Aktorik und Sensorik mit intelligenten Materialsystemen 4

Computer Lecture: Numerical Linearization of a SMA-Spring Actuator in Matlab

Gianluca Rizzello

gianluca.rizzello@mmsl.uni-saarland.de

Saarland University

Computer Lecture Overview

- GOALS:

  – Obtain a linearized model describing the response of a SMA-spring actuator around an equilibrium state via a numerical identification method

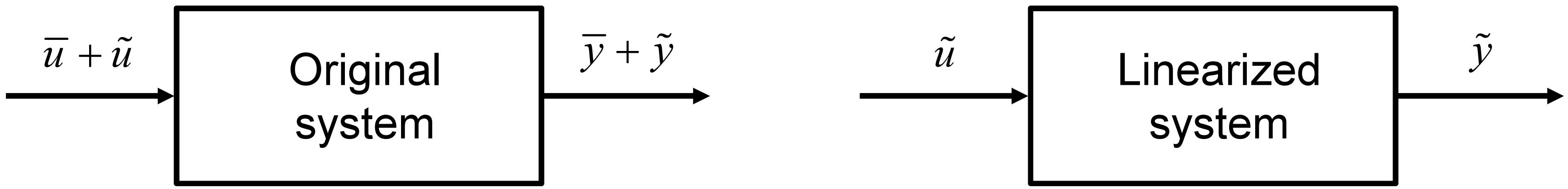
  – Identification of Joule heating – displacement linearized dynamics

  – Identification of environmental temperature – displacement linearized dynamics

  – Identification of load force – displacement linearized dynamics

System Identification Toolbox

- Numerical linearization: obtain a linearized model describing relationship between input deviations an output deviations from an equilibrium point



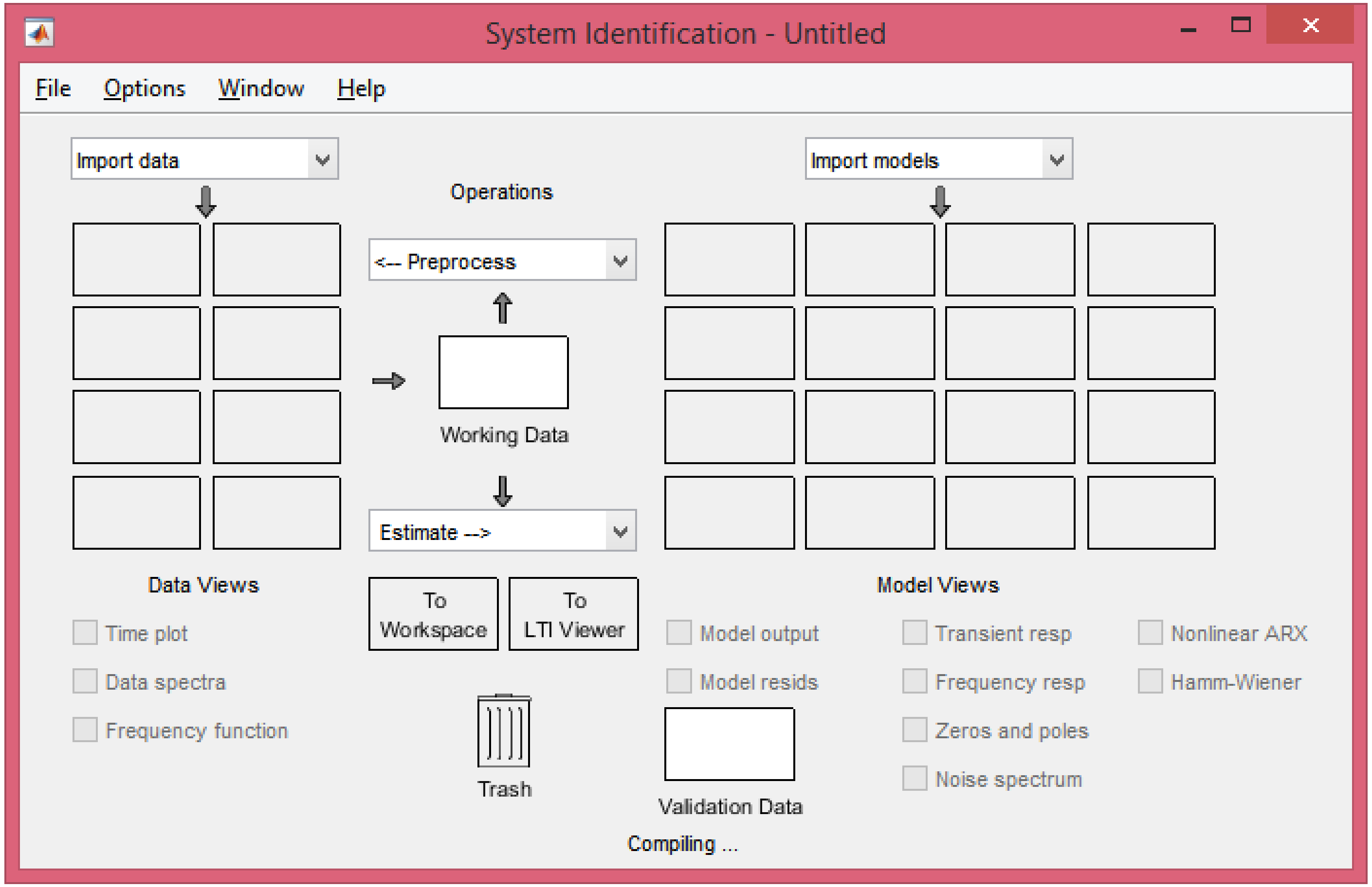
- This operation can be performed in Matlab via System Identification Toolbox

- To open the System Identification Toolbox GUI, type the following command in Matlab:

```
systemIdentification
```

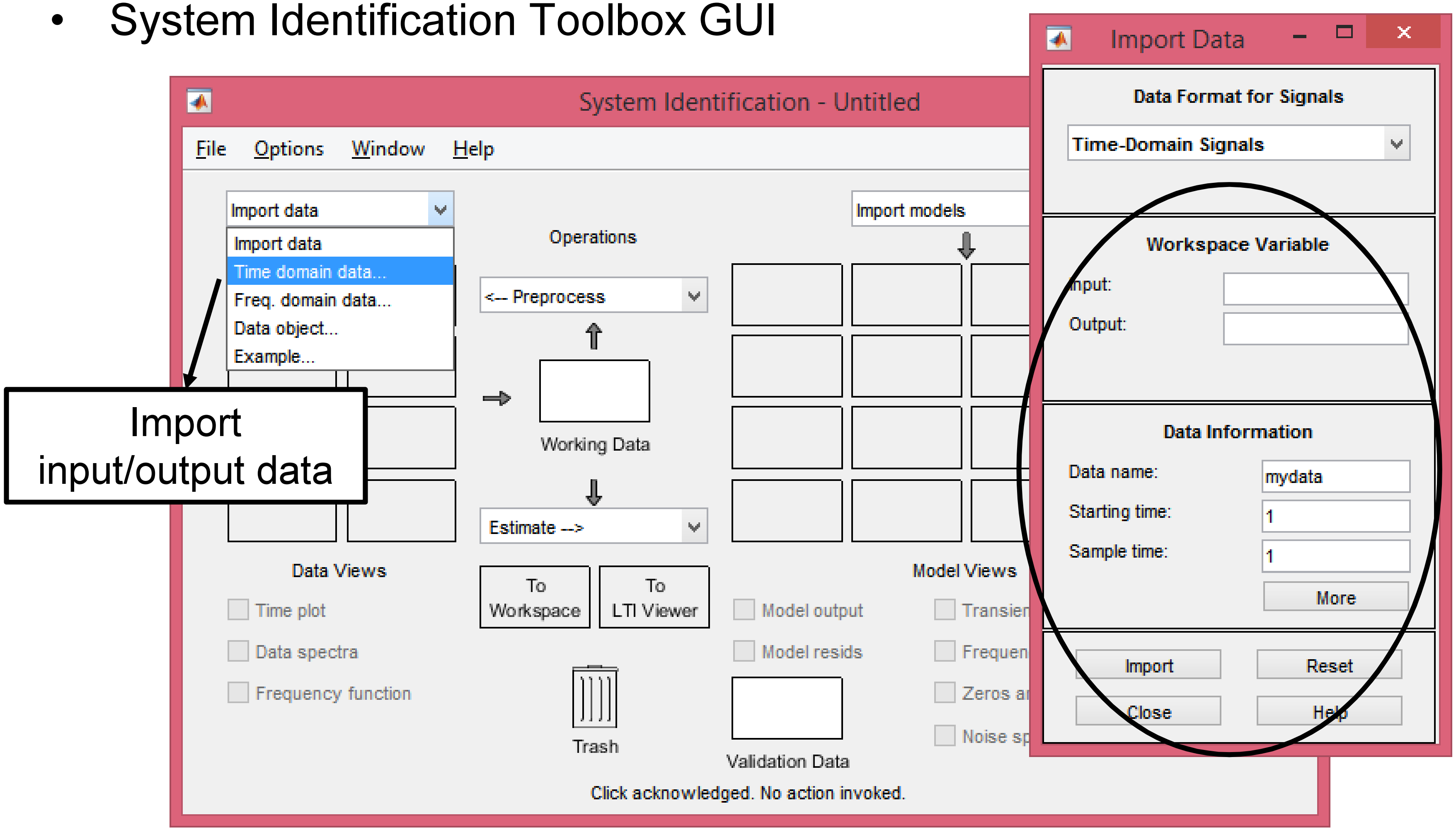
System Identification Toolbox

- System Identification Toolbox GUI



System Identification Toolbox

- System Identification Toolbox GUI

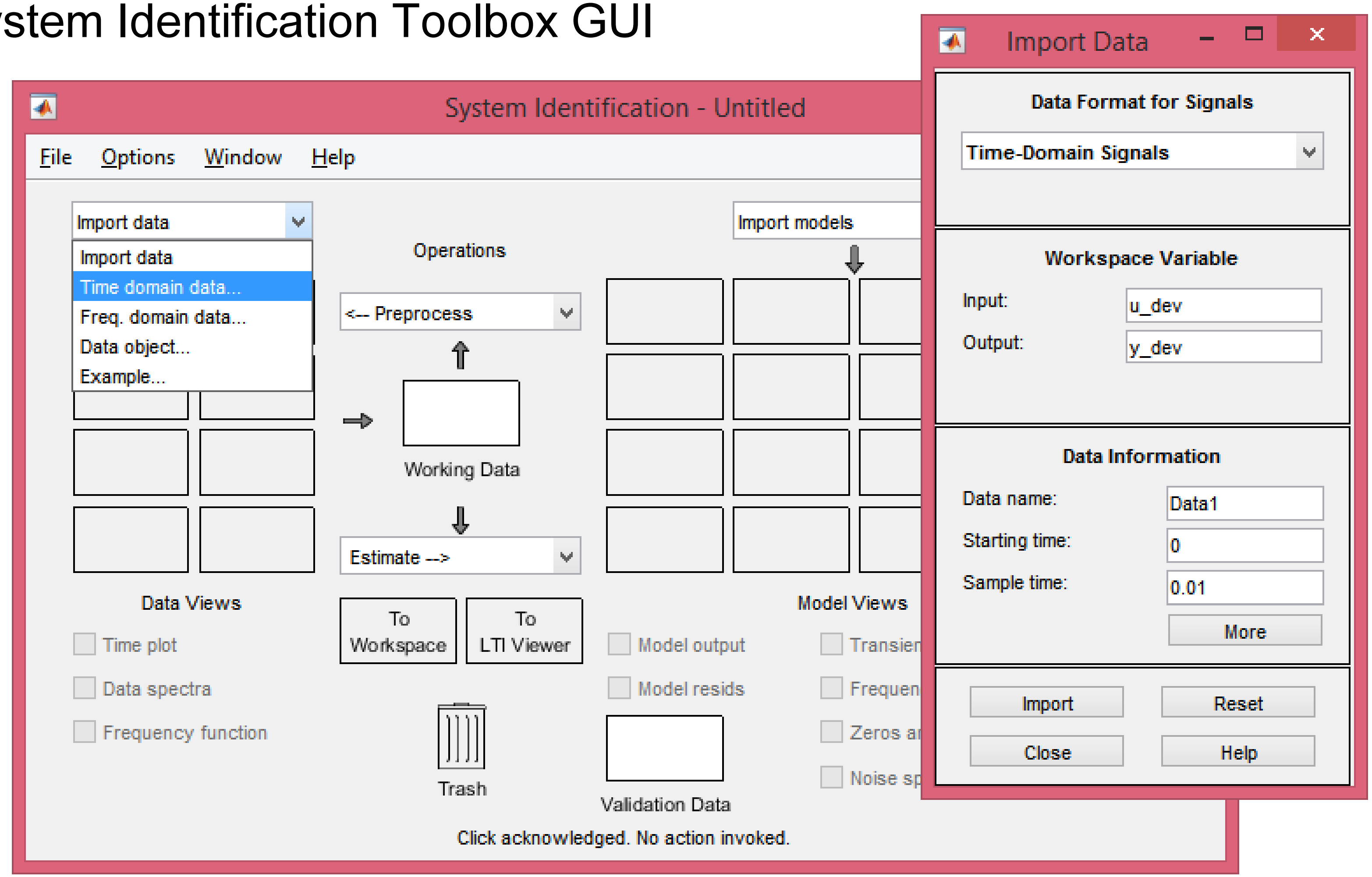


Specify here all the required information, then select ‘Import’

UNIVERSITÄT DES SAARLANDES

System Identification Toolbox

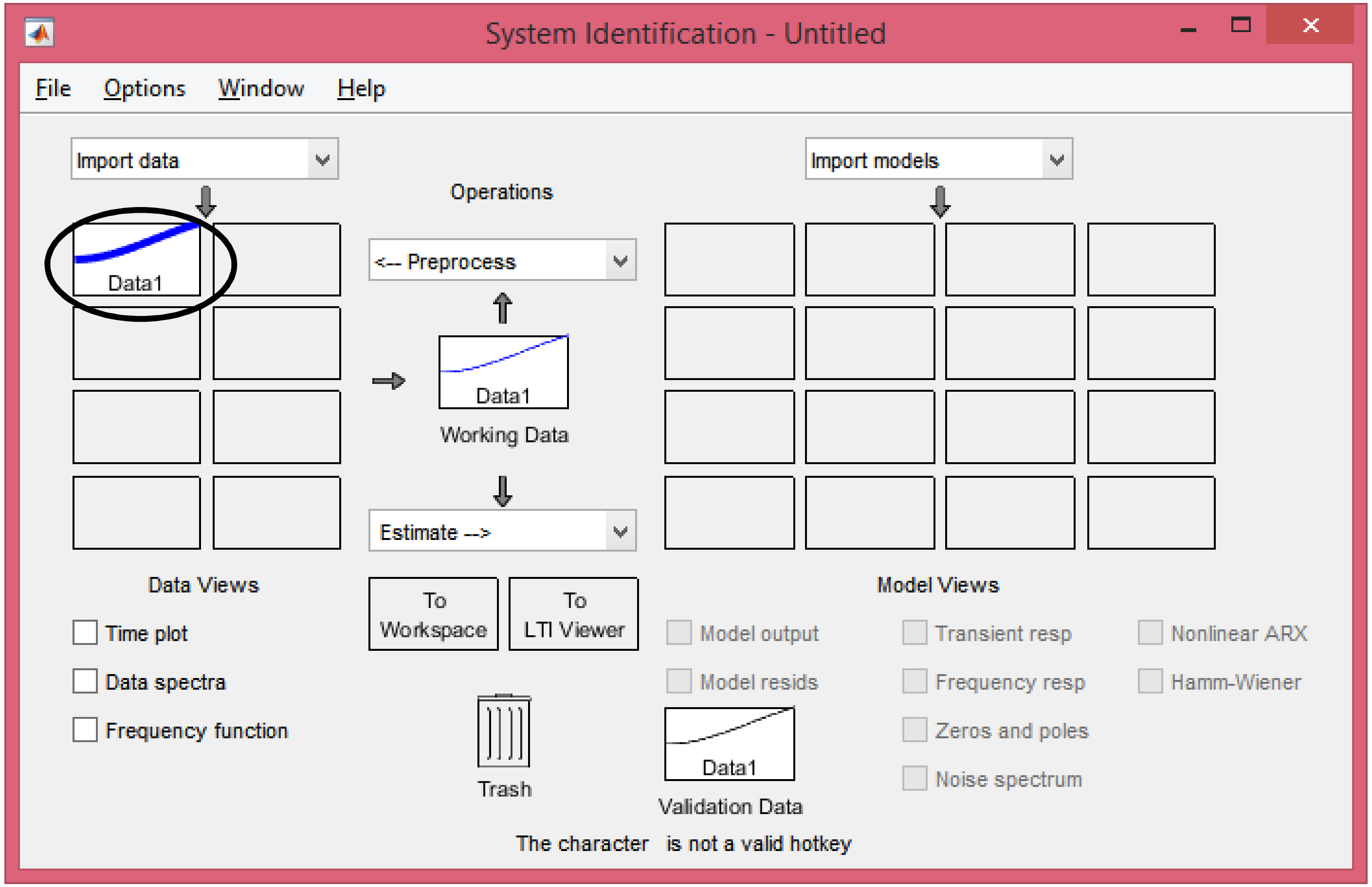
- System Identification Toolbox GUI



Input and output must me column vectors with same size, sampled with uniform sampling time

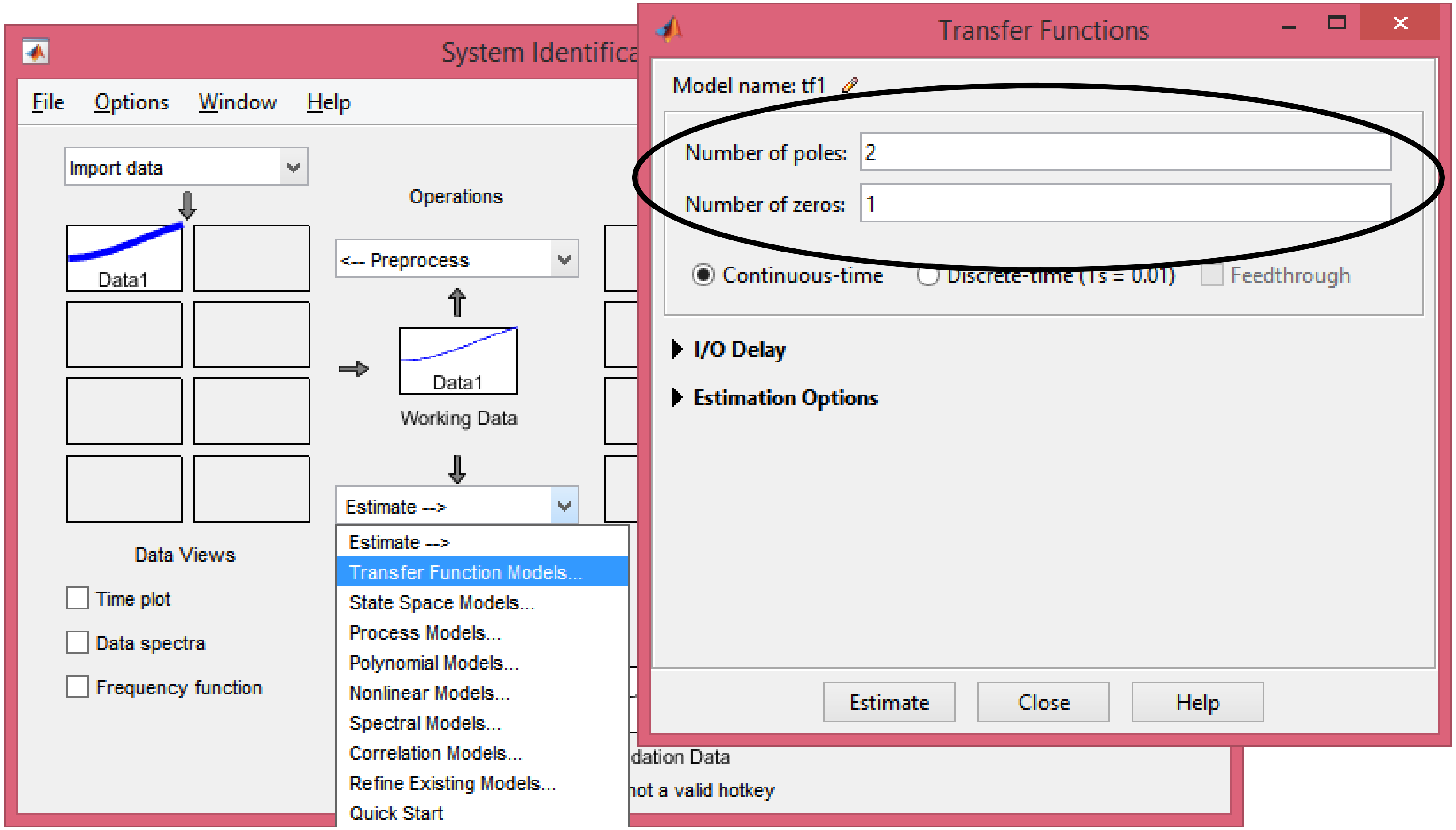
System Identification Toolbox

- System Identification Toolbox GUI



Data have now been imported

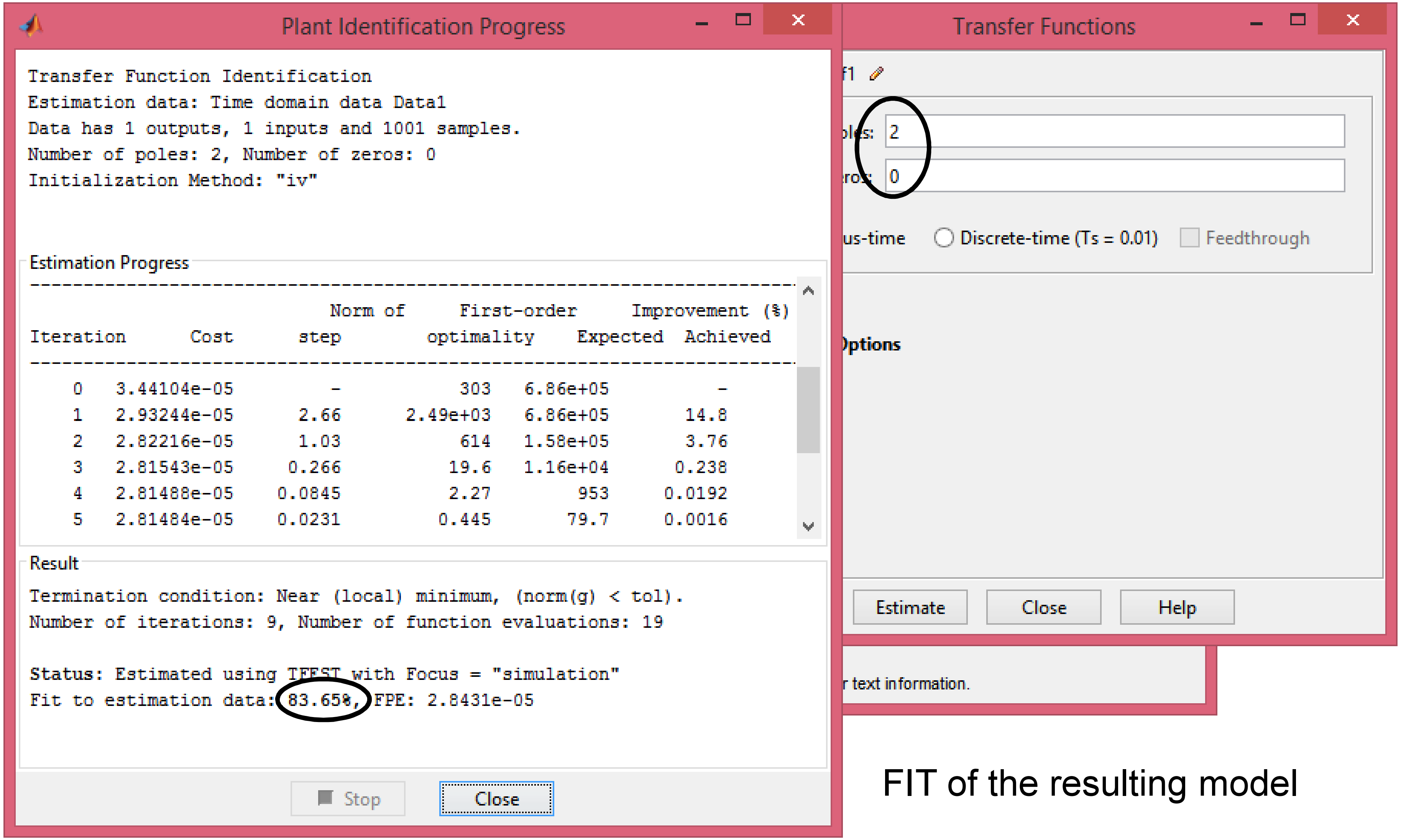
- System Identification Toolbox GUI



From the new window, choose number of zeros and poles for the optimal model, then select 'Estimate'

System Identification Toolbox

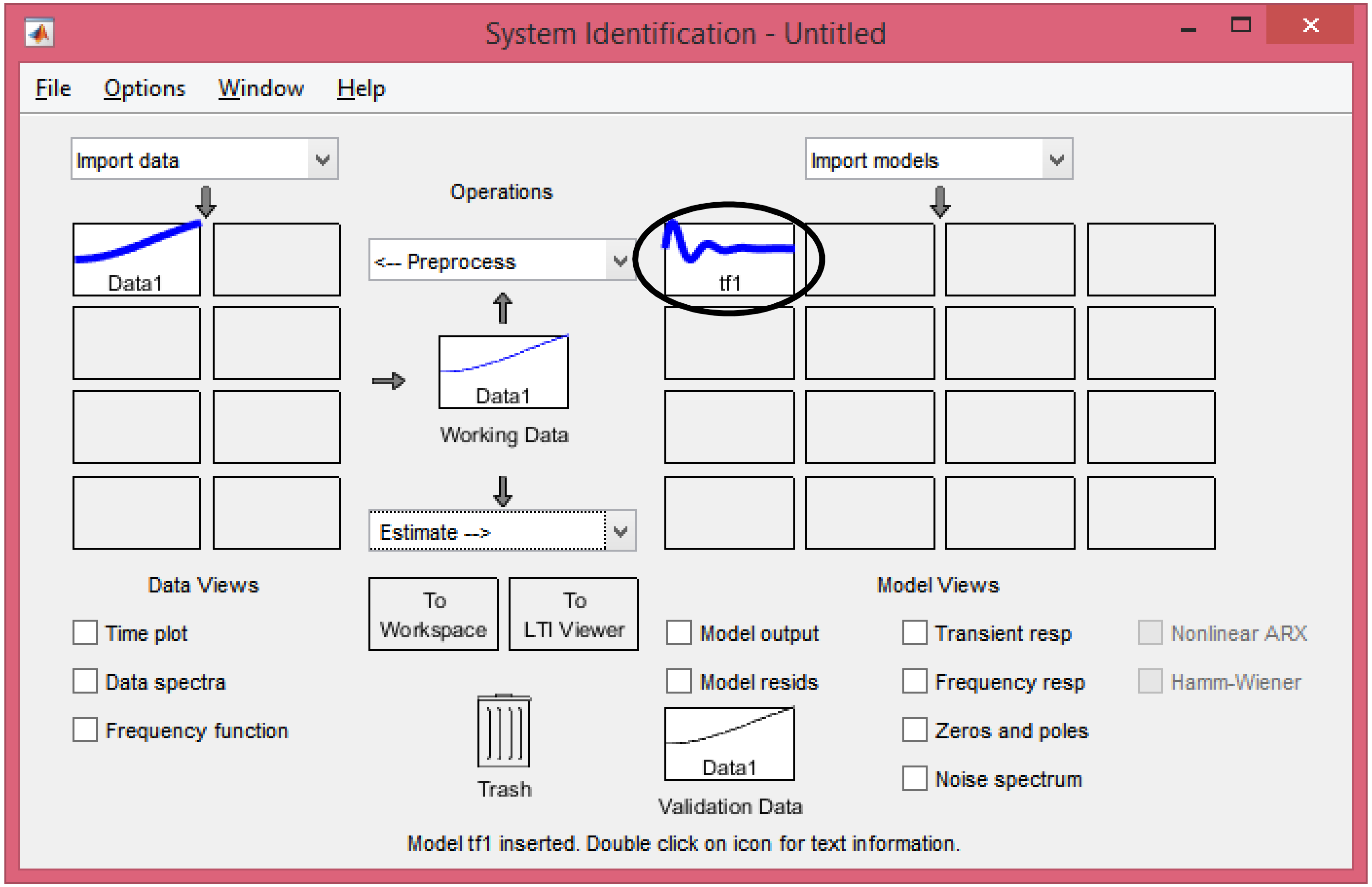
- System Identification Toolbox GUI



FIT of the resulting model

System Identification Toolbox

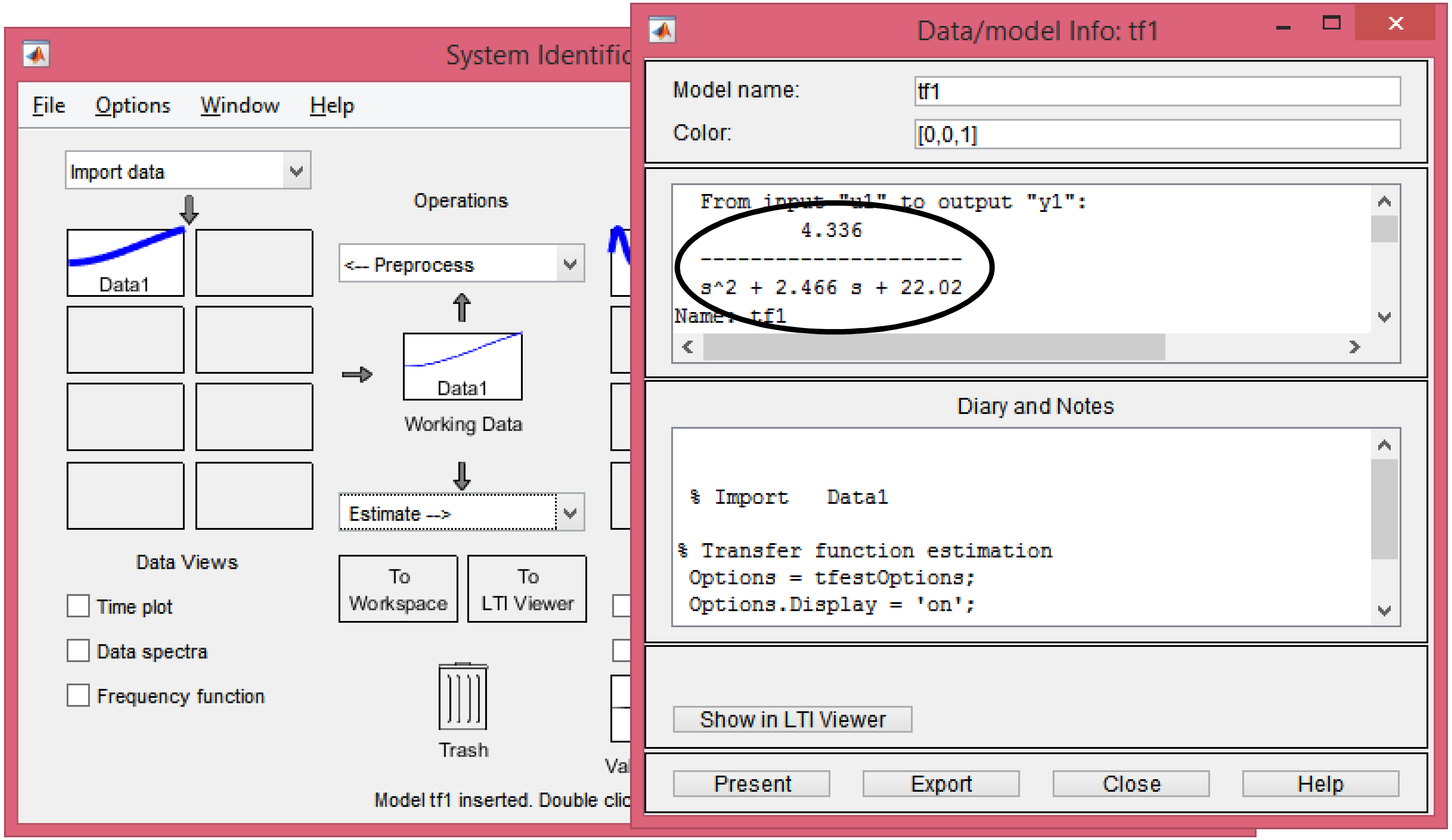
- System Identification Toolbox GUI



The resulting model is saved here

System Identification Toolbox

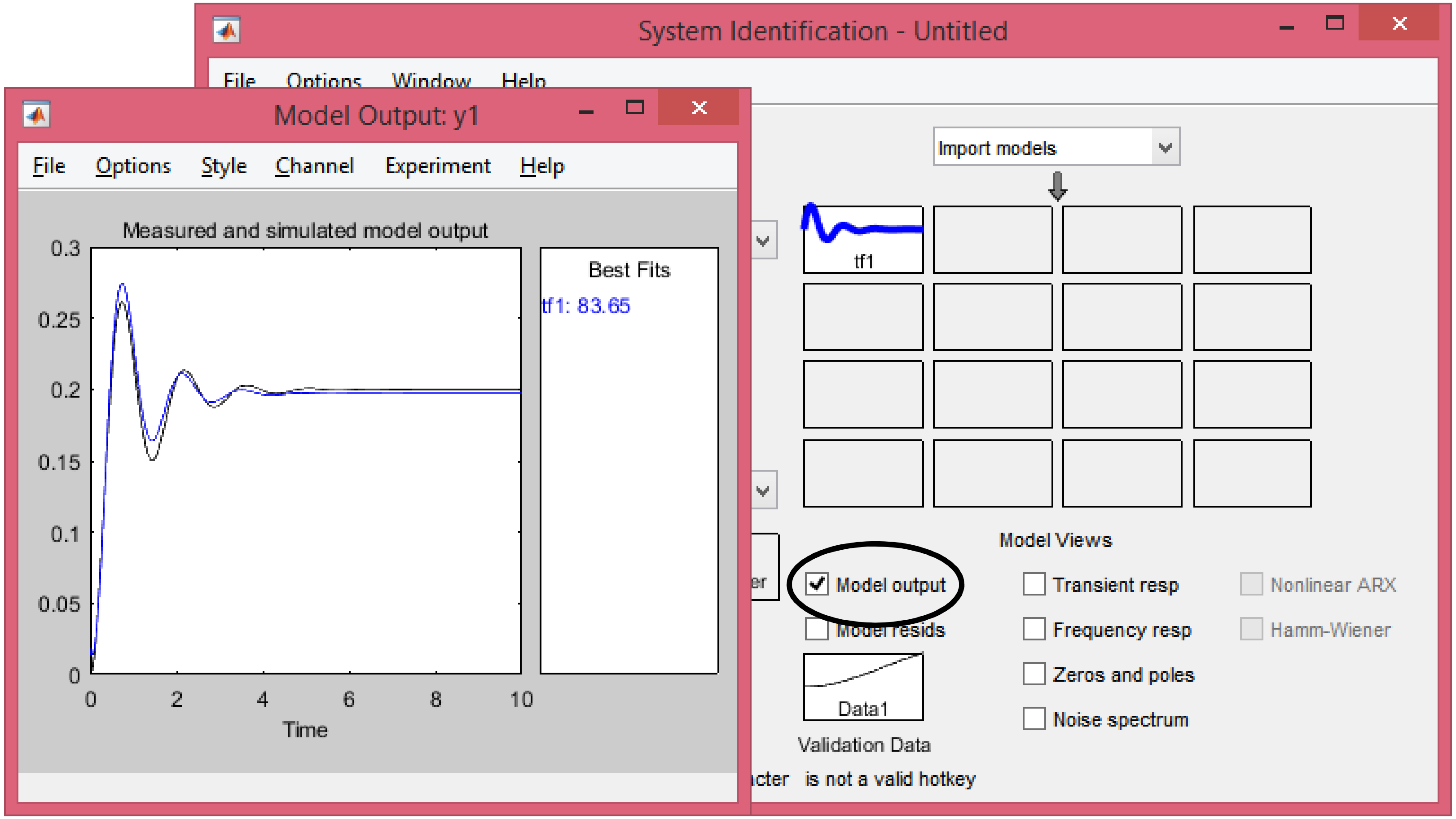
- System Identification Toolbox GUI



Right-click to show the identified transfer function model, which can be eventually exported in Matlab as a transfer function object

System Identification Toolbox

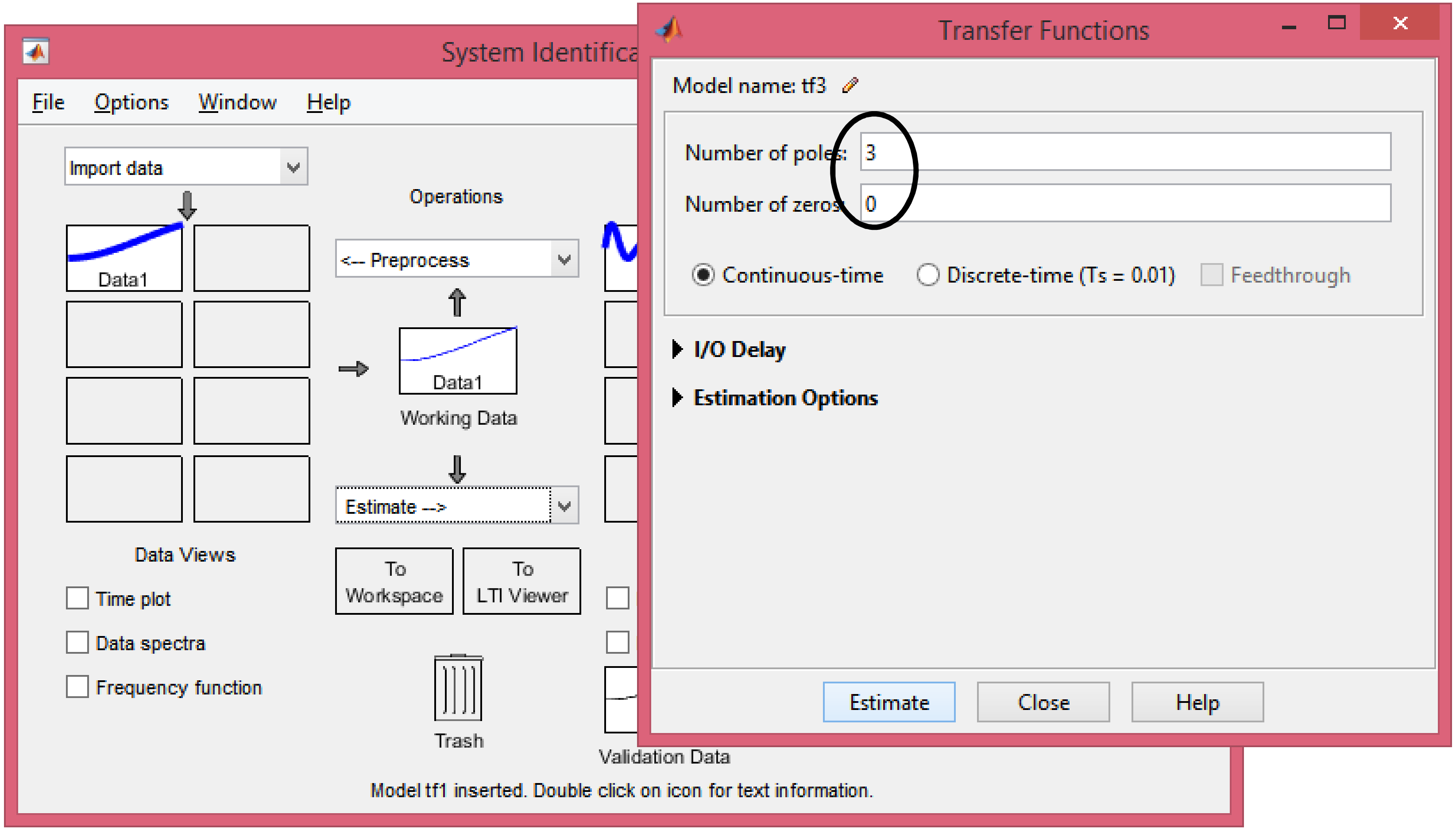
- System Identification Toolbox GUI



To show time-domain response, select ‘Model output’

System Identification Toolbox

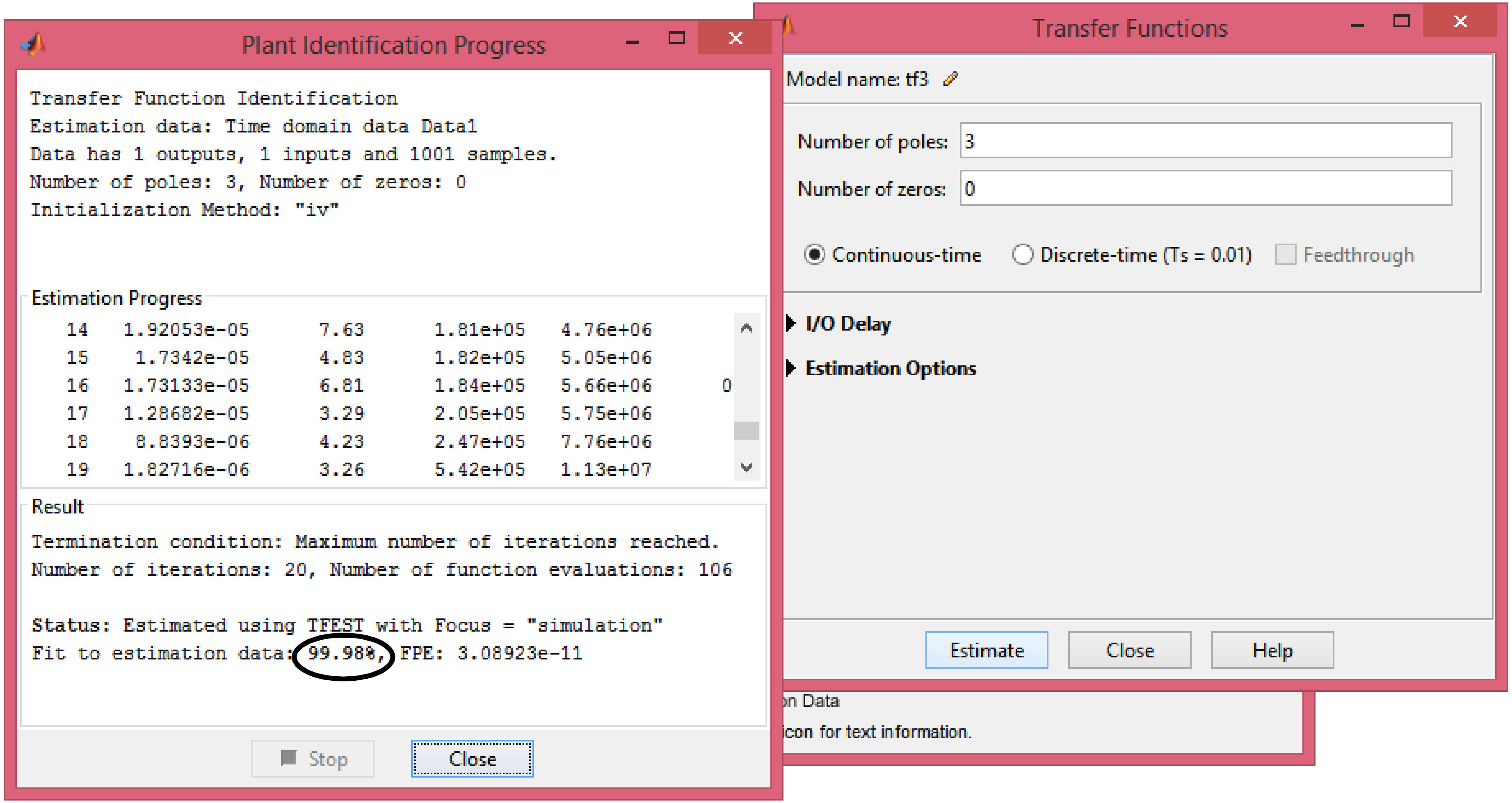
- System Identification Toolbox GUI



To improve the accuracy, we can increase the complexity of the model, e.g., by increasing the number of zeros and poles

System Identification Toolbox

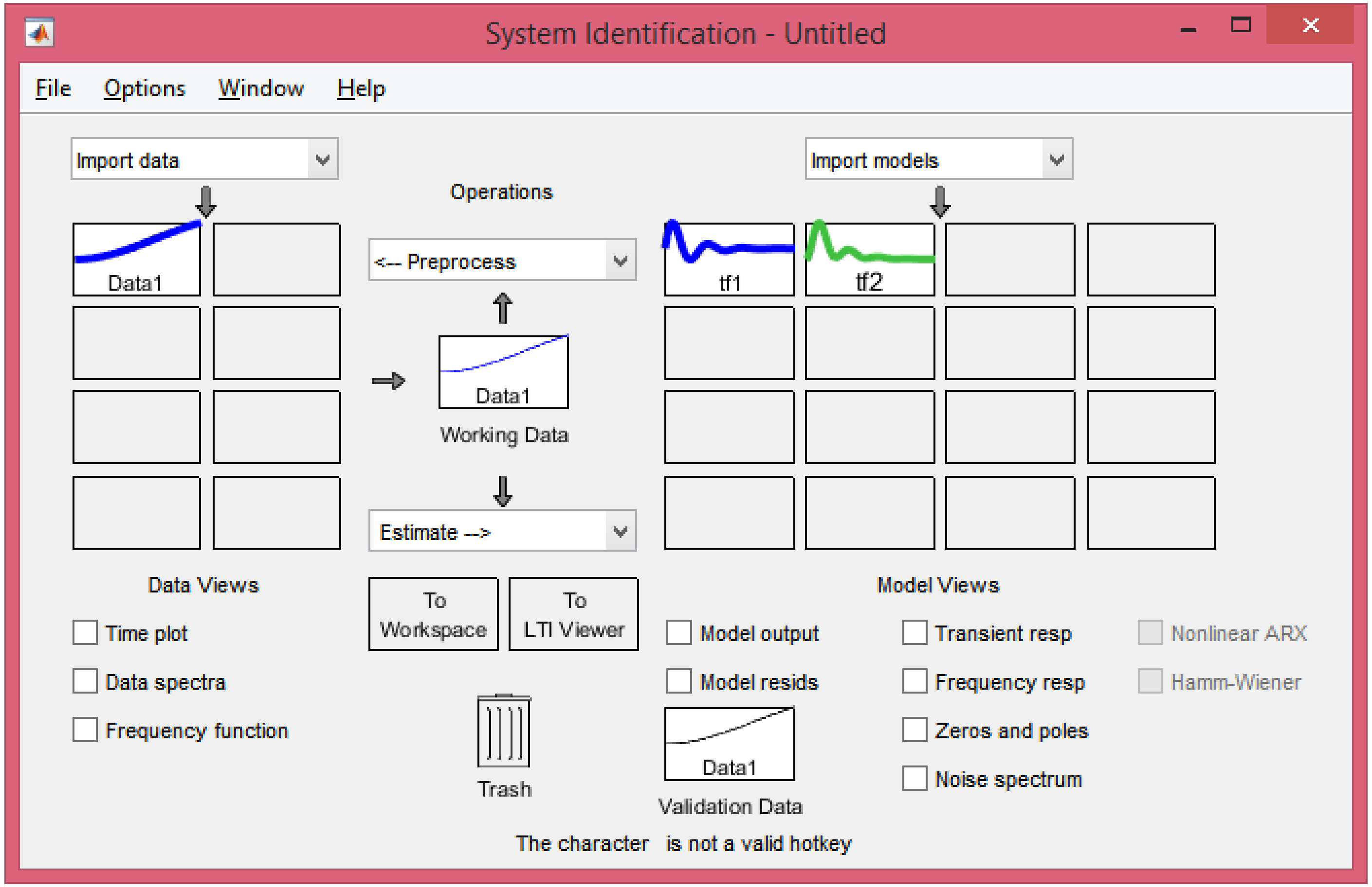
- System Identification Toolbox GUI



In this particular case, we get an almost perfect FIT

System Identification Toolbox

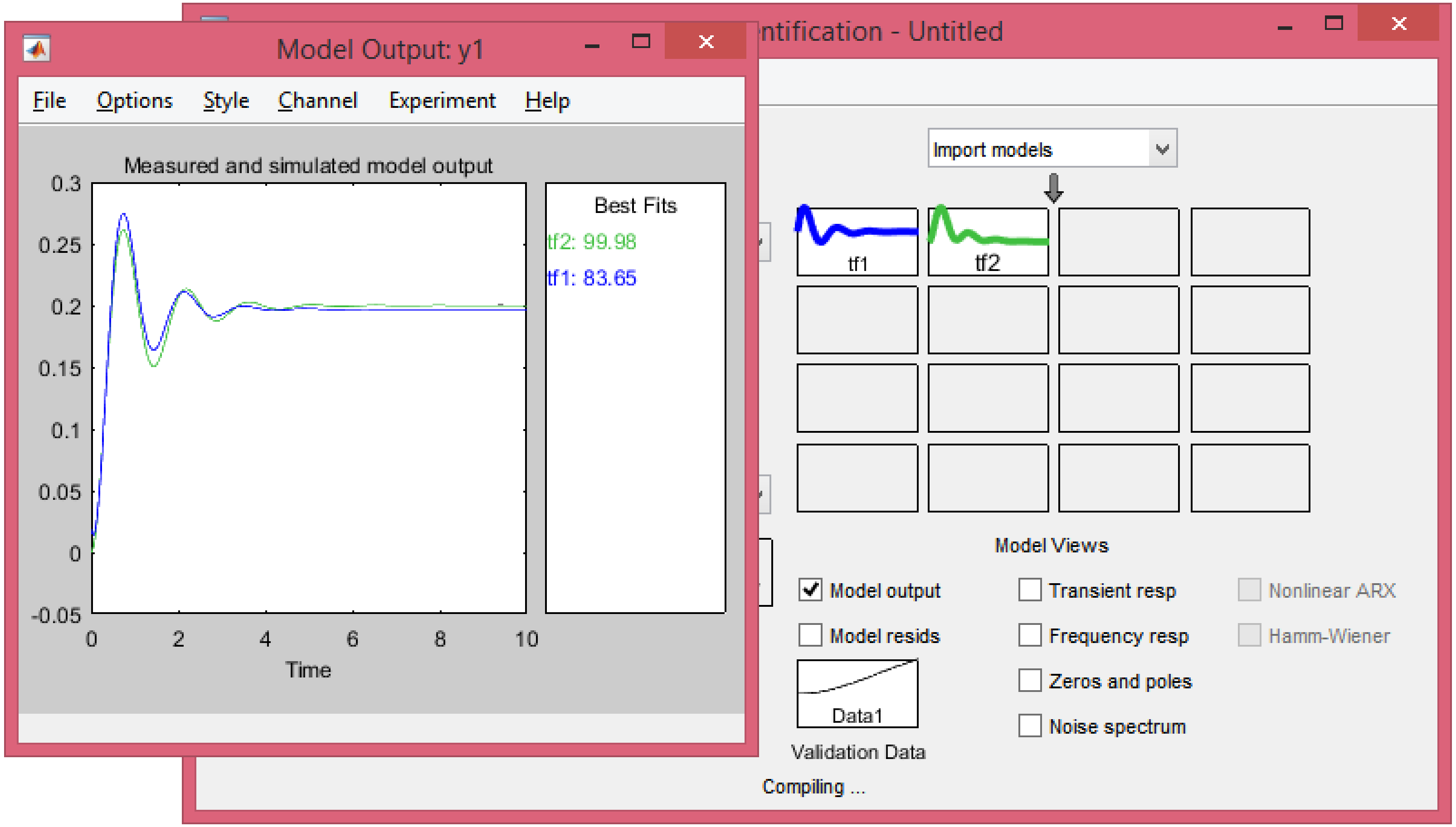
- System Identification Toolbox GUI



In this particular case, we get an almost perfect FIT

System Identification Toolbox

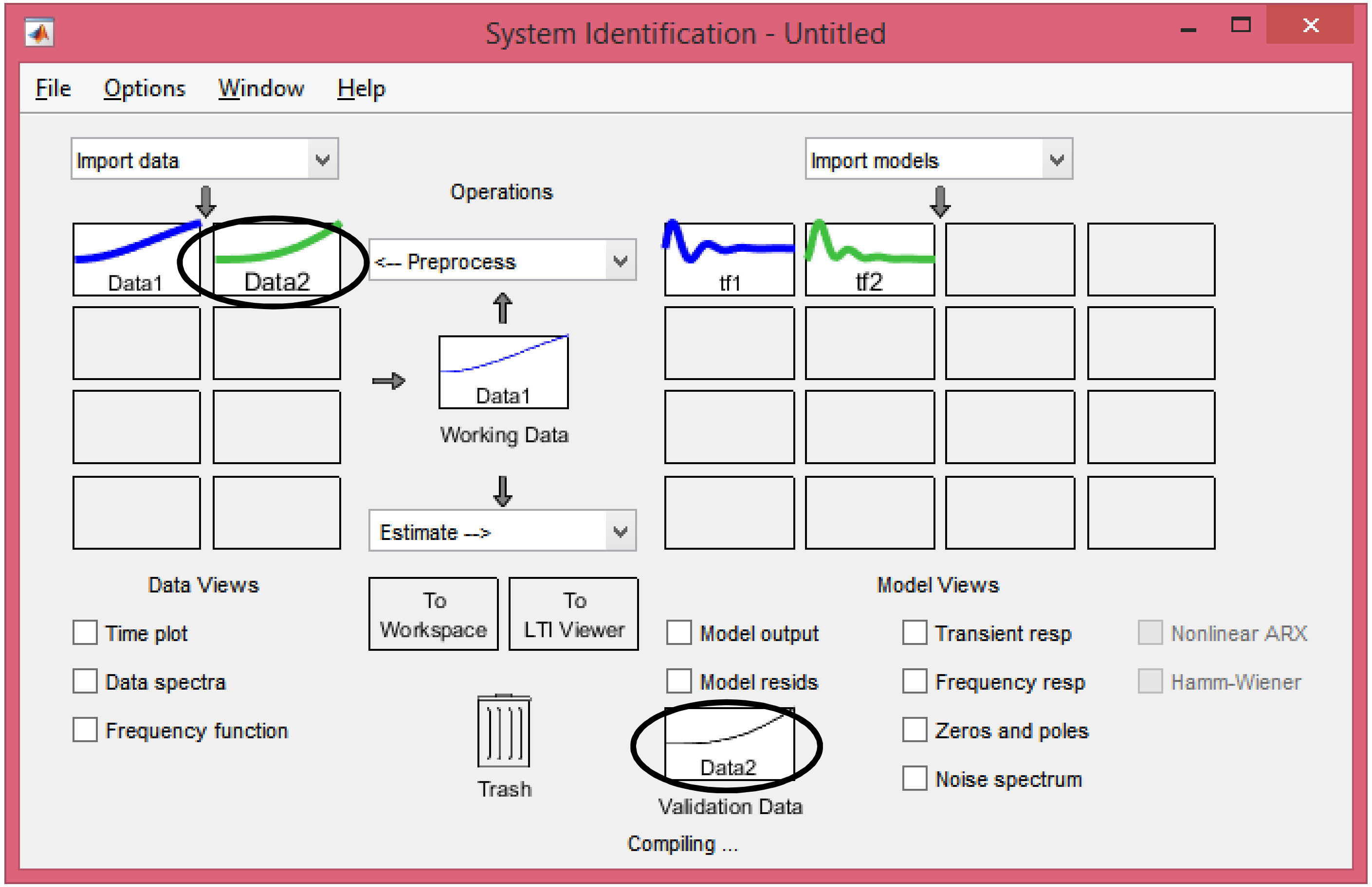
- System Identification Toolbox GUI



In this particular case, we get an almost perfect FIT

System Identification Toolbox

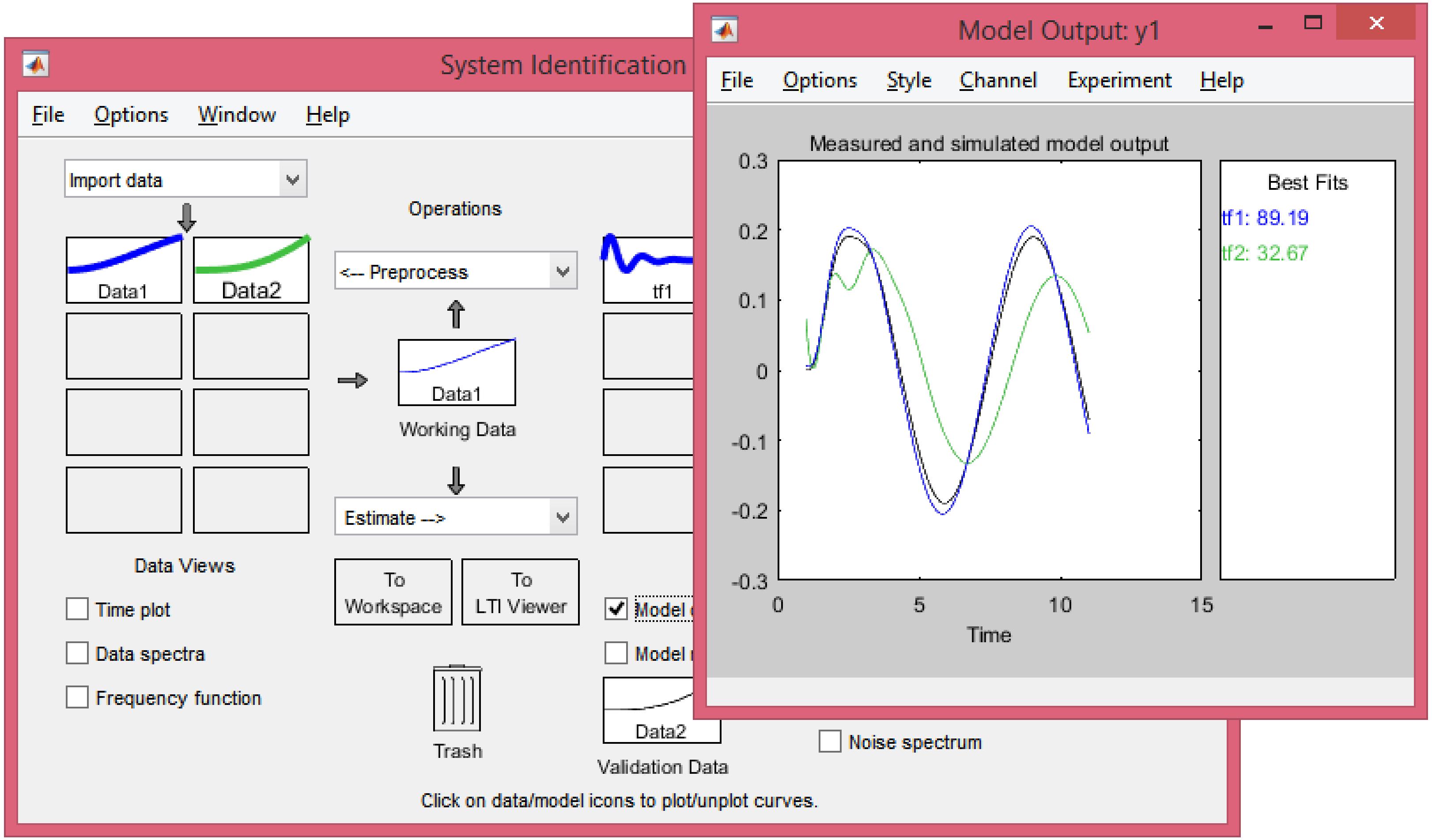
- System Identification Toolbox GUI



New data can be eventually imported and used to perform validation of the model

System Identification Toolbox

- System Identification Toolbox GUI



New data can be eventually imported and used to perform validation of the model

Transfer functions in Matlab

- To represent a transfer function object in Matlab, one can use the command

```
G = tf(num,den)
```

where `num` and `den` are arrays containing coefficients of numerator and denominator polynomials, respectively, while `G` is the transfer function object

- Example 1: to represent the following transfer function

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

we use the following command

```
G = tf([2 1],[1 4 4 2])
```

- Example 2: to represent products between factors, such as in

$$G(s) = \frac{2(s+3)}{(s^2+2s+2)(s+5)}$$

one can use the command `conv`

```
G = tf(2*[1 3],conv([1 2 2],[1 5]))
```

Transfer functions in Matlab

- Useful commands on transfer functions:
  - `step(G)` – shows step response of `G`
  - `impulse(G)` – shows impulse response of `G`
  - `lsim(G,u,t)` – computes response of `G` to input `u` with respect to time `t`
  - `pzmap(G)` – shows poles **x** and zeros **o** of `G` on the complex plane
  - `bode(G)` – shows Bode plots of `G`
  - `nyquist(G)` – shows Nyquist plot of `G`
  - `zpk(G)` – factors zeros and poles of `G`
  - `G3 = G1+G2` – computes `G3` as the sum of `G1` and `G2`, i.e., parallel connection
  - `G3 = G1*G2` – computes `G3` as the product of `G1` and `G2`, i.e., series connection
  - `T = G/(1+G)` – computes `T` as the negative feedback connection of `G`

- All of these functions are also valid for state-space objects as well

- Transfer function objects can be also converted to state-space and vice-versa
  - `S = tf2ss(G)` – converts transfer function `G` into state-space object `S`
  - `G = ss2tf(S)` – converts state-space object `S` into transfer function `G`

Assignment

- Given the SMA-spring model, consider the equilibrium state corresponding to the following constant inputs:
  - $J = 0.125\ W$
  - $T_E = 293\ K$
  - $F_L = 0\ N$

1) By using the Simulink model, bring the system to the desired equilibrium point.
2) Apply a very small variations of J, and measure the corresponding small variation of output displacement. By using the System Identification Toolbox GUI, identify a suitable transfer function model which relates the two deviations from equilibrium.
   HINT: for generating the small input deviation you can use different signals, including steps, bipolar square waves, or sinewaves.
3) Repeat the previous step by progressively increasing the amount of variation of J from the equilibrium, and compare the resulting transfer functions obtained.
4) Repeat the previous identification by only producing variations on T_E.
5) Repeat the previous identification by only producing variations on F_L.