FSID - A Frequency Weighted MIMO Frequency Domain Identification and Rational Matrix Approximation Method for Python, Julia and Matlab

Downloaded from: https://research.chalmers.se, 2026-11-14 02:09 UTC

Citation for the original published paper (version of record):
McKelvey, T., Gibanica, M. (2021). FSID - A Frequency Weighted MIMO Frequency Domain Identification and Rational Matrix
Approximation Method for Python, Julia and Matlab. IFAC-PapersOnLine, 54(7): 403-408.
http://dx.doi.org/10.1016/j.ifacol.2021.08.393

N.B. When citing this work, cite the original published paper.

research.chalmers.se offers the possibility of retrieving research publications produced at Chalmers University of Technology. It covers all kind of research output: articles, dissertations, conference papers, reports etc. since 2004. research.chalmers.se is administrated and maintained by Chalmers Library

(article starts on next page)

Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnline CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 54-7 (2021) 403–408

FSID - A Frequency Weighted MIMO Frequency Domain Identification and Rational Matrix Approximation Method for Python, Julia and Matlab

T. McKelvey * **M. Gibanica** **

* *Chalmers University of Technology, 412 96 Gothenburg, Sweden (e-mail: tomas.mckelvey@chalmers.se).*
** *Volvo Cars, SE-405 31 Göteborg, Sweden (e-mail: mladen.gibanica@volvocars.com).*

Abstract: An open source toolbox, FSID, implemented in the Python, Julia and MATLAB programming languages 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. The algorithms can handle frequency dependent weighting which enable to obtain approximate BLUE estimates. 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.

Copyright © 2021 The Authors. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0)

Keywords: Software for system identification, Subspace methods, Frequency domain identification, Continuous time system estimation

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. Besides being available as a MATLAB package, the FSID toolbox is ported also as Python and Julia language packages. 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 macOS and Microsoft's Windows. A much more recent open source language suited for scientific computing is Julia, see Bezanson et al. (2017), which has gained much interest, e.g. in the control community with many software packages available.

To enable ease of access to the general public, the FSID toolbox is released as open source McKelvey (2019) 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.

Frequency domain identification refers to methods and algorithms where the model fitting is formulated in the frequency domain Ljung (1999); Pintelon and Schoukens (2001). Frequency domain subspace methods are a subclass of methods which produces models in the state space form and can directly be used for identification of MIMO systems Van Overschee (1995); McKelvey et al. (1996); Van Overschee and De Moor (1996); Van Overschee et al. (1997); Pintelon (2002).

The rest of the paper is organized as follows. In Section 2 we describe the various estimation scenaries the toolbox covers and describe the FSID algorithm and the novel frequency weighting approach which extends the ideas for SISO systems given in Pintelon (2002) to the MIMO setting. In Section 3 the key functions implemented in the FSID toolbox are briefly described together with the available options. In the final section the numerical efficiency with respect to the execution time for the three language implementations is investigated and compared.

2405-8963 Copyright © 2021 The Authors. This is an open access article under the CC BY-NC-ND license.
Peer review under responsibility of International Federation of Automatic Control.
10.1016/j.ifacol.2021.08.393

2. ESTIMATION PROBLEMS CONSIDERED

The FSID algorithms provide estimation of matrix valued rational functions given numerical samples of the rational function. The frequency domain system identification problem can be seen as a particular instance to this more general class of problems. We consider the measurement relation is given by

$$Y_k = G(z_k)U_k + V_k$$

where $z_k \in \mathbb{C}$, $Y_k \in \mathbb{C}^p$ is the measured output, $U_k \in \mathbb{C}^m$ is the input and $V_k \in \mathbb{C}^p$ is noise which we model as zero mean complex random variables with covariance matrix $R_k = \mathbf{E}\{V_k V_k^*\}$ where $(\cdot)^*$ denote Hermitian transpose and $\mathbf{E}\{\cdot\}$ denote expectation. Define the matrix W_k such that $W_k^* W_k = R_k^{-1}$. A possible choice of W_k is the Cholesky factor of R_k^{-1}. We assume we are faced with data in a set of samples in the form

$$\mathcal{D} = \{(z_k, Y_k, U_k, W_k)\}_{k=1}^M \tag{1}$$

and we seek a rational function matrix $G(z) \in \mathbb{C}^{p\times m}$ which minimize

$$\sum_{k=1}^{M} \|W_k(Y_k - G(z_k)U_k)\|_2^2 \tag{2}$$

The weighting matrix W_k will whiten the equation error $Y_k - G(z_k)U_k$ and minimizing (2) would result in the best linear unbiased estimate (BLUE) Kay (1993) if the transfer function $G(z)$ would be linear in the unknown parameters.

An alternative formulation is obtained if samples of the transfer matrix constitues the given data, i.e.

$$\mathcal{D}_G = \{(z_k, G_k, W_k)\}_{k=1}^M \tag{3}$$

and we seek an approximation

$$\sum_{k=1}^{M} \|W_k(G_k - G(z_k))\|_F^2 \tag{4}$$

where the norm $\|\cdot\|_F$ is the Frobenius norm. However, the approximation problem in (3) and (4) can be recast to the original problem by for each sample G_k in $\mathcal{D}_G$ generate the samples for $i = 1, \ldots, m$

$$U_{m(k-1)+i} = e_i, \quad Y_{m(k-1)+i} = G_k e_i, \quad z_{m(k-1)+i} = z_k \tag{5}$$

and e_i is column i in I_m the identity matrix of size $m \times m$. Hence, we will focus the algorithm presentation on the problem formulation in (2).

To solve the minimization of (2) the FSID algorithm is based on the frequency domain subspace algorithm described in McKelvey et al. (1996) augmented with the frequency weighting introduced above. From the data set $\mathcal{D}$, the algorithm generates a state-space tuple (A, B, C, D) of order n and the estimated rational matrix function is given by

$$G(z) = D + C(zI - A)^{-1}B. \tag{6}$$

where A is of size $n \times n$, I is the identity matrix and the size of the other matrices follows from the size of the rational matrix function. For a discrete time (DT) system identification problem the argument to the rational function in (6) is $z = e^{j\omega}$ while if the problem is of continuous time (CT) type then $z = j\omega$. Hence, $G(e^{j\omega})$ and $G(j\omega)$ are the DT and CT frequency function matrices of the underlying linear systems respectively see, e.g. Ljung (1999); Pintelon and Schoukens (2001).

In the next section we summarize the FSID algorithm.

2.1 The FSID algorithm

The FSID algorithm can be summarized as a multi step procedure:

(a) From the data particular structured matrices are formed and a low dimensional range space is approximated which form an estimate of the observability matrix for a state-space realization.
(b) From the estimated observability matrix the A and C matrix in the state-space tuple is determined.
(c) Fixing A and C the rational matrix function $G(z)$ is linear in D and B and optimizing (2) w.r.t. D and B is a linear least-squares problem.
(d) Fixing A and B the rational matrix function $G(z)$ is linear in D and C and optimizing (2) w.r.t. D and C is a linear least-squares problem.

As demonstrated in Gumussoy et al. (2018) iterating between step (c) and step (d) a few times can reduce the approximation error significantly for true MIMO systems.

The details of the step (a) is as follows. Let integer n denote the desired model order. From the data set $\mathcal{D}$ we define the weighted block Vandermonde matrix with q block rows satisfying $q > n$. The parameter q is known as the auxiliary model order.

$$\mathbf{U}_q = \begin{bmatrix} U_1 & U_2 & \cdots & U_M \\ z_1 w_1 U_1 & z_2 U_2 & \cdots & z_M U_M \\ z_1^2 U_1 & z_2^2 U_2 & \cdots & z_M^2 U_M \\ \vdots & \vdots & \vdots & \vdots \\ z_1^{q-1} U_1 & z_2^{q-1} U_2 & \cdots & z_M^{q-1} U_M \end{bmatrix} \mathbf{W} \tag{7}$$

which we assume has full rank mq. The weighting matrix $\mathbf{W}$ is defined as

$$\mathbf{W} = \mathrm{diag}(w_1, w_2, \ldots, w_M) \tag{8}$$

where the operator $\mathrm{diag}(w_1, w_2, \ldots)$ results in a diagonal matrix with the ordered arguments on the main diagonal and $w_k = \mathrm{tr}(W_k)/p$, i.e. the average of the diagonal elements of the weight matrix. Define the projection matrix

$$\mathbf{P}_q = I - \mathbf{U}_q^*(\mathbf{U}_q \mathbf{U}_q^*)^{-1}\mathbf{U}_q \tag{9}$$

which projects onto the nullspace of $\mathbf{U}_q$ so $\mathbf{U}_q \mathbf{P}_q = 0$. From the samples Y_k in the data set $\mathcal{D}$ we define the similar scaled and weighted block Vandermonde matrix

$$\mathbf{Y}_q = \begin{bmatrix} Y_1 & Y_2 & \cdots & Y_M \\ z_1 Y_1 & z_2 Y_2 & \cdots & z_M Y_M \\ z_1^2 Y_1 & z_2^2 Y_2 & \cdots & z_M^2 Y_M \\ \vdots & \vdots & \vdots & \vdots \\ z_1^{q-1} Y_1 & z_2^{q-1} Y_2 & \cdots & z_M^{q-1} Y_M \end{bmatrix} \mathbf{W} \tag{10}$$

If the data in the set $\mathcal{D}$ are related as

$$Y_k = (D + C(z_k I - A)^{-1}B)U_k \tag{11}$$

and (A, B, C, D) is a minimal realization of order n then, if $M \geq n + mq$

$$\mathrm{range}(\mathbf{Y}_q \mathbf{P}_q) = \mathrm{range}(\mathbf{O}_q(A, C)) \tag{12}$$

where

$$\mathbf{O}_q(A, C) \triangleq \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{q-1} \end{bmatrix} \tag{13}$$

is the (extended) observability matrix and range(X) denote the range space of matrix X. See McKelvey et al. (1996) for the details.

A matrix representing the range space is extracted by solving the structured matrix approximation problem

$$\hat{Z}, \hat{L} = \arg \min_{Z \in \mathbb{C}^{pq \times n}, L \in \mathbb{C}^{n \times M}} \|ZL - \mathbf{Y}_q \mathbf{P}_q\|_F^2. \quad (14)$$

The matrix $\hat{Z}\hat{L}$ is the optimal rank n matrix approximation to $\mathbf{Y}_q\mathbf{P}_q$. A singular value decomposition of $\mathbf{Y}_q\mathbf{P}_q$ provide the solution to (14). If, again, the data is related as in (11) we have range $\hat{Z}$ = range $\mathbf{O}_q(A, C)$ which imply that there exist a non-singular matrix T such that $\hat{Z}T = \mathbf{O}_q(A, C)$ and hence $\hat{Z} = \mathbf{O}_q(\hat{A}, \hat{C})$ where $C = \hat{C}T$ and $A = T^{-1}\hat{A}T$. This yields the estimates

$$\begin{aligned} \hat{C} &= [I_p \ 0]\, \hat{Z} \\ \hat{A} &= \arg\min_X \| \left[I_{(q-1)p} \ 0\right] \hat{Z} X - \left[0 \ I_{(q-1)p}\right] \hat{Z} \|_F^2 \end{aligned} \quad (15)$$

where the last problem is a standard least-squares problem.

Step (c) in the algorithm can be formulated as

$$\hat{D}, \hat{B} = \arg\min_{D,B} \sum_{k=1}^{M} \|W_k(Y_k - (D + \hat{C}(z_k I - \hat{A})^{-1} B) U_k)\|_2^2 \quad (16)$$

and Step (d) as

$$\hat{D}, \hat{C} = \arg\min_{D,C} \sum_{k=1}^{M} \|W_k(Y_k - (D + C(z_k I - \hat{A})^{-1} \hat{B}) U_k)\|_2^2. \quad (17)$$

Both (16) and (17) are linear least-squares problem but requires some further reorganization to yield the standard LS formulation $\min_x \sum_k \|y_k - A_k x\|^2$. Using the Kronecker product, the vectorization operator and the identity $\text{vec}(ABC) = C^T \otimes A\,\text{vec}(B)$, the vector inside the norm in (16) can be recast as

$$\begin{aligned} &W_k(Y_k - (D + \hat{C}(z_k I - \hat{A})^{-1} B) U_k) \\ &\qquad = W_k Y_k - Q_k \begin{bmatrix} \text{vec}\, D \\ \text{vec}\, B \end{bmatrix} \end{aligned} \quad (18)$$

where

$$Q_k = \left[U_k^T \otimes W_k, \ U_k^T \otimes W_k \hat{C}(z_k I - \hat{A})^{-1}\right] \quad (19)$$

For the expression in (17) we have

$$\begin{aligned} &W_k(Y_k - (D + C(z_k I - \hat{A})^{-1} \hat{B}) U_k) \\ &\qquad = W_k Y_k - \tilde{Q}_k \begin{bmatrix} \text{vec}\, D \\ \text{vec}\, C \end{bmatrix} \end{aligned} \quad (20)$$

where

$$\tilde{Q}_k = \left[U_k^T \otimes W_k, \ ((z_k I - \hat{A})^{-1} \hat{B} U_k)^T \otimes W_k\right] \quad (21)$$

In the next sections we summarize the different system identification cases that arises when primary data is given directly as frequency domain data and the case when the primary data is time domain samples.

2.2 Frequency domain input and output data

When the data is given as the set of tuples $\{(z_k, Y_k, U_k, W_k)\}_{k=1}^M$ 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 data by minimizing

$$\sum_{k=1}^{M} \|W_k(D + C(z_k I - A)^{-1} B) U_k - Y_k)\|_F^2. \quad (22)$$

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

$$\sum_{k=1}^{M} \|W_k(DU_k + C(z_k I - A)^{-1} \left[B \ x_t\right] \begin{bmatrix} U_k \\ z_k \end{bmatrix} - Y_k)\|_F^2. \quad (23)$$

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 data. In CT we have the function to be minimized as $(z_k = j\omega_k)$.

2.3 DFT data from DT time domain samples

Assume 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 $u(n)$ and $y(n)$ respectively, a transient effects occur which needs to be accounted for McKelvey (2000a). In this case the frequencies are given by $\omega_k = \frac{2\pi(k-1)}{N}$, $k = 1, \ldots, N$ since the frequency domain samples come from the DFT . If we assume the underlying linear system is given by the state-space realization (A, B, C, D) then

$$Y_k = DU_k + C(e^{j\omega_k} I - A)^{-1} \left[B \ x_t\right] \begin{bmatrix} U_k \\ e^{j\omega_k} \end{bmatrix} \quad (24)$$

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

$$\sum_{k \in \mathcal{K}} \|W_k(DU_k + C(e^{j\omega_k} I - A)^{-1} \left[B \ x_t\right] \begin{bmatrix} U_k \\ e^{j\omega_k} \end{bmatrix} - Y_k)\|^2. \quad (25)$$

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. MAIN FSID FUNCTIONS

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

All estimation functions estimate a real valued state-space tuple by default. If a complex valued solution is desired an option can be given. If a strictly proper rational function is desired the option gan be given which results in that the D matrix is not estimated and a zero matrix is returned. The calling syntax for the different language implementations are slightly different and we refer to the help texts for the specific syntax.

3.1 Function gfdsid

Function `gfdsid` provides an implementation of the FSID algorithm described in Section 2.1 and is based on the algorithms described in McKelvey et al. (1996); McKelvey (1997); McKelvey et al. (2002) augmented with the BCD-iterations introduced in Gumussoy et al. (2018). The frequency weighting approach is novel to this paper. Given samples of the input and output data the algorithm solves the problem in (22) or (23) depending on the function arguments. The implementation has the following options:

- Choice of model order n.
- Choice of number of block rows q (auxiliary order) in the algorithm which must satisfy $q > n$.
- As default the x_t vector is estimated as illustrated in (23). By supplying an optional argument the x_t vector is not estimated and problem (22) is solved and a zero x_t vector is returned.

3.2 fdsid

The function `fdsid` is an interface to the general function `gfdsid` suitable when we have access to the Fourier transform of the input and output vectors for a linear system either from a CT problem $z_k = \omega_k$ or a DT problem $z_k = e^{j\omega_k}$. For the CT case, the CT problem is internally converted to DT problem to improve the numerical conditioning. Besides the options listed for `gfdsid` we have:

- As default the DT problem (25) is solved. By Supplying an option a CT model is estimated according to the problem with $z_k = j\omega_k$ and the transient term x_t is not estimated.
- The x_t vector is estimated as illustrated in (25). By supplying an optional argument the x_t vector is not estimated and problem (22) is solved instead.
- If a CT problem is considered the scaling 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 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

$$X_k = (z_k I - A)^{-1} B, \quad k = 1, \ldots, N \tag{26}$$

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 $T \in \mathbb{C}^{n \times n}$ be a matrix with the n eigenvectors as columns. Then T is invertible and

$$T^{-1} A T = \Lambda \tag{27}$$

where Λ is a diagonal matrix with the n eigenvalues $\{\lambda_i\}_{i=1}^n$ on the diagonal. It directly follows that (26) can be rewritten as

$$X_k = (z_k I - T\Lambda T^{-1})^{-1} B = T^{-1}(z_k I - \Lambda)^{-1} T B \tag{28}$$

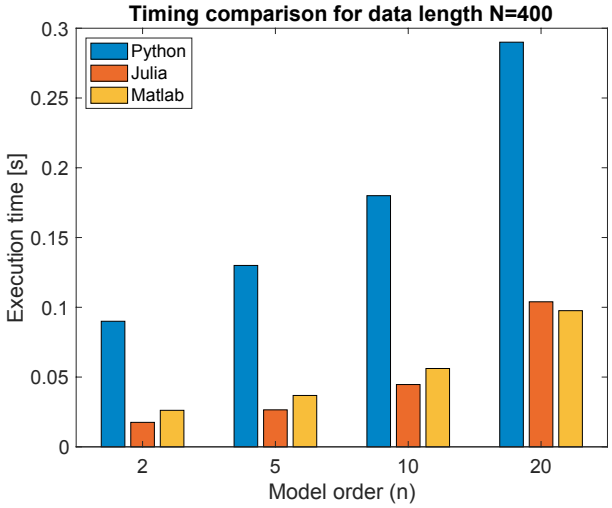

Fig. 1. Results from execution time comparison. Bar graphs shows execution time variations for varying model orders.

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 (26). The FSID function `ltifr` provides this solution.

4. EXECUTION TIME COMPARISON

In this section we compare the implementations in the three different languages with respect to the execution time for different model orders and number of frequency data. All tests are performed using `gfdestim` for estimating a systems with an input dimension 2 ($m = 2$) and an output dimension of 8 ($p = 8$). A full factorial test is conducted where the state-space model order (n) is varied with values from the set $\{2, 5, 10, 20\}$ and the number of frequency samples (N) is varied with values from the set $\{100, 200, 400, 800\}$. For each setting investigated, 100 model estimations are performed and the average execution time is recorded. The auxiliary model order q is selected such that $q = n + 1$ and the options are set to estimate real-valued system matrices, estimate the D matrix and the transient vector x_t. The frequency weighing was disabled which implies $W_k = I$ for all k. The computer used for the test is an Apple MacBook Pro with a 2.9 GHz Quad-Core Intel Core i7 processor and 16 GB 2133 MHz LPDDR3 memory running macOS version 11.3.1. The Julia version is 1.1.0, the Python version is 2.7.16 and the MATLAB version is R2020b Update 1 (9.9.0.1495850).

The results on how the average execution time varies for different model orders is illustrated in Figure 1 when the number of frequency data samples is 400. In Figure 2 the bar graph illustrates how the execution time varies with the number of data (N) for a fixed model order of 10. From these results we note that the Julia and MATLAB langage implementations outperform the Python language version and that the Julia implementation has slightly better performance than the MATLAB implementation for long data sets while the MATLAB version is slightly better than the Julia version for larger model orders.

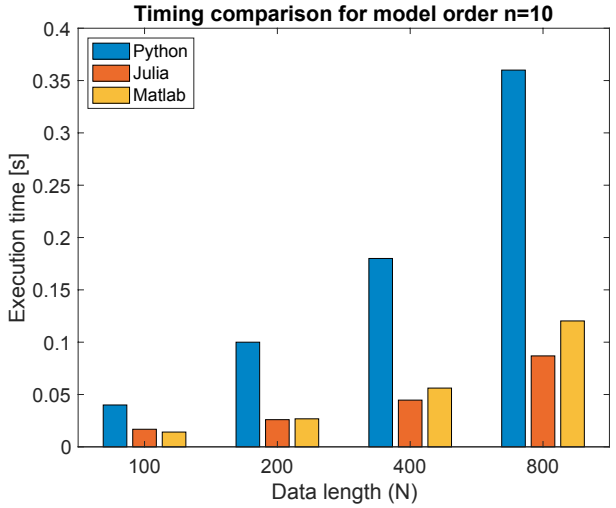

Fig. 2. Results from execution time comparison. Bar graphs shows execution time variations for varying data lengts.

5. OPEN SOURCE IMPLEMENTATION

The open source implementation of the FSID toolbox, for all three platforms, can be downloaded from `https://github.com/tomasmckelvey/fsid`

REFERENCES

Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., and Others (2016). Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, 265–283.

Bezanson, J., Edelman, A., Karpinski, S., and Shah, V.B. (2017). Julia: A fresh approach to numerical computing. *SIAM review*, 59(1), 65–98. URL `https://doi.org/10.1137/141000671`.

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

Garnier, H. and Gilson, M. (2018). 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.

Gumussoy, S., Ozdemir, A.A., McKelvey, T., Ljung, L., Gibanica, M., and Singh, R. (2018). Improving Linear State-Space Models with Additional Iterations. *IFAC-PapersOnLine*, 51(15), 341–346. doi:10.1016/j.ifacol.2018.09.158.

Kay, S.M. (1993). *Fundamentals of Statistical Signal Processin - Estimation Theory*. Prentice Hall.

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

Ljung, L. (1999). *System Identification: Theory for the User*. Prentice-Hall, Englewood Cliffs, New Jersey, second edition.

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

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

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

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

McKelvey, T. (2019). fsid - Frequency Domain Subspace Based Identification - Software. Technical report. URL `https://github.com/tomasmckelvey/fsid`.

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

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

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

Oliphant, T.E. (2006). *A guide to NumPy*, volume 1. Trelgol Publishing USA.

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

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., and Others (2011). Scikit-learn: Machine learning in Python. *Journal of machine learning research*, 12(Oct), 2825–2830.

Pintelon, R. (2002). Frequency-domain subspace system identification using non-parametric noise models. *Automatica*, 38, 1295–1311.

Pintelon, R. and Schoukens, J. (2001). *System Identification - A frequency domain approach*. IEEE Press.

Van Overschee, P. (1995). *Subspace Identification, Theory - Implementation - Application*. Ph.D. thesis, Katholieke Universiteit Leuven, Kard. Mercierlaan 94, 3001 Leuven (Heverlee), Belgium.

Van Overschee, P., De Moor, B., Dehandschutter, W., and Swevers, J. (1997). A subspace algorithm for the identification of discrete time frequency domain power spectra. *Automatica*, 33(12), 2147–2157.

Van Overschee, P. and De Moor, B. (1996). Continuous-time frequency domain subspace system identification. *Signal Processing*, 52(2), 179–194. doi:10.1016/0165-1684(96)00052-7.

VanRossum, G. and Drake, F.L. (2010). *The python language reference*. Python Software Foundation Amsterdam, Netherlands.

Virtanen, P., Gommers, R., Oliphant, T.E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S.J., Brett, M., Wilson, J., Jarrod Millman, K., Mayorov, N., Nelson, A.R., Jones, E., Kern, R., Larson, E., Carey,

C.J., Polat, b., Feng, Y., Moore, E.W., Vand erPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E., Harris, C.R., Archibald, A.M., Ribeiro, A.H., Pedregosa, F., van Mulbregt, P., and Contributors, S... (2019). SciPy 1.0–Fundamental Algorithms for Scientific Computing in Python. *arXiv e-prints*, arXiv:1907.10121.