The acados software package

Fast, embeddable solvers for nonlinear optimal control

Jonathan Frey and Katrin Baumgärtner

Systems Control and Optimization Laboratory (syscop)

acados workshop for FOR2041 & friends

April 29, 2021

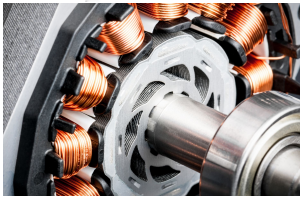
Workshop Outline

- Presentation
  - Introduction
  - acados optimal control problem formulation
  - Overview on acados
  - Focus on the MATLAB, Simulink interfaces, code generation
- Interactivity – Please, ask questions!
- Exercise session and problem shooting

Introduction

- Jonathan Frey
  - Master thesis: "Structure Exploiting Integrators for Model Predictive Control"
  - GNSF-IRK implementation in acados
  - 2019: research internship at MERL in Boston - ASIPM solver & paper
  - PhD student at syscop – acados maintainer
  - on efficient optimization algorithms and problem formulations for MPC
- Katrin Baumgärtner
  - Master thesis: "Asymptotic Comparison of Bayesian State Estimates and Numerical Methods for Moving Horizon Estimation"
  - PhD student at syscop with focus on state and parameter estimation methods
- latest publication on acados:
  - acados: a modular open-source framework for fast embedded optimal control, Robin Verschueren, Gianluca Frison, Dimitris Kouzoupis, Jonathan Frey, Niels van Duijkeren, Andrea Zanelli, Branimir Novoselnik, Thivaharan Albin, Rien Quirynen, Moritz Diehl, https://arxiv.org/abs/1910.13753

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
  - **Low-temperature combustion?**

Continuous-time optimal control problem (OCP):

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

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

OCP structured NLP handled in acados

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

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

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

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

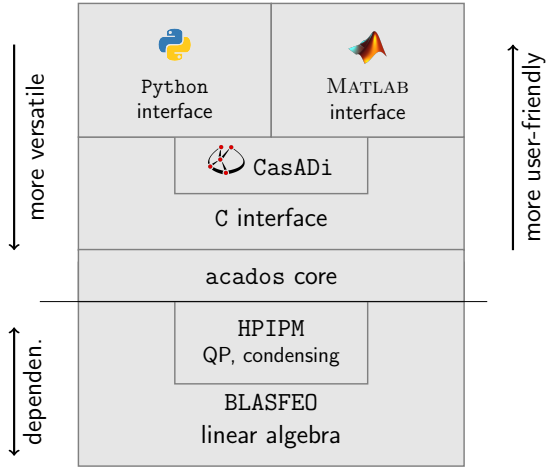
$$0 \leq s_k \qquad k = 0, \dots, N. \tag{2e}$$

- 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

- 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
- solver interfaces for
  - OCP structured NLP (2)
  - Initial value problems for ODEs and DAEs – integrators

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	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 – partly proprietary

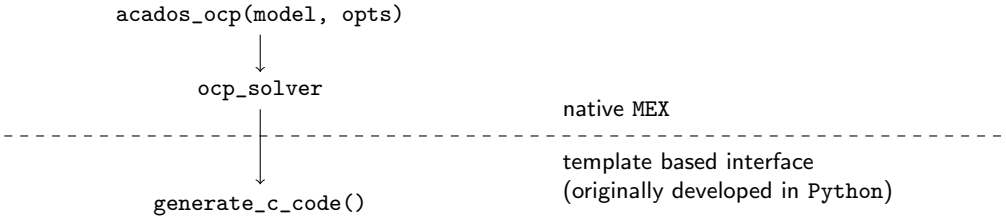
efficient condensing from HPIPM:

- condensing: OCP structured $\rightarrow$ dense, expand solution
- partial condensing: OCP structured with horizon $N \rightarrow$ 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)
- `sim_method` in MATLAB, supports `'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`
- GNSF-IRK: implicit structure-exploiting Runge-Kutta method
  - Gauss-Legendre Butcher tableaus
  - integration order $2 \cdot$ `sim_method_num_stages`
  - 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

MATLAB template based interface



- builds ocp_json object, mimicing acados OCP formulation in Python
- ocp_json → .json file

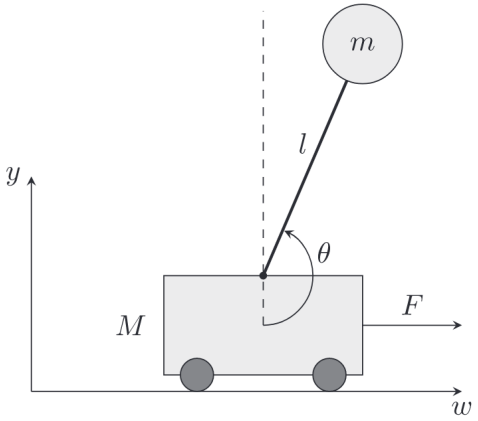
problem specific OCP C solver and integrator

- → Simulink S functions
- → OCP MEX solver (Linux only)
- → Makefile
- → main.c

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
  - dSPACE deployment – currently only MicroLabBox; soon MAB II, MAB III.
- acados MATLAB problem formulation PDF: https://github.com/acados/acados/blob/master/docs/problem_formulation/problem_formulation_ocp_mex.pdf
- latest acados publication https://arxiv.org/abs/1910.13753
- acados forum https://discourse.acados.org
- Github examples https://github.com/acados/acados/tree/master/examples

Questions & Exercise Session



https://www.syscop.de/event/acados-workshop-for2041

Problem Shooting – OCP

- check OCP solver status
  - 0 – Success, other values defined here `https://github.com/acados/acados/blob/master/acados/utils/types.h`
- `ocp.print`: print information on SQP iterations
  - 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, `ocp_opts.set('nlp_solver_exact_hessian', val)`
  - `val = 'true'`: exact Hessian
  - `val = 'false'`: for Gauss-Newton Hessian; always positive definite, good for convergence (only applicable for least-squares problems)
- iterates don't converge:
  - add a Levenberg Marquardt term, `ocp_opts.set('levenberg_marquardt', 1e-3)`
  - reduce the step size, `ocp_opts.set('step_size', alpha)` with `alpha` < 1.
  - globalization (preliminary implementation), `ocp_opts.set('globalization', 'merit_backtracking')` and `ocp_opts.set('alpha_min', amin)`, `ocp_opts.set('alpha_reduction', areduce)` with $0 <$ `amin`, `areduce` < 1.