

acados

A rapid prototyping tool for NMPC

Robin V, Dimitris K, Andrea Z, Rien Q, Niels v. D, Joris G, and
Moritz Diehl

Systems Control and Optimization laboratory

October 6, 2016





`www.acadotoolkit.org`

Key Properties of ACADO Toolkit [Houska et al 2009]

- ▶ Open Source (LGPL)
- ▶ **A**utomatic **C**ontrol **A**nd **D**ynamic **O**ptimization
- ▶ User friendly interface close to mathematical syntax



`www.acadotoolkit.org`

Key Properties of ACADO Toolkit [Houska et al 2009]

- ▶ Open Source (LGPL)
- ▶ **A**utomatic **C**ontrol **A**nd **D**ynamic **O**ptimization
- ▶ User friendly interface close to mathematical syntax

Multiplatform support

- ▶ C++: Linux, OS X, Windows
- ▶ MATLAB

ACADO developers (past and current)



Moritz Diehl
Scientific
advisor



Hans Joachim Ferreau
Main developer



Boris Houska
Main developer



Filip Logist
Multi-objective optimization



Rien Quirynen
Code generation



Dries Telen
Optimal Experimental
Design



Mattia Valerio
Multi-objective optimal
control



Milan Vukov
Code generation for MPC &
MHE



Pro

- ▶ Easy to use,

Con

- ▶ ... but codebase difficult to maintain



Pro

- ▶ Easy to use,
- ▶ Fast NMPC/MHE solvers,

Con

- ▶ ... but codebase difficult to maintain
- ▶ ... but with a lot of global data



Pro

- ▶ Easy to use,
- ▶ Fast NMPC/MHE solvers,
- ▶ Interfaced to external solvers,

Con

- ▶ ... but codebase difficult to maintain
- ▶ ... but with a lot of global data
- ▶ ... but hard coupling with new ones



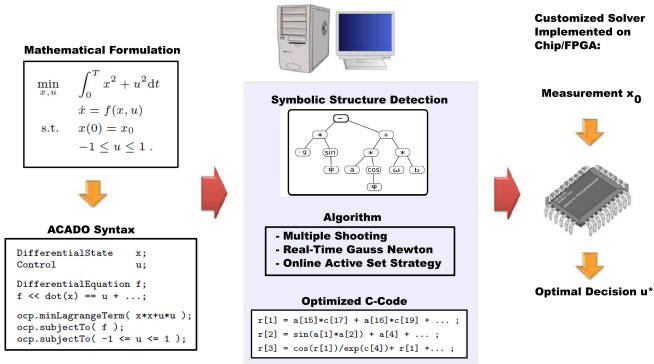
Pro

- ▶ Easy to use,
- ▶ Fast NMPC/MHE solvers,
- ▶ Interfaced to external solvers,
- ▶ Comes with own AD,

Con

- ▶ ... but codebase difficult to maintain
- ▶ ... but with a lot of global data
- ▶ ... but hard coupling with new ones
- ▶ ... but not compatible with CasADi





But is codegen really necessary?



Why code generation?

- ▶ eliminate computations
- ▶ known dimensions and sparsity patterns
- ▶ no dynamic memory
- ▶ code reorganization, ...



Why code generation?

- ▶ eliminate computations
- ▶ known dimensions and sparsity patterns
- ▶ no dynamic memory
- ▶ code reorganization, ...

But code generation

- ▶ increases code size
- ▶ creates unreadable code
- ▶ only really makes sense at the bottleneck \rightarrow linear algebra

A different idea



Instead of code-generated code, we want

- ▶ maintainable,
- ▶ extensible,
- ▶ fast code,
- ▶ with efficient linear algebra kernels,
- ▶ available from MATLAB and Python.

→ **acados**



The structure of acados is based on internal interfaces:

- ▶ OCP QP interface,
- ▶ condensing interface,
- ▶ integrator interface,
- ▶ AD interface,
- ▶ Hessian regularization interface,
- ▶ ...

If a solver sticks to a certain interface, it can be swapped quickly for another one

→ Rapid Prototyping



As of now: one NMPC example in C99 with

- ▶ Model: [Chen&Allgoewer 1998]
- ▶ Integrator: Runge-Kutta 4
- ▶ Condensing: N^2
- ▶ QP solver: qpOASES3.0



As of now: one NMPC example in C99 with

- ▶ Model: [Chen&Allgoewer 1998]
- ▶ Integrator: Runge-Kutta 4
- ▶ Condensing: N^2
- ▶ QP solver: qpOASES3.0

0.63 ms per iteration,
only 2x slower than ACADO code generated code,
not bad for first 'plain vanilla' attempt 😊



Located at `https://github.com/acados/acados`

- ▶ Completely written in C99
- ▶ Each interface has its own header file
- ▶ Data passed around by pointers to structs



Located at <https://github.com/acados/acados>

- ▶ Completely written in C99
- ▶ Each interface has its own header file
- ▶ Data passed around by pointers to structs

TODO:

- ▶ Interface with CasADi (and thus MATLAB/Python)
- ▶ Use BlasFEO's (former HPMPC) highly optimized linear algebra
- ▶ Implement many more new solvers
- ▶ Real-life examples
- ▶ ...