given. At each iteration k, $k = 1, \dots, N$, a data point $(\boldsymbol{x}_k, y_k)$ is assigned to the submodel with the highest likelihood, i.e., with the highest probability to generate that data point. Then, the pdf $p_{\boldsymbol{\theta}_j}(\boldsymbol{\theta}; k)$ is updated with that classified point. The recursive algorithm is continued for the whole data set to get the final pdfs. Then, the submodel parameter vectors can be computed from the pdfs and the switching sequence $\{q_k\}_{k=1}^N$ is determined from the highest likelihood.

Algorithm 2 presents the method in details. The integrations in (2.17) and (2.18) are approximated by the particle filtering approach, see, e.g., [3] and the references therein.

The key issue of this method is to set a priori the initial pdfs on which highly depends the quality of the identification result. The computation is complex with the integration approximation by the particle filter which uses a number of particles proportional to the size of the parameter domain.

Algorithm 2 Bayesian procedure

```
Require: The initial pdfs of the submodel parameter vectors $p_{\boldsymbol{\theta}_j}(\boldsymbol{\theta};0)$, $j=1,\dots,s$, and
  a pdf of the noise $p_v(.)$.
  Initialize $k=1$.
  while $k \leq N$ do
    Assign the data pair $(\boldsymbol{x}_k, y_k)$ to the submodel with the highest likelihood
        $$\hat{q}_k = \arg\max_{j=1,\dots,s} p((\boldsymbol{x}_k, y_k)|q_k = j), \quad (2.16)$$
    where $p((\boldsymbol{x}_k, y_k)|q_k = j)$ is the conditional probability of the pair $(\boldsymbol{x}_k, y_k)$ given the
    $j$th submodel, and is evaluated by
        $$p((\boldsymbol{x}_k, y_k)|q_k = j) = \int_\Theta p_v\left([y_k - \boldsymbol{\theta}^\top \boldsymbol{x}_k]\right) p_{\boldsymbol{\theta}_j}(\boldsymbol{\theta}; k-1) d\boldsymbol{\theta}. \quad (2.17)$$
    Update the pdfs $p_{\boldsymbol{\theta}_j}(\boldsymbol{\theta}; k-1)$ as
        $$p_{\boldsymbol{\theta}_j}(\boldsymbol{\theta};k) = p_{\boldsymbol{\theta}_j}(\boldsymbol{\theta};k-1), \quad \forall j \neq \hat{q}_k$$
        $$p_{\boldsymbol{\theta}_j}(\boldsymbol{\theta};k) = \frac{p_v\left([y_k - \boldsymbol{\theta}_j^\top \boldsymbol{x}_k]\right) p_{\boldsymbol{\theta}_j}(\boldsymbol{\theta};k-1)}{\int_\Theta p_v\left([y_k - \boldsymbol{\theta}^\top \boldsymbol{x}_k]\right) p_{\boldsymbol{\theta}_j}(\boldsymbol{\theta};k-1) d\boldsymbol{\theta}}, \quad j = \hat{q}_k \quad (2.18)$$
    $k = k+1$.
  end while
  Estimate the parameters: $\hat{\boldsymbol{\theta}}_j = E_{\boldsymbol{\theta}_j}(\boldsymbol{\theta}) = \int_{\mathbb{R}^d} \boldsymbol{\theta} p_{\boldsymbol{\theta}_j}(\boldsymbol{\theta};N) d\boldsymbol{\theta}$.
  Classify the data set: $S_j = \{(\boldsymbol{x}_k, y_k)|\hat{q}_k = j\}$.
  return $p_{\boldsymbol{\theta}_j}(\boldsymbol{\theta}, N)$, $\hat{\boldsymbol{\theta}}_j$ and $S_j$, $j=1,\dots,s$.
```

2.2.2 Free number of modes

In this section, the number of modes is estimated such that the obtained model satisfies a fixed bound on the error.

Bounded-error method

The main idea of the bounded-error approach [12] is to look for a hybrid model having a predefined maximal error on the data. Two combinatorial problems are proposed to obtain such a model: MAX FS which estimates a parameter vector $\hat{\boldsymbol{\theta}}$ that satisfies a maximal number of the following constraints,

$$|y_k - \hat{\boldsymbol{\theta}}^\top \boldsymbol{x}_k| \leq \delta, \quad k \in \{1,\dots,N\}, \tag{2.19}$$

and MIN PFS which finds a MINimun number $\hat{s}$ of Feasible Subsystems satisfying the condition (2.4),

$$\forall k, \exists j \in \{1,\dots,\hat{s}\}, \ |y_k - \hat{\boldsymbol{\theta}}_j^\top \boldsymbol{x}_k| \leq \delta. \tag{2.20}$$

There are approximate algorithms proposed in [1, 12] to obtain suboptimal solutions for these optimization problems. Nevertheless, the number of submodels is usually found larger than the actual mode number of the system due to various reasons: the threshold δ is unknown and maybe set too small; the noise is unbounded, e.g, Gaussian; the estimation error of submodels leads to a prediction error which is larger than δ, even if the noise is bounded by δ. Moreover, there are many *undecidable* points which belong to more than one submodel. Thus, a refinement step is proposed in [12] to reduce the number of submodels. If two submodels are similar, i.e., the parameter vectors are almost the same, these submodels are merged. With a positive threshold δ_θ, two submodels i and j are merged if

$$\frac{\|\hat{\boldsymbol{\theta}}_i - \hat{\boldsymbol{\theta}}_j\|_2}{\min\{\|\hat{\boldsymbol{\theta}}_i\|_2, \|\hat{\boldsymbol{\theta}}_j\|_2\}} < \delta_\theta. \tag{2.21}$$

On the other hand, if the number of data points of mode j is too small, i.e., less than a small fraction of the data, these points can be outliers. In this case, the corresponding submodel is discarded.

Therefore, the hyperparameters of this method are the error bound δ in (2.19), the threshold δ_θ in (2.21) to merge two similar submodels and the threshold on the number of points to discard a submodel.

Error sparsification method (ES)

The error sparsitifcation method proposed by Bako [5] estimates the submodels one by one, and thus the number of submodels, as in the bounded-error approach. Under the assumption of noiseless data, an error vector $\boldsymbol{e} = [e_1 \dots e_N]^\top$, with entries $e_k = y_k - \boldsymbol{\theta}^\top \boldsymbol{x}_k$, is sparse, i.e., with many zero entries, if $\boldsymbol{\theta}$ is one of the true submodel parameter vectors. Moreover, there exists a parameter vector, corresponding to the mode dominating the data set, which maximizes the number of zero entries of $\boldsymbol{e}$. In other words, it is the solution to the following sparse optimization problem,

$$\begin{aligned} \min_{\boldsymbol{e},\boldsymbol{\theta}} \ & \|\boldsymbol{e}\|_0, \\ \text{s.t. } & \boldsymbol{e} = \boldsymbol{y} - \boldsymbol{X}^\top \boldsymbol{\theta}, \end{aligned} \tag{2.22}$$

where $\|\boldsymbol{e}\|_0 = |\{k : e_k \neq 0\}|$ denotes the ℓ_0-norm of $\boldsymbol{e}$, that is the number of nonzero entries of $\boldsymbol{e}$, $\boldsymbol{y} = [y_1 \dots y_N]^\top$ and $\boldsymbol{X} = [\boldsymbol{x}_1 \dots \boldsymbol{x}_N]$. After finding the first submodel, all data points corresponding to $e_k = 0$ are removed from the data set to estimate the next submodel. This procedure is repeated until the data set is empty.

The following theorem gives a necessary condition for the uniqueness of the solution to (2.22).

Theorem 2. *(Theorem 8 in [5]) If there is a vector $\boldsymbol{\theta}$ satisfying*

$$\|\boldsymbol{e}(\boldsymbol{\theta})\|_0 \leq \frac{N - \nu_d(\boldsymbol{X})}{2} \tag{2.23}$$

where $\nu_d(\boldsymbol{X})$ is the minimum integer m such that any $d \times m$ submatrix of $\boldsymbol{X}$ has rank d, then $\boldsymbol{\theta}$ is necessarily the unique vector that achieves the sparsest possible error $\boldsymbol{e}(\boldsymbol{\theta})$.

The problem (2.22) is a hard non-convex optimization problem in general [5]. A convex relaxation of the optimization program (2.22) using the ℓ_1-norm is proposed,

$$\begin{aligned} \min_{\boldsymbol{e},\boldsymbol{\theta}} \ & \|\boldsymbol{W_x e}\|_1 \\ \text{s.t. } & \boldsymbol{Pe} = \boldsymbol{Py}, \\ & \boldsymbol{X}^\top \boldsymbol{\theta} + \boldsymbol{e} = \boldsymbol{y} \end{aligned} \tag{2.24}$$

with the projection matix $\boldsymbol{P} = \boldsymbol{I}_N - \boldsymbol{X}^\top(\boldsymbol{X}\boldsymbol{X}^\top)^{-1}\boldsymbol{X}$ and a weighting matrix $\boldsymbol{W_x} = \operatorname{diag}(\|\boldsymbol{P}_1\|_2, \dots, \|\boldsymbol{P}_k\|_2, \dots, \|\boldsymbol{P}_N\|_2)$, where $\boldsymbol{P}_k$ refers to the kth column of $\boldsymbol{P}$.

The quality of the relaxation (2.24) is analyzed by way of the following theorem.

Theorem 3. *(Theorem 10 in [5]) If there is a vector $\boldsymbol{\theta}$ achieving an error $\boldsymbol{e}(\boldsymbol{\theta})$ such that*

$$\|\boldsymbol{e}(\boldsymbol{\theta})\|_0 \leq \frac{1}{2}\left(1 + \frac{1}{m(\boldsymbol{X})}\right) \tag{2.25}$$

with

$$m(\boldsymbol{X}) = \max_{1 \leq i,k \leq N} \frac{|M_{ik}|}{\sqrt{(1 - M_{ii})(1 - M_{kk})}}, \tag{2.26}$$

where M_{ik} is the (i,k)-entry of $\boldsymbol{M} = \boldsymbol{X}^\top(\boldsymbol{X}\boldsymbol{X}^\top)^{-1}\boldsymbol{X}$, then $\boldsymbol{\theta}$ is the unique solution to (2.24).

The equivalence between the relaxed (2.24) and the original (2.22) formulations is ensured by Theorem 3. Indeed, if (2.25) holds, then (2.23) holds and $\boldsymbol{\theta}$ in Theorem 3 corresponds to the solution of (2.22) via Theorem 2.

The sparsity of $\boldsymbol{e}$ can be improved by using the reweighted technique of Candès *et al.* [25] which was summarized in Sect. 1.4.2. This finally leads to

$$\min_{\boldsymbol{\theta}} \ \|\boldsymbol{W}^{(i)}(\boldsymbol{y} - \boldsymbol{X}^\top\boldsymbol{\theta})\|_1. \tag{2.27}$$

To deal with the noise, an error threshold is added in the algorithm to decide which points belong to the estimated submodel and must be removed from the data set: as in the bounded-error approach, the condition $|e_k| \leq \delta$ is used instead of $e_k = 0$.

In this procedure, the main tuning parameters are the error threshold δ and the parameter ϵ of the iteratively reweighted algorithm presented in Sect. 1.4.2. This method requires solving convex optimization problems to estimate submodels and thus avoids local minima. But, from Theorem 3, the convex relaxation requires a condition on the fraction, $\frac{N - \|\boldsymbol{e}\|_0}{N}$, of data generated by each mode:

$$\frac{N - \|\boldsymbol{e}\|_0}{N} \geq \frac{N - \frac{1}{2}\left(1 + \frac{1}{m(\boldsymbol{X})}\right)}{N}.$$

In addition, it is proved in [5] that

$$1 + \frac{1}{m(\boldsymbol{X})} \leq N - \nu_d(\boldsymbol{X}) + 1.$$

Therefore, it leads to

$$\frac{N - \|\boldsymbol{e}\|_0}{N} \geq \frac{N - \frac{1}{2}\left(N - \nu_d(\boldsymbol{X}) + 1\right)}{N} = \frac{1}{2} + \frac{\nu_d(\boldsymbol{X}) - 1}{2N}. \tag{2.28}$$

Moreover, from the definition of $\nu_d(\boldsymbol{X})$ (see Theorem 2), we know that $\nu_d(\boldsymbol{X}) \geq d \geq 1$. Thus, the condition becomes

$$\frac{N - \|\boldsymbol{e}\|_0}{N} \geq \frac{1}{2}. \tag{2.29}$$

This condition is difficult to satisfy in practice, particularly for systems with more than two modes. However, this condition is sufficient but not necessary and the method might also work in cases where it is violated.

Parameter sparsification method (PS)

The parameter sparsification method is proposed by Ozay *et al.* [72] to find a SARX model with a minimum number of switches or a minimum number of submodels that is consistent with the experimental data and the a priori information on the noise via an optimization problem. It deals with two cases for the data set.

Arbitrary data. For data sets which are not arranged in time order, Ozay *et al.* propose a procedure which applies the bounded-error principle in Sect. 2.2.2 (page 20) to find a SARX model with a minimum number of submodels.

For the first submodel, the following sparsification problem allows one to find one submodel parameter vector $\tilde{\boldsymbol{\theta}}$ which has a prediction error less than δ for a maximal number of data points:

$$\begin{aligned} \min_{\{\boldsymbol{\theta}(k)\}_{k=1}^N, \tilde{\boldsymbol{\theta}}} \quad & \|\{\boldsymbol{\theta}(k) - \tilde{\boldsymbol{\theta}}\}\|_0 \\ \text{s.t} \quad & |y_k - \boldsymbol{\theta}(k)^\top \boldsymbol{x}_k| \leq \delta, \quad k = 1, \dots, N, \end{aligned} \tag{2.30}$$

where $\{\boldsymbol{\theta}(k) - \tilde{\boldsymbol{\theta}}\}$ denotes a vector in $\mathbb{R}^N$ whose kth element is $\|\boldsymbol{\theta}(k) - \tilde{\boldsymbol{\theta}}\|_\infty$.

By applying the reweighted technique of Candès *et al.* which was summarized in Sect. 1.4.2, the program (2.30) is relaxed to a reweighted ℓ_1 optimization program,

$$\begin{aligned} \min_{\{\boldsymbol{\theta}(k)\}_{k=1}^N, \tilde{\boldsymbol{\theta}}, \boldsymbol{z}} \quad & \|\boldsymbol{W}\boldsymbol{z}\|_1 \\ \text{s.t} \quad & \|\boldsymbol{\theta}(k) - \tilde{\boldsymbol{\theta}}\|_\infty \leq z_k, \\ & |y_k - \boldsymbol{\theta}(k)^\top \boldsymbol{x}_k| \leq \delta, \quad k = 1, \dots, N. \end{aligned} \tag{2.31}$$

In this program, $\forall \delta$, the constraints $|y_k - \boldsymbol{\theta}(k)^\top \boldsymbol{x}_k| \leq \delta, \quad k = 1, \dots, N$, are always feasible. The relaxation by a ℓ_1-norm no longer ensures that the estimated parameter vector has a prediction error less than δ for a maximal number of data points. Contrary to the ES method (page 21), the conditions under which (2.30) and (2.31) are equivalent are not studied yet.

For the first obtained submodel, all the data satisfying the error bound δ are removed from the data set. The remaining is used to find the second submodel. This procedure continues until all data are removed.

The hyperparameters are the noise bound δ in (2.30) and (2.31) and the parameter ϵ of the iteratively reweighted algorithm presented in Sect. 1.4.2. The optimization problem has a number of constraints as well as of variables proportional to the number of data points N. Thus the method becomes slow for large data sets.

Data in time order. For a dynamical system, data points are typically consecutively collected in time order. Each data point at the time k is linked to a parameter vector $\boldsymbol{\theta}(k)$. These parameter vectors are expected to be unchanged during dwell times, i.e., between two mode switchings. Therefore, they can be estimated via the following sparsification problem:

$$\begin{aligned} \min_{\{\boldsymbol{\theta}(k)\}_{k=1}^N, \boldsymbol{e}} \quad & \|\{\boldsymbol{\theta}(k+1) - \boldsymbol{\theta}(k)\}\|_0 \\ \text{s.t} \quad & y_k - \boldsymbol{\theta}(k)^\top \boldsymbol{x}_k = e_k, \quad k = 1, \dots, N, \\ & \|\boldsymbol{e}\|_* \leq \delta, \end{aligned} \tag{2.32}$$

where $\{\boldsymbol{\theta}(k+1)-\boldsymbol{\theta}(k)\}$ denotes a vector in $\mathbb{R}^{N-1}$ whose kth element is $\|\boldsymbol{\theta}(k+1)-\boldsymbol{\theta}(k)\|_\infty$, $\boldsymbol{e} = [e_1, \dots, e_N]^\top \in \mathbb{R}^N$ is the error vector and $\|\cdot\|_*$ denotes an appropriate norm used to bound the error by δ.

The ℓ_0-norm is replaced by the ℓ_1-norm as a convex relaxation,

$$\begin{aligned} \min_{\{\boldsymbol{\theta}(k)\}_{k=1}^N, \boldsymbol{z}, \boldsymbol{e}} \quad & \|\boldsymbol{W}\boldsymbol{z}\|_1 \\ \text{s.t} \quad & \|\boldsymbol{\theta}(k) - \boldsymbol{\theta}(k-1)\|_\infty \leq z_k, \\ & y_k - \boldsymbol{\theta}(k)^\top \boldsymbol{x}_k = e_k, \qquad k = 1, \dots, N, \\ & \|\boldsymbol{e}\|_* \leq \delta, \end{aligned} \tag{2.33}$$

and the iteratively reweighted ℓ_1-norm method in Sect. 1.4.2 is used to improve the sparsity.

If $\|.\|_*$ is the ℓ_∞-norm, a simpler alternative for the above problem is a greedy sliding window algorithm [43]. This algorithm finds the first submodel that satisfies the error bound for the largest possible time interval $[1, \tau_1]$,

$$|y_k - \boldsymbol{\theta}_k^\top \boldsymbol{x}_1| \leq \delta, \quad \forall k \in [1, \tau_1]. \tag{2.34}$$

It starts with $\tau_1 = 1$, then, increases τ_1 until (2.34) is no longer satisfied. The second submodel is started from $\tau_1 + 1$. This procedure continues for all data points.

Sum-of-norms optimization

For a data set arranged in time order, Ohlsson *et al.* [70] deal with the identification of SARX systems by using a sum-of-norms regularization in the optimization program

$$\min_{\{\boldsymbol{\theta}(k)\}_{k=1}^N} \quad \sum_{k=1}^N \left(y_k - \boldsymbol{\theta}(k)^\top \boldsymbol{x}_k\right)^2 + \lambda \sum_{k=2}^N \|\boldsymbol{\theta}(k) - \boldsymbol{\theta}(k-1)\|_*, \tag{2.35}$$

where $\|.\|_*$ is the norm used for regularization and λ is a regularization parameter to control the trade-off between the model fit and the variation of the model parameters. A switching at time k incurs a variation between $\boldsymbol{\theta}(k-1)$ and $\boldsymbol{\theta}(k)$, and increases the cost function. Therefore, the regularization prevents the parameter variation if it is not necessary. This helps to find a suitable number of submodels and the submodel parameter vectors.

Notice that this principle is similar to the PS method. In these two methods, if the hyperparameter λ of (2.35) or δ of (2.33) is set sufficiently large, the parameter variation term is minimized to zero and we have only one mode. On the contrary, if it is small, we can have N different modes.

To ease the hyperparameter tuning, the authors propose a method to determine λ_{max} such that for all $\lambda \geq \lambda_{max}$, the optimization program (2.35) produces only one mode. Then, λ is tuned in the range $0.01\lambda_{max} \leq \lambda \leq \lambda_{max}$ to find a solution with a fixed number of modes.

2.3 PWARX system identification

We rewrite Problem 5 of Chapter 1 for PWARX models.

Problem 6. *Given a dat set $\mathcal{D} = \{(\boldsymbol{x}_k, y_k)\}_{k=1}^N$ generated by a PWARX system, estimate the number of submodels s, the submodel parameter vectors $\boldsymbol{\theta}_j$, $j = 1, \dots, s$, and the regions $\Re_j$, $j = 1, \dots, s$, or the switching boundaries in the regression space so that the discrete state $q_k = j$ if $\boldsymbol{x}_k \in \Re_j$.*

All methods for SARX systems can be extended to PWARX systems by adding a region estimation step. When the switching sequence $\{q_k\}_{k=1}^N$ in Problem 4 is estimated, the data can be labeled. For PWARX systems, each data group corresponding to a mode is distributed in a region. These labeled data points can be used to determine the regions $\Re_j$. This is a popular problem in machine learning and pattern recognition known as supervised classification. From a given set of data with labels, we find a model (a classifier) assigning a previously unseen input data to a class, i.e., labeling any input data. Some classical classification methods are Support Vector Machines (SVM) [29] and Multicategory Robust Linear Programming (M-RLP) [15, 14]. Note that reconstructing the regions from a finite data set always causes classification errors, with a small but nonzero probability, even when all data are correctly labeled. This can lead to large prediction errors for the PWA model, especially for discontinuous systems or dynamical systems where the prediction error is accumulated in the regression vectors. To reduce this, a larger data set or a re-classification procedure with points close to region boundaries can be useful.

Beside methods for SARX systems, there exist other methods which address only PWARX systems and exploit their properties. In this section, we review these methods.

2.3.1 Fixed number of modes

Clustering-based procedure

The clustering-based procedure proposed by Ferrari-Trecate *et al.* [39] is considered as a main approach for the PWARX system identification problem. As presented in Sect. 2.1, one needs local parameter vectors to classify the data. To have these vectors, local data sets (LDs) are created. Each of N LDs consists of one regressor point and its $(c-1)$ nearest neighbors. Consequently, among LDs, there are ones comprising only points of a single mode, called *pure* LDs, and ones comprising points of multiple modes, called *mixed* LDs. It is expected that the number of pure LDs is significantly greater than the number of mixed LDs.

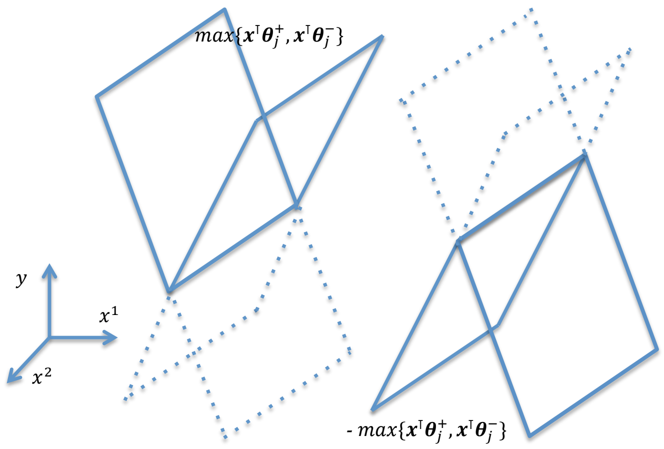
For each kth LD, a parameter vector $\boldsymbol{\theta}_k$ is calculated by least squares estimation as

$$\boldsymbol{\theta}_k = (\boldsymbol{\Phi}_k \boldsymbol{\Phi}_k^\top)^{-1} \boldsymbol{\Phi}_k \boldsymbol{y}_k, \tag{2.36}$$

where $\boldsymbol{\Phi}_k = [\boldsymbol{x}_{k_1}, \dots, \boldsymbol{x}_{k_c}]$ with $\boldsymbol{x}_{k_i}$, $i = 1, \dots, c$, the regression vectors of the kth LD, and $\boldsymbol{y}_k$ is the corresponding output vector. The mean $\boldsymbol{m}_k$ of the kth LD is calculated by $\boldsymbol{m}_k = \frac{1}{c}\sum_{i=1}^{c} \boldsymbol{x}_{k_i}$. Each point $(\boldsymbol{x}_k, y_k)$ is mapped to a feature vector $\boldsymbol{\xi}_k = [\boldsymbol{\theta}_k^\top \ \boldsymbol{m}_k^\top]^\top$. Then, classifying the feature vectors into groups is a standard problem of unsupervised classification (clustering). This is a non-convex optimization problem, typically solved by local methods, such as k-means [64], which are sensitive to the initialization and need to fix the number of groups, in our case, the number of modes. Finally, the model parameter vectors $\boldsymbol{\theta}_j$ and the regions $\Re_j$ are estimated from the groups.

In this procedure, tuning appropriately the nearest neighbor parameter c is important to obtain a good model.

Extensions of the clustering-based procedure [39] also start by creating N local data sets and submodels. But the number of submodels is reduced to obtain s submodels by grouping similar submodels. The similarity is measured by different ways. In [18], the proposed procedure merges clusters by using distance-fitting measures (Dempster-Shafer theory), while, in [89], a bounded-error criterion is used.



$max\{x^\top\theta_j^+, x^\top\theta_j^-\}$
y
x^1
x^2
$- max\{x^\top\theta_j^+, x^\top\theta_j^-\}$

Figure 2.1: Hinging function $g_j(\boldsymbol{x}) = \pm \max\{\boldsymbol{x}^\top\boldsymbol{\theta}_j^+, \boldsymbol{x}^\top\boldsymbol{\theta}_j^-\}$.

Global optimization for hinging hyperplane models

The hinging hyperplane model, presented by Breiman in [21], is a sum of hinge functions $g_j(\boldsymbol{x}) = \pm \max\{\boldsymbol{x}^\top\boldsymbol{\theta}_j^+, \boldsymbol{x}^\top\boldsymbol{\theta}_j^-\}$, i.e., a function consisting of two half-hyperplanes with two parameter vectors $\boldsymbol{\theta}_j^+$ and $\boldsymbol{\theta}_j^-$ (see Fig. 2.1). The $\pm$ sign shows a convex or concave hinge function. While the first algorithms in [21, 36, 76] for estimating the hinging hyperplane models guarantee to converge only to a local minimum, an algorithm for obtaining a global solution is proposed in [77].

In [77], the following hinging hyperplane ARX model,

$$f(\boldsymbol{x}) = \boldsymbol{x}^\top\boldsymbol{\theta}_0 + \sum_{j=1}^{s} w_j \max\{0, \boldsymbol{x}^\top\boldsymbol{\theta}_j\},$$

with a priori fixed $w_j \in \{-1, +1\}$, is used to approximate continuous PWA systems. The parameter vectors are found by minimizing a sum of absolute errors $\sum_{k=1}^{N} |f(\boldsymbol{x}_k) - y_k|$ or a sum of squared errors $\sum_{k=1}^{N}(f(\boldsymbol{x}_k) - y_k)^2$. The authors recast these optimization programs as a mixed integer linear program (MILP) and a mixed integer quadratic program (MIQP), respectively. These programs can be solved globally with branch-and-bound techniques, but only for small data sets since the number of integer variables as well as the number of constraints is proportional to the number of data points.

2.3.2 Free number of modes

Sum-of-norms optimization

In [68, 69], the sum-of-norms optimization method of Sect. 2.2.2 (page 24) is adapted to PWARX systems. Instead of the time information of data, it employs distances between data in regression space to build the regularization term in the optimization program

$$\min_{\{\boldsymbol{\theta}(k)\}_{k=1}^{N}} \sum_{k=1}^{N} \left(y_k - \boldsymbol{\theta}(k)^\top\boldsymbol{x}_k\right)^2 + \lambda \sum_{k,j=1}^{N} K(\boldsymbol{x}_k, \boldsymbol{x}_j)\|\boldsymbol{\theta}(k) - \boldsymbol{\theta}(j)\|_*, \tag{2.37}$$

where $\|.\|_*$ is the norm used for regularization, λ is a regularization parameter to control the trade-off between the model fit and the variation of the model parameters and $K(\boldsymbol{x}_k, \boldsymbol{x}_j)$ is a proximity function to evaluate the similarity between two regressors. It may be a binary function, i.e, $K(\boldsymbol{x}_k, \boldsymbol{x}_j) = 1$ if two points are considered as neighbors and $K(\boldsymbol{x}_k, \boldsymbol{x}_j) = 0$ otherwise. Similarly as in Sect. 1.4.2, to get more zeros in the regularization term, the iterative reweighted technique can be used as

$$\min_{\{\boldsymbol{\theta}(k)\}_{k=1}^N} \quad \sum_{k=1}^N \left(y_k - \boldsymbol{\theta}(k)^\top \boldsymbol{x}_k \right)^2 + \lambda \sum_{k,j=1}^N w_{kj}^{(i)} K(\boldsymbol{x}_k, \boldsymbol{x}_j) \|\boldsymbol{\theta}(k) - \boldsymbol{\theta}(j)\|_*, \tag{2.38}$$

where $w_{kj}^{(i)}$ is the weight at the ith iteration, with $w_{kj}^{(0)} = 1$. Finally, the number of modes is estimated as the number s of distinct $\boldsymbol{\theta}$-values in $\{\boldsymbol{\theta}_k\}_{k=1}^N$ which are also the s parameter vector estimates.

The hyperparameters of this procedure are the trade-off parameter λ of the cost function in (2.37) and the parameter ϵ of the reweighted algorithm (see Sect. 1.4.2). One also has to choose a proximity function to decide if two points are neighbors, which may introduce additional hyperparameters which can be crucial in obtaining s distinct $\boldsymbol{\theta}$-values in $\{\boldsymbol{\theta}_k\}_{k=1}^N$. For example, the following proximity function, based on the c-nearest neighbors as in Sect. 2.3.1 (page 25), is considered in [68]:

$$K(\boldsymbol{x}_k, \boldsymbol{x}_j) = \begin{cases} 1, & \text{if } \boldsymbol{x}_k \text{ is one of the } c \text{ nearest neighbors} \\ & \text{of } \boldsymbol{x}_j \text{ among all the data points,} \\ 0, & \text{otherwise.} \end{cases} \tag{2.39}$$

This method requires solving the convex optimization program (2.38) with dN variables. Thus the method becomes slow for systems with a high dimension d and large data sets.

2.4 Conclusions

In this chapter, we presented the main points of view for the identification of hybrid systems in the literature. Then, the existing methods were classified w.r.t the two types of systems, either SARX or PWARX, and the quantity that is fixed, either the number of modes or the bound on the model error. Such a classification of the methods shows that a complete experimental comparison of all methods, as partially realized in [51], does not make much sense. Each method is dedicated to a particular context and may work well in a situation but not in others. Instead, we summarized at the end of each method its hyperparameters, complexity and specific properties.

The existing methods are either suboptimal or limited to particular cases such as: small data sets for the method based on the global optimization for hinging hyperplane models (page 26), noiseless data for the algebraic methods (page 15) or some conditions of convex relaxation for the methods based on sparsity and convex relaxations (ES, PE and sum-of-norms methods presented in Sect. 2.2.2). Therefore, developing an optimal method for the noiseless case which remains robust to noise and can be applied to large data sets is still an open issue.

Note that our review is not exhaustive. The reader is referred to other reviews that use a different classification [73, 42]. In addition, in this thesis, we focus only on the ARX model which is used in most of works in the literature on hybrid system identification. However, a state-space model is more convenient in the automatic control domain since it can benefit from many existing tools. Though the ARX models can be transformed into a state-space form, there exist some other works [74, 97, 48, 16, 9, C4] which produce directly a state-space model under some assumptions.

Finally, linear hybrid system identification is closely related to the problem of subspace separation [100]. For instance the algebraic approach described in Sect. 2.2.1 is basically a subspace separation method known as Generalized Principal Component Analysis (GPCA) [102]. However, in this thesis we focus only on hybrid system identification and the reader is referred to [100] for a review of subspace separation.

Chapter 3

Geometric approach

ABSTRACT. *This chapter proposes a new approach for the identification of linear hybrid systems based on the geometric properties of hybrid systems in parameter space. More precisely, the data are mapped in that space such that each submodel is represented by a hypersphere. Then, we show how these hyperspheres can be separated by Principal Component Analysis (PCA) and derive a condition under which this separation is optimal for systems with two modes. Finally, classical (robust) regression is applied to estimate the system parameters from the classified data set. A simple procedure is also proposed to extend the method to the identification of switched systems with multiple modes. Experiments show that the final algorithm can accurately estimate both the parameters and the number of modes while being simple to apply and far more robust to noise than other methods.*

THIS chapter presents a new approach for the identification of linear hybrid systems. Particularly, Problem 4 in Chapter 1 is dealt with.

Problem 4. *Given a set $\mathcal{D} = \{(\boldsymbol{x}_k, y_k)\}_{k=1}^N$ generated by a linear hybrid system, estimate the number of submodels s, the submodel parameter vectors $\boldsymbol{\theta}_j$, $j = 1, \dots, s$, and the switching sequence $\{q_k\}_{k=1}^N$.*

We consider properties of linear hybrid systems directly in the parameter space instead of in the data space where other methods typically minimize an error criterion. This is an original point of view compared to previous approaches in the literature. In the parameter space, we derive a mapping of the data to construct hyperspheres which represent each of the submodels. Then, for hybrid systems with two modes, we propose a simple method based on Principal Component Analysis (PCA) [50] to separate the two hyperspheres. After this data classification step, the submodel parameter vectors can be estimated by any robust linear regression or identification method. In comparison with methods in Chapter 2, this approach does not include any hyperparameter to tune and does not rely on non-convex optimization. Finally, a simple iterative procedure is also proposed for the identification of systems with more than two modes.

We start in Sect. 3.1 by emphasizing main properties of hybrid systems in parameter space. Then, Sect. 3.2 is dedicated to the identification of hybrid systems with two modes, while Sect. 3.3 extends the proposed method to systems with multiple modes. The chapter ends with numerical examples in Sect. 3.4. The results of this chapter have been published in [C2] (see the Author's bibliography, page XVIII).

3.1 Hyperspheres of hybrid systems

In this section, the identification problem for SISO systems in SARX form (1.3),

$$y_k = \boldsymbol{\theta}_{q_k}^\top \boldsymbol{x}_k + v_k, \tag{3.1}$$

is considered from the viewpoint of the parameter space, first for the noiseless case in Sect. 3.1.1, then for the noisy one in Sect. 3.1.2.

3.1.1 Noiseless case

In the parameter space $\mathbb{R}^d$ of $\boldsymbol{\theta}$, a data pair $(\boldsymbol{x}_k, y_k)$ defines a subspace of dimension $d-1$ (a hyperplane) $\mathbb{P}_k$ as

$$\mathbb{P}_k : \quad \boldsymbol{x}_k^\top \boldsymbol{\theta} - y_k = 0. \tag{3.2}$$

Property 1. *Let $\mathcal{D}_j$ be the data subset of $\mathcal{D}$ generated by the jth subsystem. If there are at least d linearly independent data points in $\mathcal{D}_j$ then the corresponding parameter vector $\boldsymbol{\theta}_j$ is at the intersection of all $\mathbb{P}_k$ with $(\boldsymbol{x}_k, y_k) \in \mathcal{D}_j$, i.e.,*

$$\boldsymbol{\theta}_j = \bigcap_{k \in I_j} \mathbb{P}_k, \tag{3.3}$$

where

$$I_j = \{k : (\boldsymbol{x}_k, y_k) \in \mathcal{D}_j\}. \tag{3.4}$$

Figure 3.1 illustrates Property 1 for a noiseless data set generated by a SARX system with two modes.

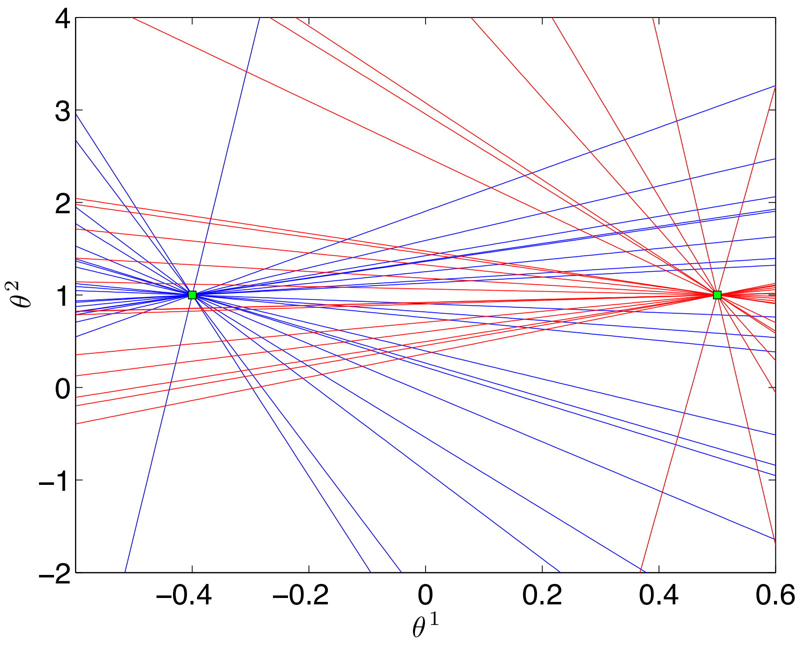


Figure 3.1: Parameter space $\mathbb{R}^2$ of a SARX system with 2 modes. The hyperplanes (lines) $\mathbb{P}_k$ intersect either at $\boldsymbol{\theta}_1$ or $\boldsymbol{\theta}_2$.

As all hyperplanes $\{\mathbb{P}_k\}_{k \in I_j}$, $j = 1, \dots, s$, intersect at the true parameter points $\boldsymbol{\theta}_j$ in the parameter space, the identification problem reduces to finding these points. However, directly searching for them as the solution to a system of equations is known as a hard combinatorial problem, which is intractable in practice for a large N. This is due to the fact that the discrete state q_k is unknown. On the other hand, if the data were well classified into each submodel, many classical regression methods could be applied to estimate the submodel parameters from the data subsets.

Property 1 is now exploited to transform the data classification problem in the data space to a problem of separating hyperspheres representing submodels in the parameter space. More precisely, each hyperplane $\mathbb{P}_k, k \in I_j$, will be mapped to a point $\boldsymbol{z}_k$ lying on a particular hypersphere for the mode j.

Let $\boldsymbol{z}_k$ be the orthogonal projection of an arbitrary point $\boldsymbol{\theta}$ on $\mathbb{P}_k$,

$$\boldsymbol{z}_k = \boldsymbol{z}(\boldsymbol{\theta}, \mathbb{P}_k) = \boldsymbol{\theta} + \boldsymbol{g}_k, \tag{3.5}$$

with

$$\boldsymbol{g}_k = \boldsymbol{g}_k(\boldsymbol{\theta}) = \boldsymbol{g}(\boldsymbol{\theta}, \mathbb{P}_k) = -\frac{\boldsymbol{x}_k^\top \boldsymbol{\theta} - y_k}{\|\boldsymbol{x}_k\|_2^2} \boldsymbol{x}_{\boldsymbol{k}}. \tag{3.6}$$

This mapping has the following properties:

- $\forall k \in I_j$, $\boldsymbol{g}_k \perp (\boldsymbol{\theta}_j - \boldsymbol{z}_k)$,
- $\boldsymbol{g}_k = \boldsymbol{0}$ if and only if $\boldsymbol{\theta} \in \mathbb{P}_k$,
- $\|\boldsymbol{g}_k\|_2 = |d_k|$,

where d_k is the algebraic distance from $\boldsymbol{\theta}$ to $\mathbb{P}_k$,

$$d_k = d(\boldsymbol{\theta}, \mathbb{P}_k) = \frac{\boldsymbol{x}_k^\top \boldsymbol{\theta} - y_k}{\|\boldsymbol{x}_k\|_2}. \tag{3.7}$$

By fixing an arbitrary point $\boldsymbol{\theta} \neq \boldsymbol{\theta}_j$, each point triplet $\{\boldsymbol{\theta}_j, \boldsymbol{\theta}, \boldsymbol{z}_k\}$, $k \in I_j$, forms a right triangle with the right angle at $\boldsymbol{z}_k$ and the hypotenuse $\boldsymbol{\theta\theta}_j$. We know from basic geometry that for fixed points $\boldsymbol{\theta}$, $\boldsymbol{\theta}_j$, all the points $\boldsymbol{z}_k$ lie on a hypersphere whose diameter is the line segment $\boldsymbol{\theta\theta}_j$, as illustrated in Fig. 3.2 (left) for $d = 2$.

Keeping $\boldsymbol{\theta}$ fixed and switching $\boldsymbol{\theta}_j$ for a different j, we obtain a different hypersphere. Therefore, each hypersphere represents a submodel and all these hyperspheres intersect at the chosen point $\boldsymbol{\theta}$. Figure 3.2 (right) shows an example with 2 modes and the dimension $d = 2$.

Hereafter, the data classification into each mode is considered as a hypersphere separation in the parameter space. The difficulty of this separation problem is intimately related to the choice of $\boldsymbol{\theta}$, which will be discussed in Sect. 3.2 for systems with two modes.

3.1.2 Noisy case

We now consider the effect of noise in the parameter space. According to model (3.1), $(\boldsymbol{x}_k, y_k, v_k)$ defines a hyperplane $\mathbb{P}_{k,v_k}$ as

$$\mathbb{P}_{k,v_k}: \quad \boldsymbol{x}_k^\top \boldsymbol{\theta} - y_k + v_k = 0. \tag{3.8}$$

Thus, $\mathbb{P}_k$ as defined by (3.2) in the noiseless case and $\mathbb{P}_{k,v_k}$ are parallel and at a distance $|d_{v_k}| = |\frac{v_k}{\|\boldsymbol{x}_k\|_2}|$. When the point $\boldsymbol{\theta}_j$ is the intersection of all $\{\mathbb{P}_k\}_{k \in I_j}$, the distance from $\boldsymbol{\theta}_j$ to $\mathbb{P}_{k,v_k}$ is also $|d_{v_k}|$. Then, all $\{\mathbb{P}_{k,v_k}\}_{k \in I_j}$ intersect the hypersphere of center $\boldsymbol{\theta}_j$ and radius $r_v = \max_{k \in I_j} |d_{v_k}|$.

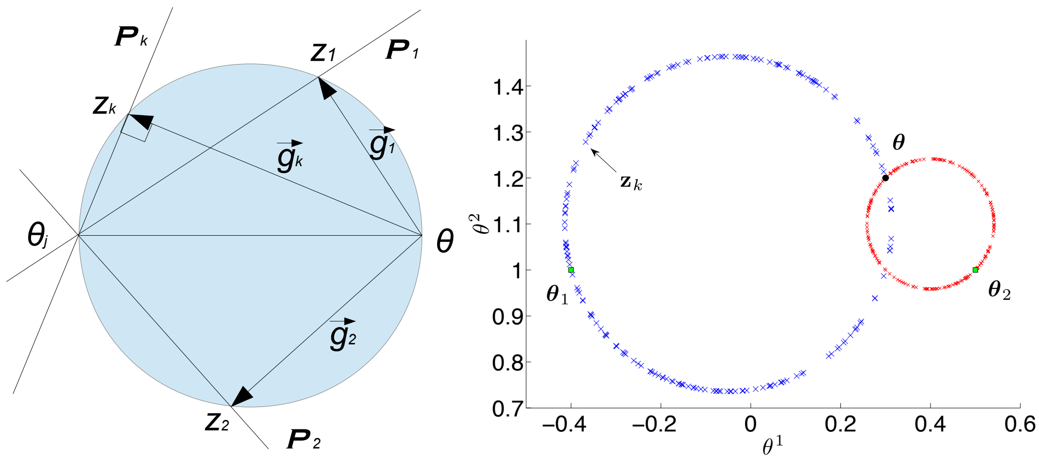


Figure 3.2: Vectors $\boldsymbol{g}_k$, points $\boldsymbol{z}_k$ and hyperplanes $\mathbb{P}_k$ of a submodel (left) and an illustration of two modes (right): the point $\boldsymbol{\theta}$ (dot), two submodel parameter points $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2$ (squares) and points $\boldsymbol{z}_k$ (crosses).

Property 2. *In the case where the data are corrupted by a bounded noise with* $\|\boldsymbol{v}\|_\infty = \max_{k\in\{1,\dots,N\}} |v_k| \leq \epsilon$, *one has*

$$\boldsymbol{\theta}_j \in \bigcap_{k\in I_j} \mathbb{P}_k^\epsilon, \tag{3.9}$$

where $\mathbb{P}_k^\epsilon$ *is a slab of thickness* 2ϵ*, defined by*

$$\mathbb{P}_k^\epsilon : \quad -\epsilon \leq \boldsymbol{x}_k^\top \boldsymbol{\theta} - y_k \leq \epsilon. \tag{3.10}$$

Figure 3.3 illustrates Property 2 for a noisy data set generated by a SARX system with two modes.

We note that $|d_{v_k}|$ is large for a small $\|\boldsymbol{x}_k\|_2$. This implies that for linear submodels, the effect of the noise on the parameter vector estimates is more serious with regressors close to the origin, whereas for affine models, this effect is reduced since $\|\tilde{\boldsymbol{x}}_k\|_2 \geq 1$ with $\tilde{\boldsymbol{x}}_k = [\boldsymbol{x}_k^\top, 1]^\top$.

The distance from the orthogonal projection $\boldsymbol{z}'_k$ of $\boldsymbol{\theta}$ on $\mathbb{P}_{k,v_k}$,

$$\boldsymbol{z}'_k = \boldsymbol{\theta} - \frac{\boldsymbol{x}_k^\top \boldsymbol{\theta} - y_k + v_k}{\|\boldsymbol{x}_k\|_2^2} \boldsymbol{x}_k, \tag{3.11}$$

with $k \in I_j$ to the jth mode hypersphere obtained for the noiseless case is $|\Delta_{k,v_k}|$ with

$$\Delta_{k,v_k} = \sqrt{R_j^2 + d_{v_k}^2 + 2R_j d_{v_k} \cos(\overrightarrow{\boldsymbol{O}_j \boldsymbol{z}_k}, \vec{\boldsymbol{g}_k})} - R_j, \tag{3.12}$$

where $\boldsymbol{O}_j, R_j$ are the center and the radius of the jth hypersphere. Since $|\Delta_{k,v_k}|$ is the smallest distance from $\boldsymbol{z}'_k$ to the hypersphere, $|\Delta_{k,v_k}| \leq |d_{v_k}|$.

We see that when $\boldsymbol{z}_k \to \boldsymbol{\theta}$, $\cos(\overrightarrow{\boldsymbol{O}_j \boldsymbol{z}_k}, \vec{\boldsymbol{g}_k}) \to 0$ and the points $\boldsymbol{z}'_k$ close to $\boldsymbol{\theta}$ are less affected by the noise, in which case,

$$|\Delta_{k,v_k}| \to \sqrt{R_j^2 + d_{v_k}^2} - R_j = \frac{d_{v_k}^2}{\sqrt{R_j^2 + d_{v_k}^2} + R_j} \leq \frac{d_{v_k}^2}{2R_j}.$$

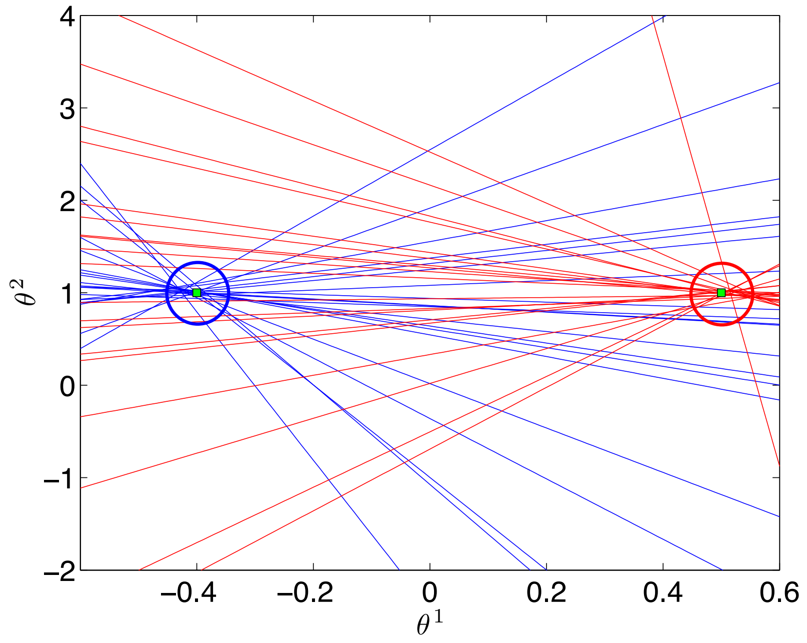


Figure 3.3: Parameter space $\mathbb{R}^2$ of a SARX system with 2 modes and noisy data set.

This property implies also a decreased effect of the noise in data classification since the area around $\boldsymbol{\theta}$ is the most difficult one to separate. This property is illustrated in Fig. 3.4. With a uniform noise, points $\boldsymbol{z}'_k$ close to $\boldsymbol{\theta}$ almost lie on the hyperspheres, whereas other $\boldsymbol{z}'_k$ are more perturbed.

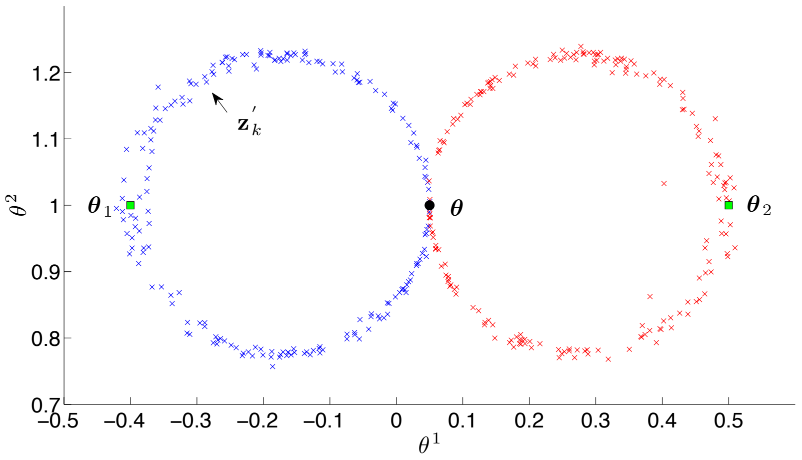


Figure 3.4: Illustration of 2 modes with a uniform noise ($d = 2$): the point $\boldsymbol{\theta}$ (dot), two submodel parameter points $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2$ (squares) and points $\boldsymbol{z}'_k$ (crosses).

3.2 Identification of linear hybrid systems with two modes

In this section, we concentrate on the identification of two-mode SARX systems which have a structure of two hyperspheres in parameter space. We propose a procedure including the following four steps.

1. Choose a point $\boldsymbol{\theta}$.

2. Map the data points $(\boldsymbol{x}_k, y_k)$ to points $\boldsymbol{z}_k$ on hyperspheres as in Sect. 3.1.1, i.e.,

$$\boldsymbol{z}_k = \boldsymbol{\theta} - \frac{\boldsymbol{x}_k^\top \boldsymbol{\theta} - y_k}{\|\boldsymbol{x}_k\|_2^2} \boldsymbol{x}_{\boldsymbol{k}}. \tag{3.13}$$

3. Separate the two hyperspheres to classify the data into two groups.

4. Estimate the two submodels from the data in each group.

Steps 1, 3 and 4 are further detailed in Sect. 3.2.1 and 3.2.2 below.

3.2.1 Choice of the point $\boldsymbol{\theta}$

We now search for a $\boldsymbol{\theta}$ which will ease the separation of the hyperspheres. If $\boldsymbol{\theta}$ belongs to the segment $\boldsymbol{\theta}_1\boldsymbol{\theta}_2$, i.e.,

$$\boldsymbol{\theta} = \boldsymbol{\theta}_1 + \mu(\boldsymbol{\theta}_2 - \boldsymbol{\theta}_1), \quad \mu \in (0,1), \tag{3.14}$$

the two hyperspheres are tangent at $\boldsymbol{\theta}$. With this ideal structure, a linear classification is sufficient to separate the two hyperspheres. However, $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2$ are not known yet, and we must find a $\boldsymbol{\theta}$ satisfying (3.14) from the data only. The normalized least squares solution $\boldsymbol{\theta}^* \in \mathbb{R}^d$, minimizing the cost function

$$L(\boldsymbol{\theta}) = \frac{1}{2}\sum_{k=1}^{N} \frac{(\boldsymbol{x}_k^\top \boldsymbol{\theta} - y_k)^2}{\|\boldsymbol{x}_k\|_2^2}, \tag{3.15}$$

is considered as a candidate for $\boldsymbol{\theta}$. The following proposition gives a case where this candidate is a good choice.

Proposition 1. *Given a data set $\mathcal{D}$ generated by a switched system with two modes such that the matrix $[\boldsymbol{x}_1, \dots, \boldsymbol{x}_N]$ is of full row rank and the condition*

$$\sum_{k=1}^{N} \frac{\boldsymbol{x}_k^T \boldsymbol{\theta}_1 - y_k}{\|\boldsymbol{x}_k\|_2^2} \boldsymbol{x}_k = \gamma \sum_{k=1}^{N} \frac{\boldsymbol{x}_k^T \boldsymbol{\theta}_2 - y_k}{\|\boldsymbol{x}_k\|_2^2} \boldsymbol{x}_k, \tag{3.16}$$

where $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ are the subsystem parameter vectors and $\gamma \in \mathbb{R}$, holds, the normalized least squares solution $\boldsymbol{\theta}^*$ minimizing (3.15) *belongs to the segment $\boldsymbol{\theta}_1\boldsymbol{\theta}_2$, i.e., $\boldsymbol{\theta}^* = \boldsymbol{\theta}_1 + \mu(\boldsymbol{\theta}_2 - \boldsymbol{\theta}_1)$ with $\mu = \frac{-\gamma}{1-\gamma} \in [0,1]$.*

The condition (3.16) states that the sum of all vectors $\boldsymbol{g}_k(\boldsymbol{\theta}_1)$ must be collinear with the sum of all vectors $\boldsymbol{g}_k(\boldsymbol{\theta}_2)$, or equivalently that $\nabla L(\boldsymbol{\theta}_1) = \gamma \nabla L(\boldsymbol{\theta}_2)$.

Proof. Since with $\boldsymbol{g}_k$ given by (3.6), $\boldsymbol{g}_k(\boldsymbol{\theta}_j) = \mathbf{0}$, $\forall k \in I_j, j \in \{1,2\}$, condition (3.16) is rewritten as

$$\sum_{k\in I_2} \boldsymbol{g}_k(\boldsymbol{\theta}_1) = \gamma \sum_{k\in I_1} \boldsymbol{g}_k(\boldsymbol{\theta}_2). \tag{3.17}$$

From the geometric properties of the vectors (see Fig. 3.5), it is easy to see that

$$\boldsymbol{g}_k(\boldsymbol{\theta}_1)^\top(\boldsymbol{\theta}_2 - \boldsymbol{\theta}_1) \geq 0,\ \forall k \in I_2,$$
$$\boldsymbol{g}_k(\boldsymbol{\theta}_2)^\top(\boldsymbol{\theta}_2 - \boldsymbol{\theta}_1) \leq 0,\ \forall k \in I_1.$$

This leads to $\gamma \leq 0$.
For any $\boldsymbol{\theta}$ in the segment $\boldsymbol{\theta}_1\boldsymbol{\theta}_2$ such that $\boldsymbol{\theta} - \boldsymbol{\theta}_1 = \mu(\boldsymbol{\theta}_2 - \boldsymbol{\theta}_1)$, we have

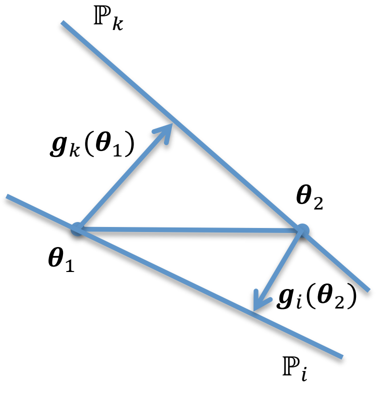

$\mathbb{P}_k$
$\boldsymbol{g}_k(\boldsymbol{\theta}_1)$
$\boldsymbol{\theta}_2$
$\boldsymbol{\theta}_1$
$\boldsymbol{g}_i(\boldsymbol{\theta}_2)$
$\mathbb{P}_i$

Figure 3.5: Geometry of the vectors $\boldsymbol{g}_k(\boldsymbol{\theta}_j)$ in the parameter space.

$$\boldsymbol{g}_k(\boldsymbol{\theta}) = \mu\boldsymbol{g}_k(\boldsymbol{\theta}_2),\ \forall k \in I_1,$$
$$\boldsymbol{g}_k(\boldsymbol{\theta}) = (1-\mu)\boldsymbol{g}_k(\boldsymbol{\theta}_1),\ \forall k \in I_2.$$

Thus, (3.17) is equivalent to

$$\sum_{k\in I_2} \boldsymbol{g}_k(\boldsymbol{\theta}) = \frac{(1-\mu)}{\mu}\gamma \sum_{k\in I_1} \boldsymbol{g}_k(\boldsymbol{\theta}).$$

Then, if we take $\boldsymbol{\theta}$ with $0 \leq \mu = \frac{-\gamma}{1-\gamma} \leq 1$, we get

$$\sum_{k=1}^{N} \boldsymbol{g}_k(\boldsymbol{\theta}) = \sum_{k\in I_1} \boldsymbol{g}_k(\boldsymbol{\theta}) + \sum_{k\in I_2} \boldsymbol{g}_k(\boldsymbol{\theta}) = \left(1 + \frac{1-\mu}{\mu}\gamma\right) \sum_{k\in I_1} \boldsymbol{g}_k(\boldsymbol{\theta}) = \mathbf{0}. \tag{3.18}$$

On the other hand, the gradient of the cost function $L(\boldsymbol{\theta})$ is zero at $\boldsymbol{\theta}^*$, i.e., $\nabla L(\boldsymbol{\theta})|_{\boldsymbol{\theta}^*} = \sum_{k=1}^{N} \boldsymbol{g}_k(\boldsymbol{\theta}^*) = \mathbf{0}$. Since $\boldsymbol{\theta}^*$ is unique if $\text{rank}[\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N] = d$, we conclude that $\boldsymbol{\theta} \equiv \boldsymbol{\theta}^*$. □

Example. Given a data set $\mathcal{D}$ generated by a switched system with two modes such that for each value $\boldsymbol{x}$ in the data set, the data set includes both of the outputs $f_1(\boldsymbol{x})$ and $f_2(\boldsymbol{x})$ and the matrix $[\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N]$ is of full row rank, the least squares solution $\boldsymbol{\theta}^* \in \mathbb{R}^d$ minimizing (3.15) is the midpoint of the segment $\boldsymbol{\theta}_1\boldsymbol{\theta}_2$, i.e.,

$$\boldsymbol{\theta}^* = \frac{1}{2}(\boldsymbol{\theta}_1 + \boldsymbol{\theta}_2). \tag{3.19}$$

Indeed, with such a data set, it is easy to see that $\gamma = -1$ in (3.16). Then, according to the proof of Proposition 1, $\mu = 0.5$ and we have (3.19)

For particular data sets not satisfying (3.16), the point $\boldsymbol{\theta}^*$ may be out of alignment with $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$. Then the two hyperspheres are no longer tangent and the separation task is more difficult. In particular, points in the intersection of the two corresponding hyper-balls will be badly classified by a linear classifier. However, a good submodel can be estimated from such a data set including a few misclassified points (outliers) by a robust regression method.

3.2.2 Data classification and submodel estimation

We propose a linear classification method for classifying data points into two groups based on the Principal Component Analysis (PCA) (see Appendix A) for the matrix

$\boldsymbol{Z} = [\boldsymbol{z}_1, \dots, \boldsymbol{z}_N, \boldsymbol{\theta}]$. As a pre-processing step in PCA, the matrix $\boldsymbol{Z}$ is centered to the matrix $\tilde{\boldsymbol{Z}} = [\tilde{\boldsymbol{z}}_1, \dots, \tilde{\boldsymbol{z}}_N, \tilde{\boldsymbol{z}}_{\boldsymbol{\theta}}]$ such that $\sum_{k=1}^N \tilde{\boldsymbol{z}}_k + \tilde{\boldsymbol{z}}_{\boldsymbol{\theta}} = \boldsymbol{0}$. Note that with the choice $\boldsymbol{\theta} = \boldsymbol{\theta}^*$ as in Proposition 1, the centered matrix is simply $\tilde{\boldsymbol{Z}} = [\boldsymbol{g}_1, \dots, \boldsymbol{g}_N, \boldsymbol{0}]$ since $\sum_{k=1}^N \boldsymbol{g}_k = \boldsymbol{0}$. The eigenvector corresponding to the largest eigenvalue of $\tilde{\boldsymbol{Z}}\tilde{\boldsymbol{Z}}^\top$ represents a direction on which the data of $\tilde{\boldsymbol{Z}}$ has the largest variance. This eigenvector has the same direction as the vector $\overrightarrow{\boldsymbol{\theta}_1\boldsymbol{\theta}_2}$ in the case of two tangent hyperspheres that are uniformly sampled. In addition, the hyperspheres are tangent at $\boldsymbol{\theta}^*$ which is the origin of the eigenspace. Then, a projection of vectors $\tilde{\boldsymbol{z}}_k$ on this eigenvector can separate the data into two groups corresponding to the submodels.

Let $q_1, \boldsymbol{v}_1$ be the largest eigenvalue and the corresponding eigenvector of $\tilde{\boldsymbol{Z}}\tilde{\boldsymbol{Z}}^\top$. Then, two data groups are found by

$$\begin{aligned} \hat{I}_1 &= \left\{k \in \{1, \dots, N\} : \tilde{\boldsymbol{z}}_k^\top \boldsymbol{v}_1 \geq 0\right\}, \\ \hat{I}_2 &= \left\{k \in \{1, \dots, N\} : \tilde{\boldsymbol{z}}_k^\top \boldsymbol{v}_1 < 0\right\}. \end{aligned} \tag{3.20}$$

Note that this gives preliminary estimates of the mode as

$$\hat{q}_k = \begin{cases} 1, & \text{if } k \in \hat{I}_1 \\ 2, & \text{if } k \in \hat{I}_2. \end{cases} \tag{3.21}$$

Now, we can use any robust regression method to estimate the submodel parameter vectors. A simple robust and convex method minimizes the ℓ_1-norm of the error as

$$\hat{\boldsymbol{\theta}}_j = \arg\min_{\boldsymbol{\theta}} \sum_{k \in \hat{I}_j} |y_k - \boldsymbol{x}_k^\top \boldsymbol{\theta}|, \qquad j = 1, 2. \tag{3.22}$$

Finally, one can re-estimate the mode $\hat{q}_k$ with

$$\hat{q}_k = \arg\min_{j \in \{1,2\}} |y_k - \hat{\boldsymbol{\theta}}_j^\top \boldsymbol{x}_k|, \qquad k = 1, \dots, N. \tag{3.23}$$

For a refinement, the submodels can be re-estimated by (3.22) with new subsets $\hat{I}_1$, $\hat{I}_2$ corresponding to the classification (3.23).

3.3 Systems with more than two modes

Inspired by bounded-error like methods reviewed in Sect. 2.2.2 (page 20), which estimate the submodels one by one, we propose a simple procedure to identify hybrid systems with more than two modes. In these methods, after estimating a parameter vector $\hat{\boldsymbol{\theta}}$, the data points verifying the error condition, $|y_k - \boldsymbol{x}_k^\top \hat{\boldsymbol{\theta}}| \leq \delta$, where δ is a fixed threshold, are associated to the estimated submodel and removed from the data set to estimate the next parameter vector. The number of modes is finally estimated as the number of submodels required to satisfy the error condition for all data points.

Algorithm 3 extends the method proposed in Sect. 3.2 in a similar manner. At each iteration, the remaining data set is separated into two groups. One group is for the estimated submodel and another contains the other points.

This procedure includes two hyperparameters: the error threshold δ and the number N_{min} of data left aside at the end of the procedure. For a bounded noise, δ can be chosen such that $\delta \geq \|\boldsymbol{v}\|_\infty$ and N_{min} can be set to 0. If the noise is a white Gaussian noise

Algorithm 3 Geometric approach with more than two modes

```
Require: a data set $\mathcal{D} = \{(\boldsymbol{x}_k, y_k)\}_{k=1}^N$ and thresholds $\delta$ and $N_{min}$.
  - Initialize $\mathcal{D}_{train} = \mathcal{D}$ , $j = 0$.
  while $|\mathcal{D}_{train}| \geq N_{min}$ do
    - Set $j = j + 1$.
    - Minimize the normalized least squares cost function (3.15) with the data in $\mathcal{D}_{train}$
    to get $\boldsymbol{\theta}^*$.
    - Map the data points $(\boldsymbol{x}_k, y_k)$ to points $\boldsymbol{z}_k$ by (3.13) with $\boldsymbol{\theta} = \boldsymbol{\theta}^*$.
    - Classify the data set $\mathcal{D}_{train}$ into two groups via the points $\boldsymbol{z}_k$ and (3.20).
    - Estimate two parameter vectors $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2$ corresponding to the groups as in (3.22).
    - Identify the $j$th submodel parameter vector by
          $$\hat{\boldsymbol{\theta}}_j = \underset{\boldsymbol{\theta} \in \{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2\}}{\arg\max} |\mathcal{D}(\boldsymbol{\theta})| ,$$
    where $\mathcal{D}(\boldsymbol{\theta}) = \left\{(\boldsymbol{x}_k, y_k) \in \mathcal{D}_{train} : |y_k - \boldsymbol{x}_k^\top \boldsymbol{\theta}| \leq \delta\right\}$.
    - Set $\mathcal{D}_{train} = \mathcal{D}_{train} \backslash \mathcal{D}(\hat{\boldsymbol{\theta}}_j)$.
  end while
  return the estimated number of modes $\hat{s} = j$ and the estimated parameter matrix
  $\hat{\boldsymbol{\Theta}} = [\hat{\boldsymbol{\theta}}_1, \dots, \hat{\boldsymbol{\theta}}_{\hat{s}}]$.
```

with a standard deviation σ_v, the threshold δ may be chosen in $[\sigma_v, 3\sigma_v]$. In this case or with unbounded noise in general, to avoid the effect of outliers, i.e., points with large noise terms ($|v_k| > 3\sigma_v$), the algorithm should be stopped with $N_{min} > 0$, typically set as a small percentage of N.

As for the bounded-error like methods in Sect. 2.2.2, when the threshold δ is appropriately chosen, the number of modes s is recovered. Nevertheless, experiments in the next section will show that the proposed method is less sensitive to the choice of δ.

3.4 Numerical experiments

In this section, we show the efficiency of the proposed method via some numerical examples. To evaluate the quality of the results, we compute the Normalized Parametric Error (NPE) (1.23), the Classification Error rate (CE) (1.24) and the estimated number of modes. Over 100 trials with different input, switching and noise sequences, we report the means and standard deviations. In the plots of NPE, the Ref line corresponds to the reference model estimated with knowledge of the true mode.

3.4.1 Two mode examples

For the examples with two modes, the proposed method (GEO) is compared with the methods using a fixed number of modes (Sect. 2.2.1) as the algebraic approach (ALG) (page 15), the continuous optimization based approach (CO) (page 18) and the alternating optimization method (ALT) (page 19). In these experiments, the data are generated with a uniform distribution $\mathcal{U}(-4, 4)$ of the regressors or of the input u_k for dynamical systems and an additive Gaussian noise v_k of standard deviation σ_v which is varied from 0 to 0.9 to evaluate the sensitivity to noise of the methods. While two methods ALG and GEO have no hyperparameter to tune with two modes for the latter, in the ALT method, we set the number of restarts to 100 and randomly initialize each parameter in the interval

$[-100, 100]$. In the CO method, we use the PE estimator (2.14) with Hampel's loss function (1.11), $\delta_v = 1$ and the MCS global solver with its default parameters.

Regession example

We take a one-dimensional switched function with intersecting submodels

$$y_k = \begin{cases} x_k + 2 + v_k, & \text{if } q_k = 1, \\ -x_k + v_k, & \text{if } q_k = 2. \end{cases} \tag{3.24}$$

For (3.24), training sets of $N = 200$ points are generated with a uniformly distributed random sequence of $q_k \in \{1, 2\}$. Figure 3.6 shows the submodel estimated with a noisy data set ($\sigma_v = 0.5$).

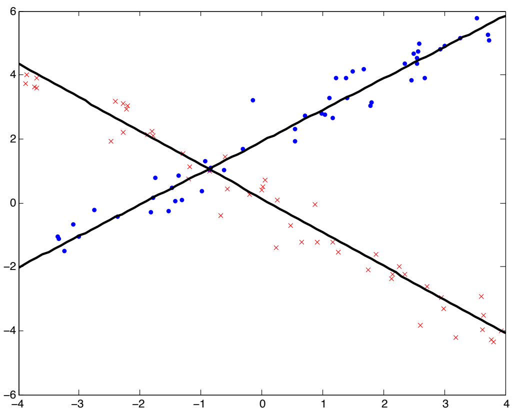


Figure 3.6: System (3.24): noisy data (●, ×) and estimated submodels (−).

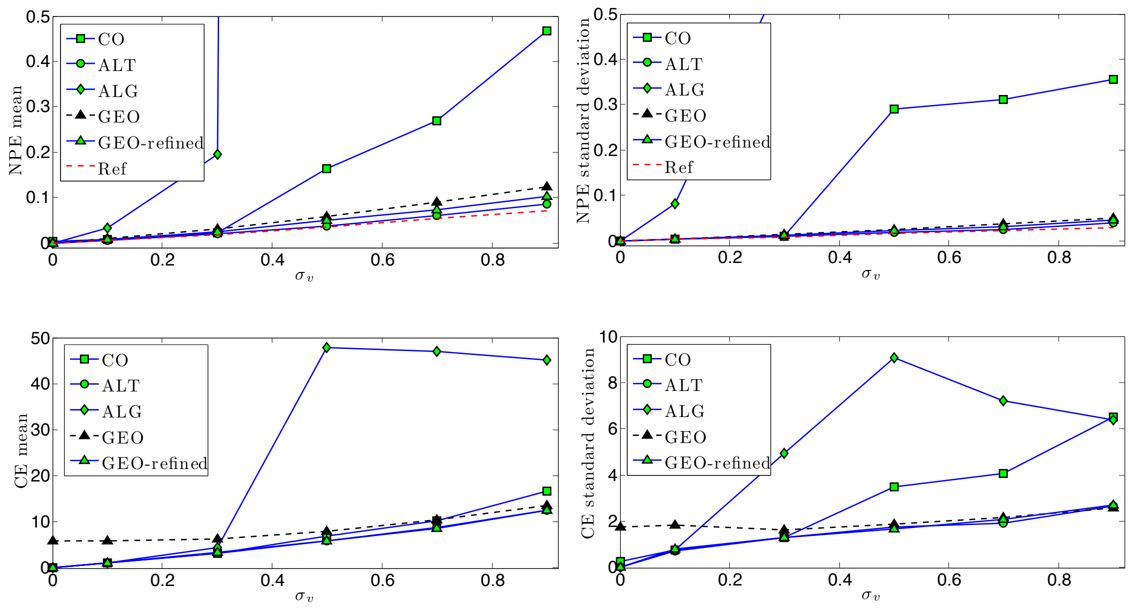


Figure 3.7: System (3.24): parametric error (NPE) (top) and classification error rate (CE) (bottom) versus Gaussian noise standard deviation.

Figure 3.7 reports the comparisons of the NPE and the CE with the other methods for a varying noise standard deviation. In this figure, the results of the proposed method are presented for two steps. The GEO line reports the results with the parameter vectors estimated by (3.22) using the classification (3.21). The GEO-refined line presents the results with a refinement given by the classification (3.23) and the re-estimation of the parameter vectors by (3.22) using this classification. They show that the proposed method can handle high noise regimes as well as the ALT method. The model estimated by the GEO with refinement is as accurate as the reference model (Ref).

Dynamical example

Consider the linear hybrid system used in [72]:

$$y_k = \begin{cases} 0.2y_{k-1} + 0.24y_{k-2} + 2u_{k-1} + v_k, & \text{if } q_k = 1, \\ -1.4y_{k-1} - 0.53y_{k-2} + u_{k-1} + v_k, & \text{if } q_k = 2. \end{cases} \tag{3.25}$$

Training sets of $N = 400$ points are generated by (3.25) with $\boldsymbol{x}_1 = [y_0,\ y_{-1},\ u_0] = [0.1, 0.1, 0.1]$ and a uniformly distributed random sequence of $q_k \in \{1, 2\}$. We note the standard deviation of the output $\sigma_y \simeq 3.6$ to compare with the standard deviation of the noise $\sigma_v = 0.9$.

Figure 3.8 shows the NPE and the CE of the methods. The proposed method gives models with a small error and is robust to noise.

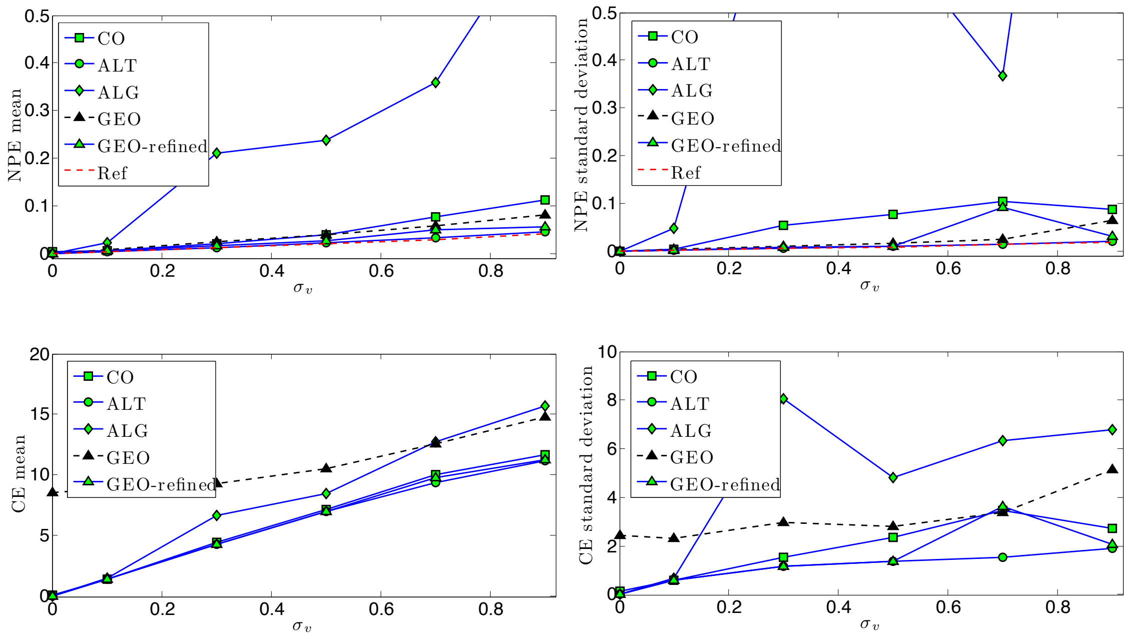


Figure 3.8: System (3.25): parametric error (NPE) (top) and classification error rate (CE) (bottom) versus Gaussian noise standard deviation.

3.4.2 Three mode examples

For the next examples, the proposed method (GEO) is compared with the sparsification based approaches of Sect. 2.2.2, ES (page 21) and PS (page 23), where the number of modes is estimated by fixing an error threshold δ. These methods estimate the parameter vectors one by one until the data set is empty. Due to Gaussian noise, the stopping criterion is set with $N_{min} = 0.05N$ in order to avoid estimating too many irrelevant submodels. Since the number of modes cannot be fixed, we retain the best estimated parameter vectors (submodels) as the ones closest to the true ones to compute the NPE and to estimate the switching sequence, then, the CE. The threshold δ used to assign data points to a submodel in these methods is varied from σ_v to $10\sigma_v$ to evaluate the sensitivity to the tuning parameter δ in these methods.

Regression examples

We consider the following switched hybrid system with 3 modes:

$$y_k = \begin{cases} [1, 0.5]\boldsymbol{x}_k + 2 + v_k, & \text{if } q_k = 1, \\ [-0.5, -1.3]\boldsymbol{x}_k + v_k, & \text{if } q_k = 2, \\ [-1, 1]\boldsymbol{x}_k - 1 + v_k, & \text{if } q_k = 3, \end{cases} \tag{3.26}$$

where $\boldsymbol{x}_k \in \mathbb{R}^2$. Training sets of $N = 900$ points are generated by (3.26) with a uniform noise with a fixed $\sigma_v = 0.3$ and with two scenarios for the random switching sequence $\{q_k\}_{k=1}^{900}$:

- $|\{k : q_k = 1\}| = 100, |\{k : q_k = 2\}| = 300, |\{k : q_k = 3\}| = 500;$
- a uniformly distributed random sequence of $q_k \in \{1, 2, 3\}$.

The first scenario satisfies the necessary condition on the data set (2.28). Figure 3.9 presents the NPE, the CE and the estimated number of modes for the methods. The proposed method gives the best results and estimates well the number of modes with $\delta \geq 3\sigma_v$.

Figures 3.10 shows the results for the second scenario in which the proposed method (GEO) still correctly works, whereas the ES method breaks down in these unfavorable conditions. In the two cases, the PS method does not work correctly.

Dynamical example

Consider the switched linear system with the three modes used in the experiments of [5]:

$$y_k = \begin{cases} -0.40y_{k-1} + 0.25y_{k-2} - 0.15u_{k-1} + 0.08u_{k-2} + v_k, & \text{if } q_k = 1, \\ 1.55y_{k-1} - 0.58y_{k-2} - 2.10u_{k-1} + 0.96u_{k-2} + v_k, & \text{if } q_k = 2, \\ 1.00y_{k-1} - 0.24y_{k-2} - 0.65u_{k-1} + 0.30u_{k-2} + v_k, & \text{if } q_k = 3. \end{cases} \tag{3.27}$$

Training sets of $N = 600$ points are generated by this system with $\boldsymbol{x}_1 = [y_0,\ y_{-1},\ u_0,\ u_{-1}] = [0.1, 0.1, 0.1, 0.1]$, a uniformly distributed random sequence of $q_k \in \{1, 2, 3\}$ and $\sigma_v = 0.1$. We note the standard deviation of the output $\sigma_y \simeq 1.8$ to compare with the standard deviation of the noise σ_v.

Figure 3.11 shows that the proposed method (GEO) can estimate the system (3.27) as accurately as the method using knowledge of the true mode (Ref). Moreover, it produces the best results for the NPE, the CE and the estimated number of modes over a large interval of δ in comparison with the other methods.

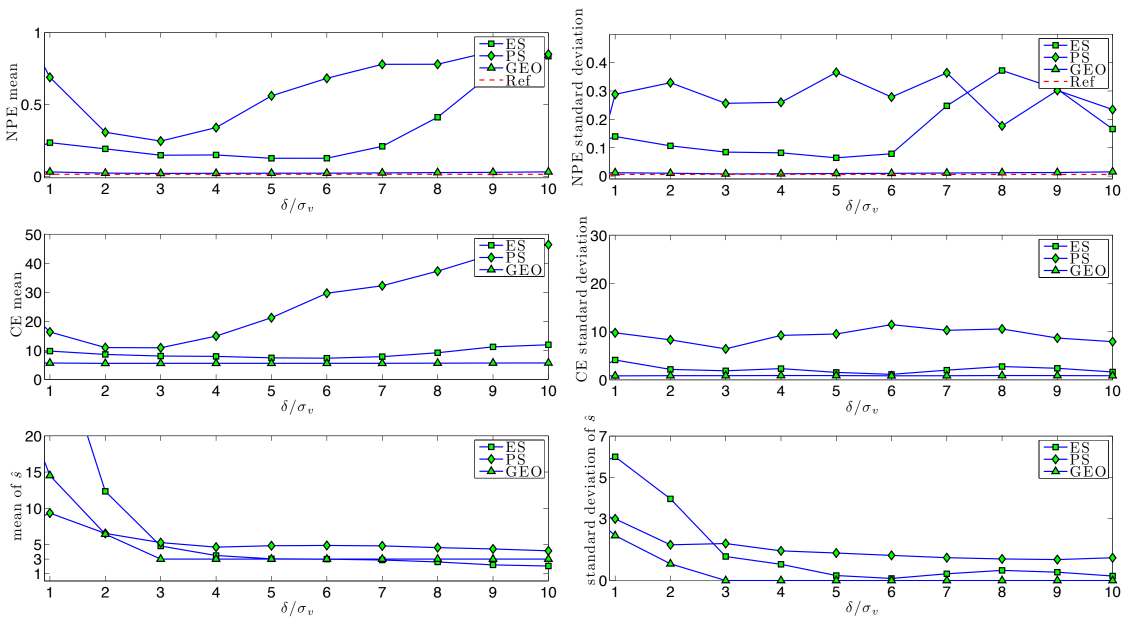


Figure 3.9: System (3.26) with the first scenario: Parametric error (NPE) (top), classification error rate (CE) (middle) and estimated number of modes (bottom) versus the threshold δ.

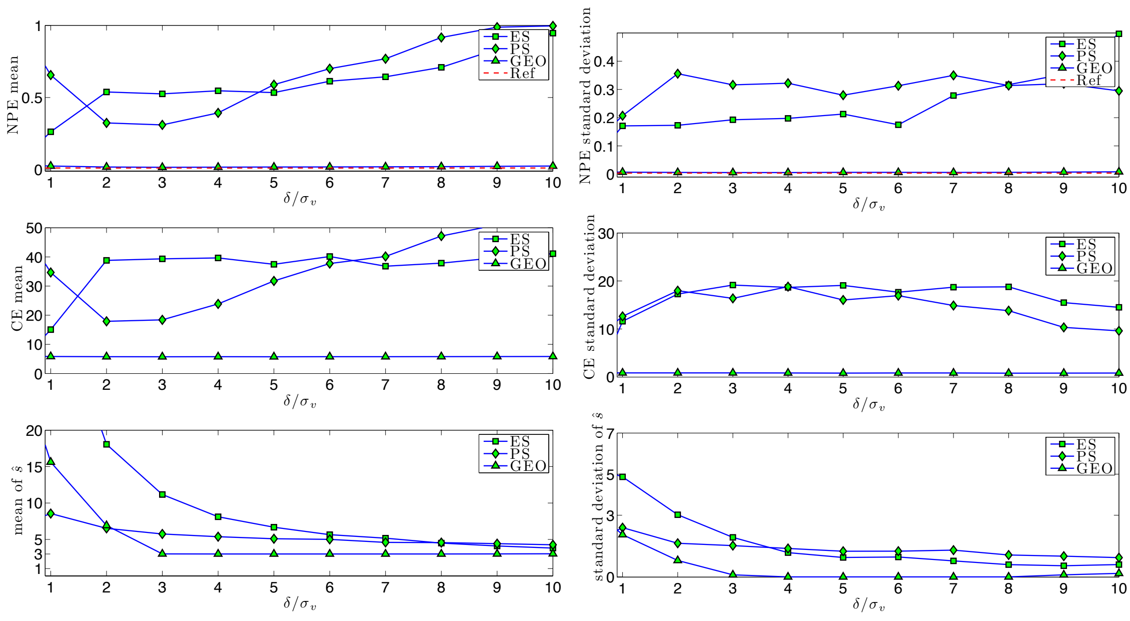


Figure 3.10: System (3.26) with the second scenario: Parametric error (NPE) (top), classification error rate (CE) (middle) and estimated number of modes (bottom) versus the threshold δ.

3.5 Discussion

In the proposed method (GEO), the linear classification of data points into groups is based on PCA and is under the influence of:

- the position of the chosen point $\boldsymbol{\theta}$ which leads to the tangency or the intersection of

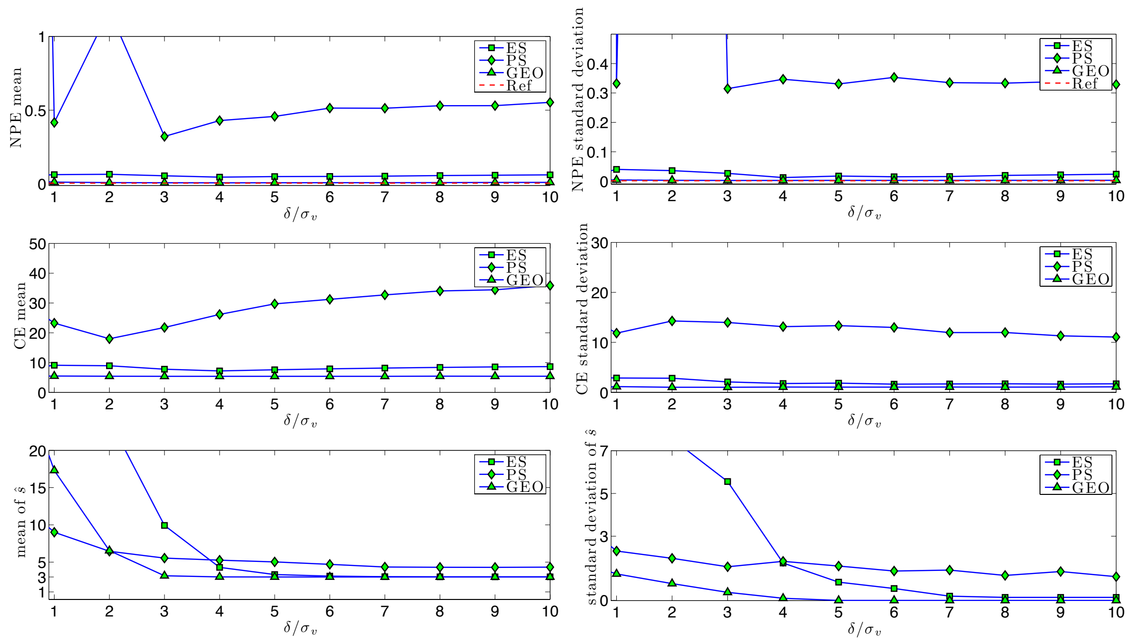


Figure 3.11: Switched linear system (3.27): Parametric error (NPE) (top), classification error rate (CE) (middle) and estimated number of modes (bottom) versus the threshold δ.

the hyperspheres,

- the distribution of the points $\boldsymbol{z}_k$ on the hyperspheres which are used to find the first principal component.

To analyze these influences on the classification, we consider the system

$$y_k = \begin{cases} \theta_1^1 x_k + \theta_1^2, & \text{if } q_k = 1, \\ \theta_2^1 x_k + \theta_2^2, & \text{if } q_k = 2. \end{cases} \tag{3.28}$$

Different data sets generated by (3.28) and the hyperspheres are presented in Figures 3.12, 3.13 and 3.14.

The first case (Fig. 3.12) is almost ideal. The points $\boldsymbol{z}_k$ are distributed quite uniformly on two tangent hyperspheres and the first eigenvector has the same direction as the vector $\overrightarrow{\boldsymbol{\theta}_2\boldsymbol{\theta}_1}$.

The second case (Fig. 3.13) gives a correct classification although the two hyperspheres are not complete and the direction of the first eigenvector is slightly changed.

The third case (Fig. 3.14) gives two intersecting and incomplete hyperspheres where the linear classification via PCA cannot work.

Proposition 1 (page 34) gives one of the cases where the least squares solution is a good point $\boldsymbol{\theta}$ to build appropriate hyperspheres for a linear classification based on the projection on the first eigenvector of PCA. In other cases, it is still an open issue to choose $\boldsymbol{\theta}$ and methods to separate the hyperspheres.

3.6 Conclusions

This chapter presented a geometric approach with an original point of view with respect to the literature for hybrid system identification. We first focused on systems switching

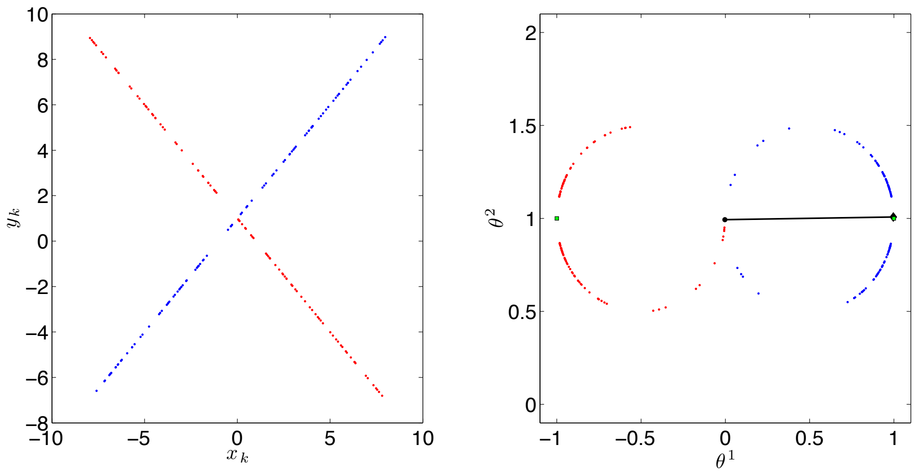


Figure 3.12: Data set (left) and parameter space (right) with true parameter vectors (green squares), first eigenvector (black segment) and points (dots) on the hyperspheres.

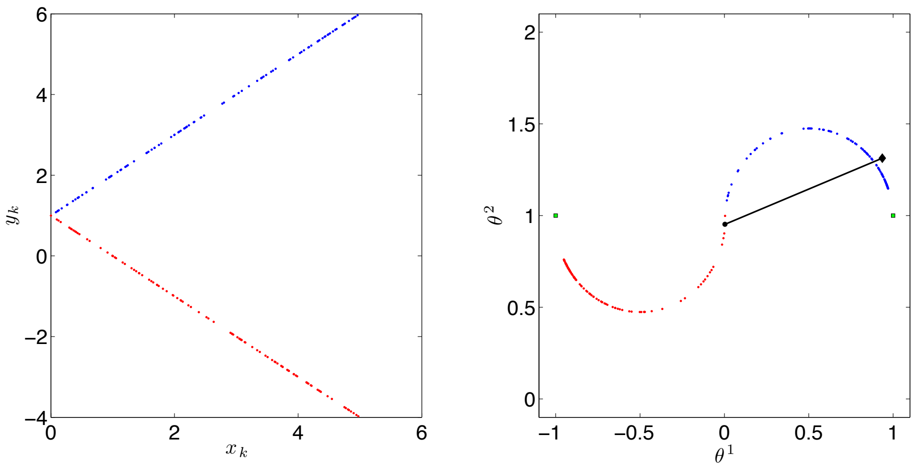


Figure 3.13: Data set (left) and parameter space (right) with true parameter vectors (green squares), first eigenvector (black segment) and points (dots) on the hyperspheres.

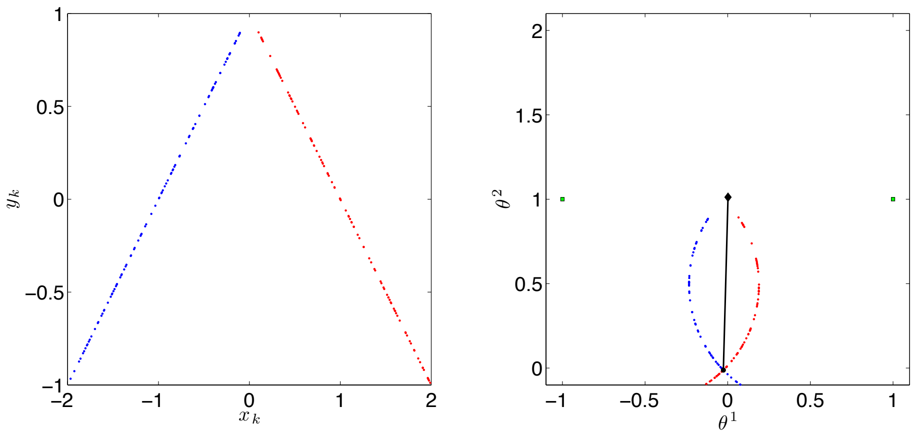


Figure 3.14: Data set (left) and parameter space (right) with true parameter vectors (green squares), first eigenvector (black segment) and points (dots) on the hyperspheres.

between two modes and then extended the proposed method to deal with an arbitrary number of modes. The proposed procedures proved their simplicity, accuracy and robustness to noise in numerical experiments. For hybrid systems with two modes, there is no hyperparameter to tune and no (non-convex) optimization problem to solve, contrary to most of the other methods. For hybrid systems with multiple modes, there is only two classical hyperparameters, δ, to which the proposed method is less sensitive than others, and N_{min}, which is easy to set as a small fraction of the number of data.

The distribution of points $\boldsymbol{z}_k$ on two hyperspheres depends on the type of hybrid system (switched or piecewise) and the data dispersion. A more in-depth study of these issues is left as future work.

Chapter 4

Selective ℓ_1-norm minimization for sparsity optimization

ABSTRACT. *The first part of the chapter deals with the recovery of sparse solutions of underdetermined systems of linear equations. More precisely, we focus on the convex relaxation of the problem and propose a new iteratively reweighted scheme in order to improve the conditions under which this relaxation provides the sparsest solution. We prove the convergence of the new scheme and derive sufficient conditions for the convergence towards the sparsest solution. Experiments show that the new scheme significantly improves upon the previous approaches for compressive sensing. Then, the second part of the chapter shows how these results can be used in linear hybrid system identification. In particular, we follow the error sparsification approach described in Sect. 2.2.2 (page 21), where the new scheme allows us to relax the conditions on the data.*

THIS chapter considers a convex relaxation approach for the recovery of sparse solutions of underdetermined systems of linear equations, with an application to Problem 4 of hybrid system identification.

Problem 4. *Given a data set $\mathcal{D} = \{(\boldsymbol{x}_k, y_k)\}_{k=1}^N$ generated by a linear switched system, estimate the number of submodels s, the submodel parameter vectors $\boldsymbol{\theta}_j$, $j = 1, \dots, s$ and the switching sequence $\{q_k\}_{k=1}^N$.*

More particularly, we follow the error sparsification approach described in Sect. 2.2.2 (page 21), which iteratively estimates each parameter vector individually by maximizing the sparsity of the error vector. This sparse optimization problem is solved via its ℓ_1-norm convex relaxation. In order to improve the sparsity of the solution, the iteratively reweighted ℓ_1-norm minimization scheme developed in the compressive sensing literature [25], and summarized in Sect. 1.4, is used. Note that most of the results on the equivalence between the convex relaxation and the original problem were also developed in this field, see e.g., [30, 41, 33, 23].

This chapter proposes a new iterative method based on ℓ_1-norm minimization for the recovery of sparse solutions. As in [25], we consider a weighted form of the ℓ_1-norm convex relaxation of the problem. But, instead of updating the weights in a soft manner, we explicitly set a weight to zero at each iteration. The proposed scheme offers three major advantages when compared with the one of [25]: i) it converges in a finite number of steps, ii) theoretical guarantees of convergence towards the sparsest solution can be obtained, and iii) experiments showed that it allows the sparsest solution to be recovered in a larger range of sparsity level. Then, the advantages of this new sparsity enhancing scheme allow us to improve the framework described in Sect. 2.2.2 (page 21) for hybrid system identification.

We present in Sect. 4.1 the sparse recovery problem and its convex relaxation. The classical reweighted scheme is recalled in Sect. 4.1.1, while the proposed one is described in Sect. 4.1.2. Then, Sect. 4.2 presents its application to hybrid system identification. The chapter ends with numerical examples in Sect. 4.3. The results of this chapter have been published in [J2] (see the Author's bibliography, page XVIII).

Note that in Sect. 4.1, the vectors $\boldsymbol{x}, \boldsymbol{x}_0$ and $\boldsymbol{x}_1$, are generic vectors of $\mathbb{R}^d$ and do not represent data vectors of an estimation problem.

4.1 Sparse recovery

Consider an underdetermined system of linear equations, $\boldsymbol{Ax} = \boldsymbol{b}$, with a full row rank matrix $\boldsymbol{A} \in \mathbb{R}^{m\times d}$ and a non-zero vector $\boldsymbol{b} \in \mathbb{R}^m$, where $m \ll d$. We are interested in sparse solutions of this system, i.e., solutions with few nonzero components, which are specifically obtained by solving

$$\begin{aligned} \min_{\boldsymbol{x}\in\mathbb{R}^d} \ & \|\boldsymbol{x}\|_0 \,, \\ \text{s.t.} \quad & \boldsymbol{Ax} = \boldsymbol{b}, \end{aligned} \tag{4.1}$$

where the ℓ_0-pseudo norm is defined as $\|\boldsymbol{x}\|_0 = |\{i : x_i \neq 0\}|$. More precisely, we concentrate on instances of (4.1) with a unique minimizer and assume that the following assumption holds in the rest of the chapter.

Assumption 1. *Problem* (4.1) *has a unique minimizer.*

The following theorem shows that Assumption 1 holds in many cases.

Theorem 4. *(Uniqueness-Spark) [34]: If a system of linear equations $\boldsymbol{Ax} = \boldsymbol{b}$ has a solution $\boldsymbol{x}$ obeying*

$$\|\boldsymbol{x}\|_0 < \frac{spark(\boldsymbol{A})}{2},$$

where spark($\boldsymbol{A}$) is the smallest number of linearly-dependent columns of $\boldsymbol{A}$, this solution is necessarily the sparsest possible.

Even when having a unique solution, (4.1) remains a nonconvex optimization problem which is intractable for large d due to its combinatorial search nature. Nonetheless, it has been the focus of many works over the last decade in various fields and particularly in the context of compressive sensing, see e.g. [30, 41, 33, 23].

As discussed in [34] and references therein, a common alternative for (4.1) is to consider the convex relaxation based on the ℓ_1-norm. This leads to

$$\begin{aligned} \min_{\boldsymbol{x}\in\mathbb{R}^d} \ & \|\boldsymbol{x}\|_1 \,, \\ \text{s.t.} \quad & \boldsymbol{Ax} = \boldsymbol{b}. \end{aligned} \tag{4.2}$$

This problem, known as basis pursuit, is convex and can typically be solved efficiently [19]. Many results are available regarding the equivalence between problems (4.2) and (4.1). These often rely on properties of the matrix $\boldsymbol{A}$, such as the mutual coherence.

Definition 1. *(Mutual coherence) [35]: For any matrix* $\boldsymbol{A} = [\boldsymbol{A}_1, \ldots, \boldsymbol{A}_d] \in \mathbb{R}^{m \times d}$,

$$\mu = \mu(\boldsymbol{A}) = \max_{1 \le i,j \le d, i \ne j} \frac{\left|\boldsymbol{A}_i^\top \boldsymbol{A}_j\right|}{\|\boldsymbol{A}_i\|_2 \|\boldsymbol{A}_j\|_2} \tag{4.3}$$

is the mutual coherence of $\boldsymbol{A}$.

As shown in [34], for a column normalized matrix $\boldsymbol{A}$, i.e., $\|\boldsymbol{A}_i\|_2 = 1$, problems (4.2) and (4.1) are equivalent if, for a solution of $\boldsymbol{Ax} = \boldsymbol{b}$, the condition

$$\|\boldsymbol{x}\|_0 < \frac{1}{2}\left(1 + \frac{1}{\mu(\boldsymbol{A})}\right) \tag{4.4}$$

holds.

Remark 1. *In the case of an unnormalized matrix* $\boldsymbol{A}$*, a similar equivalence is obtained by considering a weighted version of* (4.2)*, i.e.,*

$$\begin{aligned} &\min_{\boldsymbol{x} \in \mathbb{R}^d} \|\boldsymbol{W}_A \boldsymbol{x}\|_1, \\ &s.t. \quad \boldsymbol{Ax} = \boldsymbol{b}, \end{aligned} \tag{4.5}$$

where $\boldsymbol{W}_A = diag(\|\boldsymbol{A}_1\|_2, \ldots, \|\boldsymbol{A}_d\|_2)$.

However, in many applications, the matrix $\boldsymbol{A}$ cannot be freely chosen and the sufficient condition (4.4) might be violated. Yet, the problem defined in (4.1) may have a unique solution $\boldsymbol{x}_0$, as stated by Theorem 4. Thus, since $1 + \frac{1}{\mu(\boldsymbol{A})} \le \text{spark}(\boldsymbol{A})$ [34], there is a range of problems with

$$\frac{1}{2}\left(1 + \frac{1}{\mu(\boldsymbol{A})}\right) \le \|\boldsymbol{x}_0\|_0 \le \frac{\text{spark}(\boldsymbol{A})}{2},$$

where $\boldsymbol{x}_0$ is the unique solution to (4.1). For these problems, recovering the sparsest solution is therefore a well-defined problem, but maybe not directly solvable through the ℓ_1-norm convex relaxation: a solution $\boldsymbol{x}_1$ to (4.2) may have more nonzero elements than $\boldsymbol{x}_0$.

4.1.1 The classical iteratively reweighted approach

The classical iteratively reweighted approach, briefly outlined in Sect. 1.4, is recalled here. In order to improve the sparsity of the solutions of (4.2), a reweighted ℓ_1-norm minimization scheme is proposed in [25]. At each iteration l, the following problem is solved:

$$\begin{aligned} \boldsymbol{x}_1^{(l)} \in \ &\arg\min_{\boldsymbol{x} \in \mathbb{R}^d} \|\boldsymbol{W}_l \boldsymbol{x}\|_1, \\ &\text{s.t.} \quad \boldsymbol{Ax} = \boldsymbol{b}, \end{aligned} \tag{4.6}$$

where $\boldsymbol{W}_l = \text{diag}\left([w_1^{(l)}, \ldots, w_d^{(l)}]^\top\right)$ is a weighting diagonal matrix which penalizes differently the entries of $\boldsymbol{x}$. At the first iteration, the weights are equal, i.e., $\boldsymbol{W}_1 = \boldsymbol{I}_d$. Then, $\boldsymbol{W}_l$ is updated with

$$w_i^{(l+1)} = \frac{1}{\left|x_{1i}^{(l)}\right| + \epsilon},$$

where the $w_i^{(l+1)}$ are the weights at the $(l+1)$th iteration, $x_{1i}^{(l)}$ is the ith element of the solution of (4.6) at the lth iteration and $\epsilon > 0$ is a parameter preventing a division by zero. Note that the choice of ϵ has an influence on the convergence of $\boldsymbol{x}_1^{(l)}$.

Important open issues, highlighted in [25], regarding this scheme are:

- *What are smart and robust rules for selecting the parameter ϵ?*
- *Under what conditions does the algorithm converge?*

Though preliminary results on the convergence are given in [105], these rely on a condition involving both the matrix $\boldsymbol{A}$ and the solution to (4.1). Another convergence analysis of the reweighted ℓ_1-minimization by introducing the concept of Range Space Property (RSP) of $\boldsymbol{A}$ is given in [106].

The following subsection presents another reweighting mechanism with a convergence analysis relying only on conditions on $\boldsymbol{A}$.

Note that other methods without reweighting have been proposed to recover sparse solutions. In [67], the sparsity of the solution $\boldsymbol{x}_1$ to the ℓ_1-norm minimization is improved by setting to zero a maximal number of variables x_{1i} with smallest absolute value, subject to $\boldsymbol{A}\boldsymbol{x}_1 = \boldsymbol{b}$. In [93], a greedy algorithm selects one by one the columns of $\boldsymbol{A}$ such as to obtain the best ℓ_2-norm approximation of $\boldsymbol{b}$ at each iteration.

4.1.2 Selective ℓ_1-norm minimization

In this section, we propose a new method for updating the weighting matrix $\boldsymbol{W}_l$ in (4.6) in order to improve the sparsity of the solution. The new method, named Selective ℓ_1-norm Minimization (Sℓ_1M), is given in Algorithm 4.

Algorithm 4 Sℓ_1M

```
Require: $\boldsymbol{A} \in \mathbb{R}^{m\times d}$ and $\boldsymbol{b} \in \mathbb{R}^m$.
  - Initialize $l = 0$, $\boldsymbol{W}_1 = \boldsymbol{I}_d$.
  repeat
    - Set $l = l + 1$.
    - Get any $\boldsymbol{x}_1^{(l)}$ in the solution set of (4.6):
        $$\boldsymbol{x}_1^{(l)} \in \arg\min_{\boldsymbol{x}\in\mathbb{R}^d} \|\boldsymbol{W}_l\boldsymbol{x}\|_1, \quad \text{s.t.} \quad \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}.$$
    - Select the smallest index $q^{(l)} \in \arg\max_{i=1,\dots,d} w_i^{(l)}|x_{1i}^{(l)}|$.
    - Calculate $\boldsymbol{W}_{l+1}$ with $w_i^{(l+1)} = \begin{cases} w_i^{(l)}, & \text{if } i \neq q^{(l)}, \\ 0, & \text{if } i = q^{(l)}. \end{cases}$
  until $\left\|\boldsymbol{W}_l\boldsymbol{x}_1^{(l)}\right\|_\infty = 0$ or $\left\|\boldsymbol{W}_{l+1}\boldsymbol{x}_1^{(l)}\right\|_\infty = 0$.
  return $\boldsymbol{x}_1^* = \boldsymbol{x}_1^{(l)}$.
```

Algorithm 4 relaxes the optimization of the nonzero variables by setting their weights w_i to 0 in the cost function of (4.6), thus putting more weight on the other variables that are pulled towards 0. When the stopping criterion is met, we have $\|\boldsymbol{W}_l\boldsymbol{x}_1^{(l)}\|_0 \leq 1$. Hence, if it returns at iteration $l < d$, the algorithm yields a sparse solution.

The following provides an analysis of the proposed iterative scheme. The convergence in a finite number of steps is proved and a condition on the matrix $\boldsymbol{A}$ and the sparsity level guaranteeing the convergence towards the desired solution is derived.

Convergence in a finite number of steps

The following theorem guarantees that the algorithm Sℓ_1M converges in a finite number of steps.

Theorem 5. *The solution $\boldsymbol{x}_1^*$, returned by Algorithm 4, is found in at most $m+1$ iterations and $\|\boldsymbol{x}_1^*\|_0 \leq m$, where m is the number of rows in $\boldsymbol{A}$.*

For the proof of Theorem 5, we first need the following notation and lemmas.

For a vector $\boldsymbol{x} \in \mathbb{R}^d$ and a set of index $T \subset \{1, \dots, d\}$, we denote $\boldsymbol{x}_T$ the subvector of $\boldsymbol{x}$ containing its components of indexes in T and $\|\boldsymbol{x}_T\|_1 = \sum_{i \in T} |x_i|$.

Lemma 1. *Given a solution $\boldsymbol{x}_1$ of*

$$\begin{aligned} &\min_{\boldsymbol{x} \in \mathbb{R}^d} \|\boldsymbol{x}_T\|_1, \\ &s.t. \quad \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \end{aligned} \tag{4.7}$$

if $x_{1i} \neq 0$ for $i \in T$, then, with $\tilde{\boldsymbol{x}}_1$ a solution of

$$\begin{aligned} &\min_{\boldsymbol{x} \in \mathbb{R}^d} \left\|\boldsymbol{x}_{\tilde{T}}\right\|_1, \\ &s.t. \quad \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \end{aligned} \tag{4.8}$$

where $\tilde{T} = T \backslash i$, we have $\tilde{x}_{1i} \neq 0$.

Proof. We know that $\|\boldsymbol{x}_{1T}\|_1 \leq \|\tilde{\boldsymbol{x}}_{1T}\|_1$ or equivalently $\left\|\boldsymbol{x}_{1\tilde{T}}\right\|_1 + |x_{1i}| \leq \left\|\tilde{\boldsymbol{x}}_{1\tilde{T}}\right\|_1 + |\tilde{x}_{1i}|$ while $\left\|\boldsymbol{x}_{1\tilde{T}}\right\|_1 \geq \left\|\tilde{\boldsymbol{x}}_{1\tilde{T}}\right\|_1$. Therefore $|\tilde{x}_{1i}| \geq |x_{1i}| > 0$. $\square$

Lemma 2. *For $\boldsymbol{b} \neq \boldsymbol{0}$, after l iterations, if $\left\|\boldsymbol{W}_l \boldsymbol{x}_1^{(l)}\right\|_\infty \neq 0$, Algorithm 4 leads to a solution $\boldsymbol{x}_1^{(l)}$ such that $\left|x_{1q^{(i)}}^{(l)}\right| \neq 0$, $\forall i = \{1, \dots, l\}$.*

Proof. The proof is a direct consequence of Lemma 1 and the fact that $\left\|\boldsymbol{W}_1 \boldsymbol{x}_1^{(1)}\right\|_\infty = \left|x_{1q^{(1)}}^{(1)}\right| \neq 0$ when $\boldsymbol{b} \neq \boldsymbol{0}$. $\square$

Lemma 3. *In Algorithm 4, if $\forall l \leq m$ and $l \geq 2$, $\left|x_{1q^{(l)}}^{(l)}\right| \neq 0$, then the columns $\boldsymbol{A}_{q^{(i)}}$, $i = 1, \dots, l$, of the full row rank matrix $\boldsymbol{A}$ are linearly independent.*

Proof. To prove this Lemma, we show that at the jth iteration, $\forall j \geq 2$, if $\left|x_{1q^{(l)}}^{(l)}\right| \neq 0, \forall l \leq j$, then $\boldsymbol{A}_{q^{(j)}}$ is linearly independent with $\left\{\boldsymbol{A}_{q^{(l)}}\right\}_{l=1}^{j-1}$. Assume that this is not true, i.e., $\boldsymbol{A}_{q^{(j)}} = \sum_{l=1}^{(j-1)} \beta_l \boldsymbol{A}_{q^{(l)}}$. Then,

$$x_{1q^{(j)}}^{(j)} \sum_{l=1}^{j-1} \beta_l \boldsymbol{A}_{q^{(l)}} = x_{1q^{(j)}}^{(j)} \boldsymbol{A}_{q^{(j)}}. \tag{4.9}$$

On the other hand,

$$\boldsymbol{b} = \boldsymbol{A}\boldsymbol{x}_1^{(j)} = \sum_{l=1}^{j-1} x_{1q^{(l)}}^{(j)} \boldsymbol{A}_{q^{(l)}} + x_{1q^{(j)}}^{(j)} \boldsymbol{A}_{q^{(j)}} + \sum_{i \notin \left\{q^{(l)}\right\}_{l=1}^{j}} x_{1i}^{(j)} \boldsymbol{A}_i.$$

By using (4.9), we get another solution $\boldsymbol{x}^{*(j)}$ of $\boldsymbol{b} = \boldsymbol{A}\boldsymbol{x}$ whose elements are identified via the following expression

$$\boldsymbol{b} = \sum_{l=1}^{j-1} \left(x_{1q^{(l)}}^{(j)} + x_{1q^{(j)}}^{(j)} \beta_l \right) \boldsymbol{A}_{q^{(l)}} + 0\boldsymbol{A}_{q^{(j)}} + \sum_{i \notin \left\{q^{(l)}\right\}_{l=1}^{j}} x_{1i}^{(j)} \boldsymbol{A}_i = \boldsymbol{A}\boldsymbol{x}^{*(j)}.$$

Moreover,

$$\begin{aligned} |x_{1q^{(j)}}^{(j)}| + \sum_{i \notin \left\{q^{(l)}\right\}_{l=1}^{j}} \left| x_{1i}^{(j)} \right| &= \left\| \boldsymbol{W}_j \boldsymbol{x}_1^{(j)} \right\|_1 \\ &> \left\| \boldsymbol{W}_j \boldsymbol{x}^{*(j)} \right\|_1 = \sum_{i \notin \left\{q^{(l)}\right\}_{l=1}^{j}} \left| x_{1i}^{(j)} \right| \end{aligned}$$

This contradicts with the definition of $\boldsymbol{x}_1^{(j)}$ by (4.6). We conclude that $\boldsymbol{A}_{q^{(j)}}$ is linearly independent of $\left\{ \boldsymbol{A}_{q^{(l)}} \right\}_{l=1}^{j-1}$. □

Now we present the proof of Theorem 5.

Proof. If Algorithm 4 does not converge after m iterations, Lemma 2 implies that $\left| x_{1q^{(l)}}^{(l)} \right| \neq 0, \forall l \leq m$. Then, according to Lemma 3, the columns $\boldsymbol{A}_{q^{(l)}}, l \in \{1, \ldots, m\}$, of $\boldsymbol{A}$ are linearly independent and $\boldsymbol{b} \in \mathbb{R}^m$ can be expressed as a linear combination of these columns. Therefore, the minimum value of the sum $\sum_{i \notin \left\{q^{(l)}\right\}_{l=1}^{m}} |x_{1i}|$ is 0. Thus, the solution $\boldsymbol{x}_1^{(m+1)}$ to

$$\begin{aligned} &\min_{\boldsymbol{x} \in \mathbb{R}^d} \left\| \boldsymbol{W}_{(m+1)} \boldsymbol{x} \right\|_1, \\ &\text{s.t.} \quad \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \end{aligned}$$

satisfies $\left\| \boldsymbol{W}_{(m+1)} \boldsymbol{x}_1^{(m+1)} \right\|_\infty = 0$ which leads to the convergence of the algorithm at iteration $m+1$.

In addition, if Algorithm 4 converges at the lth iteration, $\left\| \boldsymbol{x}_1^{(l+1)} \right\|_0 \leq l - 1$. Therefore $\|\boldsymbol{x}_1^*\|_0 \leq m$. □

It is worth noting that if $\|\boldsymbol{x}_1^*\|_0 < \frac{\text{spark}(\boldsymbol{A})}{2}$, according to Theorem 4, $\boldsymbol{x}_1^*$ is the sparsest solution of (4.1).

Convergence towards the sparsest solution

In order for the iterative algorithm to converge to the solution $\boldsymbol{x}_0$ of (4.1) under Assumption 1, we need to ensure that the variables removed from the weighted sum in the cost function of (4.6) correspond to nonzeros in $\boldsymbol{x}_0$. The following proposition provides a sufficient condition under which the choice of the index $q^{(l)}$ in Algorithm 4 corresponds to a nonzero element in the solution of (4.1), $\boldsymbol{x}_0$, for which we define the two sets

$$I_0 = \{i : x_{0i} = 0\}, \quad I_1 = \{i : x_{0i} \neq 0\}. \tag{4.10}$$

Proposition 2. *If* $\left\| \boldsymbol{x}_0 - \boldsymbol{x}_1^{(l)} \right\|_1 < \frac{\mu+1}{2\mu} \left\| \boldsymbol{W}_l \boldsymbol{x}_0 \right\|_\infty$, *where* μ *is defined by Definition 1 with a column normalized matrix* $\boldsymbol{A}$, *then* $q^{(l)} \in I_1$ *with* $q^{(l)} = \arg\max_i w_i^{(l)} \left| x_{1i}^{(l)} \right|$.

For the proof of Proposition 2, we need the following lemma, see e.g. [34, 44].

Lemma 4. *For any $\boldsymbol{\delta} \in \mathbb{R}^d$ and $\boldsymbol{A} \in \mathbb{R}^{m\times d}$ such that $\boldsymbol{A}\boldsymbol{\delta} = \mathbf{0}$ and all columns of $\boldsymbol{A}$ are unit vectors, the following bound holds:*

$$|\delta_i| \le \frac{\mu}{\mu+1}\|\boldsymbol{\delta}\|_1,$$

where μ is the mutual coherence of $\boldsymbol{A}$.

Proof. If $\boldsymbol{A}\boldsymbol{\delta} = \mathbf{0}$, then $\boldsymbol{A}^\top\boldsymbol{A}\boldsymbol{\delta} = \mathbf{0}$ and $\left(\boldsymbol{A}^\top\boldsymbol{A} - \boldsymbol{I}_d\right)\boldsymbol{\delta} = -\boldsymbol{\delta}$. Thus,

$$\begin{aligned} |\delta_i| &= \left|\sum_{j=1}^{d} \delta_j \left(\boldsymbol{A}^\top\boldsymbol{A} - \boldsymbol{I}_d\right)_{i,j}\right| \\ &\le \sum_{j=1}^{d} |\delta_j| \left|\left(\boldsymbol{A}^\top\boldsymbol{A} - \boldsymbol{I}_d\right)_{i,j}\right| \\ &\le \sum_{j=1}^{d} \mu\,|\delta_j| - \mu\,|\delta_i|\,. \end{aligned}$$

Rearranging the terms yields the sought statement. □

Now we present the proof of Proposition 2.

Proof. Assume $q^{(l)} \notin I_1$, $x_{0q^{(l)}} = 0$, then $\left|\left(\boldsymbol{x}_0 - \boldsymbol{x}_1^{(l)}\right)_{q^{(l)}}\right| = |x_{1q^{(l)}}^{(l)}| = \max_i w_i^{(l)}\left|x_{1i}^{(l)}\right|$.

Let $j = \arg\max_i w_i^{(l)}|x_{0i}|$ and $|x_{0j}| = \|\boldsymbol{W}_l\boldsymbol{x}_0\|_\infty > \frac{2\mu}{\mu+1}\left\|\boldsymbol{x}_0 - \boldsymbol{x}_1^{(l)}\right\|_1$. Note that due to the assumption of Proposition 2, $\|\boldsymbol{W}_l\boldsymbol{x}_0\|_\infty > 0$. Then, since $\left|x_{1j}^{(l)}\right| \ge |x_{0j}| - \left|\left(\boldsymbol{x}_0 - \boldsymbol{x}_1^{(l)}\right)_j\right|$, Lemma 4 applied to $\boldsymbol{\delta} = \boldsymbol{x}_0 - \boldsymbol{x}_1^{(l)}$ leads to $\left|x_{1j}^{(l)}\right| > \frac{\mu}{\mu+1}\left\|\boldsymbol{x}_0 - \boldsymbol{x}_1^{(l)}\right\|_1$. Thus

$$\left|x_{1q^{(l)}}^{(l)}\right| = \left|\left(\boldsymbol{x}_0 - \boldsymbol{x}_1^{(l)}\right)_{q^{(l)}}\right| \ge \left|x_{1j}^{(l)}\right| > \frac{\mu}{\mu+1}\left\|\boldsymbol{x}_0 - \boldsymbol{x}_1^{(l)}\right\|_1. \tag{4.11}$$

On the other hand, Lemma 4 implies that $\left|(\boldsymbol{x}_0 - \boldsymbol{x}_1^{(l)})_{q^{(l)}}\right| \le \frac{\mu}{\mu+1}\left\|\boldsymbol{x}_0 - \boldsymbol{x}_1^{(l)}\right\|_1$, which contradicts (4.11). Therefore, the assumption $q^{(l)} \notin I_1$ is wrong and we conclude that $q^{(l)} \in I_1$. □

Influence of the weighting matrix on sparsity recovery

The following lemma, see e.g. [54, 8], shows that, with a good choice of $\boldsymbol{W}$ such that $w_i = 0, i \in I_1$, solving problem (4.6) gives exactly $\boldsymbol{x}_0$.

Lemma 5. *Given a diagonal matrix $\boldsymbol{W}$, with entries $w_i \ge 0$, if for all nonzero $\boldsymbol{\delta} \in \mathbb{R}^d$ such that $\boldsymbol{A}\boldsymbol{\delta} = \mathbf{0}$ the following condition holds*

$$\sum_{i\in I_1} w_i|\delta_i| < \sum_{i\in I_0} w_i|\delta_i|, \tag{4.12}$$

where I_0 and I_1 are defined by (4.10), then the solution $\boldsymbol{x}_0$ to (4.1) under Assumption 1 uniquely solves problem (4.6), i.e., $\boldsymbol{x}_0 = \arg\min_{\boldsymbol{x}\in\mathbb{R}^d}\|\boldsymbol{W}\boldsymbol{x}\|_1 \quad s.t. \quad \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$.

Proof. $\boldsymbol{x}_0$ uniquely solves problem (4.6) if for all nonzero $\boldsymbol{\delta} \in \mathbb{R}^d$ such that $\boldsymbol{A\delta} = \boldsymbol{0}$,

$$\|\boldsymbol{W}\boldsymbol{x}_0\|_1 < \|\boldsymbol{W}(\boldsymbol{x}_0 + \boldsymbol{\delta})\|_1$$

or, equivalently, if

$$\sum_{i \in I_1} w_i (|x_{0i}| - |x_{0i} + \delta_i|) < \sum_{i \in I_0} w_i |\delta_i|,$$

which is implied by the condition (4.12) since $\sum_{i \in I_1} w_i (|x_{0i}| - |x_{0i} + \delta_i|) \leq \sum_{i \in I_1} w_i |\delta_i|$. □

If the condition in Proposition 2 is satisfied for l_0 iterations, the condition (4.12) is relaxed to

$$\sum_{i \in I_1} |\delta_i| - \sum_{i \in \{q^{(l)}\}_{l=1}^{l_0}} |\delta_i| < \sum_{i \in I_0} |\delta_i|, \tag{4.13}$$

since all nonzero weights w_i are equal to 1 in Algorithm 4.

When this condition is satisfied, the solution of (4.6) with $\boldsymbol{W}_l = \boldsymbol{W}_{l_0+1}$ is the sparsest vector $\boldsymbol{x}_0$ thanks to Lemma 5. We see that the left-hand side of (4.13) can decrease to zero after $|I_1|$ iterations. This also shows that the algorithm converges after at most $|I_1| + 1$ iterations if the indexes $q^{(l)}$ are all well-chosen within I_1.

Sparse recovery condition

We now show, in Theorem 7, a condition on the matrix $\boldsymbol{A}$ and the sparsity level which can guarantee the convergence towards the desired solution. But this requires a few additional definitions.

Definition 2. *(Definition 1 in [41]) A matrix $\boldsymbol{A} \in \mathbb{C}^{m \times d}$ is said to satisfy the Null Space Property (NSP) of order k_0 with constant $\gamma \in (0,1)$ if*

$$\|\boldsymbol{\delta}_T\|_1 \leq \gamma \|\boldsymbol{\delta}_{T^c}\|_1,$$

for all sets $T \subset \{1, \dots, d\}$ with $|T| \leq k_0$, $T^c = \{1, \dots, d\} \backslash T$ and for all $\boldsymbol{\delta} \in Ker(\boldsymbol{A})$.

Definition 3. *The set of k_0-sparse vectors is defined as $\Sigma_{k_0} := \{\boldsymbol{x} \in \mathbb{R}^d : \|\boldsymbol{x}\|_0 \leq k_0\}$.*

Definition 4. *(taken from [41]) The best* k_0-term approximation error *in terms of ℓ_p-norm of a vector $\boldsymbol{x}$ is defined as*

$$\sigma_{k_0}(\boldsymbol{x})_p = \inf_{\boldsymbol{z} \in \Sigma_{k_0}} \|\boldsymbol{x} - \boldsymbol{z}\|_p.$$

When the matrix $\boldsymbol{A} \in \mathbb{C}^{m \times d}$ satisfies the null space property (NSP) of order k_0 with constant $\gamma \in (0,1)$, the following theorem gives an error bound between $\boldsymbol{x}_0$ and a solution $\boldsymbol{x}_1$ of the ℓ_1-minimization problem (4.2).

Theorem 6. *(Theorem 1 in [41]) Let $\boldsymbol{A} \in \mathbb{C}^{m \times d}$ be a matrix that satisfies the NSP of order k_0 with constant $\gamma \in (0,1)$. Let $\boldsymbol{x}_0 \in \mathbb{C}^d$ and $\boldsymbol{b} = \boldsymbol{A}\boldsymbol{x}_0$ and let $\boldsymbol{x}_1$ be a solution of the ℓ_1-minimization problem (4.2). Then*

$$\|\boldsymbol{x}_0 - \boldsymbol{x}_1\|_1 \leq \frac{2(1+\gamma)}{1-\gamma} \sigma_{k_0}(\boldsymbol{x}_0)_1. \tag{4.14}$$

In particular, if $\boldsymbol{x}_0$ is k_0-sparse then $\boldsymbol{x}_1 = \boldsymbol{x}_0$.

Therefore, when $\boldsymbol{x}_0$ is (k_0+h)-sparse with $h>0$, the NSP of $\boldsymbol{A}$ cannot guarantee that the ℓ_1-minimization problem (4.2) gives the true solution $\boldsymbol{x}_0$. The following theorem extends Theorem 6 to cases where $h>0$ and provides a sparse recovery condition for the iterative Algorithm 4.

Theorem 7. *Given a matrix $\boldsymbol{A}$ that satisfies the NSP of order k_0 with constant $\gamma \in (0, \frac{1}{2})$, if $\boldsymbol{x}_0$ is a (k_0+h)-sparse vector with $h \in \mathbb{N}^*$, such that*

$$\gamma < \frac{1-(4h-1)\mu}{1+(4h+1)\mu}, \tag{4.15}$$

$$h \le k_0, \tag{4.16}$$

where μ is defined as in (4.3)*, Algorithm 4 converges to $\boldsymbol{x}_0$ in at most $h+1$ iterations.*

Proof. The proof is decomposed in two main steps:

1. showing that $q^{(l)} \in I_1$, $\forall l \le h$,
2. showing that, after Step 1, the algorithm converges to the unique solution $\boldsymbol{x}_0$ at iteration $h+1$.

Step 1. First, we will prove by induction that $\forall l \le h$,

$$\left\| \boldsymbol{x}_0 - \boldsymbol{x}_1^{(l)} \right\|_1 < \frac{\mu+1}{2\mu} \left\| \boldsymbol{W}_l \boldsymbol{x}_0 \right\|_\infty, \tag{4.17}$$

from which we will use Proposition 2 to conclude that $q^{(l)} \in I_1$, $\forall l \le h$.

To prove (4.17), we follow a path similar to that of the proof of Theorem 1 in [41]. We define the sets $T_q^{(j)} = \left\{q^{(i)}\right\}_{i=1}^{j-1}$, $T_s^{(j)} = \{1, \dots, d\} \setminus T_q^{(j)}$, $T^{(j)}$ as the set of index of k_0 entries of $\boldsymbol{x}_0$ with largest magnitude such that $T^{(j)} \cap T_q^{(j)} = \emptyset$ and $T_r^{(j)} = \{1, \dots, d\} \setminus \left\{T^{(j)} \cup T_q^{(j)}\right\}$.

Let $\boldsymbol{\delta}^{(j)} = \boldsymbol{x}_0 - \boldsymbol{x}_1^{(j)}$. We have

$$\left\| \boldsymbol{x}_0 - \boldsymbol{x}_1^{(l)} \right\|_1 = \left\| \boldsymbol{\delta}^{(l)} \right\|_1 = \left\| \boldsymbol{\delta}_{T^{(l)}}^{(l)} \right\|_1 + \left\| \boldsymbol{\delta}_{T_q^{(l)}}^{(l)} \right\|_1 + \left\| \boldsymbol{\delta}_{T_r^{(l)}}^{(l)} \right\|_1. \tag{4.18}$$

The first term on the right-hand side of (4.18) can be bounded as follows. By the fact that $\boldsymbol{A}$ satisfies the NSP as in Definition 2 and that $\boldsymbol{\delta}^{(l)} \in \operatorname{Ker}(\boldsymbol{A})$, the definitions of $T^{(l)}$ and $T_r^{(l)}$ imply

$$\left\| \boldsymbol{\delta}_{T^{(l)}}^{(l)} \right\|_1 \le \gamma \left(\left\| \boldsymbol{\delta}_{T_q^{(l)}}^{(l)} \right\|_1 + \left\| \boldsymbol{\delta}_{T_r^{(l)}}^{(l)} \right\|_1 \right), \tag{4.19}$$

where we used $\left\| \boldsymbol{\delta}_{T_q^{(l)} \cup T_r^{(l)}}^{(l)} \right\|_1 = \left\| \boldsymbol{\delta}_{T_q^{(l)}}^{(l)} \right\|_1 + \left\| \boldsymbol{\delta}_{T_r^{(l)}}^{(l)} \right\|_1$.

Therefore, by using (4.19) in (4.18), $\left\| \boldsymbol{\delta}^{(l)} \right\|_1$ is bounded by

$$\left\| \boldsymbol{\delta}^{(l)} \right\|_1 \le (1+\gamma) \left(\left\| \boldsymbol{\delta}_{T_q^{(l)}}^{(l)} \right\|_1 + \left\| \boldsymbol{\delta}_{T_r^{(l)}}^{(l)} \right\|_1 \right). \tag{4.20}$$

Now, we bound the second term of the right-hand side of (4.20) as follows.

For $l \geq 2$, we have

$$\begin{aligned}
\|\boldsymbol{x}_{0T^{(l)}}\|_1 + \left\|\boldsymbol{x}_{0T_q^{(l)}}\right\|_1 + \left\|\boldsymbol{x}_{0T_r^{(l)}}\right\|_1 &= \|\boldsymbol{x}_0\|_1 \\
&\geq \left\|\boldsymbol{x}_1^{(1)}\right\|_1 \geq \left\|\boldsymbol{x}_{1T_s^{(2)}}^{(2)}\right\|_1 + \left|x_{1q^{(1)}}^{(1)}\right| \\
&\geq \left\|\boldsymbol{x}_{1T_s^{(3)}}^{(3)}\right\|_1 + \left|x_{1q^{(1)}}^{(1)}\right| + \left|x_{1q^{(2)}}^{(2)}\right| \\
&\geq \cdots \geq \left\|\boldsymbol{x}_{1T_s^{(l)}}^{(l)}\right\|_1 + \sum_{i=1}^{l-1} \left|x_{1q^{(i)}}^{(i)}\right| \\
&= \left\|\boldsymbol{x}_{1T^{(l)}}^{(l)}\right\|_1 + \left\|\boldsymbol{x}_{1T_r^{(l)}}^{(l)}\right\|_1 + \sum_{i=1}^{l-1} \left|x_{1q^{(i)}}^{(i)}\right|. \qquad (4.21)
\end{aligned}$$

Then, by using the triangle inequalities

$$\begin{aligned}
\left\|\boldsymbol{x}_{1T^{(l)}}^{(l)}\right\|_1 &\geq \|\boldsymbol{x}_{0T^{(l)}}\|_1 - \left\|\boldsymbol{\delta}_{T^{(l)}}^{(l)}\right\|_1, \\
\left\|\boldsymbol{x}_{1T_r^{(l)}}^{(l)}\right\|_1 &\geq \left\|\boldsymbol{\delta}_{T_r^{(l)}}^{(l)}\right\|_1 - \left\|\boldsymbol{x}_{0T_r^{(l)}}\right\|_1,
\end{aligned}$$

in the right-hand side of (4.21), we have

$$\|\boldsymbol{x}_{0T^{(l)}}\|_1 + \left\|\boldsymbol{x}_{0T_q^{(l)}}\right\|_1 + \left\|\boldsymbol{x}_{0T_r^{(l)}}\right\|_1 \geq \|\boldsymbol{x}_{0T^{(l)}}\|_1 - \left\|\boldsymbol{\delta}_{T^{(l)}}^{(l)}\right\|_1 + \left\|\boldsymbol{\delta}_{T_r^{(l)}}^{(l)}\right\|_1 - \left\|\boldsymbol{x}_{0T_r^{(l)}}\right\|_1 + \sum_{i=1}^{l-1} \left|x_{1q^{(i)}}^{(i)}\right|.$$

Therefore, by keeping the term with $\boldsymbol{\delta}_{T_r^{(l)}}^{(l)}$ on the left-hand side,

$$\left\|\boldsymbol{\delta}_{T_r^{(l)}}^{(l)}\right\|_1 \leq \left\|\boldsymbol{\delta}_{T^{(l)}}^{(l)}\right\|_1 + 2\left\|\boldsymbol{x}_{0T_r^{(l)}}\right\|_1 + \left\|\boldsymbol{x}_{0T_q^{(l)}}\right\|_1 - \sum_{i=1}^{l-1} \left|x_{1q^{(i)}}^{(i)}\right|. \qquad (4.22)$$

The second term of the right-hand side of (4.22) can be computed as a best k_0-term approximation error $\sigma_{k_0}(\boldsymbol{W}_l\boldsymbol{x}_0)_1$ as defined in Definition 4. The entries of $\boldsymbol{z}^* = \arg\inf_{\boldsymbol{z}\in\Sigma_{k_0}} \|\boldsymbol{W}_l\boldsymbol{x}_0 - \boldsymbol{z}\|_1$ belong to 3 subvectors and are given as follows. $\boldsymbol{z}^*_{T_q^{(l)}} = \boldsymbol{0}$ because $T_q^{(l)}$ contains the indexes of zeros on the diagonal of $\boldsymbol{W}_l$ and $(\boldsymbol{W}_l\boldsymbol{x}_0)_{T_q^{(l)}} = \boldsymbol{0}$. $\boldsymbol{z}^*_{T^{(l)}} = \boldsymbol{x}_{0T^{(l)}}$ because $T^{(l)}$ contains the indexes of the components of $\boldsymbol{W}_l\boldsymbol{x}_0$ with largest magnitude. $\boldsymbol{z}^*_{T_r^{(l)}} = \boldsymbol{0}$ because $|T^{(l)}| = k_0$ and $\boldsymbol{z}^* \in \Sigma_{k_0}$.

Thus, $\sigma_{k_0}(\boldsymbol{W}_l\boldsymbol{x}_0)_1 = \|\boldsymbol{W}_l\boldsymbol{x}_0 - \boldsymbol{z}^*\|_1 = \left\|(\boldsymbol{W}_l\boldsymbol{x}_0)_{T_r^{(l)}}\right\|_1$, and, since $T_r^{(l)} \cap T_q^{(l)} = \emptyset$ and the entries on the diagonal of $\boldsymbol{W}_l$ with indexes in $T_r^{(l)}$ are equal to 1, we have

$$\left\|\boldsymbol{x}_{0T_r^{(l)}}\right\|_1 = \sigma_{k_0}(\boldsymbol{W}_l\boldsymbol{x}_0)_1. \qquad (4.23)$$

For the two last terms on the right-hand side of (4.22), the triangle inequality $\left\|\boldsymbol{x}_{0T_q^{(l)}}\right\|_1 - \left\|\boldsymbol{x}_{1T_q^{(l)}}\right\|_1 \leq \left\|(\boldsymbol{x}_0 - \boldsymbol{x}_1)_{T_q^{(l)}}\right\|_1$ can be written as

$$\left\|\boldsymbol{x}_{0T_q^{(l)}}\right\|_1 - \sum_{i=1}^{l-1} \left|x_{1q^{(i)}}^{(i)}\right| \leq \sum_{i=1}^{l-1} \left|\delta_{q^{(i)}}^{(i)}\right|. \qquad (4.24)$$

Then, introducing (4.19), (4.23) and (4.24) in (4.22) gives

$$\left\|\boldsymbol{\delta}^{(l)}_{T_r^{(l)}}\right\|_1 \leq \gamma \left(\left\|\boldsymbol{\delta}^{(l)}_{T_q^{(l)}}\right\|_1 + \left\|\boldsymbol{\delta}^{(l)}_{T_r^{(l)}}\right\|_1\right) + 2\sigma_{k_0}\left(\boldsymbol{W}_l\boldsymbol{x}_0\right)_1 + \sum_{i=1}^{l-1}\left|\delta^{(i)}_{q^{(i)}}\right|,$$

which can be rewritten by keeping all the terms with $\boldsymbol{\delta}^{(l)}_{T_r^{(l)}}$ on the left-hand side as

$$\left\|\boldsymbol{\delta}^{(l)}_{T_r^{(l)}}\right\|_1 \leq \frac{\gamma}{(1-\gamma)}\left\|\boldsymbol{\delta}^{(l)}_{T_q^{(l)}}\right\|_1 + \frac{2}{(1-\gamma)}\sigma_{k_0}\left(\boldsymbol{W}_l\boldsymbol{x}_0\right)_1 + \frac{1}{(1-\gamma)}\sum_{i=1}^{l-1}\left|\delta^{(i)}_{q^{(i)}}\right|. \tag{4.25}$$

Thus, introducing (4.25) in (4.20) leads to

$$\left\|\boldsymbol{\delta}^{(l)}\right\|_1 \leq \frac{2(1+\gamma)}{(1-\gamma)}\sigma_{k_0}\left(\boldsymbol{W}_l\boldsymbol{x}_0\right)_1 + \frac{(1+\gamma)}{(1-\gamma)}\sum_{i=1}^{l-1}\left|\delta^{(i)}_{q^{(i)}}\right| + \frac{(1+\gamma)}{(1-\gamma)}\left\|\boldsymbol{\delta}^{(l)}_{T_q^{(l)}}\right\|_1. \tag{4.26}$$

By applying Lemma 4 to $\boldsymbol{\delta}^{(j)}, \forall j \in \{1, \dots, l\}$, we have

$$\forall i \in \{1, \dots, d\}, \quad \left|\delta^{(j)}_i\right| \leq \frac{\mu}{\mu+1}\|\boldsymbol{\delta}^{(j)}\|_1.$$

With $j = i, i = 1 \dots l-1$, this gives

$$\sum_{i=1}^{l-1}\left|\delta^{(i)}_{q^{(i)}}\right| \leq \frac{\mu}{\mu+1}\sum_{i=1}^{l-1}\left\|\boldsymbol{\delta}^{(i)}\right\|_1 \tag{4.27}$$

and, with $j = l$,

$$\left\|\boldsymbol{\delta}^{(l)}_{T_q^{(l)}}\right\|_1 \leq (l-1)\frac{\mu}{\mu+1}\left\|\boldsymbol{\delta}^{(l)}\right\|_1. \tag{4.28}$$

On the other hand, the condition (4.15) leads to

$$\frac{(1+\gamma)}{(1-\gamma)} < \frac{\mu+1}{4h\mu}. \tag{4.29}$$

Therefore, using the bounds (4.27), (4.28) and (4.29) in (4.26) leads to

$$\left\|\boldsymbol{\delta}^{(l)}\right\|_1 \leq \frac{\mu+1}{2h\mu}\sigma_{k_0}\left(\boldsymbol{W}_l\boldsymbol{x}_0\right)_1 + \frac{1}{4h}\sum_{i=1}^{l-1}\left\|\boldsymbol{\delta}^{(i)}\right\|_1 + \frac{(l-1)}{4h}\left\|\boldsymbol{\delta}^{(l)}\right\|_1,$$

and by keeping the terms with $\left\|\boldsymbol{\delta}^{(l)}\right\|_1$ on the left-hand side,

$$\left\|\boldsymbol{\delta}^{(l)}\right\|_1 \leq \frac{(\mu+1)}{\mu}\frac{2}{(4h-l+1)}\sigma_{k_0}\left(\boldsymbol{W}_l\boldsymbol{x}_0\right)_1 + \frac{1}{(4h-l+1)}\sum_{i=1}^{l-1}\left\|\boldsymbol{\delta}^{(i)}\right\|_1. \tag{4.30}$$

Now, we prove by induction that the inequality (4.17) holds $\forall l \in \{1, \dots, h\}$. It is rewritten as

$$\left\|\boldsymbol{\delta}^{(l)}\right\|_1 < \frac{\mu+1}{2\mu}\left\|\boldsymbol{W}_l\boldsymbol{x}_0\right\|_\infty. \tag{4.31}$$

Let denote $\bar{x}_{0j}(i)$ the ith largest absolute value of $\boldsymbol{W}_j\boldsymbol{x}_0$.

For $l = 1$, Theorem 6 yields

$$\left\|\boldsymbol{x}_0 - \boldsymbol{x}_1^{(1)}\right\|_1 \leq \frac{2(1+\gamma)}{1-\gamma}\sigma_{k_0}(\boldsymbol{x}_0)_1. \tag{4.32}$$

Since

$$\sigma_{k_0}(\boldsymbol{x}_0)_1 = \sigma_{k_0}(\boldsymbol{W}_1\boldsymbol{x}_0)_1 \leq h\bar{x}_{01}(k_0+1) \leq h\bar{x}_{01}(1) = h\left\|\boldsymbol{W}_1\boldsymbol{x}_0\right\|_\infty$$

and (4.29), the result (4.32) leads to

$$\left\|\boldsymbol{\delta}^{(1)}\right\|_1 < \frac{(\mu+1)}{2\mu}\|\boldsymbol{W}_1\boldsymbol{x}_0\|_\infty$$

So, (4.31) is true for $l = 1$. Now, by assuming that it is true until $l-1$ with $l \geq 2$, we prove that it is true for l. To do that, we need to bound the sum $\sum_{i=1}^{l-1}\left\|\boldsymbol{\delta}^{(i)}\right\|_1$ involved in (4.30). In particular, we prove that each term in the sum is bounded by

$$\left\|\boldsymbol{\delta}^{(j)}\right\|_1 < \frac{\mu+1}{2\mu}\left\|\boldsymbol{W}_l\boldsymbol{x}_0\right\|_\infty, \forall j \in \{1,\ldots,l-1\} \tag{4.33}$$

From (4.30) by replacing l by j, we have

$$\left\|\boldsymbol{\delta}^{(j)}\right\|_1 \leq \frac{(\mu+1)}{\mu}\frac{2}{(4h-j+1)}\sigma_{k_0}\left(\boldsymbol{W}_j\boldsymbol{x}_0\right)_1 + \frac{1}{(4h-j+1)}\sum_{i=1}^{j-1}\left\|\boldsymbol{\delta}^{(i)}\right\|_1. \tag{4.34}$$

To prove (4.33), we need to bound the terms with $\sigma_{k_0}\left(\boldsymbol{W}_j\boldsymbol{x}_0\right)_1$ involved in (4.34) by the terms with $\left\|\boldsymbol{W}_l\boldsymbol{x}_0\right\|_\infty$. Let denote

$$\boldsymbol{z}^* = \arg\inf_{\boldsymbol{z}\in\Sigma_{k_0}}\left\|\boldsymbol{W}_j\boldsymbol{x}_0 - \boldsymbol{z}\right\|_1.$$

Then

$$\sigma_{k_0}\left(\boldsymbol{W}_j\boldsymbol{x}_0\right)_1 = \inf_{\boldsymbol{z}\in\Sigma_{k_0}}\left\|\boldsymbol{W}_j\boldsymbol{x}_0 - \boldsymbol{z}\right\|_1 \leq (h-j+1)\left\|\boldsymbol{W}_j\boldsymbol{x}_0 - \boldsymbol{z}^*\right\|_\infty \leq (h-j+1)\bar{x}_{0j}(k_0+1). \tag{4.35}$$

where $\bar{x}_{0j}(i)$ denotes the ith largest absolute value of $\boldsymbol{W}_j\boldsymbol{x}_0$. On the other hand

$$\left\|\boldsymbol{W}_l\boldsymbol{x}_0\right\|_\infty \geq \bar{x}_{0j}(l-j+1) \tag{4.36}$$

since the zeros on the diagonal of $\boldsymbol{W}_j$ are included in $\boldsymbol{W}_l$. Therefore, if $l-j+1 \leq k_0+1$ or $l \leq k_0+1$ we have $\bar{x}_{0j}(l-j+1) \geq \bar{x}_{0j}(k_0+1)$, then by using (4.35) and (4.36),

$$\sigma_{k_0}\left(\boldsymbol{W}_j\boldsymbol{x}_0\right)_1 \leq (h-j+1)\left\|\boldsymbol{W}_l\boldsymbol{x}_0\right\|_\infty. \tag{4.37}$$

We now prove by induction (4.33).

For $j = 1$, using (4.32) with (4.29) and (4.37) leads to the following result

$$\left\|\boldsymbol{\delta}^{(1)}\right\|_1 < \frac{(\mu+1)}{2\mu}\|\boldsymbol{W}_l\boldsymbol{x}_0\|_\infty$$

So, (4.33) is true for $j = 1$. Now, by assuming that it is true until $j-1$ with $j \geq 2$, we prove that it is true for j.

By using (4.34) with (4.37), we have

$$\begin{aligned}
\left\|\boldsymbol{\delta}^{(j)}\right\|_1 \leq & \frac{(\mu+1)}{\mu}\frac{2}{(4h-j+1)}\sigma_{k_0}\left(\boldsymbol{W}_j\boldsymbol{x}_0\right)_1 + \frac{1}{(4h-j+1)}\sum_{i=1}^{j-1}\left\|\boldsymbol{\delta}^{(i)}\right\|_1 \\
\leq & \frac{(\mu+1)}{\mu}\frac{2\left(h-j+1\right)}{(4h-j+1)}\left\|\boldsymbol{W}_l\boldsymbol{x}_0\right\|_\infty + \frac{1}{(4h-j+1)}(j-1)\frac{(\mu+1)}{2\mu}\left\|\boldsymbol{W}_l\boldsymbol{x}_0\right\|_\infty \\
= & \frac{(4h-3j+3)(\mu+1)}{(4h-j+1)2\mu}\left\|\boldsymbol{W}_l\boldsymbol{x}_0\right\|_\infty \\
< & \frac{(\mu+1)}{2\mu}\left\|\boldsymbol{W}_l\boldsymbol{x}_0\right\|_\infty,
\end{aligned}$$

since $j \geq 2$.

So, (4.33) is true $\forall j \in \{1,\dots,l-1\}$.

Similar steps are used to complete the induction on l:

$$\begin{aligned}
\left\|\boldsymbol{\delta}^{(l)}\right\|_1 \leq & \frac{(\mu+1)}{\mu}\frac{2}{(4h-l+1)}\sigma_{k_0}\left(\boldsymbol{W}_l\boldsymbol{x}_0\right)_1 + \frac{1}{(4h-l+1)}\sum_{i=1}^{l-1}\left\|\boldsymbol{\delta}^{(i)}\right\|_1 \\
\leq & \frac{(\mu+1)}{\mu}\frac{2\left(h-l+1\right)}{(4h-l+1)}\left\|\boldsymbol{W}_l\boldsymbol{x}_0\right\|_\infty + \frac{1}{(4h-l+1)}(l-1)\frac{(\mu+1)}{2\mu}\left\|\boldsymbol{W}_l\boldsymbol{x}_0\right\|_\infty \\
= & \frac{(4h-3l+3)(\mu+1)}{(4h-l+1)2\mu}\left\|\boldsymbol{W}_l\boldsymbol{x}_0\right\|_\infty \\
< & \frac{(\mu+1)}{2\mu}\left\|\boldsymbol{W}_l\boldsymbol{x}_0\right\|_\infty
\end{aligned}$$

So, (4.31), or equivalently (4.17), holds $\forall l \in \{1,\dots,h\}$

Therefore, by using Proposition 2 we conclude that $q^{(l)} \in I_1$, $\forall l \leq h$ with $h \leq k_0$.

Step 2. Now, we prove that in the next iteration, the reweighted ℓ_1-norm minimization problem (4.6) gives the unique solution $\boldsymbol{x}_0$.

By using Lemma 5, $\boldsymbol{x}_0$ uniquely solves problem (4.6) if for all nonzero $\boldsymbol{\delta} \in \mathbb{R}^d$ such that $\boldsymbol{A\delta} = \boldsymbol{0}$, the following condition holds

$$\sum_{i\in I_1} w_i^{(h+1)}|\delta_i| < \sum_{i\in I_0} w_i^{(h+1)}|\delta_i|. \tag{4.38}$$

Indeed, the left-hand side of (4.38) can be rewritten as

$$\sum_{i\in I_1} w_i^{(h+1)}|\delta_i| = \sum_{i\in I_1\setminus T_q^{(h+1)}}|\delta_i| < \gamma\sum_{i\in I_0}|\delta_i| + \gamma\sum_{i\in T_q^{(h+1)}}|\delta_i| \tag{4.39}$$

since $w_i^{(h+1)} = 0, \forall i \in T_q^{(h+1)}$ and the NSP of order k_0 (Definition 2) is applied with $T = I_1 \setminus T_q^{(h+1)}$ and $|T| = k_0$. Then, we apply one more time the NSP of order k_0 with $T = T_q^{(h+1)}$ and $|T| = h \leq k_0$ for the right-hand side of (4.39) to get

$$\sum_{i\in I_1\setminus T_q^{(h+1)}}|\delta_i| < \gamma\sum_{i\in I_0}|\delta_i| + \gamma^2\sum_{i\in I_0}|\delta_i| + \gamma^2\sum_{i\in I_1\setminus T_q^{(h+1)}}|\delta_i|,$$

and by keeping the terms with $\sum_{i\in I_1\setminus T_q^{(h+1)}}|\delta_i|$ on the left-hand side,

$$\sum_{i\in I_1\setminus T_q^{(h+1)}}|\delta_i| < \frac{\gamma}{1-\gamma}\sum_{i\in I_0}|\delta_i|. \tag{4.40}$$

Thus the assumption $\gamma < \frac{1}{2}$ guarantees that

$$\sum_{i \in I_1 \setminus T_q^{(h+1)}} |\delta_i| < \sum_{i \in I_0} |\delta_i| \tag{4.41}$$

and that (4.38) holds, which implies that we get the unique solution $\boldsymbol{x}_0$ in $h+1$ iterations. □

Note that the NSP of $\boldsymbol{A}$ and the value of γ can be difficult to determine directly, but can be related to the Restricted Isometry Property (RIP) which can be easier to handle [41].

4.2 Hybrid system identification

We now come back to hybrid system identification in the form of Problem 4.

We take the error sparsification approach of [5] described in Sect. 2.2.2 (page 21), in which we replace the reweighted scheme of [25] described in Sect. 4.1.1 by the Sℓ_1M scheme proposed in Sect. 4.1.2. This yields Algorithm 5 for the estimation of a single parameter vector.

Algorithm 5 Estimation of a parameter vector from a data set

```
Require: A data set $\mathcal{D} = \{(\boldsymbol{x}_k, y_k)\}_{k=1}^N$ and a number of iterations $N_s$.
  - Initialize $l = 0$, $\boldsymbol{W} = \boldsymbol{I}_N$.
  while $l < N_s$ do
    - Set $l = l + 1$.
    - Solve
        $\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \|\boldsymbol{W}\boldsymbol{e}(\boldsymbol{\theta})\|_1,$
      where $\boldsymbol{e}(\boldsymbol{\theta}) = \left[y_1 - \boldsymbol{x}_1^\top\boldsymbol{\theta}, \dots, y_N - \boldsymbol{x}_N^\top\boldsymbol{\theta}\right]^\top$ is the error vector.
    - Select an index in the maximal absolute error set (break the tie arbitrarily if necessary):
        $q \in \arg\max_k w_k |e_k(\boldsymbol{\theta}^*)|.$
    - Set $w_q = 0$ in the matrix $\boldsymbol{W}$.
  end while
  return $\boldsymbol{\theta}^*$ and $\boldsymbol{W}$.
```

Algorithm 5 requires a number of iterations, N_s, in order to deal with noisy data (for which true sparsity cannot be obtained). However, with knowledge of the number of modes, s, we can set $N_s = \frac{s-1}{s}N$, since in this case the largest fraction of points of a mode in the data set is at least N/s.

As the bounded-error like methods described in Sect. 2.2.2, the identification procedure obtains the submodels one by one. After applying Algorithm 5 to estimate a parameter vector $\hat{\boldsymbol{\theta}}_j$, the data points verifying the error condition, $|y_k - \boldsymbol{x}_k^\top \hat{\boldsymbol{\theta}}_j| \leq \delta$ where δ is a fixed threshold, are associated to this submodel and removed from the data set. Then, the next parameter vectors are iteratively estimated from reduced data sets until all data points are removed, at which point the number of modes is finally estimated as the number of obtained submodels. Note that, if the noise is unbounded, e.g., Gaussian, the procedure should stop before that, i.e., when a small and predefined fraction of the data N_{min}/N

remains, to avoid creating irrelevant submodels for small groups of points corrupted by large noise terms.

This procedure is presented in Algorithm 6.

Algorithm 6 Complete identification procedure

```
Require: a data set $\mathcal{D} = \{(\boldsymbol{x}_k, y_k)\}_{k=1}^N$ and thresholds $\delta$ and $N_{min}$.
  Initialize $\mathcal{D}_{train} = \mathcal{D}$, $j = 0$.
  while $|\mathcal{D}_{train}| \geq N_{min}$ do
    - Set $j = j + 1$.
    - Estimate the parameter vector $\hat{\boldsymbol{\theta}}_j$ by applying Algorithm 5 to $\mathcal{D}_{train}$.
    - Assign data points of $\mathcal{D}_{train}$ to the $j$th estimated submodel,
          $$\mathcal{D}(\hat{\boldsymbol{\theta}}_j) = \left\{(\boldsymbol{x}_k, y_k) \in \mathcal{D}_{train} : \left|y_k - \boldsymbol{x}_k^\top \hat{\boldsymbol{\theta}}_j\right| \leq \delta\right\}.$$
    - Set $\mathcal{D}_{train} = \mathcal{D}_{train} \backslash \mathcal{D}(\hat{\boldsymbol{\theta}}_j)$.
  end while
  return the estimated number of modes $\hat{s} = j$ and the estimated parameter matrix
  $\hat{\boldsymbol{\Theta}} = [\hat{\boldsymbol{\theta}}_1, \dots, \hat{\boldsymbol{\theta}}_{\hat{s}}]$.
```

4.3 Numerical experiments

4.3.1 Compressive sensing example

We consider a classical example of sparse signal recovery used in many works to show the efficiency of the proposed iteratively reweighted method. The goal is to recover a sparse signal $\boldsymbol{x}$ of length d with $\|\boldsymbol{x}\|_0 = k_0$. The k_0 nonzero positions are chosen randomly, and the nonzero values are randomly drawn according to a zero-mean unit-variance Gaussian distribution. The matrix $\boldsymbol{A} \in \mathbb{R}^{m \times d}$ is a Gaussian matrix, i.e., with entries following a Gaussian distribution with expectation 0 and variance $1/m$. The sparsity level k_0 is increased from 10 to 60 to see the capacity of our method in signal recovery.

To compare with the classical reweighting method of Candès *et al.* in Sect. 4.1.1, we set the experiment as in [25] with $d = 256, m = 100$. For each value of k_0, we run 500 trials to estimate the probability of perfect signal recovery (be successful if $\|\boldsymbol{x}_0 - \hat{\boldsymbol{x}}_0\|_\infty \leq 10^{-3}$). Figure 4.1 reports the successful recovery probability, Pr(recovery), for the unweighted ℓ_1-norm minimization (Unweighted ℓ_1), the Orthogonal Matching Pursuit (OMP) [93], the reweighted scheme of [25] (Reweighted) with 8 iterations, and the proposed one (Sℓ_1M). We see that the requisite oversampling factor for perfect recovery [25],

$$\begin{aligned} ROF = &\min_{k_0} \ \frac{m}{k_0}, \\ &\text{s.t.} \quad Pr(recovery) = 1, \end{aligned}$$

decreases from approximately 4 for Unweighted ℓ_1 or 3 for the method of [25] with $\epsilon = 1$ to $100/40 = 2.5$ for our method. Moreover, in our method there is no hyperparameter to tune, whereas ϵ can influence the results for the classical scheme [25].

From Theorem 4 in Sect. 4.1, we know that a value of $k_0 < \text{spark}(\boldsymbol{A})/2 \leq (m+1)/2 = 50.5$ is sufficient to guarantee the uniqueness of the solution in (4.1). This explains why all methods have a small successful recovery probability with the sparsity level k_0 close to 50. Nonetheless, our method shows a successful recovery probability greater than 0.9

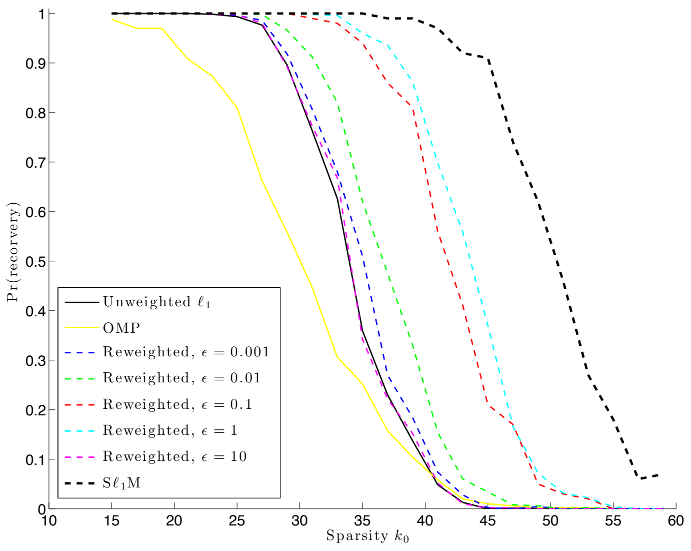


Figure 4.1: Empirical probability of successful recovery versus the sparsity level k_0 for the OMP method [93], the classical reweighting of [25] with various ϵ, and the proposed one (Sℓ_1M).

at $k_0 = 45$. We note that the successful recovery probability of the proposed method does not decrease at $k_0 = 59$. This may be due to the non uniqueness of the solution at these sparsity levels or be an artifact of the experiment using a limited number of trails to estimate the probability $Pr(recovery)$.

These improvements are paid for by the computational cost of the proposed method, which requires a larger number of iterations compared with [25]. This trade-off is consistent with the results of the OMP [93], which obtains a low probability of success but at a much lower computational cost.

4.3.2 Hybrid system example

We now consider the switched linear system with 3 modes defined by (3.27) in Sect. 3.4.2 to test the identification Algorithm 6. Training sets of $N = 600$ points are generated with a uniformly distributed random sequence of $q_k \in \{1, 2, 3\}$ and an additional Gaussian noise with $\sigma_v = 0.1$.

We compare the NPE (1.23), the CE (1.24) and the estimated number of modes of the proposed method (Sℓ_1M) with the ones of the methods with free number of modes: ES (page 21), PS (page 23) and GEO in Chapter 3. More precisely, over 100 trials with different input, switching and noise sequences, we report the mean and standard deviation of the NPE. Since the methods estimate the parameter vectors one by one until the data set is empty, the number of modes cannot be fixed. In this case, we retain the best parameter vectors as the ones closest to the true ones to compute the NPE. The threshold δ used to assign data points to a submodel is varied in the range $[\sigma_v, 10\sigma_v]$. Due to the unbounded Gaussian noise, all methods are stopped with $N_{min} = 0.05N$ data points left unassigned to a mode in order to avoid estimating too many irrelevant submodels.

Figure 4.2 shows that the proposed method (with N_s set as suggested in Sect. 4.2) yields a model with smaller NPE and CE than the other methods. A better estimate of the number of modes than the ES method is also obtained. Moreover, this is true over a large range of values for δ. The Ref line corresponds to the model estimated with knowledge

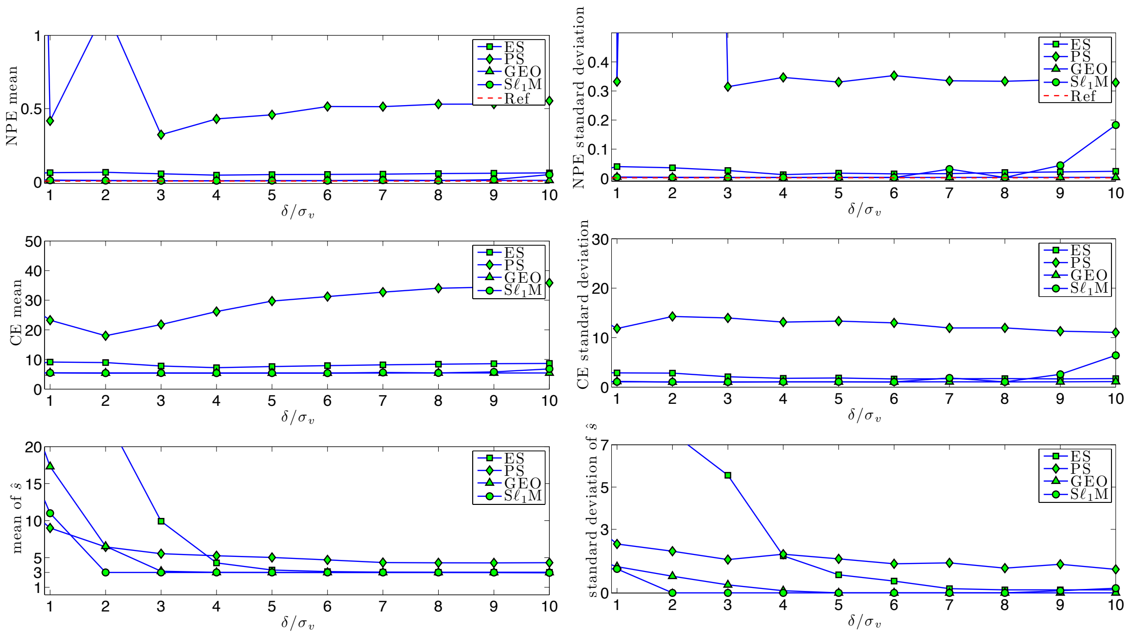


Figure 4.2: Parametric error (NPE), classification error rate (CE) and estimated number of modes for different values of the threshold δ used in the switched system identification example.

of the true mode.

4.4 Conclusions

The chapter improved a method for hybrid system identification, which relies on finding the sparse solutions to systems of linear equations through convex relaxations of ℓ_0-norm minimization problems. A new iterative algorithm was proposed to improve the sparsity of the solution of a generic convex relaxation based on the ℓ_1-norm. Compared with the state of the art (the iteratively reweighted scheme in Sect. 4.1.1), the proposed algorithm benefits from the absence of hyperparameters and a finite convergence in a number of steps at most equal to the number of linear equations. In addition, a sparse recovery condition, guaranteeing the convergence towards the sparsest solution, was proved and experiments showed that the new scheme can recover the sparsest solution in difficult cases with larger sparsity levels. Finally, an experimental comparison of the resulting identification algorithm with the most recent hybrid system identification approaches based on convex optimization highlighted the advantages of the proposed method.

Chapter 5

Nonlinear hybrid system identification

Abstract. *This chapter focuses on the identification of nonlinear hybrid dynamical systems in two frameworks: the continuous optimization framework and the error sparsification framework. To be able to approximate arbitrary nonlinearities, kernel submodels are considered.*
Within the first framework, a preprocessing step is added to fix the submodel sizes and limit the number of optimization variables in order to maintain efficiency for large data sets. Several approaches to build sparse kernel submodels are reviewed and adapted. These are compared in numerical experiments, which show that the proposed approach achieves the simultaneous classification of data points and approximation of the nonlinear behaviors in an efficient and accurate manner.
Within the second framework, the submodels are iteratively estimated one by one by maximizing the sparsity of the corresponding error vector. We relax the sparsity condition by introducing robust sparsity, which can be optimized through the minimization of a modified ℓ_1-norm or, equivalently, of the ε-insensitive loss function. Then, we show that, depending on the choice of regularizer, the method is equivalent to different forms of support vector regression. More precisely, the submodels can be estimated by iteratively solving a classical support vector regression problem, in which the sparsity of support vectors relates to the sparsity of the error vector in the considered hybrid system identification framework. This allows us to transfer theoretical results as well as efficient optimization algorithms from the field of machine learning to the hybrid system framework.

As presented in Chapter 1, a nonlinear hybrid system is modeled by a collection of nonlinear functions,

$$y_k = f_{q_k}(\boldsymbol{x}_k) + v_k. \tag{5.1}$$

where $y_k \in \mathbb{R}$ is the output at discrete time k, $\boldsymbol{x}_k = [y_{k-1}, \dots, y_{k-n_a},\ u_{k-1}, \dots, u_{k-n_b}]^\top \in \mathbb{R}^{n_a+n_b}$ is the regression vector with the past outputs y_{k-i} and inputs u_{k-i}, with n_a, n_b the model orders, $q_k \in \{1, \dots, s\}$ is the discrete state, s is the number of submodels, $v_k \in \mathbb{R}$ is a noise term and $\{f_j\}_{j=1}^s$ are s nonlinear submodels.

Then, Problem 3 is considered, for switched nonlinear hybrid systems.

Problem 3. *Given a data set $\mathcal{D} = \{(\boldsymbol{x}_k, y_k)\}_{k=1}^N$ generated by a nonlinear hybrid system, estimate the number of submodels s, the submodels $\{f_j\}_{j=1}^s$, and the switching sequence $\{q_k\}_{k=1}^N$.*

There are only a few methods [58, 59, 7, 38] to deal with this problem. They are developed within the frameworks of continuous optimization presented in Sect. 2.2.1 (page 18) for [58, 59], error sparsification presented in Sect. 2.2.2 (page 21) for [7] or sum-of-norms optimization presented in Sect. 2.2.2 (page 24) for [38]. This chapter presents new contributions to the two first frameworks in which the arbitrary nonlinearities of hybrid systems are learned in a given Reproducing Kernel Hilbert Space (RKHS) with a kernel function κ (see Sect. 1.3.2).

A kernel function κ implicitly computes inner products, $\kappa(\boldsymbol{x}_k, \boldsymbol{x}) = \langle \boldsymbol{\phi}(\boldsymbol{x}_k), \boldsymbol{\phi}(\boldsymbol{x})\rangle_{\mathcal{F}}$, between points in a *feature space* $\mathcal{F}$ obtained by a hidden nonlinear mapping

$$\boldsymbol{\phi} : \boldsymbol{x} \mapsto \boldsymbol{\phi}(\boldsymbol{x}), \tag{5.2}$$

of the points $\boldsymbol{x}$ in the original input space. In a feature space $\mathcal{F}$, the nonlinear function f can become linear (see, e.g., [85]), i.e.,

$$f(\boldsymbol{x}) = \langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x})\rangle_{\mathcal{F}}. \tag{5.3}$$

Let $\boldsymbol{K}$ be the so-called Gram matrix of the kernel κ with respect to a data set $\mathcal{D}_{\boldsymbol{x}} = \{\boldsymbol{x}_k\}_{k=1}^N$, i.e., with all components given by $K_{ik} = \kappa(\boldsymbol{x}_i, \boldsymbol{x}_k)$ and $\boldsymbol{K}_S$ be a submatrix of $\boldsymbol{K}$ containing only the rows and columns corresponding to data points in $S \subset \mathcal{D}_{\boldsymbol{x}}$.

5.1 Continuous optimization framework

As presented in Sect. 2.2.1 (page 18), the continuous optimization framework proposes to solve the problem of linear hybrid system identification with the fixed number of modes s by a continuous optimization program. This framework, i.e., the minimum-of-errors (ME) estimator (2.13) or product-of-errors (PE) estimator (2.14), can be extended to estimate nonlinear submodels.

5.1.1 Extension to nonlinear submodels

To approximate submodels $\{f_j\}_{j=1}^s$ of arbitrary nonlinear form, we look for a general model class used in nonlinear system identification.

In Sect. 1.3.2, the generalized representer theorem (Theorem 1 on page 7) showed that the finite linear combination of kernel functions computed at the training points $\{\boldsymbol{x}_i : i = 1, \dots, N\}$,

$$f(\boldsymbol{x}) = \sum_{i=1}^N \alpha_i \kappa(\boldsymbol{x}_i, \boldsymbol{x}) = \boldsymbol{\alpha}^\top \boldsymbol{\kappa}(\boldsymbol{x}), \tag{5.4}$$

where $\boldsymbol{\alpha} = [\alpha_1, \dots, \alpha_N]^\top$ and $\boldsymbol{\kappa}(\boldsymbol{x}) = [\kappa(\boldsymbol{x}_1, \boldsymbol{x}), \dots, \kappa(\boldsymbol{x}_N, \boldsymbol{x})]^\top$, is the solution in the RKHS $\mathcal{H}$ to the optimization problem (1.14)

$$\min_{f \in \mathcal{H}} \sum_{k=1}^N \ell(y_k, f(\boldsymbol{x}_k)) + \lambda \|f\|_{\mathcal{H}}^2, \tag{5.5}$$

where the norm in the RKHS, $\|f\|_{\mathcal{H}}$, is defined by (1.17) and ℓ is a loss function defined by (1.7). Therefore, the kernel form (5.4) can be used to approximate the system nonlinearity.

Introducing submodels in kernel form as (5.4),

$$f_j(\boldsymbol{x}) = \sum_{i=1}^N \alpha_{ji} \kappa_j(\boldsymbol{x}_i, \boldsymbol{x}) = \boldsymbol{\alpha}_j^\top \boldsymbol{\kappa}_j(\boldsymbol{x}), \tag{5.6}$$

where $\boldsymbol{\alpha}_j = [\alpha_{j1}, \dots, \alpha_{jN}]^\top$ and $\boldsymbol{\kappa}_j(\boldsymbol{x}) = [\kappa_j(\boldsymbol{x}_1, \boldsymbol{x}), \dots, \kappa_j(\boldsymbol{x}_N, \boldsymbol{x})]^\top$, leads to the PE estimator for nonlinear hybrid systems, expressed as the solution to

$$\min_{\{\boldsymbol{\alpha}_j\}} \frac{1}{s} \sum_{j=1}^{s} \mathcal{R}(\boldsymbol{\alpha}_j) \ + \frac{C}{N} \sum_{k=1}^{N} \prod_{j=1}^{s} \ell \left(y_k, \sum_{i=1}^{N} \alpha_{ji} \kappa_j(\boldsymbol{x}_i, \boldsymbol{x}_k) \right), \tag{5.7}$$

where ℓ is a smooth loss function and $\mathcal{R}(\boldsymbol{\alpha}_j)$ is a regularizer acting on the parameters $\boldsymbol{\alpha}_j$ of the submodel f_j. For instance, the model complexity can be measured via the ℓ_1-norm of the parameter vectors, i.e., $\mathcal{R}(\boldsymbol{\alpha}_j) = \|\boldsymbol{\alpha}_j\|_1$. This regularizer penalizes non-smooth functions and ensures sparsity as a certain number of parameters α_{ji} will tend towards zero. Regularization over the ℓ_2-norm of the parameter vectors, i.e., $\mathcal{R}(\boldsymbol{\alpha}_j) = \|\boldsymbol{\alpha}_j\|_2^2$, is also possible, but may result in less sparse models.

Different kernel functions κ_j can be used in (5.6) for the different submodels f_j. It is thus possible to take prior knowledge into account such as the number of modes governed by linear dynamics or information on the type of a particular nonlinearity, if available.

Solving the optimization problem (5.7) with a global optimization solver may become prohibitively time consuming because of the large number of variables. Indeed, there are sN variables α_{ji}, with s the number of submodels and N the number of data points. In the next section, we build reduced-size kernel submodels to overcome this issue. The results have been published in [J1] (see the Author's bibliography, page XVIII).

5.1.2 Reduced-size kernel model based approach

As in Support Vector Machines (SVMs) [95, 85], we refer to the vectors $\boldsymbol{x}_i$ for which the associated $\{\alpha_{ji}\}_{j=1}^s$ parameters are nonzero as the Support Vectors (SVs), since these are the only data points kept in the final model. But here, we need to reduce *beforehand* the number of SVs in (5.6), and thus the number of variables in the optimization program (5.7), by *reduced-size kernel models.* Let

$$S_j = \{\boldsymbol{x}_i^j\}_{i=1}^{M_j} \tag{5.8}$$

denote the set of M_j SVs selected from the data set $\mathcal{D}_{\boldsymbol{x}} = \{\boldsymbol{x}_i\}_{i=1}^N$ for the jth submodel

$$\tilde{f}_j(\boldsymbol{x}) = \sum_{i=1}^{M_j} \beta_{ji} \kappa_j(\boldsymbol{x}_i^j, \boldsymbol{x}). \tag{5.9}$$

The M_j parameters of submodel $\tilde{f}_j$ are now given by $\boldsymbol{\beta}_j = [\beta_{j1}, \dots, \beta_{jM_j}]^\top$.

The complete identification procedure of the continuous optimization framework applied to nonlinear hybrid system identification with the product-error (PE) estimator is summarized in Procedure 7.

Note that the reduced-size submodels (5.9) are based on an intrinsically sparse representation of the data, hence the choice of the smooth ℓ_2-norm regularization over the low-dimensional parameter vectors $\boldsymbol{\beta}_j$ in (5.10).

The final optimization program (5.10) involves only $\sum_{j=1}^s M_j$ variables instead of sN as in (5.7). This allows the complexity of the algorithm (5.10) to scale only linearly with respect to the number of training data N (through the summation term), as experimentally verified in [60].

In the following, we present four methods to build reduced-size kernel submodels (5.9) for step 1 in Procedure 7. Note that all these methods are only based on the input data

Procedure 7 Learning procedure based on reduced-size kernel form

1. Find the reduced-size kernel form $\tilde{f}_j(\boldsymbol{x})$ (5.9) for each submodel.
2. Train the hybrid model by solving

$$\min_{\{\boldsymbol{\beta}_j\}} \frac{1}{s}\sum_{j=1}^{s}\frac{\boldsymbol{\beta}_j^\top\boldsymbol{\beta}_j}{M_j} + \frac{C}{N}\sum_{k=1}^{N}\prod_{j=1}^{s}\ell\left(y_k, \tilde{f}_j(\boldsymbol{x}_k)\right). \tag{5.10}$$

3. Estimate the mode $\hat{q}_k$ for each data point by

$$\hat{q}_k = \arg\min_{j=1,\dots,s} \ell(y_k, \tilde{f}_j(\boldsymbol{x}_k)), \;\; k = 1, \dots, N. \tag{5.11}$$

and do not use the target output, which is undetermined in the context of hybrid system identification where the switches are unknown. The number of modes s is a priori fixed and the following proposed algorithms are repeated s times to build s reduced-sized submodels.

Entropy maximization

The first method to build a reduced-size kernel model for nonlinear hybrid system identification is inspired by the fixed-size Least Squares SVM (LS-SVM) [53] which is based on the maximization of an entropy criterion to ensure a sufficient coverage of the feature space by the SVs. Then the selected SVs are used to build an approximation of the nonlinear mapping $\boldsymbol{\phi}$ (5.2) hidden in the kernel function κ, which is in turn used to recast the problem into a linear form in the approximated feature space. In experiments, the fixed-size LS-SVM method was rather sensitive to the number of selected SVs. Hence, in [59], a similar but more straightforward method for Gaussian RBF kernels is applied, where no approximation of the nonlinear mapping is built, but instead the RBF centers are directly used as SVs.

As in fixed-size LS-SVM [53], the selection algorithm maximizes the quadratic Rényi entropy H, which quantifies the diversity, uncertainty or randomness of a system. For a particular mode j, H is approximated by

$$H_j \approx -\log\frac{1}{M_j^2}\boldsymbol{1}^\top\boldsymbol{K}_{S_j}\boldsymbol{1}, \tag{5.12}$$

where S_j as (5.8) is a selected subset of the data set $\mathcal{D}_{\boldsymbol{x}}$, M_j is the size of S_j and

$$\boldsymbol{K}_{S_j} = \begin{bmatrix} \kappa_j(\boldsymbol{x}_1^j, \boldsymbol{x}_1^j) & \dots & \kappa_j(\boldsymbol{x}_1^j, \boldsymbol{x}_{M_j}^j) \\ \vdots & \ddots & \\ \kappa_j(\boldsymbol{x}_{M_j}^j, \boldsymbol{x}_1^j) & \dots & \kappa_j(\boldsymbol{x}_{M_j}^j, \boldsymbol{x}_{M_j}^j) \end{bmatrix}, \tag{5.13}$$

is the partial kernel matrix.

Algorithm 8 presents the method to select the SVs for the kernel submodel j.

In this approach, the numbers of SVs $\{M_j\}_{j=1}^s$ are hyperparameters that must be fixed a priori. Following [59], for Gaussian RBF kernels with bandwidth parameter σ_j, the numbers M_j can be set according to the heuristic

$$M_j = \left\lfloor \frac{1}{\sigma_j}\max_{i=1,\dots,p}\left(\max_{k=1,\dots,N} x_{ki} - \min_{k=1,\dots,N} x_{ki}\right)\right\rfloor, \tag{5.14}$$

Algorithm 8 Entropy maximization selecting algorithm for the jth submodel

```
Require: a data set $\mathcal{D}_{\boldsymbol{x}} = \{\boldsymbol{x}_k\}_{k=1}^N$, a number of SVs $M_j$ and a maximum number of iterations $n_{max}$.
  1. Randomly select a subset $S_j$ with $M_j$ SVs from the data set $\mathcal{D}_{\boldsymbol{x}}$, and initialize $\overline{S} = \mathcal{D}_{\boldsymbol{x}} \setminus S_j$.
  2. Randomly select an SV in $S_j$, $\boldsymbol{x}^\star$, and one of the remaining data vectors, $\boldsymbol{x}^\dagger \in \overline{S}$.
  3. If the criterion (5.12) increases by replacing $\boldsymbol{x}^\star$ by $\boldsymbol{x}^\dagger$, retain $\boldsymbol{x}^\dagger$ as an SV in $S_j$ and replace $\boldsymbol{x}^\dagger$ by $\boldsymbol{x}^\star$ in $\overline{S}$.
  4. Repeat from 2 until the maximum number of iterations $n_{max}$ is reached.
  return $S_j$.
```

where $\lfloor \cdot \rfloor$ denotes the integer part of its argument and x_{ki} is the ith component of $\boldsymbol{x}_k$.

This heuristic is not optimal in the sense of minimizing the generalization error, but it ensures sufficient support for the model over the whole input space. The numbers M_j in (5.14) strongly depend on the bandwidths σ_j, since more SVs are needed to cover the whole input space with a smaller bandwidth. In practice, the values of σ_j can influence the quality of the model as they control the smoothness of the submodels. Proper tuning of these values may require multiple trials or prior knowledge on the relative smoothness of the subsystems in the model. However, suboptimal numbers M_j are sufficient to obtain rough mode estimates $\hat{q}_k$. Then the resulting data classification can be used to re-estimate the submodels separately on a subset of the data for each mode. If these refined submodels are learned by SVM techniques for instance, then the final number of SVs is automatically determined.

Feature vector selection (FVS)

The second method is based on another approach for LS-SVM which has been considered in [11, 27, 26], where a minimal set of training vectors is selected such that it induces a basis for the subspace containing the data mapped in feature space. Therefore, the kernel model (5.4) can be expressed in a reduced-size form via only these selected training vectors.

Indeed, if the kernel model (5.4) can be rewritten in terms of inner products in a feature space $\mathcal{F}$ of a mapping $\boldsymbol{\phi}$ (5.2),

$$f(\boldsymbol{x}) = \sum_{i=1}^N \alpha_i \kappa(\boldsymbol{x}_i, \boldsymbol{x}) = \sum_{i=1}^N \alpha_i \langle \boldsymbol{\phi}(\boldsymbol{x}_i), \boldsymbol{\phi}(\boldsymbol{x}) \rangle_{\mathcal{F}} \doteq \langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}) \rangle_{\mathcal{F}}, \tag{5.15}$$

it leads to a linear form (5.3) with respect to

$$\boldsymbol{w} = \sum_{i=1}^N \alpha_i \boldsymbol{\phi}(\boldsymbol{x}_i). \tag{5.16}$$

Let us define a subspace $\mathcal{F}_D \subset \mathcal{F}$ containing all data feature vectors. Since there are N data feature vectors $\{\boldsymbol{\phi}(\boldsymbol{x}_k)\}_{k=1}^N$, the dimension of $\mathcal{F}_D$ is not larger than N. If there exists a small set of vectors which can span $\mathcal{F}_D$, i.e., all the data feature vectors can be expressed as a linear combination of these vectors, the parameter vector $\boldsymbol{w}$ defined by (5.16) is a linear combination of these vectors. In this case, the kernel model (5.4) can be expressed with a reduced size.

Let the set $S = \{\boldsymbol{x}_i^*\}_{i=1}^M$ be the set of M selected vectors from the data set $\mathcal{D}_{\boldsymbol{x}}$ corresponding to the basis vectors of $\mathcal{F}_D$. The mapping of any vector $\boldsymbol{x}_k$ can be represented as

$$\boldsymbol{\phi}(\boldsymbol{x}_k) = \boldsymbol{\Phi}_S \boldsymbol{\alpha}_k, \tag{5.17}$$

where $\boldsymbol{\Phi}_S = [\boldsymbol{\phi}(\boldsymbol{x}_1^*), \dots, \boldsymbol{\phi}(\boldsymbol{x}_M^*)]$ is the matrix of the selected vectors in $\mathcal{F}$ and $\boldsymbol{\alpha}_k$ is a coefficient vector.

For a given set S, the vector $\boldsymbol{\alpha}_k$ is found such that the estimate $\boldsymbol{\Phi}_S\boldsymbol{\alpha}_k$ is as close as possible to the mapping $\boldsymbol{\phi}(\boldsymbol{x}_k)$. That refers to the following optimization problem

$$\min_{\boldsymbol{\alpha}_k} \frac{\|\boldsymbol{\phi}(\boldsymbol{x}_k) - \boldsymbol{\Phi}_S\boldsymbol{\alpha}_k\|_2^2}{\|\boldsymbol{\phi}(\boldsymbol{x}_k)\|_2^2}. \tag{5.18}$$

With its least squares solution $\boldsymbol{\alpha}_k = (\boldsymbol{\Phi}_S^\top \boldsymbol{\Phi}_S)^{-1}\boldsymbol{\Phi}_S^\top \boldsymbol{\phi}(\boldsymbol{x}_k)$, the minimum of (5.18) is

$$1 - \frac{1}{\|\boldsymbol{\phi}(\boldsymbol{x}_k)\|_2^2}\left(\boldsymbol{\phi}(\boldsymbol{x}_k)^\top \boldsymbol{\Phi}_S \left(\boldsymbol{\Phi}_S^\top \boldsymbol{\Phi}_S\right)^{-1} \boldsymbol{\Phi}_S^\top \boldsymbol{\phi}(\boldsymbol{x}_k)\right) \tag{5.19}$$

By using the kernel function to avoid the explicit expression of $\boldsymbol{\phi}$, we rewrite this result as

$$1 - \frac{\boldsymbol{\kappa}_{Sk}^\top \boldsymbol{K}_S^{-1} \boldsymbol{\kappa}_{Sk}}{\kappa_{kk}}, \tag{5.20}$$

where $\boldsymbol{K}_S$ is the kernel matrix with respect to the data set $S = \{\boldsymbol{x}_i^*\}_{i=1}^M$, i.e., with $K_{Sit} = \langle \boldsymbol{\phi}(\boldsymbol{x}_i^*), \boldsymbol{\phi}(\boldsymbol{x}_t^*)\rangle$, and $\boldsymbol{\kappa}_{Sk} = [\kappa_{1k}, \dots, \kappa_{ik}, \dots, \kappa_{Mk}]^\top$, is a vector with $\kappa_{ik} = \langle \boldsymbol{\phi}(\boldsymbol{x}_i^*), \boldsymbol{\phi}(\boldsymbol{x}_k)\rangle$, for $i = 1, \dots, M$.

Therefore, the set S is found such that it minimizes the following average,

$$J(S) = 1 - \frac{1}{N}\sum_{k=1}^{N} \frac{\boldsymbol{\kappa}_{Sk}^\top \boldsymbol{K}_S^{-1} \boldsymbol{\kappa}_{Sk}}{\kappa_{kk}}. \tag{5.21}$$

Now, this feature vector selection is applied to the problem of nonlinear hybrid system identification with an additional subscrit/superscript j to refer to the corresponding mode of the hybrid system.

The sets S_j are found to maximize the following criterion

$$J(S_j) = \frac{1}{N}\sum_{k=1}^{N} \frac{\boldsymbol{\kappa}_{S_jk}^\top \boldsymbol{K}_{S_j}^{-1} \boldsymbol{\kappa}_{S_jk}}{\kappa_{jkk}}. \tag{5.22}$$

Then, $\boldsymbol{w}_j$ of the jth submodel can be expressed as a linear combination of feature vectors mapped from the selected vectors of $S_j = \{\boldsymbol{x}_i^j\}_{i=1}^{M_j}$,

$$\boldsymbol{w}_j = \sum_{i=1}^{N} \alpha_{ji} \boldsymbol{\phi}_j(\boldsymbol{x}_i) = \sum_{i=1}^{M_j} \beta_{ji} \boldsymbol{\phi}_j(\boldsymbol{x}_i^j). \tag{5.23}$$

This leads to the jth reduced-size submodel as in the form (5.9),

$$\tilde{f}_j(\boldsymbol{x}) = \sum_{i=1}^{M_j} \beta_{ji} \langle \boldsymbol{\phi}_j(\boldsymbol{x}_i^j), \boldsymbol{\phi}_j(\boldsymbol{x})\rangle = \sum_{i=1}^{M_j} \beta_{ji} \kappa_j(\boldsymbol{x}_i^j, \boldsymbol{x}). \tag{5.24}$$

Note that in comparison to the previous method, M_j is not fixed *a priori*, but simply corresponds to the dimension of the smallest subspace containing the data in feature space.

Though the method proposed in [11] to maximize (5.22) can be improved for efficiency as in [26], it remains rather time consuming for large data sets. In order to maintain as low as possible the computational cost of the overall estimation procedure, in which the basis selection is only the first step, we instead propose the randomized algorithm presented in Algorithm 9.

Algorithm 9 Feature vector selection algorithm for jth submodel

```
Require: a data set $\mathcal{D}_{\boldsymbol{x}} = \{\boldsymbol{x}_k\}_{k=1}^N$ and a small threshold $\xi$.
  1. Initialize $S_j = \emptyset$, $\overline{S} = \mathcal{D}_{\boldsymbol{x}}$, $i = 1$ and define $J(\emptyset) = 0$.
  2. Append to $S_j$ a randomly selected vector, $\boldsymbol{x}_i^j$, from the set $\overline{S}$ and compute $J(S_j)$ (5.22).
  3. If $J(S_j)$ increases, retain $\boldsymbol{x}_i^j$ in $S_j$, update $\overline{S} = \overline{S} \setminus \boldsymbol{x}_i^j$ and set $i = i + 1$, otherwise remove $\boldsymbol{x}_i^j$ from $S_j$.
  4. Repeat from step 2 until $\boldsymbol{K}_{S_j}$ is no longer invertible $\big(\det(\boldsymbol{K}_{S_j}) < \xi\big)$.
  return $S_j$.
```

Kernel Principal Component Regression (KPCR)

Following the Kernel Principal Component Regression (KPCR) method in [78], the number of parameters in (5.4) can be reduced by using only several principal components of the kernel matrix which are sufficient to account for most of the structure in the data.

Indeed, when we take the linear form (5.3) in an appropriate feature space $\mathcal{F}$,

$$f(\boldsymbol{x}_k) = \langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}_k)\rangle_{\mathcal{F}}, \tag{5.25}$$

we see that due to the high dimension of $\boldsymbol{\phi}(\boldsymbol{x}_k)$, this model has a parameter vector $\boldsymbol{w}$ in large dimension to be determined. When the dimension reduction technique PCA (see in Appendix A) is applied to the feature data $\{\boldsymbol{\phi}(\boldsymbol{x}_k)\}_{k=1}^N$ to find a new representation with small dimension, it also leads to a small size model.

To write equations in matrix forms, the inner product in a feature space $\mathcal{F}$, $\langle \boldsymbol{\phi}(\boldsymbol{x}_1), \boldsymbol{\phi}(\boldsymbol{x}_2)\rangle_{\mathcal{F}}$, is rewritten as $\boldsymbol{\phi}(\boldsymbol{x}_1)^\top \boldsymbol{\phi}(\boldsymbol{x}_2)$. Let $\lambda_1, \dots, \lambda_L$ and $\boldsymbol{v}_1, \dots, \boldsymbol{v}_L$ be the eigenvalues arranged in decreasing order and the corresponding normalized orthogonal eigenvectors of the covariance matrix $\frac{1}{N}\boldsymbol{\Phi}\boldsymbol{\Phi}^\top$ with $\boldsymbol{\Phi} = [\boldsymbol{\phi}(\boldsymbol{x}_1), \dots, \boldsymbol{\phi}(\boldsymbol{x}_N)] \in \mathbb{R}^{L\times N}$ the feature data matrix.

The projection of a feature vector $\boldsymbol{\phi}(\boldsymbol{x})$ onto the first M eigenvectors ($M < L$) gives a new representation of $\boldsymbol{\phi}(\boldsymbol{x})$ with low dimension,

$$\boldsymbol{\phi}^{new}(\boldsymbol{x}) = \boldsymbol{V}^\top \boldsymbol{\phi}(\boldsymbol{x}). \tag{5.26}$$

where $\boldsymbol{V} = [\boldsymbol{v}_1, \dots, \boldsymbol{v}_M]$. Then, a linear model in low dimension space can be written with a new representation as

$$\tilde{f}(\boldsymbol{x}) = \boldsymbol{\beta}^\top \boldsymbol{\phi}^{new}(\boldsymbol{x}) = \boldsymbol{\beta}^\top \boldsymbol{V}^\top \boldsymbol{\phi}(\boldsymbol{x}), \tag{5.27}$$

where $\boldsymbol{\beta} \in \mathbb{R}^M$ is a coefficient vector.

Let μ_k and $\boldsymbol{\gamma}_k$ ($k = 1, \dots, N$) be eigenvalues and eigenvectors of the matrix $\boldsymbol{K}$. There is an eigenvalue equivalence between the kernel matrix $\boldsymbol{K}$ and the covariance matrix $\frac{1}{N}\boldsymbol{\Phi}\boldsymbol{\Phi}^\top$ [80] as follows

$$\lambda_k = \frac{\mu_k}{N}, \tag{5.28}$$

$$\boldsymbol{v}_k = \boldsymbol{\Phi}\frac{\boldsymbol{\gamma}_k}{\sqrt{\mu_k}}, \quad k = 1, \dots, N, \tag{5.29}$$

By replacing the eigenvectors $\boldsymbol{v}_k$ (5.29) in (5.27) and using the kernel function to avoid the explicit expression of $\boldsymbol{\phi}(.)$, we obtain a reduced-size kernel model

$$\tilde{f}(\boldsymbol{x}) = \boldsymbol{\beta}^\top \boldsymbol{A}\ \boldsymbol{\kappa}(\boldsymbol{x}), \tag{5.30}$$

where $\boldsymbol{A} = \left[\frac{\gamma_1}{\sqrt{\mu_1}}, \dots, \frac{\gamma_M}{\sqrt{\mu_M}}\right]^\top$ and $\boldsymbol{\kappa}(\boldsymbol{x}) = [\kappa(\boldsymbol{x}_1, \boldsymbol{x}), ..., \kappa(\boldsymbol{x}_N, \boldsymbol{x})]^\top$.

Only eigenvalues and eigenvectors of the kernel matrix are used to build a reduced-size kernel model. To apply PCA, the mapped data are centered in feature space, i.e., $\sum_{k=1}^N \boldsymbol{\phi}(\boldsymbol{x}_k) = \mathbf{0}$. Therefore, the kernel matrix $\boldsymbol{K} = \boldsymbol{\Phi}^\top \boldsymbol{\Phi}$ is substituted in computations by

$$\hat{\boldsymbol{K}} = \left(\boldsymbol{I}_N - \frac{1}{N}\mathbf{1}_{N\times N}\right) \boldsymbol{K} \left(\boldsymbol{I}_N - \frac{1}{N}\mathbf{1}_{N\times N}\right), \tag{5.31}$$

as proposed in [80].

Now, this technique is applied to the problem of nonlinear hybrid system identification. Formally, for a particular mode j, we are interested in finding the kernel principal components that can represent all data points associated to this mode. However, as the discrete state q_k (determining to which mode belongs a data point) is unknown for the training data, we have to compute the kernel principal components from the whole data set $\mathcal{D}_{\boldsymbol{x}}$ for each mode. Note nevertheless that these principal components can be different from one mode to another if the kernel functions κ_j are different. The algorithm to build the reduced-size kernel form for a particular mode j is presented in Algorithm 10.

Algorithm 10 KPCA algorithm to build a jth reduced-size kernel submodel

```
Require: a data set $\mathcal{D}_{\boldsymbol{x}} = \{\boldsymbol{x}_k\}_{k=1}^N$ and a number of selected largest eigenvalues $M_j$.
  1. Compute the kernel matrix $\boldsymbol{K}_j$ from $\mathcal{D}_{\boldsymbol{x}}$.
  2. Compute the $M_j$ largest eigenvalues, $\{\mu_{jk}\}_{k=1}^{M_j}$, and corresponding eigenvectors, $\{\gamma_k^j\}_{k=1}^{M_j}$, of $\boldsymbol{K}_j$.
  3. Calculate $\boldsymbol{A}_j = \left[\frac{\gamma_1^j}{\sqrt{\mu_{j1}}}, \dots, \frac{\gamma_{M_j}^j}{\sqrt{\mu_{jM_j}}}\right]$.
  4. Build a reduced-size kernel model for $j$th mode
       $\tilde{f}_j(\boldsymbol{x}) = \boldsymbol{\beta}_j^\top \boldsymbol{A}_j \boldsymbol{\kappa}_j(\boldsymbol{x})$,   (5.32)
     where $\boldsymbol{\beta}_j$ is a parameter vector to be determined and $\boldsymbol{\kappa}_j(\boldsymbol{x}) = [\kappa_j(\boldsymbol{x}_1, \boldsymbol{x}), ..., \kappa_j(\boldsymbol{x}_N, \boldsymbol{x})]^\top$.
  return the form (5.32) of $\tilde{f}_j(\boldsymbol{x})$.
```

The number of nonlinear principal components M_j must be sufficient to describe the structure of the data. For a given $\rho \in [0, 1]$, the Inertia Percentage Criterion IPC can be used to estimate M_j as the smallest number m such that

$$IPC(m) = \frac{\sum_{i=1}^m \mu_{ji}}{\sum_{i=1}^N \mu_{ji}} \geq \rho, \tag{5.33}$$

where μ_{ji}, $i = 1, ..., N$, are the eigenvalues arranged in decreasing order and $\sum_{i=1}^N \mu_{ji} = \text{Trace}(\boldsymbol{K}_j)$. Note that $\text{Trace}(\boldsymbol{K}_j) = N$ in case of a Gaussian RBF kernel matrix.

Reduced Kernel Principal Component Regression (RKPCR)

In the method above, one obtains a reduced-size kernel submodel form (5.30) with only M parameters that need to be estimated. However, the resulting model needs to retain the N original data points to compute its output for a new input $\boldsymbol{x}$ which involves the vector $\boldsymbol{\kappa}(\boldsymbol{x})$. Moreover, the eigenvalue decomposition of a too large kernel matrix $\boldsymbol{K}$ can be prohibitive. To avoid these issues, the kernel matrix can be approximated by a low rank matrix $\tilde{\boldsymbol{K}}$. Since we need only some first largest eigenvalues and the related eigenvectors of the kernel matrix $\boldsymbol{K}$, a low rank matrix $\tilde{\boldsymbol{K}}$ is sufficient for that purpose. Most of the computations for the low rank approximation $\tilde{\boldsymbol{K}}$ of a kernel matrix $\boldsymbol{K}$ involve only a subset of the training data. In the original Nyström method, the subset selection is random with the subset size fixed beforehand. Such a selection influences the accuracy of the solution and leads to a more complex implementation. Thus, the Nyström method based on an incomplete Cholesky decomposition is proposed in [92].

The incomplete Cholesky decomposition [40] of matrix $\boldsymbol{K}$ provides a matrix $\boldsymbol{C} = [\boldsymbol{L}\ \boldsymbol{L}_r] \in \mathbb{R}^{R\times N}$ and a data subset S' of size R $(R < N)$ such that

$$\tilde{\boldsymbol{K}} = \boldsymbol{C}^\top \boldsymbol{C} \tag{5.34}$$

$$\boldsymbol{K}_{S'} = \boldsymbol{L}^\top \boldsymbol{L} \tag{5.35}$$

where $\tilde{\boldsymbol{K}}$ is the low rank matrix of $\boldsymbol{K}$, $\boldsymbol{K}_{S'}$ is a kernel matrix of the data set S' and $\boldsymbol{L} \in \mathbb{R}^{R\times R}$ is a triangular matrix.

Then, the R eigenvalues of the covariance matrix $\boldsymbol{Q} = \boldsymbol{C}\boldsymbol{C}^\top \in \mathbb{R}^{R\times R}$ are identical to the largest ones of $\tilde{\boldsymbol{K}}$. The eigendecomposition of $\boldsymbol{Q}$ is expressed as

$$\boldsymbol{Q} = \boldsymbol{E}\boldsymbol{D}\boldsymbol{E}^\top, \tag{5.36}$$

where $\boldsymbol{D}$ is the diagonal matrix of the eigenvalues in decreasing order and $\boldsymbol{E}$ is the related eigenvector matrix. As before, only the first M, $(M \leq R)$, eigenvector columns of $\boldsymbol{E}$ are selected according to the criterion (5.33) to form a reduced model as in (5.30):

$$\tilde{f}(\boldsymbol{x}) = \boldsymbol{\beta}^\top \boldsymbol{A}_R \tilde{\boldsymbol{\kappa}}(\boldsymbol{x}), \tag{5.37}$$

where $\boldsymbol{\beta}$ is the M-dimensional parameter vector, $\boldsymbol{A}_R$ is calculated by $\boldsymbol{A}_R = \boldsymbol{E}_M^\top \left(\boldsymbol{L}^\top\right)^{-1} \in \mathbb{R}^{M\times R}$ with $\boldsymbol{E}_M$ the submatrix of the first M eigenvector columns of $\boldsymbol{E}$ and the reduced vector $\tilde{\boldsymbol{\kappa}}(\boldsymbol{x}) = [\kappa(\boldsymbol{x}_1, \boldsymbol{x}), \dots, \kappa(\boldsymbol{x}_i, \boldsymbol{x}), \dots, \kappa(\boldsymbol{x}_R, \boldsymbol{x})]^\top$ is calculated for an $\boldsymbol{x}$ with $\boldsymbol{x}_i$, $i = 1, \dots, R$, in the selected subset S'.

The procedure to build the reduced-size kernel form for a particular mode j is presented in Algorithm 11.

Algorithm 11 RKPCA algorithm to build a reduced-size kernel model for jth mode

```
Require: a data set $\mathcal{D}_{\boldsymbol{x}} = \{\boldsymbol{x}_k\}_{k=1}^N$ and a number of selected largest eigenvalues $M_j$.
  1. Compute the kernel matrix $\boldsymbol{K}_j$ from $\mathcal{D}_{\boldsymbol{x}}$.
  2. Obtain the matrix $\boldsymbol{C}_j$ and the subset $S'_j$ by an incomplete Cholesky decomposition of $\boldsymbol{K}_j$.
  3. Compute the $M_j$ largest eigenvalues and corresponding eigenvectors of $\boldsymbol{Q}_j$ and calculate $\boldsymbol{A}_{R_j}$.
  return the form $\tilde{f}_j(\boldsymbol{x})$ (5.37) as a reduced-size kernel submodel.
```

5.1.3 Numerical experiments

This section presents numerical results on two examples. The first one involves the estimation of a function switching between two unknown nonlinear functions, while the second one considers the identification of a switched nonlinear dynamical system.

As proposed in [60], all optimization programs are solved with the Multilevel Coordinate Search (MCS) algorithm [49]. Though the MCS algorithm can deal with unbounded variables, box constraints are used to limit the search space and restrain the variables to the interval $[-100, 100]$ (which is not very restrictive). All experiments are performed using only Matlab code on a standard desktop computer.

This section compares the four proposed methods for building reduced-size kernel submodels: Entropy maximization, FVS, KPCR and RKPCR. In the following Tables of results, the size of the SV sets for mode 1 (M_1) and mode 2 (M_2) is given by (5.14) for the Entropy maximization while being automatically determined for the other methods. For the KPCR and RKPCR methods, the IPC is set with $\rho = 0.9$. The quality of the models is evaluated on an independent and noise-free test set of $N_t = 2000$ data points by three performance indexes : the normalized criterion FIT$= 100\left(1 - \frac{\|\hat{\boldsymbol{y}}-\boldsymbol{y}\|_2}{\|\boldsymbol{y}-\bar{y}\mathbf{1}\|_2}\right)$ (1.22) where $\boldsymbol{y}$ is the output vector, $\bar{y}$ is its mean and $\hat{\boldsymbol{y}}$ is the predicted output vector, with $\hat{\boldsymbol{y}}$ computed by using the estimated discrete state $\hat{q}_k$ (5.11) (FITa), with $\hat{\boldsymbol{y}}$ computed by using the true discrete state q_k (FITb); and the classification error rate on the test set (Test Classif. err.). The classification error rate on the training set (Train. Classif. err.) is also given in order to analyze the ability of the methods to separate between the modes. The computing times of the methods are reported by distinguishing between the time required by the SV selection step (Selection Time) and the time required by the MCS solver (Optimization Time). The re-estimation tables correspond to the refinement of the submodels by standard SVM for regression [85] applied independently to each group of data according to the classification estimated by $\hat{q}_k$ (5.11). All the compared methods use the same kernel hyperparameters, regularization trade-off $C = 100$ and quadratic loss function (1.9). So does the re-estimation procedure. Note that all numbers in the tables below account for averages and standard deviations over 100 trials with different random noise sequences.

Regression Example

Consider the function arbitrarily switching between two nonlinear behaviors as

$$y(x) = \begin{cases} x^2, & \text{if } q = 1, \\ \sin(3x) + 2, & \text{if } q = 2. \end{cases} \tag{5.38}$$

A training set of $N = 2000$ points is generated by (6.11) with additive zero-mean Gaussian noise (standard deviation $\sigma_v = 0.3$) for uniformly distributed random $x_k \in [-3, 3]$ and uniformly distributed random q_k. The data are shown in Figure 5.1 as black dots. The difficulty of this toy example lies in the crossing of the submodels, which results in strongly mixed data at particular locations (e.g., for $-1 < x < -0.2$ in Fig. 5.1).

In this experiment, the training data are normalized to zero mean and unit variance. The optimization program (5.10) is solved with two reduced-size submodels of the form (5.9) using Gaussian RBF kernels of width $\sigma_1 = 0.8$ and $\sigma_2 = 0.2$, respectively. Representative examples of the resulting submodels are shown in Figure 5.1. Table 5.1 shows the results. For a comparison, the FIT of the reference model obtained by applying the re-estimation procedure from the true classification is 93.50 ± 2.91.

The classification error rates on the training set as low as 10% show that the algorithm is able to correctly separate between the two modes. Remaining classification errors are mostly due to indistinguishable points at the intersection of the two nonlinear functions. Thus they do not incur significant errors in the re-estimation step, which, according to the FITa, leads to accurately refined models, especially for the RKPCR method.

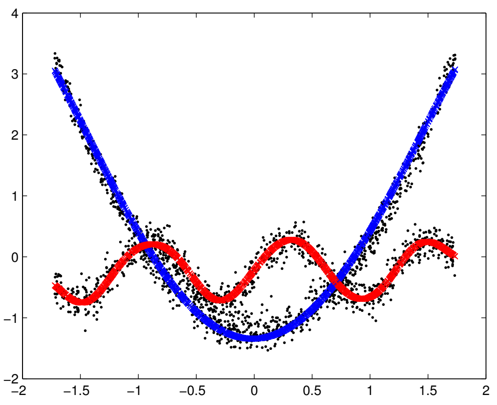


Figure 5.1: Estimation of a switched nonlinear function from 2000 noisy data points (black dots). The red and blue curves show the estimated reduced-size submodels based on the KPCR method.

Table 5.1: Comparison of the four proposed methods to build and estimate reduced-size kernel hybrid models.

Method	Entropy max.	FVS	KPCR	RKPCR
Estimation				
M_1	4	7.0 ± 0.7	3 ± 0	3 ± 0
M_2	17	14.9 ± 1.8	10 ± 0	10 ± 0
FITa(%)	87.06 ± 2.03	87.95 ± 3.83	88.33 ± 3.89	86.20 ± 2.17
FITb(%)	81.51 ± 21.04	78.25 ± 26.48	88.21 ± 3.97	82.35 ± 21.43
Test Classif. err. (%)	5.22 ± 9.94	7.71 ± 10.63	2.14 ± 1.08	4.32 ± 5.35
Train. Classif. err. (%)	8.47 ± 9.65	10.80 ± 9.92	5.25 ± 0.58	7.30 ± 4.80
Selection Time (s)	0.94 ± 0.01	7.27 ± 1.13	8.80 ± 0.80	0.06 ± 0.04
Optimization Time (s)	3.1 ± 0.7	3.3 ± 1.1	1.9 ± 0.5	1.3 ± 0.4
Re-estimation				
FITa(%)	91.51 ± 3.50	92.36 ± 2.44	89.81 ± 4.26	92.75 ± 2.70
FITb(%)	85.77 ± 22.33	82.00 ± 27.4	89.69 ± 4.340	88.92 ± 22.20
Test Classif. err. (%)	4.75 ± 10.26	6.65 ± 10.70	2.13 ± 1.10	3.05 ± 5.45

Switched Nonlinear Dynamical System

This next example considers the identification of a dynamical system arbitrarily switching between two modes as

$$y_k = \begin{cases} 0.9y_{k-1} + 0.2y_{k-2}, & \text{if } q_k = 1, \\ (0.8 - 0.5\exp(-y_{k-1}^2))y_{k-1} - (0.3 + 0.9\exp(-y_{k-1}^2))y_{k-2} + & \\ \quad 0.4\sin(2\pi y_{k-1}) + 0.4\sin(2\pi y_{k-2}), & \text{if } q_k = 2. \end{cases} \tag{5.39}$$

A training set of $N = 2000$ points is generated by (5.54) with a uniformly distributed random sequence of $q_k \in \{1, 2\}$ and an additive zero-mean Gaussian noise (standard deviation $\sigma_v = 0.1$) from the initial condition $y_0 = y_{-1} = 0.1$, whereas the noise-free test set uses $y_0 = 0.4$, $y_{-1} = -0.3$. Note that the noise is added to y_k during the data generation process, resulting in colored noise.

For the identification, the submodel f_1 uses a linear kernel with an arbitrary number of SVs $M_1 = 5$ for the entropy maximization method (this is a fictive number, as the two linear parameters can be recovered from linear combinations of the SVs), while f_2 uses a Gaussian RBF kernel ($\sigma = 0.3$). Corresponding results are reported in Table 5.2. For a comparison, the FIT of the reference model with known mode is 92.79 ± 2.67. In these experiments, the PCA-based methods (KPCR and RKPCR) yield better FITs and fewer classification errors for a low computing time.

Table 5.2: Estimation of an arbitrarily switched nonlinear ARX system.

Method	Entropy max.	FVS	KPCR	RKPCR
Estimation				
M_1	5	2.0 ± 0	2.0 ± 0	2.0 ± 0
M_2	30	28.1 ± 3.8	36.8 ± 2.4	37.0 ± 2.7
FITa(%)	71.22 ± 2.75	69.57 ± 4.01	80.76 ± 3.44	81.67 ± 3.24
FITb(%)	53.86 ± 8.59	56.77 ± 8.50	73.17 ± 5.31	75.81 ± 4.75
Test Classif. err. (%)	20.16 ± 4.37	17.26 ± 4.84	8.85 ± 2.61	7.74 ± 2.31
Train. Classif. err. (%)	21.69 ± 3.60	19.34 ± 4.18	12.67 ± 2.28	11.94 ± 2.00
Selection Time (s)	1.07 ± 0.06	18.21 ± 4.15	1.85 ± 0.13	1.93 ± 0.15
Optimization Time (s)	6.5 ± 2.0	4.9 ± 2.0	6.60 ± 2.90	7.42 ± 3.07
Re-estimation				
FITa(%)	86.03 ± 2.36	85.17 ± 4.39	89.05 ± 4.63	90.03 ± 3.69
FITb(%)	77.19 ± 7.95	77.05 ± 8.95	83.88 ± 4.99	84.71 ± 4.08
Test Classif. err. (%)	12.29 ± 4.45	9.19 ± 5.16	4.40 ± 1.73	3.86 ± 1.18

5.2 Error sparsification framework

This section extends the error sparsification method (ES) presented in Sect. 2.2.2 (page 21) for switched linear systems in several ways to the case of switched nonlinear systems. Firstly, we introduce the notion of robust sparsity to relax the conditions on the noise under which the method can yield optimal estimates. Then, a convex relaxation is proposed to allow for the optimization of the robust sparsity through the minimization of a modified ℓ_1-norm. Finally, we show that the resulting convex optimization program can be equivalently formulated as the minimization of the ε-insensitive loss function proposed in the machine learning community for Support Vector Regression (SVR) [85], a particular instance of the Support Vector Machine (SVM) [96].

The results have been published in [C3] (see the Author's bibliography, page XVIII).

5.2.1 Extension to nonlinear submodels

When the subsystems are nonlinear, the error sparsification framework can be extended by replacing the linear submodels $f_j(\boldsymbol{x}_k) = \boldsymbol{\theta}^\top \boldsymbol{x}_k$ by an expansion over a set of basis functions or a function f_j in a Reproducing Kernel Hilbert Space (RKHS).

The error sparsification framework estimates submodels one by one. For each estimated submodel, data points associated to it are removed from the data set before continuing (see Sect. 2.2.2). More precisely, for a data set $\mathcal{D}$ and a given function class $\mathcal{H}$, submodels $\{f_j\}_{j=1}^s$ are learned one by one by solving

$$\min_{f \in \mathcal{H}} \sum_{k=1}^{N} \|y_k - f(\boldsymbol{x}_k)\|_0 + \lambda \mathcal{R}(f). \tag{5.40}$$

By defining the error vector $\boldsymbol{e} \in \mathbb{R}^N$ with components $e_k = y_k - f(\boldsymbol{x}_k)$, we recover the optimization program (2.22) with an additional regularization term $\mathcal{R}(f)$,

$$\begin{aligned} &\min_{\boldsymbol{e}, f \in \mathcal{H}} \ \|\boldsymbol{e}\|_0 + \lambda \mathcal{R}(f), \\ &\text{s.t. } \boldsymbol{e} = \boldsymbol{y} - \boldsymbol{f}, \end{aligned} \tag{5.41}$$

where $\boldsymbol{f} = [f(\boldsymbol{x}_1), \dots, f(\boldsymbol{x}_N)]^\top$ and $\boldsymbol{y} = [y_1, \dots, y_N]^\top$. Another equivalent form of (5.41) is

$$\min_{f \in \mathcal{H}} \ \|\boldsymbol{y} - \boldsymbol{f}\|_0 + \lambda \mathcal{R}(f). \tag{5.42}$$

These optimization problems are non-convex due to the ℓ_0-norm. Then, a convex relaxation uses the ℓ_1-norm instead of the ℓ_0-norm:

$$\min_{f \in \mathcal{H}} \ \|\boldsymbol{y} - \boldsymbol{f}\|_1 + \lambda \mathcal{R}(f). \tag{5.43}$$

The regularizer $\mathcal{R}(f)$ depends on an arbitrary choice of nonlinear structure for the model. In the next section, we present the typical choices of the regularizer and the nonlinear model.

5.2.2 Choice of the regularizer

In machine learning, it is a well-known fact that one cannot learn without a minimal set of assumptions on the target function. In the most general case, where no prior knowledge is available, the less informative assumption concerns the smoothness of the target function. Indeed, without assuming that function values should be close for two points that are close in the regression space $\mathcal{X}$, one cannot learn from a finite set of points and generalize to others. In practice, the smoothness assumption is typically implemented by regularization. We now discuss two particular choices for the regularizer $\mathcal{R}(f)$.

Sparsity inducing regularization

In [7], the nonlinear model is fixed a priori in the form of a kernel expansion (5.4). Then, a regularization term based on the ℓ_0-norm of the parameter vector $\boldsymbol{\alpha}$ is introduced, before being relaxed to the convex ℓ_1-norm. While the ℓ_1-norm is a typical choice for regularization, which is also known for its sparsity inducing feature, ℓ_0-norm regularization is more ambiguous regarding the resulting smoothness of f. Therefore, in this case, the aim of minimizing the ℓ_1-norm is not to recover the smallest ℓ_0-norm solution through a convex relaxation, and we will not delve into theoretical guarantees of convergence of the ℓ_1-solution to the ℓ_0-solution.

Then, with these choices, a submodel is estimated by solving

$$\min_{\boldsymbol{\alpha} \in \mathbb{R}^N} \ \|\boldsymbol{y} - \boldsymbol{K}\boldsymbol{\alpha}\|_1 + \lambda \|\boldsymbol{\alpha}\|_1, \tag{5.44}$$

where $\boldsymbol{K}$ is the Gram matrix of the kernel κ.

Capacity control regularization

The typical approach used in machine learning to estimate nonlinear functions is to control the capacity of the model by penalizing the nonsmoothness of f. This can be measured through a norm of f.

Using the natural RKHS squared norm defined in (1.17), $\mathcal{R}(f) = \frac{1}{2}\|f\|_{\mathcal{H}}^2$, the nonlinear submodels are estimated by solving the convex optimization problem

$$\min_{f \in \mathcal{H}} \|\boldsymbol{y} - \boldsymbol{f}\|_1 + \frac{\lambda}{2}\|f\|_{\mathcal{H}}^2. \tag{5.45}$$

With these choices, the obtained optimization problem is identical to (1.14) where the absolute loss function (1.10) is used. Therefore, by application of the representer theorem (Theorem 1), we conclude that the solution of (5.45) is in the form of (5.4) whereas it is fixed a priori in the previous choice of the regularizer.

5.2.3 Robust sparsity

A preliminary condition to the sparse recovery conditions, such as the ones in Theorems 2 and 3, used by the ES method for hybrid system identification is that the data must be noiseless. Indeed, with noisy data, no (or very few) entries of the error vector can be zero[1], hence breaking the sparsity of the optimal solution.

In order to circumvent the issue of the lack of zeros in the error vector, we introduce robust sparsity as defined through the pseudo-norm

$$\|\boldsymbol{e}\|_{0,\varepsilon} = \left|\{k \ : \ |e_k| > \varepsilon\}\right|.$$

Under a bounded noise assumption of the type $\|\boldsymbol{v}\|_\infty \leq \varepsilon$, the error vector, $\boldsymbol{e} = [y_1 - f(\boldsymbol{x}_1), \ \dots, \ y_N - f(\boldsymbol{x}_N)]^\top$, can be *robustly sparse*, i.e., with a small value of $\|\boldsymbol{e}\|_{0,\varepsilon}$, if f is a sufficiently good approximation of one of the target submodels f_j.

Instead of the nonconvex pseudo-norm above, we consider the following convex relaxation based on a modified ℓ_1-norm:

$$\|\boldsymbol{e}\|_{1,\varepsilon} = \sum_k (|e_k| - \varepsilon)_+ = \sum_k \max\{0, \ |e_k| - \varepsilon\},$$

which is defined as a sum of pointwise maximum of convex functions of e_k and hence is convex with respect to all components e_k. In the following, we will refer to the pseudo norm above as the $\ell_{1,\varepsilon}$-norm. With this norm, we obtain from (5.44) and (5.45) the two following robust convex optimization programs,

$$\min_{\boldsymbol{\alpha} \in \mathbb{R}^N} \|\boldsymbol{y} - \boldsymbol{K\alpha}\|_{1,\epsilon} + \lambda\|\boldsymbol{\alpha}\|_1, \tag{5.46}$$

and

$$\min_{f \in \mathcal{H}} \|\boldsymbol{y} - \boldsymbol{f}\|_{1,\epsilon} + \frac{\lambda}{2}\|f\|_{\mathcal{H}}^2, \tag{5.47}$$

where f has the form of (5.4).

5.2.4 Connection with Support Vector Machines

We will present in this section connections with support vector machines of the two robust convex optimization programs (5.46) and (5.47).

[1]With nonlinear models of sufficient capacity, the error vector can actually be zero. But, as already discussed, this is not a desirable case, since this would clearly indicate overfitting. Here, we focus on sufficiently regularized (and desirable) solutions, for which the error vector cannot be sparse.

Firstly, Problem (5.46) can be written as the linear program

$$\begin{aligned}\min_{\boldsymbol{\alpha},\boldsymbol{a},\boldsymbol{\xi}} \quad & \mathbf{1}^\top \boldsymbol{a} + C\mathbf{1}^\top \boldsymbol{\xi} \\ \text{s.t.} \quad & -\boldsymbol{\xi} - \varepsilon\mathbf{1} \leq \boldsymbol{y} - \boldsymbol{K}\boldsymbol{\alpha} \leq \varepsilon\mathbf{1} + \boldsymbol{\xi} \\ & -\boldsymbol{a} \leq \boldsymbol{\alpha} \leq \boldsymbol{a},\end{aligned} \tag{5.48}$$

with $C = 1/\lambda$. Here, the objective function has been divided by λ in order to emphasize the equivalence with the training algorithm of the so-called Linear Programming Support Vector Regression (LP-SVR) proposed in [65] for nonlinear function approximation.

Secondly, it is known that the form (5.4) can be written as a linear form (5.3) into a feature space $\mathcal{F}$,

$$f(\boldsymbol{x}) = \langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x})\rangle_{\mathcal{F}}, \tag{5.49}$$

with parameters $\boldsymbol{w} \in \mathcal{F}$. In order to emphasize the relationship with SVMs below, we further consider the affine model $\tilde{f} = f + b$, with $b \in \mathbb{R}$.

With these notations, and a simple substitution of $\tilde{f}$ for f in the computation of the loss, problem (5.47) can be written as

$$\begin{aligned}\min_{\boldsymbol{w},b,\xi_k,\xi_k'} \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{k=1}^N (\xi_k + \xi_k') \\ \text{s.t.} \quad & y_k - \langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}_k)\rangle_{\mathcal{F}} - b \leq \varepsilon + \xi_k \\ & y_k - \langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}_k)\rangle_{\mathcal{F}} - b \geq -\varepsilon - \xi_k' \\ & \xi_k \geq 0,\ \xi_k' \geq 0,\end{aligned} \tag{5.50}$$

with $C = 1/\lambda$, which is the primal form of the training algorithm of a support vector machine for nonlinear regression (SVR) [85]. Note that, $\boldsymbol{\phi}$ and $\mathcal{F}$ are only implicit and need not be known nor finite-dimensional, and so does $\boldsymbol{w}$. What is known however is that, by construction, $\langle \boldsymbol{\phi}(\boldsymbol{x}), \boldsymbol{\phi}(\boldsymbol{x}')\rangle_{\mathcal{F}} = \kappa(\boldsymbol{x}, \boldsymbol{x}')$. Thus, by Lagrangian duality, this problem can be reformulated as the finite-dimensional quadratic program

$$\begin{aligned}\max_{\boldsymbol{\beta},\boldsymbol{\beta}'} \quad & -\frac{1}{2}(\boldsymbol{\beta} - \boldsymbol{\beta}')^\top \boldsymbol{K}(\boldsymbol{\beta} - \boldsymbol{\beta}') - \varepsilon\mathbf{1}^\top(\boldsymbol{\beta} + \boldsymbol{\beta}') + \boldsymbol{y}^\top(\boldsymbol{\beta} - \boldsymbol{\beta}') \\ \text{s.t.} \quad & \mathbf{1}^\top(\boldsymbol{\beta} - \boldsymbol{\beta}') = 0 \\ & 0 \leq \boldsymbol{\beta} \leq C,\ 0 \leq \boldsymbol{\beta}' \leq C,\end{aligned} \tag{5.51}$$

which involves $\boldsymbol{\phi}$ only through the (computable) matrix $\boldsymbol{K}$. Then, the solution of the primal is given by $\boldsymbol{w} = \sum_{k=1}^N \alpha_k \boldsymbol{\phi}(\boldsymbol{x}_k)$, where $\alpha_k = \beta_k - \beta_k'$. The reader is referred to [85] for more details on the derivation of (5.51) and on the computation of b.

Finally, in the context of SVR, the ϵ-insensitive loss function is defined as

$$\ell_\epsilon(y, \hat{y}) = \max\{0, |y - \hat{y}| - \epsilon\}, \tag{5.52}$$

and can be used to interpret the robust sparsity in (5.47) as

$$\|\boldsymbol{y} - \boldsymbol{f}\|_{1,\epsilon} = \sum_{k=1}^N \ell_\epsilon\left(y_k, f(\boldsymbol{x}_k)\right).$$

5.2.5 Tuning of the threshold ε

Regarding the tuning of the threshold ε, the following different cases must be considered.

Bounded-error approach

As in the bounded-error approaches presented in Sect. 2.2.2 (page 20) for linear hybrid system identification, the number of submodels is estimated in order to satisfy, for a predefined threshold δ, a bound of the form $|y_k - f(\boldsymbol{x}_k)| < \delta$, for all data points. Such a bound is optimal in the bounded noise case, with $\delta = \|\boldsymbol{v}\|_\infty$, where $\boldsymbol{v}$ is the concatenation of all noise terms. But it is also more general in the sense that it does not require a noise model. Indeed, the aim is to obtain a set of submodels which approximate the data with a given tolerance. Thus, the parameter δ allows one to tune the trade-off between the model complexity (measured as the number of submodels) and the fit to the data.

Following these works, a similar strategy applies to the proposed method, where ε plays a similar role as δ.

With assumptions on the noise model

Optimal values for ε have been investigated in the context of SVR under various noise models by different authors. The results in [55] are summarized in Table 5.3.

Table 5.3: Optimal values of ε with a noise model of standard deviation equal to σ_v.

Noise model	Uniform	Laplacian	Gaussian
Optimal ε	σ_v	0	$1.0043\sigma_v$

An additional difficulty with Gaussian or Laplacian noise is that the criterion, $|y_k - f(\boldsymbol{x}_k)| \leq \varepsilon$, used to remove data points after the estimation of a submodel becomes suboptimal: data points with larger noise terms are not removed. In this case, the complete procedure must be stopped when a sufficiently small, but not too small, number of data points remain in the data set. The rationale here is that points with a low noise magnitude are used to estimate the submodels, while the others are considered as outliers. We further assume that these outliers represent a small fraction of the data set. Since at each iteration of the error sparsification approach, a submodel is estimated from a set of points representing a large fraction of the remaining data, it is expected that outliers are left unused until the end of the procedure.

Automatic tuning

In the SVM literature, problem (5.50) is sometimes referred to as ε-SVR to distinguish it from the alternative ν-SVR [81] which allows for the automatic tuning of ε. In the derivation of ν-SVR, the trick is to add a term in the objective function of (5.50) in order to minimize ε while learning the model. This leads to

$$\min_{f\in\mathcal{H},\varepsilon\in\mathbb{R}^+} \ \frac{1}{2}\|f\|_{\mathcal{H}}^2 + \frac{C}{N}\sum_{k=1}^{N} \ell_\varepsilon(y_k, f(\boldsymbol{x}_k)) + C\nu\varepsilon, \tag{5.53}$$

where $\nu \geq 0$ is a new hyperparameter tuning the trade-off between the minimization of ε and the minimization of the errors larger than ε. As for the ε-SVR, the solution to this new fomulation is obtained in the form of (5.4) by solving the dual. However, in this case, the hyperparameter ν enjoys a number of properties which can ease its tuning when compared to ε in (5.50). In particular, it is shown in [81] that $\nu > 1$ yields $\varepsilon = 0$ and that, if $\varepsilon > 0$, $\nu \in [0, 1]$ can be interpreted as the fraction of data points outside of the ε-tube of insensitivity, i.e., $\nu \approx \|\boldsymbol{e}\|_{0,\varepsilon}$.

A similar approach can be followed in the case of ℓ_1-norm regularization. This leads to a formulation of the LP-SVR allowing for the automatic tuning of ε via linear programming as proposed in [84] or [65].

5.2.6 Iteratively reweighted scheme

As in the classical case for error sparsification, a reweighted scheme can be used to improve the recovery of robustly sparse solutions with low sparsity. This leads for instance to

$$\min_{f\in\mathcal{H}} \frac{1}{2}\|f\|^2_{\mathcal{H}} + C\sum_{k=1}^{N} w_k\ \ell_\varepsilon(y_k, f(\boldsymbol{x}_k)),$$

with w_k defined as in Sect. 1.4. Such a formulation corresponds to a Weighted-SVR, which has been proposed (with fixed weights) by [91] and others in order to deal with a varying confidence in the data points or to introduce various forms of prior knowledge.

5.2.7 Algorithmic and implementation issues

The theoretical equivalence between nonlinear hybrid system identification via error sparsification and support vector regression also yields direct algorithmic benefits. In particular, this means that the problem can be solved efficiently even for large data sets (e.g., with more than ten thousand points).

First, note that all the considered convex formulations, e.g., (5.44) or (5.46), are theoretically simple optimization problems due to their convexity. However, despite the possibility to write them as linear programs, e.g., as in (5.48), solving large-scale instances of such problems requires much more care in practice. In particular, a major issue concerns the memory requirements: the data of the problem, including the (typically dense) N-by-N Gram matrix $\boldsymbol{K}$, simply cannot be stored in the memory of most computers. This basic limitation prevents any subsequent call to a general purpose optimization solver in many cases.

On the other hand, dedicated optimization algorithms have been proposed to train SVMs and benefit from numerous advances in this active field of research, see, e.g., [17]. SVM algorithms typically use decomposition techniques such as sequential minimal optimization (SMO) [75, 82] to avoid the storage of the matrix $\boldsymbol{K}$ in memory. With a proper working set selection strategy, the solution can even be found without having to compute all the elements of the matrix $\boldsymbol{K}$, thus reducing both the memory and computing load. Good SVM solvers implementing these ideas are for instance SVM^{light} [88] or LibSVM [22]. The latter also implements the Weighted-SVR and can be used in the iteratively reweighted version of the procedure for hybrid system identification, as discussed in Sect. 5.2.6. For ℓ_1-norm regularization, efficient algorithms are developed in [65]. Finally, these solvers also apply to the original error sparsification approach presented in Sect. 2.2.2 (page 21) (without robust sparsity) simply by setting $\varepsilon = 0$.

Thus, by showing the equivalence between the robust sparsity optimization approach and support vector regression, we also make the problem tractable for off-the-shelf (and usually freely available) solvers.

5.2.8 Numerical experiments

We now turn to illustrative examples of application with a regression example (Sect. 5.2.8) and switched system identification examples (Sect. 5.2.8).

Regression example

The first illustrative example considers the approximation of two overlapping nonlinear functions (a sinusoid and a quadratic) from a set of 3000 data points with Gaussian noise of standard deviation $\sigma_v = 0.5$. The submodels are estimated by solving (5.51) with LibSVM for a Gaussian RBF kernel ($\sigma = 0.5$), $C = 100$ and ε set as in Table 5.3 for the Gaussian noise model ($\varepsilon = 0.50215$). The first row of Figure 5.2 shows the first submodel obtained when either one of the two functions dominates the other in terms of the fraction of data points. In both cases, the method correctly estimates the submodel corresponding to the dominating mode. Then, after removing the points close to this submodel, a second submodel is estimated. By thresholding the absolute error, $|y_k - f(\boldsymbol{x}_k)|$, at either 3ε (2nd row of Fig. 5.2) or ε (last row) in the test for removing points, a sufficient fraction of data are eliminated to allow for the recovery of the second submodel. However, with a threshold of ε, a significant fraction (a bit less than 1/3) of the data remains at the end of the procedure. Then, either the number of submodels is assumed fixed to 2 and the algorithm returns the 2 submodels, or the bounded-error approach is applied and the algorithm continues to estimate additional submodels until all the data are removed.

In this example, we observed that the reweighted scheme of Sect. 5.2.6 slightly improves the submodels, while the first iteration already yields a satisfactory discrimination between the two modes due to the large fraction of points associated to the dominating one (about 66%). Figure 5.3 shows the influence of reweighting when this fraction is closer to 50%. For 50.25%, the first iteration is not very accurate, but 10 iterations of the reweighted scheme provide a good approximation of the target submodel. For exactly 50% of data of each mode, the estimated model switches between the two target submodels and cannot discriminate between the modes.

Switched nonlinear system examples

We now consider the switched nonlinear system example, where the aim is to identify a dynamical system arbitrarily switching between two modes as

$$y_k = \begin{cases} 0.9y_{k-1} + 0.2y_{k-2} + v_k, & \text{if } q_k = 1, \\ (0.8 - 0.5\exp(-y_{k-1}^2))y_{k-1} - (0.3 + 0.9\exp(-y_{k-1}^2))y_{k-2} + & \\ \quad 0.4\sin(2\pi y_{k-1}) + 0.4\sin(2\pi y_{k-2}) + v_k, & \text{if } q_k = 2. \end{cases} \tag{5.54}$$

A training set of $N = 3000$ points is generated by (5.54) with a random sequence of q_k ($P(q_k = 1) = 2/3$ and $P(q_k = 2) = 1/3$), initial conditions $y_0 = y_{-1} = 0.1$, and an additive zero-mean Gaussian noise v_k of standard deviation $\sigma_v = 0.1$.

In this example, the linearity of the first mode is assumed to be known. Thus, a first submodel, f_1, is estimated with a linear kernel and $\varepsilon = 1.0043\sigma_v$, yielding the parameter estimates[2] reported in Table 5.4 (first column). Then, the points with $k \in I_R = \{k : |y_k - f_1(\boldsymbol{x}_k)| \leq 3\varepsilon\}$ are removed and a nonlinear submodel with a Gaussian RBF kernel ($\sigma = 0.5$) is estimated.

Similar experiments with the reversed order (nonlinear submodel estimated first) are also conducted on a data set with $P(q_k = 1) = 1/3$ (results in Table 5.4, second column).

The quality of the estimation is evaluated for each mode j in terms of the FIT criterion computed on a test set of 2000 data (generated without noise from the initial conditions

[2]With a linear kernel, the parameters of a linear submodel (5.4) are recovered by $\boldsymbol{\theta} = \sum_{i=1}^{N} \alpha_i \boldsymbol{x}_i$.

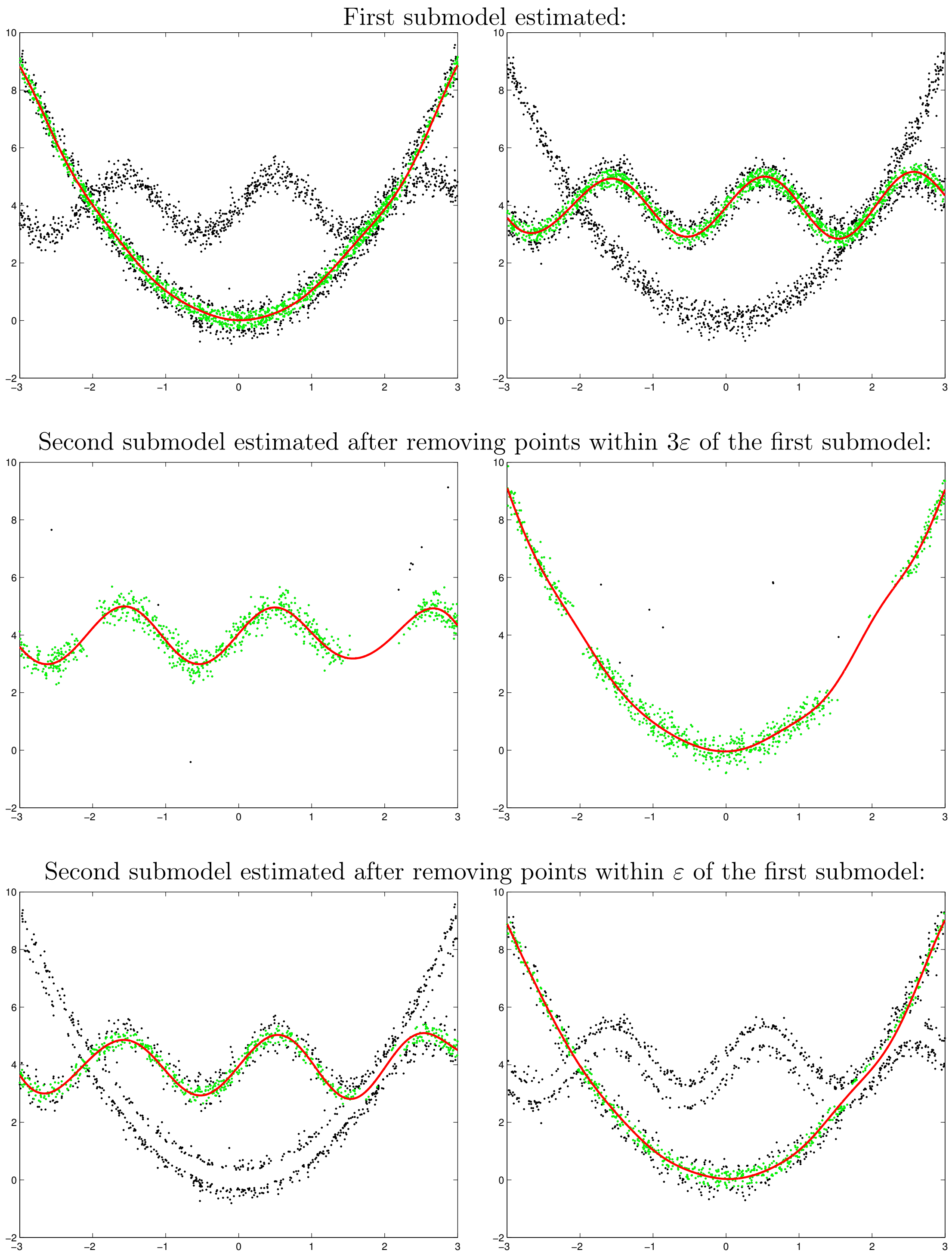


Figure 5.2: Illustration of the procedure depending on which one of the quadratic (left column) or the sinusoidal (right column) mode dominates the data set.

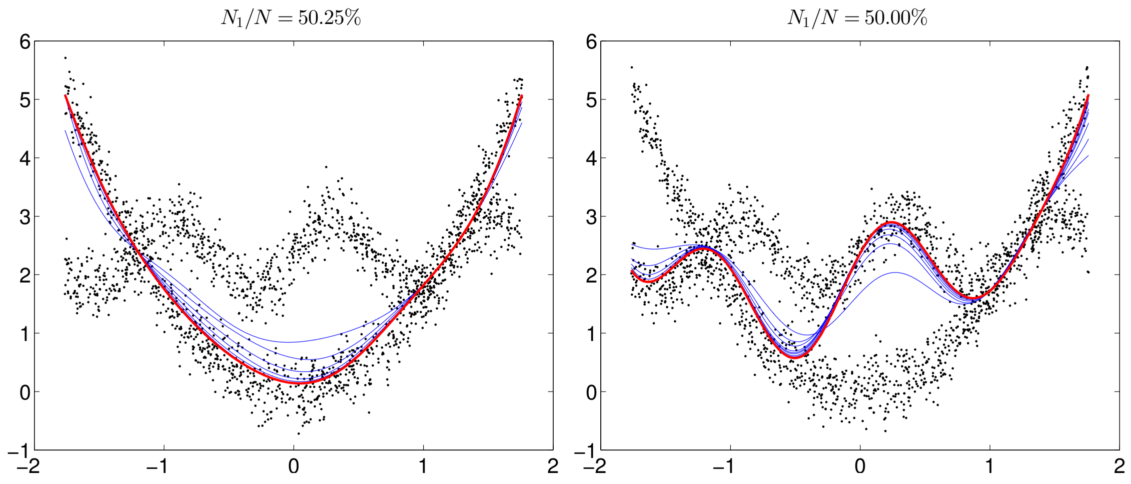


Figure 5.3: Iterations of the reweighting process for $N_1/N = 50.25\%$ (left) and $N_1/N = 50\%$ (right), where N_1 is the number of points generated by the quadratic.

$y_0 = 0.4,\ y_{-1} = -0.3$) as

$$\mathrm{FIT}_j = \left(1 - \frac{\sqrt{\sum_{k\in I_j}(y_k - f_{q_k}(\boldsymbol{x}_k))^2}}{\sqrt{\sum_{k\in I_j}(y_k - m_j)^2}}\right),$$

where $I_j = \{k \ : \ q_k = j\}$ and m_j is the mean of y_k over all $k \in I_j$. Two additional performance indexes are used to evaluate the ability of the method to discriminate between the two modes during training: the fraction of data that must be removed and have been,

$$\mathrm{D1} = \frac{|I_1 \cap I_R|}{|I_1|},$$

and the fraction of data that must be removed among those that have been,

$$\mathrm{D2} = \frac{|I_1 \cap I_R|}{|I_R|}.$$

Note that these numbers are computed on the training data. The results, shown in Table 5.4, emphasize the accuracy of the estimated submodels and the fact that the proposed method correctly discriminates between the two modes, independently of the dominating mode.

Table 5.5 shows similar results for a SNARX system with two nonlinear modes given by

$$y_k = \begin{cases} 0.4y_{k-1}^2 + 0.2y_{k-2} + v_k, & \text{if } q_k = 1, \\ (0.8 - 0.5\exp(-y_{k-1}^2))y_{k-1} - (0.3 + 0.9\exp(-y_{k-1}^2))y_{k-2} + & \\ \quad 0.4\sin(2\pi y_{k-1}) + 0.4\sin(2\pi y_{k-2}) + v_k, & \text{if } q_k = 2. \end{cases} \tag{5.55}$$

For this example, training trajectories of $N = 16000$ points are generated with 6000 points for mode 1 and 10000 points for mode 2 ($P(q_k = 1) = 0.375$). On these large-scale data sets, the average computing time was about one minute for each SVR training by LibSVM, i.e., for each iteration of the reweighted scheme.

Table 5.4: Estimation of the system (5.54) switching between a linear mode (with parameters θ_1, θ_2) and a nonlinear mode. Numbers are averages and standard deviations over 100 trials with different noise, v_k, and mode, q_k, sequences.

$P(q_k = 1)$	2/3	1/3
θ_1 (= 0.9)	0.9008 ± 0.0092	0.9000 ± 0.0070
θ_2 (= 0.2)	0.1824 ± 0.0111	0.2019 ± 0.0068
FIT_1 (%)	97.9148 ± 0.8136	99.1472 ± 0.3261
FIT_2 (%)	83.7052 ± 5.3668	84.8237 ± 6.2603
D1 (%)	99.6840 ± 0.1383	98.7180 ± 0.4904
D2 (%)	88.4713 ± 0.6781	85.7150 ± 0.7850

Table 5.5: Estimation of the system (5.55). Numbers are averages and standard deviations over 100 trials with different noise, v_k, and mode, q_k, sequences.

FIT_1 (%)	73.2515 ± 4.2561
FIT_2 (%)	88.6945 ± 1.0960
D1 (%)	99.1160 ± 0.1618
D2 (%)	78.6506 ± 0.2756

5.3 Conclusions

This chapter focused on the switched regression problem at the core of hybrid system identification in the particular case of systems switching between unknown nonlinear dynamics. In the first part of the chapter, the continuous optimization approach relies on the ability to express each submodel in a sparse kernel form, which allows a global optimization solver to efficiently estimate the parameters of the model. Four methods were proposed and compared for the selection of a subset of the training data on the basis of which such reduced-size models can be built. The entropy maximization approach requires to fix the model size arbitrarily or through the heuristic (5.14) for Gaussian RBF kernels. On the other hand, the other approaches can determine the model size either as a byproduct of the procedure or through a high-level parameter such as the ratio of cumulative energy content. Experiments showed that these latter methods can sufficiently reduce the model size to allow the overall problem to be solved.

The second part of the chapter focused on the error sparsification approach. Conditions of application of this approach were relaxed with the introduction of robust sparsity as a means to deal with noise in the data. We then emphasized the connections between this approach and the support vector machines developed in the field of machine learning. In particular, we have shown that nonlinear hybrid systems can be identified efficiently from large data sets by a sequence of SVM trainings. In addition, this formal equivalence allowed for the derivation of a modified algorithm for the automatic determination of the main hyperparameter (the threshold ε) in the robust sparsity approach.

Chapter 6

Piecewise smooth system identification

ABSTRACT. *This chapter deals with the problem of piecewise smooth system identification. In the proposed approach, the model belongs to a class of smooth functions. Though constrained to be globally smooth, the trained model can have very large derivatives at particular locations to approximate the nonsmoothness of the target function. This is obtained through the definition of new regularization terms which penalize the derivatives in a location-dependent manner and training algorithms in the form of convex optimization problems. Then, the obtained smooth model is transformed to a piecewise smooth model.*

IN this chapter, we focus on piecewise smooth (PWS) systems in which submodels are smooth functions operating in different regions of the regression space corresponding to different modes. The piecewise affine (PWA) systems are a subclass of the PWS systems and some methods specific to their identification are presented in Sect. 2.3. Here, we propose more generally an approach to deal with the PWS system identification problem, Problem 5 in Chapter 1.

Problem 5. *Given a data set $\mathcal{D} = \{(\boldsymbol{x}_k, y_k)\}_{k=1}^N$ generated by a piecewise smooth system f defined in each region $\Re_j$ by*

$$\forall \boldsymbol{x} \in \Re_j, \quad y = f(\boldsymbol{x}) = f_j(\boldsymbol{x}), \qquad j = 1, \dots, s, \tag{6.1}$$

estimate the number of submodels s, the submodels f_j, $j = 1, \dots, s$ and the regions $\Re_j$, $j = 1, \dots, s$, or the switching boundaries in the regression space.

The proposed approach directly estimates a model of the target function f instead of a collection of submodels. Note that the target function f is nonsmooth due to discontinuities in the function or the derivatives at particular locations or boundaries between regions. By using only one model belonging to a class of smooth functions, the unknown switching sequence no longer introduces difficulties in training. However, the trained model must have very large derivatives at particular locations to approximate the nonsmoothness of the target function and simultaneously be able to avoid the overfitting due to the noise. This is obtained through the definition of new regularization terms which penalize the derivatives in a location-dependent manner. We show how these can be chosen to obtain convex optimization programs leading to the desired properties for the model. Finally, the obtained smooth model can be transformed to a piecewise smooth model as in (6.1) with a post-processing step detailed in Sect. 6.2.

The results have been published in [C1] (see the Author's bibliography, page XVIII).

6.1 Proposed learning framework

As presented in Sect. 1.3.2, given a data set $\mathcal{D} = \{(\boldsymbol{x}_k, y_k)\}_{k=1}^N$, a model in a form of a kernel expansion (as with the Gaussian RBF kernel) over the training set can perfectly fit these data, i.e., $\sum_{k=1}^N \ell(y_k, f(\boldsymbol{x}_k)) = 0$ with any loss function ℓ defined by (1.7). However, this leads to overfit the noise in the data. Hence, a regularizer $\mathcal{R}(f)$ is added in the cost function to avoid this issue. Typical regularizers, used in Support Vector Regression (SVR), are based on the induced norm $\|f\|_{\mathcal{H}}$ in RKHS (1.17).

However, these "global" regularizers (in the sense that minimizing $\|f\|_{\mathcal{H}}$ influences the shape of f over the entire regression space $\mathcal{X}$) are not always suitable. For instance, $\|f\|_{\mathcal{H}}^2$ similarly penalizes a noisy function with many oscillations and a very smooth function with large jumps at a few locations. Thus, in the case of piecewise smooth functions, minimizing $\|f\|_{\mathcal{H}}^2$ yields globally smoother functions and discards optimal solutions without distinguishing them from noisy functions. Therefore, we need to build "local" regularizers which allow us to penalize locally the oscillations due to noise but do not prevent jumps of the model to fit the data.

To build local regularizers, we need some definitions.

For $n \in \mathbb{N}$, we recursively define the differential operator of order n in dimension d, $D^{(n)}$, by

$$D^{(n)} = \nabla \otimes D^{(n-1)} = \begin{bmatrix} \frac{\partial}{\partial x_1} D^{(n-1)} \\ \vdots \\ \frac{\partial}{\partial x_d} D^{(n-1)} \end{bmatrix}, \tag{6.2}$$

where $D^{(0)}$ is the identity operator. In particular, if the function $f : \mathbb{R}^d \to \mathbb{R}$ is of class C^n, then $D^{(1)} f = \nabla f$ is the gradient of f, $D^{(2)} f = \operatorname{vec}(Hf)$ is the vector representation of the Hessian and $D^{(n)} f$ is a function of $\mathbb{R}^d$ to $\mathbb{R}^{d^n}$ that computes all the nth order derivatives of f.

The notations $\odot$ and $\otimes$ denote the Hadamard (entrywise) and Kronecker products, respectively.

6.1.1 Learning with local regularization of higher order

We define a local regularization functional of order (n, p) as

$$\forall f \in C^n,\ \forall \boldsymbol{x} \in \mathcal{X}, \quad R_{n,p}(\boldsymbol{x}, f) = \|D^{(n)} f(\boldsymbol{x})\|_p, \tag{6.3}$$

where $D^{(n)}$ is a differential operator of order n as defined in (6.2) and p is a parameter that selects a particular ℓ_p-norm. Given a function class $\mathcal{H} \subseteq C^n$ and a regularization parameter $\lambda > 0$, the learning problem with a local regularizer as in (6.3) reads

$$\min_{f \in \mathcal{H}} \sum_{k=1}^N \ell(y_k, f(\boldsymbol{x}_k)) + \lambda \sum_{k=1}^M R_{n,p}(\boldsymbol{z}_k, f), \tag{6.4}$$

where local regularization terms are minimized for M sample points $\boldsymbol{z}_k$ in order to globally regularize f over $\mathcal{X}$. Various sampling strategies can be considered here. The $\boldsymbol{z}_k$ can be chosen on a grid in order to obtain a sufficient coverage of $\mathcal{X}$ or equal to the training points,

$\boldsymbol{x}_k$, in order to obtain a representative sample of the data distribution (the latter is used in all experiments of Sect. 6.3). Another choice combining the two features is to sample $\boldsymbol{z}_k$ as perturbed versions of the $\boldsymbol{x}_k$.

6.1.2 Learning algorithms for nonsmooth functions

To be more specific, we now consider the square loss function (1.9), and the Gaussian RBF kernel, $\kappa(\boldsymbol{x}', \boldsymbol{x}) = \exp(-\|\boldsymbol{x}' - \boldsymbol{x}\|_2^2/2\sigma^2)$. We further restrain ourselves to models f, as (5.4), in form of kernel expansions over the training set,

$$f(\boldsymbol{x}) = \sum_{i=1}^{N} \alpha_i \kappa(\boldsymbol{x}_i, \boldsymbol{x}), \tag{6.5}$$

where $[\alpha_1, \dots, \alpha_N]^\top = \boldsymbol{\alpha}$ is the vector of parameters to estimate. The form of f in (6.5) indeed constraints the problem since Theorem 1 (page 7) (Generalized Representer Theorem, [79]) does not apply to (6.4).

By defining the kernel matrix $\boldsymbol{K}$ of elements $K_{ik} = \kappa(\boldsymbol{x}_i, \boldsymbol{x}_k)$ and the target vector $\boldsymbol{y} = [y_1, \dots, y_N]^\top$, the data term of (6.4) can be written as $\|\boldsymbol{y} - \boldsymbol{K}\boldsymbol{\alpha}\|_2^2$. Since f is a linear combination of kernel functions with weights α_i, any derivative of f is linear w.r.t. $\boldsymbol{\alpha}$ and we can rewrite $D^{(n)} f(\boldsymbol{z}_k)$ as

$$D^{(n)} f(\boldsymbol{z}_k) = [\boldsymbol{d}_{k1}, \ \dots, \ \boldsymbol{d}_{kd^n}]^\top \boldsymbol{\alpha} = \boldsymbol{D}_k \boldsymbol{\alpha}. \tag{6.6}$$

This yields the finite-dimensional optimization problem

$$\min_{\boldsymbol{\alpha}} \ \|\boldsymbol{y} - \boldsymbol{K}\boldsymbol{\alpha}\|_2^2 + \lambda \sum_{k=1}^{M} \|\boldsymbol{D}_k \boldsymbol{\alpha}\|_p, \tag{6.7}$$

where the convexity of the data term is obvious from the choice of a convex loss function and where the regularization term is a sum of norms of linear functions, hence convex. Therefore any local solution of (6.7) is a global solution. However, the regularization term makes the cost function nonsmooth.

Sparsity optimization point of view. In (6.7), we are looking for a sparse solution in terms of the vector

$$\boldsymbol{r} = \left[\|D^{(n)} f(\boldsymbol{z}_1)\|_p, \ \dots, \ \|D^{(n)} f(\boldsymbol{z}_M)\|_p \right]^\top$$

by minimizing its ℓ_1-norm, since $\lambda \sum_{k=1}^{M} \|D^{(n)} f(\boldsymbol{z}_k)\|_p = \lambda \|\boldsymbol{r}\|_1$. Therefore, the vector of derivatives, $\boldsymbol{D}_k\boldsymbol{\alpha}$, will be drawn to zero at points where its norm is small, while large derivatives will be left at few points. Note that it is crucial here not to use *squared* ℓ_p-norms in (6.7). For instance, using *squared* ℓ_2-norms amounts to minimizing $\|\boldsymbol{r}\|_2^2$, which would lead to a smooth optimization problem, but not to sparse solutions.

6.1.3 Choice of the ℓ_p-norm

We now describe the algorithms obtained by the choice of p.

ℓ_2-norm ($p = 2$). The most natural choice is $p = 2$, which yields the nonsmooth convex optimization program

$$\min_{\boldsymbol{\alpha}} \ \|\boldsymbol{y} - \boldsymbol{K\alpha}\|_2^2 + \lambda \sum_{k=1}^{M} \sqrt{\boldsymbol{\alpha}^\top \boldsymbol{D}_k^\top \boldsymbol{D}_k \boldsymbol{\alpha}}. \tag{6.8}$$

While practical algorithms have been proposed to solve such problems, we instead eliminate the nonsmoothness of the cost function by considering a constrained (and convex) formulation. This yields the Second-Order Cone Program (SOCP):

$$\begin{aligned} \min_{\boldsymbol{\alpha},\xi,\boldsymbol{t}} \ & \xi + \lambda \sum_{k=1}^{M} t_k \\ \text{s.t.} \ & \left\| \begin{matrix} 1 - \xi/4 \\ \boldsymbol{K\alpha} - \boldsymbol{y} \end{matrix} \right\|_2 \leq 1 + \xi/4, \\ & \|\boldsymbol{D}_k \boldsymbol{\alpha}\|_2 \leq t_k, \quad \forall k = 1, \dots, M. \end{aligned} \tag{6.9}$$

ℓ_∞-norm ($p = \infty$). Another norm which can be employed here is the ℓ_∞-norm, i.e.,

$$R_{n,\infty}(\boldsymbol{z}_k, f) = \|D^{(n)} f(\boldsymbol{z}_k)\|_\infty = \max_{\{i_1,\dots,i_n\} \in \{1,\dots,d^n\}} \left| \frac{\partial^n f(\boldsymbol{z}_k)}{\partial z_{i_1} \dots \partial z_{i_n}} \right|.$$

In this case, the resulting optimization problem can be written as a quadratic program (QP) with $2Mm$ linear constraints, where m is the number of partial derivatives of order n:

$$\begin{aligned} \min_{\boldsymbol{\alpha},\boldsymbol{t}} \ & \boldsymbol{\alpha}^\top \boldsymbol{KK\alpha} - 2\boldsymbol{y}^\top \boldsymbol{K\alpha} + \lambda \sum_{k=1}^{M} t_k \\ \text{s.t.} \ & -t_k \leq \boldsymbol{d}_{ki}^\top \boldsymbol{\alpha} \leq t_k, \quad \forall k = 1, \dots, M, \forall i = 1, \dots, m. \end{aligned} \tag{6.10}$$

Note that by the symmetry of the mixed derivatives, in practice when $d > 1$ and $n > 1$, we have $m < d^n$.

6.1.4 Choice of the regularization order n

The following discusses the choice of n in order to gradually deal with piecewise constant, affine and nonlinear functions.

Piecewise constant (PWC) functions. By choosing $n = 1$, we have $R_{n,p}(\boldsymbol{z}_k, f) = \|\nabla f(\boldsymbol{z}_k)\|_p$, where, for the Gaussian RBF kernel (see Appendix B),

$$\nabla f(\boldsymbol{z}_k) = \frac{1}{\sigma^2} \left(\boldsymbol{X} - \boldsymbol{z}_k \boldsymbol{1}_N^\top \right) \operatorname{diag}\left(\boldsymbol{\kappa}(\boldsymbol{z}_k) \right) \boldsymbol{\alpha},$$

i.e., in (6.7), we set $\boldsymbol{D}_k = \frac{1}{\sigma^2}(\boldsymbol{X} - \boldsymbol{z}_k \boldsymbol{1}_N^\top)\operatorname{diag}(\boldsymbol{\kappa}(\boldsymbol{z}_k))$, where $\boldsymbol{X}$ is the matrix composed of the regression vectors $\boldsymbol{x}_k$ as columns and $\boldsymbol{\kappa}(\boldsymbol{z}_k) = [\kappa(\boldsymbol{x}_1, \boldsymbol{z}_k), \dots, \kappa(\boldsymbol{x}_N, \boldsymbol{z}_k)]^\top$.

The *top row* of Fig. 6.1 shows an example where the aim is to learn a PWC target function over $\mathcal{X} = [-10, 10]$ from a noisy data set (left). We first applied SVR with a Gaussian RBF kernel ($\sigma = 0.2$), but failed to find a satisfactory tuning of the regularization constant, as expected. Indeed, the middle plot of Fig. 6.1 shows that the value $\lambda = 1$

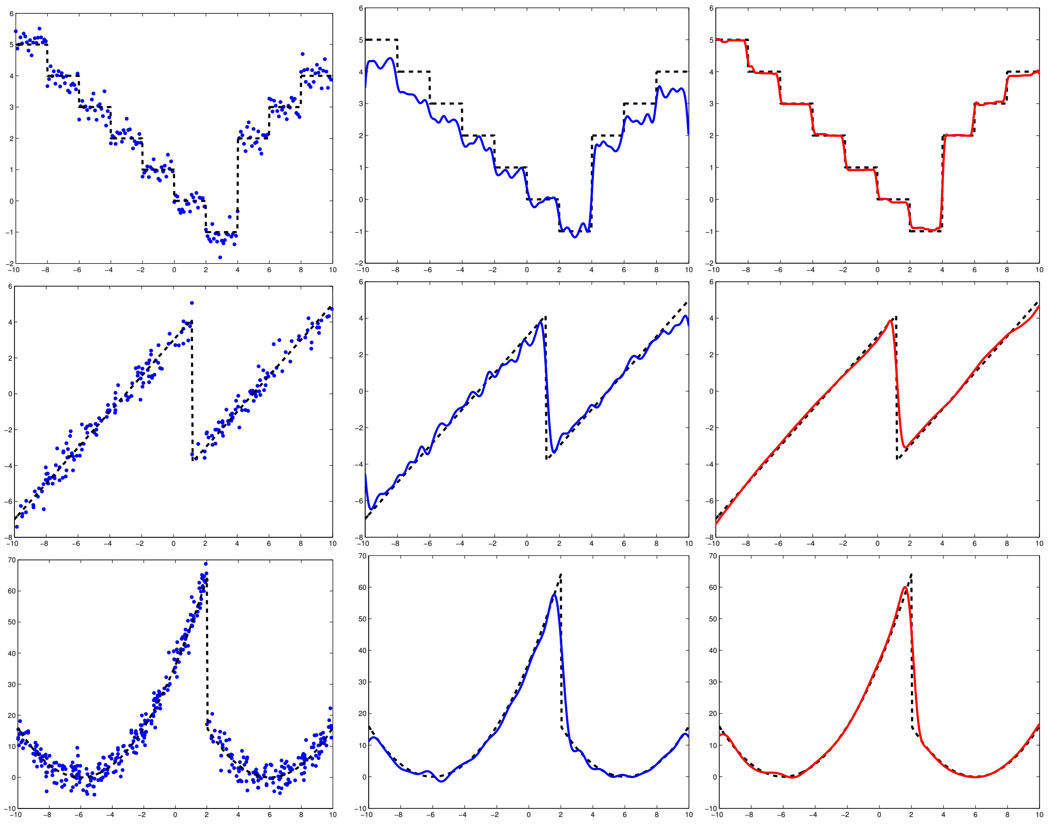


Figure 6.1: From top to bottom: examples of piecewise constant (PWC), affine (PWA) and quadratic (PWQ) function approximation. Dashed lines: nonsmooth target functions. Left: data points. Middle: SVR. Right: proposed method with $n = 1$ for PWC, $n = 2$ for PWA and $n = 3$ for PWS.

is already too large to allow the model to correctly estimate the large variations of the function, while being also too small to counter the effect of the noise. On the contrary, the proposed method with $\lambda = 0.005$ (right plot) yields a good model in terms of both noise removal and accuracy of the nonsmoothness approximation. In addition, regularization constants in the range $0.002 \leq \lambda \leq 0.01$ yield satisfactory solutions.

Piecewise affine (PWA) functions. PWA functions have piecewise constant gradient almost everywhere, which can be obtained by applying the regularizer defined for PWC functions to the gradient vector field, i.e., by minimizing $\|(\nabla \otimes \nabla f)(\boldsymbol{z}_k)\|_p$. This amounts to setting $n = 2$ in (6.6) and (6.7), and to penalizing the Frobenius norm of the Hessian matrix, $\boldsymbol{H}_k$, for $p = 2$ as $R_{2,2}(\boldsymbol{z}_k, f) = \|\boldsymbol{H}_k\|_F$, or its max-norm with $R_{2,\infty}(\boldsymbol{z}_k, f) = \|\boldsymbol{H}_k\|_{\max}$. In such cases, the elements of the Hessian at point $\boldsymbol{z}_k$, are given (see Appendix B) by

$$(\boldsymbol{H}_k)_{jl} = \frac{1}{\sigma^4} \sum_{i=1}^{N} \alpha_i \kappa(\boldsymbol{x}_i, \boldsymbol{z}_k) \left((x_{ij} - z_{kj})(x_{il} - z_{kl}) - \delta_{j,l}\sigma^2 \right),$$

and we set the $((l-1)d + j)$-th row of $\boldsymbol{D}_k$ to

$$\boldsymbol{d}_{k((l-1)d+j)}^\top = \frac{1}{\sigma^4} \left[((\boldsymbol{X}^\top)_j - z_{kj}\mathbf{1}_N) \odot ((\boldsymbol{X}^\top)_l - z_{kl}\mathbf{1}_N) - \delta_{j,l}\sigma^2 \mathbf{1}_N \right]^\top \operatorname{diag}(\boldsymbol{\kappa}(\boldsymbol{z}_k)),$$

where $(\boldsymbol{X}^\top)_j$ is the jth column of $\boldsymbol{X}^\top$.

The example in the *middle row* of Fig. 6.1 considers learning a PWA target function over $\mathcal{X} = [-10, 10]$ with $\sigma = 0.5$. The proposed method (right plot) with $\lambda = 0.01$ can accurately estimate the large central jump while preserving smoothness of the function at all other points. On the other hand, SVR tuned in order to correctly approximate the jump yields a very perturbed model (middle plot). Here, applying SVR with a larger regularization constant would reduce the effect of noise, but also dramatically decrease the accuracy near the nonsmoothness of the target function.

General piecewise smooth (PWS) functions. Regularization of higher order derivatives can be considered in order to learn PWS functions with nonlinear pieces. Here, we present results with third order derivatives particularly suitable for piecewise quadratic (PWQ) functions, while the extension to higher orders is straightforward.

For f defined as in (6.5) and a Gaussian RBF kernel, we have $\partial^3 f(\boldsymbol{z}_k)/\partial z_{kj}\partial z_{kl}\partial z_{km} = \boldsymbol{d}^\top_{k((m-1)d^2+(l-1)d+j)}\boldsymbol{\alpha}$, with (see Appendix B)

$$\begin{aligned}\boldsymbol{d}^\top_{k((m-1)d^2+(l-1)d+j)} = \frac{1}{\sigma^6}\Bigg[&\left((\boldsymbol{X}^\top)_j - z_{kj}\mathbf{1}_N\right) \odot \left((\boldsymbol{X}^\top)_l - z_{kl}\mathbf{1}_N\right) \odot \left((\boldsymbol{X}^\top)_m - z_{km}\mathbf{1}_N\right) \\ &- \delta_{l,m}\sigma^2\left((\boldsymbol{X}^\top)_j - z_{kj}\mathbf{1}_N\right) - \delta_{j,m}\sigma^2\left((\boldsymbol{X}^\top)_l - z_{kl}\mathbf{1}_N\right) \\ &- \delta_{j,l}\sigma^2\left((\boldsymbol{X}^\top)_m - z_{km}\mathbf{1}_N\right)\Bigg]^\top \operatorname{diag}\left(\boldsymbol{\kappa}(\boldsymbol{z}_k)\right).\end{aligned}$$

The last row of Fig. 6.1 shows an example of such a procedure with $\sigma = 0.5$ and $\lambda = 0.05$.

6.2 Obtaining piecewise smooth models

For piecewise smooth system identification, the simple Procedure 12 transforms the solution of (6.7) into a PWS model (6.1) with s modes.

Step 2 of Procedure 12 detects points close to the boundaries of the regions $\Re_j$ by thresholding the norm of derivatives used in (6.7) at a given τ, since the smoothness assumptions on f in (6.7) and f_j in (6.1) imply a zero norm inside each region $\Re_j$. Note that, after solving (6.7) through the optimization of (6.9) or (6.10), the values of the norm at all points are directly given by the slack variables t_k. Steps 3 and 4 assume that the partition $\cup_{j=1}^s \Re_j$ can be represented by a Voronoi diagram [4]. However, more complex partitions can still be represented by increasing the number of regions s. The classifier h estimating the partition in Step 6 can be directly given by the clustering algorithm of Step 4, e.g, k-means yields the centers of Voronoi cells which can be used to estimate labels of new data points. Alternatively, we can train a new classifier h with a supervised algorithm to correct potential errors of the clustering algorithm. Predictions for test points $\boldsymbol{x}$ are then given by $f_q(\boldsymbol{x})$, where $q = h(\boldsymbol{x})$.

Figure 6.2 shows the recovery of the partition of the input space and the submodels by the procedure above on two examples: a PWA target function (the 'ML' shape) and a PWQ function (the 'cursive $\mathcal{M}$L' shape) with 6 pieces each.

Procedure 12 Transformation to a PWS model

1. Solve (6.7) with $\{\boldsymbol{z}_k\}_{k=1}^M = \{\boldsymbol{x}_k\}_{k=1}^N$.
2. Build a new data set $\overline{D}_{\boldsymbol{x}}$ by excluding the points close to the boundaries of the regions $\Re_j$ from the training set, i.e.,

$$\overline{D}_{\boldsymbol{x}} = \{\boldsymbol{x}_k \ : \ k \in [1, N], \ \|D^n f(\boldsymbol{x}_k)\|_p < \tau\}.$$

3. Map these points to a feature space with

$$\forall \boldsymbol{x}_k \in \overline{D}_{\boldsymbol{x}}, \ \boldsymbol{x}_k \mapsto \varphi(\boldsymbol{x}_k) = [\boldsymbol{x}_k^\top, \ D^{(n-1)} f(\boldsymbol{x}_k)^\top]^\top.$$

4. Apply a clustering algorithm, e.g., k-means, in that feature space to estimate the labels q_k for all $\boldsymbol{x}_k \in \overline{D}_{\boldsymbol{x}}$.
5. Build s local submodels, f_j, from the s data subsets, $\mathcal{D}_j = \{\boldsymbol{x}_k \in \overline{D}_{\boldsymbol{x}} : q_k = j\}$, with a classical regression method suitable for the choice of n.
6. Build a classifier h on the data set $\{(\boldsymbol{x}_k, \hat{q}_k)\}_{k=1}^N$, labeled by $\hat{q}_k = \arg\min_{j=1,\dots,s} |y_k - f_j(\boldsymbol{x}_k)|$.

Table 6.1: Comparison of one-step-ahead MSE (mean and standard deviation).

Method	Clustering-based procedure in Sect. 2.3.1 (page 25)	Solving the SOCP (6.9)	PWA model created by Procedure 12
MSE	4.31 ± 4.01	1.06 ± 0.41	0.81 ± 0.43

6.3 Examples and applications

6.3.1 PWA system identification

We consider the PWA system identification problem with the PWA system studied in the Example 1 of [12],

$$y_k = \begin{cases} -0.4 y_{k-1} + u_{k-1} + 1.5 + v_k, & \text{if } 4y_{k-1} - u_{k-1} + 10 < 0, \\ 0.5 y_{k-1} - u_{k-1} - 0.5 + v_k, & \text{if } 4y_{k-1} - u_{k-1} + 10 \geq 0 \\ & \text{and } 5y_{k-1} + u_{k-1} - 6 \leq 0, \\ -0.3 y_{k-1} + 0.5 u_{k-1} - 1.7 + v_k, & \text{if } 5y_{k-1} + u_{k-1} - 6 > 0, \end{cases} \tag{6.11}$$

where $\boldsymbol{x}_k = [y_{k-1}, u_{k-1}]^\top$. In each of the following 100 experiments, we generate $N = 400$ points with this system and split them in two subsets, a training set and a validation set, of 200 points each. Another test set with $N_t = 500$ data points is used to computed the one-step-ahead mean square error, MSE (1.21). Problem (6.9) is solved on the training data while tuning of $\lambda \in \{10^{-4}, 10^{-3}, 10^{-2}, 0.05, 0.1\}$ is performed on the validation set. We use an RBF kernel with $\sigma = 0.5$. Solving (6.9) yields a smooth approximation of the PWA system, which is then used in the procedure of Sect. 6.2 to learn a PWA model with $s = 3$ (in this case, λ is tuned w.r.t. the validation error of the PWA model). We compare with the clustering-based method of [39] for PWA system identification, described in Sect. 2.3.1 (page 25), for a parameter c tuned in the range $\{6, 7, 8, 9, 10, 12, 15\}$.

Table 6.1 shows that the proposed procedures outperform the method of [39] on average. The latter also leads to a large standard deviation of the MSE due to large errors in about

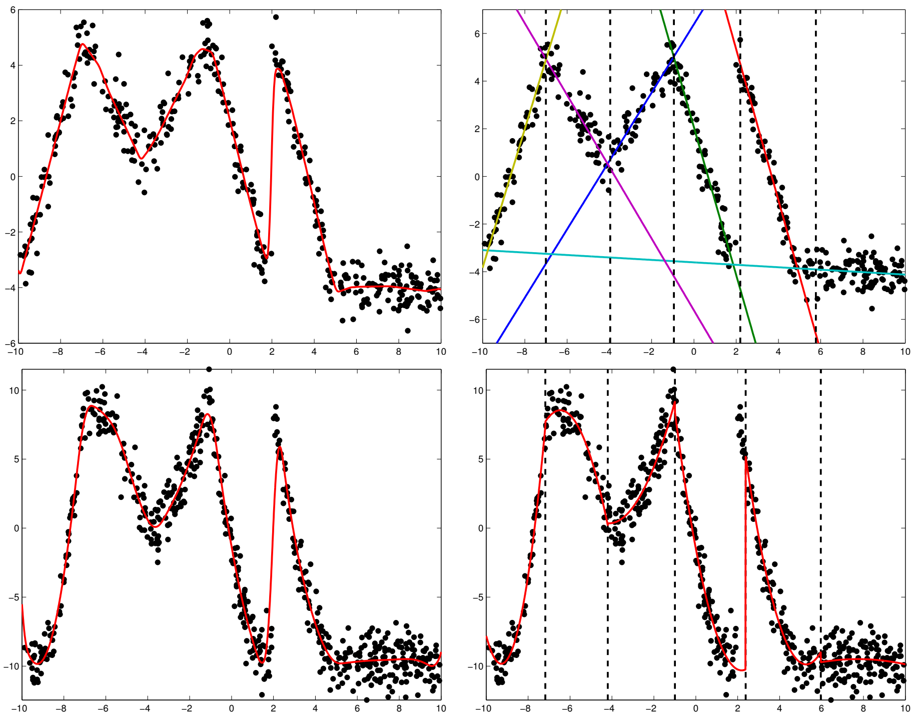


Figure 6.2: Learning a PWA (top) and a PWQ (bottom) model. Left: smooth model obtained after Step 1 of Procedure 12 with $n = 2$ (top) and $n = 3$ (bottom). Right: affine (top) and quadratic (bottom) submodels with the partitions of the input space (dash lines).

20% of the experiments.

6.3.2 Image denoising with missing pixels

Image denoising aims at recovering an original image I from a noisy image, $I_n = I + E$, while preserving edges. Image denoising with missing pixels considers the slightly more difficult task where only parts of a noisy image are available. The proposed framework particularly fits such cases, where reconstructing the entire image simply amounts to computing predictions with the trained model at all pixels, i.e., for an N_x-by-N_y image, for all $\boldsymbol{x} \in [1, N_x] \times [1, N_y]$. Figure 6.3 shows the results of such experiments for an original PWA image and Fig. 6.4 for a piecewise quadratic (PWQ) image. Note that since the proposed approach is a general regression method rather than a dedicated image processing technique, the aim is not to compare the performance with other denoising methods but rather to illustrate potential applications. Figures 6.3 and 6.4 also show the results of the procedure of Sect. 6.2 to obtain piecewise models (6.1). Note that these PWA and PWQ models are mostly applicable when the optimal target model is truly (or close to) PWA or PWQ. However, the smooth models trained by solving (6.9) or (6.10) are always applicable as shown by Fig. 6.5, where the illuminated peach image is piecewise smooth, but not piecewise quadratic.

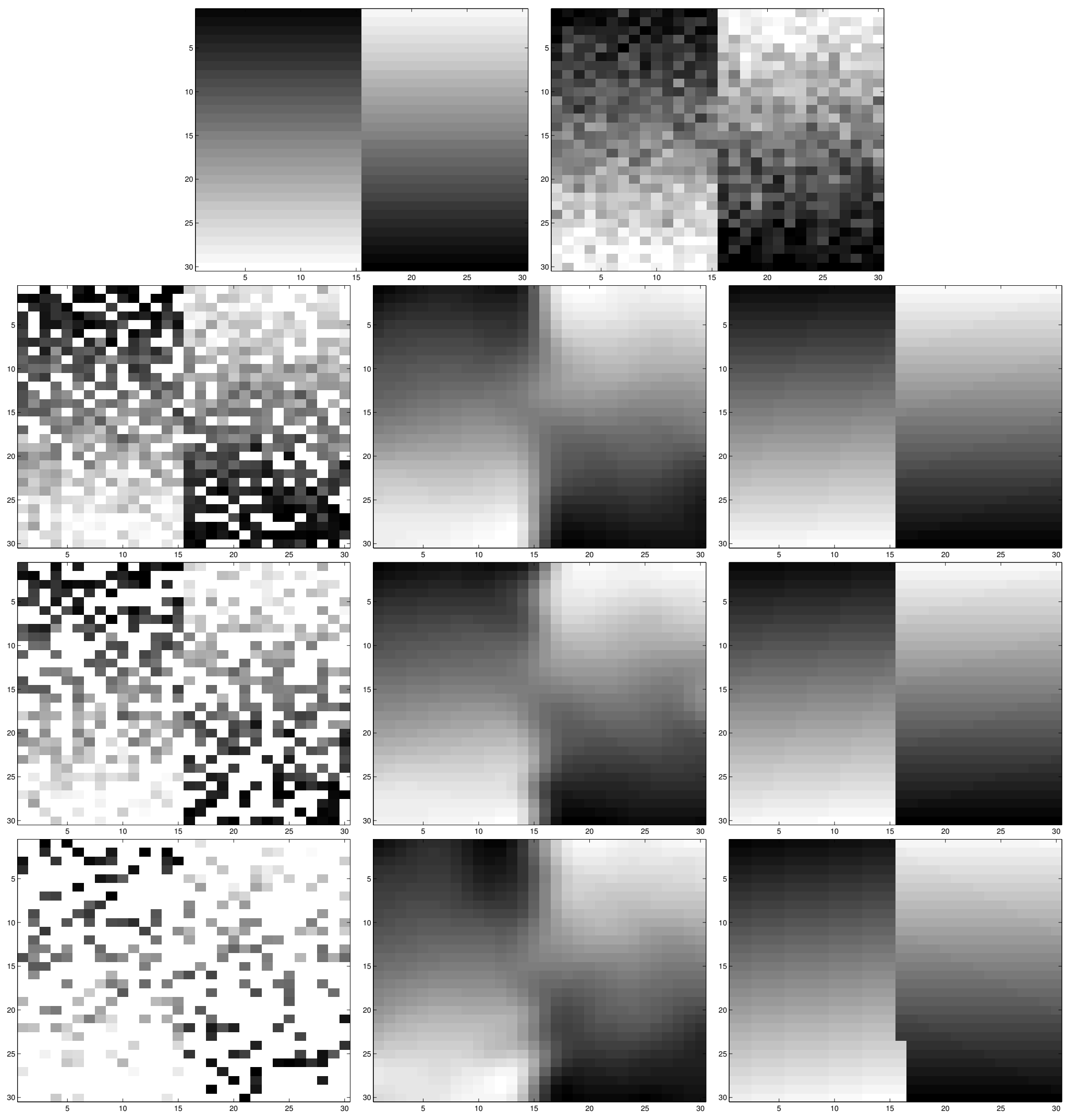


Figure 6.3: Top row: original and complete noisy images. Next rows show results obtained with an increasing percentage of missing pixels (25%, 50%, 75%): noisy image with holes (left), smooth model (6.5) solution to (6.9) obtained in Step 1 of Procedure 12 (middle) and PWA model given by Procedure 12 (right).

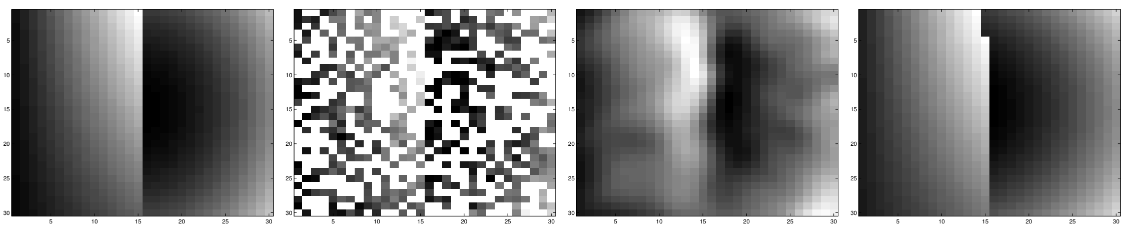


Figure 6.4: Left to right: original image, noisy image with 50% of missing pixels, smooth model (6.5), and PWQ model (6.1).

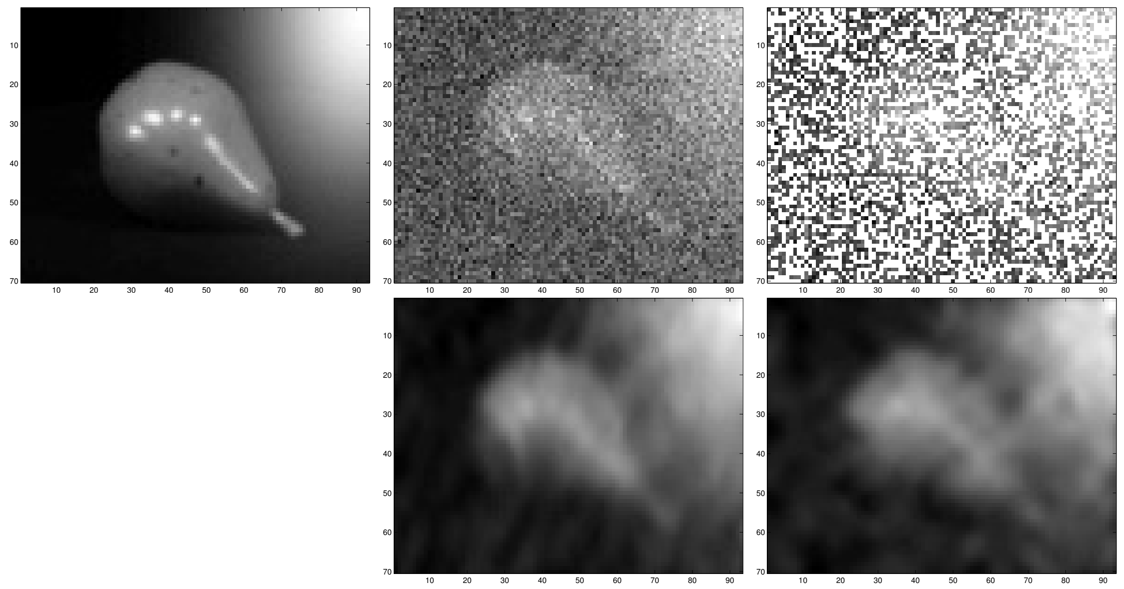


Figure 6.5: Top: original and noisy images (0 and 50% of missing pixels). Bottom: smooth models trained by (6.9) with $n = 3$.

6.4 Conclusions

In this chapter, we derived a new learning approach in the spirit of regularized nonlinear regression for PWS functions. To benefit from the flexibility of the RBF kernel model while avoiding the overfitting due to noise, a new regularization which penalizes the derivatives in a location-dependent manner is proposed. It leads to solving a convex optimization problem, thus avoiding local minima issues. We proposed also different choices of the order and the norm involved in the regularization functional to adapt to different systems. The obtained smooth model can be transformed to a PWS model with a simple procedure.

In the proposed approach, as an option, straightforward extensions to other loss functions may be considered depending on the desired properties of the model (sparsity, robustness to outliers. . .). For instance, if the ℓ_1-loss or ε-insensitive loss are used with $p = 2$, the first conic constraint in (6.9) simply becomes a linear constraint; with $p = \infty$, this yields linear programs instead of the QP (6.10).

Conclusions

In this thesis, the problems related to hybrid system identification are considered with different approaches.

Geometric approach

We proposed in Chapter 3 a new point of view for the identification of linear hybrid systems. In the parameter space, estimating the parameter vectors of submodels is equivalent to finding the intersections of the hyperplanes which are determined by the data points. A similar point of view was independently developed in [104]. However, the method of [104] based on the Hough transform estimator, is computationally intractable as it scans a number of cells in parameter space that is exponentially growing with the dimension to find the intersection of the hyperplanes.

The approach of Chapter 3 relies on the fact that the data set can be projected in the parameter space to build the hyperspheres which represent the modes. With this structure, the data classification into modes may be easier than with the original structure. Particularly, the hyperspheres of systems switching between two modes can be optimally separated in some cases and the resulting algorithm can identify hybrid systems with an arbitrary number of modes with speed, accuracy and robustness to noise.

Open issues

- As discussed in Chapter 3, for switched systems with two modes, the distribution of data points on the two hyperspheres influences the classification of the data points by PCA. Therefore, it should be more carefully studied under what conditions on the data set this classification allows parameter vectors to be well estimated.
- Although a simple iterative algorithm inspired by the bounded-error approach was proposed to deal with more than two modes, a theoretical study for this algorithm is necessary. In addition, the multiple hypersphere separation problem which cannot be processed with PCA should be investigated.

Selective ℓ_1 minimization for sparsity optimization

In Chapter 4, we improved a method for hybrid system identification, which relies on finding the sparse solutions to systems of linear equations through convex relaxations of ℓ_0-norm minimization problems. A new iterative algorithm was proposed to improve the sparsity of the solution of a generic convex relaxation based on the ℓ_1-norm. Compared with the state of the art (the iteratively reweighted scheme of [25]), the proposed algorithm benefits from the absence of hyperparameters and a finite convergence in a number of steps at most equal to the number of linear equations. In addition, a sparse recovery condition, guaranteeing the convergence towards the sparsest solution, was proved and experiments showed that the new scheme can recover the sparsest solution in more difficult cases with larger sparsity levels.

Open issues

- Future work could focus on the analysis of the sparse recovery condition and its comparison with classical results obtained for basis pursuit (equivalent to the first iteration of the proposed algorithm).
- Regarding hybrid system identification, though the method proved robust to noise in experiments, further investigations should be conducted in order to extend the theoretical results to noisy data sets.
- From a broader perspective, other applications of the generic sparsity enhancing algorithm could be considered, as suggested by the large variety of problems formulated as sparse optimization problems in various fields, see, e.g., [25].

Nonlinear hybrid system identification

Chapter 5 focused on two frameworks for nonlinear hybrid system identification: the continuous optimization framework and the error sparsification framework.

Within the first framework, in order to maintain efficiency for large data sets, some approaches to build sparse kernel submodels were proposed. These were compared in numerical experiments, which showed that the proposed approach achieves the simultaneous classification of data points and approximation of the nonlinear behaviors in an efficient and accurate manner.

Within the second framework, firstly, we relaxed the sparsity condition by introducing robust sparsity, which can be optimized through the minimization of a modified ℓ_1-norm or, equivalently, of the ϵ-insensitive loss function. Then, we showed that, depending on the choice of regularizer, the method is equivalent to different forms of support vector regression. This allows us to extend theoretical results as well as efficient optimization algorithms from the field of machine learning to the hybrid system framework.

Open issues

- In the continuous optimization framework, techniques for tuning the hyperparameters σ_j of the RBF kernel functions should be investigated since it is a crucial issue in the methods proposed to build reduced-size kernel models.
- The error sparsification algorithm with automatic tuning of the parameter ϵ of the modified ℓ_1-norm introduces a new parameter, ν, which can be interpreted as the fraction of data considered as outliers for the model. The precise relationship between this parameter and the fraction of data generated by each mode, which is also involved in the sparse recovery conditions, should be investigated. In particular, the characterization of the influence of the reweighting scheme on the choice of ν remains an open issue.

- An alternative direction of research for the error sparsification method concerns the computation of the full solution paths w.r.t. the regularization constant λ and the hyperparameters ε or ν. Here, the aim is to obtain the models for all possible values of the hyperparameters at a low computational cost. In this respect, we should once again take advantage of the equivalence with support vector regression and the large collection of results on this topic.

Piecewise smooth system identification

In Chapter 6, we derived a new learning approach in the spirit of regularized kernel regression for piecewise smooth (PWS) functions. To profit from the flexibility of the RBF kernel model and to avoid the overfitting due to noise, a new regularization term which penalizes the derivatives in a location-dependent manner was proposed. It leads to solving a convex optimization problem, thus avoiding local minima issues. We proposed also different choices of the derivative order and the norm involved in the local regularization functional to adapt to different systems. The obtained smooth model can be transformed to a PWS model with a simple procedure, which proved its efficiency in experiments.

Open issues

- To solve the learning problem with generic solvers, the proposed convex, but nonsmooth, optimization problems with the new regularization term was reformulated as two constrained programs: the Second-Order Cone Program (SOCP) (6.9) and the Quadratic Program (QP) (6.10). However, directly solving them can become prohibitive for very large data sets due to a very large number of constraints. Future work could consider faster algorithms for a smoothed version of the original unconstrained optimization.

- Another research direction with practical consequences concerns the derivation of the full solution path w.r.t. the regularization constant λ.

Appendix A

Principal component analysis

The main idea of principal component analysis (PCA) [50] is to reduce the dimensionality of a data set consisting of a large number of interrelated variables, while retaining as much as possible the variance of the data set. This is achieved by transforming to a new set of variables, the so-called principal components (PCs), which are uncorrelated and arranged in decreasing order of important level. A PC is more important than another if the variance of data according to this PC is larger than according to another. Finally, d PCs are used to represent data in lower dimension space.

PCA concentrates on variances of variables. The first direction described by a unit vector $\boldsymbol{q}_1 \in \mathbb{R}^D$ is determined such that the projection of $\boldsymbol{x}$ on this vector, i.e., $\boldsymbol{q}_1^\top \boldsymbol{x}$, has a maximum variance,

$$\begin{aligned} \boldsymbol{q}_1 = &\arg\max_{\boldsymbol{q}} \operatorname{var}[\boldsymbol{q}^\top \boldsymbol{x}] \\ &\text{s.t.} \quad \|\boldsymbol{q}\|_2 = 1, \end{aligned} \tag{A.1}$$

where

$$\operatorname{var}[\boldsymbol{q}^\top \boldsymbol{x}] = \boldsymbol{q}^\top \boldsymbol{\Sigma} \boldsymbol{q}, \tag{A.2}$$

and $\boldsymbol{\Sigma} = \mathbb{E}\left[(\boldsymbol{x} - \mathbb{E}[\boldsymbol{x}])(\boldsymbol{x} - \mathbb{E}[\boldsymbol{x}])^\top\right]$ is the covariance matrix of the vector variable $\boldsymbol{x}$ whose (i,j)th element is the covariance between the ith and jth entries of $\boldsymbol{x}$.

Next, the second direction $\boldsymbol{q}_2$ is found such that $\boldsymbol{q}_2^\top \boldsymbol{x}$ is uncorrelated with $\boldsymbol{q}_1^\top \boldsymbol{x}$ and has a possibly maximum variance, and so on for l directions with $\boldsymbol{q}_l$. The variables defined by $\boldsymbol{q}_l^\top \boldsymbol{x}$ are the PCs. If the variance of one of the variables is zero (or too small), that PC is unnecessary to represent the data set. Then, the data set can be expressed with a lower dimension.

In practice, the covariance matrix $\boldsymbol{\Sigma}$ is unknown and is estimated by the sample covariance matrix $\operatorname{cov}(\boldsymbol{X})$. For centered data, $\operatorname{cov}(\boldsymbol{X}) = \frac{1}{N}\sum_{k=1}^N \boldsymbol{x}_k \boldsymbol{x}_k^\top$ since $\frac{1}{N}\sum_{k=1}^N \boldsymbol{x}_k = 0$.

The Lagrangian form of Problem (A.1) is

$$\max_{\boldsymbol{q}} \ \boldsymbol{q}^\top \boldsymbol{\Sigma} \boldsymbol{q} - \lambda(\boldsymbol{q}^\top \boldsymbol{q} - 1), \tag{A.3}$$

where λ is a Lagrange multiplier. Differentiation of the objective function of (A.3) with respect to $\boldsymbol{q}$ gives

$$2(\boldsymbol{\Sigma}\boldsymbol{q} - \lambda \boldsymbol{q}) = \boldsymbol{0}, \tag{A.4}$$

or

$$(\boldsymbol{\Sigma} - \lambda \boldsymbol{I}_D)\boldsymbol{q} = \boldsymbol{0}. \tag{A.5}$$

Thus, λ is an eigenvalue of $\boldsymbol{\Sigma}$ and $\boldsymbol{q}$ is the corresponding eigenvector. Moreover,

$$\boldsymbol{q}^\top \boldsymbol{\Sigma} \boldsymbol{q} = \boldsymbol{q}^\top \lambda \boldsymbol{q} = \lambda \boldsymbol{q}^\top \boldsymbol{q} = \lambda. \tag{A.6}$$

Then, introducing (A.5) and (A.6) in (A.3) leads to

$$\begin{aligned} &\max_{\lambda,\boldsymbol{q}} \ \lambda \\ &\text{s.t.} \quad (\boldsymbol{\Sigma} - \lambda \boldsymbol{I}_D)\boldsymbol{q} = \boldsymbol{0}. \end{aligned} \tag{A.7}$$

Therefore, the directions $\boldsymbol{q}_1, \boldsymbol{q}_2, \dots$ are the eigenvectors corresponding to the largest eigenvalues $\lambda_1 \geq \lambda_2 \geq \dots$ of $\boldsymbol{\Sigma}$.

To construct a d-dimensional representation of the data set, PCA chooses the d first eigenvectors to project the data with

$$\boldsymbol{Z} = \boldsymbol{Q}^\top \boldsymbol{X}, \tag{A.8}$$

where $\boldsymbol{Q} = [\boldsymbol{q}_1, \dots, \boldsymbol{q}_d]$.

Appendix B

Derivatives of a Gaussian RBF kernel expansion

Consider the kernel expansion

$$f(\boldsymbol{z}_k) = \sum_{i=1}^{N} \alpha_i \kappa(\boldsymbol{x}_i, \boldsymbol{z}_k) = \boldsymbol{\alpha}^\top \boldsymbol{\kappa}(\boldsymbol{z}_k),$$

with the RBF kernel $\kappa(\boldsymbol{x}_i, \boldsymbol{z}_k) = \exp\left(-\|\boldsymbol{x}_i - \boldsymbol{z}_k\|_2^2 / 2\sigma^2\right)$ and $\boldsymbol{\kappa}(\boldsymbol{z}_k) = [\kappa(\boldsymbol{x}_1, \boldsymbol{z}_k), \ldots, \kappa(\boldsymbol{x}_N, \boldsymbol{z}_k)]^\top$. The derivative of f w.r.t. z_{kj}, the jth component of $\boldsymbol{z}_k$ is

$$\begin{aligned}
\frac{\partial f(\boldsymbol{z}_k)}{\partial z_{kj}} &= \frac{1}{\sigma^2} \sum_{i=1}^{N} \alpha_i \kappa(\boldsymbol{x}_i, \boldsymbol{z}_k)(x_{ij} - z_{kj}) \\
&= \frac{1}{\sigma^2} \left((\boldsymbol{X}^\top)_j - \mathbf{1}_N z_{kj} \right)^\top \operatorname{diag}\left(\boldsymbol{\kappa}(\boldsymbol{z}_k)\right) \boldsymbol{\alpha},
\end{aligned}$$

where $(\boldsymbol{X}^\top)_j$ represents the jth column of $\boldsymbol{X}^\top$. Thus

$$\nabla f(\boldsymbol{z}_k) = \begin{bmatrix} \frac{\partial f(\boldsymbol{z}_k)}{\partial z_{k1}} \\ \vdots \\ \frac{\partial f(\boldsymbol{z}_k)}{\partial z_{kd}} \end{bmatrix} = \frac{1}{\sigma^2} \left(\boldsymbol{X} - \boldsymbol{z}_k \mathbf{1}_N^\top \right) \operatorname{diag}\left(\boldsymbol{\kappa}(\boldsymbol{z}_k)\right) \boldsymbol{\alpha}.$$

The elements of the Hessian matrix at point $\boldsymbol{z}_k$ are given by

$$\begin{aligned}
(\boldsymbol{H}_k)_{jl} &= \frac{\partial^2 f(\boldsymbol{z}_k)}{\partial z_{kj} \partial z_{kl}} \\
&= \frac{\partial}{\partial z_{kl}} \left(\frac{1}{\sigma^2} \sum_{i=1}^{N} \alpha_i \kappa(\boldsymbol{x}_i, \boldsymbol{z}_k)(x_{ij} - z_{kj}) \right) \\
&= \begin{cases} \frac{1}{\sigma^4} \sum_{i=1}^{N} \alpha_i \kappa(\boldsymbol{x}_i, \boldsymbol{z}_k)(x_{ij} - z_{kj})(x_{il} - z_{kl}), & \text{if } l \neq j, \\ \frac{1}{\sigma^4} \sum_{i=1}^{N} \alpha_i \kappa(\boldsymbol{x}_i, \boldsymbol{z}_k) \left((x_{ij} - z_{kj})^2 - \sigma^2 \right), & \text{if } l = j, \end{cases} \\
&= \frac{1}{\sigma^4} \sum_{i=1}^{N} \alpha_i \kappa(\boldsymbol{x}_i, \boldsymbol{z}_k) \left((x_{ij} - z_{kj})(x_{il} - z_{kl}) - \delta_{j,l} \sigma^2 \right) \\
&= \frac{1}{\sigma^4} \left[((\boldsymbol{X}^\top)_j - z_{kj} \mathbf{1}_N) \odot ((\boldsymbol{X}^\top)_l - z_{kl} \mathbf{1}_N) - \delta_{j,l} \sigma^2 \mathbf{1}_N \right]^\top \operatorname{diag}\left(\boldsymbol{\kappa}(\boldsymbol{z}_k)\right) \boldsymbol{\alpha}.
\end{aligned}$$

The third order derivatives, $\partial^3 f(\boldsymbol{z}_k)/\partial z_{kj}\partial z_{kl}\partial z_{km}$, are given by

$$\frac{\partial^3 f(\boldsymbol{z}_k)}{\partial z_{kj}\partial z_{kl}\partial z_{km}} = \frac{1}{\sigma^6}\sum_{i=1}^{N}\alpha_i\kappa(\boldsymbol{x}_i,\boldsymbol{z}_k)\Big[(x_{ij}-z_{kj})(x_{il}-z_{kl})(x_{im}-z_{km}) - \delta_{j,l}\sigma^2(x_{im}-z_{km}) - \delta_{j,m}\sigma^2(x_{il}-z_{kl}) - \delta_{l,m}\sigma^2(x_{ij}-z_{kj})\Big].$$

In the matrix form, this becomes

$$\frac{\partial^3 f(\boldsymbol{z}_k)}{\partial z_{kj}\partial z_{kl}\partial z_{km}} = \frac{1}{\sigma^6}\Bigg[\Big((\boldsymbol{X}^\top)_j - z_{kj}\mathbf{1}_N\Big)\odot\Big((\boldsymbol{X}^\top)_l - z_{kl}\mathbf{1}_N\Big)\odot\Big((\boldsymbol{X}^\top)_m - z_{km}\mathbf{1}_N\Big) - \delta_{l,m}\sigma^2\Big((\boldsymbol{X}^\top)_j - z_{kj}\mathbf{1}_N\Big) - \delta_{j,m}\sigma^2\Big((\boldsymbol{X}^\top)_l - z_{kl}\mathbf{1}_N\Big) - \delta_{j,l}\sigma^2\Big((\boldsymbol{X}^\top)_m - z_{km}\mathbf{1}_N\Big)\Bigg]^\top \operatorname{diag}\left(\boldsymbol{\kappa}(\boldsymbol{z}_k)\right)\boldsymbol{\alpha}.$$

Bibliography

[1] E. Amaldi and M. Mattavelli. The MIN PFS problem and piecewise linear model estimation. *Discrete Applied Mathematics*, 118(1-2):115–143, 2002.

[2] N. Aronszajn. Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 68(3):337–404, 1950.

[3] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188, 2002.

[4] F. Aurenhammer. Voronoi diagrams–a survey of a fundamental geometric data structure. *ACM Computing Surveys (CSUR)*, 23(3):345–405, 1991.

[5] L. Bako. Identification of switched linear systems via sparse optimization. *Automatica*, 47(4):668–677, 2011.

[6] L. Bako, K. Boukharouba, E. Duviella, and S. Lecoeuche. A recursive identification algorithm for switched linear/affine models. *Nonlinear Analysis: Hybrid Systems*, 5(2):242–253, 2011.

[7] L. Bako, K. Boukharouba, and S. Lecoeuche. An ℓ_0-ℓ_1 norm based optimization procedure for the identification of switched nonlinear systems. In *Proceedings of the 49th IEEE International Conference on Decision and Control*, pages 4467–4472, 2010.

[8] L. Bako, V. L. Le, F. Lauer, and G. Bloch. Identification of mimo switched state-space models. In *Proceedings of American Control Conference*, 2013.

[9] L. Bako, G. Mercère, and S. Lecoeuche. On-line structured subspace identification with application to switched linear systems. *International Journal of Control*, 82(8):1496–1515, 2009.

[10] L. Bako and R. Vidal. Algebraic identification of MIMO SARX models. *Hybrid Systems: Computation and Control*, pages 43–57, 2008.

[11] G. Baudat and F. Anouar. Feature vector selection and projection using kernels. *Neurocomputing*, 55(1-2):21–38, 2003.

[12] A. Bemporad, A. Garulli, S. Paoletti, and A. Vicino. A bounded-error approach to piecewise affine system identification. *IEEE Transactions on Automatic Control*, 50(10):1567–1580, 2005.

[13] A. Bemporad and M. Moran. Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35(3):407–428, 1999.

[14] K. P. Bennett and O. L. Mangasarian. Robust linear programming discrimination of two linearly inseparable sets. *Optimization Methods and Software*, 1(1):23–34, 1992.

[15] K. P. Bennett and O. L. Mangasarian. Multicategory discrimination via linear programming. *Optimization Methods and Software*, 3(1-3):27–39, 1994.

[16] J. Borges, V. Verdult, and M. Verhaegen. Iterative subspace identification of piecewise linear systems. In *Proceedings of the 14th IFAC Symp. on System Identification*, pages 368–373, 2006.

[17] L. Bottou, O. Chapelle, D. DeCoste, and J. Weston. *Large scale kernel machines.* MIT Press, Cambridge, MA, 2007.

[18] K. Boukharouba, L. Bako, and S. Lecoeuche. Identification of piecewise affine systems based on dempster-shafer theory. In *Proceeding of 15th IFAC Symposium on System Identification, 2009*, volume 15, pages 1662–1667, 2009.

[19] S. Boyd and L. Vandenberghe. *Convex optimization.* Cambridge University Press, 2004.

[20] M. S. Branicky. Introduction to hybrid systems. *Handbook of Networked and Embedded Control Systems*, pages 91–116, 2005.

[21] L. Breiman. Hinging hyperplanes for regression, classification, and function approximation. *IEEE Transactions on Information Theory*, 39(3):999–1013, 1993.

[22] Chang C. and Lin C. LibSVM: a library for support vector machines, 2001. http://www.csie.ntu.edu.tw/∼cjlin/libsvm/.

[23] E. J. Candès. Compressive sampling. In *Proceedings of the International Congress of Mathematicians: Madrid, August 22-30, 2006: invited lectures*, pages 1433–1452, 2006.

[24] E. J. Candès and T. Tao. Decoding by linear programming. *IEEE Transactions on Information Theory*, 51(12):4203–4215, 2005.

[25] E. J. Candès, M. B. Wakin, and S. P. Boyd. Enhancing sparsity by reweighted ℓ_1 minimization. *Journal of Fourier Analysis and Applications*, 14(5):877–905, 2008.

[26] G. C. Cawley and N. L. C. Talbot. Efficient formation of a basis in a kernel induced feature space. In *Proceedings of the European Symposium on Artificial Neural Networks*, pages 1–6, 2002.

[27] G. C. Cawley and N. L. C. Talbot. Reduced rank kernel ridge regression. *Neural Processing Letters*, 16(3):293–302, 2002.

[28] A. Cochocki and R. Unbehauen. *Neural networks for optimization and signal processing.* John Wiley & Sons, 1993.

[29] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[30] M. A. Davenport, M. F. Duarte, Y. C. Eldar, and G. Kutyniok. Introduction to compressed sensing. *Compressed Sensing: Theory and Applications*, 2012.

[31] B. De Schutter and B. De Moor. The extended linear complementarity problem and the modeling and analysis of hybrid systems. *Hybrid systems V*, 1567:635–636, 1999.

[32] B. De Schutter and T. Van den Boom. On model predictive control for max-min-plus-scaling discrete event systems. *Automatica*, 37(7):1049–1056, 2001.

[33] D. L. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52(4):1289–1306, 2006.

[34] D. L. Donoho and M. Elad. Optimally sparse representation in general (nonorthogonal) dictionaries via ℓ_1 minimization. *Proceedings of the National Academy of Sciences*, 100(5):2197–2202, 2003.

[35] D. L. Donoho and X. Huo. Uncertainty principles and ideal atomic decomposition. *IEEE Transactions on Information Theory*, 47(7):2845–2862, 1999.

[36] S. Ernst. Hinging hyperplane trees for approximation and identification. In *Proceedings of the 37th IEEE Conference on Decision and Control*, volume 2, pages 1266–1271, 1998.

[37] T. Evgeniou, M. Pontil, and T. Poggio. Regularization networks and support vector machines. *Advances in Computational Mathematics*, 13(1):1–50, 2000.

[38] T. Falck. *Nonlinear System Identification using Structured Kernel Based Models.* PhD thesis, KU Leuven, 2013.

[39] G. Ferrari-Trecate, M. Muselli, D. Liberati, and M. Morari. A clustering technique for the identification of piecewise affine systems. *Automatica*, 39(2):205–217, 2003.

[40] S. Fine and K. Scheinberg. Efficient svm training using low-rank kernel representations. *The Journal of Machine Learning Research*, 2:243–264, 2002.

[41] M. Fornasier and H. Rauhut. Compressive sensing. In O. Scherzer, editor, *Handbook of Mathematical Methods in Imaging*, pages 187–229. Springer, 2011.

[42] A. Garulli, S. Paoletti, and A. Vicino. A survey on switched and piecewise affine system identification. In *Proceedings of the 16th IFAC Symposium on System Identification*, pages 344–355, 2012.

[43] A. Gionis and H. Mannila. Segmentation algorithms for time series and sequence data. In *The SIAM International Conference on Data Mining: A Tutorial*, 2005.

[44] R. Gribonval and M. Nielsen. Sparse representations in unions of bases. *IEEE Transactions on Information Theory*, 49(12):3320–3325, 2003.

[45] Y. Hashambhoy and R. Vidal. Recursive identification of switched ARX models with unknown number of models and unknown orders. In *Proceedings of the 44th IEEE Conference on Decision and Control and European Control Conference*, pages 6115–6121, 2006.

[46] W. Heemels, J. M. Schumacher, and S. Weiland. Linear complementarity systems. *SIAM Journal on Applied Mathematics*, 60(4):1234–1269, 2000.

[47] W. P. M. H. Heemels, B. De Schutter, and A. Bemporad. Equivalence of hybrid dynamical models. *Automatica*, 37(7):1085–1091, 2001.

[48] K. Huang, A. Wagner, and Y. Ma. Identification of hybrid linear time-invariant systems via subspace embedding and segmentation. In *Proceedings of the 43rd IEEE Conference on Decision and Control*, volume 3, pages 3227–3234, 2005.

[49] W. Huyer and A. Neumaier. Global optimization by multilevel coordinate search. *Journal of Global Optimization*, 14(4):331–355, 1999.

[50] I. Jolliffe. *Principal component analysis.* Wiley Online Library, 2005.

[51] A. Juloski, W. P. M. H. Heemels, G. Ferrari-Trecate, R. Vidal, S. Paoletti, and J. Niessen. Comparison of four procedures for the identification of hybrid systems. *Hybrid Systems: Computation and Control*, pages 354–369, 2005.

[52] A. L. Juloski, S. Weiland, and W. Heemels. A bayesian approach to identification of hybrid systems. *IEEE Transactions on Automatic Control*, 50(10):1520–1533, 2005.

[53] Suykens J. A. K., Van Gestel T., De Brabanter J., De Moor B., and Vandewalle J. *Least Squares Support Vector Machines.* World Scientific, River Edge, NJ, USA, 2002.

[54] A. Khajehnejad, W. Xu, A. Avestimehr, and B. Hassibi. Weighted ℓ_1 minimization for sparse recovery with prior information. In *IEEE International Symposium on Information Theory*, pages 483–487, 2009.

[55] J. T. Kwok and I. W. Tsang. Linear dependency between ε and the input noise in ε-support vector regression. *IEEE Transactions on Neural Networks*, 14(3):544–553, 2003.

[56] C. Y. Lai, C. Xiang, and T. H. Lee. Identification and control of nonlinear systems via piecewise affine approximation. In *Proceedings of the 49th IEEE Conference on Decision and Control*, pages 6395–6402, 2010.

[57] F. Lauer. Estimating the probability of success of a simple algorithm for switched linear regression. *Nonlinear Analysis: Hybrid Systems*, 8:31–47, 2013.

[58] F. Lauer and G. Bloch. Switched and piecewise nonlinear hybrid system identification. In *Proceedings of the 11th International Conference on Hybrid Systems: Computation and Control*, volume 4981, pages 330–343, 2008.

[59] F. Lauer, G. Bloch, and R. Vidal. Nonlinear hybrid system identification with kernel models. In *Proceedings of the 49th IEEE International Conference on Decision and Control*, pages 696–701, 2010.

[60] F. Lauer, G. Bloch, and R. Vidal. A continuous optimization framework for hybrid system identification. *Automatica*, 47(3):608–613, 2011.

[61] L. Ljung. *System identification: theory for the user.* Prentice-Hall, Inc., Upper Saddle River, NJ, 1999.

[62] J. Lunze and F. Lamnabhi-Lagarrigue. *Handbook of hybrid systems control: theory, tools, applications.* Cambridge University Press, New York, 2009.

[63] Y. Ma and R. Vidal. Identification of deterministic switched ARX systems via identification of algebraic varieties. In *Proceedings of the 8th International Conference on Hybrid Systems: Computation and Control*, volume 3414, pages 449–465, 2005.

[64] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, 1967.

[65] O. L. Mangasarian and D. R. Musicant. Large scale kernel regression via linear programming. *Machine Learning*, 46(1):255–269, 2002.

[66] I. Maruta, T. Sugie, and T.H. Kim. Identification of multiple mode models via distributed particle swarm optimization. In *Proceedings of the 18th IFAC World Congress*, volume 18, pages 7743–7748, 2011.

[67] C. Novara. Sparse identification of nonlinear functions and parametric set membership optimality analysis. *IEEE Transactions on Automatic Control*, 57(12):3236–3241, 2012.

[68] H. Ohlsson and L. Ljung. Piecewise affine system identification using sum-of-norms regularization. In *Proceedings of the 18th IFAC World Congress, Milano, Italy*, pages 6640–6645, 2011.

[69] H. Ohlsson and L. Ljung. Identification of switched linear regression models using sum-of-norms regularization. *Automatica*, 49(4), 2013.

[70] H. Ohlsson, L. Ljung, and S. Boyd. Segmentation of ARX-models using sum-of-norms regularization. *Automatica*, 46(6):1107–1111, 2010.

[71] N. Ozay, C. Lagoa, and M. Sznaier. Robust identification of switched affine systems via moments-based convex optimization. In *Proceedings of the 48th IEEE Conference on Decision and Control, held jointly with the 28th Chinese Control Conference*, pages 4686–4691, 2009.

[72] N. Ozay, M. Sznaier, C. Lagoa, and O. Camps. A sparsification approach to set membership identification of switched affine systems. *IEEE Transactions on Automatic Control*, 57(3):634–648, 2012.

[73] S. Paoletti, A.L. Juloski, G. Ferrari-Trecate, and R. Vidal. Identification of hybrid systems: A tutorial. *European Journal of Control*, 13(2-3):242–260, 2007.

[74] K. M. Pekpe, G. Mourot, K. Gasso, and J. Ragot. Identification of switching systems using change detection technique in the subspace framework. In *Proceedings of the 43rd IEEE Conference on Decision and Control*, volume 4, pages 3720–3725, 2004.

[75] J. C. Platt. Fast training of support vector machines using sequential minimal optimization. In Schölkopf B., Burges C. J. C., and Smola A. J., editors, *Advances in Kernel Methods: Support Vector Learning*, pages 185–208. MIT Press, 1999.

[76] P. Pucar and J. Sjöberg. On the hinge-finding algorithm for hingeing hyperplanes. *IEEE Transactions on Information Theory*, 44(3):1310–1319, 1998.

[77] J. Roll, A. Bemporad, and L. Ljung. Identification of piecewise affine systems via mixed-integer programming. *Automatica*, 40(1):37–50, 2004.

[78] R. Rosipal, M. Girolami, L. J. Trejo, and A. Cichocki. Kernel pca for feature extraction and de-noising in nonlinear regression. *Neural Computing & Applications*, 10(3):231–243, 2001.

[79] B. Schölkopf, R. Herbrich, and A. Smola. A generalized representer theorem. In *Computational Learning Theory*, pages 416–426, 2001.

[80] B. Schölkopf, A. Smola, and K. R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5):1299–1319, 1998.

[81] B. Schölkopf, A. J. Smola, R. C. Williamson, and P. L. Bartlett. New support vector algorithms. *Neural Computation*, 12(5):1207–1245, 2000.

[82] S. K. Shevade, S. S. Keerthi, C. Bhattacharyya, and K. R. K. Murthy. Improvements to the SMO algorithm for SVM regression. *IEEE Transactions on Neural Networks*, 11(5):1188–1193, 2000.

[83] J. Sjöberg, Q. Zhang, L. Ljung, A. Benveniste, B. Delyon, P. Y. Glorennec, H. Hjalmarsson, and A. Juditsky. Nonlinear black-box modeling in system identification: a unified overview. *Automatica*, 31(12):1691–1724, 1995.

[84] A. Smola, B. Schölkopf, and G. Rätsch. Linear programs for automatic accuracy control in regression. In *Proceedings of the 9th International Conference on Artificial Neural Networks*, pages 575–580, 1999.

[85] A. J. Smola and B. Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, 14(3):199–222, 2004.

[86] T. Söderström and P. Stoica. *System identification.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988.

[87] E. Sontag. Nonlinear regulation: The piecewise linear approach. *IEEE Transactions on Automatic Control*, 26(2):346–358, 1981.

[88] Joachims T. SVM^{light}, 1998. Available at http://svmlight.joachims.org/.

[89] M. Tabatabaei-Pour, M. Gholami, K. Salahshoor, and H. R. Shaker. A clustering-based bounded-error approach for identification of PWA hybrid systems. In *Proceedings of the 9th International Conference on Control, Automation, Robotics and Vision*, pages 1–6, 2006.

[90] M. Tabatabaei-Pour, K. Salahshoor, and B. Moshiri. A modified k-plane clustering algorithm for identification of hybrid systems. In *Proceedings of the 6th World Congress on Intelligent Control and Automation*, pages 1333–1337, 2006.

[91] F. E. H. Tay and L. J. Cao. Modified support vector machines in financial time series forecasting. *Neurocomputing*, 48(1):847–861, 2002.

[92] A. R. Teixeira, A. M. Tomé, and E. W. Lang. Unsupervised feature extraction via kernel subspace techniques. *Neurocomputing*, 74(5):820–830, 2011.

[93] J. Tropp and A. C. Gilbert. Signal recovery from partial information via orthogonal matching pursuit. *IEEE Transactions on Information Theory*, 53(12):4655–4666, 2007.

[94] A. J. Van der Schaft and J. M. Schumacher. Complementarity modeling of hybrid systems. *IEEE Transactions on Automatic Control*, 43(4):483–490, 1998.

[95] V. Vapnik. *The nature of statistical learning theory.* Springer, 1995.

[96] V. N. Vapnik. *The Nature of Statistical Learning Theory.* Springer-Verlag, 1995.

[97] V. Verdult and M. Verhaegen. Subspace identification of piecewise linear systems. In *Proceedings of the 43rd IEEE Conference on Decision and Control*, pages 3838–3843, 2005.

[98] R. Vidal. Identification of PWARX hybrid models with unknown and possibly different orders. In *Proceedings of the American Control Conference*, pages 547–552, 2004.

[99] R. Vidal. Recursive identification of switched ARX systems. *Automatica*, 44(9):2274–2287, 2008.

[100] R. Vidal. Subspace clustering. *IEEE Signal Processing Magazine*, 28(2):52–68, 2011.

[101] R. Vidal and B. D. O. Anderson. Recursive identification of switched ARX hybrid models: Exponential convergence and persistence of excitation. In *Proceedings of the 43rd IEEE Conference on Decision and Control*, pages 32–37, 2005.

[102] R. Vidal, Y. Ma, and S. Sastry. Generalized principal component analysis (GPCA). *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(12):1945–1959, 2005.

[103] R. Vidal, S. Soatto, Y. Ma, and S. Sastry. An algebraic geometric approach to the identification of a class of linear hybrid systems. In *Proceedings of the 42nd IEEE Conference on Decision and Control*, pages 167–172, 2003.

[104] J. Wang. *Identification of Switched Linear Systems.* PhD thesis, University of Alberta, 2013.

[105] J. Xu, X. Huang, and S. Wang. Adaptive hinging hyperplanes and its applications in dynamic system identification. *Automatica*, 45(10):2325–2332, 2009.

[106] Y. Zhao and D. Li. Reweighted ℓ_1-minimization for sparse solutions to underdetermined linear systems. *SIAM Journal on Optimization*, 22(3):1065–1088, 2012.

Résumé

En automatique, l'obtention d'un modèle du système est la pierre angulaire des procédures comme la synthèse d'une commande, la détection des défaillances, la prédiction... Cette thèse traite de l'identification d'une classe de systèmes complexes, les systèmes dynamiques hybrides. Ces systèmes impliquent l'interaction de comportements continus et discrets. Le but est de construire un modèle à partir de mesures expérimentales d'entrée et de sortie. Une nouvelle approche pour l'identification de systèmes hybrides linéaires basée sur les propriétés géométriques des systèmes hybrides dans l'espace des paramètres est proposée. Un nouvel algorithme est ensuite proposé pour le calcul de la solution la plus parcimonieuse (ou creuse) de systèmes d'équations linéaires sous-déterminés. Celui-ci permet d'améliorer une approche d'identification basée sur l'optimisation de la parcimonie du vecteur d'erreur. De plus, de nouvelles approches, basées sur des modèles à noyaux, sont proposées pour l'identification de systèmes hybrides non linéaires et de systèmes lisses par morceaux.

Mots clés : Systèmes hybrides, systèmes à commutation, systèmes lisses par morceaux, identification, régression, parcimonie, méthodes à noyaux.

Abstract

In automatic control, obtaining a model is always the cornerstone of the synthesis procedures such as controller design, fault detection or prediction... This thesis deals with the identification of a class of complex systems, hybrid dynamical systems. These systems involve the interaction of continuous and discrete behaviors. The goal is to build a model from experimental measurements of the system inputs and outputs. A new approach for the identification of linear hybrid systems based on the geometric properties of hybrid systems in the parameter space is proposed. A new algorithm is then proposed to recover the sparsest solutions of underdetermined systems of linear equations. This allows us to improve an identification approach based on the error sparsification. In addition, new approaches based on kernel models are proposed for the identification of nonlinear hybrid systems and piecewise smooth systems.

Key words: Hybrid systems, switched systems, piecewise smooth systems, identification, regression, sparsity, kernel methods.