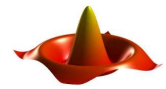
Creation of linear time-invariant Models (LTI-Models)



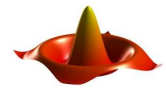
MATLAB Control System Toolbox
Linear, time-invariant modes (LTI–models)

Parametric Model

- Transfer Function (TF)
- Zero–Pole–Gain (ZPK)
- State–Space (SS)

Non-parametric models

- Frequency Response Data (FRD)



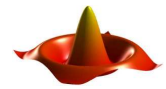
MATLAB Control System Toolbox
Transfer Function (TF)

- Transfer behaviour
- Rational expression of Laplace variable s:

$$h(s) = \frac{num(s)}{den(s)} = \frac{a_m\, s^m + a_{m-1}\, s^{m-1} + \ldots + a_1\, s + a_0}{b_n\, s^n + b_{n-1}\, s^{n-1} + \ldots + b_1\, s + b_0}$$

Numerator polynomial num and denominator polynomial den

Order of numerator m and order of denominator n



MATLAB Control System Toolbox
Create TF–SISO–function

1. **Command `tf`(num, den):** vectors with coefficients of num, den in *descending order of powers of* s.

```
>> h = tf([2 -3],[1 1])
```

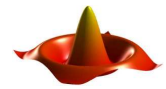
2. **Rational expression of** s

   a) Define **`s`** as TF–Model

```
>> s = tf('s')
```

   b) Transfer function as rational expression in **`s`**

```
>> h = (s+2) / (s^2 + 5*s + 4)
```

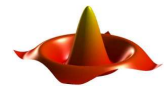
MATLAB Control System Toolbox
TF–MIMO–Model

- Two-dimensional $Ny \times Nu$–matrix $\mathbf{H}$ of SISO–TF:

$$\mathbf{H} = \begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{bmatrix} = \begin{bmatrix} \frac{num_{11}}{den_{11}} & \frac{num_{12}}{den_{12}} \\ \frac{num_{21}}{den_{21}} & \frac{num_{22}}{den_{22}} \end{bmatrix}$$

- Matrix element h_{ij}: SISO-TF-Transfer function from input j to output i

- Ny rows $\widehat{=}$ outputs, Nu columns $\widehat{=}$ inputs



MATLAB Control System Toolbox
Create TF–MIMO–modell

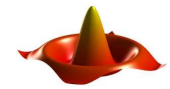
1. Define single SISO–TF:

   a) command `tf(`num, den`)`/rational expression in s

   b) create matrix $\mathbf{H}$

2. Command `tf(`$\mathbf{NUM, DEN}$`)`: Cell Array $\mathbf{NUM}$ for nominator polynomial und $\mathbf{DEN}$ for denominator polynomial $(Ny \times Nu)$

$$\mathbf{NUM} = \begin{bmatrix} num_{11} & num_{12} \\ num_{21} & num_{22} \end{bmatrix} \qquad \mathbf{DEN} = \begin{bmatrix} den_{11} & den_{12} \\ den_{21} & den_{22} \end{bmatrix}$$

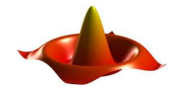


MATLAB Control System Toolbox
Zero–Pole–Gain Model (ZPK)

- Transfer behaviour
- Rational expression of Laplace variable s with Zeros and Poles:

$$h(s) = k \cdot \frac{(s - z_1) \cdot \ldots \cdot (s - z_{m-1}) \cdot (s - z_m)}{(s - p_1) \cdot \ldots \cdot (s - p_{n-1}) \cdot (s - p_n)}$$

k	Gain (scalar)
$z_1, \ldots, z_m$	roots of nominator (real/conj. complex)
$p_1, \ldots, p_m$	roots of denominator (real/conj. complex)



MATLAB Control System Toolbox
Create ZPK–SISO–function

1. **Command `zpk` (z,p,k):** vector of zeros z, vector of poles p and gain k.

```
>> h =  zpk([-6 1 1],[-5 1],3)
```

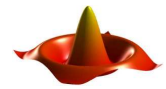
2. **Rational expression in s**

   a) Define **`s`** as ZPK–model

```
>> s = zpk('s')
```

   b) Transfer function as rational expression in **`s`**

```
>> h = 2 * 1 / ((s-1)*(s+2))
```

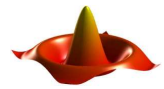
MATLAB Control System Toolbox
ZPK–MIMO–Model

- Two-dimensional $Ny \times Nu$–matrix $\mathbf{H}$ of SISO–ZPK:

$$\mathbf{H} = \begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{bmatrix} = \begin{bmatrix} k_{11} \cdot \dfrac{z_{11}}{p_{11}} & k_{12} \cdot \dfrac{z_{12}}{p_{12}} \\ k_{21} \cdot \dfrac{z_{21}}{p_{21}} & k_{22} \cdot \dfrac{z_{22}}{p_{22}} \end{bmatrix}$$

- Matrix element h_{ij}: SISO-ZPK-Transfer function from input j to output i

- Ny rows $\widehat{=}$ outputs, Nu columns $\widehat{=}$ inputs



MATLAB Control System Toolbox
Create ZPK–MIMO–modell

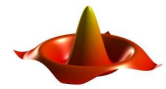
1. Define single SISO–ZPK:

   a) command **zpk**(z,p,k) / rational expression in s

   b) create matrix $\mathbf{H}$

2. Command **zpk** ($\mathbf{Z}$,$\mathbf{P}$,$\mathbf{K}$): Cell Array $\mathbf{Z}$ for polynomial of zeros und $\mathbf{P}$ for polynomial of poles ($Ny \times Nu$), $Ny \times Nu$–Matrix $\mathbf{K}$ of gains

$$\mathbf{Z} = \begin{bmatrix} z_{11} & z_{12} \\ z_{21} & z_{22} \end{bmatrix} \quad \mathbf{P} = \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{bmatrix} \quad \mathbf{K} = \begin{bmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{bmatrix}$$



MATLAB Control System Toolbox
State–Space–model (SS)

ODE of first order for each single integrator $\Rightarrow$ n ODE of first order instead of single ODE of nth order

State-equation:

$$\dot{\mathbf{x}} = \mathbf{A}\,\mathbf{x} + \mathbf{B}\,\mathbf{u}$$

Output equation:

$$\mathbf{y} = \mathbf{C}\,\mathbf{x} + \mathbf{D}\,\mathbf{u}$$

$\mathbf{x}$:	State vector	$(Nx \times 1)$
$\mathbf{u}$:	Input (control) vector	$(Nu \times 1)$
$\mathbf{y}$:	Output vector	$(Ny \times 1)$
$\mathbf{A}$:	State matrix	$(Nx \times Nx)$
$\mathbf{B}$:	Input matrix	$(Nx \times Nu)$
$\mathbf{C}$:	Output matrix	$(Ny \times Nx)$
$\mathbf{D}$:	feedthrough matrix	$(Ny \times Nu)$

MATLAB Control System Toolbox
Create SS–Model

- **Command `ss(A,B,C,D)`:**

$$\mathbf{A} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \quad \mathbf{C} = \begin{bmatrix} 0 & 1 \\ 1 & 2 \\ 3 & 1 \end{bmatrix} \quad \mathbf{D} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

```
>> ss([1 2 ; 3 4],[1 1 ; 0 1],[0 1 ; 1 2 ; 3 1],0)
```

- Feedthrough often $\mathbf{0}$: `D = 0`
- SISO–Model: $\mathbf{u} \rightarrow u$ $\mathbf{y} \rightarrow y$ $\mathbf{B} \rightarrow \mathbf{b}$ $\mathbf{C} \rightarrow \mathbf{c}^T$

MATLAB Control System Toolbox
Frequency response data model (FRD)

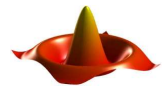
- Frequency response data (measurements/simulation)
- Sinusoidal excitation: $y(t) = |G(\omega)| \cdot \sin(\omega t + \varphi(\omega))$

  $\Rightarrow$ $y(t)$ phase shifted and amplitude-modulated

- Frequency response: $F(j\omega) = |F(j\omega)| \cdot e^{j\varphi(\omega)}$

  Abs. value: $|F(j\omega)| = \sqrt{\mathrm{Re}\{F(j\omega)\}^2 + \mathrm{Im}\{F(j\omega)\}^2}$

  Phase : $\varphi(\omega) = \arctan \dfrac{\mathrm{Im}\{F(j\omega)\}}{\mathrm{Re}\{F(j\omega)\}}$



MATLAB Control System Toolbox
Create Frequency response data model

- **Command** `frd`$(resp, freq, unit)$:

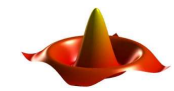
  $resp$ Vector with conj. complex frequency responses

  $freq$ Vector of frequencies

  $unit$ Units of frequency (`'rad/s'` or `'Hz'`)

- Example:
```
>> freq = [0.01 0.1 1 10 100 1000 10000] ;
>> resp = (1-j*freq) ./(1+freq.^2)
>> sysfrd = frd(resp,freq,'Units','rad/s')
```

- MIMO–System: $resp$ is a tensor $(Ny \times Nu \times Nf)$



MATLAB Control System Toolbox
Discrete-time Models

- Additional parameter: Sampling time `Ts`

```
tf(num,den,Ts)        zpk(z,p,k,Ts)
ss(a,b,c,d,Ts)        frd(ant,freq,Ts)
```

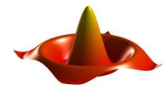
- Sampling time unspecified: `Ts = -1`

- DSP–Format: `h = filt ([1 -0.5],[1 1 -2],0.01)`

- State space model:

$$\mathbf{x}(k+1) = \mathbf{A}\,\mathbf{x}(k) + \mathbf{B}\,\mathbf{u}(k)$$

$$\mathbf{y}(k) = \mathbf{C}\,\mathbf{x}(k) + \mathbf{D}\,\mathbf{u}(k)$$



MATLAB Control System Toolbox
Time delays

Continuous-time:

- between Inputs and Outputs (TF, ZPK, FRD):

  `set(sys,'IODelay',Td)`

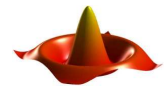
- At inputs and outputs (SS):

  `set(sys,'InputDelay',Tde)`

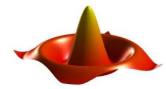
  `set(sys,'OutputDelay',Tda)`

Discrete-time:

- Pade approximation: `pade(sys,n)`



Converting and Working with LTI–Models



MATLAB Control System Toolbox
LTI–Objects

- LTI–Models are cell arrays with predefined model properties and property values
- Common model properties:

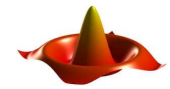
  `Ts, InputDelay, OutputDelay, ioDelayMatrix, InputName, OutputName, InputGroup, OutputGroup, Notes, Userdata`
- Model specific model properties:

  TF: `num, den, Variabl (s, p; z, q, z^-1)`

  ZPK: `z, p, k, Variabl (s, p; z, q, z^-1)`

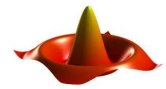
  SS: `a, b, c, d, StateName`

  FRD: `Frequency, ResponseData, Units`



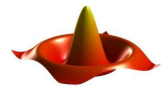
MATLAB Control System Toolbox
Get, set and change model properties

- Get model properties:
  - Command **get**: `get(`sys`,'`$PropName$`')`
  - "."–Command: $sys.PropName$
- Set and change model properties:
  - Direct: sys `= tf(...,'`$PropName$`',`$PropValue$`)`
  - Comd. **set**: `set(`sys`,'`$PropName$`',`$PropValue$`,...)`
  - "."–Comd.: $sys.PropName = PropWert$
- Fast data retrieval: `tfdata`, `zpkdata`, `ssdata`, `frddata`



MATLAB Control System Toolbox
Precedence and Property Inheritance

- Precedence list: FRD > SS > ZPK > TF
- Conversion: – a priori: `sys = systf + tf(sysss)`
  – a posteriori: `sys = tf(systf + sysss)`
- Property inheritance of model properties
  – Discret: all sampling times identical or unspecified
  – Variable `Variable`: ◇ cont.: `p > s`
    ◇ discret: `z^-1 > q > z`



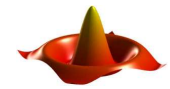
MATLAB Control System Toolbox
Conversion between model types

- Express conversion: Apply commands `tf(sys)`, `ss(sys)`, `zpk(sys)` or `frd(sys,freq)` on models

- Conversion with MATLAB commands:

`zp2tf(z,p,k)`	`tf2zp(num,den)`	`tf2ss(num,den)`
`ss2tf(A,B,C,D,iu)`	`ss2zp(A,B,C,D,i)`	`zp2ss(z,p,k)`

- Automatic conversion: Operations convert models automatically in respective model type



MATLAB Control System Toolbox
Arithmetic operations

- Addition and subtraction: parallel interconnection

  $\mathbf{y} = \mathbf{G_1} \cdot \mathbf{u} + \mathbf{G_2} \cdot \mathbf{u}$ $\Leftrightarrow$ `sys = sys1 + sys2`

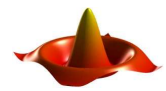
- Multiplication: series interconnection

  $\mathbf{y} = \mathbf{G_1} \cdot \mathbf{v} = \mathbf{G_1} \cdot (\mathbf{G_2} \cdot \mathbf{u})$ $\Leftrightarrow$ `sys = sys1 * sys2`

- Inversion: $\mathbf{u} = \mathbf{G}^{-1}\mathbf{y}$ $\Leftrightarrow$ `sys = inv(sys)`

- Left and right division:

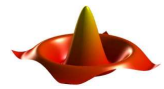
  $\mathbf{G_1}^{-1}\mathbf{G_2}$ $\Leftrightarrow$ `sys1 \ sys2` $\mathbf{G_1}\mathbf{G_2}^{-1}$ $\Leftrightarrow$ `sys1 / sys2`



MATLAB Control System Toolbox
Extracting and modifying LTI models

- Address inputs and outputs with `sys(`i`,`j`)`
- Subsystems:
  - extraction: `subsys = systf(1,2)`
  - modification: `systf(2,1) = subsys`
- Inputs/Outputs:
  - deletion: `systf(:,1) = []`
  - adding: `systf = [systf,sys]` (typ by precedence)

    `systf(:,2) = sys` (typ `systf`)



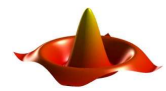
MATLAB Control System Toolbox
Concatenation of LTI models

- Horizontal: sys = [$sys1$, $sys2$]
- Vertical: sys = [$sys1$; $sys2$]
- Diagonal: sys = append($sys1$,$sys2$)
- Parallel and serial interconnection:

  sys = parallel($sys1$,$sys2$,$in1$,$in2$,$out1$,$out2$)

  sys = series($sys1$,$sys2$,$outputs1$,$inputs2$)

- Feedback: sys = feedback($sys1$,$sys2$)



MATLAB Control System Toolbox
Continuous-time and discret-time models

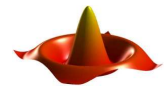
- continuous $\rightarrow$ discret: `sysd = c2d(sysc,Ts)`

  discret $\rightarrow$ continuous: `sysc = d2c(sysd,`$method$`)`

- Discretization $method$:
  - Zero order hold: `zoh`
  - First order hold: `foh`
  - Tustin–Approximation: `tustin`

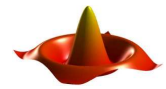
- Discretization of delayed models

- Resampling: `sys = d2d(sys,Ts)`



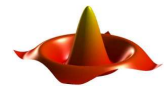
Model

analysis



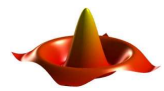
MATLAB Control System Toolbox
Overview

- General properties
- Model dynamics
- Time Response
- Frequency Response
- Model order reduction
- State space realizations



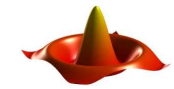
MATLAB Control System Toolbox
General properties

- Get and check of model properties
- Useful for programming complex scripts and functions
  - Evaluation routines
  - Create complex plots
- Return values are model properties or logical values



MATLAB Control System Toolbox
Model types, time dates, structures, size

- Model types:
  - show: `class(`*sys*`)`
  - test: `isa(`*sys*`,'`*classname*`')`
- Time dates:
  - continuous-time: `isct(`*sys*`)`
  - discret-time: `isdt(`*sys*`)`
  - delays: `hasdelay(`*sys*`)`
- Structures:
  - In- & Outputs: `isempty(`*sys*`)`
  - Order: `isproper(`*sys*`)`
  - SISO–Model: `issiso(`*sys*`)`
- Size: `size(`*sys*`)`



MATLAB Control System Toolbox
Model dynamics

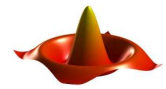
- Low-frequency (DC) gain: `dcgain(`sys`)`
  - Frequency is $s = 0$ or $z = 1$
  - Integrators: gain ∞

- Natural frequency & damping ratio: `damp(`sys`)`
  - Pole: $p_i = \alpha \pm j\,\beta$
  - Frequency: $\omega_n = \sqrt{\alpha^2 + \beta^2}$
  - Damping: $D = -\dfrac{\alpha}{\sqrt{\alpha^2 + \beta^2}}$

MATLAB Control System Toolbox
Zeros and Poles of LTI models

- Zeros: `zero(`sys`)`
- Poles:
  - Eigenvalues of matrix $\mathbf{A}$: `pole(`sys`)`
  - Roots of polynomial c: `roots(`c`)`
- Sorting:
  - continuous-time: [s,ndx] = `esort(`p`)`
  - discret-time: [s,ndx] = `dsort(`p`)`
- Pole-zero map: [p,z] = `pzmap(`sys`)`

  Constant damping factors & natural frequencies grid:
  - continuous-time: `sgrid`
  - discret-time: `zgrid`



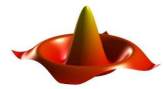
MATLAB Control System Toolbox
Time responses

- Response continuous-time/discret-time:

$$\mathbf{y}(t) \;=\; \mathbf{C}e^{\mathbf{A}t}\mathbf{x_0} \;+\; \int_{\tau=0}^{t} \left(\mathbf{C}e^{\mathbf{A}(t-\tau)}\mathbf{B}+\mathbf{D}\right)\mathbf{u}(\tau)\,d\tau$$

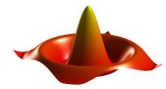
$$\mathbf{y}[k] \;=\; \mathbf{C}A^{k}\mathbf{x_0} \;+\; \sum_{i=0}^{k-1} \left(\mathbf{C}\mathbf{A}^{k-i-1}\mathbf{B}+\mathbf{D}\right)\mathbf{u}[i]$$

- Command: `[`y`,`t`,`x`] = `$command$`(`sys`,`par`)`
- Time vector: t `= 0:`dt`:`Tf
- Output y:
  - SIMO: `length(`t`)` $\times\, Ny$
  - MIMO: `length(`t`)` $\times\, Ny \times Nu$



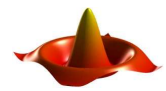
MATLAB Control System Toolbox
Initial, Impulse and Step response

- **Initial response:** `[`y`,`t`,`x`] = initial(`sys`,`x_0`,[,`t`])`
  - Inputs set to 0
  - Initial conditions in vector x_0
- **Impulse response:** `[`y`,`t`,`x`] = impulse(`sys`[,`t`])`
  - continuous-time: Dirac input $\delta(t)$
  - discret-time: unit pulse $\delta[k]$
- **Step response:** `[`y`,`t`,`x`] = step(`sys`[,`t`])`
  - continuous-time: unit step $\sigma(t)$
  - discret-time: series of unite pulses $\delta[k]$



MATLAB Control System Toolbox
Time response to arbitrary inputs

- **Time response:** `[`y`,`t`,`x`] = lsim(`sys`,`u`,`t`,[,`x_0`])`
  - Arbitrary input vector u
  - suggest more appropriate sampling time dt
- **Test signal:** `[`u`,`t`] = gensig(`typ`,`tau`[,`Tf`,`Ts`])`
  - typ : `'sin'` Sine; `'pulse'` Pulse; `'square'` Periodic pulse
  - tau : Period
  - Tf : Duration
  - Ts : Sample time
- **Linear simulation tool:** `lsimplot(`sys`)`



MATLAB Control System Toolbox
Frequency response

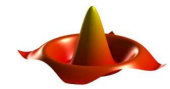
Frequency resp.: conj. complex answer to sinusoidal input in steady state

Precondition: model asymptotically stable, ie. real parts of all Eigenvalues lower 0

Frequency response-TF: $F(j\omega) = |F(j\omega)| \cdot e^{j\varphi(\omega)}$

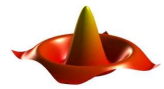
Abs. Value: $|F(j\omega)| = \sqrt{\mathrm{Re}\{F(j\omega)\}^2 + \mathrm{Im}\{F(j\omega)\}^2}$

Phase : $\varphi(\omega) = \arctan \dfrac{\mathrm{Im}\{F(j\omega)\}}{\mathrm{Re}\{F(j\omega)\}}$



MATLAB Control System Toolbox
Frequency response and Bode plot

- Calculate frequency response:
  - Single frequency f: $frsp$ = `evalfr(`sys`,`f`)`
  - Frequency vector w: H = `freqresp(`sys`,`w`)`
- Bode plot: [mag`,`$phase$`,`w`]` = `bode(`sys`)`
  - Frequency vector ω (w) as logarithmic x-axis
  - Magnitude $|F(j\omega)|$ (mag) in log-log-style
  - Phase $\varphi(\omega)$ ($phase$) semi-logarithmic



MATLAB Control System Toolbox
Gain and phase margin

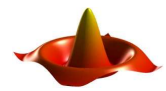
- Command [Gm,Pm,Wcg,Wcp] = `margin`(sys)

  Gain margin Gm, phase crossing frequency Wcg
  Phase margin Pm, gain crossing frequency Wcp

  Stable for: $\omega_A < \omega_\varphi$ $F_R > 1$ $\varphi_R > 0$

- Structure $stable$ = `allmargin`(sys)

  Gain margin: `GainMargin` and `GMFrequency`
  Phase margin: `PhaseMargin` and `PMFrequency`
  Delay margin: `DelayMargin` and `DMFrequency`
  Stable for: `Stable`, `1` if stable, otherwise `0`



MATLAB Control System Toolbox
Nyquist plot

- Nyquist plot: `[re,im,w] = nyquist(sys)`

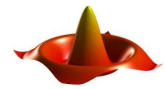
  Real and imaginary part of transfer function of open loop circuit from $\omega = 0$ to ∞ (root locus):

$$-F_0(j\omega) = \frac{Z_0(j\omega)}{N_0(j\omega)} \cdot e^{-j\omega T_t}\,; \quad n_0 > m_0\,,\; T_t \geq 0$$

- Stability: $\overset{\omega=+\infty}{\underset{\omega=+0}{\Delta}} \phi_{soll} = n_r \cdot \pi + n_a \cdot \frac{\pi}{2}$

  n_r, n_a: numb. of in-/marginal stable poles

- Critical point: $-1 + j\,0$

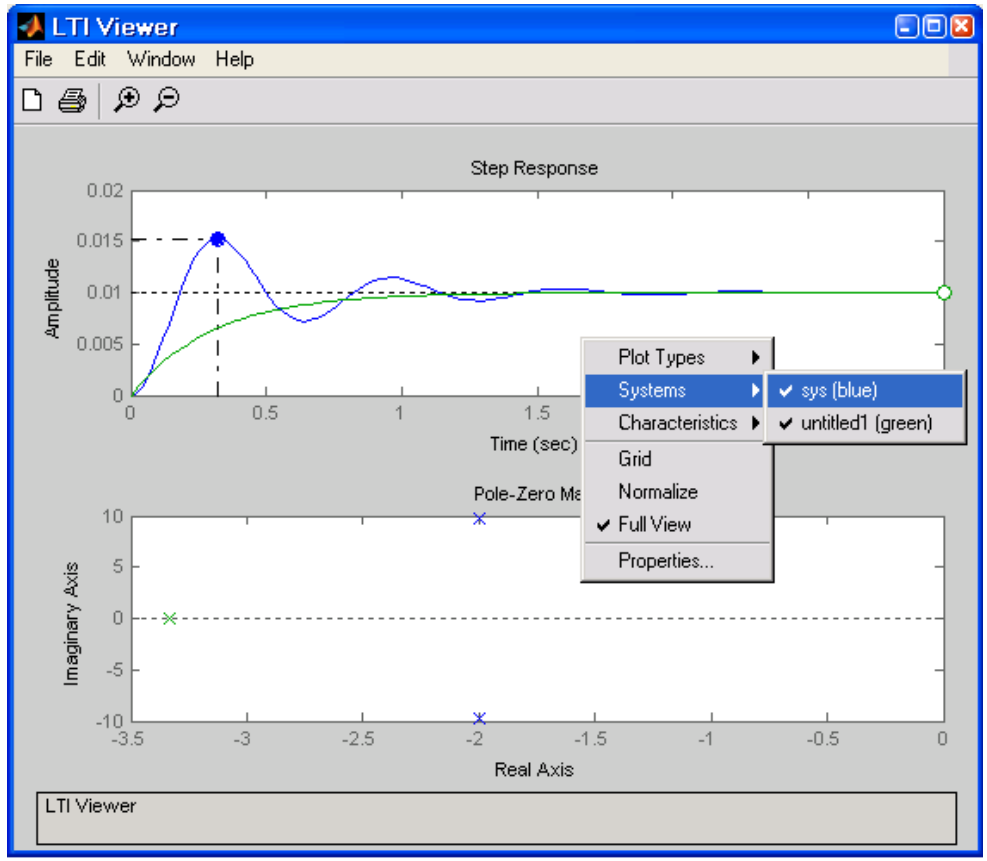


MATLAB Control System Toolbox
LTI viewer

GUI for LTI models

Import and Export of Models

Start by: `>> ltiview`



MATLAB Control System Toolbox
Observability

- Observability matrix: `obsv(`A`,`C`)`
  - $\mathbf{Ob} = \left[\mathbf{C}^T \;\; \mathbf{A}^T\mathbf{C}^T \;\; (\mathbf{A}^T)^2\mathbf{C}^T \;\; \ldots \;\; (\mathbf{A}^T)^{n-1}\mathbf{C}^T \right]^T$
  - rank($\mathbf{Ob}$) $\widehat{=}$ number of observability states

- Observability staircase form: `obsvf(`A`,`B`,`C`,[`tol`])`
  - Transformation: $\overline{A} = \mathbf{TAT}^T$, $\overline{B} = \mathbf{TB}$, $\overline{C} = \mathbf{CT}^T$
  - Staircase transformed model:

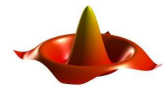
$$\overline{A} = \begin{bmatrix} \mathbf{A_{no}} & \mathbf{A_{12}} \\ \mathbf{0} & \mathbf{A_o} \end{bmatrix} \quad \overline{B} = \begin{bmatrix} \mathbf{B_{no}} \\ \mathbf{B_o} \end{bmatrix} \quad \overline{C} = \begin{bmatrix} \mathbf{0} & \mathbf{C_o} \end{bmatrix}$$

MATLAB Control System Toolbox
Controllability

- Controllability matrix: `ctrb(`A`,`B`)`
  - $\mathbf{Co} = \left[\mathbf{B} \;\; \mathbf{AB} \;\; \mathbf{A^2B} \;\; \ldots \;\; \mathbf{A^{n-1}B} \right]$
  - rank($\mathbf{Co}$) $\widehat{=}$ number of controllable states

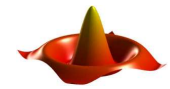
- Controllability staircase form: `ctrbf(`A`,`B`,`C`,[`tol`])`
  - Transformation: $\overline{A} = \mathbf{TAT}^T$, $\overline{B} = \mathbf{TB}$, $\overline{C} = \mathbf{CT}^T$
  - Staircase transformed model:

$$\overline{A} = \begin{bmatrix} \mathbf{A_{nc}} & \mathbf{0} \\ \mathbf{A_{21}} & \mathbf{A_c} \end{bmatrix} \quad \overline{B} = \begin{bmatrix} \mathbf{0} \\ \mathbf{B_c} \end{bmatrix} \quad \overline{C} = \begin{bmatrix} \mathbf{C_{nc}} & \mathbf{C_c} \end{bmatrix}$$



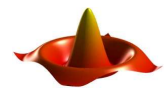
Designing

Compensators



MATLAB Control System Toolbox
Overview

- Root locus design
- Interactive design with CETM and SISO Design Tool
- State-feedback control with pole placement and state estimation
- Linear quadratic Gaussian (LQG) methods
- Kalman estimator for white noise signals



MATLAB Control System Toolbox
Root locus design

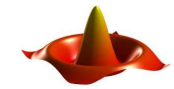
- Root locus: Tune the loop gain of a control system by specifying a designed set of closed-loop pole locations in root locus diagram.

- Open loop transfer function

$$-G_0 \;=\; k \cdot G_R \cdot G_S \cdot G_M \;=\; k \cdot \frac{n_R\, n_S\, n_M}{d_R\, d_S\, d_M}$$

- Poles of G_0 = roots of denominator polynomial of G

$$d_0 + k \cdot n_0 \;=\; d_R\, d_S\, d_M + k \cdot n_R\, n_S\, n_M \;=\; 0$$



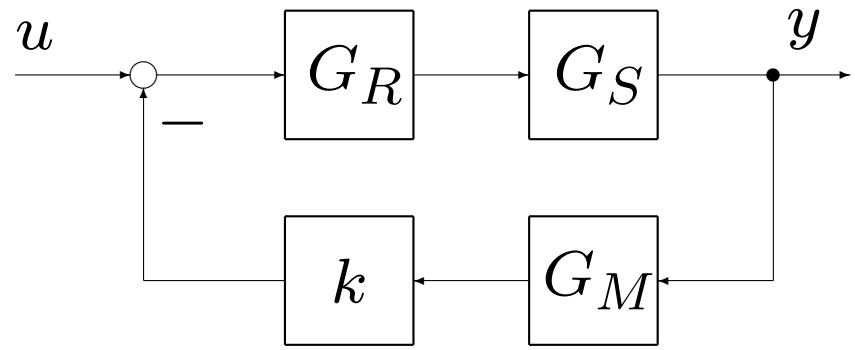
MATLAB Control System Toolbox
Root locus design commands

- Evans root locus plot:

  `rlocus(`sys`[,`k`])`

  `[`r`,`k`] = rlocus(`sys`)`

  r `= rlocus(`sys`,`k`)`



- Interactive gain selection from root locus plot:

  `[`k`,`r`] = rlocfind(`sys`)`

  `[`k`,`r`] = rlocfind(`sys`,`p`)`

MATLAB Control System Toolbox
SISO Design Tool

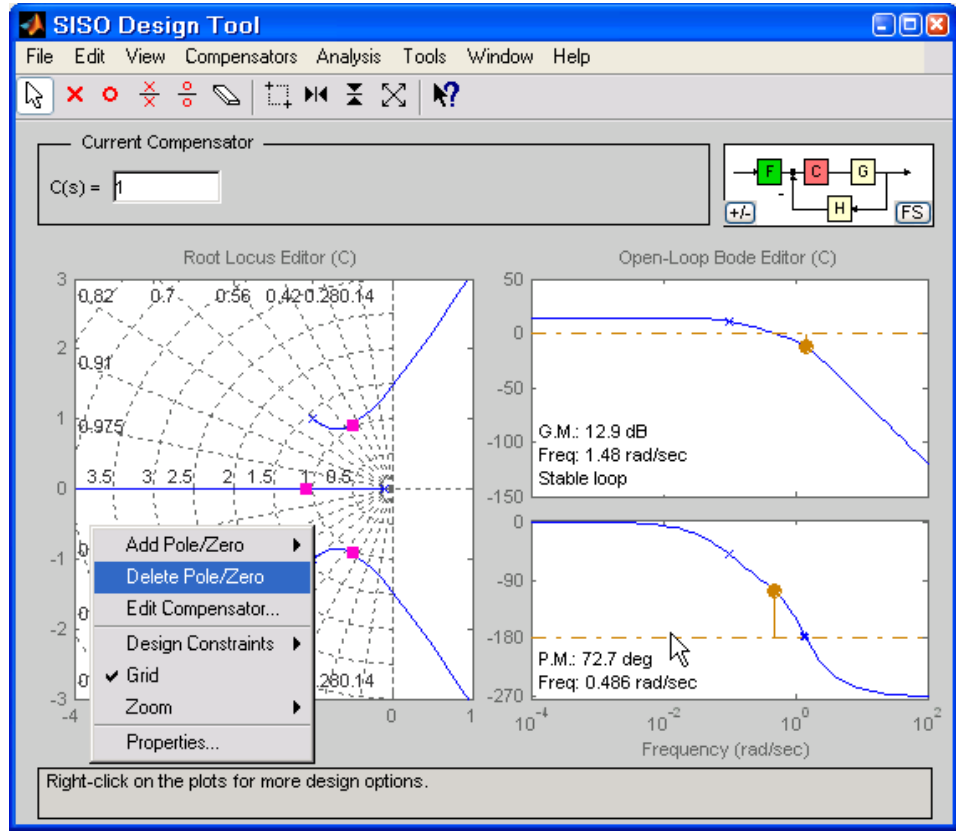
SISO Design Tool

Compensator design:

- Bode diagramm
- Root locus

Start by:

`sisotool(`sys`)`



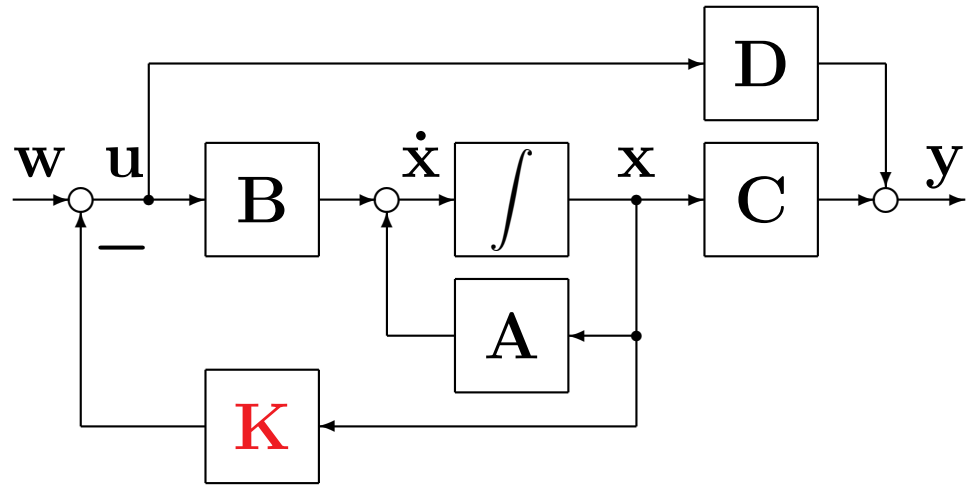
MATLAB Control System Toolbox
State-feedback control

State feedback control with compensator gain matrix $\mathbf{K}$

Plant:

$$\dot{\mathbf{x}} = \mathbf{A}\,\mathbf{x} + \mathbf{B}\,\mathbf{u}$$

$$\mathbf{y} = \mathbf{C}\,\mathbf{x} + \mathbf{D}\,\mathbf{u}$$



State-feedback law ($\mathbf{w} = \mathbf{0}$): $\mathbf{u} = -\mathbf{K}\,\mathbf{x}$

Closed loop: $\dot{\mathbf{x}} = (\mathbf{A} - \mathbf{B}\,\mathbf{K}) \cdot \mathbf{x}$

MATLAB Control System Toolbox
State estimation - Luenberger estimator

Feedback of output error with estimator gain matrix $\mathbf{L}$

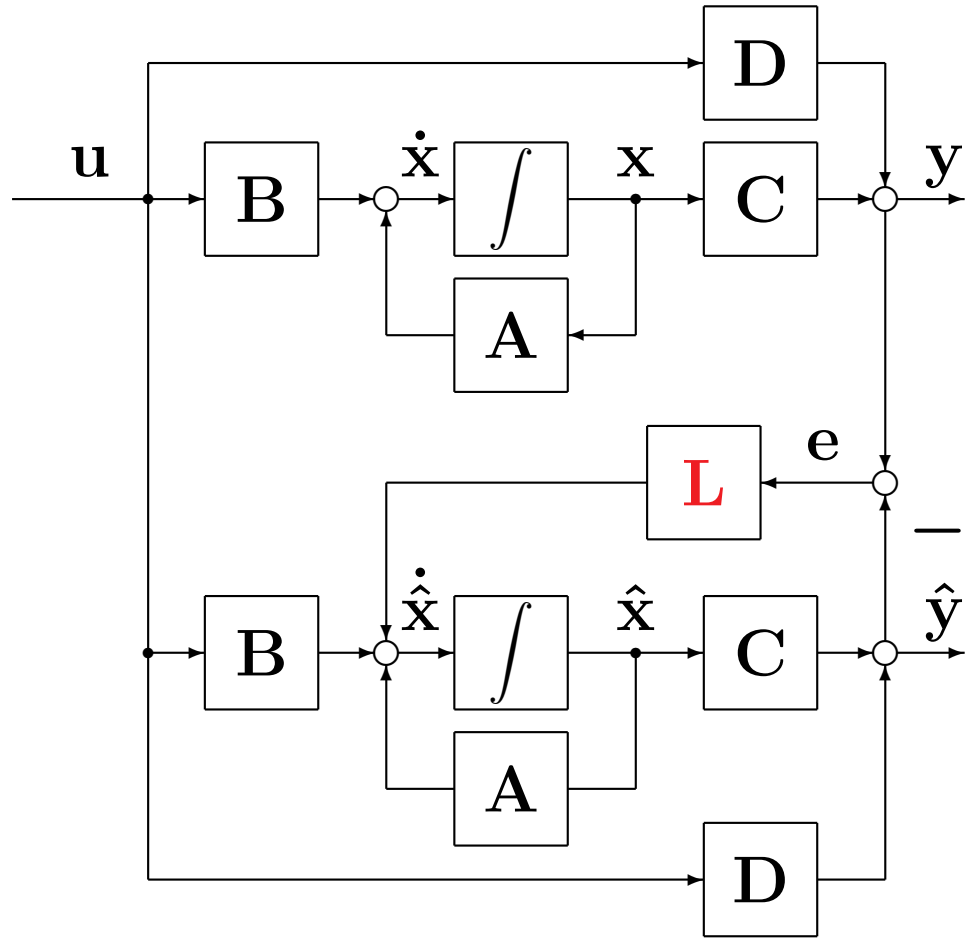
Estimator:

$$\dot{\hat{\mathbf{x}}} = \mathbf{A}\,\hat{\mathbf{x}} + \mathbf{B}\,\mathbf{u} + \mathbf{L}\,\mathbf{e}$$

$$\hat{\mathbf{y}} = \mathbf{C}\,\hat{\mathbf{x}} + \mathbf{D}\,\mathbf{u}$$

Output error:

$$\mathbf{e} = \mathbf{y} - \hat{\mathbf{y}} = \mathbf{C}\,(\mathbf{x} - \hat{\mathbf{x}})$$



MATLAB Control System Toolbox
Polplazierung

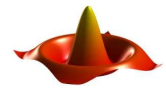
- Pole placement: Calculate compensator gain matrix $\mathbf{K}$ such that closed-loop poles are assigned to desired poles in complex plane

- Compensator gain vector $\mathbf{k}$ / matrix $\mathbf{K}$

  `k = acker(A,b,p)`
  `K = place(A,B,p)`
  `[K,prec,message] = place(A,B,p)`

- Observer gain matrix $\mathbf{L}$

  `L = acker(A',c',p) .'`
  `L = place(A',C',p) .'`



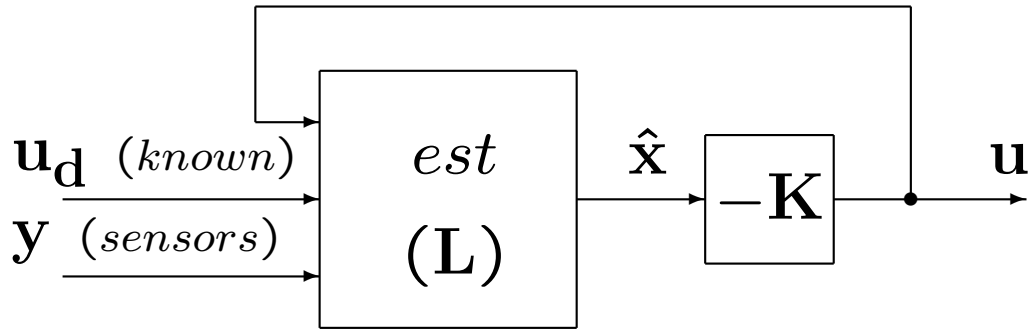
MATLAB Control System Toolbox
Design state-feedback compensator and estimator

- Design state-feedback estimator

  est = `estim(`sys`,`L`)`
  est = `estim(`sys`,`L`,`$sensors$`,`$known$`)`

- Design state-feedback compensator and estimator

  $rsys$ = `reg(`sys`,`K`,`L`)`
  $rsys$ = `reg(`sys`,`K`,`L`,`$sensors$`,`$known$`,`$controls$`)`



MATLAB Control System Toolbox
Linear quadratic Gaussian (LQG) methods

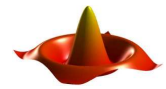
- Consider process disturbances & measurement noise
- Minimization of quadratic performance criterion:
  ($\mathbf{Q}$ weighted states, $\mathbf{R}$ weighted inputs)

$$J(\mathbf{x},\mathbf{u}) = \int_{t=0}^{\infty} \left(\mathbf{x}^T \mathbf{Q}\, \mathbf{x} + \mathbf{u}^T \mathbf{R}\, \mathbf{u} + 2\, \mathbf{x}^T \mathbf{N}\, \mathbf{u} \right) dt$$

- Solve algebraic Riccati equatiion

$$\mathbf{0} = \mathbf{A}^T \mathbf{S} + \mathbf{S}\, \mathbf{A} - (\mathbf{S}\, \mathbf{B} + \mathbf{N})\, \mathbf{R}^{-1} (\mathbf{B}^T \mathbf{S} + \mathbf{N}^T) + \mathbf{Q}$$

- LQ-optimal gain: $\mathbf{K} = \mathbf{R}^{-1} (\mathbf{B}^T \mathbf{S} + \mathbf{N}^T)$



MATLAB Control System Toolbox
LQG optimized compensator gain matrix K

- LQG optimized compensator gain matrix $\mathbf{K}$ $\mathbf{K}$

$$\begin{aligned}
[K,S,e] &= \texttt{lqr}(A,B,Q,R[,N]) \\
[K,S,e] &= \texttt{dlqr}(A,B,Q,R[,N]) \\
[Kd,S,e] &= \texttt{lqrd}(A,B,Q,R[,N],Ts) \\
[K,S,e] &= \texttt{lqry}(sys,Q,R[,N])
\end{aligned}$$

- Command `lqrd`: Sample Time Ts

- Command `lqry`:

$$J(\mathbf{y},\mathbf{u}) = \int_{t=0}^{\infty} \left(\mathbf{y}^T\,\mathbf{Q}\,\mathbf{y} + \mathbf{u}^T\,\mathbf{R}\,\mathbf{u} + 2\,\mathbf{y}^T\,\mathbf{N}\,\mathbf{u}\right) dt$$