XPCA: A MATLAB® Application Toolbox for EIV system identification

Ranjeet Nagarkar[1], Arun K.Tangirala[2], and Shankar Narasimhan[3]

[1]Systems Engineering Group, Department of Chemical Engineering, Indian Institute of Technology Madras, Chennai, Tamil Nadu 600036, India

Abstract **- Through this paper we introduce an exclusive toolbox for errors-in-variable (EIV) system identification on the Matlab® platform. This application, the first of its kind on this platform, is designed primarily for classroom training, academic purposes, industrial and commercial Applications. The toolbox contains options for constructing mathematical data-driven models for EIV linear-time invariant (LTI) systems for *dynamic* and *static* models from measured input-output data in the presence of heteroskedastic error variables. In the present version of the App Toolbox the class of systems are restricted to single-input single-output (SISO) dynamic systems. The toolbox is also useful for model validation using bootstrapping for confidence interval calculation and residual analysis using EIV Kalman filter. The App Toolbox has been designed for novice and expert User to estimate and validate models using multiple options and methods. The back end of the App Toolbox has been designed in the "strategic pattern" and "singleton pattern" such that future developers can plug new options and methods very easily without disturbing the integrity of other components in the application.**

I Introduction

In this article, we introduce a System Identification Toolbox Application on the Matlab® platform. The Application, the first of its kind on this platform, is designed primarily for classroom training, academic purposes, Industrial and Commercial Application. Key features of the Application include:

- Data Processing module for loading data from a file or Matlab Workspace Environment as a MATLAB Matrix file or IDData Object.
- Model Configuration Module allows user to choose the type of model i.e. Noisy or Noise-free, the model type Static or Dynamic and known or unknown error variance.
- Model Estimation Module allows users to Estimate order, Error Variance, and Build Models only for SISO systems. We use the IPCA (1) and DIPCA (2) algorithms in the toolbox. the
- Model Validation Module allows user to Noise-freehe Confidence interval(**? ?**) and EIV Kalman Filter(4) residual Auto-correlation Function(ACF) to validate the model.
- Expert users can change the Type of Model order estimation and Variance Estimation methods, along with Bootstrapping parameters for Confidence Interval calculation.
- Cross Validation module allows users to Validate model against a test data set.

MATLAB® is a programming platform designed specifically for engineers and scientists. The heart of MATLAB is the MATLAB language, a matrix-based language allowing the most natural expression of computational mathematics. We chose MATLAB because of MATLAB App Designer feature, which lets you create professional apps without having to be a professional software developer. It allows you to drag and drop visual components to lay out the design of your graphical user interface (GUI) and use the integrated editor to quickly program its behavior.

There are quite a few actively maintained commercial libraries for dynamic system identification. The most commonly used one is the MathWorks system identification toolbox(5), which has exhaustive routines for system identification. There are relatively few non-commercial libraries. UNIT (6) is a freely available MATLAB ®based toolbox that supports a wide range of model structures, estimation methods and algorithms. The ITSIE toolbox(7) is a standalone software with emphasis on training and education of system identification principles.

The sysid toolbox Application developed in this work incorporates the Iterative Principal Component Analysis (IPCA) algorithm(1) for iteratively calculating a static model when measurement errors in different variables are unequal and are correlated, and Dynamic Iterative Principal Component Analysis (DIPCA) algorithm (2) for constructing Mathematical data-driven models for EIV LTI SISO systems.

This sysid toolbox Application has some advantages over the other sysid toolbox in that it helps even the most novice user to develop a basic model without much knowledge on the subject and to store and apply the model quickly. The modular design of the Strategic pattern(8) also allows a new developer to quickly plug-in a new method into the App without making changes elsewhere.

The article is organised as follows. We present a brief review of the theoretical background in Section II. In Section III we explore various components of the Application. In Section IV we use examples to illustrate the Toolbox. Section V provides concluding remarks

II Theoretical Background Overview

A. PCA for EIV.

Principal Component Analysis (PCA) has primarily been considered in the area of multivariate process/signal analysis, as a statistical technique for dimensionality reduction or signal compression. PCA is defined as an orthogonal linear transformation that transforms the data to a new coordinate system such that the greatest variance by some scalar projection of the data comes to lie on the first coordinate (called the first principal component), the second greatest variance on the second coordinate, and so on. For our purpose we look at an LTI process:

Consider a linear time-invariant process described by M (noise-free) variables $x_i[k], i = 1,\ldots,M$ that are related or constrained by d < M linear equations at any sampling instant k,

$$\mathbf{A}_0\mathbf{x}[k] = \mathbf{0}, \quad \mathbf{A}_0\mathbf{R}^{d\times M} \tag{1}$$

where A_0 is the constraint matrix and $x[k] = [x_1[k]\ldots xM[k]]^T$ is the vector of noise-free variables at the kth instant. The objective of PCA-based regression in the EIV scenario is twofold: (i) determine the number of constraints (row dimension) d and (ii) estimate the constraint matrix A_0, from N noisy measurements of the M variables $x_i[k], k = 0,1,\ldots,N-1$. The measurements are assumed to be generated according to

$$\mathbf{z[k]= x[k]+ e[k]} \tag{2}$$

Introduce

$$\mathbf{X = [x[0]\ x[1]\ \ldots x[N\ \ 1]]}^T \tag{3a}$$

$$\mathbf{Z = [z[0]\ z[1]\ \ldots z[N\ \ 1]]}^T \tag{3b}$$

so that eq2 can be written in the matrix form as

$$\mathbf{Z = X + E} \tag{4}$$

The following assumptions are made on the random errors:

- $e[k] \sim \mathcal{N}(0, \sigma^2\mathbf{I})$
- $E(e[j]e^T[k]) = \sigma^2\delta_{jk}\,\mathbf{I}_{M\times M}$
- $E(x[j]e^T[k]) = \mathbf{0}, \forall_j,\ k$

where $E(.)$ is the expectation operator, δ_{jk} is the Dirac delta function. For case when all elements have identical variances σ^2, the model can be formulated as an optimization problem as described below:

$$\min_{\mathbf{A,x[k]}} \sum_{k=1}^{N} (z[k]-x[k])^T(z[k]-x[k]) \tag{5a}$$

$$\text{subject to} \quad \mathbf{Ax}[k] = \mathbf{0}_{d\times 1}, k = 1,\ldots,N \tag{5b}$$

$$\mathbf{AA}^T \tag{5c}$$

expressed as $\mathbf{XA}^T = \mathbf{0}_{N\times d}$ for no errors i.e. $e[k] = 0$ an orthonormal basis for the null space of $\mathbf{X}$ can be obtained using singular value decomposition (svd) as:

$$\mathbf{X} = \mathbf{USV}^T, \quad \mathbf{X} \in \mathbb{R}, N > M \tag{6}$$

where U and V are $N\times N$ and $M\times M$ orthogonal unitary matrix, while S is $\begin{bmatrix}\mathbf{S}_1 & \mathbf{0}\\ \mathbf{0} & \mathbf{S}_2\end{bmatrix}$,

$$\mathbf{S}_1 = \begin{bmatrix}\sigma_1 & 0 & \ldots & 0\\ 0 & \sigma_2 & \ldots & 0\\ \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \ldots & \sigma_{M-d}\end{bmatrix},\ \mathbf{S}_2 = \mathbf{0}_{d\times d} \tag{7}$$

, where $\sigma_r \geq \sigma_{r+1}$ till σ_{M-d} while remaining singular values are 0.

Inside $\mathbf{V}$ (eq 6), the matrix of orthonormal right singular vectors, corresponding to the d zero singular values provides a basis for the constraint matrix $\mathbf{A}_0$.

If measurements of $\mathbf{X}$ are noisy, then an eigenvalue analysis of the sample covariance matrix $\mathbf{S} \triangleq \frac{1}{N}\mathbf{Z}^T\mathbf{Z}$ does not result in any zero eigenvalue since there exists no pair of columns of $\mathbf{Z}$ that are linearly related. From the assumptions made regarding the measurements errors, it can be proved that

$$\sum\nolimits_z = \sum\nolimits_x + \sum\nolimits_e \tag{8}$$

where

$$\sum_x = \lim_{N\to\infty} E\mathbf{[S]}_x = \lim_{N\to\infty} \frac{1}{N} E\mathbf{X}^T\mathbf{X} \tag{9a}$$

$$\sum_z = \lim_{N\to\infty} E\mathbf{[S]}_z = \lim_{N\to\infty} \frac{1}{N} E[\mathbf{Z}^T\mathbf{Z}] \tag{9b}$$

$\sum_x$ exists under the quasi-stationary assumption Ljung(5) on $z[k]$. Under the assumption of "homoskedasticity", (eq 8) can be restated as:

$$\sum_z = \sum_x + \sigma_e^2 \mathbf{I} \tag{10}$$

The last d eigenvalues of diagonal matrix $\sigma_e^2\mathbf{I}$ are equal to σ_e^2, with eigenvectors of σ_z and σ_x being identical. Hence S_2 (eq7) here becomes $\sqrt{N}\sigma_e \mathbf{I}_{d\times d}$

However for a general scenario, i.e., non-diagonal $\sum_z$ with unequal diagonal entries, it is not possible to consistently estimate the constraint matrix. In this toolbox we use Iterative PCA (IPCA) as proposed by Narasimhan and Shah(1).

B. Iterative PCA..

Here we transform the heteroskedastic errors case to the homoskedastic case with appropriate scaling of the data, i.e., scaling Z with ZW, where $W \in \mathbb{R}^{M\times M}$ is a suitable scaling matrix. IPCA proposes to scale the data with the inverse square root of noise co-variance matrix (eq8), i.e., $W = \sum^{\frac{-1}{2}}$. Hence, Transformed data matrix becomes $Z_s \triangleq Z\sum_e^{\frac{-1}{2}}$, with variance matrix as, $\sum_{Z_s} = \sum_{X_s} + I$. Hence, one applies PCA to the scaled data and recovers the scaled matrix from the original variables from that for the scaled ones.

A key feature of IPCA is that both the noise co-variance and constraint matrices are estimated simultaneously by iterating between two steps. One step consists of estimating the (basis for) constraint matrix given the co-variance matrix, while the second step estimates the co-variance matrix from the estimate of A_0, i.e., the number of constraints is known.

Denoting the estimate of A_0 at the ith iteration by $\hat{\mathrm{A}}^{(i)}$, one generates constraint residuals as

$$\mathbf{r}^{(i)}[k] \triangleq \hat{\mathbf{A}}^{(i)}\mathbf{z}[k] = \hat{\mathbf{A}}^{(i)}\mathbf{x}[k] + \hat{\mathbf{A}}^{(i)}\mathbf{e}[k] \tag{11}$$

Hence, one can then conclude $\hat{\mathbf{A}}^{(i)} = \mathbf{T}\mathbf{A}_0$. As residuals are uncorrelated and normally distributed with zero mean and covariance matrix $\sum_r^{(i)}$ given by $\sum_r^{(i)} = \hat{\mathbf{A}}^{(i)}\sum_e(\hat{\mathbf{A}}^{(i)})^T$ to estimate $\sum_e$ we solve the following likelihood problem,

$$\min_{\sum_e} Nlog|\hat{\mathbf{A}}^{(i)}\sum_e(\hat{\mathbf{A}}^{(i)})^T| + \sum_{k=1}^{N}(\mathbf{r}^{(i)}[k])^T(\hat{\mathbf{A}}^{(i)}\sum_e(\hat{\mathrm{A}}^{(i)})^T)^{-1} \times \mathbf{r}^{(i)}[k] \tag{12}$$

The number of constraints d i.e. size of $\mathbf{A}_0$. Similar to (eq8) eigenvalues of $\sum_{x_s}$ map to unity valued eigenvalues of $\sum_{z_s}$. Here we begin with some minimum value of d governed by an identifiability constraint

$$\frac{d(d+1)}{2} \geq P \tag{13}$$

where P is the number of elements of $\sum_e$ that needs to be estimated, which in the case of $M \times M$ diagonal matrix is $P = M$.

C. Dynamic Iterative PCA.

DIPCA extends the application of IPCA for identifying dynamic process models. In this App we are interested in the class of parametric deterministic SISO linear time-invariant dynamic input (u^*)output (y^*) systems described by:

$$(y^*[k]) + \sum_{i=1}^{n_a} a_i y^*[k-i] = \sum_{j=D}^{n_b} b_j u^*[k-j] \tag{14}$$

where n_a and n_b are output and input order, respectively, and $D \leq n_b$ is the input-output delay. Also here we introduce,

$$\eta \triangleq max(n_a, n_b) \tag{15}$$

which is the equation order.

The EIV dynamic identification problem is that of estimating the following quantities:

- Output and input orders n_a and n_b, respectively
- Input-output delay, D
- The coefficients of the difference equation, $a_{i\,i=1}^{n_a}$ and $b_{j\,j=D}^{n_b}$
- Variances of the errors in output and input measurements are $\sigma_{e_y}^2$ and $\sigma_{e_u}^2$, respectively

Here we construct a matrix of lagged variables since the dynamic model in (eq 14) can be viewed as a static model on lagged variables. The columns of this lagged matrix are constructed by stacking $\{y[k]\}$, $\{y[k-1]\}$ up to $\{y[k-L_y]\}$, and $\{u[k]\}$, $\{u[k-1]\}$ up to $\{u[k-L_u]\}$, where L_y and L_u are user-specified maximum lags for the output and input, respectively.

We apply IPCA on this lagged matrix to find both input-output variance and the model coefficients, however this gives us multiple constraints, hence we need an extra equation to relate number of constraints d, the stacking lag L, and the equation order η can be derived.

$$\hat{\eta} = L - \hat{d} + 1 \tag{16}$$

As there are only two input-output variances, i.e.,

$$\begin{bmatrix} \sigma_{e_y}^2\mathbf{I}_{L+1} & \mathbf{0} \\ \mathbf{0} & \sigma_{e_u}^2\mathbf{I}_{L+1} \end{bmatrix} \tag{17}$$

similar to the identifiability constraint from eq13 here we get,

$$d(d+1)/2 > 2 \quad \textbf{(18)}$$

hence $L_{min} = \eta + 1$, so the user must choose a "large enough" value of L(lag).

Here the optimization problem for estimating $\sum_e$ as in eq12 in case of DIPCA is,

$$\min_{\sum_e} (N-L) log|\hat{\mathbf{A}}^{(i)} \sum_e (\hat{\mathbf{A}}^{(i)})^T| +$$

$$\sum_{k=1}^{N-L} (\mathbf{r}^{(i)}[k])^T (\hat{\mathbf{A}}^{(i)} \sum_e (\hat{\mathrm{A}}^{(i)})^T)^{-1} \mathbf{r}^{(i)}[k] \quad \textbf{(19)}$$

subject to $\begin{bmatrix} \sigma_{e_y}^2 \mathbf{I}_{L+1} & \mathbf{0} \\ \mathbf{0} & \sigma_{e_u}^2 \mathbf{I}_{L+1} \end{bmatrix}$

Finally, similar to IPCA, we obtain the difference equation model from the eigenvector corresponding to the unity (last) eigenvalue of the covariance matrix of $Z_{\hat{\eta}}$. For further reading please refer (2).

D. Strategic Pattern.

In the Strategic pattern, a class behavior or its algorithm can be changed at run time. This type of design pattern comes under behavior pattern.

In Strategy pattern, we create objects which represent various strategies and a context object whose behavior varies as per its strategy object. The strategy object changes the executing algorithm of the context object.

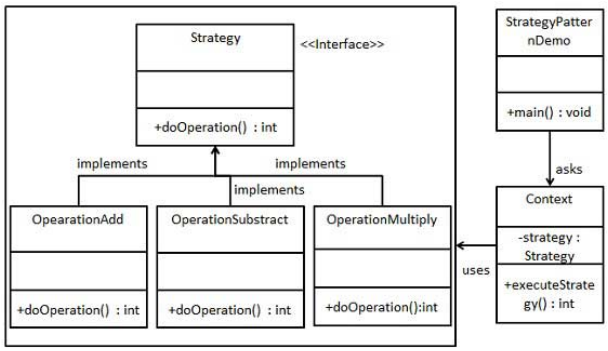

Fig. 1. An image of the Toolbox

From a developer point of view too it becomes very easy to add a new method or feature to the Toolbox. This pattern makes the design very modular and reusable, thus any feature can be added just by plugging in a template function into the already existing framework.

III The SYSID Toolbox

The Toolbox Application has advantages over the already available System identification toolbox:

- This is the only App which allows the user to build model for the EIV case.
- The user can also build a static model with this Toolbox.
- The Toolbox also contains a special case of Kalman filter for EIV (Vipul(4)) for model validation using Kalman filter.
- The Toolbox has been designed to accommodate a novice as well as an expert user, i.e., a user with basic understanding of systems Engineering can also build, validate and use the model very easily.
- The back-end design of the Toolbox follows a "strategic pattern" which allows any new developer to very easily plug-n-play their contribution into the App, without tampering with any other backend code.
- The User also has to complete steps in one panel only then the user may proceed to the next panel, which avoids confusion and mistakes while building the Model.

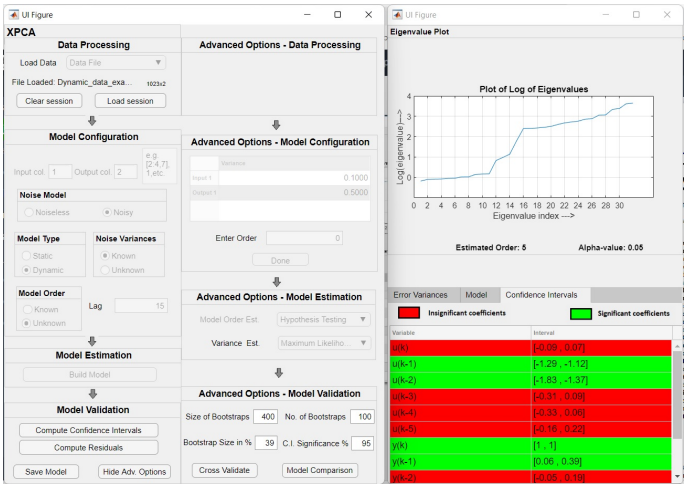

Fig. 2. An image of the Toolbox

A. Data Processing Panel.

The user loads Data in this panel as a matrix or as an IDDATA object. The user can also load a variable from the base workspace in Matlab. The `Load session` button allows user to load either a previous session or any stored session in the past. The `Clear session` button allows the user to reset the Application if they want to make changes in the previous panel.

B. Model Configuration Panel.

The user must choose the type of model they want to build as:

- Noise-free or Noisy
- Static or Dynamic
- If Model is Noisy only then the user may choose between Known or Unknown Variance
- If the user chooses Known variance the user must enter the Variance for each variable
- If the user chooses Dynamic model only then the Lag option becomes visible to the user and the user must choose a big enough value as the Lag.

C. Model Estimation Panel.

In this panel the user presses the `Estimate model order` button, `Estimate Noise Variances` button and `Build Model` button. Each of the button are only active depending on the data the user has fed into the App, i.e. if the order or the variance is known, the corresponding button is inactive. All these results are displayed in another window in multiple tabs as shown in the figure 2.

D. Model Validation Panel.

In this Panel the user may calculate the Confidence interval for each coefficient of the model to determine which of them are significant. This result also gets displayed in the second window.

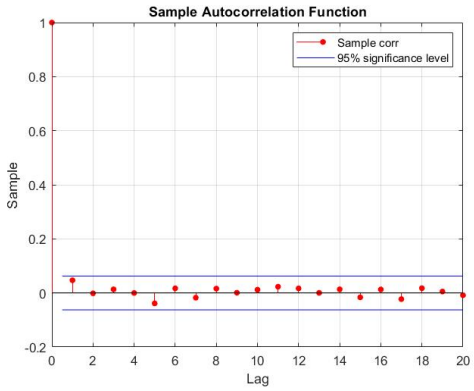

Fig. 3. EIV Kalman Filter for model validation

Kalman filter validates the model by plotting Auto-Correlation Function(ACF) of the residuals obtained from the Kalman filter, thus indicating that the residuals are just uncorrelated white noise.

The other two buttons `Save Model` button saves model in a standard format so that it can be used on other platforms in Matlab. The `Show/Hide Advanced Options` button are designed for the advanced user and has the following modules:

a. Advanced Options -Model Estimation.

The user may choose a method for Order Estimation, currently the Toolbox only has Hypothesis testing and Known Order features, however more can be added in the future.

The toolbox also contains the feature to decide the Variance Estimation method, currently their are only two methods "Keller solution" and "Maximum likelihood estimation" , the latter being the default method.

b. Advanced Options -Model Validation Panel.

Finally this panel allows the user to decide the size and number of bootstraps for the confidence interval estimation.

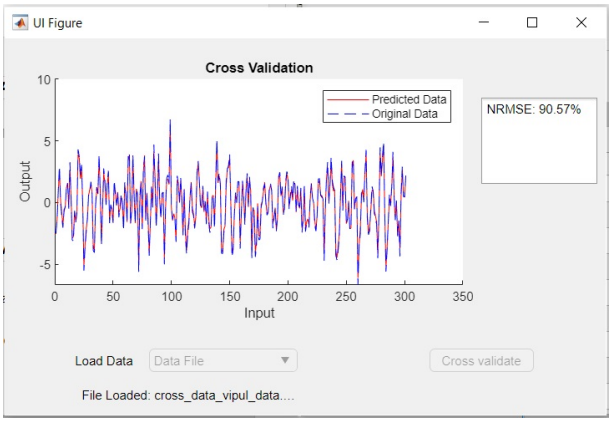

Fig. 4. Cross Validation feature

The `Cross Validate` button plots comparison between Predicted and Original data with Goodness of fit test as shown in figure 4.

IV Simulation Case-Studies

A. Dynamic System case.

Simulating the equation,

$$y^*[k] = -0.2y^*[k-1] - 0.6y^*[k-5] + 1.2u^*[k-1] + 1.6u^*[k-2] \quad \textbf{(20)}$$

for which the equation order η is 5. For data generation purposes, the input, u*, is chosen to be a full-length, full-band (white-noise-like) PRBS signal of length N = 1023 observations. The noise-free output, y*, is generated as per (eq 20). Measurements $y[k]$ and $u[k]$ are generated by adding zero-mean Gaussian white noise to the noise-free inputs with $\sigma^2_{e_u} = 0.1214$ and $\sigma^2_{e_y} = 0.4351$. The lagged matrix here would be constructed by setting $L = 15$. We also keep the

variance unknown we get, order $\eta = 5$, variance $\sigma^2_{e_u} = 0.1115$ and $\sigma^2_{e_y} = 0.4699$ and the estimated model is,

$$0.9426y^*[k] = -0.2327y^*[k-1] - 0.0739y^*[k-2]$$
$$-0.0143y^*[k-3] - 0.0110y^*[k-4] - 0.5302y^*[k-5] + 0.0037u^*[k]$$
$$+1.1583u^*[k-1] + 1.5501u^*[k-2] + 0.1431u^*[k-3] + 0.1495u^*[k-4]$$
$$- 0.0187u^*[k-5] \quad \textbf{(21)}$$

It can be observed from eq 21 that the estimated parameters are in close agreement with true values in eq 20.

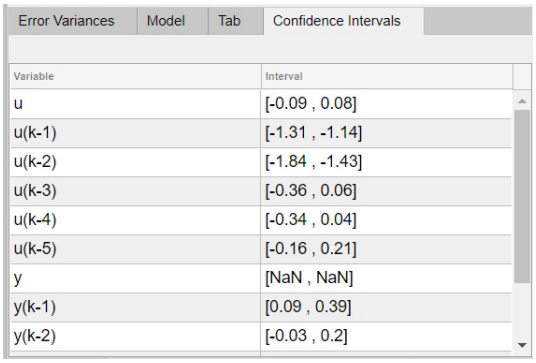

Fig. 5. snippet of Confidence Interval in the Toolbox

One can observe that coefficient of u^*, $u^*[k-3]$, $u^*[k-4]$, $u^*[k-5]$ are not significant, similarly $y^*[k-2]$, $y^*[k-3]$, $y^*[k-4]$ have insignificant coefficient. Hence, the toolbox is able to filter out the insignificant coefficients in the model successfully.

B. Static System case.

Simulating the equations,

$$F3 = F1 + F2 \quad \textbf{(22a)}$$
$$F4 = F3 \quad \textbf{(22b)}$$
$$F5 = F4 - F2 \quad \textbf{(22c)}$$

Here, the number of constraints is 3. For data generation purposes, F1 has mean 25 and standard deviation 30 and F2 has mean 12 and standard deviation 1, both are absolute value and the signal is of length 1000. The error variance introduced in F1, F2, F3, F4 and F5 are 0.1, 0.2, 0.3, 0.2, 0.1 respectively. Applying Static model option in the toolbox we get 3 constraints, error variance as 0.1061, 0.1911, 0.3226, 0.2098 and 0.1121 respectively and we get the model

$$1.51F1 - 0.01F2 + 0.97F3 - 0.97F4 - 1.51F5 = 0 \quad \textbf{(23a)}$$
$$-0.84F1 - 1.64F2 + 0.64F3 + 0.99F4 - 0.79F5 = 0 \quad \textbf{(23b)}$$
$$-1.51F1 + 0.97F3 - 0.97F4 + 1.51F5 = 0 \quad \textbf{(23c)}$$

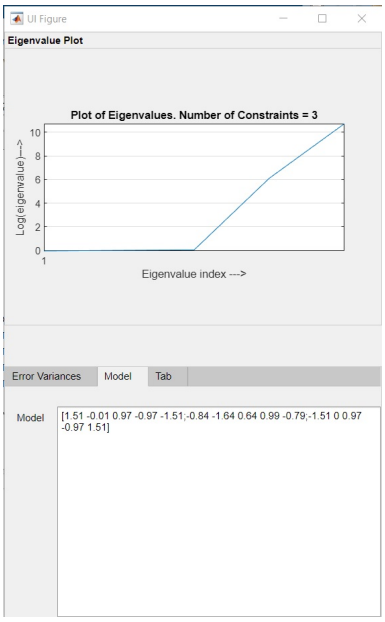

Fig. 6. Static case Toolbox snippet

One can observe and calculate that the equations are fairly close to the real model.

V Conclusions

In this paper, we introduced a new SYSID toolbox on the Matlab®platform, used for creating data-driven models especially EIV data for both Static and Dynamic systems. The Toolbox is designed for both a novice and Expert user. The App contains Model estimation and validation with multiple user options. The backend design of the Application is "Strategic pattern" which makes it easy for any new developer to add new features and methods to the App. The future of the Application would have more features and methods for Order and Model estimation, with multiple model comparisons at a time.

References

1. Narasimhan, S.; Shah, S. Model identification and error covariance matrix estimation from noisy data using PCA *Control Engineering Practice.* 2008, 16, 146155.
2. D. Maurya, A. K. Tangirala, and S. Narasimhan, “Identification of linear dynamic systems using dynamic iterative principal component analysis,” *IFAC-PapersOnLine.* , vol. 49, no. 7, pp. 1014–1019, 2016.
3. Tibshirani,R.,Efron,B.(1994).*An Introduction to the Bootstrap.*United Kingdom:CRC Press.
4. Vipul Mann, Deepak Maurya, Arun K. Tangirala, and Shankar Narasimhan *Industrial Engineering Chemistry Research* 2020 59 (5), 1953-1965 DOI: 10.1021/acs.iecr.9b04561
5. L. Ljung, *MATLAB System Identification Toolbox: User's Guide Version 8.* The Mathworks, 2012
6. B. Ninness, A. Wills, and A. Mills, “Unit: A freely available system identification toolbox,” *Control Engineering Practice*, vol. 21, no. 5, pp. 631–644, 2013.
7. J. L. Guzman, D. Rivera, S. Dormido, and M. Berenguel, “An interac- ´ tive software tool for system identification,” *Advances in Engineering Software*, vol. 45, no. 1, pp. 115–123, 2012.
8. Steve Holzner. 2006. *Design patterns for dummies®* John Wiley Sons, Inc., USA.