The acados software package

Fast, embeddable solvers for nonlinear optimal control

Jonathan Frey, Katrin Baumgärtner and Moritz Diehl

Systems Control and Optimization Laboratory (syscop)

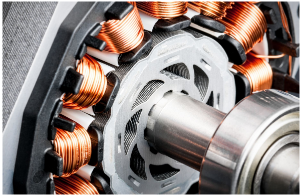
acados workshop @ ELO-X summer school

July 19, 2022

Workshop Outline

- Presentation
  - Introduction
  - acados optimal control problem formulation
  - Overview on acados
  - QP solvers
  - integration methods
  - Python interface
- Interactive Exercise / Demo Session

Intro



- Real world optimal control applications with
  - fast dynamics,
  - nonlinear optimal control problem formulations,
  - strict hardware limitations

  require tailored high-performance algorithms.
- acados implements such algorithms based on
  - Sequential Quadratic Programming (SQP)
  - Real-Time Iteration (RTI)
- Application projects include
  - Wind turbines
  - Electric drives (PMSM)
  - Race cars
  - Nano-drones
  - Microgrid
  - Thermal derating for electric machines

Intro – Model Predictive Control

Continuous-time optimal control problem (OCP):

$$
\begin{aligned}
\underset{x(\cdot),z(\cdot),u(\cdot)}{\text{minimize}} \quad & \int_{t=0}^{T} \ell(x(t), z(t), u(t))\,\mathrm{d}t + M(x(T)) \\
\text{subject to} \quad & x(0) = \bar{x}_0, \\
& 0 = f(\dot{x}(t), x(t), z(t), u(t)), \quad t \in [0, T], \\
& 0 \geq g(x(t), z(t), u(t)), \quad t \in [0, T].
\end{aligned}
\tag{1}
$$

In MPC, instances of these problems are solved repeatedly, with current state $\bar{x}_0$.

OCP structured NLP handled in acados

$$\underset{\substack{x_0,\ldots,x_N,\\ u_0,\ldots,u_{N-1},\\ z_0,\ldots,z_{N-1},\\ s_0,\ldots,s_N}}{\text{minimize}} \quad \sum_{k=0}^{N-1} l_k(x_k,u_k,z_k) + M(x_N) + \sum_{k=0}^{N} \rho_k(s_k) \tag{2a}$$

$$\text{subject to} \quad \begin{bmatrix} x_{k+1} \\ z_k \end{bmatrix} = \phi_k(x_k,u_k), \qquad k=0,\ldots,N-1, \tag{2b}$$

$$0 \geq g_k(x_k,z_k,u_k) - J_{\mathrm{s},k} s_k \qquad k=0,\ldots,N-1, \tag{2c}$$

$$0 \geq g_N(x_N) - J_{\mathrm{s},N} s_N, \tag{2d}$$

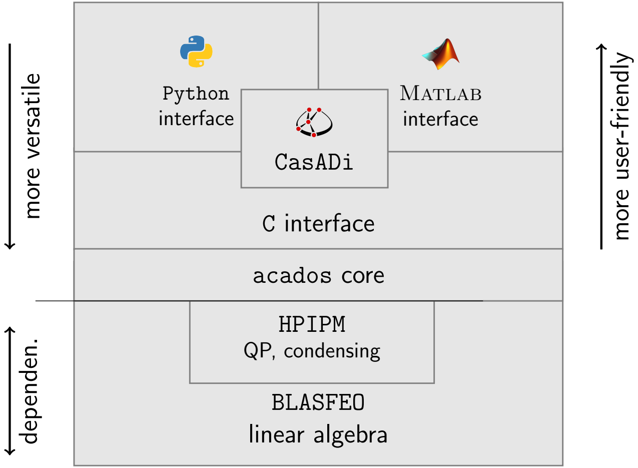
$$0 \leq s_k \qquad k=0,\ldots,N. \tag{2e}$$

- ϕ_k – discrete time dynamics on $[t_k, t_{k+1}]$ – typically acados integrator from ODE or DAE
- l_k discrete version of Lagrange cost term ℓ on $[t_k, t_{k+1}]$
- slack variables s_k separate from controls – handled efficiently
- inequality constraints g_k
- problem functions can vary stage wise in `C`
- from high-level interfaces
  - initial and terminal shooting node handled separately – MHE support
  - parameters can be varied conveniently
- more detailed problem formulation can be found *here*.

Philosophy & History

- Solvers and interfaces for
  - OCP structured NLP (2)
  - Initial value problems for ODEs and DAEs – integrators
- Exploit block-sparse structure inherent in OCP / MHE formulation → specialized solver
- Successor of the `ACADO Toolkit`
  - Code generation for all parts of the SQP method
- Principles of `acados`
  - efficiency – `BLASFEO`, `HPIPM`, `C`
  - flexibility – general formulation
  - modularity – encapsulation
  - portability – self-contained `C` library with little dependencies
- Model functions code generation using `CasADi`
- Problem formulation in high-level interface (`Python`, MATLAB, `Octave`)
- Generate corresponding `C` code for problem specific solver
  - uses only `acados` `C` interface
  - first developed in `Python` interface
  - used for `S-function` generation – `Simulink` interface

Structure of the acados software



The interplay between the acados dependencies, the 'core' C library and its interfaces.

- BLASFEO: Basic Linear Algebra for Embedded Optimization
- HPIPM: High-Performance Interior Point Method

QP solver types and sparsity – an overview

	Active-Set	Interior-Point	First-Order
dense	qpOASES, DAQP	HPIPM	
sparse	[PRESAS]	CVXGEN, OOQP	FiOrdOs, OSQP
OCP structure	qpDUNES, [ASIPM]	HPMPC, HPIPM, [ASIPM], [FORCES]	

Table: Overview: QP solver types and their way to handle sparsity.
underline: available in acados + support in Simulink
gray: not interfaced in acados, [proprietary]

efficient condensing from HPIPM:

- condensing: OCP structured → dense, expand solution
- partial condensing: OCP structured with horizon N → OCP structured with horizon $N_2 < N$, expand solution, $N_2 \hat{=}$ qp_solver_cond_N

Integration methods in acados

- solve Initial Value Problems (IVP) for
  - Ordinary Differential Equations (ODE)
  - Differential-Algebraic Equations (DAE)
  - $+$ sensitivity propagation (derivative of result with respect to initial state, control input)
- integrators in `Python`, are `'ERK'`, `'IRK'`, `'IRK_GNSF'`
- size of Butcher table: `sim_method_num_stages`
- time step is divided into `sim_method_num_steps` intervals, use the integration method on each interval
- ERK: explicit Runge-Kutta
  - integration order `sim_method_num_stages` $= 1, 2, 4$
- IRK: implicit Runge-Kutta
  - Gauss-Legendre Butcher tableaus
    - integration order $2 \cdot$ `sim_method_num_stages`, A-stable, but not L-stable
  - Gauss-Radau IIA
    - integration order is $2 \cdot$ `sim_method_num_stages` $- 1$, L-stable
- IRK_GNSF: implicit structure-exploiting Runge-Kutta method
  - Butcher tableaus as IRK
  - Detecting and Exploiting Generalized Nonlinear Static Feedback Structures in DAE Systems for MPC, J. Frey, R. Quirynen, D. Kouzoupis, G. Frison, J. Geisler, A. Schild, M. Diehl, ECC 2019

Python interface

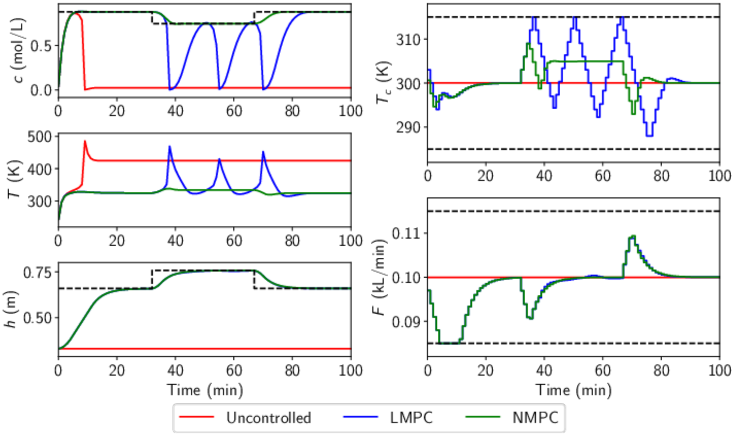
- Continuous time formulation
  - Discretization flexible, cost multiplied with time step, nonuniform grid possible
- Model functions provided as `CasADi` expressions
- Template workflow
  - Model function and derivatives generated using `CasADi`
  - `C` code to set up the OCP solver using the `C` interface
  - `Makefile` to compile everything into a shared library
  - Shared library is loaded in `Python` and used via a wrapper
  - Generated solver can be used in alternative wrapper and embedded framework, `C++`, `ROS`

Important Ressources

- https://docs.acados.org/
  - Python API - documents all options in template interface: https://docs.acados.org/python_api
  - Installation instructions https://docs.acados.org/installation
- acados MATLAB problem formulation PDF: https://github.com/acados/acados/blob/master/docs/problem_formulation/problem_formulation_ocp_mex.pdf
- latest acados publication:

  acados – a modular open-source framework for fast embedded optimal control R. Verschueren, G. Frison, D. Kouzoupis, J. Frey, N. van Duijkeren, A. Zanelli, B. Novoselnik, T. Albin, R. Quirynen, M. Diehl Mathematical Programming Computation 2021
- acados forum https://discourse.acados.org
- Github examples https://github.com/acados/acados/tree/master/examples

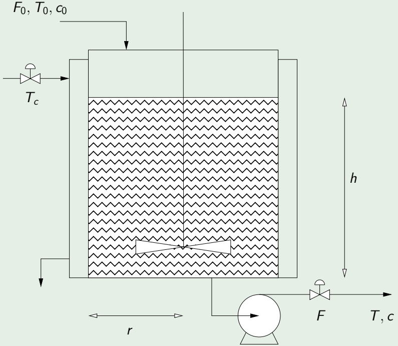
- Replicate plot from slides of James Rawlings



Simulation exercise: LQG control of an exothermic CSTR

Example 5

We consider a well-stirred chemical reactor as in Pannocchia and Rawlings (2003). An irreversible, first-order reaction A$\longrightarrow$ B occurs in the liquid phase and the reactor temperature is regulated with external cooling.



Mass and energy balances

Mass and energy balances lead to the following nonlinear state space model:

$$
\begin{aligned}
\frac{dc}{dt} &= \frac{F_0(c_0 - c)}{\pi r^2 h} - k_0 \exp\left(-\frac{E}{RT}\right) c \\
\frac{dT}{dt} &= \frac{F_0(T_0 - T)}{\pi r^2 h} + \frac{-\Delta H}{\rho C_p} k_0 \exp\left(-\frac{E}{RT}\right) c + \frac{2U}{r\rho C_p}(T_c - T) \\
\frac{dh}{dt} &= \frac{F_0 - F}{\pi r^2}
\end{aligned}
$$

Steady-state operating point

- The controlled variables are h, the level of the tank, and c, the molar concentration of species A. The additional state variable is T, the reactor temperature
- The manipulated variables are T_c, the coolant liquid temperature, and F, the outlet flowrate.
- Moreover, it is assumed that the inlet flowrate acts as an unmeasured disturbance.
- The open-loop stable steady-state operating conditions are the following:

$$c^s = 0.878\,\mathrm{kmol/m^3} \qquad T^s = 324.5\,\mathrm{K} \qquad h^s = 0.659\,\mathrm{m}$$
$$T_c^s = 300\,\mathrm{K} \qquad F^s = 0.1\,\mathrm{m^3/min}$$

- The model parameters in nominal conditions are reported in the following table.

Problem Shooting – OCP

- check OCP solver status, 0 – Success, other values defined in `types.h`
- `ocp_solver.print_statistics()`
  - KKT residuals: `stat` stationarity – Lagrange gradient, `eq`: equality constraints, `ineq`: inequality constraints, `comp`: complementarity
  - `qp_stat`: status of the QP solver, should be 0
  - `qp_iter`: number of iterations in QP solver
- initialization: `set()` – x, u, multipliers
- infeasibility: introduce slacks (soft constraints)
- try different Hessian approximations: Definiteness & Exactness
  - `ocp.solver_options.hessian_approx = 'GAUSS_NEWTON'` or `'EXACT'`
  - Option to turn off exact hessian contributions from cost, constraints or dynamics
  - Set numerical approximation for cost hessian
  - add a Levenberg Marquardt term, `ocp.solver_options.levenberg_marquardt`
- iterates don't converge:
  - reduce the step size, `ocp_opts.set('step_size', alpha)` with `alpha` < 1.
  - globalization (preliminary implementation) of a merit function based backtracking:
    - `ocp.solver_options.globalization = 'FIXED_STEP'` or `'MERIT_BACKTRACKING'`
    - Second order correction: set `globalization_use_SOC` to 1
    - Armijo condition: set `line_search_use_sufficient_descent` to 1.