NMPC for stiff, distributed parameter system: Semi-Automatic Code Generation and optimality condition evaluation

R. Noga[1] T. Ohtsuka[2]

[1]European Organization for Nuclear Research (CERN)

[2]Osaka University

18th International Conference on Process Control, 2011

Table of Contents

Layout

AutoGenU

AutoGenU is a Mathematica program to automatically generate simulation programs for Nonlinear Model Predictive Control (NMPC).

AutoGenU input

- System model
- Constraints
- Parameters
- Cost function
- C/GMRES parameters

AutoGenU output

C code of NMPC simulation using **analytically calculated Jacobians** and Continuation/GMRES algorithm

AutoGenU and the Superfluid Helium Cryogenic Circuit

AutoGenU has been applied to generate the NMPC for the Superfluid Helium Cryogenic Circuit (SHCC) at the Large Hadron Collider (LHC).

Simulation independent implementation of the NMPC based on the C/GMRES optimization

has been separated from the simulation program and

- has been used in Matlab® simulations, accessed via MEX functions,
- has been integrated into the PVSS II® SCADA of the LHC cryogenic system as a prototype implementation of NMPC for the SHCC.

The Superfluid Helium Cryogenic Circuit (SHCC)

A 213.8 m long Sub-Sector of the SHCC cools down and stabilizes at 1.9 K the temperature of 16 superconducting magnets of the LHC. Two over 100 m long Bayonet Heat Exchangers with two-phase flow of superfluid helium provide the cooling power.

The circuit has been modeled as a Distributed Parameter System

with

- magnet temperatures
- coolant mass flow rates in the Bayonet Heat Exchangers
- helium saturation temperature in the Bayonet Heat Exchangers

distributed over the Sub-Sector length.

Motivation for the Semi-Automatic Code Generation

After spatial discretization, the magnet temperature dynamics

$$\mathrm{d}x_i/\mathrm{d}t = f_i(x_{i-1}, x_i, x_{i+1}, a_i, b_i), \tag{1}$$

with the saturation temperature a and the coolant mass flow rate b

$$a_{i+1}(a_i, b_i), \quad b_{i+1}(a_i, b_i, x_i). \tag{2}$$

Automatic Code Generation for the SHCC

- the Jacobians expressed directly in terms of inputs
- cascading relations between a_is and b_is, enable propagation of complex expressions
- big output contains very complex expressions
- prohibitive operational memory requirement during generation

Layout

Semi-Automatic Code Generation: intermediate variables

Intermediate variables are introduced by hand and the generated output is expressed in terms of inputs and these variables.

Semi-Automatic Code Generation for the SHCC

- intermediate variables chosen to be a_is and b_is
- the subexpressions corresponding to intermediate variables are evaluated and do not further propagate
- the generated expressions are much less complex
- the generation procedure requires much less operational memory
- once the intermediate variables are available, other variables have smaller additional computing cost than if calculated directly

Semi-Automatic Code Generation: the chain rule

The chain rule is applied to find the Jacobians necessary to evaluate the optimality condition, using the intermediate variables, e.g.

$$\frac{\mathrm{d}f_i}{\mathrm{d}x_j} = \frac{\partial f_i}{\partial x_j} + \frac{\partial f_i}{\partial a_i}\frac{\mathrm{d}a_i}{\mathrm{d}x_j} + \frac{\partial f_i}{\partial b_i}\frac{\mathrm{d}b_i}{\mathrm{d}x_j} \tag{3}$$

The chain rule is also used to exploit the relations between the intermediate variables and evaluate their Jacobians with respect to the states and inputs, e.g.

$$\begin{bmatrix} \frac{\mathrm{d}a_{i+1}}{\mathrm{d}x_j} \\ \frac{\mathrm{d}b_{i+1}}{\mathrm{d}x_j} \end{bmatrix} = \begin{bmatrix} \frac{\partial a_{i+1}}{\partial a_i} & \frac{\partial a_{i+1}}{\partial b_i} \\ \frac{\partial b_{i+1}}{\partial a_i} & \frac{\partial b_{i+1}}{\partial b_i} \end{bmatrix} \begin{bmatrix} \frac{\mathrm{d}a_i}{\mathrm{d}x_j} \\ \frac{\mathrm{d}b_i}{\mathrm{d}x_j} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{\partial b_{i+1}}{\partial x_i}\delta_{i,j} \end{bmatrix} \tag{4}$$

Semi-Automatic Code Generation

- intermediate variables introduced and the chain rule realized **by hand**
- the relations between the intermediate variables, inputs and outputs generated **automatically**
- much faster generation that requires much less operational memory than the automatic generation used in AutoGenU. The observed generation time was seconds vs. 30 min. and the maximum memory used to store all the data for the Mathematica session was 7.9 MB vs. 1034 MB
- possibility to apply this type of the semi-automatic code generation process to more complicated systems.

Layout

C/GMRES OPTIMALITY CONDITION

In case of stiff dynamics, the system dynamics integration step must be very short and in order to separate its length from the control horizon discretization grid, the optimality condition is evaluated using the integrals

$$H_{u,i} := \int_{t'_i}^{t'_{i+1}} H_u \, \mathrm{d}t', \;\; C_i := \int_{t'_i}^{t'_{i+1}} C \, \mathrm{d}t', \tag{5}$$

with the vectors of system inputs $u(t)$ and states $x(t)$, the Hamiltonian $H = L(x,u) + \lambda^T \; f(x,u,t) + \mu^T \; C(x,u,t)$, time t, a performance index L,

$$\dot{x} = f(x,u), \;\; C(x,u) = 0 \tag{6}$$

that represent the system state dynamics and constraints respectively and the Lagrange multipliers $\lambda(t)$ and $\mu(t)$.

C/GMRES OPTIMALITY CONDITION

Automatic Code Generation for a stiff system

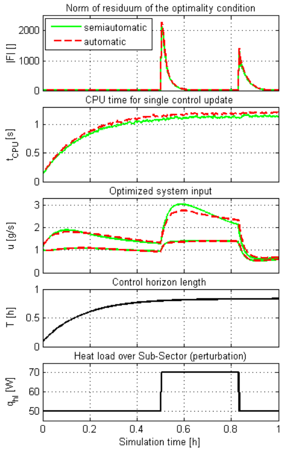
- H_u expressed directly in terms of system inputs and states has high computational cost
- H_u is evaluated exclusively at the horizon grid
- the integrals $H_{u,i}$ are calculated using simple quadratures

Semi-Automatic Code Generation for a stiff system

- intermediate variables available at each state dynamics integration step
- H_u using these variables has low additional computational cost
- H_u evaluated at each integration step
- integrals $H_{u,i}$ calculated more precisely

Simulation results

The time step for system dynamics integration is 62 times shorter than the horizon grid interval, thus the integrals $H_{u,i}$ and C_i are evaluated using one point and 62 points in automatic and semi-automatic code, respectively. The semi-automatically generated code is slightly faster than that automatically generated and the controller performance is similar in both cases.



Layout

Conclusions

In case of the Distributed Parameter System with cascading relations between variables and stiff dynamics:

- analytical Jacobians calculated automatically in terms of inputs and states are prohibitively complex
- semiautomatic generation using intermediate variables and the chain rule reduces the complexity
- optimality condition is evaluated more precisely at low additional computing cost using these variables available at each system state dynamics integration step,
- the semi-automatic code has slightly less computational cost, the controller performance is similar in both cases,
- **this semi-automatic code generation process requires significantly less memory**.

Layout

ACKNOWLEDGMENTS

Part of this work was done during the stay of the first author at Osaka University for the "FrontierLab@OsakaU" programme.
This work was supported by:

- Spanish Government (project DPI2009-12805),
- University of Valladolid,
- CERN,
- Osaka University,
- Japan Student Services Organization and
- Grant-in-Aid for Scientific Research (C) No. 21560465.

This support is very gratefully acknowledged.