Modeling and Learning for Dynamical Systems

Lecture 3

Martin Enqvist

LINKÖPING UNIVERSITY

DAE Models

DAE Models

- First-principles modeling often results in **Differential Algebraic Equations (DAEs):** A set of equations describing the relation between some physical variables z, their derivatives $\dot{z}$ and input signals u:

$$F(\dot{z}, z, u) = 0$$

- In general, some components of F contain only undifferentiated variables (algebraic equations)
- The vector z is called **generalized state** or **internal variables**
- Higher order derivatives $\ddot{z}$ can be included in the equation above by introducing new variables $z_d = \dot{z}$ such that $\dot{z}_d = \ddot{z}$
- An output equation can also be included:

$$y = h(z, u)$$

Example: A Small Nonlinear DAE

A small nonlinear DAE model:

$$
\begin{aligned}
C\dot{z}_1 - z_2 &= 0 \\
z_1 + R_1 z_2 + R_2 z_2^5 - u &= 0 \\
y &= z_1
\end{aligned}
$$

Index

The **(differentiation) index** of a DAE model

$$F(\dot{z}, z, u) = 0 \qquad \text{(A)}$$

is the number of differentiations needed to be able to solve $\dot{z}$ from the larger set of equations obtained in this way, resulting in

$$\dot{z} = \phi(z, u, \dot{u}, \ldots, u^{(k)})$$

- If $\dot{z}$ can be solved directly from (A): state-space model, index = 0
- Otherwise: Differentiate equation (A)

$$\frac{\partial F}{\partial \dot{z}}\ddot{z} + \frac{\partial F}{\partial z}\dot{z} + \frac{\partial F}{\partial u}\dot{u} = 0 \qquad \text{(B)}$$

  If $\dot{z}$ can be solved from (A) and (B): index = 1
- Otherwise: Differentiate again, which gives another equation (C). If $\dot{z}$ can be solved from (A), (B) and (C): index = 2
- Otherwise: ...

The Implicit Function Theorem can be used to draw conclusions about the solvability of $\dot{z}$

Example: A Small Nonlinear DAE...

A small nonlinear DAE model:

$$\begin{aligned} C\dot{z}_1 - z_2 &= 0 \\ z_1 + R_1 z_2 + R_2 z_2^5 - u &= 0 \\ y &= z_1 \end{aligned}$$

Here, $\dot{z}_1$ can be obtained from the first equation:

$$\dot{z}_1 = \frac{z_2}{C}$$

Differentiating the second equation gives:

$$\dot{z}_1 + R_1\dot{z}_2 + 5R_2 z_2^4 \dot{z}_2 - \dot{u} = 0 \quad \Rightarrow$$

$$\dot{z}_2 = \frac{\dot{u} - \dot{z}_1}{R_1 + 5R_2 z_2^4} = \frac{\dot{u} - z_2/C}{R_1 + 5R_2 z_2^4}$$

Hence, this DAE model has index = 1 (since we can determine both derivatives after one differentiation)

Index...

In principle, a solution to a DAE model can be obtained by selecting initial conditions that satisfy $F(\dot{z}, z, u) = 0$ and then simulate

$$\dot{z} = \phi(z, u, \dot{u}, \ldots, u^{(k)})$$

using standard numerical methods for Ordinary Differential Equations (ODEs). Observation: DAEs are more complicated than ODEs:

- We might lack an explicit expression for ϕ
- The solution is less smooth than the input signal since it depends also on derivatives of the input signal (higher order derivatives in the general case)
- The initial conditions must satisfy all equations (including algebraic ones)
- The index can be viewed as a measure of how far a DAE is from an ODE
- In practice, DAEs with index $= 1$ are relatively easy to solve numerically whereas DAEs with higher index are more challenging

Linear DAE Models

Linear DAE models can be written

$$E\dot{z} + Fz = Gu$$

where E and F are square matrices

- If E is invertible:

$$\dot{z} = -E^{-1}Fz + E^{-1}Gu \quad \text{(a normal state-space model)}$$

- Genuine DAE models with nontrivial indices are obtained for singular E matrices
- Result: The DAE $E\dot{z} + Fz = Gu$ is uniquely solvable if $sE + F$ is invertible for some value of s

Index for Linear DAE Model

1. A linear DAE model where $\mathrm{rank}(E) = r$ can be written as
$$\begin{bmatrix} E_1 \\ E_2 \end{bmatrix} \dot{z} + \begin{bmatrix} F_1 \\ F_2 \end{bmatrix} z = \begin{bmatrix} G_1 \\ G_2 \end{bmatrix} u$$
where E_1 has full rank (r) and the rows of E_2 are linear combinations of the rows of E_1
2. Eliminate E_2 via row operations
$$\begin{bmatrix} E_1 \\ 0 \end{bmatrix} \dot{z} + \begin{bmatrix} F_1 \\ \tilde{F}_2 \end{bmatrix} z = \begin{bmatrix} G_1 \\ \tilde{G}_2 \end{bmatrix} u$$
3. Differentiate the lower part:
$$\begin{bmatrix} E_1 \\ \tilde{F}_2 \end{bmatrix} \dot{z} + \begin{bmatrix} F_1 \\ 0 \end{bmatrix} z = \begin{bmatrix} G_1 \\ 0 \end{bmatrix} u + \begin{bmatrix} 0 \\ \tilde{G}_2 \end{bmatrix} \dot{u}$$
4. If $\begin{bmatrix} E_1 \\ \tilde{F}_2 \end{bmatrix}$ is invertible: Solve for $\dot{z}$, index $= 1$
5. Otherwise: Repeat the procedure, index = number of differentiations

Model Transformations

A linear DAE model

$$E\dot{z} + Fz = Gu$$

can be transformed by a change of variables $z = Qw$ and multiplication with P from the left (P and Q non-singular matrices). This gives:

$$PEQ\dot{w} + PFQw = PGu$$

An output equation $y = Hz + Ju$ is transformed to $y = HQw + Ju$.

Standard Form I

Assume that there exists an s_0 that makes $s_0 E + F$ invertible. The matrices P and Q can then be chosen to yield the following form

$$\begin{bmatrix} I & 0 \\ 0 & N \end{bmatrix} \begin{bmatrix} \dot{w}_1 \\ \dot{w}_2 \end{bmatrix} + \begin{bmatrix} -A & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} B \\ \bar{B} \end{bmatrix} u$$

$$y = \begin{bmatrix} C & \bar{C} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} + Ju$$

where N is nilpotent, i.e., $N^k = 0$ for some integer k. The index is equal to the smallest k for which $N^k = 0$.

- w_1 obeys a normal state-space ODE:

$$\dot{w}_1 = Aw_1 + Bu$$

- Equation for w_2:

$$N\dot{w}_2 + w_2 = \bar{B}u$$

Standard Form I...

$$N\dot{w}_2 + w_2 = \bar{B}u$$

- If $N = 0$ (index = 1) then
$$w_2 = \bar{B}u$$
- If $N \neq 0$ but $N^2 = 0$ (index = 2) then
$$w_2 = \bar{B}u - N\bar{B}\dot{u}$$
- In general (index = k):
$$w_2 = \bar{B}u - N\bar{B}\dot{u} + \ldots + (-N)^{k-1}\bar{B}u^{(k-1)}$$

(we can solve for $\dot{w}_2$ by differentiating these equations once, which shows that the previous index concept is used also here)

Standard Form II

Hence, we have a second standard form:

$$\begin{aligned} \dot{x} &= Ax + Bu \quad (x = w_1) \\ y &= Cx + Du + D_1\dot{u} + \ldots + D_{k-1}u^{(k-1)} \end{aligned}$$

where $D = J + \bar{C}\bar{B}$, $D_1 = -\bar{C}N\bar{B}$, $\ldots$, $D_{k-1} = \bar{C}(-N)^{k-1}\bar{B}$

Object-Oriented Modeling

Modelica

Modelica is a standardized modeling language:

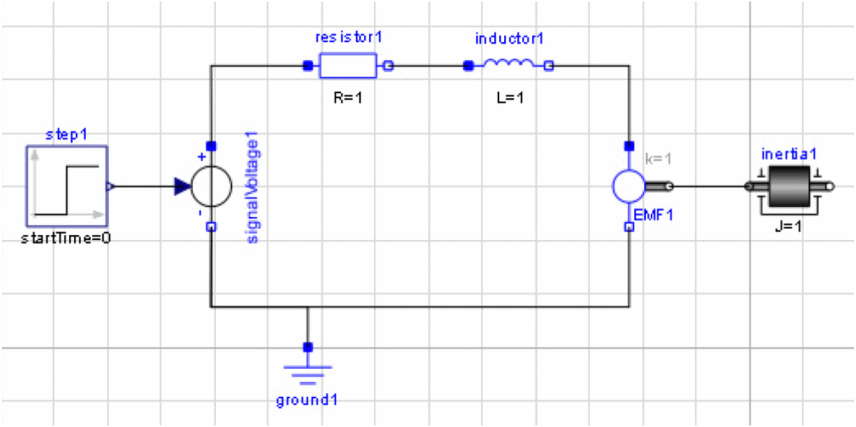
- based on equation descriptions (not necessarily of state-space type)
- facilitates hierarchical modeling using standard components
- object-oriented (inheritance, etc.)
- a standardized and wide-spread language for complex technical systems

A Small Modelica Model

$$
\begin{aligned}
C\dot{z}_1 - z_2 &= 0 \\
z_1 + R_1 z_2 + R_2 z_2^5 - u &= 0 \\
y &= z_1
\end{aligned}
$$

```
model Circuit
    Real z1, z2, u, y;
    parameter Real C=1, R1=2, R2=3;
equation
    u=10*sin(time);
    C*der(z1)-z2=0;
    z1+R1*z2+R2*z2^5-u=0;
    y=z1;
end Circuit;
```

DC Motor in Modelica



Connectors

Connectors in Modelica are used to connect submodels:

- A small submodel specifying the interface between component submodels
- Variables that correspond to each other in two submodels that should interact are set equal or to sum up to zero depending on type
- The use of connectors is one of the reasons why Modelica models often are DAE models

Simulation

Simulation of State-Space Models

A model in state-space form:

$$\dot{x}(t) = f(x(t), u(t))$$

Consider a particular input signal $u(t) = \bar{u}(t)$ and a particular initial state x_0. The influence of $\bar{u}$ can then be represented with a time-dependency of f (different f):

$$\begin{aligned} \dot{x}(t) &= f(t, x(t)) \\ x(0) &= x_0 \end{aligned}$$

Assume that we want a numerical approximation of x at the time instances

$$0 < t_1 < t_2 < \ldots < t_f$$

i.e. we want values x_1, x_2, …that approximate $x(t_1)$, $x(t_2)$, …

Euler's Methods

The **standard Euler method** is based on a simple derivative approximation:

$$\frac{x_{n+1} - x_n}{h} \approx \dot{x}(t_n) = f(t_n, x_n), \quad \text{where } h = t_{n+1} - t_n$$

This gives

$$x_{n+1} = x_n + hf(t_n, x_n) \quad \text{(explicit method)}$$

The **reversed Euler method** is on the other hand based on the approximation:

$$\frac{x_n - x_{n-1}}{h} \approx \dot{x}(t_n) = f(t_n, x_n), \quad \text{where } h = t_{n+1} - t_n$$

which gives

$$x_n = x_{n-1} + hf(t_n, x_n) \quad \text{(implicit method)}$$

General k-Step Methods

A general **k-step method** for numerical simulations of ODEs can be written

$$x_{n+1} = G(t, x_{n-k+1}, x_{n-k+2}, \ldots, x_n, x_{n+1})$$

- k previous values are used
- Explicit methods: G does not depend on x_{n+1}
- Implicit methods: G depends on $x_{n+1} \Rightarrow$ an equation has to be solved to get x_{n+1}

Interesting properties:

- Global error $E_n = x(t_n) - x_n$ (hard to compute in general, typically proportional to h^p for some p)
- Local error $e_n = x(t_n) - z_n$, where $z_n = G(t, x(t_{n-k}), \ldots, x(t_{n-1}), z_n)$ (can be computed from the Taylor series expansion, typically proportional to h^{p+1})
- Here, p is called the order of accuracy
- Stability: Is often investigated via the scalar test equation $\dot{x} = \lambda x$, $x(0) = 1$ (check stability of the resulting difference equation)

Methods and Solvers

- Families of numerical methods: Runge-Kutta methods, Adams's methods, . . . (these are better than Euler's methods for simulations)
- Useful idea: variable step length (idea: estimate the local error by comparing the result of taking two steps of length h and one of length $2h$, adjust h to maintain a chosen tolerance without using an unnecessarily small h)
- Solvers in Matlab (`ode45`, `ode23`, `ode113`, `ode78`, `ode89`, `ode15s`, `ode23s`, `ode23t`, `ode23tb`, `ode15i`)
- The different methods and solvers have their advantages and disadvantages, which are usually described in literature and help texts. The concepts discussed here should make the choice of solver easier.

Stiff Differential Equations

- For some models, the solutions contain both fast and slow components with large differences between their time constants. Such models are called **stiff**.
- The stability requirement can limit the step length in this case, making the simulations very slow
- Methods for stiff problems are often implicit since such methods often have larger stability regions than explicit ones

Simulation of DAE Models

Consider a DAE model $F(\dot{z}, z, u) = 0$. One numerical approach is to approximate the derivative $\dot{z}$ using z_n and k earlier z values (a backwards difference formula, BDF)

$$\dot{z} \approx \sum_{i=0}^{k} \alpha_i z_{n-i} =: \frac{1}{h}\rho^k z_n$$

and to solve the equation

$$F(\frac{1}{h}\rho^k z_n, z_n, u(t_n)) = 0$$

recursively. The accuracy depends on the index of the DAE model and models with higher index than one might require special methods.

Summary

DAE Models:

- General DAE models are frequent when working with first-principles modeling
- Key property: Index
- Standard forms for linear DAEs

Modelica:

- Standardized object-oriented modeling language
- Equation-based, both at component/object level and through connectors
- Results in DAE models

Simulation:

- Numerical methods for simulations
- Explicit and implicit methods
- Accuracy (local and global error), stability
- Variable step length
- Stiff differential equations
- Numerical methods for (low-index) DAEs

www.liu.se

LINKÖPING UNIVERSITY