CYBERNETICA

Model based control systems for the process industry

DAE MODEL REDUCTION FOR MPC

Stefan C. de Graaf, Pål Kittilsen, Heinz A. Preisig
Cybernetica AS, Leirfossveien 27, 7038 Trondheim, Norway
E-mail: stefan.degraaf@cybernetica.no

The integration process of DAE-models is often characterised by a high compuational load, which represents the main obstacle for the implementation of such models in real-time applications. The emphasis here is on the need to reduce both the number of differential equations and the number of algebraic equations in order to achieve a reduction of the compuational load. Different methods to do this are suggested . They are currently put to the test in a case study, provided by StatoilHydro, about reducing a distillation model for MPC.

Integrating DAE-models

The following DAE-systems typically cause high computational loads:

$$\dot{\underline{x}} = \underline{f}\left(\underline{x},\underline{z},\underline{u},\underline{d}\right) \qquad x\left(t_0\right) = x_0$$

$$0 = \underline{g}\left(\underline{x},\underline{z},\underline{u},\underline{d}\right)$$

Backward difference formulae (BDF) methods are commonly used for integration. The general form of this method is:

$$\begin{bmatrix}\underline{x}(t_n)\\ \underline{z}(t_n)\end{bmatrix} = \sum_{i=1}^{k}\alpha_i\begin{bmatrix}\underline{x}(t_{n-i})\\ \underline{z}(t_{n-i})\end{bmatrix} + h\beta_0\begin{bmatrix}\underline{f}(t_n)\\ \underline{g}(t_n)\end{bmatrix}$$

where α_i and β_0 are constants, h is the time step and n the mesh number. It shows that BDF methods use several past values of x and their rates of change f and g to move the integration forward from t_{n-1} to t_n. This is done iteratively until a convergence criterion is met.

This iteration uses the Jacobian that contains both partial derivatives of state equations and algebraic equations with respect to state and algebraic variables. Its complexity predominantly affects the computational load associated to the application of BDF methods. It can be shown that the following changes will reduce the complexity of the Jacobian:

- reduction of number and nonlinearity of state AND algebraic equations
- model adjustments that result in a smaller difference between the largest and the smallest eigenvalues of the Jacobian.

This can be done on reduction levels 1 and 2 that should be carried out complementary to each other.

Figure: One of StatoilHydro's gas processing plants with a C4-splitter for which a reduced model is implemented in an MPC (case study).

Reduction level 1: Modifying DAE-s

Transform and split state equations using e.g. balancing, POD or perturbation techniques to obtain three sub-models:

$$\dot{\tilde{\underline{x}}} = T\dot{\underline{x}} = T\underline{f}\left(T^{-1}\tilde{\underline{x}},\underline{u},\underline{z},\underline{d}\right) \quad \Rightarrow \quad \begin{bmatrix}0\\ \dot{\tilde{\underline{x}}}_2\\ \dot{\tilde{\underline{x}}}_3\end{bmatrix} = \begin{bmatrix}T_1 f\left(T^{-1}\tilde{x},u,z,d\right)\\ T_2 f\left(T^{-1}\tilde{x},u,z,d\right)\\ 0\end{bmatrix}$$

The new states may have the following properties:

Table: properties of the new states

reduction technique	$\tilde{x}_1$	$\tilde{x}_2$	$\tilde{x}_3$
perturbation	Fast changing states	Moderately changing states	Slowly changing states
POD	Fast changing states that are controllable	Moderately changing states that are controllable	States that are uncontrollable
balancing, balanced POD	Fast changing states that are controllable and observable	Moderately changing states that are controllable and observable	States that are uncontrollable or unobservable or both

Then reduce number and nonlinearity of all algebraic equations by fitting a simpler algebraic model to the original one. For example by

- (Multiple) linear regression in connection with discriminant analysis, principal component analysis and sub-space methods.
- Identification of an artificial neural network.

Reduction level 2: Adjusting the implementation of DAE-systems in integration routines

Three possibilities are:

1. Implement functions and solver algorithms in another, faster programming language. This is relevant when equation-based languages, e.g. gPROMS, are used.

2. Solve the algebraic equations off-line for different values of the dependent variables. Then store the solutions in tables and retrieve them during on-line applications.

3. Providing coded analytical Jacobians and storing Jacobians efficiently. Three options are:
   a. Use symbolic manipulation packages such as Mathematica and Maple.
   b. Generate Jacobians by using automatic differentiation.
   c. Store sparse and compressed structure of Jacobians.

Acknowledgement: This work is supported by StatoilHydro and the European Community's Marie-Curie Research and Training Network program under grant MRT-CT-2004-512441