CONTSID : un outil logiciel pour l'identification de modèles paramétriques à temps continu à partir de données expérimentales

Hugues GARNIER, Marion GILSON, Thierry BASTOGNE, Hamza ZBALI
Centre de Recherche en Automatique de Nancy, Nancy-Université
CNRS, BP 239, F-54506 Vandoeuvre-lès-Nancy Cedex, France

hugues.garnier@cran.uhp-nancy.fr, marion.gilson@cran.uhp-nancy.fr,
thierry.bastogne@cran.uhp-nancy.fr

Résumé—*L'objectif principal de cet article est de décrire la boîte à outils Matlab : CONtinuous-Time System IDentification (CONTSID) ; elle permet l'identification directe de modèles à temps continu représentés sous forme de fonction de transfert ou de modèle d'état, à partir de données échantillonnées régulièrement ou non. Cet article présente également l'interface utilisateur facilitant l'analyse et le pré-traitement des données, l'estimation des paramètres ainsi que la validation du modèle obtenu. Par ailleurs, les avantages de l'identification directe de modèles à temps continu à partir de données échantillonnées sont également résumés.*

Mots-clés— *modèle à temps continu, données échantillonnées, interface utilisateur, boîte à outils Matlab, outil logiciel, identification des systèmes.*

I. INTRODUCTION

L'identification de modèles à temps continu (TC) est un large problème ayant des applications dans de nombreuses disciplines. Les premiers travaux d'identification des systèmes traitaient uniquement d'identification de modèles à temps continu à partir de données recueillies à temps continu. Ensuite, les développements rapides du tout-numérique pour l'acquisition des données ont contribué à l'émergence puis à la domination complète des méthodes d'identification de modèles à temps discret, à partir de données échantillonnées. Ainsi, très peu de travaux ont été développés dans le cadre de l'identification de modèles à temps continu et de ce fait, ces méthodes sont, encore aujourd'hui, assez mal connues.

Alors que de nombreux logiciels rassemblant les différentes techniques d'identification de modèles à temps discret sont disponibles, aucun outil n'existait dans le cas de l'identification de modèles à temps continu. C'est à partir de ce constat que nous avons décidé de réaliser une boîte à outils Matlab appelée CONTSID (pour *CONtinuous-Time System IDentification*) [1]. Plusieurs mises à jour ont été développées depuis sa création et présentées aux congrès récents [2], [3], [4]. L'objectif est de mettre à disposition des utilisateurs potentiels le plus grand nombre de méthodes permettant d'identifier des systèmes linéaires (SISO, MISO et MIMO) représentés sous la forme de modèles à temps continu, directement à partir de données échantillonnées. Cette boîte à outils Matlab rassemble la plupart des méthodes développées au cours des trente dernières années et est conçue comme une extension de la boîte à outils commerciale "*System Identification*" de Matlab développée par L. LJUNG. Les intérêts des méthodes directes d'identification de modèles à temps continu ont été récemment illustrés à l'aide de nombreuses simulations numériques [5]. Par ailleurs, la récriture récente orientée objet des fonctions de la boîte à outils CONTSID permet d'étendre son usage [3]. L'objectif de cet article est de donner un aperçu de cette boîte à outils et de décrire les principales nouvelles fonctions apportées dans la dernière version qui dispose désormais d'une interface utilisateur (GUI).

Cet article est organisé comme suit. Un aperçu de la boîte à outil est donné dans la section II. La section III résume les principales étapes de la procédure d'identification d'un modèle à temps continu. Un exemple d'introduction du mode de commande est décrit dans la section IV. L'interface utilisateur de la CONTSID est ensuite présentée dans la section V. Enfin, quelques développements récents sont rapidement décrits dans la section VI.

II. VUE D'ENSEMBLE DE LA BOÎTE À OUTILS CONTSID

Les thèmes principaux de la boîte à outils CONTSID sont résumés ci-dessous :

- elle rassemble la plupart des méthodes développées au cours des trente dernières années [6] pour identifier des modèles linéaires de type boîte noire à temps continu de systèmes dynamiques à partir de données d'entrée/sortie échantillonnées ;
- elle regroupe des méthodes d'identification de modèles décrits sous forme de fonction de transfert ou sous forme d'état, pour les systèmes mono- et multi-variables ; elle inclut des techniques traditionnelles comme celle du filtre de variable d'état et de plus récentes comme les techniques de décomposition en sous-espaces ;
- elle peut être vue comme un supplément à la boîte à outils *System Identification* (SITB) de Matlab. Afin de faciliter son utilisation, la syntaxe des fonctions a été calquée, dans la mesure du possible, sur celle des programmes de la boîte à outils SITB[1] ;
- elle peut aisément traiter le cas de données échantillonnées à pas variable ;
- elle possède à présent une interface utilisateur (GUI) qui facilite l'application de la procédure complète d'identification ;
- la version 5 est compatible avec Matlab 6.x et Matlab 7.x et est téléchargeable à l'adresse `http ://www.cran.uhp-nancy.fr/contsid/`

[1] Il faut noter que deux boîtes à outils de Matlab sont nécessaires : *Control* et *System Identification.*

Les programmes disponibles dans la boîte à outils CONTSID couvrent les différentes étapes de la méthodologie complète de l'identification. Un grand nombre de fonctions sont cependant dédiées à l'étape d'estimation paramétrique et peuvent être distinguées selon le type de modèle à identifier :

– le modèle à identifier se présente sous la forme d'une *fonction de transfert*. Ces approches sont adaptées au cas des systèmes mono- ou multi-entrées mono-sortie. Les techniques disponibles reposent soit sur la minimisation d'une erreur d'équation nécessitant l'utilisation d'une transformation linéaire couplée à une méthode issue des moindres carrés ou de la variable instrumentale (*srivc* par exemple), soit sur la minimisation d'une erreur de sortie (*coe* par exemple) (voir partie IV) ;
– le modèle à identifier se présente sous la forme d'une *représentation d'état*. Ces approches sont particulièrement appropriées aux systèmes multi-entrées multi-sorties. Une première classe de méthodes disponibles est fondée sur la connaissance *a priori* des indices structuraux définissant la forme canonique de la représentation d'état. Le principe consiste à transformer la forme d'état canonique sous une forme polynomiale externe équivalente qui présente l'avantage d'être linéaire par rapport aux paramètres. Les paramètres de la forme externe peuvent être ensuite estimés à l'aide de méthodes dérivées de celles utilisées dans le cas de l'identification des modèles sous forme de fonction de transfert. Quelques algorithmes d'estimation paramétrique couplés à la technique des moments de Poisson sont disponibles [7]. La seconde classe de méthodes est fondée sur les techniques de décomposition en sous-espaces. La méthode des sous-espaces de la famille des *4sid* a été associée aux techniques de transformation linéaire présentant les meilleures performances [8].

La boîte à outils incorpore également quelques programmes de démonstration donnant un aperçu rapide de son utilisation. Pour les personnes habituées à obtenir un modèle à temps continu après conversion d'un premier modèle à temps discret estimé, une question peut se poser : qu'offrent ces techniques ?

Les méthodes à temps continu offrent plusieurs avantages par rapport à leurs alternatives à temps discret, pourtant mieux connues. Les principaux sont :

– de fournir directement un modèle à temps continu à partir de données échantillonnées. Les paramètres du modèle identifié sont fortement liés aux propriétés et aux coefficients physiques du système à identifier. Ce modèle peut être interprété physiquement beaucoup plus aisément que l'équivalent discret délivré par les approches traditionnelles d'identification de modèles à temps discret ; cet aspect est particulièrement attrayant pour un ingénieur ;
– de simplifier l'application de la procédure complète d'identification de systèmes pour l'utilisateur non-spécialiste. Comme dans beaucoup d'autres domaines, il y a la théorie et la pratique en identification. La théorie de l'identification de modèles à temps discret est bien établie ; la pratique, quant à elle, n'est pas simple à mettre en œuvre pour l'utilisateur non-spécialiste qui se retrouve souvent démuni devant les choix multiples à effectuer concernant, par exemple, le pré-traitement des signaux bruts avant d'effectuer l'estimation paramétrique. Cette étape de pré-filtrage est souvent primordiale, comme l'ont démontré les résultats de simulation présentés dans [9]. Les approches directes intègrent toutes un pré-filtrage implicite des données (le filtre est de plus choisi de manière automatique et optimale dans le cas de l'estimateur *srivc*) ;
– de s'appuyer sur une paramétrisation parcimonieuse. La connaissance *a priori* de l'ordre relatif du système est facilement prise en compte ce qui permet d'éviter l'estimation de zéros de discrétisation rencontrés lors de l'identification de modèles à temps discret [10] ;
– de pouvoir plus naturellement traiter des données prélevées rapidement. Ces approches sont en effet beaucoup moins sensibles aux problèmes numériques que les approches discrètes. Elles sont par conséquent très bien adaptées aux équipements actuels d'acquisition de données qui délivrent des mesures quasiment à temps continu ;
– de pouvoir aisément traiter le cas de données échantillonnés à pas variable [11]. Ce type de données est souvent rencontré lorsque les données proviennent d'analyses en temps différé, dans le cas des systèmes mécaniques où un échantillonnage angulaire est réalisé, ou encore dans le domaine biomédical [12] ou biologique.

Des exemples illustrant ces avantages sont disponibles dans les programmes de démonstration de la boîte à outils CONTSID (voir fichier *idcdemo.m*). Une analyse a été récemment conduite sur un exemple de simulation du quatrième ordre afin de comparer les approches directes et indirectes [13], [6], [9], [5]. Cet exemple illustre notamment certaines difficultés bien connues de la modélisation à temps discret (sensibilité à l'initialisation, problèmes numériques en cas d'échantillonnage rapide, connaissance *a priori* du degré relatif difficile à prendre en compte, préfiltrage des données non inhérent), et dont on peut s'affranchir en utilisant des méthodes directes d'identification de modèles à temps continu.

III. Procédure d'identification avec la boîte à outils CONTSID

La procédure visant à déterminer directement un modèle à temps continu d'un système dynamique à partir de données d'entrée/sortie échantillonnées est identique à celle utilisée pour l'identification de modèles à temps discret ; elle se fonde sur trois ingrédients majeurs :

– les données d'entrée/sortie temporelles à temps discret ;
– un ensemble de modèles (la structure du modèle) ;
– un critère pour sélectionner le modèle particulier à partir des données expérimentales (la méthode d'identification).

La procédure d'identification consiste alors, de façon itérative, à sélectionner une structure de modèle, calculer le meilleur modèle dans la structure choisie, et évaluer le modèle identifié. Cette procédure nécessite les étapes suivantes :

1. concevoir une expérimentation et recueillir les données d'entrée/sortie sur le procédé à identifier ;
2. analyser et traiter les données : éliminer les valeurs moyennes et les tendances des signaux, puis sélectionner des portions informatives sur les données initiales ;
3. sélectionner et définir une structure de modèle dans laquelle les paramètres du modèle et leur incertitude seront estimés ;

4. calculer le meilleur modèle dans la structure choisie à partir des données d'entrée/sortie et de la minimisation d'un critère ;
5. examiner les propriétés du modèle obtenu.

Si le modèle est suffisamment bon, alors la procédure est arrêtée ; sinon, retour à l'étape 3 pour essayer une autre structure de modèle. Il est également possible de tester une autre méthode d'estimation (étape 4) ou de retravailler les données d'entrée/sortie (étapes 1 et 2).

IV. Un exemple d'introduction

Le principal programme de démonstration, nommé *idcdemo.m* fournit plusieurs exemples illustrant l'utilisation et les intérêts des approches de la boîte à outils CONTSID. Ces démonstrations donnent également un aperçu d'une session typique de la boîte à outils CONTSID. Une partie du premier programme de démonstration est présentée ci-après.
Cet exemple a été conçu pour prendre en main rapidement la boîte à outils CONTSID. Il est possible d'exécuter le programme de démonstration (en tapant *idcdemo1* dans la fenêtre de commande de Matlab) et de suivre pas à pas les instructions. De plus, l'aide sur chacune des fonctions peut être obtenue à partir de la fenêtre de commande de Matlab en entrant classiquement `help nom_fonction`.
Soit un système à temps continu du deuxième ordre, sans retard, décrit par la fonction de transfert suivante, où s représente l'opérateur de différentiation par rapport au temps.

$$G(s) = \frac{3}{s^2 + 4s + 3} \tag{1}$$

Il faut tout d'abord créer un objet définissant la structure du modèle de type IDPOLY. Les polynômes sont entrés en puissance descendante de s

```
Nc=[3];
Dc=[1 4 3];
M0=idpoly(1,Nc,1,1,Dc,'Ts',0);
```

`'Ts',0` indique que le modèle est à temps continu.

Une séquence binaire pseudo aléatoire de longueur maximale et de 1016 points est choisie comme entrée u. La période d'échantillonnage est choisie égale à $0.05s$.

```
u = prbs(7,8);
Ts= 0.05;
```

On crée alors un objet DATA pour le signal d'entrée avec aucune sortie, l'entrée `u` et la période d'échantillonnage `Ts`. Le comportement de l'entrée entre deux instants d'échantillonnage est spécifié en précisant la propriété `'Intersample'` à `'zoh'` puisque l'entrée est constante par morceau, dans ce cas.

```
datau = iddata([],u,Ts,'InterSample','zoh');
```

La sortie non bruitée est simulée avec la fonction SIMC et enregistrée dans `ydet`. On crée alors un objet DATA avec la sortie `ydet`, l'entrée `u` et la période d'échantillonnage `Ts`.

```
ydet = simc(M0,datau);
datadet = iddata(ydet,u,Ts,'InterSample','zoh');
```

On peut alors estimer les paramètres d'un modèle à temps continu à partir de l'objet `datadet`, en utilisant la méthode traditionnelle des moindres carrés associée au filtre des variables d'état (`lssvf`). Les informations supplémentaires requises sont :

– le nombre de paramètres au numérateur et au dénominateur, le nombre d'échantillons pour le retard du modèle `[na nb nk]=[2 1 0]`,
– la fréquence de coupure (en rad/s) du filtre SVF, choisie ici à 2.

L'algorithme `lssvf` peut alors être utilisé comme suit :

```
Mls = lssvf(datadet,[2 1 0],2);
present(Mls);
```

ce qui conduit à :

```
CT IDPOLY model : A(s)y(t) = B(s)u(t) + e(t)
A(s) = s^2 + 3.999 s + 2.999
B(s) = 2.999
Estimated using LSSVF
Loss function 6.03708e-15 and FPE 6.07284e-15
```

On peut remarquer que les systèmes réel et estimé sont très proches. Ceci est bien sûr dû au fait que seules des données non bruitées ont été utilisées. On peut toutefois remarquer que, même dans le cas non bruité, les vrais paramètres ne sont pas exactement retrouvés. Ceci est dû aux erreurs de simulation numérique des filtres de variables d'état (SVF).

On considère désormais qu'un bruit blanc gaussien perturbe la sortie du système. La variance de ce bruit additif est ajustée de façon à obtenir un rapport signal à bruit égal à 10 dB

```
snr=10;
y = simc(M0,datau,snr);
data = iddata(y,u,Ts);
```

Les données d'entrée/sortie sont représentées sur la figure 1.

```
idplot(data,1 :1000,Ts)
xlabel('Time (sec)')
```

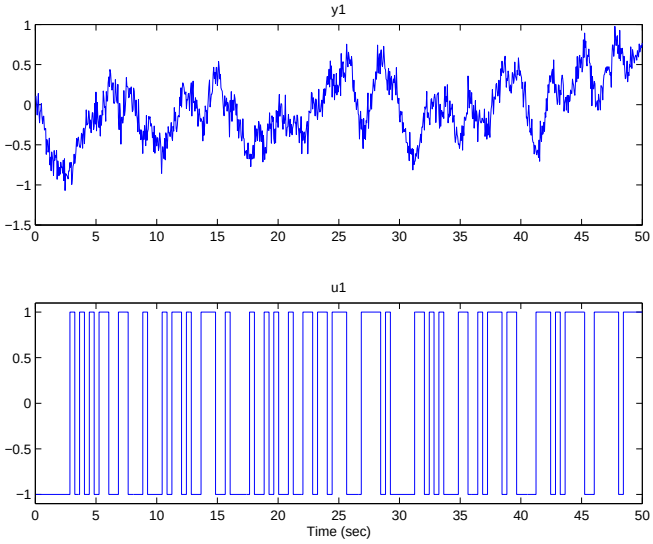

Fig. 1. Données d'entrée/sortie (cas bruité)

L'utilisation de ces données bruitées avec la fonction `lssvf` conduit inévitablement à des estimés asymptotiquement biaisées. Un algorithme de type variable instrumentale à modèle auxiliaire (`ivsvf`) peut alors être utilisé pour réduire

ce biais.

```
Miv= ivsvf(data,[2 1 0],2);
present(Miv);
```

qui conduit à :

```
CT IDPOLY model : A(s)y(t) = B(s)u(t) + e(t)
A(s) = s^2 + 3.988 s + 3.076
B(s) = 3.008
Estimated using IVSVF
Loss function 0.217742 and FPE 0.219032
```

On peut remarquer que les paramètres estimés sont proches des paramètres réels. Cependant, cette méthode de variable instrumentale utilisant les filtres de variable d'état, présente deux inconvénients même si les estimés sont asymptotiquement non biaisés :

– elle est sous-optimale, dans le sens où la variance des estimés n'est pas minimale ;
– elle requiert le choix d'un hyper-paramètre : la fréquence de coupure des filtres SVF (dans ce cas).

Il est donc recommandé d'utiliser plutôt la méthode de variable instrumentale optimale (dans ce contexte de bruit blanc de mesure en sortie) qui résout ces deux derniers problèmes. Le modèle recherché prend alors la forme d'un modèle OE à temps continu. Les ordres deviennent donc : `[nb nf nk]=[1 2 0]`. L'algorithme `srivc` peut alors être utilisé comme suit :

```
Msrivc = srivc(data,[1 2 0]);
```

Les paramètres estimés et leur écart-type peuvent alors être affichés :

```
present(Msrivc);
```

ce qui conduit à :

```
CT IDPOLY model : y(t) = [B(s)/F(s)]u(t) + e(t)
B(s) = 3.002 (+-0.1113)
F(s) = s^2 + 3.992 (+-0.1619) s + 3.067 (+-0.1061)
Estimated using SRIVC
Loss function 0.0135763 and FPE 0.0136567
```

Comparons la sortie du modèle à la sortie mesurée. Comme les données ont été simulées, nous avons la possibilité de comparer la sortie du modèle à la sortie non bruitée du système. Ceci peut être réalisé facilement en utilisant la fonction `comparec` :

```
comparec(datadet,Msrivc,1 :1000);
```

qui représente les sorties mesurées et simulées et qui affiche le coefficient de détermination défini par

$$R_T^2 = 1 - \frac{\sigma_\varepsilon^2}{\sigma_y^2} \qquad (2)$$

où ε est l'erreur de simulation.

Comme on le voit sur la figure 2, les deux sorties coïncident très bien.

Il est également possible d'examiner les résidus (erreur de simulation) de ce modèle, puis de représenter l'auto-corrélation des résidus et l'inter-corrélation entre l'entrée et les résidus :

```
residc(data,Msrivc);
```

D'après la figure 3, on peut remarquer que les résidus sont blancs et totalement non corrélés avec le signal d'entrée. Le modèle estimé est donc satisfaisant.

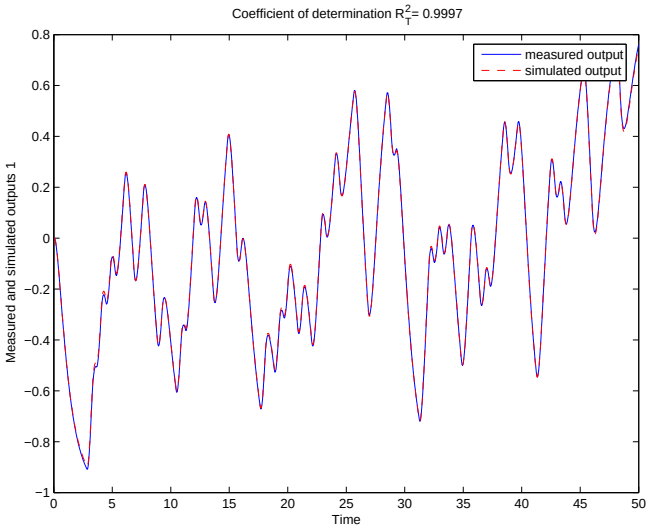

Fig. 2. Sortie non bruitée du système et sortie du modèle estimé par `srivc`

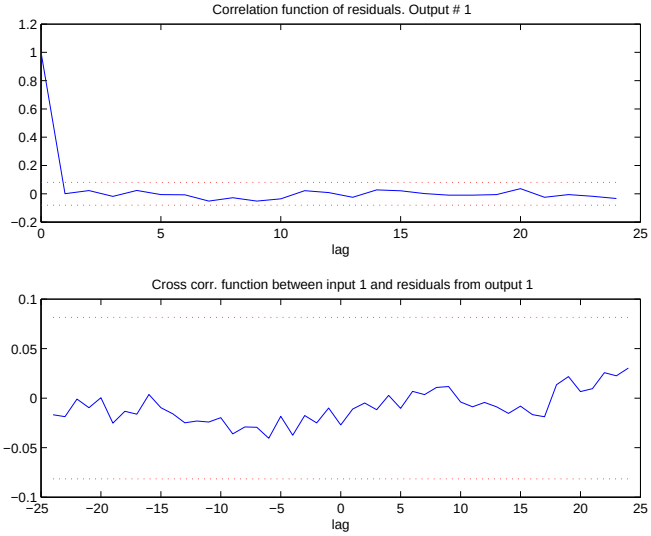

Fig. 3. Test de corrélation pour le modèle estimé par `srivc`

Comparons finalement les diagrammes de Bode du modèle estimé et du vrai système.

```
bode(Msrivc,'sd',3,'fill',M0,'y-')
```

Les régions de confiance correspondant à 3 fois l'écart-type sont également affichées. A partir de la figure 4, on peut observer que les diagrammes de Bode estimés coïncident très bien avec les diagrammes de Bode du vrai système et que les régions de confiance sont très petites.

V. L'INTERFACE UTILISATEUR DE LA BOÎTE À OUTILS CONTSID

L'interface utilisateur de la boîte à outils CONTSID se présente sous la forme d'une fenêtre principale, comme on le voit sur la figure 5, qui se divise en trois parties principales :

– une partie *données* sur la gauche, où les données sont importées, affichées, pré-traitées et sélectionnées ;
– une partie *estimation de modèle*, au milieu, où différentes structures de modèle et différentes méthodes d'identification permettant d'estimer directement un modèle à temps

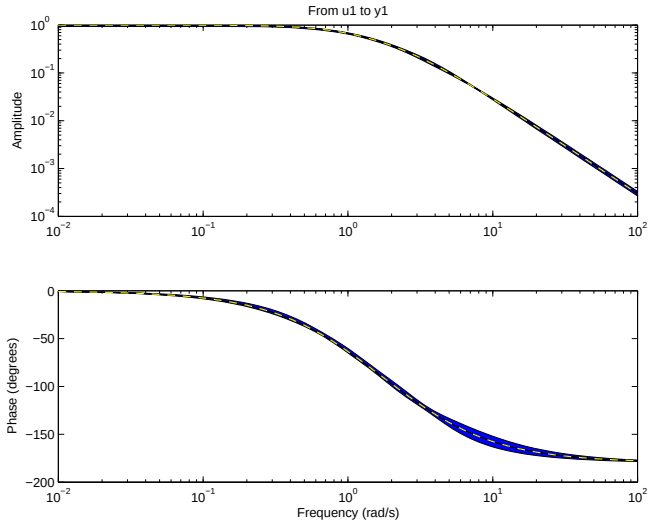

Fig. 4. Diagramme de Bode pour le modèle estimé par `srivc`

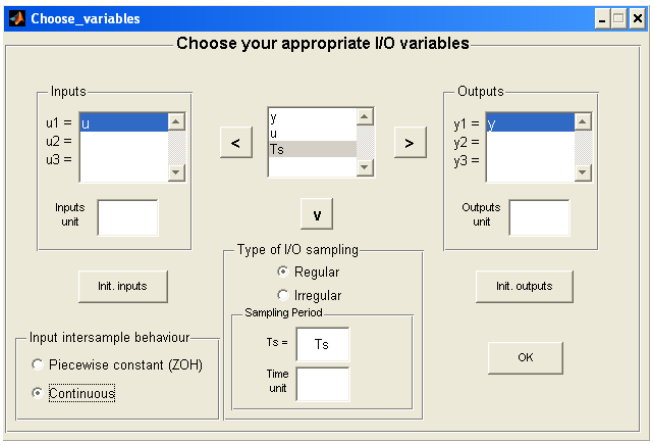

Fig. 6. Fenêtre de chargement des données

continu peuvent être utilisées ;
– une partie *validation de modèle*, à droite, où les propriétés du modèle identifié peuvent être étudiées.

L'interface utilisateur de la CONTSID peut être lancée en tapant `contsidgui` dans la fenêtre de commande de Matlab.

A. La partie «données»

Comme on peut le voir sur la figure 5, l'interface utilisateur permet à l'utilisateur de définir deux jeux de données : un pour identifier le modèle et l'autre pour effectuer des tests de validation croisée.

A.1 Importer des données mesurées.

En cliquant sur le bouton *Load data*, il est possible d'importer des données temporelles échantillonnées à partir d'un fichier $.mat$, pour des systèmes à une ou plusieurs entrées/sorties, comme le montre la figure 6. A partir de cette fenêtre, on peut sélectionner des variables d'entrée et de sortie, spécifier le type d'échantillonnage (à pas constant ou non), la période d'échantillonnage (T_s) et l'hypothèse sur le comportement de l'entrée entre deux échantillons (constant par morceaux (ZOH) ou continu).

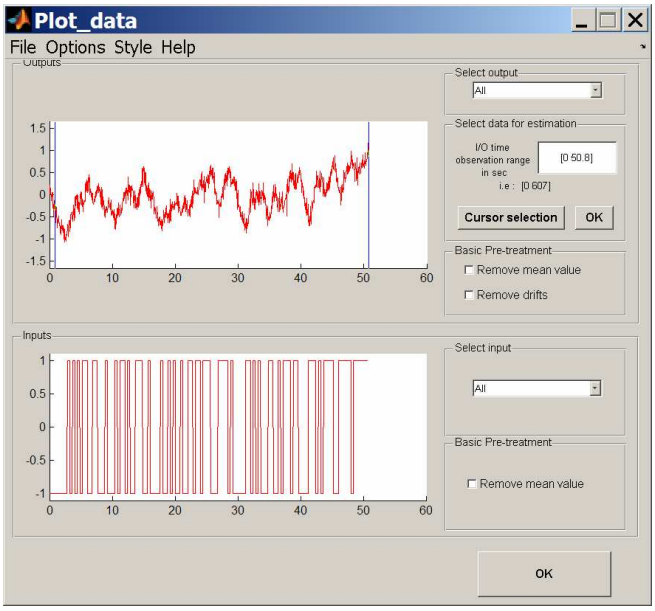

Fig. 7. Fenêtre de tracé et de pré-traitement des données

A.2 Pré-traitement et sélection des données observées.

Après l'importation des données, les opérations classiques d'analyse et de pré-traitement des données peuvent être appliquées. Un exemple de la fenêtre obtenue après avoir cliqué sur le bouton *Plot and select data* est représenté sur la figure 7. Les données d'entrée/sortie sont représentées. Cette fenêtre permet également le pré-traitement des données incluant l'élimination de l'offset, de la dérive et l'affichage des résultats après l'opération.

Très souvent, le jeu entier des données n'est pas intéressant pour l'identification. Ceci est notamment dû à deux raisons :
– les vecteurs de données incluent des valeurs erronées (aberrantes) qu'il est essentiel d'éliminer ;
– si un seul jeu de données est disponible, il est recommandé de le diviser en deux parties, une pour l'estimation du modèle et l'autre pour la validation croisée.

Le bouton *Cursor selection* permet l'insertion de deux axes verticaux sur la sortie pour définir la portion des données mesurées que l'on souhaite sélectionner.

B. La partie «estimation de modèle»

Bien que la CONTSID permette l'identification de modèles représentés sous forme de fonction de transfert et sous forme de modèle d'état, l'interface utilisateur permet uniquement l'identification de modèles représentés par une fonction de transfert à temps continu ; deux structures de modèle sont prédéfinies

$$A(s)y(t_k) = B(s)u(t_k) + e(t_k) \tag{3}$$

$$y(t_k) = \frac{B(s)}{F(s)}u(t_k) + e(t_k) \tag{4}$$

où s représente l'opérateur de différentiation, c'est à dire $s := \frac{d}{dt}$; $u(t_k)$ et $y(t_k)$ représentent respectivement les signaux d'entrée et de sortie à l'instant $t = kT_s$; $e(t_k)$ est une séquence de bruit blanc gaussien à temps discret ; $A(s)$, $B(s)$ et $F(s)$ sont des polynômes en s. L'équation (3) définit une structure de modèle ARX à temps continu alors

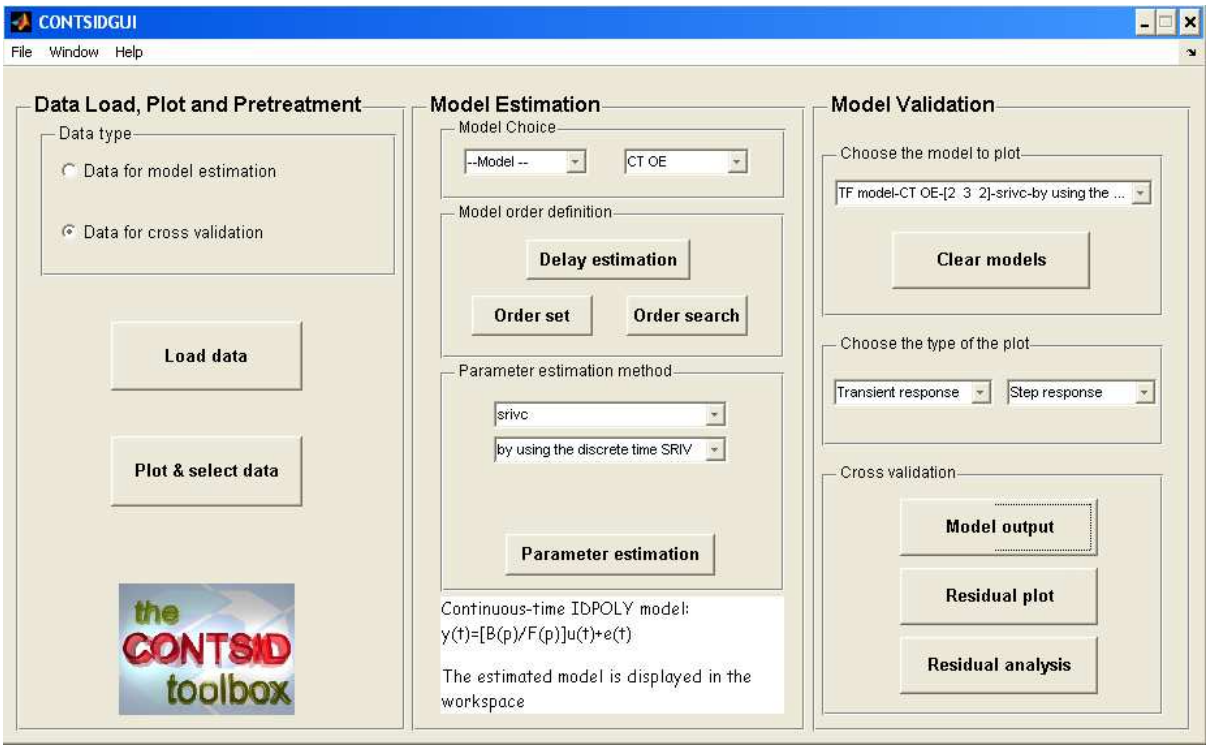

Fig. 5. Fenêtre principale de l'interface utilisateur de la CONTSID

que l'équation (4) défini une structure de modèle OE à temps continu. L'utilisateur est donc invité à choisir le type et la structure de son modèle dans les deux menus déroulant en haut de la partie *estimation de modèle*, comme montré à la figure 5.

Après avoir choisi la structure de modèle, l'utilisateur doit spécifier les ordres des polynômes et le retard du modèle à estimer.

Une première option est de déduire une estimée du nombre d'échantillons du retard à partir de l'estimation de la réponse impulsionnelle par une analyse de corrélation.

Ensuite, si les ordres de la fonction de transfert du modèle ne sont pas connus *a priori*, le bouton *Order search* permet à l'utilisateur d'effectuer une recherche automatique sur une large gamme de structure de modèles, comme illustré à la figure 8. L'utilisateur peut choisir différents critères puis afficher les résultats d'estimation dans la fenêtre de commande de Matlab. A partir de ces résultats, l'utilisateur peut choisir les meilleurs ordres pour son modèle puis sélectionner l'ordre de son modèle final en cliquant sur le bouton *Order set* depuis la fenêtre principale, comme le montre la figure 9.

Une fois le retard et le nombre de coefficients du modèle polynomial sélectionnés, les paramètres du modèle peuvent être estimés en utilisant une des méthodes paramétriques disponibles choisies dans un menu déroulant :

– dans le cas d'une structure de modèle de type OE à temps continu, l'utilisateur peut choisir d'utiliser la méthode COE (continuous-time output error) ou la méthode SRIVC (simplified refined instrumental variable) ;
– dans le cas d'une structure de modèle de type ARX à temps continu, l'utilisateur peut sélectionner six méthodes de transformées linéaires. Ces transformées linéaires sont couplées aux méthodes des moindres carrés ou de la variable instrumentale à modèle auxiliaire et requièrent le choix préalable d'un hyper paramètre [6]. Il faut noter que toutes ces transformées linéaires peuvent être interprétées comme des filtres passe-bas appliqués sur les signaux d'entrée et de sortie. Ainsi, l'hyper paramètre doit être choisi en fonction de la bande de fréquence d'intérêt.

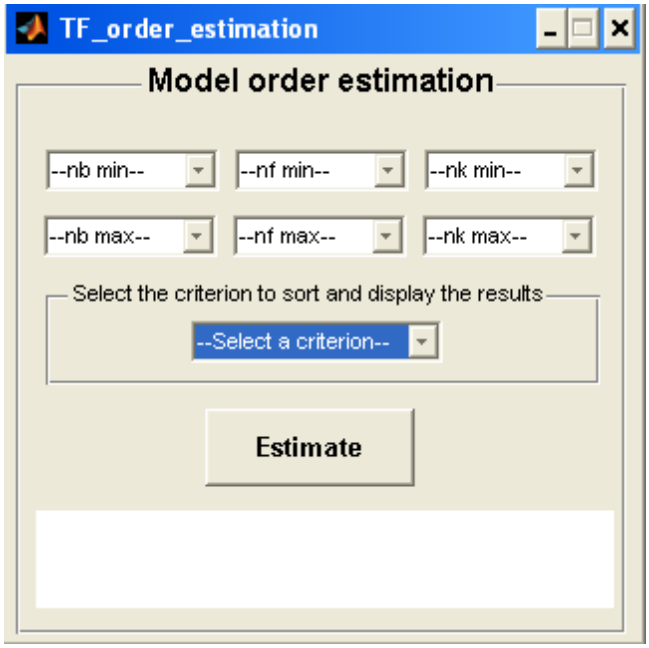

Fig. 8. Fenêtre d'estimation de l'ordre d'un modèle

C'est pourquoi, nous recommandons d'utiliser la méthode SRIVC car elle peut être initialisée de façon automatique. De plus amples détails sur les méthodes d'estimation para-

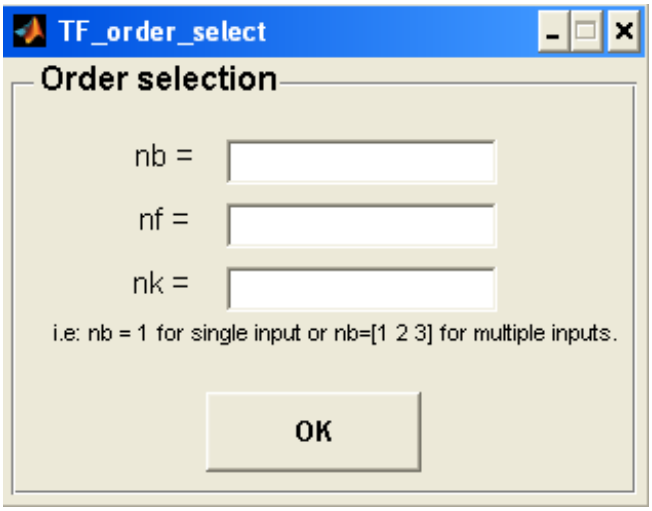

Fig. 9. Fenêtre de sélection de l'ordre du modèle

```
SRIVC initiated from SRIV
Continuous-time IDPOLY model: y(t) = [B(s)/F(s)]u(t) + e(t)
B(s) = 0.7607

F(s) = s + 1.487 (+-0.01639)

Input delays (listed by channel): 0.5
Estimated using SRIVC
Loss function 0.00622771 and FPE 0.00624084
Created:       02-Dec-2005 13:00:27
Last modified: 02-Dec-2005 13:00:27
```

Fig. 10. Résultats de l'estimation paramétrique

métrique peuvent être trouvés dans [6] et [14].
Après avoir choisi la méthode d'estimation paramétrique, le modèle identifé est représenté dans la fenêtre de commande de Matlab, comme l'illustre la figure 10, après un click sur le bouton *Parameter estimation*.

C. La partie «validation de modèle»

Lorsqu'un modèle est estimé, il apparaît dans le menu situé en haut de la partie *validation de modèles*. Plusieurs propriétés usuelles du modèle peuvent être alors évaluées à partir d'un menu déroulant en utilisant tout d'abord les données ayant servi à l'identification (validation simple) :

- *model output comparison* : représente et compare la sortie du modèle simulé avec la sortie mesurée, comme l'illustre la figure 11. Ce test permet de voir si les dynamiques du système ont bien été estimées ;
- *residual plot* : affiche les résidus, c'est à dire l'erreur de simulation de sortie ;
- *transient response* : affiche la réponse du modèle à un signal d'excitation de type impulsion ou échelon ;
- *frequency response* : affiche les diagrammes de Nyquist ou de Bode afin de visualiser les niveaux d'amortissement et les fréquences de résonance ;
- *zeros and poles* : affiche les pôles et les zéros des modèles identifiés et effectue un test sur une éventuelle compensation pôle-zéro indiquant une sur-paramétrisation ;
- *correlation test* : affiche la fonction d'auto-covariance des résidus et la fonction d'inter-covariance entre le signal d'excitation et les résidus, comme l'illustre la figure 13.

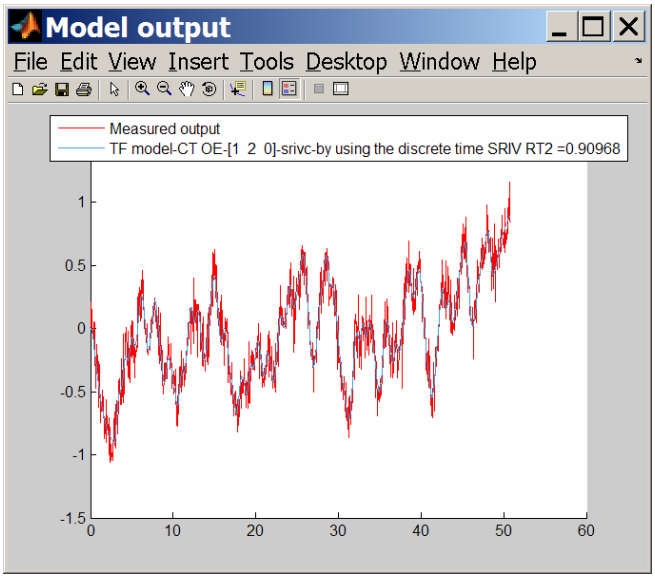

Fig. 11. Sorties mesurée et simulée du modèle

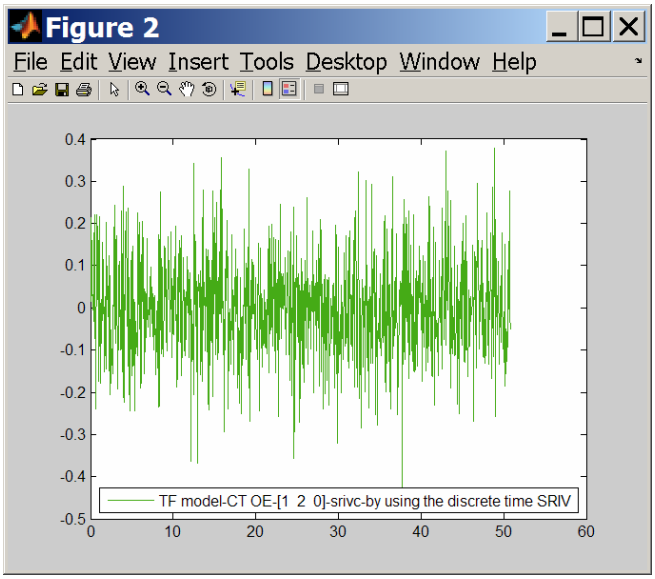

Fig. 12. Résidus

Si un jeu de données supplémentaire est disponible, alors une validation croisée peut être effectuée ; ceci consiste à comparer la sortie mesurée à la sortie du modèle simulé, puis à analyser les résidus.

VI. Quelques nouveautés dans CONTSID V5

Parmi l'ensemble des méthodes d'identification disponibles dans la CONTSID [6], celle nommée SRIVC (pour *Simplified Refined Instrumental Variable*) est robuste à de nombreuses situations pratiques [15] ; cette méthode est également disponible dans l'interface utilisateur (GUI) de la CONTSID. Elle présente l'avantage de fournir des résultats précis (et optimaux en terme de variance) dans le cas d'un bruit blanc sur la sortie. Les extensions récentes visent à étendre cette méthode. En particulier, trois nouveautés sont disponibles dans la version 5 de la CONTSID :

- une version récursive de la méthode SRIVC ;

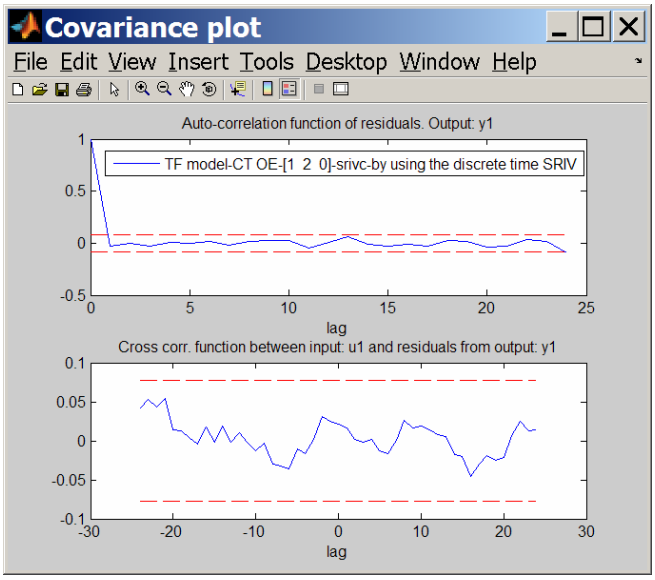

Fig. 13. Test de corrélation

– le développement d'une version permettant d'identifier des modèles multi-entrées représentés par des fonctions de transfert à dénominateurs différents ;
– le développement d'une approche de VI optimale pour identifier des modèles hybrides de type Box-Jenkins, où le modèle est représenté par une fonction de transfert à temps continu et le bruit par un processus AR ou ARMA à temps discret.

Ces développements, récemment illustrés dans [4], contribuent à l'élargissement du champ d'application de la boîte à outils CONTSID.

VII. Conclusion

Les principales nouveautés de la boîte à outils Matlab CONTSID ont été présentées. En particulier, la nouvelle interface utilisateur (GUI) a été décrite et illustrée. Dans sa version actuelle, l'interface utilisateur permet l'identification de modèles à temps continu représentés par une fonction de transfert, pour des données échantillonnées à pas constants ou non. Ces méthodes ont prouvé leur robustesse face à différents situations pratiques et nous espérons sincèrement que la nouvelle interface utilisateur va augmenter le nombre d'utilisateurs et de fans.

VIII. Remerciements

Nous souhaiterions remercier très sincèrement Michel Mensler, pour sa participation significative au développement de la première version de la boîte à outils CONTSID, ainsi que pour ses remarques constructives à propos de l'interface utilisateur.

Références

[1] H. Garnier et M. Mensler. CONTSID : a continuous-time system identification toolbox for Matlab. *5th European Control Conference (ECC'99)*, Karlsruhe (Germany), September 1999.

[2] H. Garnier et M. Mensler. The CONTSID toolbox : a Matlab toolbox for CONtinuous-Time System IDentification. *12th IFAC Symposium on System Identification (SYSID'2000)*, Santa Barbara (USA), June 2000.

[3] H. Garnier, M. Gilson, et E. Huselsein. Developments for the Matlab CONTSID toolbox. *13th IFAC Symposium on System Identification (SYSID'2003)*, pages 1007–1012, Rotterdam (The Netherlands), August 2003.

[4] H. Garnier, M. Gilson, et O. Cervellin. Latest developments for the Matlab CONTSID toolbox. *14th IFAC Symposium on System Identification (SYSID'2006)*, pages 714–719, Newcastle (Australia), March 2006.

[5] G.P. Rao et H. Garnier. Identification of continuous-time systems : direct or indirect ? *Systems Science*, 30(3) :25–50, 2004.

[6] H. Garnier, M. Mensler, et A. Richard. Continuous-time model identification from sampled data. Implementation issues and performance evaluation. *International Journal of Control*, 76(13) :1337–1357, 2003.

[7] H. Garnier, P. Sibille, et T. Bastogne. A bias-free least-squares parameter estimator for continuous-time state-space models. *36th IEEE Conference on Decision and Control (CDC'97)*, volume 2, pages 1860–1865, San Diego, California (USA), December 1997.

[8] T. Bastogne, H. Garnier, et P. Sibille. A PMF-based subspace method for continuous-time model identification. Application to a multivariable winding process. *International Journal of Control*, 74(2) :118–132, 2001.

[9] L. Ljung. Initialisation aspects for subspace and output-error identification methods. *European Control Conference (ECC'2003)*, Cambridge (U.K.), September 2003.

[10] K.J. Aström, P. Hagander, et J. Sternby. Zeros of sampled systems. *Automatica*, 20(1) :31–38, 1984.

[11] E. Huselstein et H. Garnier. An approach to continuous-time model identification from non-uniformly sampled data. *41st IEEE Conference on Decision and Control (CDC'02)*, Las-Vegas, Nevada (USA), December 2002.

[12] K.P. Wong, D. Feng, et W.C. Siu. Generalized linear least squares algorithm for non-uniformly sampled biomedical system identification with possible repeated eigenvalues. *Computer methods and programs in biomedicine*, 57 :167–177, 1998.

[13] G.P. Rao et H. Garnier. Numerical illustrations of the relevance of direct continuous-time model identification. *15th Triennial IFAC World Congress on Automatic Control*, Barcelona (Spain), July 2002.

[14] P.C. Young et H. Garnier. Identification and estimation of continuous-time data-based mechanistic (DBM) models for environmental systems. *Environmental Modelling and Software*, 21(8) :1055–1072, August 2006.

[15] H. Garnier et P.C. Young. Time-domain approaches for continuous-time model identification from sampled data. *Invited tutorial paper for the American Control Conference (ACC'2004)*, Boston, MA (USA), June 2004.