Polynomial Methods for Control Analysis and Design

[PolyX]

3/ Polynomials
in control systems

Overview

[PolyX]

- Ch. 1. Polynomials and polynomial matrices
- Ch. 2. Polynomial toolbox ← computer session
- Ch. 3. Polynomials in control systems
- Ch. 4. Discrete-time systems
- Ch. 5. Continuous-time and MIMO systems
- Ch. 6. CAD based on polynomial methods ← computer session
- Ch. 7. Future perspectives

Overview

Ch. 3. Polynomials in control systems

IO systems and polynomials
  - LTI systems
  - Example: RLC circuit
  - System equations
  - State equations
  - Descriptor equations
  - IO equations
  - SS and IO
  - Hidden modes
  - Descriptor and IO

Properties of SISO IO
  - Solution of IO
  - poles and zeros
  - properness
  - realization

Feedback SISO systems
  - Plant
  - Controller
  - Feedback systems
  - Closed-loop char. pol.
  - Analysis and synthesis
  - Polynomial equation
  - Preview of design tasks

Running examples
  - Four simple plants
  - pend
  - ball
  - dpend
  - heli
  - instructions

Input-output
systems and polynomials

LTI systems
Example: RLC circuit
Input-output model
System equations
State equations
Descriptor equations
IO equations
SS and IO
Hidden modes
Descriptor and IO

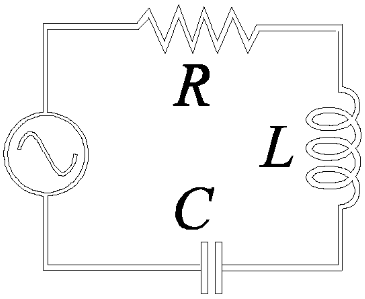
LTI systems

Modelling of linear systems from first principles typically leads to systems of linear differential or difference equations in

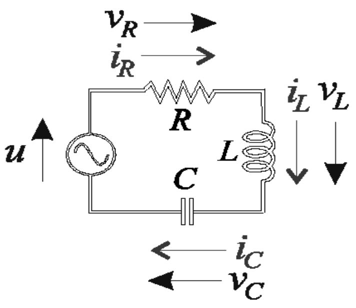
- outputs
- inputs
- latent (internal) variables

Modelling Example

Example: Electrical circuit



Circuit Model Equations 1



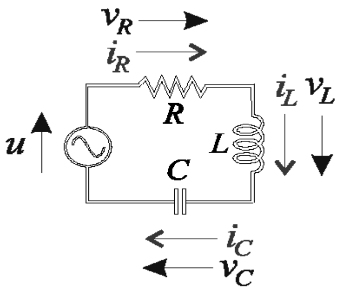
- Element equations

$$R: \quad v_R = Ri_R$$

$$L: \quad v_L = L\frac{di_L}{dt}$$

$$C: \quad i_C = C\frac{dv_C}{dt}$$

Circuit Model Equations 2



- Interconnection equations
  Kirchhoff voltage law

$$u = v_R + v_L + v_C$$

Kirchhoff current law

$$i_R = i_L \quad i_L = i_C$$

- Output equation

$$y = v_L$$

Circuit Model Equations 3

Organize as

$$\begin{bmatrix} -R & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -L\frac{d}{dt} & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -C\frac{d}{dt} & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & -1 & 0 \\ 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} i_r \\ v_r \\ i_l \\ v_L \\ i_C \\ v_C \\ u \\ y \end{bmatrix} = 0$$

Circuit Model Equations 4

May rewrite this equation in the form

$$M(\frac{d}{dt}) \begin{bmatrix} w \\ u \\ y \end{bmatrix} = 0$$

latent variables

that imposes a relation on the signals involved in the system (behavioral view)

polynomial matrix

$$M(s) = \begin{bmatrix} -R & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -Ls & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -Cs & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & -1 & 0 \\ 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

System equations

There are several useful ways of arranging the system equations

- state equations
- descriptor equations
- IO-equations

State equations

State equations

$$\dot{x} = Ax + Bu$$

$$y = Cx + Du$$

- Extensively studied
- Many useful properties
- There often is a direct model interpretation

In polynomial matrix form

$$\begin{bmatrix} sI - A & -B & 0 \\ -C & -D & I \end{bmatrix} \begin{bmatrix} x \\ u \\ y \end{bmatrix} = 0$$

Descriptor representation

Descriptor equations

$$E\dot{x} = Ax + Bu$$

$$y = Cx + Du$$

E may be singular

In polynomial matrix form

$$\begin{bmatrix} sE - A & -B & 0 \\ -C & -D & I \end{bmatrix} \begin{bmatrix} x \\ u \\ y \end{bmatrix} = 0$$

- Typically arises from first principle modelling
- Many useful properties
- Well studied

IO equation

Input-output equation

By eliminating all latent (internal) variables a set of differential equations in the output y and input u results:

$$p\left(\frac{d}{dt}\right) y = q\left(\frac{d}{dt}\right) u$$

polynomials

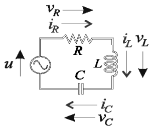
For zero initial conditions:

$$p(s)y = q(s)u$$

$$y = \frac{q(s)}{p(s)}u$$

polynomial fraction: transfer function

Circuit example



$$LC\frac{d^2y}{dt^2}+RC\frac{dy}{dt}+y=LC\frac{d^2u}{dt^2}$$

$$p(s)y=q(s)u$$

$$p(s)=LCs^2+RCs+1$$
$$q(s)=LCs^2$$

SS and IO representation - 1

Computation of the IO representation

$$\dot{x}=Ax+Bu \qquad x(0)=0$$
$$y=Cx+Du$$

Using Laplace transform $\mathcal{L}(\dot{x})=sx(s)-x(0)=sx$

$$\left.\begin{aligned}sx&=Ax+Bu\\ y&=Cx+Du\end{aligned}\right\}\Rightarrow\left.\begin{aligned}(sI-A)x&=Bu\\ y&=Cx+Du\end{aligned}\right\}\Rightarrow\begin{aligned}x&=(sI-A)^{-1}Bu\\ y&=Cx+Du\end{aligned}$$

$$\Rightarrow\; y(s)=\boxed{\left(C(sI-A)^{-1}B+D\right)}u(s)$$

Transfer function

$$h(s)=C(sI-A)^{-1}B+D=\frac{q(s)}{p(s)}$$

$$p(s)=\det(sI-A)$$

SS and IO representation - 2

Computation of the IO representation

$$\dot{x} = Ax + Bu \qquad x(0) = x_0$$

$$y = Cx + Du$$

Using Laplace transform $\mathcal{L}(\dot{x}) = sx(s) - x(0) = sx(s) - x_0$

$$y(s) = \boxed{\left(C(sI - A)^{-1}B + D\right)} u(s) + \boxed{C(sI - A)^{-1}x_0}$$

$$y(s) = \frac{q(s)}{p(s)} u(s) + \frac{r_{x_0}(s)}{p(s)}$$

r_{x_0} is not used in design

$$p(s) = \det(sI - A)$$

SS and IO representation - 3

- Poles corresponding to unobservable modes do not exist in $y(s)$ and, hence, are not roots of $p(s)$
- Poles corresponding to uncontrollable modes do exist in $y(s)$ and, hence, are roots of $p(s)$

Conclusion: To obtain the correct IO representation

- cancel all identical pole-zero pairs in $q(s)/p(s)$ that correspond to unobservable modes
- retain all identical pole-zero pairs that correspond to uncontrollable pole-zero pairs

Hidden modes

Hidden modes:

Let $p(\frac{d}{dt})y = q(\frac{d}{dt})u$ is IO model of $\begin{aligned}\dot{x} &= Ax + Bu\\ y &= Cx + Du\end{aligned}$

that is $\frac{q(s)}{p(s)} = \left(C(sI - A)^{-1}B + D\right),\ \frac{r_{x_0}(s)}{p(s)} = C(sI - A)^{-1}x_0$

We say that it has no hidden modes iff

$$p(s) = \det(sI - A)$$

Equivalent expressions:

- Nothing has been cancelled during computation
- “What you see is what you have”

Convention

It is always assumed that there is
no common factor present (at the same time)
in all the polynomials p, q, r_{x_0}

Remember that r_{x_0} is a “set” !

Then no hidden modes requirement means

- the system is observable
- the system is controllable iff p and q are coprime

SS and IO representation - 4

Facts about the IO system

$$p(s)y = q(s)u$$

- The system is always observable
- It is controllable if and only if $p(s)$ and $q(s)$ are coprime (that is, have no common roots)

SS and IO representation - 5

Example

$$x = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -2 & 0 \\ 0 & 0 & -3 \end{bmatrix} x + \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} u$$

$$y = \begin{bmatrix} 1 & 0 & 1 \end{bmatrix} x$$

Poles

- –1 corresponds to a controllable and observable mode

cancel

- –2 corresponds to a controllable but unobservable mode

retain

- –3 corresponds to an uncontrollable but observable mode

SS and IO representation - 6

$$\det(sI - A) = (s+1)(s+2)(s+3)$$
$$q(s) = (s+2)(s+3)$$

Canceling the pole at –2 we obtain the equivalent IO representation

$$p(s)y = q(s)u$$

with

$$p(s) = (s+1)(s+3)$$
$$q(s) = (s+3)$$

Descriptor and IO representation

Computation of the IO representation from

$$E\dot{x} = Ax + Bu \qquad x(0) = x_0$$
$$y = Cx + Du$$

$$y(s) = \boxed{\left(C(sE-A)^{-1}B + D\right)}u(s) + \boxed{C(sE-A)^{-1}x_0}$$

$$\swarrow \qquad\qquad \swarrow$$

$$y(s) = \frac{q(s)}{p(s)}\, u(s) + \frac{r_{x_0}(s)}{p(s)} \qquad p(s) = \det(sE - A)$$

Properties
of SISO input-output systems

[PolyX]

Solution of IO equations
Poles and zeros
Stability
Realization

Solution of IO equations

[PolyX]

Consider scalar IO equation

$$p\left(\frac{d}{dt}\right)\,y = q\left(\frac{d}{dt}\right)\,u$$

Facts

- General solution = particular solution + solution of the homogenous equation

multiplicity of λ_i

distinct roots of

- Homogenous solution = $\sum_{i=1}^{k}\sum_{j=0}^{m_i-1}\alpha_{ij}t^k e^{\lambda_i t}$

constants determined by initial conditions

Solution of IO equations - 2

Facts - continued

- Particular solution

$$y_{\text{part}}(t) = \int_0^\infty h(\tau)u(t-\tau)\,d\tau$$

impulse response = inverse Laplace transform of the transfer function

$$h(s) = \frac{q(s)}{p(s)}$$

Poles and zeros

The IO system

$$p(\frac{d}{dt})\,y = q(\frac{d}{dt})\,u$$

has the transfer function

$$h(s) = \frac{q(s)}{p(s)}$$

- Poles of the system: Roots of p
- Zeros of the system: Roots of q

Poles and zeros - 2

Facts:

- Fundamental Theorem of Algebra
  A polynomial of degree n

$$p(s) = p_0 + p_1 s + p_2 s^2 + \cdots + p_n s^n, \quad p_n \neq 0$$

has exactly n roots (when counted with multiplicities).

- Complex conjugate pairs
  If all the coefficients p_i are real, then complex roots come in complex conjugate pairs $(\alpha + i\beta, \alpha - i\beta)$

Poles and zeros - 3

Definition:

- Poles/zeros at Infinity
  Transfer function

$$h(s) = \frac{q(s)}{p(s)}$$

has a pole/zero at infinity iff the transfer function

$$h(1/s)$$

has a pole/zero at 0.

Poles and zeros - 4

Facts: about $h(s)=\frac{q(s)}{p(s)}$

- If $\deg p(s) > \deg q(s)$ then it has at infinity a zero with multiplicity $k = \deg p(s) - \deg q(s)$
- If $\deg p(s) < \deg q(s)$ then it has at infinity a zero with multiplicity $l = \deg q(s) - \deg p(s)$
- When taking poles/zeros at infinity into account, then

  No. of poles = No. of zeros

Stability

Reminder:

The IO system has no hidden modes iff the charact. polynomial of its SS realization equals its denominator polynomial, i.e.

$$\det(sI - A) = p(s)$$

Fact:

The IO is stable iff

- all the roots of p have strictly negative real parts

Equivalent terminology

- all the roots of p are in the open left-half plane
- p is Hurwitz

Properness

Transfer function

$$h(s) = \frac{q(s)}{p(s)}$$

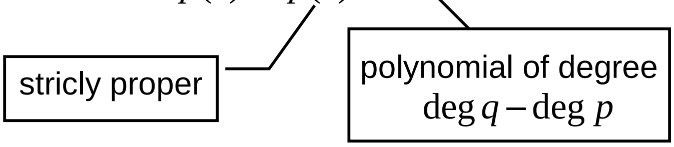
is

- proper if $\deg p(s) \geq \deg q(s)$
- strictly proper if $\deg p(s) > \deg q(s)$
- improper if $\deg p(s) < \deg q(s)$
- biproper if $\deg p(s) = \deg q(s)$

Properness - 2

Can always write

$$h(s) = \frac{q(s)}{p(s)} = \frac{r(s)}{p(s)} + d(s)$$



stricly proper

polynomial of degree $\deg q - \deg p$

- A strictly proper system has no direct feedthrough
- An nonproper system has “derivative feedthrough”

Realization of IO system

Given a strictly proper n th order scalar IO system

$$p(\frac{d}{dt})\,y = q(\frac{d}{dt})\,u$$

how does one construct an equivalent state system?

Well-known realization:

Define the state variables

$$x_i = \frac{s^{i-1}}{p(s)}u, \quad i = 1, 2, \ldots, n$$

Realization of IO system - 2

If $p(s) = p_0 + p_1 s + \cdots + p_{n-1}s^{n-1} + s^n$

$q(s) = q_0 + q_1 s + \cdots + q_{n-1}s^{n-1}$

than we have the controllable realization

companion matrix

$$\dot{x} = \begin{bmatrix} 0 & 1 & 0 & \ldots & \ldots & 0 \\ 0 & 0 & 1 & 0 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & \ldots & \ldots & \ldots & 0 & 1 \\ -p_0 & -p_1 & \ldots & \ldots & \ldots & -p_{n-1} \end{bmatrix} x + \begin{bmatrix} 0 \\ 0 \\ \ldots \\ 0 \\ 1 \end{bmatrix} u$$

$$y = \begin{bmatrix} q_0 & q_1 & \ldots & \ldots & \ldots & q_{n-1} \end{bmatrix} x$$

equivalent to the IO systems if p and q are coprime.

Realization of IO system - 3

Dual realization

$$\dot{x} = \begin{bmatrix} 0 & 0 & 0 & \dots & \dots & -p_0 \\ 1 & 0 & 0 & 0 & \dots & -p_1 \\ 0 & 1 & 0 & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 0 & 1 & -p_{n-1} \end{bmatrix} x + \begin{bmatrix} q_0 \\ q_1 \\ \dots \\ \dots \\ q_{n-1} \end{bmatrix} u$$

$$y = \begin{bmatrix} 0 & 0 & \dots & \dots & 0 & 1 \end{bmatrix} x$$

This observable realization is equivalent to the IO system, also if it is not controllable (if p and q are not coprime)

Feedback SISO systems

Feedback systems
Closed-loop characteristic polynomial
Analysis and synthesis
Polynomial equation
Preview of design tasks

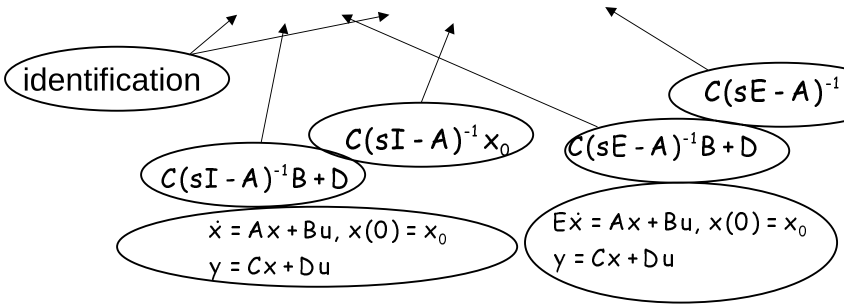
Plant

Plant

The given system to be controlled

- No hidden modes
- Don’t care about c_{x_0}

$$y(s)=\frac{b(s)}{a(s)}u(s)+\frac{c_{x_0}(s)}{a(s)}$$



identification

$C(sI-A)^{-1}x_0$

$C(sI-A)^{-1}B+D$

$C(sE-A)^{-1}x_0$

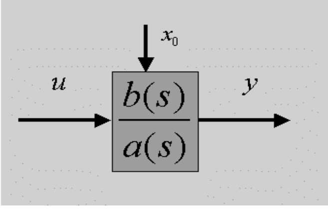
$C(sE-A)^{-1}B+D$

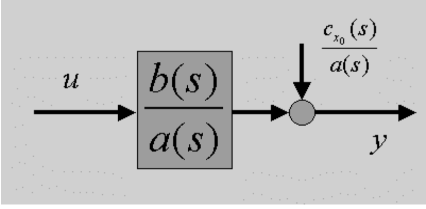
$\dot{x}=Ax+Bu,\ x(0)=x_0$
$y=Cx+Du$

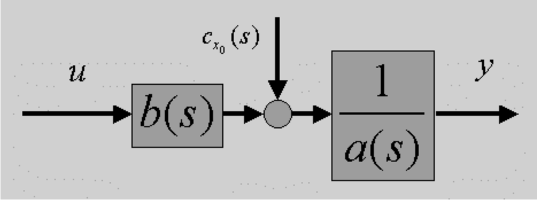
$E\dot{x}=Ax+Bu,\ x(0)=x_0$
$y=Cx+Du$

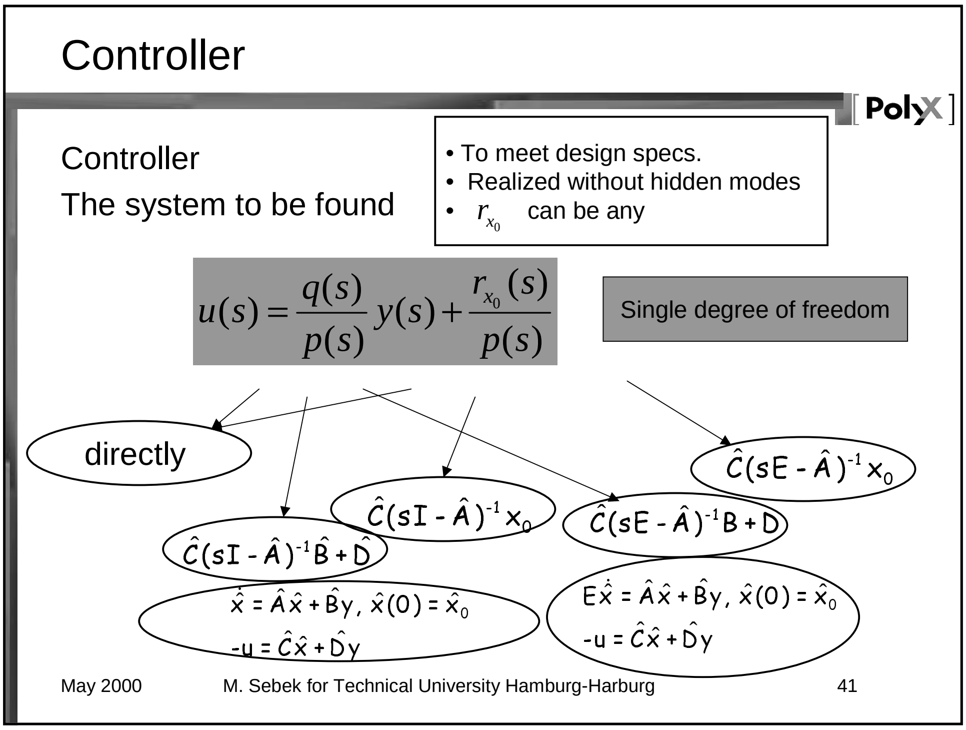
Plant

Some equivalent pictures









Controller

Controller

The system to be found

- To meet design specs.
- Realized without hidden modes
- r_{x_0} can be any

$$u(s) = \frac{q(s)}{p(s)} y(s) + \frac{r_{x_0}(s)}{p(s)}$$

Single degree of freedom

directly

$\hat{C}(sE - \hat{A})^{-1} x_0$

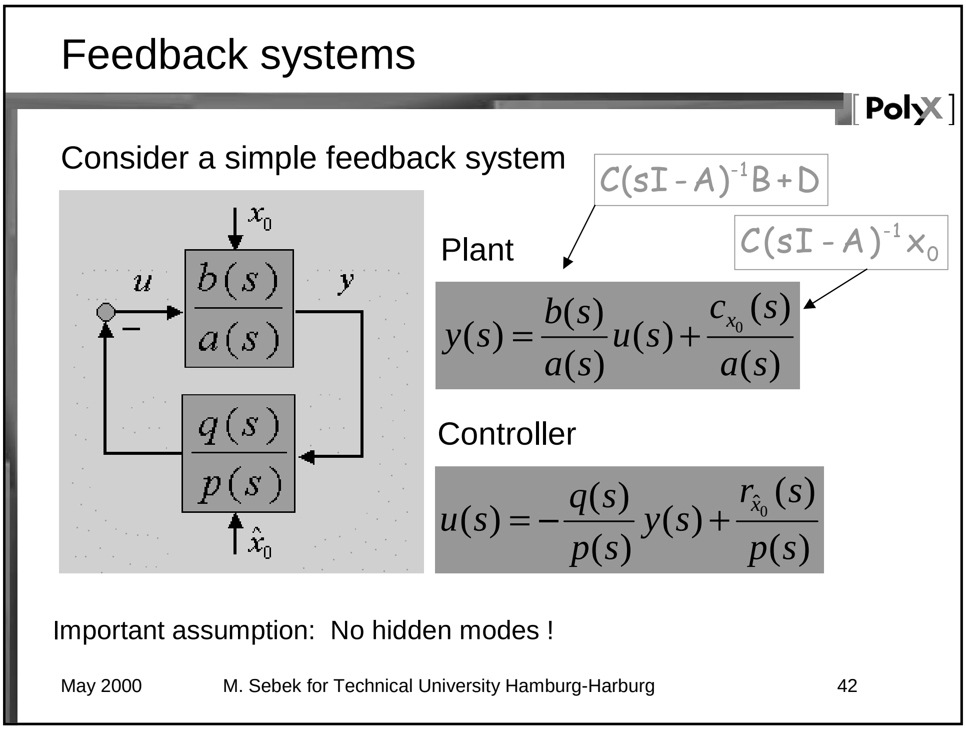
$\hat{C}(sI - \hat{A})^{-1} x_0$

$\hat{C}(sE - \hat{A})^{-1} B + D$

$\hat{C}(sI - \hat{A})^{-1} \hat{B} + \hat{D}$

$\dot{\hat{x}} = \hat{A}\hat{x} + \hat{B}y,\ \hat{x}(0) = \hat{x}_0$
$-u = \hat{C}\hat{x} + \hat{D}y$

$E\dot{\hat{x}} = \hat{A}\hat{x} + \hat{B}y,\ \hat{x}(0) = \hat{x}_0$
$-u = \hat{C}\hat{x} + \hat{D}y$



Feedback systems

Consider a simple feedback system

$C(sI - A)^{-1}B + D$

$C(sI - A)^{-1} x_0$

Plant

$$y(s) = \frac{b(s)}{a(s)} u(s) + \frac{c_{x_0}(s)}{a(s)}$$

Controller

$$u(s) = -\frac{q(s)}{p(s)} y(s) + \frac{r_{\hat{x}_0}(s)}{p(s)}$$

Important assumption: No hidden modes !

Feedback systems

$$y = \frac{b}{a}u + \frac{c_{x_0}}{a}$$

$$y = -\frac{bq}{ap}y + \frac{br_{\hat{x}_0}}{ap} + \frac{pc_{x_0}}{ap}$$

$$(ap+bq)y = br_{\hat{x}_0} + pc_{x_0}$$

$$y = \frac{br_{\hat{x}_0} + pc_{x_0}}{ap+bq}$$

$$y = \frac{b}{\boxed{ap+bq}}r_{\hat{x}_0} + \frac{p}{\boxed{ap+bq}}c_{x_0}$$

$$u = -\frac{q}{p}y + \frac{r_{\hat{x}_0}}{p}$$

$$u = -\frac{bq}{ap}u - \frac{qc_{x_0}}{ap} + \frac{ar_{\hat{x}_0}}{ap}$$

$$(ap+bq)u = -qc_{x_0} + ar_{\hat{x}_0}$$

$$u = \frac{ar_{\hat{x}_0} - qc_{x_0}}{ap+bq}$$

$$u = \frac{a}{\boxed{ap+bq}}r_{\hat{x}_0} - \frac{q}{\boxed{ap+bq}}c_{x_0}$$

closed-loop characteristic polynomial

Feedback systems

When considered as a two-output system

$$\begin{bmatrix} y \\ u \end{bmatrix} = \begin{bmatrix} \dfrac{p}{ap+bq} & \dfrac{b}{ap+bq} \\ \dfrac{-q}{ap+bq} & \dfrac{a}{ap+bq} \end{bmatrix} \begin{bmatrix} c_{x_0} \\ r_{\hat{x}_0} \end{bmatrix}$$

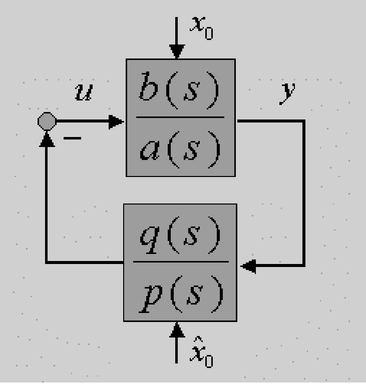
$$\begin{bmatrix} y \\ u \end{bmatrix} = \begin{bmatrix} p & b \\ -q & a \end{bmatrix}^{-1} \begin{bmatrix} c_{x_0} \\ r_{\hat{x}_0} \end{bmatrix}$$

Closed-loop characteristic polynomial

If there are no hidden modes in the plant and controller descriptions, then

$$a(s)p(s)+b(s)q(s)$$

is the characteristic polynomial of the closed loop



Analysis

Task A: Analysis

Given plant $a(s),b(s)$ and controller $p(s),q(s)$,

compute $a(s)p(s)+b(s)q(s)$ and analyze it.

May check

- stability
- position of poles
- robustness
- ....

Synthesis

Task B: Synthesis

Desired c-l characteristic polynomial

Given plant $a(s), b(s)$ and a polynomial $c(s)$,

compute a controller $p(s), q(s)$ so that

$$a(s)p(s) + b(s)q(s) = c(s)$$

The resulting controller guarantees that the closed-loop characteristic polynomial is $c(s)$ and hence has the desired properties!

Polynomial equation

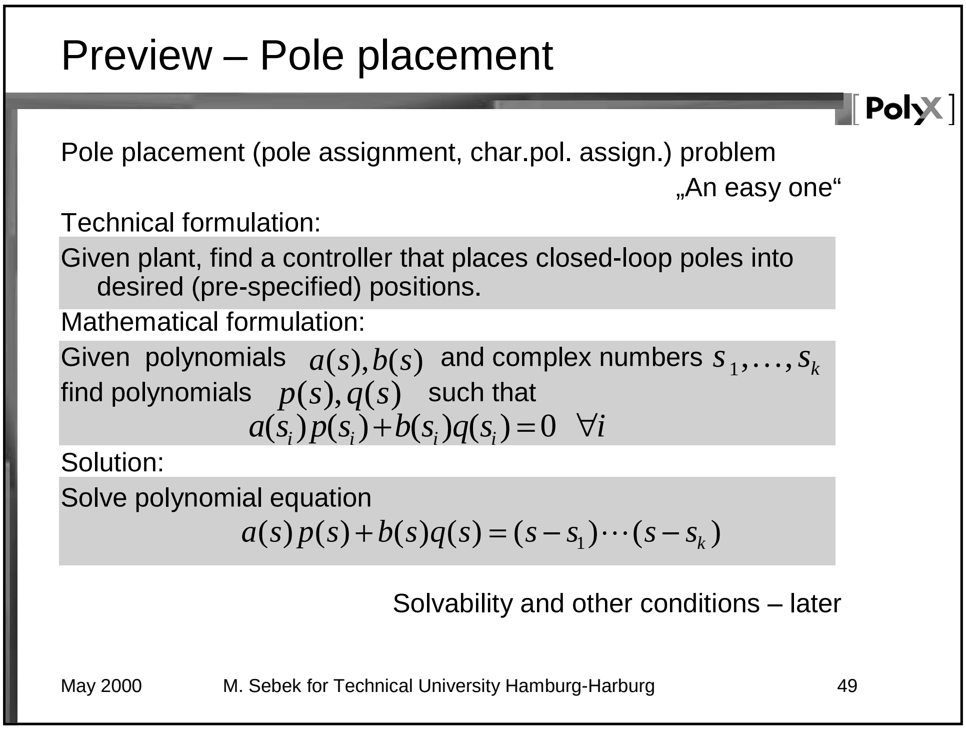
Basic problem:

Given $a(s), b(s)$ and $c(s)$, how one gets $p(s), q(s)$

such that $a(s)p(s) + b(s)q(s) = c(s)$???

It must be solved as an equation in polynomials !

This is a LINEAR POLYNOMIAL EQUATION



Preview – Pole placement

Pole placement (pole assignment, char.pol. assign.) problem

„An easy one“

Technical formulation:

Given plant, find a controller that places closed-loop poles into desired (pre-specified) positions.

Mathematical formulation:

Given polynomials $a(s), b(s)$ and complex numbers $s_1, \ldots, s_k$
find polynomials $p(s), q(s)$ such that

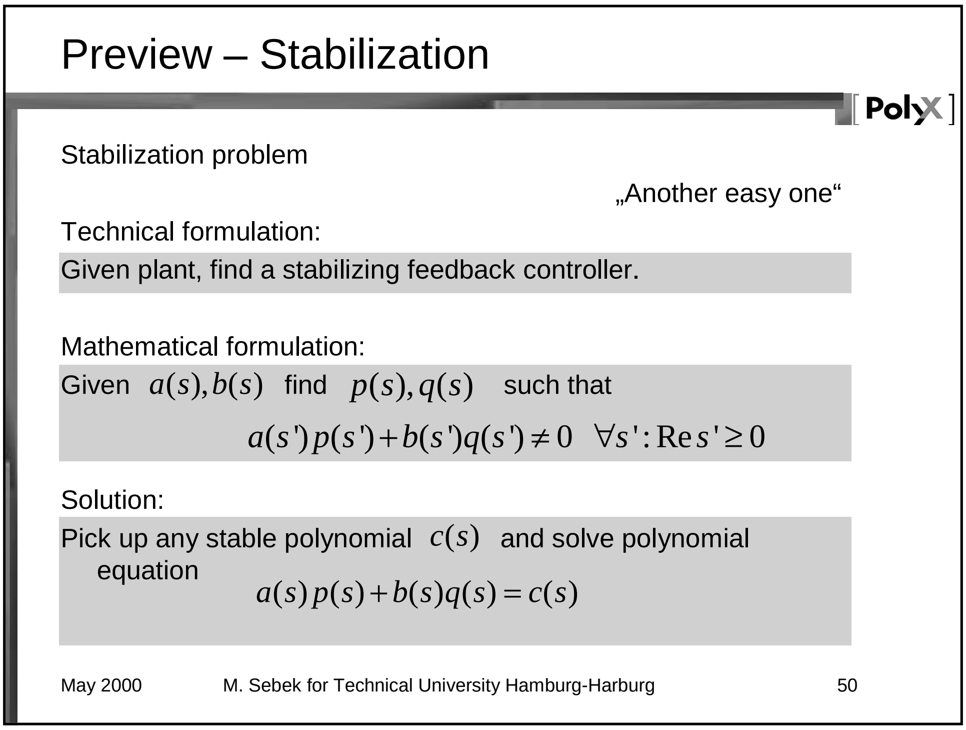
$$a(s_i)\,p(s_i) + b(s_i)q(s_i) = 0 \quad \forall i$$

Solution:

Solve polynomial equation

$$a(s)\,p(s) + b(s)q(s) = (s - s_1)\cdots(s - s_k)$$

Solvability and other conditions – later



Preview – Stabilization

Stabilization problem

„Another easy one“

Technical formulation:

Given plant, find a stabilizing feedback controller.

Mathematical formulation:

Given $a(s), b(s)$ find $p(s), q(s)$ such that

$$a(s')\,p(s') + b(s')q(s') \neq 0 \quad \forall s' : \mathrm{Re}\, s' \geq 0$$

Solution:

Pick up any stable polynomial $c(s)$ and solve polynomial equation

$$a(s)\,p(s) + b(s)q(s) = c(s)$$

Running examples

Four simple plants
pend
ball
dpend
heli
Instructions

Four simple plants

The following four simple plants

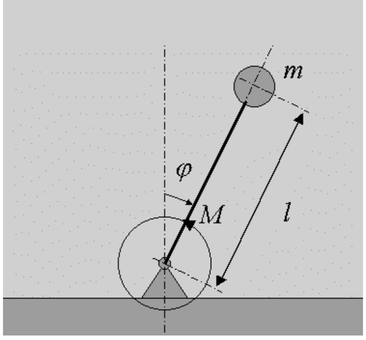
- PEND - undamped simple pendulum control
- BALL - ball & beam
- DPEND - damped double pendulum control
- HELI - helicopter azimuth control

will be used as running examples.

- Both C-T and D-T models provided.
- SISO (PEND and BALL), MIMO (DPEND) and uncertain SISO (HELI) systems covered.

PEND

Simple inverted pendulum (undamped)



- input: momentum
- output: angle

We suppose that

- friction in joint is negligible
- all mass is concentrated in the ball

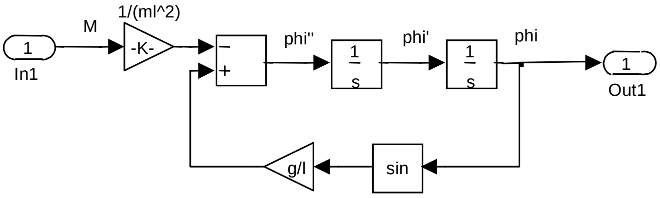
Then it is described by

$$ml^2 \frac{d^2\varphi}{dt^2} - mgl\sin(\varphi) = -M$$

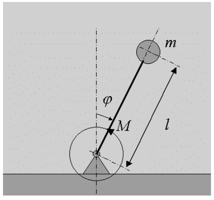
nonlinear differential equation

PEND – 2: Simulink model

Appropriate (nonlinear) SIMULINK model

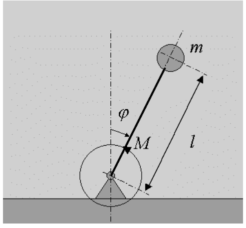


will be used for simulations of



PEND – 3: Linear model

For design, we shall use a linear model .



For

$$\varphi = 0,\ \left.\frac{d\varphi}{dt}\right|_0 = 0$$

and for

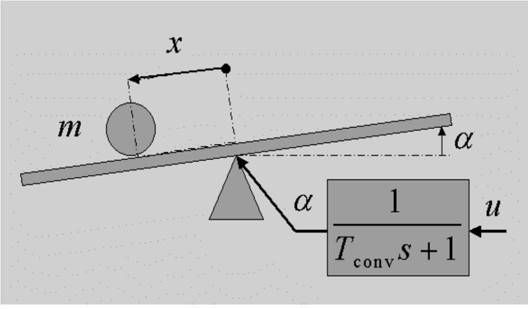
l = 1m, m = 1kg and g = 10ms^{-2}

it has a simple transfer function

$$g(s) = \frac{1}{10 - s^2}$$

BALL

Motor driven ball & beam



- influence of $\Delta\alpha$ on x neglected;
- equivalent single point mass

$$m_{red} = \frac{7}{5}m$$

Voltage driven motor:

$$\alpha(s) = \frac{1}{T_{\text{conv}}s + 1}u(s)$$

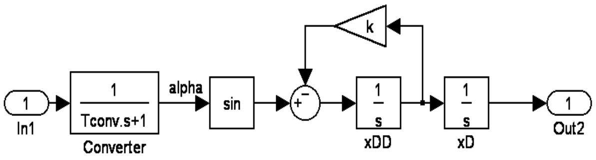
Ball & beam:

- k friction coefficient

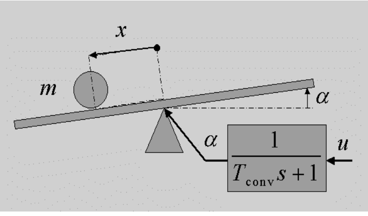
$$m_{red}\frac{d^2x}{dt^2} + k\frac{dx}{dt} = -mg\sin\alpha$$

BALL – 2: Simulink model

Appropriate (nonlinear) SIMULINK model

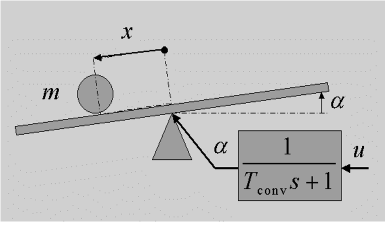


will be used for simulations of



BALL – 3: linear model

Linear model



Linearization at

$$\alpha = 0,\ x = 0,\ \dot{x} = 0$$

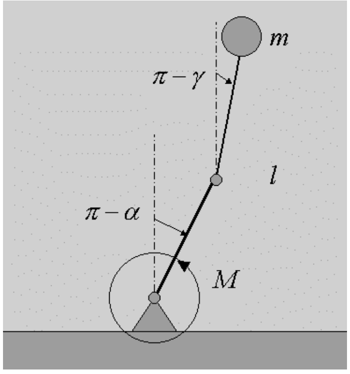
yields IO model

$$x(s) = -\frac{1}{(Ts+1)(s+k)s}u(s)$$

that will be used for design

DPEND

Double inverted pendulum (damped)



- input: momentum at 1. joint
- 2 outputs: 2 angles α,γ

We suppose that

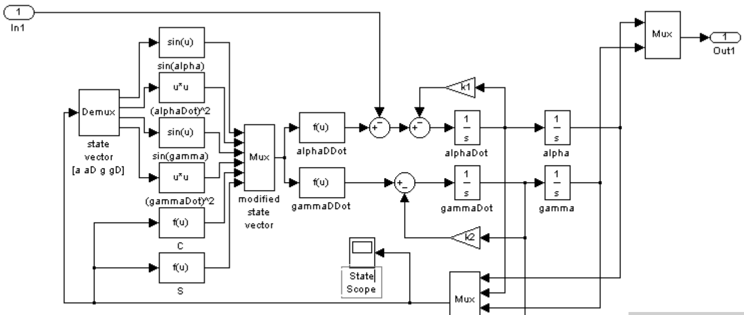
- moment influences only $\ddot{\alpha}$
- all mass is concentrated in the ball

Then it is described by a couple of nonlinear differential equations

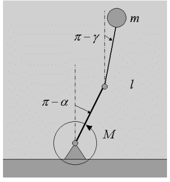
$$\ddot{\alpha}=f_\alpha(\alpha,\gamma,M)$$
$$\ddot{\gamma}=f_\gamma(\alpha,\gamma)$$

DPEND – 2: Simulink model

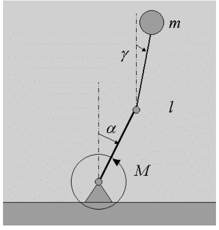
SIMULINK model of double pendulum (by the MathWorks)



will be used for simulations of



DPEND – 3: Linear model



Linearized at setpoint

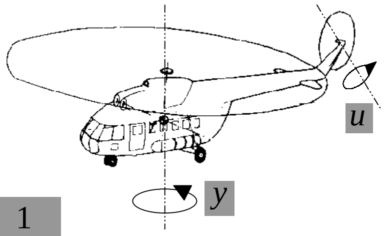
$$\alpha = \pi; \dot{\alpha} = 0; \gamma = \pi; \dot{\gamma} = 0;$$

$$\begin{bmatrix}\alpha(s)\\ \gamma(s)\end{bmatrix} = \begin{bmatrix}\dfrac{-5.7 + 0.2s + s^2}{-29 + 2.3s + 11s^2 - 0.4s^3 - s^4} \\ \dfrac{-2.9}{-29 + 2.3s + 11s^2 - 0.4s^3 - s^4}\end{bmatrix} M(s)$$

$$\begin{bmatrix}\alpha(s)\\ \gamma(s)\end{bmatrix} = \begin{bmatrix}\dfrac{-(s+2.493)(s-2.293)}{(s+2.883)(s+2.025)(s-2.683)(s-1.825)} \\ \dfrac{2.8571}{(s+2.883)(s+2.025)(s-2.683)(s-1.825)}\end{bmatrix} M(s)$$

Heli

Helicopter azimuth control



- linear model
  transfer function from
  angle of rear rotor blades
  to helicopter azimuth

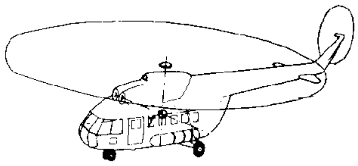
$$g(s) = \frac{1}{s}\frac{q}{(0.1s+q)}\frac{1}{s+r}$$

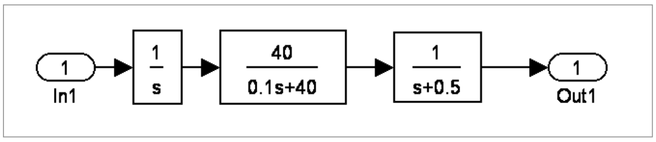
- with uncertain parameters

$$q_0 = 40 \quad (q \in [20,60])$$
$$r_0 = 0.5 \quad (r \in [0,1])$$

Heli

Simulink model (linear)





Model examples

Models and other related functions provided in directory model – download and put it on your path

Detailed description in the file Course tool usage

Each model example consists of the following steps

- plant activation
- control structure selection
- controller design
- simulation

Model activation

Model activation

- creates linear IO models of the plant
- both c-t $a(s)\backslash b(s)$ and d-t $a_z(z)\backslash b_z(z)$ models
- default sampling time $T_s = 0.1s$ unless defined before
- performed by particular model names

  `pend, ball, dpend, heli`

Control structure selection

Control structure selection

- creates control loop(s) for currently active model
- also creates a relevant Simulink model (to see it, click on the plant box in the Simulink control structure)
- also creates animation window (for pend and dpend only)
- performed by particular structure names

  `regul, track, dist, noise`

  or via menu

  `simul`

  or via model related macros

  `dpendi,...`
- expects controller model to be provided via polynomials **`p,q`** or **`p,q,r`**

Controller design

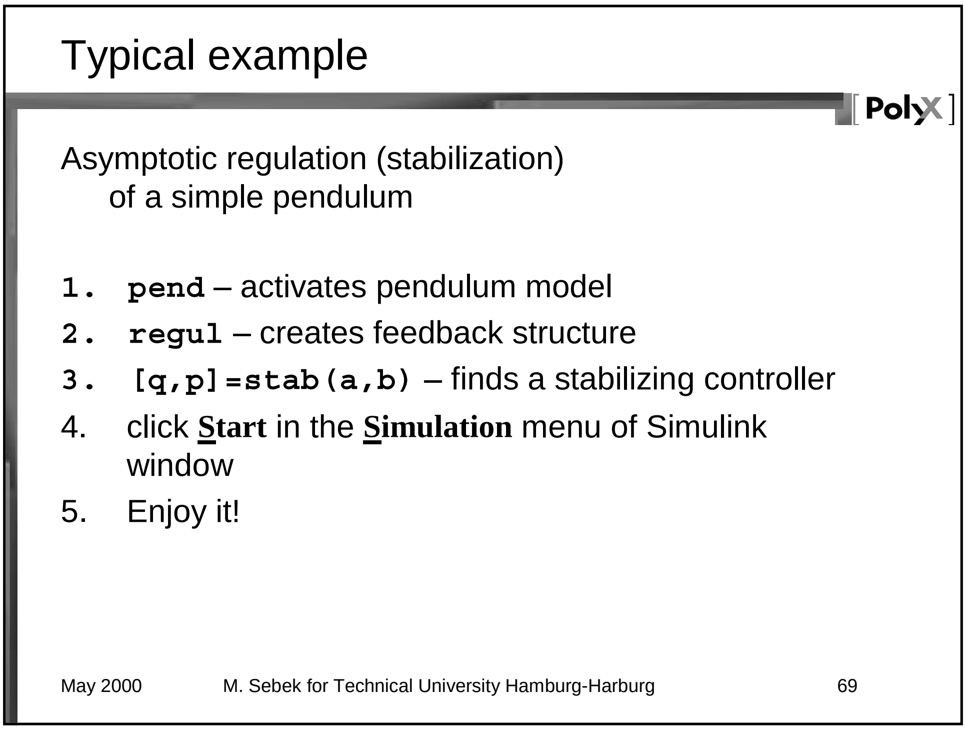
Controller design

- uses linearized models (c-t or d-t)
- creates controller model defined by polynomials
  `p,q` or `p,q,r`
- can be done manually using Polynomial Toolbox commands (`pplace,stab,lqg,...`) or even polynomial equation solvers (`axbyc`,...)
- can also be done via active model related macros (`pp,db,lqg`) or model specific macros, (e.g. `dpendpp, helipp`, ...)
- can be computed in menu style by `control`

Simulation

Simulation

- standard Simulink style
- can start after all prior steps are done
- initial conditions can be changed in Simulink or in Matlab workspace
- accompanied by animation when provided (for `pend,dpend`)



Typical example

Asymptotic regulation (stabilization)
of a simple pendulum

1. **`pend`** – activates pendulum model
2. **`regul`** – creates feedback structure
3. **`[q,p]=stab(a,b)`** – finds a stabilizing controller
4. click **Start** in the **Simulation** menu of Simulink window
5. Enjoy it!