Motor Control with Arduino

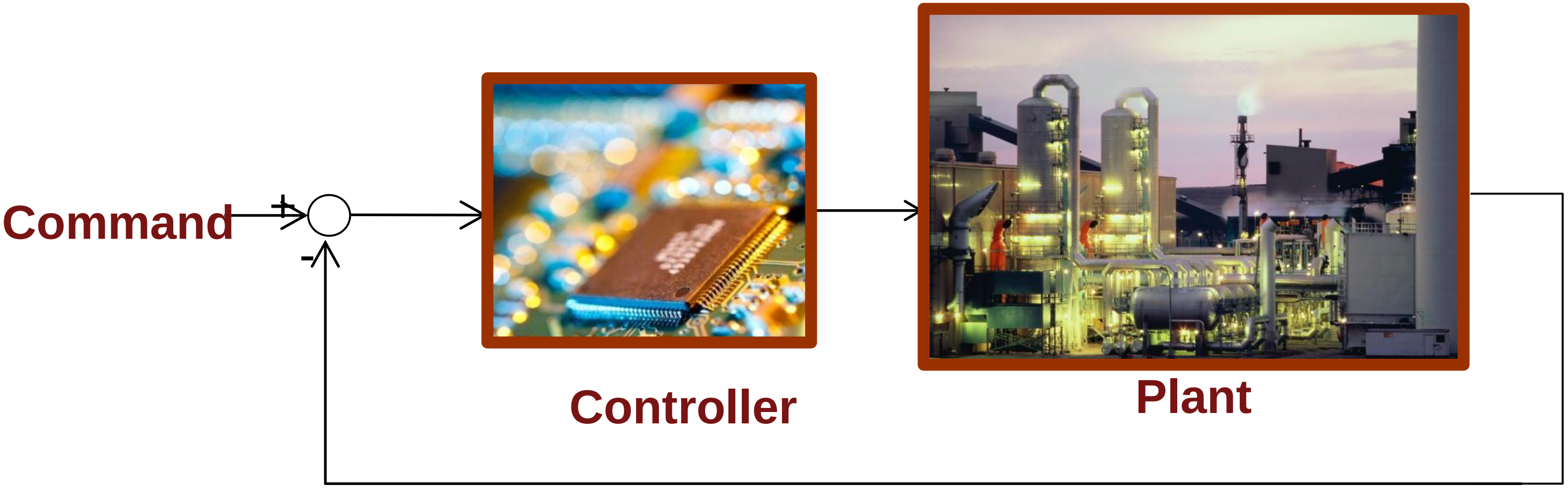
A Case Study in Data-Driven Modeling and Control Design

Toledo, 22-II-2013

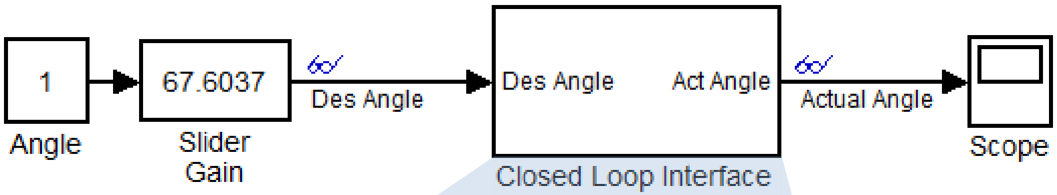
Why Data-Driven Control?

- Two approaches to modeling
  - **First-principles:** requires knowledge of math / physics of the system
  - **Data-driven:** requires measured input-output data

- Modeling from first-principles can get challenging



Demo: DC Motor Controller using Arduino Uno



1

Angle

67.6037

Slider Gain

Des Angle

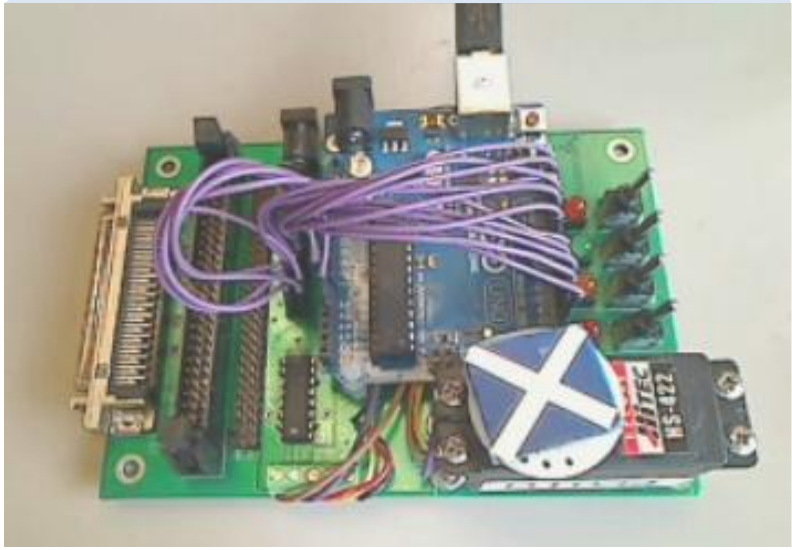
Des Angle

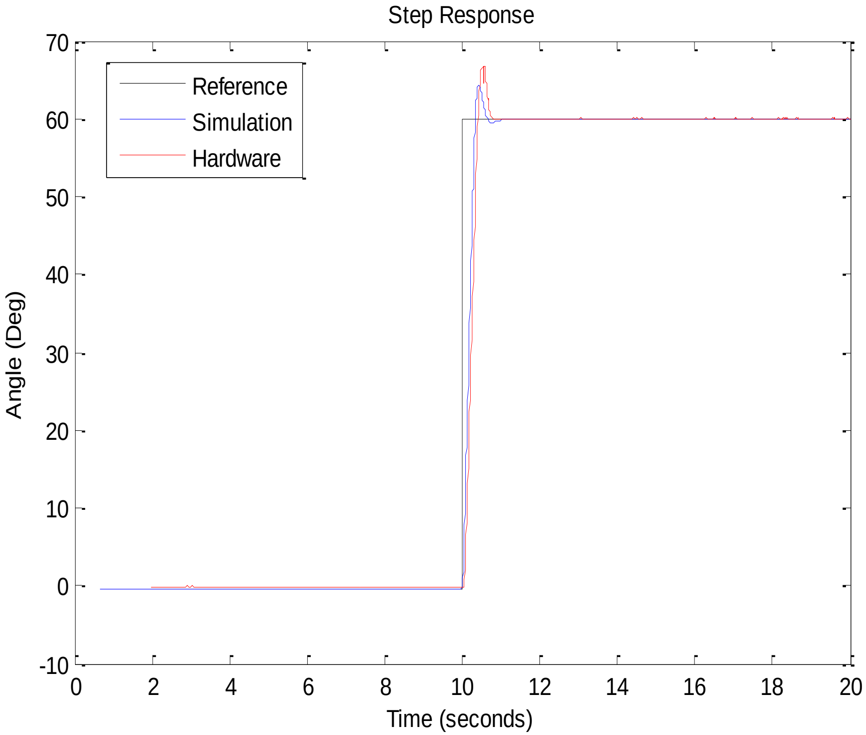
Act Angle

Closed Loop Interface

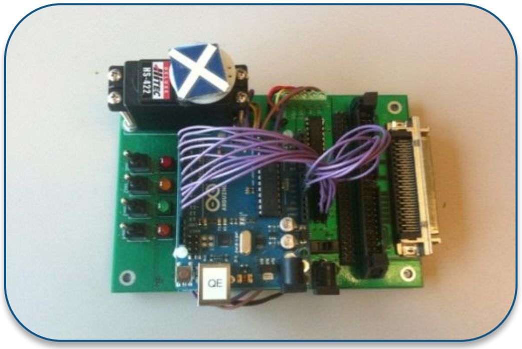
Actual Angle

Scope

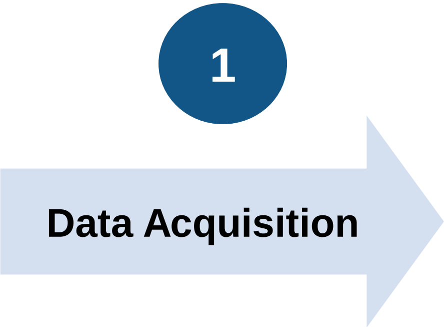




Workflow



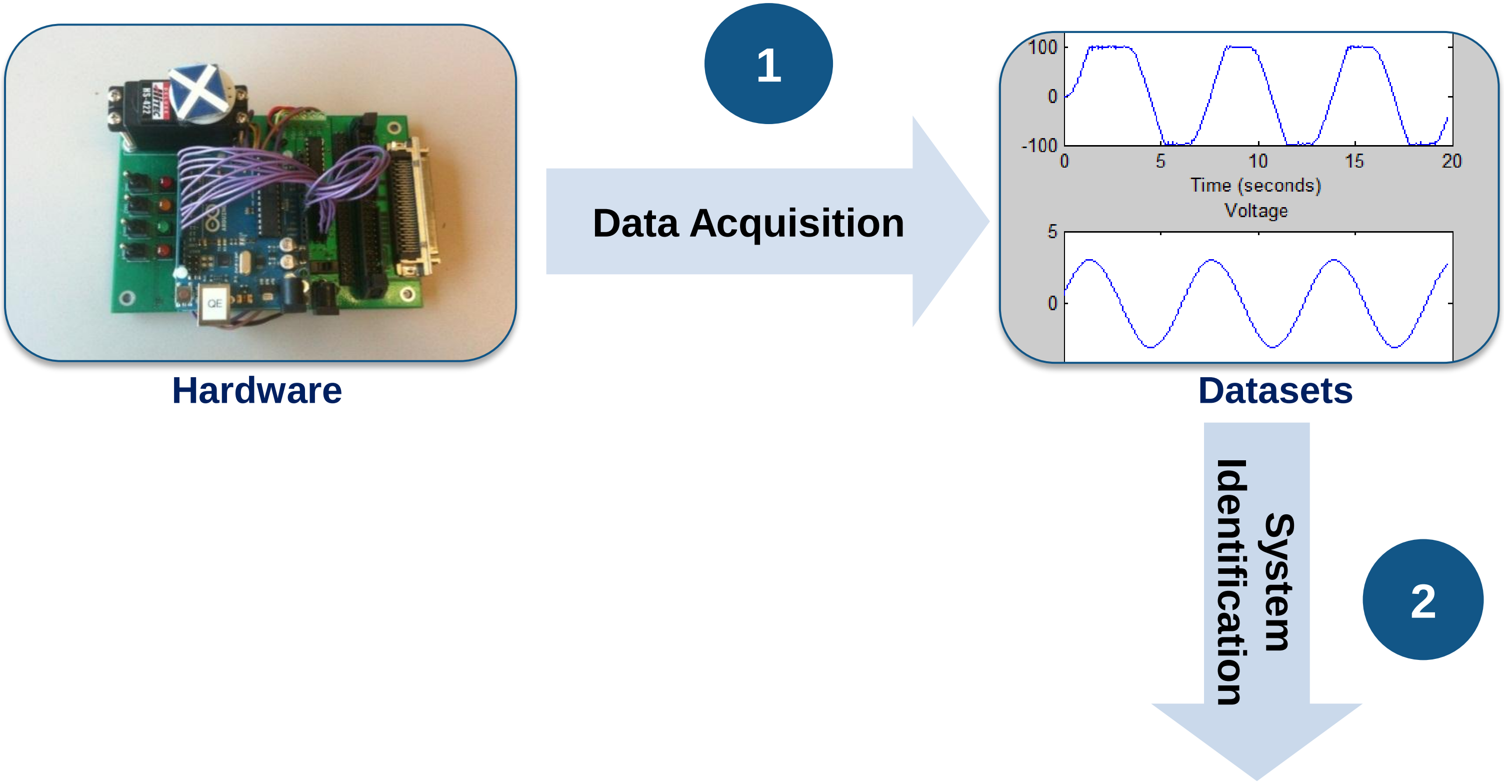
Hardware



1

Data Acquisition

Workflow



1

Data Acquisition

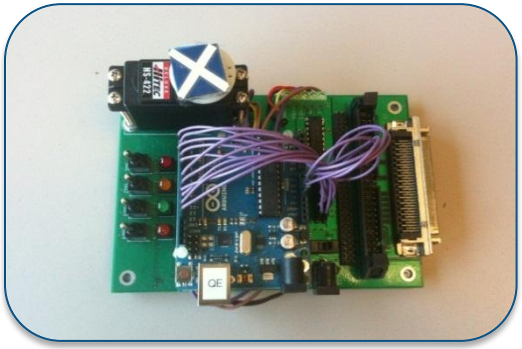
Hardware

Datasets

System Identification

2

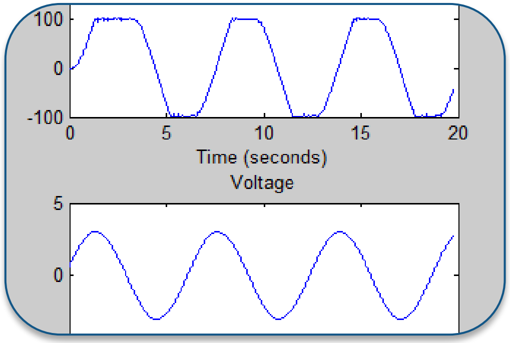
Workflow



Hardware

1

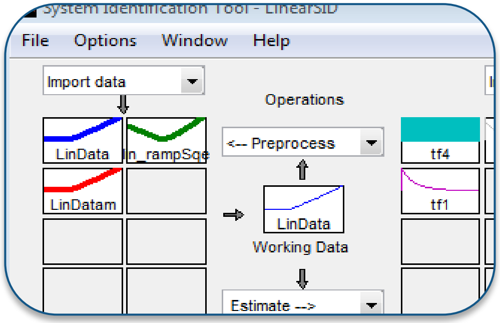
Data Acquisition



Datasets

System Identification

2

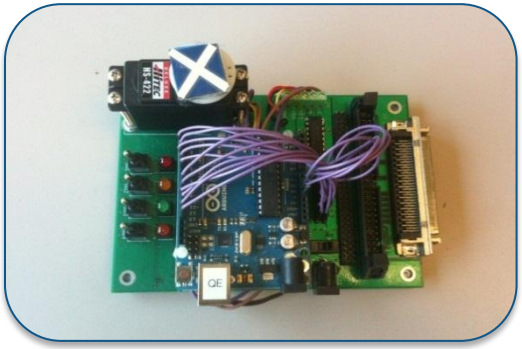


Plant Model

3

Controller Design

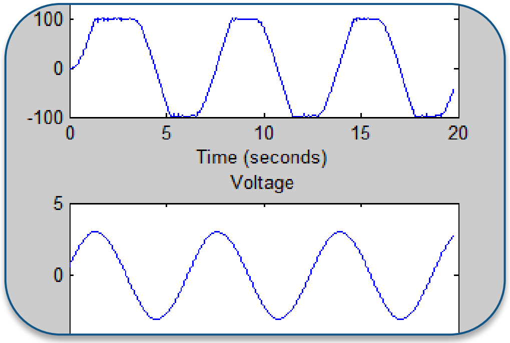
Workflow



Hardware

1

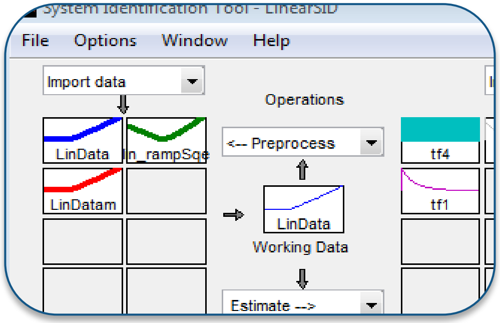
Data Acquisition



Datasets

2

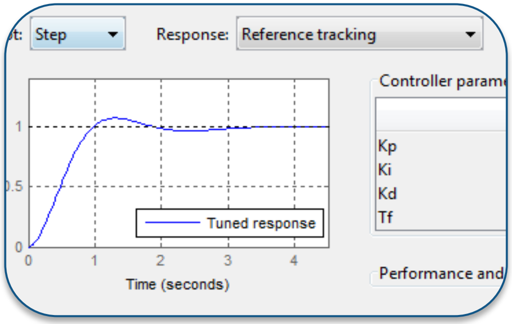
System Identification



Plant Model

3

Controller Design



Control System

4

Real-Time Testing

Agenda

- Data Acquisition
  - Hardware setup
  - Run on Target Hardware

- System Identification
  - Linear model estimation
  - Nonlinear model estimation

- Controller Design
  - PID controller tuning
  - Desktop simulation with a nonlinear model

- Real-Time Testing and Controller Implementation
  - Deployment to Arduino Uno
  - Real-time controller evaluation

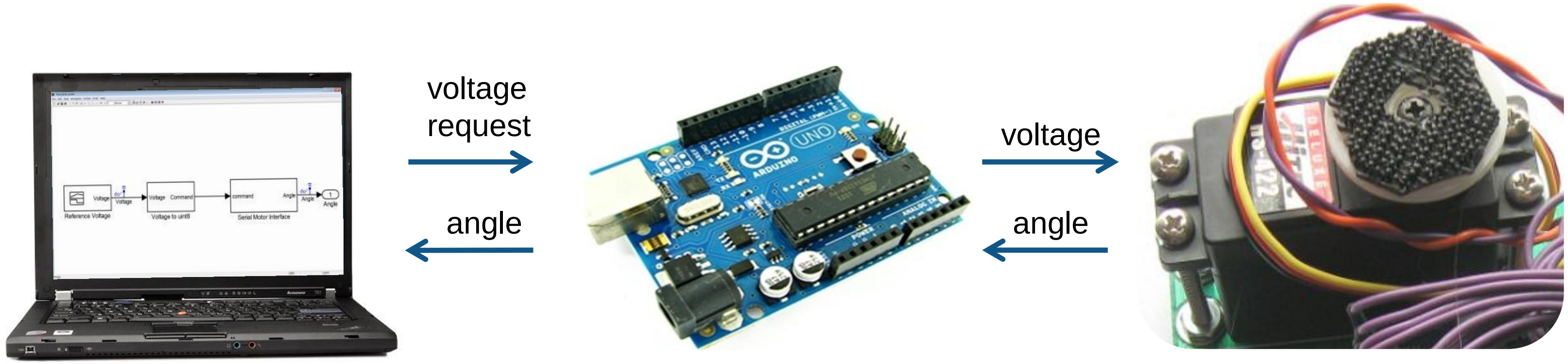
DATA ACQUISITION

Hardware Setup

Host Computer

Arduino Uno

DC Motor



voltage request

angle

voltage

angle

Serial communication

Motor driver

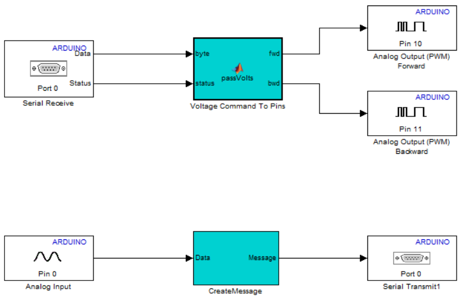
Run on Target Hardware

A Simulink feature in R2012a that:

- Creates an executable file from a model, and runs it on target hardware

- Is available from the model’s Tools menu
  - **Tools** > **Run on Target Hardware**

- Uses a *Target Installer* to install support packages for specific target hardware



ARDUINO UNO

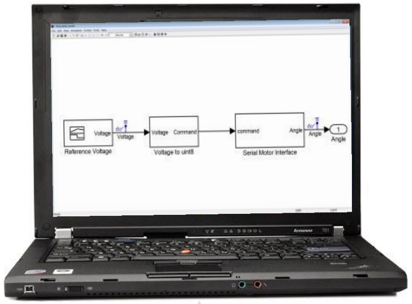
Requires Simulink®

Hardware Setup

Host Computer

Arduino Uno

DC Motor

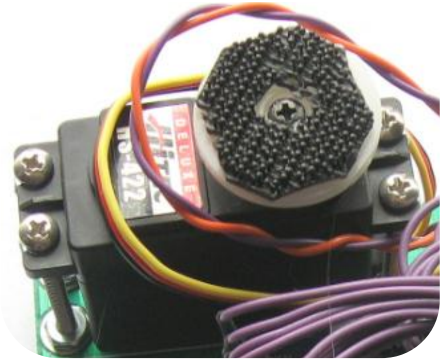


voltage request

angle

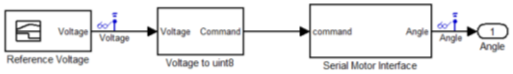
voltage

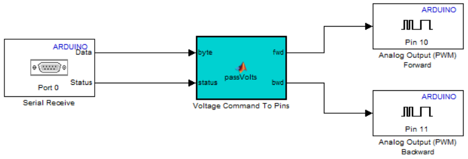
angle

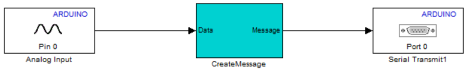


Serial communication (using Run On Target Hardware)

Motor driver

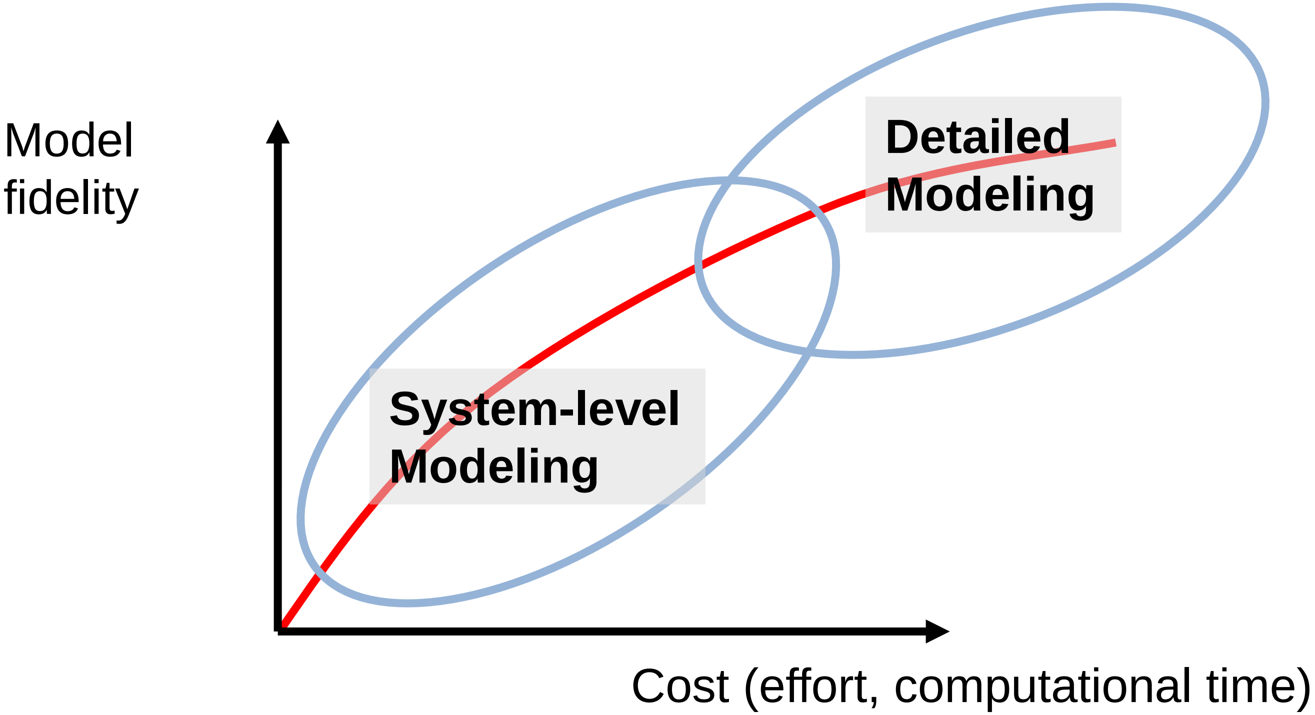






SYSTEM IDENTIFICATION

Model Fidelity vs. Cost



Model fidelity

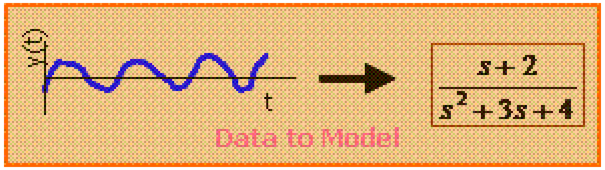
Detailed Modeling

System-level Modeling

Cost (effort, computational time)

- Model the dynamics that matter for your analysis
- Balance cost and model fidelity

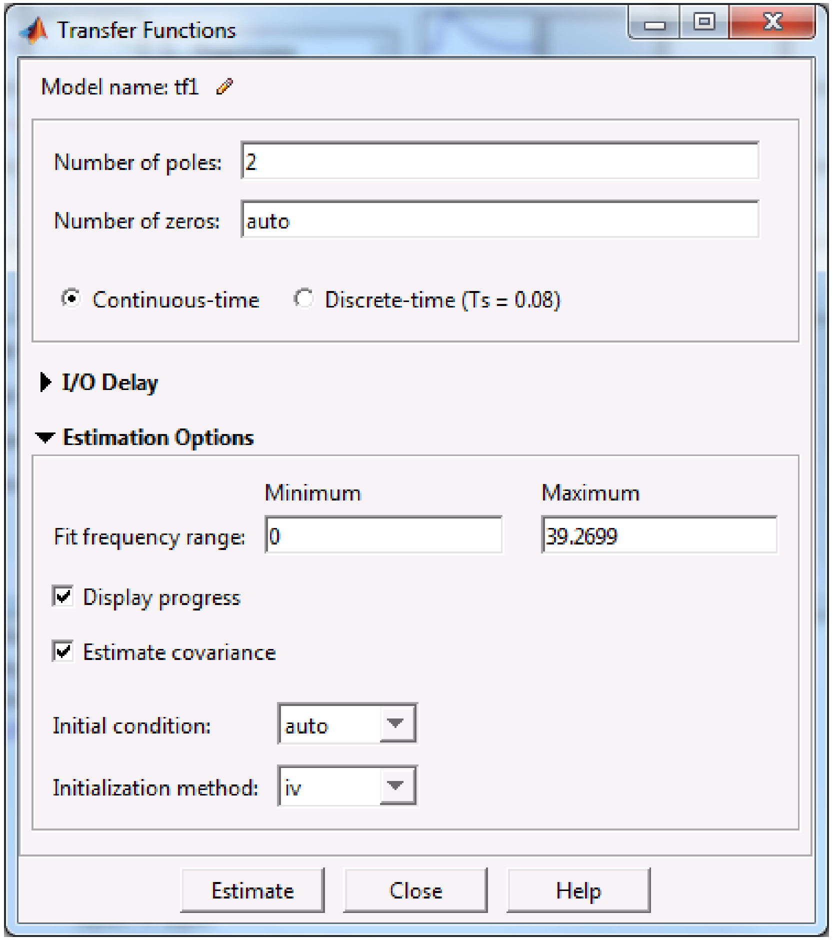
Linear System Identification



- Create continuous-time transfer functions from data

- Use either time or frequency domain data containing an arbitrary number of inputs and outputs

- Estimate other linear models: state-space, process models, and parametric models



Model name: tf1

Number of poles: 2

Number of zeros: auto

● Continuous-time ○ Discrete-time (Ts = 0.08)

	Minimum	Maximum
Fit frequency range:	0	39.2699

☑ Display progress

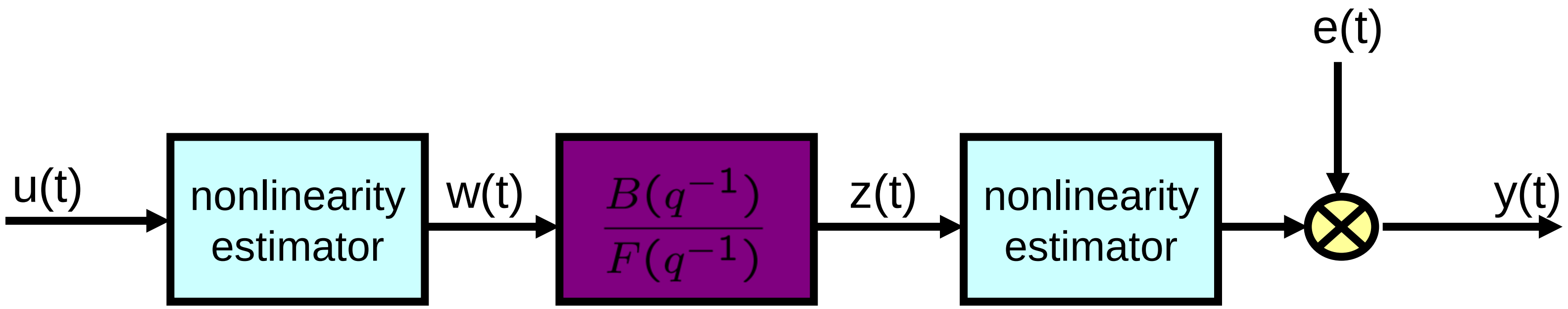
☑ Estimate covariance

Initial condition: auto

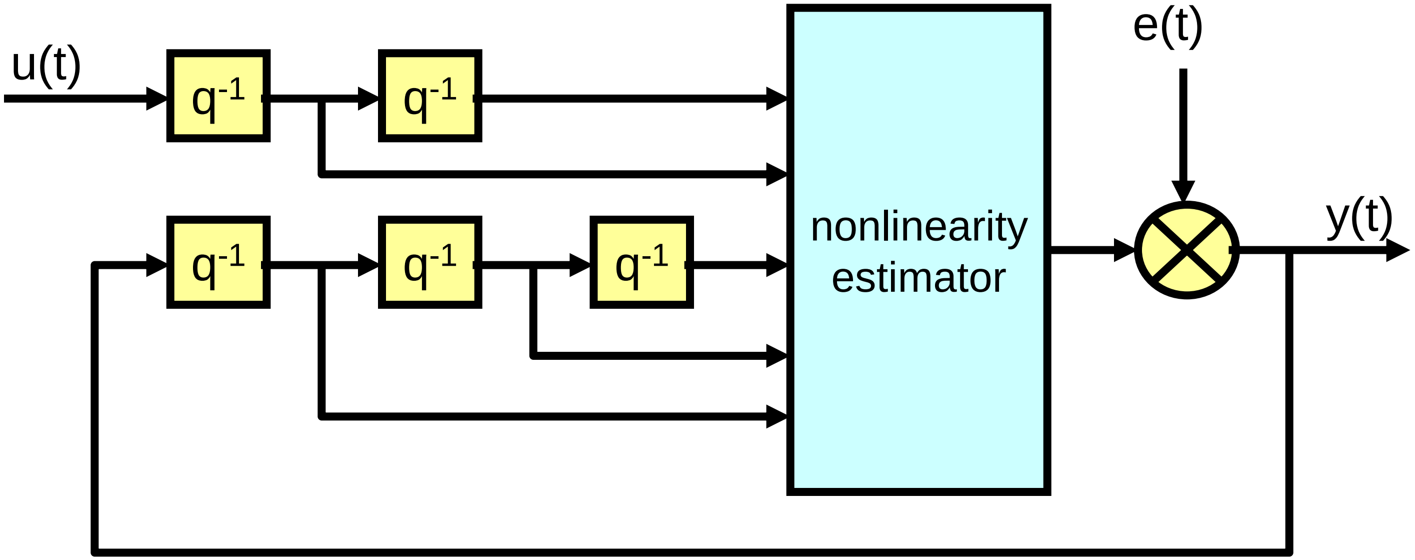
Initialization method: iv

Requires System Identification Toolbox™

Hammerstein-Wiener structure



Example of NLARX model



$$y(t) = f[y(t-1), y(t-2), y(t-3), u(t-1), u(t-2)] + e(t)$$

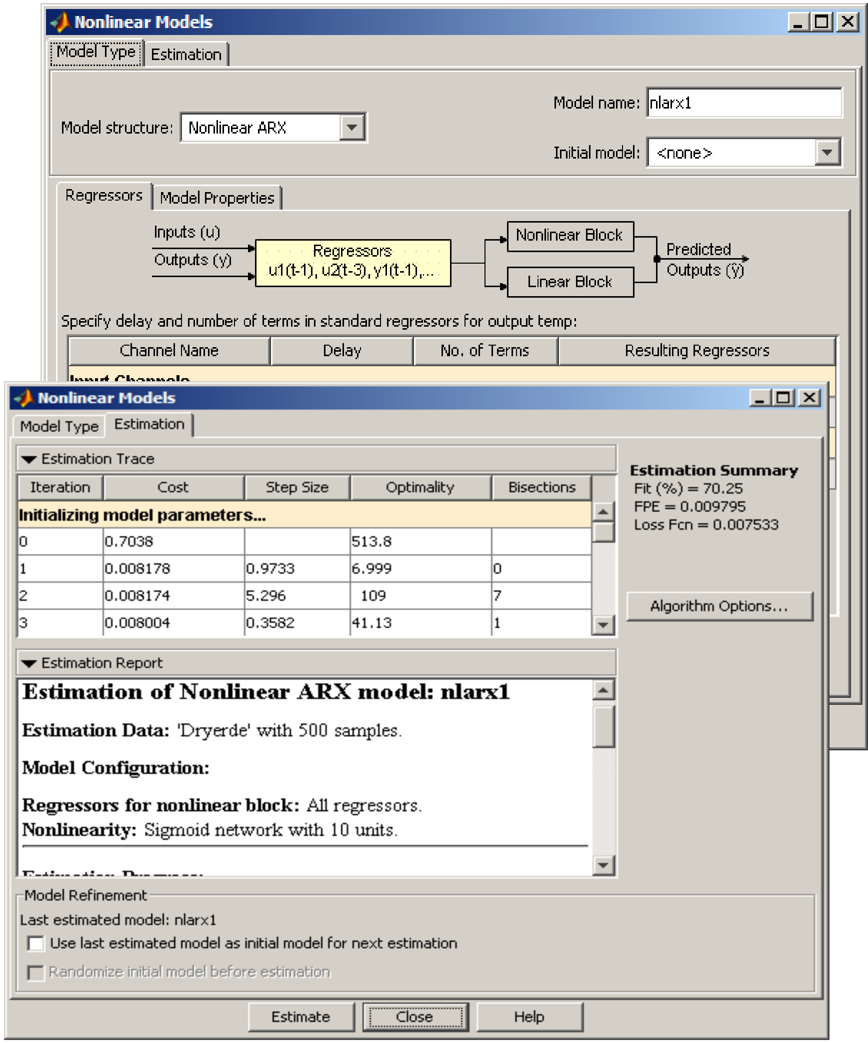
Nonlinear System Identification

- Estimate Hammerstein-Wiener models and nonlinear ARX models

- Use a variety of dynamic nonlinearities such as wavelets, neural networks, piecewise linear, etc.

- Estimate signal saturation and dead-zone behaviors affecting linear systems

- Use custom regressors in nonlinear ARX models for greater flexibility in modeling



Model structure: Nonlinear ARX

Model name: nlarx1

Initial model: <none>

Specify delay and number of terms in standard regressors for output temp:

Channel Name	Delay	No. of Terms	Resulting Regressors
Input Channels			

Iteration	Cost	Step Size	Optimality	Bisections
Initializing model parameters...				
0	0.7038		513.8	
1	0.008178	0.9733	6.999	0
2	0.008174	5.296	109	7
3	0.008004	0.3582	41.13	1

Estimation Summary
Fit (%) = 70.25
FPE = 0.009795
Loss Fcn = 0.007533

Estimation of Nonlinear ARX model: nlarx1

Estimation Data: 'Dryerde' with 500 samples.

Model Configuration:

Regressors for nonlinear block: All regressors.
Nonlinearity: Sigmoid network with 10 units.

Estimation Dynamics:

Model Refinement
Last estimated model: nlarx1
□ Use last estimated model as initial model for next estimation
□ Randomize initial model before estimation

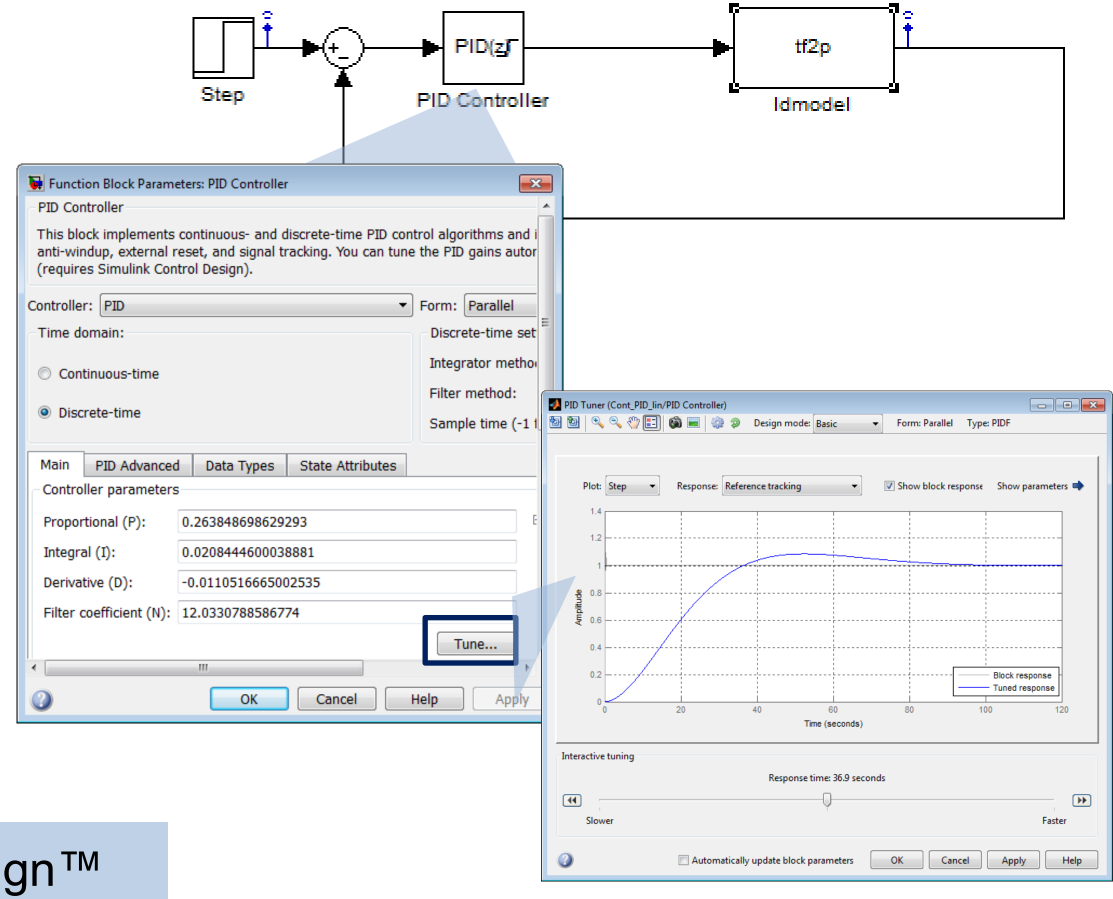
Requires System Identification Toolbox™

CONTROLLER DESIGN

Automatic PID Tuning

- Automatically linearizes Simulink models and finds gain values to meet specifications

- Provides additional fine-tuning capability with simple sliders



Function Block Parameters: PID Controller

PID Controller

This block implements continuous- and discrete-time PID control algorithms and i anti-windup, external reset, and signal tracking. You can tune the PID gains autor (requires Simulink Control Design).

Controller: PID
Form: Parallel

Time domain:
○ Continuous-time
● Discrete-time

Discrete-time set
Integrator metho
Filter method:
Sample time (-1 f

Controller parameters

Proportional (P):	0.263848698629293
Integral (I):	0.0208444600038881
Derivative (D):	-0.0110516665002535
Filter coefficient (N):	12.0330788586774

PID Tuner (Cont_PID_lin/PID Controller)

Design mode: Basic
Form: Parallel
Type: PIDF

Plot: Step
Response: Reference tracking
☑ Show block response
Show parameters

Interactive tuning

Response time: 36.9 seconds

☐ Automatically update block parameters

Requires Simulink Control Design™

For more info, watch our webinar “*PID Control Made Easy*”.

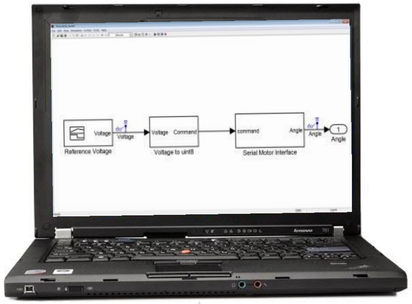
REAL-TIME TESTING AND CONTROLLER IMPLEMENTATION

Real-Time Testing

Host Computer

Arduino Uno

DC Motor

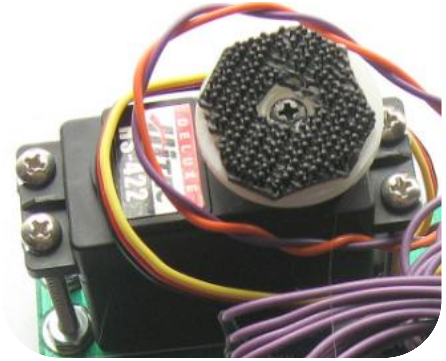


angle request

angle

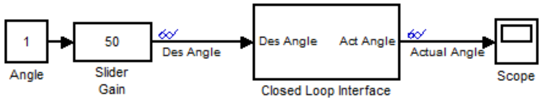
voltage

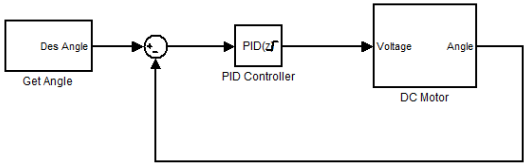
angle



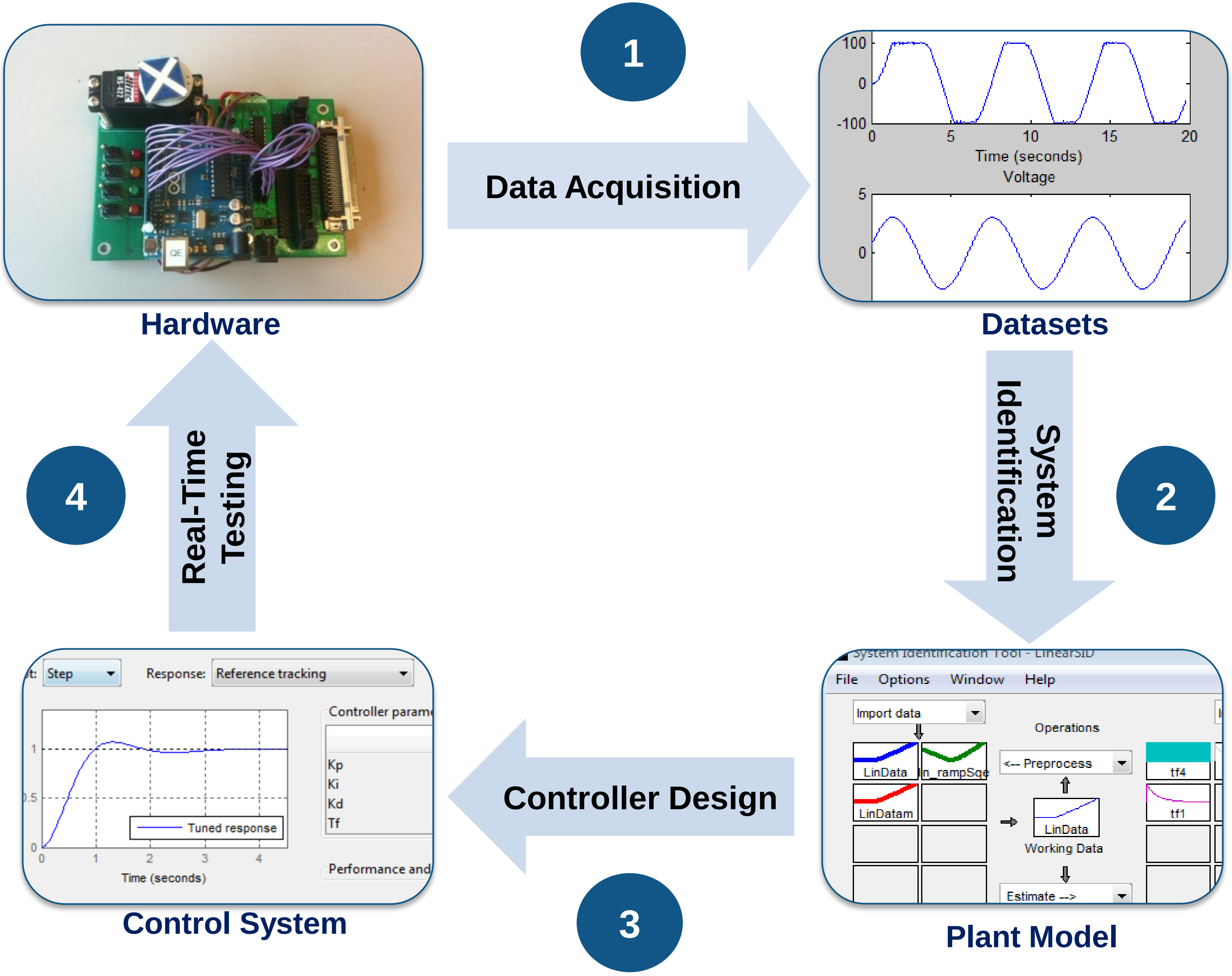
Serial communication (Using Run On Target Hardware)

Motor driver





Workflow



1 Data Acquisition

Hardware

Datasets

2 System Identification

Plant Model

3 Controller Design

Control System

4 Real-Time Testing

Summary

- MATLAB and Simulink support data-driven control design

- MATLAB and Simulink provide an environment for
  - Data acquisition
  - System identification
  - Control design
  - Real-Time testing