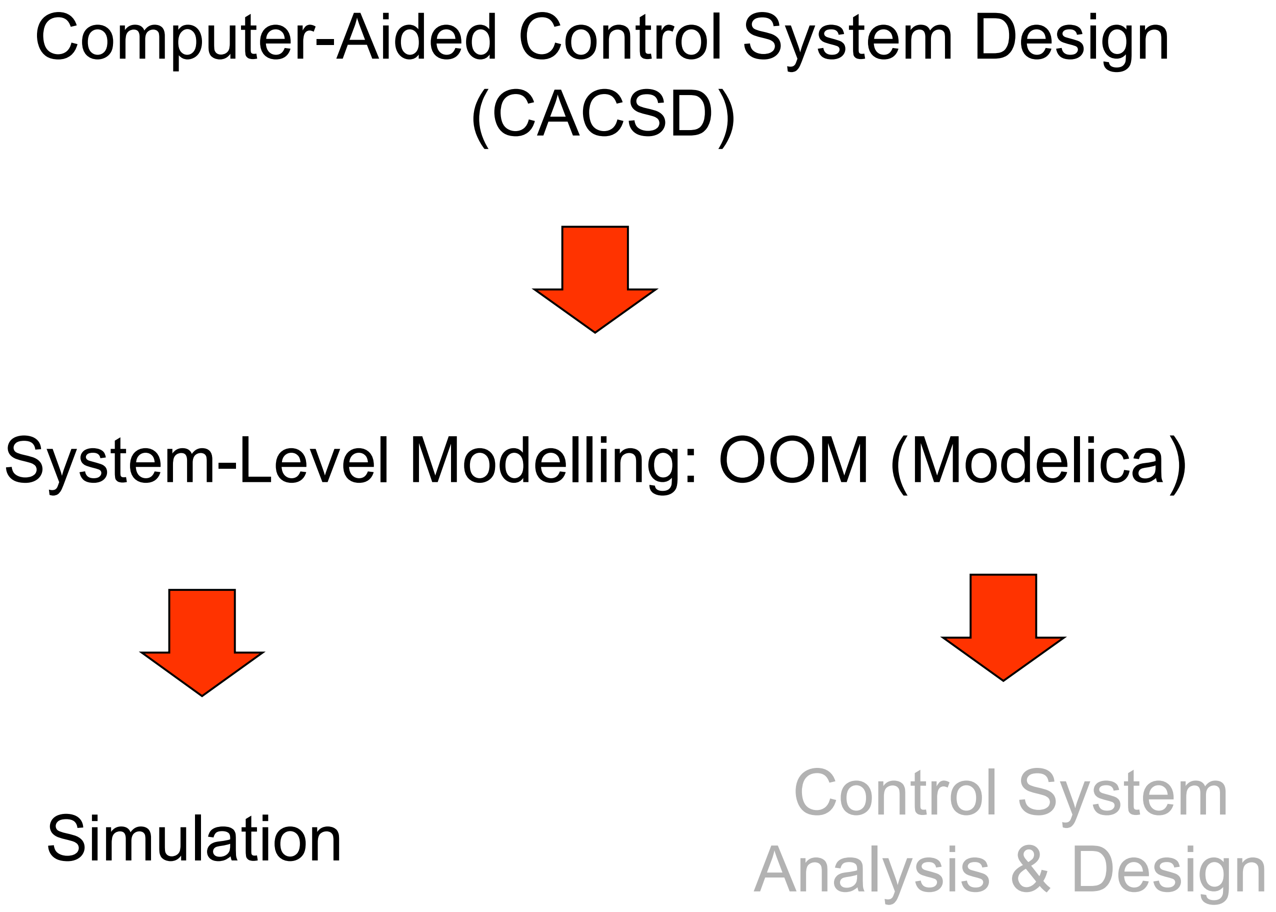
Beyond Simulation: Computer Aided Control System Design using Equation-Based Object Oriented Modelling for the Next Decade

Francesco Casella
Filippo Donida
Marco Lovera

Dipartimento di Elettronica e Informazione
Politecnico di Milano - Italy

Introduction



Computer-Aided Control System Design (CACSD)

System-Level Modelling: OOM (Modelica)

Simulation

Control System Analysis & Design

Modelling for Control System Design - I

- Critical control systems require dynamic modelling for their design
  - Knowledge about plant dynamics required for controller design (e.g. state-space equations or transfer functions)
  - Plant might not be available to gather experimental data
  - Experiments might be expensive/time-consuming/dangerous
  - Different plant design may be compared at early design stages
  - CS performance assessed and optimized before going on-line

Compact models for
control system design

Detailed models for
system simulation

Modelling for Control System Design - II

- Compact models for CS design
  - Low number of state variables (2-20)
  - Must capture the fundamental dynamics: many approximations
  - Must cover the whole operating range
  - Parameters should have a physical meaning
  - State-space form $$\begin{aligned}\dot{x}(t) &= f(x(t), u(t), p, t)\\ y(t) &= g(x(t), u(t), p, t)\end{aligned}$$
  - Linear(ized) models $$\begin{aligned}\dot{x}(t) &= Ax(t) + Bu(t)\\ y(t) &= Cx(t) + Du(t)\end{aligned} \qquad G(s) = C(sI - A)^{-1}B + D$$

- Detailed models for system simulation
  - Obtained from OOM tools and library
  - High number of state (10-500) and algebraic (100-10000) variables
  - Nonlinear DAEs $$F(x(t), \dot{x}(t), u(t), y(t), p, t) = 0$$

Current Support for CACSD in OOM tools

- Empirical identification of open-loop plant dynamics (simulation + system ID)
- Symbolic/numeric linearization
  - A, B, C, D matrices of high dimension
  - Can be reduced by standard linear MOR techniques
- Steady-state operating points (trimming) $F(\bar{x}, 0, \bar{u}, \bar{y}, p, 0) = 0$
  - Can be numerically problematic
- Closed-loop performance assessment by simulation
- Support to simplified model generation
  - by replaceable models with standard interfaces
  - usually not enough to get compact models for direct CS design
- Generation of real-time code for HIL simulation
  - Inline integration
  - Requires simplified models to begin with
- Limited optimization features

Future Perspectives

Future Perspectives

- Basic enabling technologies
  - Open standards for model and data exchange among tools
  - More open OOM tools
  - Automatic symbolic/numeric model order reduction
  - Improved initialization algorithms to solve steady-state problems

- New features for direct CS design support
  - Simplified symbolic transfer functions
  - Automatic derivation of LFT models
  - Inverse models for robotic systems
  - Fast and compact models for Model Predictive Control
  - …

Open Standards for Model/Data exchange

- Improved support for CS design requires the integration of different tools:
  - OOM compilers
  - Symbolic manipulation tools
  - CS design tools
- OOM tools should be more open
  - import/export model equations at various stages of compilation and manipulation
  - steer symbolic manipulation towards goals other than simulation
- Open standards for inter-tool data exchange should be available
- On-going work between Politecnico and Linkoping University for XML-based formats
  - easily represent complex data structures (e.g.: models)
  - easily translated to/from other representations
  - lots of available software for XML data handling
  - formally defined through DTD/XSD

Model Order Reduction

- Mixed numerical-symbolic MOR techniques have already been applied in the field of electronic circuits
- Basic steps:
  - specify relevant inputs and outputs
  - specify max error bounds
    - percentage error on steady-state values
    - max error during transients (time domain / frequency domain)
  - rank the terms in all DAEs, with respect to input/output accuracy
  - remove terms in ascending order, until error bound is exceeded
- Successful application in commercial tools (Analog Insydes by ITWM Fraunhofer Institut, Germany)
- Interfacing to OOM tools (OpenModelica) is currently being evaluated
- Same techniques could be embedded within the OOM compiler

Improved initialization

- Most analysis techniques require to solve the steady-state problem

$$F(\bar{x}, 0, \bar{u}, \bar{y}, p, 0) = 0$$

- If the problem is non-linear, the solver often fails because of convergence problems
- More robustness is required
- Strategy 1: homotopy methods

$$F_e(x, \dot{x}, y, u, p, t) = 0 \qquad F_t(x, \dot{x}, y, u, p, t) = 0$$

$$(1-\alpha)F_e(\bar{x}, 0, \bar{u}, \bar{y}, p, 0) + \alpha F_t(\bar{x}, 0, \bar{u}, \bar{y}, p, 0) = 0$$

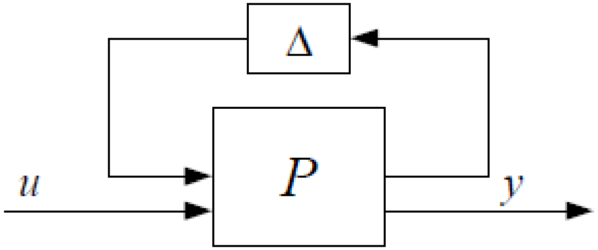
- Strategy 2: (easily!) re-use data from previous analysis to set up guess values
  - Initialization of similar models
  - Initialization of sub-models with suitable boundary conditions

Simplified Symbolic Transfer Functions

- Sometimes the plant dynamics has some critical features for CS design
- These can be identified on linearized dynamic models (transfer functions)
  - poorly damped complex conjugate poles
  - unstable poles
  - right half-plane zeros
- A nice feature is to obtain approximated transfer functions where the main dependency of such parameters on physical parameters is made explicit
- E.g., the natural frequency of conjugate poles in a mechanical system might depend mainly on the stiffness of a particular element
- This can be obtained by clever combination of OOM compilers, MOR tools, and symbolic manipulation tools

Automatic Derivation of LFT Models

- Linear Fractional Transformations are widely used in modern control science
- The system dynamics is described by a feedback connection of a dynamic LTI system and a Δ-block
- The Δ-block might represent
  - uncertain parameters
  - time-varying parameters
  - nonlinearities



- Models in this form are the starting points for
  - robust controller analysis and design
  - gain-scheduling controller design
  - uncertain parameter estimation from plant data
- These models should be obtained from the simulation model automatically (possibly after a MOR stage), as inputs for the CS design tools
- The coupling between OpenModelica and the LFR toolbox of ONERA is currently under investigation

Inverse models for robotic systems - I

- Multibody systems can be modelled with OOM languages (e.g. Modelica and the MultiBody library)
- Standard procedure: brings the model in a form suitable for simulation, given the torque inputs

Modelica model

$$
\begin{aligned}
B(q)\ddot{q} + H(q,\dot{q})\dot{q} + g(q) &= \tau \\
y_p &= K(q) \\
y_v &= \frac{\partial K}{\partial q}\dot{q},
\end{aligned}
$$

$$F_1(x, \dot{x}, y, u) = 0$$

$$x = \begin{bmatrix} x_p \\ x_v \end{bmatrix} = \begin{bmatrix} q \\ \dot{q} \end{bmatrix}, \quad y = \begin{bmatrix} y_p \\ y_v \end{bmatrix}, \quad u = \tau$$

solve for $dx/dt, y$

$$
\begin{aligned}
\dot{x} &= f(x, u) \\
y &= g(x, u).
\end{aligned}
$$

Inverse models for robotic systems - II

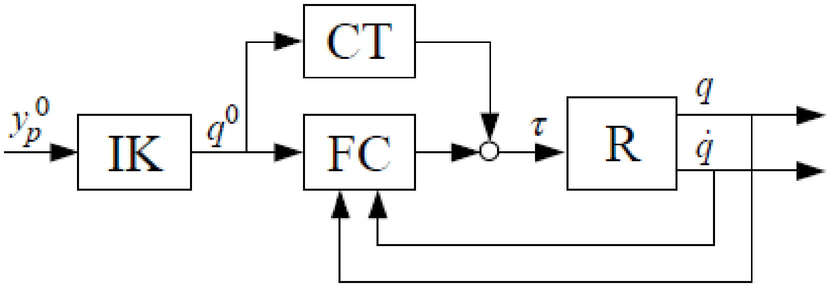
- There are other interesting problems for the control engineer:
- 1. Inverse Kinematics (IK)
  - solve for the joint angles, given the end effector positions
- 2. Computed Torque (CT)
  - solve for the torque, given the reference joint angle trajectories
- 3. Dynamic Inversion (DI)
  - solve for the torque, given a virtual joint acceleration input v

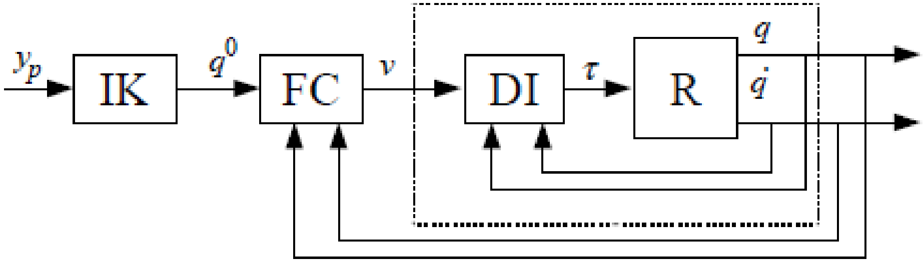
$$
\begin{aligned}
q^0 &= K^{-1}(y_p^0) \\
\dot{q}^0 &= \left(\frac{\partial K}{\partial q}\right)^{-1} y_v^0
\end{aligned}
$$

$$\tau = B(q^0)\ddot{q}^0 + H(q^0, \dot{q}^0)\dot{q}^0 + g(q^0)$$

$$
\begin{aligned}
&\ddot{q} = v \\
&\tau = B(q)v + H(q, \dot{q})\dot{q} + g(q)
\end{aligned}
$$

- The corresponding (Modelica or procedural) code can be obtained by the usual techniques (BLT, tearing, etc.)
- Then directly used for the control system implementation and validation
- Suitable tool interfaces must exist to specify this kind of problems





Fast & Compact Models for MPC

- Model Predictive Control turns a control problem into an optimization problem
  - Discrete-time control variable $u(t) = u(k), \quad kT_s \le t < (k+1)T_s$
  - Figure of merit
    - control effort
    - distance from set point
    - problem-specific performance index (e.g. energy consumption)
  - Constraints
    - min/max values for control inputs, outputs, states, and their rates
    - dynamic relationship between inputs and outputs (system model!)

$$x(k+1) = f(x(k), u(k), p, k)$$
$$y(k) = g(x(k), u(k), p, k)$$

- At each time step, a new optimization problem is solved, and the first control input is applied (*receding horizon approach*)
- Fast & compact models should be obtained from OO models
  - OOM language support: replaceable models
  - MOR techniques: can also span component boundaries!
  - Inline integration

Conclusions

- System-level modelling is essential for the control engineer
- OOM languages and tools currently provide:
  - very good support for simulation-based activities
  - limited direct support for CS design
- Future OOM tools should tackle the CS design problem more aggressively
  - (semi) automatic derivation of compact models
  - direct generation of models in the formalism required by the control technique
- This goal cannot be attained by monolithic tools, but rather by clever combinations of specialized tools
  - OOM compiler
  - MOR tools
  - LFT tools
  - CS design tools
  - …
- More open interfaces are thus required on OOM tools (both open-source and commercial!) that go beyond simulation problems