Introduction
1  Modeling and simulation of chain systems – main concepts
2  Multi-pole models
3  Modeling and simulation of fluid power systems
4  Simulation of an electro-hydraulic load-sensing system
5  Simulation of an electro-hydraulic servo-system
6  Sizes and complexities
7  History
Summary
Introduction

Advanced technical chain systems represent complex **mechatronic products**, containing hydraulic, mechanical, pneumatic, electrical and signal processing elements.

Multi-pole models with different oriented causalities are used that enable adequately describe physical processes in hydraulic and mechanical systems.

Modeling is based on principles developed in **Institute of Machinery** of Tallinn University of Technology (TUT). CoCoViLa is used as a tool for modeling and simulation.

**Electro-hydraulic systems** are considered as examples. Proposed principles and methods are applicable for simulation of technical chain systems with any physical content.
1 Modeling and simulation of chain systems – main concepts

1.1 Models and causality

**Mathematical model (MM)** - a complex of mathematical dependences composed on base acting in system physical processes.

**Graphical model** - can be composed as principal scheme, object scheme, block scheme of mathematical models or as oriented graph (signal flow graph, graph of non-linear dependences, bond graph).

**Physical causality**

**Mathematical causality**

In principle, it is natural if the mathematical causality corresponds to the physical causality. The correspondence for the whole system is obligatory, but not always obligatory for the system components.

1.2 Requirements to models

1. Functional relations must adequately describe physical processes
2. Required cause-consequence relations between variables must be followed
3. Both direct actions and feedbacks in hydraulic and mechanical components must be considered
4. Simplicity of modification
5. Visual composing
1.3 Processes in mechatronic systems

Processes in mechatronic systems:

- **Static** – caused by static inputs
- **Steady-state** – caused by steady-state inputs
- **Dynamic** – caused by inputs depending on time (disturbances)

Main disturbances of dynamic responses:

<table>
<thead>
<tr>
<th>theoretical</th>
<th>practical</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Impulse</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Sine curve</strong></td>
<td></td>
</tr>
</tbody>
</table>

Disturbances cause transient responses or continuous oscillations.
1.4 Modeling and simulation systems

Functional and component schemes, causal modeling

Systems: SimHydraulics™ MATLAB/Simulink package, ITI SimulationX, AmeSim, EASY5, VisSim, DSHplus, BATHfp, WINSIMU, Automation Studio™, HYDRO ANALYST, DYNAST, MS1™, HYVOS 7.0.

Equations are automatically constructed on the basis of network schema with mechanical, hydraulic and electrical elements. Large equation systems are to be solved.

Mostly the unidirectional graphical models for equation construction are used which are not suitable for hydraulic and mechanical components, containing feedback.

Functional and component schemes, non-causal modeling

Dymola includes a graphical object-oriented modeling based on unified object-oriented language Modelica for non-causal modeling of physical systems.

In HOPSAN bi-directional delay line (DTL) elements with time delay are used to connect elements.

Bond graphs

The key of bond graph modeling is the representation (by a bond) of power as the product of efforts and flows. Bond graphs are not oriented to components, but to bond graph elements. The bond graph elements are expressed as two-pole elements. Therefore, feedbacks are not taken into account.

1.5 Simulation methods

Mainly integration of ordinary differential equations (ODE) or differential-algebraic equations (DAE) takes place by a solver.

Most of numerical techniques are based on methods of differences. Runge–Kutta methods especially the fourth-order classical method are mostly used.
2 Multi-pole models

**Two-pole model** - represents the mathematical relation between input and output values (poles).

**Two-pole model** represents the relation between potential variables \( (A) \), between flow variables \( (B) \) or between different types of variables.

**Multi-pole model** - represents mathematical relations between several input and output values (poles).

Four forms of mathematical causality of **four-pole model** are in use. Letters \( G, H, Y \) and \( Z \) as in electrical engineering denote them.

<table>
<thead>
<tr>
<th>Element</th>
<th>Potential variable</th>
<th>Flow variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mechanical translation</td>
<td>Force ( F, N )</td>
<td>Displacement ( x, m )</td>
</tr>
<tr>
<td></td>
<td>Velocity ( v, m/s )</td>
<td>Velocity ( v, m/s )</td>
</tr>
<tr>
<td></td>
<td>Acceleration ( a, m/s^2 )</td>
<td>Acceleration ( a, m/s^2 )</td>
</tr>
<tr>
<td>Mechanical rotation</td>
<td>Torque ( M, N )</td>
<td>Rotation angle ( \varphi, \text{rad} )</td>
</tr>
<tr>
<td></td>
<td>Angular velocity ( \omega, \text{rad/s} )</td>
<td>Angular velocity ( \omega, \text{rad/s} )</td>
</tr>
<tr>
<td></td>
<td>Angular acceleration ( \varepsilon, \text{rad/s}^2 )</td>
<td>Angular acceleration ( \varepsilon, \text{rad/s}^2 )</td>
</tr>
<tr>
<td>Hydraulic</td>
<td>Pressure ( p, \text{Pa} )</td>
<td>Volumetric flow rate ( Q, \text{m}^3/\text{s} )</td>
</tr>
<tr>
<td>Electric</td>
<td>Voltage ( U, V )</td>
<td>Current ( I, A )</td>
</tr>
</tbody>
</table>

Two- and four-pole models
Hydraulic piston in cylinder with three pairs of variables:
\[ p_1, Q_1, p_2, Q_2, x \text{ (or } v) , F, \]

where
\[ p_1, p_2 \] pressures in the cylinder chambers,
\[ Q_1, Q_2 \] volume flow rates in cylinder chambers,
\[ x, v \] position and velocity of the piston rod,
\[ F \] force on the piston rod.

Four forms (causalities) of **six-pole models** for a hydraulic piston in cylinder.
3 Modeling and simulation of fluid power systems

Mathematical models of components are presented as multi-pole models.

Multi-pole model usually contains parameters, inner and outer variables (poles) and relations between variables.

Multi-pole models are implemented as visual classes in CoCoViLa environment.

Relations between variables are described as CoCoViLa equations or user-defined Java methods.

In relations state variables, inner iterations and CoCoViLa subtasks are used.

Using the visual programming environment enables one to describe graphically simulation tasks, automatically construct algorithms and perform computations.

When solving specific simulation task, model has to be adjusted by evaluating parameters of the components and adding sources that describe constant inputs and disturbances.
3.1 Computing process organization

State variables are introduced for functional elements to characterize the features of the element at each simulation step.

Simulation process starts from initial state (initstate) and includes calculation of next state (nextstate) from previous states (usually from prevstate and state). As a result of simulation final state (finalstate) is computed.

\[
\text{[prevstate, state} \rightarrow \text{nextstate], initstate} \rightarrow \text{finalstate \{process\}}
\]

The similar technique is used for calculating variables in loop dependences. One variable in each loop is split and an iteration statement for calculating variables in loops is specified:

\[
\text{[primset} \rightarrow \text{secondset], initset} \rightarrow \text{finalset \{iterator\}}
\]

where

- initset – set of approximate initial values of variables in loop,
- primset, secondset – sets of initial and recomputed split variables,
- finalset – set of variables in loop, computed using iterations.

For example in the model a loop dependence takes place.

\[\text{TM.th1} \rightarrow \text{NF.th1} \rightarrow \text{NF.Thd} \rightarrow \text{TM.Thd} \rightarrow \text{TM.th1}\]

All the loop dependences can be solved if variables \(\text{TM.th1}, \text{TM.x1}, \text{SP.th2}, \text{SP.v}\) are split and calculated using built-in iteration method.

State variables and split variables must be described in component models.

When building a particular simulation task model and performing simulations state variables and split variables are handled and used automatically.
3.2 Modeling and simulation systems developed in CoCoViLa environment

- **Hydraulic elements** (tubes, resistors, volume elasticities, interface elements) and **hydraulic circuits**
- **Electro-hydraulic load-sensing systems**
- **Electro-hydraulic servo-valves and servo-systems**

Library has been developed containing more than 100 CoCoViLa visual classes implementing multi-pole models of different components.
4 Functional scheme and multi-pole model of an electro-hydraulic load-sensing system

Pump with controller:
- PV - Variable displacement axial-piston pump
- ME - Elektric motor
- CJh - Clutch
- VP - Control valve
- RVP, RVT - Throttle edges of the control valve
- Res1, Res2, Res5, Res6 - Resistors
- ZV - Positioning cylinder

Feeding circuit of hydraulic motor:
- Tube1, Tube2 - Tubes
- RIDW - Pressure compensator
- RVW - Measuring valve
- CV - Check valve
- RSKZ - Meter-in throttle edge of directional valve
- Res3, Tube3; Res4, Tube4 - Silencers

Hydraulic motor MH
Outlet circuit of the hydraulic motor:
- RSKA - Meter-out throttle edge of directional valve
- CJhM - Clutch
**Components:**
- Spool valve
- Spool valve inflow slot
- Spool valve outflow slot
- Constant resistor
- Positioning cylinder
- Swash plate with spring

**Throttle edges:**
- Measuring throttle edge $R_{vw}$
- Pressure compensator throttle edge $R_{idw}$
- Meter-in throttle edge $R_{sk-zu}$
- Meter-out throttle edge $R_{sk-r}$
4.3 Mathematical models

4.3.1 ZV - Positioning cylinder with pump swash plate

Static displacement of the swash plate spring
\[ x = f_0 - F_c / c \]
where \( F_c = A * p_1 / k_{rp} - F - F_f \);
\[ A = P * D^2 / 4; \quad F_f = F_{f0} + k_f * p_1 \]

Difference of the swash plate velocity in the spot of spring
\[ d_v = \delta * (F_c - (f_0 - x) * c - (F_{f0} + k_f * p_1) * \text{signum}(v) - h * v) / m \]

Difference of the swash plate displacement in the spot of spring
\[ d_x = - \delta * v \]

Displacement angle of the swash plate
\[ a_l = \text{atan} (x / rz) \]

Volumetric flow to the positioning cylinder
\[ Q_1 = A * v \]

Parameters: \( f_0 = 0.0265 \text{ m}, \quad c = 20874 \text{ N/m}, \quad D = 0.018 \text{ m}, \quad F_{f0} = 0.5 \text{ N}, \quad \]
\( k_f = 5 \times 10^{-6} \text{ m}^2, \quad k_{rp} = 0.833, \quad rz = 0.065 \text{ m}, \quad al_{max} = 0.3264 \text{ rad}, \]
\( m = 0.5 \text{ kg}, \quad h = 10 \text{ N/(m/s)} \).
4.3.2 Description of positioning cylinder ZV in CoCoViLa

```java
// Description: Positioning cylinder ZV

class ZV {
  double pi, f, a1, q1;
  // Positioning parameters
  double t, tmax;
  double Dz, oz, fs0, krf, kfr, Pfr0, kfr0, Pfr;
  double almin, almax, xzmax, pm;
  double inizv, iniav1, iniav2;
  double Dz2, oz2, fs02, krf2, kfr2, Pfr02, kfr02, Pfr2;
  double almin2, almax2, xzmax2, pm2;
  double iniav12, iniav22;
  double Dz, oz, fs0, krf, kfr, Pfr0, kfr0, Pfr;
  double almin, almax, xzmax, pm;
  double inizv, iniav1, iniav2;

  // State variables
  double intav1, intav12;
  double oldav1, oldav12;
  double val1, val12;
  double nextval1, nextval12;
  double ffr, msv, hvz;
  double pi;

  // Collecting output
  double[][] out = {q1, a1, Q1};

  // Evaluating initial state components
  initav1 = iniav1;
  initav12 = iniav12;

  // Method specification
  double Dz, oz, fs0, krf, kfr, Pfr0, kfr0, Pfr;
  double almin, almax, xzmax, pm, inizv, iniav1, iniav2;

  // Default conditions
  Dz = 0.018;
  oz = 17800;
  fs0 = 0.0318;
  Pfr0 = 1.5;
  kfr = 1e-6;
  kfr0 = 1e-4;
  inizv = 0;
  iniav1 = 0.0;
  iniav2 = 0.0;

  // Preparing output
  result[0][0] = Q1;
  result[0][1] = a1;

  // Calculation of dynamics
  if (tau > 0) {
    result[1][0] = result[0][0] - result[0][1];
    result[1][1] = (result[0][0] - result[0][1]) / kfr;
  }

  // Computing of the Runge-Kutta coefficient
  if (tau > 0) {
    result[1][0] = result[0][0] - result[0][1];
    result[1][1] = (result[0][0] - result[0][1]) / kfr;
  }

  // Computing of the next state values
  if (tau > 0) {
    result[1][0] = result[0][0] - result[0][1];
    result[1][1] = (result[0][0] - result[0][1]) / kfr;
  }

  // Preparing output
  nextstate
  result[0][0] = Q1;
  result[0][1] = a1;
}
```

```java
public double[] ZV_next (double[] x, double[] u) {
  // State variables
  double intav1, intav12;
  double oldav1, oldav12;
  double val1, val12;
  double nextval1, nextval12;
  double ffr, msv, hvz;
  double pi;

  // Collecting output
  double[][] out = {q1, a1, Q1};

  // Evaluating initial state components
  initav1 = iniav1;
  initav12 = iniav12;

  // Method specification
  double Dz, oz, fs0, krf, kfr, Pfr0, kfr0, Pfr;
  double almin, almax, xzmax, pm, inizv, iniav1, iniav2;

  // Default conditions
  Dz = 0.018;
  oz = 17800;
  fs0 = 0.0318;
  Pfr0 = 1.5;
  kfr = 1e-6;
  kfr0 = 1e-4;
  inizv = 0;
  iniav1 = 0.0;
  iniav2 = 0.0;

  // Preparing output
  result[0][0] = Q1;
  result[0][1] = a1;

  // Calculation of dynamics
  if (tau > 0) {
    result[1][0] = result[0][0] - result[0][1];
    result[1][1] = (result[0][0] - result[0][1]) / kfr;
  }

  // Computing of the Runge-Kutta coefficient
  if (tau > 0) {
    result[1][0] = result[0][0] - result[0][1];
    result[1][1] = (result[0][0] - result[0][1]) / kfr;
  }

  // Computing of the next state values
  if (tau > 0) {
    result[1][0] = result[0][0] - result[0][1];
    result[1][1] = (result[0][0] - result[0][1]) / kfr;
  }

  // Preparing output
  nextstate
  result[0][0] = Q1;
  result[0][1] = a1;
```
4.5 Simulating dynamics of a load-sensing system

**Inputs:**

*dynamic Source* \( y \):
- mean = 0.0045 m,
- step = -0.0015 m

*dynamic Source* \( M \):
- mean = 65 Nm,
- step = 20 Nm

*constant Source*:
- RVP: \( p_1 = 6 \times 10^6 \) Pa,
- RVT, ResY, TubeH: \( p_2 = 2 \times 10^4 \) Pa

**Simulation conditions:**
- \( \tau = 1/\delta = 1 \times 10^6 \) 1/s
- timesteps = 1.3E4

**Multi-pole models:**
- VP – Displacement of the pump control valve;
- RVP – Meter-in throttle edge of the pump control valve;
- IEH – Hydraulic interface elements;
- RVT – Meter-out throttle edge of the pump control valve;
- ZV – Positioning cylinder with swash plate;
- PV – Variable displacement pump;
- ME – Electric motor;
- CJh – Clutch with rotor of the pump;
- RIDVWlin – Measuring valve with pressure compensator;
- TubeH – Tubes;
- Tubeg – Tubes with closed end;
- RSKZ – Check valve and meter-in throttle edge of the hydraulic motor;
- MH – Hydraulic motor;
- RSKA – Meter-out throttle edge of the hydraulic motor;
- CJhM – Clutch with rotor of the hydraulic motor and drive mechanism.

**Inputs:**

*dynamic Source* – Values of the hydraulic motor load moment \( M \) and the directional valve displacement \( y \);
*constant Source* – Constant values.
Results of simulation of a load-sensing system dynamics

Conditions:
• Both a step change +10 Nm to the load moment of the drive mechanism 65 Nm and step change 0.0015 m to the displacement of the directional valve 0.0045 m were applied simultaneously (during 0.01 s).
• Time 1.3 s, timestep 10 μs.
• Results have been calculated for 130 000 points.

Observations:
• Dynamic responses damp in 1.3 s.
• Natural frequencies are about 5 Hz.
• Pressure at the outlet of the hydraulic motor carries frequency ~40 Hz because no hydraulic silencer is used at the outlet tube.
5 Simulation of an electro-hydraulic servo-system

Model of an electro-hydraulic servo-system is considered to contain three subsystems:

- Hydraulic servo-drive
- Electro-hydraulic servo-valve
- Constant pressure feeding system with safety valve

5.1 Hydraulic servo-drive
5.1.1 Functional scheme

Components:
- SP four-way sliding spool with slots 1...4
- CYL hydraulic cylinder
- PIS piston
- AC actuator
- TR position sensor
- D/A digital-analog transformer
- FBR feedback and regulator

Variables:
- z displacement of sliding spool
- x displacement of actuator
- v velocity of actuator
- F load force
- Uin control voltage
- Ufb feedback voltage
- Utm voltage to torque motor
5.1.2 Mathematical models of sliding spool slot pairs

Parameters: \(d\) – diameter of sliding spool (8 mm), \(y\) – lengths of working slots \((y = \pi dk, k = 0.67)\), \(r\) – radius of working slot edge (4 μm), \(s\) – width of radial slot of sliding spool (3.5 μm), \(z_1...z_4\) – overlaps of working slots (– 30 μm), \(z_m\) – maximal displacement of sliding spool (± 680 μm).

Variables: \(pp\) – feeding pressure, \(pt\) – pressure of return flow, \(pc_1, pc_2\) – controlled pressures, \(Q_1...Q_4\) – volumetric flows of working slots, \(Q_{c1}, Q_{c2}\) – controlled volumetric flows, \(z\) – displacement of sliding spool.

Sliding spool with overlapped slots enables to make volumetric flows minimal in central position of spool, i.e. to act as lock of piston in cylinder. Each slot is characterized by displacement \(z\) of sliding spool, volumetric flow \(Q\) and pressure drop \(\Delta p\). By known two of them the third can be computed.

Mathematical models of volumetric flow are different for overlapped and open slots. In case of open slots the volumetric flow depends on pressure drop square root. In case of overlapped slots the volumetric flow is proportional on pressure drop.

In particular case slots are considered in pairs (1-3 and 2-4). For example the model of slot pair 1-3 includes computational procedures for different states of slots (1 open and 3 overlapped, 1 overlapped and 3 open, both 1 and 3 overlapped).

Models of slot pairs are different for steady state conditions and dynamics. In the statics pressure \(pc_1\) is calculated from model RS13pS, displacement of sliding spool \(z\) is calculated from model RS24zS. In dynamics volumetric flows \(Q_{c1}\) and \(Q_{c2}\) are calculated from models RS13q and RS24q.
Simulation results of statics and steady-state conditions

Near to load force $\text{Fac2} = 0$ the graphs of the sliding spool displacement $z$ (graphs 1) have transitions, subject on overlapped throttling slots of sliding spool. Due to amplifying coefficient $ka$ of the torque motor feeding voltage the graphs of position of the actuator $\mathbf{x}_S$ (graphs 2) have approximately linear shapes in diapason of load force $\text{Fac2} = \pm 7.5E4$ N. In the diapason of force $\text{abs(Fac2)} > 7.5E4$ N the actuator position $\text{abs(xS)}$ increases rapidly.

Amplification coefficient $ka$ for torque motor voltage as a broken-linear dependence from displacement of sliding spool $z$ is used for reducing the non-linearity of servo-system static characteristics in the area of overlapped slots.
5.2 Electro-hydraulic servo-system

5.2.1 Simulation task of servo system dynamics

Feeding system

Safety valve

Servo-drive

Servo-valve
Conditions: step disturbance 3 V of input voltage U during 0.01 s and step disturbance 8E4 N of load force F during 0.01 s

1. Actuator moves almost linearly from 0 to 0.03 m in time interval 0...0.27 s. Precise analysis shows that lack 155 μm of desired actuator position 0.03 m takes place at time moment 0.30 s.

2. Actuator velocity oscillates with high frequency 900 Hz until the time moment 0.03 s, then increases to maximum 9E-2 m/s, thereafter decreases to zero at time moment 0.30 s.

3. Step disturbance of the input voltage is 3 V.

4. Step disturbance of the load force is 8E4 N.

1. Sliding spool moves rapidly from zero to 6.7E-5 m during 0.03 s. After 0.18 s the sliding spool moves to final position 3E-5 m.

2. Flapper jumps out to 7.5E-7 m and then moves to the position zero at time 0.3 s.
1 Feeding volumetric flow achieves the maximum value $6.2 \times 10^{-4}$ m$^3$/s. After 0.18 s the feeding volumetric flow decreases to $5 \times 10^{-6}$ m$^3$/s at time moment 0.3 s.

2 At the beginning the feeding pressure decreases to $15.6 \times 10^6$ Pa during the time 0.06 s. Then the feeding pressure increases to $21.7 \times 10^6$ Pa, after which damps at $21.5 \times 10^6$ Pa.

3 Safety valve reacts to changes of pressure.

1 Pressure in left chamber of the hydraulic cylinder rapidly increases from $10.5 \times 10^6$ Pa to $17.1 \times 10^6$ Pa. Pressure stabilizes at $17.6 \times 10^6$ Pa.

2 Pressure in right chamber of the hydraulic cylinder rapidly decreases from $10.5 \times 10^6$ Pa to $4.8 \times 10^6$ Pa. Thereafter the pressure stabilizes at $5.4 \times 10^6$ Pa.

Difference of pressures remains constant $12.2 \times 10^6$ Pa from the time moment the load force reaches the value of step disturbance $8 \times 10^4$ N.
5.4 Computing 3D graphs of a transient response of an electro-hydraulic servo-system
# Sizes and complexities

<table>
<thead>
<tr>
<th></th>
<th>Load-sensing system</th>
<th>Servo-system</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Model:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of components</td>
<td>26 (+4+11)</td>
<td>37 (+4+13)</td>
</tr>
<tr>
<td>Iterated variables</td>
<td>17</td>
<td>20</td>
</tr>
<tr>
<td><strong>Simulation task of dynamics:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Java code lines</td>
<td>5113</td>
<td>8920</td>
</tr>
<tr>
<td>Time step</td>
<td>1E-5 s</td>
<td>1E-6 s</td>
</tr>
<tr>
<td>Number of simulation steps</td>
<td>130 000</td>
<td>500 000</td>
</tr>
<tr>
<td>Simulation time (3.2 GHz processor )</td>
<td>7 min</td>
<td>21 min</td>
</tr>
</tbody>
</table>

**Difficulties:**

1. Short time step and large number of simulation steps are required
2. Parameters of working fluid must be recalculated at each step
3. ...
## 7 History

<table>
<thead>
<tr>
<th>Models</th>
<th>Computers</th>
<th>Software</th>
<th>Tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1960-s</td>
<td>four-pole models of elements, signal flow graphs</td>
<td>Minsk22</td>
<td>Fortran, Malgol</td>
</tr>
<tr>
<td>1970-s</td>
<td>four-pole models and transfer functions of valves</td>
<td>Minsk32</td>
<td>Fortran</td>
</tr>
<tr>
<td>1980-s</td>
<td>multi-pole models of components, models of systems</td>
<td>ES, minicomputers</td>
<td>PRIZ, C-PRIZ</td>
</tr>
<tr>
<td>1990-s</td>
<td>visual models</td>
<td>Sun workstations</td>
<td>NUT</td>
</tr>
<tr>
<td>2007-</td>
<td>visual models of large systems</td>
<td>personal computers</td>
<td>CoCoViLa</td>
</tr>
</tbody>
</table>
Thank you for attention