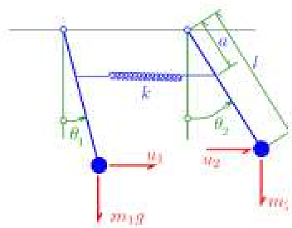
ACADO CODE GENERATION

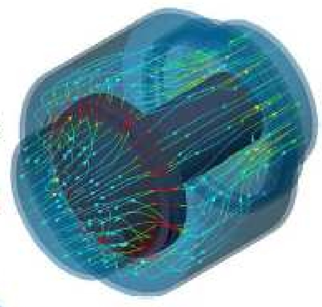
Boris Houska, Rien Quirynen, Hans Joachim Ferreau, Milan Vukov,
Moritz Diehl

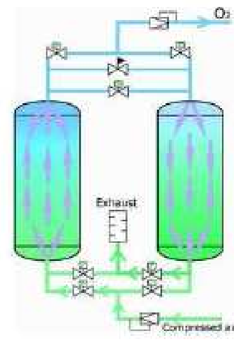
Overview

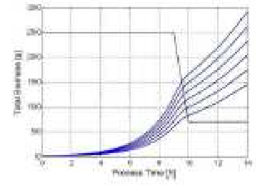
- ACADO Toolkit
- Automatic Code Generation
- Examples
- Conclusion
- Live Demo (by Rien Quirynen)

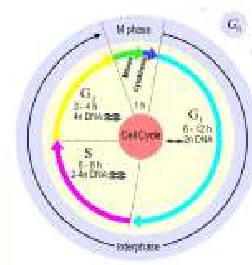
Nonlinear Dynamic Systems

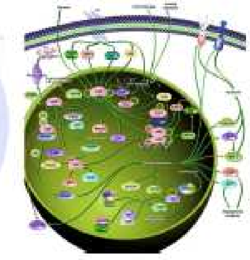


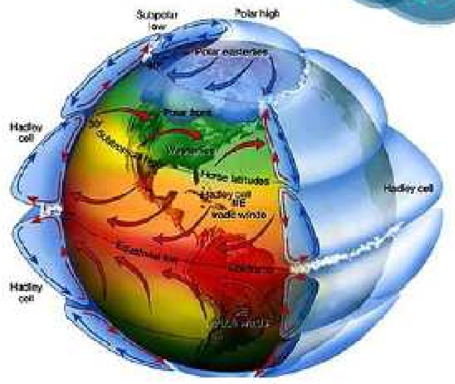




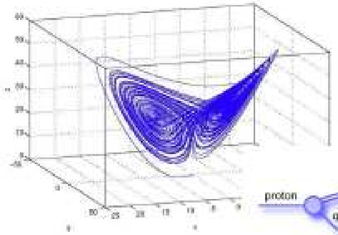


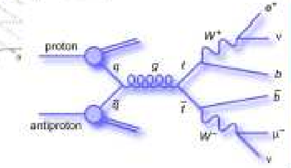


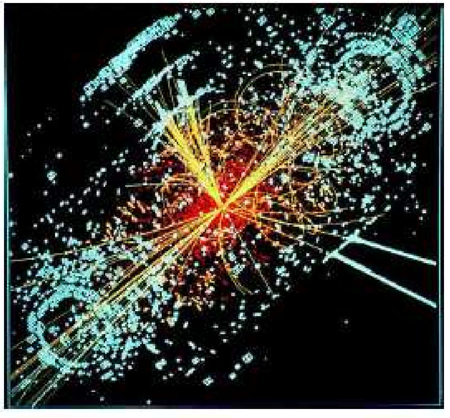


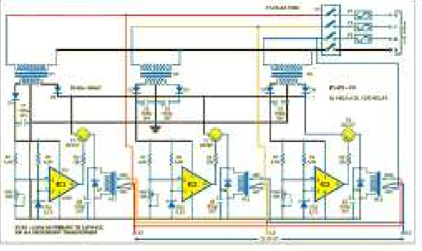


$$\dot{x}(t) = f(x(t), \ldots)$$









Optimal Control

Many Fields of Application:

- Optimal Motions in Robotics
- Operation of a Chemical Plant
- Seasonal Heat Storage
- Kite Power

Problems:

- Optimize Parameters/Controls
- Uncertainties/Disturbances

Optimal Control Software

ACADO Toolkit:

- **A**utomatic **C**ontrol **A**nd **D**ynamic **O**ptimization
- Open Source (LGPL)   `www.acadotoolkit.org`

Optimal Control Software

ACADO Toolkit:

- **A**utomatic **C**ontrol **A**nd **D**ynamic **O**ptimization
- Open Source (LGPL) www.acadotoolkit.org

Main problem class:

$$\underset{y(\cdot),u(\cdot),p,T}{\text{minimize}} \quad \int_0^T L(\tau, y(\tau), u(\tau), p)\,\mathrm{d}\tau \; + \; M(y(T), p)$$

subject to:

$$\begin{aligned} \forall t \in [0,T]: \quad 0 &= f(t, \dot{y}(t), y(t), u(t), p) \\ 0 &= r(y(0), y(T), p) \\ \forall t \in [0,T]: \quad 0 &\geq s(t, y(t), u(t), p) \end{aligned}$$

Optimal Control Software

ACADO Toolkit:

- **A**utomatic **C**ontrol **A**nd **D**ynamic **O**ptimization
- Open Source (LGPL)   `www.acadotoolkit.org`

Currently Active Developers:

Moritz Diehl
Scientific advisor

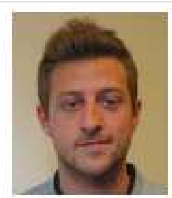
Hans Joachim Ferreau
Main developer

Boris Houska
Main developer

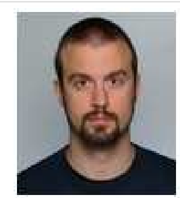
Filip Logist
Multi-objective optimization

Rien Quirynen
Code generation

Dries Telen
Optimal Experimental Design



Mattia Valerio
Multi-objective optimal control



Milan Vukov
Code generation for MPC & MHE

Tutorial Example: Time Optimal Control of a Rocket

Mathematical Formulation:

$$\underset{s(\cdot),v(\cdot),m(\cdot),u(\cdot),T}{\text{minimize}} \quad T$$

subject to

$$\begin{aligned}
\dot{s}(t) &= v(t) \\
\dot{v}(t) &= \frac{u(t)-0.2\,v(t)^2}{m(t)} \\
\dot{m}(t) &= -0.01\,u(t)^2
\end{aligned}$$

$$\begin{aligned}
s(0) &= 0 \quad s(T) = 10 \\
v(0) &= 0 \quad v(T) = 0 \\
m(0) &= 1
\end{aligned}$$

$$\begin{aligned}
-0.1 &\leq v(t) \leq 1.7 \\
-1.1 &\leq u(t) \leq 1.1 \\
5 &\leq T \leq 15
\end{aligned}$$

Tutorial Example: Time Optimal Control of a Rocket

Mathematical Formulation:

$$\underset{s(\cdot),v(\cdot),m(\cdot),u(\cdot),T}{\text{minimize}} \quad T$$

subject to

$$\begin{aligned}
\dot{s}(t) &= v(t) \\
\dot{v}(t) &= \frac{u(t) - 0.2\, v(t)^2}{m(t)} \\
\dot{m}(t) &= -0.01\, u(t)^2
\end{aligned}$$

$$\begin{aligned}
s(0) &= 0 \quad & s(T) &= 10 \\
v(0) &= 0 \quad & v(T) &= 0 \\
m(0) &= 1
\end{aligned}$$

$$\begin{aligned}
-0.1 &\leq v(t) \leq 1.7 \\
-1.1 &\leq u(t) \leq 1.1 \\
5 &\leq T \leq 15
\end{aligned}$$

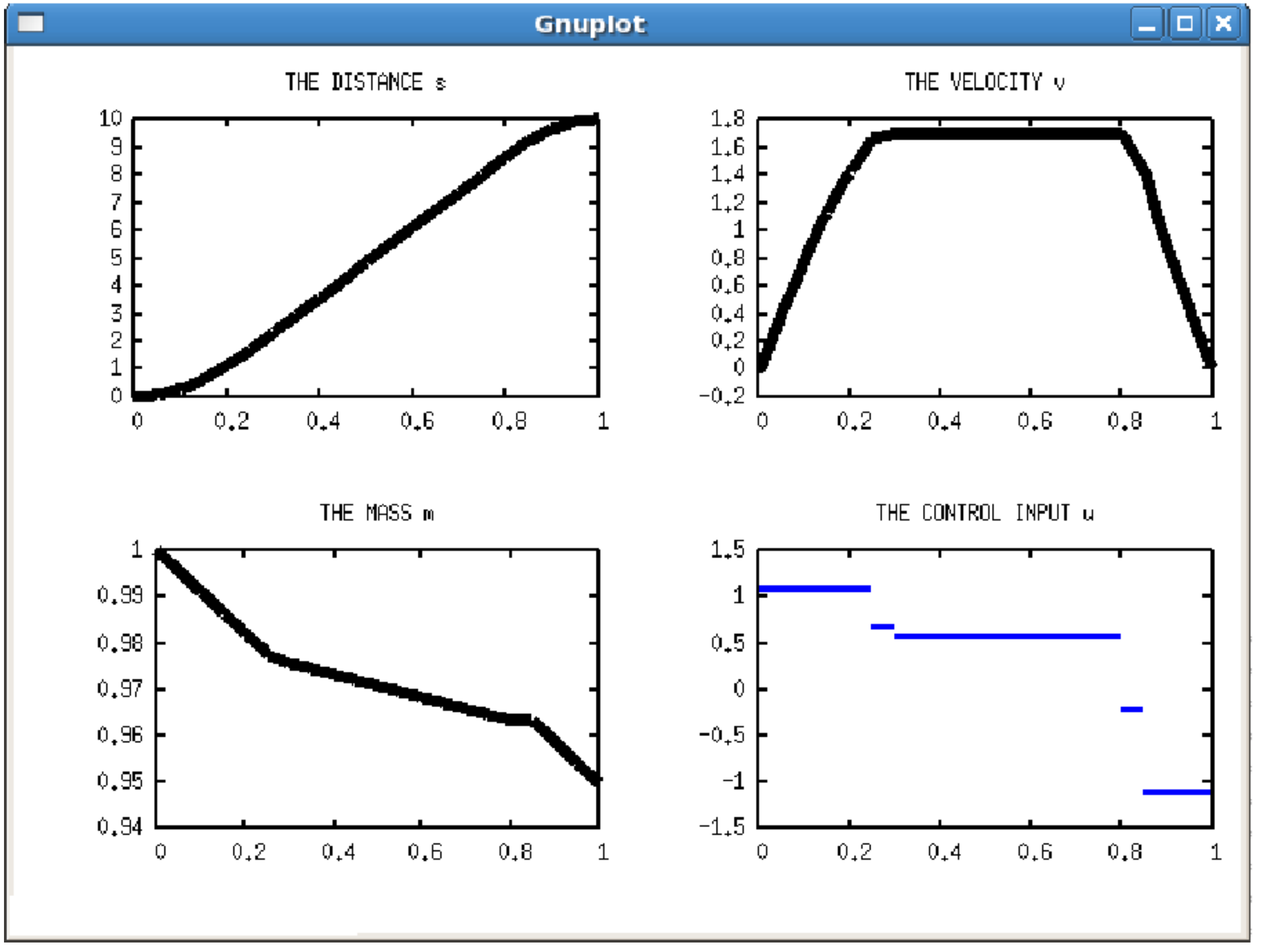
```
DifferentialState                s,v,m;
Control                              u;
Parameter                            T;
DifferentialEquation     f( 0.0, T );
OCP ocp( 0.0, T );
ocp.minimizeMayerTerm( T );

f « dot(s) == v;
f « dot(v) == (u-0.2*v*v)/m;
f « dot(m) == -0.01*u*u;
ocp.subjectTo( f                    );

ocp.subjectTo( AT_START, s ==  0.0 );
ocp.subjectTo( AT_START, v ==  0.0 );
ocp.subjectTo( AT_START, m ==  1.0 );
ocp.subjectTo( AT_END  , s == 10.0 );
ocp.subjectTo( AT_END  , v ==  0.0 );

ocp.subjectTo( -0.1 <= v <=  1.7   );
ocp.subjectTo( -1.1 <= u <=  1.1   );
ocp.subjectTo(  5.0 <= T <= 15.0   );
OptimizationAlgorithm algorithm(ocp);
algorithm.solve();
```

Optimization Results



Implemented Problem Classes in ACADO Toolkit

- **Optimal control of dynamic systems**
  (ODE, DAE)

Implemented Problem Classes in ACADO Toolkit

- **Optimal control of dynamic systems**
  (ODE, DAE)
- **Multi-objective optimization**
  (joint work with Filip Logist)

Implemented Problem Classes in ACADO Toolkit

- **Optimal control of dynamic systems**
  (ODE, DAE)
- **Multi-objective optimization**
  (joint work with Filip Logist)
- **State and parameter estimation/OED**
  (joint work with Dries Telen and Filip Logist)

Implemented Problem Classes in ACADO Toolkit

- **Optimal control of dynamic systems**
  (ODE, DAE)
- **Multi-objective optimization**
  (joint work with Filip Logist)
- **State and parameter estimation/OED**
  (joint work with Dries Telen and Filip Logist)
- **Feedback control (NMPC) and closed loop simulation tools**

Implemented Problem Classes in ACADO Toolkit

- **Optimal control of dynamic systems**
  (ODE, DAE)
- **Multi-objective optimization**
  (joint work with Filip Logist)
- **State and parameter estimation/OED**
  (joint work with Dries Telen and Filip Logist)
- **Feedback control (NMPC) and closed loop simulation tools**
- **Robust optimal control**

Implemented Problem Classes in ACADO Toolkit

- **Optimal control of dynamic systems**
  (ODE, DAE)
- **Multi-objective optimization**
  (joint work with Filip Logist)
- **State and parameter estimation/OED**
  (joint work with Dries Telen and Filip Logist)
- **Feedback control (NMPC) and closed loop simulation tools**
- **Robust optimal control**
- **Real-Time MPC and Code Export**

Overview

- ACADO Toolkit
- **Automatic Code Generation**
- Examples
- Conclusion
- Live Demo (by Rien Quirynen)

From Mathematics to Engineering: ACADO Code Generation

Mathematical Formulation

$$\min_{x,u} \quad \int_0^T x^2 + u^2 \mathrm{d}t$$
$$\text{s.t.} \quad \dot{x} = f(x,u), \quad x(0) = x_0, \quad -1 \leq u \leq 1\,.$$

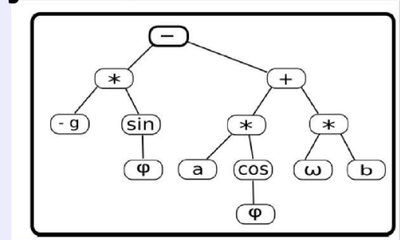
ACADO Syntax

```
DifferentialState     x;
Control               u;

DifferentialEquation f;
f << dot(x) == u + ...;

ocp.minLagrangeTerm( x*x+u*u );
ocp.subjectTo( f );
ocp.subjectTo( -1 <= u <= 1 );
```

Symbolic Structure Detection



Algorithm

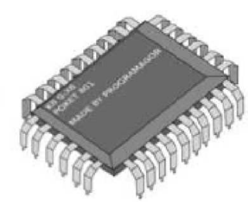
- Multiple Shooting
- Real-Time Gauss Newton
- Online Active Set Strategy

Optimized C-Code

```
r[1] = a[15]*c[17] + a[16]*c[19] + ... ;
r[2] = sin(a[1]*a[2]) + a[4] + ... ;
r[3] = cos(r[1])/exp(c[4])+ r[1] +... ;
```

Customized Solver Implemented on Chip/FPGA:

Measurement x_0



Optimal Decision u*

ACADO Code Generation

Main Idea:

- Automatically generate tailored C code for each specific application
- Faster execution as all overhead is avoided
- Fixing problem dimensions avoids dynamic memory allocation
- Plain C code is highly platform-independent

B. Houska, H.J. Ferreau, and M. Diehl. An auto-generated real-time iteration algorithm for nonlinear MPC in the microsecond range. *Automatica*, 47(10), pp:2279-2285, 2011.

B. Houska, H.J. Ferreau, and M. Diehl. ACADO Toolkit – An Open Source Framework for Automatic Control and Dynamic Optimization. *Optimal Control Applications and Methods*, 32, pp:298-312, 2011.

ACADO Code Generation in Detail

- Export ODE/DAE system and its derivatives as optimized C-code
- Generate a tailored integration method with constant stepsizes

ACADO Code Generation in Detail

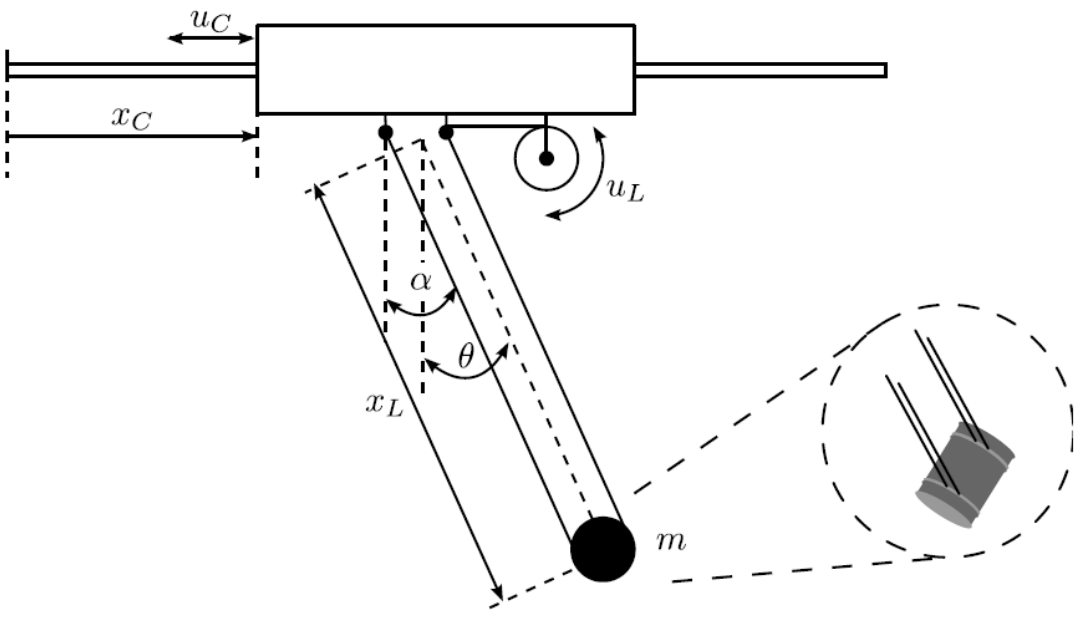
- Export ODE/DAE system and its derivatives as optimized C-code
- Generate a tailored integration method with constant stepsizes
- Generate a discretization algorithm (single- or multiple-shooting)
- Generate a real-time iteration Gauss-Newton method and employ `CVXGEN`, `qpOASES`, `FORCES`, ... (or other QP Solvers)

Overview

- ACADO Toolkit
- Automatic Code Generation
- **Examples**
- Conclusion
- Live Demo (by Rien Quirynen)

Example: Overhead Crane

Model:



6 differential states, 2 control inputs

Overhead Crane: Simulation Time

Simulation over 0.1 s: ACADO $\leftrightarrow$ SUNDIALS [5]

accuracy	IRK2	IRK4	IRK6	CVODES	speedup
1e-1	**12 µs**	18 µs	46 µs	3928 µs	327
1e-2	30 µs	**27 µs**	46 µs	4311 µs	160
1e-3	90 µs	**36 µs**	69 µs	4859 µs	135
1e-4	270 µs	**63 µs**	92 µs	4938 µs	78
1e-5	840 µs	**108 µs**	115 µs	5359 µs	50
1e-6	2700 µs	198 µs	**161 µs**	5766 µs	36
time/step	3 µs	9 µs	23 µs		

[5] Intel P8600 3MB cache, 2.40 GHz

CSTR Benchmark

- We simulate a **continuously stirred tank reactor** described by the following nonlinear ODE:

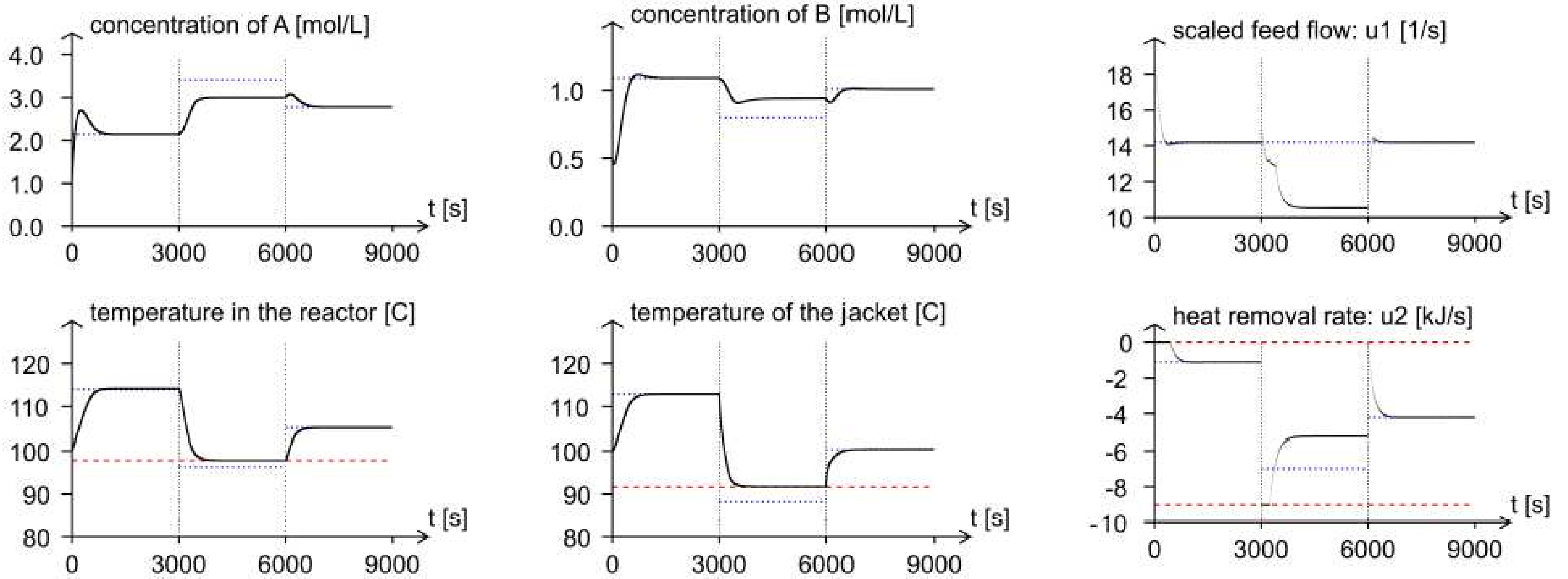
$$
\begin{aligned}
\dot{c}_A(t) &= u_1(c_{A0} - c_A(t)) - k_1(\vartheta(t))c_A(t) - k_3(\vartheta(t))(c_A(t))^2 \\
\dot{c}_B(t) &= -u_1 c_B(t) + k_1(\vartheta(t))c_A(t) - k_2(\vartheta(t))c_B(t) \\
\dot{\vartheta}(t) &= u_1(\vartheta_0 - \vartheta(t)) + \frac{k_w A_R}{\rho C_p V_R}(\vartheta_K(t) - \vartheta(t)) \\
&\quad - \frac{1}{\rho C_p}\left[k_1(\vartheta(t))c_A(t)H_1 + k_2(\vartheta(t))c_B(t)H_2 + k_3(\vartheta(t))(c_A(t))^2 H_3 \right] \\
\dot{\vartheta}_K(t) &= \frac{1}{m_K C_{PK}}\left(u_2 + k_w A_R(\vartheta(t) - \vartheta_K(t))\right)
\end{aligned}
$$

where

$$
k_i(\vartheta(t)) = k_{i0} \cdot \exp\left(\frac{E_i}{\vartheta(t)/^\circ\mathrm{C} + 273.15}\right), \quad i = 1, 2, 3
$$

- 4 states, 2 control inputs, 10 control steps

CSTR Benchmark (cont.)



Run-Time of the Auto-Generated NMPC Algorithm

- For the CSTR example, **one real-time iteration** of the auto-generated NMPC algorithm **takes about 0.2 ms**:

	CPU time	Percentage
Integration & sensitivities	$117\ \mu$s	$65\,$%
Condensing	$31\ \mu$s	$17\,$%
QP solution (with qpOASES)	$28\ \mu$s	$16\,$%
Remaining operations	$< 5\ \mu$s	$< 2\,$%
A complete real-time iteration	$181\ \mu$s	$100\,$%

Conclusion

We can solve optimal control problems really fast.

ACADO Toolkit: www.acadotoolkit.org

B. Houska, H.J. Ferreau, and M. Diehl. An auto-generated real-time iteration algorithm for nonlinear MPC in the microsecond range. *Automatica*, 47(10), pp:2279-2285, 2011.

B. Houska, H.J. Ferreau, and M. Diehl. ACADO Toolkit – An Open Source Framework for Automatic Control and Dynamic Optimization. *Optimal Control Applications and Methods*, 32, pp:298-312, 2011.

Open Postdoc (and PhD) Positions

School of Information Science and Technology

Several Open Postdoc Positions in Optimal Control

- Secure Funding for > 3 Years
- Build Center for Control and Robotics
- International Environment