FSID - A TOOLBOX FOR MIMO FREQUENCY DOMAIN IDENTIFICATION AND RATIONAL MATRIX APPROXIMATION

T. McKelvey *

* *Chalmers University of Technology, 412 96 Gothenburg, Sweden (e-mail: tomas.mckelvey@chalmers.se).*

Abstract: An open source toolbox, FSID, implemented in the Python programming language is described. The toolbox provides scripts which estimates linear multi-input multi-output state-space models from sample data using frequency-domain subspace algorithms. Algorithms which estimate models based on samples of the transfer function matrix as well as frequency domain input and output vectors are provided. The algorithms can be used for discrete-time models, continuous-time models as well as for approximation of rational matrices from samples corresponding to arbitrary points in the complex plane. To reduce the computational complexity for the estimation algorithms, an accelerated algorithm is provided which evaluate the state-space realization of the transfer function matrix at arbitrary points.

1. INTRODUCTION

Providing software implementation of estimation algorithms is a way to make them available to a larger community of both researchers and industry. In the system identification area there is a long history of implementations both commercial Kollár et al. [1997], Ljung and Singh [2012] and non-commercial and open source type Ninness et al. [2013], Garnier and Gilson [2018].

Many system identification toolboxes are script in the MATLAB language which require the commercial software MATLAB to be able to execute. The use of the Python language for scientific computing and data analysis has increased with todays huge interest in machine learning, where the most used algorithms are opens source, see e.g. Pedregosa et al. [2011], Chollet and Others [2018], Barham et al. [2016] and available in the Python language VanRossum and Drake [2010]. Python is an open source programming language Oliphant [2007] and there exists implementations available for the most common operating systems of todays computers, e.g. Linux, Apple's Mac Os X and Microsoft's Windows. To enable ease of access to the general public, the FSID toolbox is released as open source and in a Python implementation based on the Python libraries NumPy Oliphant [2006] and SciPy Virtanen et al. [2019] which provide high level access to data types for matrices and vectors (N-dimensional objects) and high level access to LAPACK numerical linear algebra algorithms like singular value decomposition, matrix inversion, and solutions to least-squares problems etc.

The rest of the paper is organized as follows. In Section 2 we describe the various estimation scenaries the toolbox covers. In Section 3 the key functions implemented in the FSID toolbox are briefly described together with the available options.

2. ESTIMATION PROBLEMS CONSIDERED

The FSID toolbox provides software implementation of estimation algorithms for frequency domain system identification and rational matrix approximation. Specifically the software solves the multivariate state-space model estimation problem when the (measured) data is either in the form of samples of the frequency function at a discrete set of frequencies or when data is in the form samples of the Fourier transform of the input and output vectors respectively. The toolbox estimates models of state-space form given by the tuple (A, B, C, D) leading to the rational matrix valued function

$$G(z) = D + C(zI - A)^{-1}B \tag{1}$$

where $A \in \mathbb{C}^{n\times n}$, $B \in \mathbb{C}^{n\times m}$, $C \in \mathbb{C}^{p\times n}$ and $D \in \mathbb{C}^{p\times m}$. All functions have as options to constrain the estimated state-space tuple to be real-valued as well as to force the rational function to be strictly proper, i.e. $D = 0$.

2.1 Frequency function data - ffdata

Let the set of tuples $\{(z_k, G_k)\}_{k=1}^N$ define the (noisy) samples $G_k \in \mathbb{C}^{p\times m}$ from some matrix valued function $G(z) : \mathbb{C} \to \mathbb{C}^{p\times m}$ evaluated at the points $z_k \in \mathbb{C}$.

For the general rational approximation case we seek a state-space realization (A, B, C, D) which minimize

$$\sum_{k=1}^{N} \|D + C(z_k I - A)^{-1}B - G_k\|_F^2. \tag{2}$$

For continuous time (CT) models $z_k = j\omega_k$ where $\omega_k \in \mathbb{R}$. and we seek a state-space realization (A, B, C, D) such that

$$\sum_{k=1}^{N} \|D + C(j\omega_k I - A)^{-1}B - G_k\|_F^2 \tag{3}$$

is minimized. For discrete time (DT) frequency functions the formulation is similar where $z_k = e^{j\omega_k}$ and $\omega_k \in \mathbb{R}$.

$$\sum_{k=1}^{N} \|D + C(e^{j\omega_k} I - A)^{-1} B - G_k\|_F^2. \quad (4)$$

2.2 Frequency domain input and output data - fddata

We also consider the related case when the data is given as the set of tuples $\{(z_k, Y_k, U_k)\}_{k=1}^N$. We can regard Y_k as the (noisy) result of the evaluation of the matrix vector product $G(z_k)U_k \in \mathbb{C}^p$ along the given vector $U_k \in \mathbb{C}^m$. Again we seek a state-space model (A, B, C, D) which matches the fddata by minimizing

$$\sum_{k=1}^{N} \|(D + C(z_k I - A)^{-1} B)U_k - Y_k\|_F^2. \quad (5)$$

The code also implements the augmented state-space model where an additional vector x_t is jointly estimated with the state-space model according to

$$\sum_{k=1}^{N} \|DU_k + C(z_k I - A)^{-1} \begin{bmatrix} B & x_t \end{bmatrix} \begin{bmatrix} U_k \\ z_k \end{bmatrix} - Y_k\|_F^2. \quad (6)$$

For linear systems we can regard Y_k as the (noisy) sample from the Fourier transform of the output $Y(\omega_k) \in \mathbb{C}^p$ and $U_k \in \mathbb{C}^m$ is the sample from the Fourier transform of the input $U(\omega_k) \in \mathbb{C}^m$. For a linear system we have $Y(\omega) = G(\omega)U(\omega)$ and we hence seek a state-space model (A, B, C, D) which matches the fddata. In CT we have the function to be minimized as $(z_k = j\omega_k)$

$$\sum_{k=1}^{N} \|(D + C(j\omega_k I - A)^{-1} B)U_k - Y_k\|_F^2 \quad (7)$$

while in DT we have $z_k = e^{j\omega_k}$

$$\sum_{k=1}^{N} \|(D + C(e^{j\omega_k} I - A)^{-1} B)U_k - Y_k\|_F^2 \quad (8)$$

2.3 DFT data from DT time domain samples

If data is given in the form of the set of tuples $\{(y(n), u(n))\}_{n=1}^N$ corresponding to DT samples of the output $y(n) \in \mathbb{C}^p$ and input $u(n) \in \mathbb{C}^m$ to a linear system. If we generate the U_k and Y_k from the DFT of the time domain samples, a transient effects occur which needs to be accounted for. In this case the frequencies are given by $\omega_k = \frac{2\pi(k-1)}{N}$, $k = 1, \ldots, N$. If we assume the underlying linear system is given by the state-space realization (A, B, C, D) then

$$Y_k = DU_k + C(e^{j\omega_k} I - A)^{-1} \begin{bmatrix} B & x_t \end{bmatrix} \begin{bmatrix} U_k \\ e^{j\omega_k} \end{bmatrix} \quad (9)$$

where the vector $x_t \in \mathbb{C}^n$ captures the term which originates from the, in general, effects of a non-periodic input. For details see McKelvey [2000a,b]. When estimating state-space models from such data we augment the model class to (A, B, C, D, x_t) and seek a minimum solution to

$$\sum_{k \in \mathcal{K}} \|DU_k + C(e^{j\omega_k} I - A)^{-1} \begin{bmatrix} B & x_t \end{bmatrix} \begin{bmatrix} U_k \\ e^{j\omega_k} \end{bmatrix} - Y_k\|^2. \quad (10)$$

Here the set $\mathcal{K} \subseteq \{i\}_{i=1}^N$ denote the frequency indices which are used in the optimization. The indices in $\mathcal{K}$ corresponds to frequencies where we want the model fit to be good.

3. FSID PYTHON FUNCTIONS

In this section we provide an overview of the main functions included in the FSID-toolbox.

3.1 gffsid

The function `gffsid` provides an implementation of the frequency domain subspace algorithm described in McKelvey et al. [1996] augmented with the BCD-iterations introduced in Gumussoy et al. [2018]. Given samples of the matrix function the algorithm solves the general problem in (2). The implementation has the following options:

- Choice of model order n.
- Choice of number of block rows q in the algorithm. Must satisfy $q > n$.
- At the default setting the solution matrices (A, B, C, D) are constrained to be real valued. By optional argument `dtype='complex'` the estimated matrices are allowed to be complex valued.
- As default the D matrix is estimated. By optional argument `estimd=False` the D matrix is not estimated and a zero matrix is returned.

3.2 ffsid

The function `ffsid` provides an interface to `gffsid` given samples of the frequency function of a linear system. The `ffsid` can handle both CT frequency functions, i.e. (3) and DT frequency functions (4). For the CT case the bilinear tranformation is used to internally recast the CT problem as a DT problem as described in McKelvey et al. [1996]. In general this significantly improves the numerical performance as compared to a direct use of `gffsid` with $z_k = j\omega_k$. The options for the function include the ones listed above for `gffsid`. In addition we have:

- As default the DT problem (4) is solved. If `CT=True` then a CT model is estimated according to the problem (3).
- If `CT=True` then the option `T` can be set to a positive real number which will scale the frequency transformation in the bilinear transformation. See McKelvey et al. [1996] for details.

3.3 gfdsid

Function `gfdsid` provides an implementation of the frequency domain subspace algorithm described in McKelvey et al. [1996], McKelvey [1997], McKelvey et al. [2002] augmented with the BCD-iterations introduced in Gumussoy et al. [2018]. Given samples of the input and output data the algorithm solves the problem in (5) or (6) depending on the function arguments. The options for this function include the ones for `gffsid`. In addition we have:

- As default the x_t vector is estimated as illustrated in (6). By optional argument `estTrans=False` the x_t vector is not estimated and problem (5) is solved and a zero x_t vector is returned.

3.4 fdsid

The function `fdsid` is an interface to the general function `gfdsid` suitable when we have access to the Fourier trans-

form of the input and output vectors for a linear system. For the DT case this function solves the problem (8) or (10) and for the CT case solves (7). In the same way as for the `ffsid` function, the CT problem is internally converted to DT problem to improve the numerical conditioning. Besides the options listed for `gffsid` we have:

- As default the DT problem (10) is solved. If `CT=True` then a CT model is estimated according to the problem (7) and `estTrans` is set to `False`.
- As for the DT case the x_t vector is estimated as illustrated in (10). By optional argument `estTrans=False` the x_t vector is not estimated and problem (8) is solved.
- If `CT=True` then the option `T` can be set to a positive real number which will scale the frequency transformation in the bilinear transformation. See McKelvey et al. [1996] for details.

3.5 ltifr

In the estimation algorithms described above it is necessary is to derive the frequency response of the state-space model. As key step towards the frequency response is the evaluation of the frequency state. Given (A, B) and $\{z_k)\}_{k=1}N$, we want to calculate

$$X_k = (z_k I - A)^{-1} B, \quad k = 1, \ldots, N \tag{11}$$

with as low complexity as possible. A brute force implementation would require a solution to N systems of linear equations, each one involving a unique $n \times n$ complex matrix $(z_k I - A)$. If $N >> n$ it is beneficial to first perform an eigen-decomposition of the A matrix. If A is non-defective the set of the n eigenvectors to the A matrix form a linearly independent set. Let be $T \in \mathbb{C}^{n \times n}$ be a matrix with the n eigenvectors as columns. Then T is invertible and

$$T^{-1} A T = \Lambda \tag{12}$$

where Λ is a diagonal matrix with the n eigenvalues $\{\lambda_i\}_{i=1}^n$ on the diagonal. It directly follows that (11) can be rewritten as

$$X_k = (z_k I - T \Lambda T^{-1})^{-1} B = T^{-1} (z_k I - \Lambda)^{-1} T B \tag{13}$$

Since the matrix $(z_k I - \Lambda)$ is diagonal its inverse is obtained by inverting the n scalars $z_k - \lambda_i$ which is significantly less complex than the original formulation. If the A matrix is defect and does not have n linearly independent eigenvectors, the algorithm falls back to the original formulation (11). The FSID function `ltifr` provides this solution.

3.6 Open Source Implementation

The open source implementation of the FSID toolbox can be downloaded from `https://github.com/tomasmckelvey/fsid.git`. The implementation is compatible with both Python version 2.7+ as well as Python 3.x+.

REFERENCES

Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, and Others. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 265–283, 2016.

François Chollet and Others. Keras: The python deep learning library. *Astrophysics Source Code Library*, 2018.

H. Garnier and M. Gilson. CONTSID: a Matlab toolbox for standard and advanced identification of black-box continuous-time models. *IFAC Symposium on System identification, IFAC-PapersOnLine*, 51(15):688–693, 2018.

Suat Gumussoy, Ahmet Arda Ozdemir, Tomas McKelvey, Lennart Ljung, Mladen Gibanica, and Rajiv Singh. Improving Linear State-Space Models with Additional Iterations. *IFAC-PapersOnLine*, 51(15):341–346, 2018.

I. Kollár, R. Pintelon, and J. Schoukens. Frequency Domain System Identification Toolbox for MATLAB: Improvements and New Possibilities. In *Prooc: IFAC Symposium on System Identification (SYSID)*, pages 943–946, 1997. doi: 10.1016/s1474-6670(17)42968-5.

Lennart Ljung and Rajiv Singh. Version 8 of the system identification toolbox. In *16th IFAC Symposium on System Identification, IFAC Proceedings Volumes (IFAC-PapersOnline)*, volume 16, pages 1826–1831. IFAC, 2012. ISBN 9783902823069.

T McKelvey. Frequency Domain System Identification with Instrumental Variable Based Subspace Algorithm. In *Proc. 16th Biennial Conference on Mechanical Vibration and Noise, DETC'97, ASME*, pages VIB–4252, Sacramento, California, USA, sep 1997.

T McKelvey. Frequency Domain Identification. In R Smith and D Seborg, editors, *Preprints of the 12th IFAC Symposium on System Identification*, Santa Barbara, CA, USA, 2000a.

T McKelvey. On the Finite Length DFT of Input-Output Signals of Multivariable Linear Systems. In *Proc.of 39th Conference on Decision and Control*, volume 5, pages 5190–5191, Sydney, Australia, dec 2000b.

T McKelvey, H Akçay, and L Ljung. Subspace-Based Multivariable System Identification from Frequency Response Data. *IEEE Trans. on Automatic Control*, 41 (7):960–979, 1996.

T McKelvey, A Fleming, and R S O Moheimani. Subspace-Based System Identification of an Acoustic Enclosure. *ASME Transactions of Vibration and Acoustics*, 124(3), jul 2002.

Brett Ninness, Adrian Wills, and Adam Mills. UNIT: A freely available system identification toolbox. *Control Engineering Practice*, 21(5):631–644, 2013. ISSN 09670661.

Travis E Oliphant. *A guide to NumPy*, volume 1. Trelgol Publishing USA, 2006.

Travis E Oliphant. Python for scientific computing. *Computing in Science & Engineering*, 9(3):10–20, 2007.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, and Others. Scikit-learn: Machine learning in Python. *Journal of machine learning research*, 12(Oct): 2825–2830, 2011.

Guido VanRossum and Fred L Drake. *The python language reference*. Python Software Foundation Amsterdam, Netherlands, 2010.

Pauli Virtanen, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan

Bright, Stéfan J van der Walt, Matthew Brett, Joshua Wilson, K Jarrod Millman, Nikolay Mayorov, Andrew R.˜J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, \.Ilhan Polat, Yu Feng, Eric W Moore, Jake Vand erPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E.˜A. Quintero, Charles R Harris, Anne M Archibald, Antônio H Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1. 0 Contributors. SciPy 1.0–Fundamental Algorithms for Scientific Computing in Python. *arXiv e-prints*, page arXiv:1907.10121, jul 2019.