# ACADO Toolkit - Automatic Control and Dynamic Optimization

Moritz Diehl, Boris Houska, Hans Joachim Ferreau

Optimization in Engineering Center (OPTEC)

K. U. Leuven

# *Overview*

OPTEC

- Scope of ACADO Toolkit
- An Optimal Control Tutorial Example
- Algorithms and Modules in ACADO
- Code Generation
- Outlook

# *Motivation: Optimal Control and Engineering Applications*

OPTEC

## Optimal Control Applications in OPTEC:

- Optimal Robot Control, Kite Control, Solar Power Plants, Bio-chemical reactions...

# Motivation: Optimal Control and Engineering Applications

OPTEC

## Optimal Control Applications in OPTEC:

- Optimal Robot Control, Kite Control, Solar Power Plants, Bio-chemical reactions...

## Need to solve optimal control problems:

$$\begin{aligned}
&\underset{y(\cdot),u(\cdot),p,T}{\text{minimize}} && \int_0^T L(\tau, y(\tau), u(\tau), p)\,\mathrm{d}\tau \;+\; M(y(T), p) \\
&\text{subject to:} && \\
\forall t \in [0,T]: \quad & 0 &=\; & f(t, \dot{y}(t), y(t), u(t), p) \\
& 0 &=\; & r(y(0), y(T), p) \\
\forall t \in [0,T]: \quad & 0 &\geq\; & s(t, y(t), u(t), p)
\end{aligned}$$

# *What is ACADO Toolkit?*

OPTEC

**ACADO Toolkit:**

- **A**utomatic **C**ontrol **A**nd **D**ynamic **O**ptimization
- great variety of numerical optimization algorithms
- open framework for users and developers

# *What is ACADO Toolkit?*

OPTEC

**ACADO Toolkit:**

- **A**utomatic **C**ontrol **A**nd **D**ynamic **O**ptimization
- great variety of numerical optimization algorithms
- open framework for users and developers

**Key Properties of ACADO Toolkit**

- Open Source (LGPL) `www.acadotoolkit.org`
- user interfaces close to mathematical syntax
- Code extensibility: use `C++` capabilities
- Self-containedness: only need `C++` compiler

# *Implemented Problem Classes in ACADO Toolkit*


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

# *Implemented Problem Classes in ACADO Toolkit*

OPTEC

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

# *Implemented Problem Classes in ACADO Toolkit*

OPTEC

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

# *Implemented Problem Classes in ACADO Toolkit*

OPTEC

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

# *Implemented Problem Classes in ACADO Toolkit*

OPTEC

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

# *Implemented Problem Classes in ACADO Toolkit*

OPTEC

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

# *Overview*

OPTEC

- Scope of ACADO Toolkit
- **An Optimal Control Tutorial Example**
- Algorithms and Modules in ACADO
- Code Generation
- Outlook

# 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

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

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

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

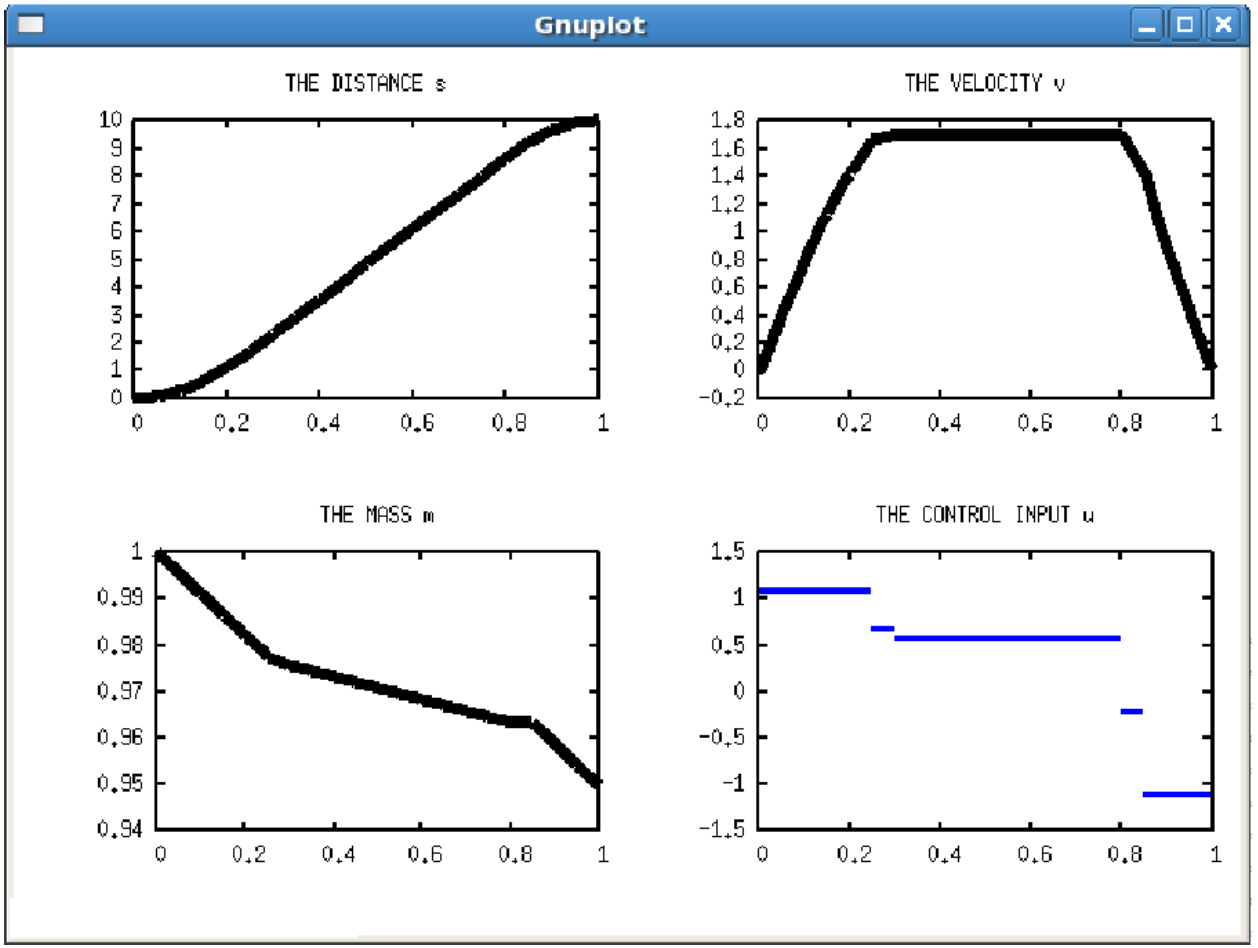
```
DifferentialState                s,v,m;
Control                             u;
Parameter                           T;
DifferentialEquation     f( 0.0, T );
OCP ocp( 0.0, T );
ocp.minimizeMayerTerm( T );

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



# *Overview*

OPTEC

- Scope of ACADO Toolkit
- An Optimal Control Tutorial Example
- **Algorithms and Modules in ACADO**
- Code Generation
- Outlook

# *The Power of Symbolic Functions*

OPTEC

**Symbolic Functions allow:**

- Dependency/Sparsity Detection
- Automatic Differentiation
- Symbolic Differentiation
- Convexity Detection
- Code Optimization
- C-code Generation

# *The Power of Symbolic Functions*

OPTEC

## Symbolic Functions allow:

- Dependency/Sparsity Detection
- Automatic Differentiation
- Symbolic Differentiation
- Convexity Detection
- Code Optimization
- C-code Generation

## Example 1:

```
DifferentialState x;
IntermediateState z;
TIME              t;
Function          f;

z = 0.5*x + 1.0    ;
f « exp(x) + t     ;
f « exp(z+exp(z)) ;

if( f.isConvex() == YES )
printf("f is convex.    ");
```

# Symbolic Functions

## Example 2 (code optimization):

- ```
  Matrix A(3,3);
  Vector b(3);
  DifferentialState x(3);
  Function f;
  A.setZero();
  A(0,0) = 1.0; A(1,1) = 2.0; A(2,2) = 3.0;
  b(0)   = 1.0; b(1)   = 1.0; b(2)   = 1.0;
  f « A*x + b;
  ```
- expect $9$ multiplications $9$ additions to evaluate `f`.
- ACADO needs $3$ multiplications and $3$ additions.
  ```

# *Integration Algorithms*

OPTEC

**DAE simulation and sensitivity generation:**

- several Runge Kutta and a BDF integrator.
- first and second order automatic differentiation.

# *Integration Algorithms*

OPTEC

**DAE simulation and sensitivity generation:**

- several Runge Kutta and a BDF integrator.
- first and second order automatic differentiation.
- The BDF routine solves fully implicit index 1 DAE's:

$$\forall t \in [0, T]: \quad F(t, \dot{y}(t), y(t), u(t), p) = 0 \ .$$

# *Integration Algorithms*

OPTEC

**DAE simulation and sensitivity generation:**

- several Runge Kutta and a BDF integrator.
- first and second order automatic differentiation.
- The BDF routine solves fully implicit index 1 DAE's:

$$\forall t \in [0, T]: \quad F(t, \dot{y}(t), y(t), u(t), p) = 0 \ .$$

- Continous output of trajectories and sensitivities.
- Integrators are also available as stand alone package.
- Sparse LA solvers can be linked.

## Nonlinear Optimal Control Problem

- ACADO solves problem of the general form:

$$\begin{aligned}
&\underset{y(\cdot),u(\cdot),p,T}{\text{minimize}} && \int_0^T L(\tau, y(\tau), u(\tau), p)\,\mathrm{d}\tau \;+\; M(y(T), p) \\
&\text{subject to:} && \\
\forall t \in [0,T]: \quad & 0 &=\; & f(t, \dot{y}(t), y(t), u(t), p) \\
& 0 &=\; & r(y(0), y(T), p) \\
\forall t \in [0,T]: \quad & 0 &\geq\; & s(t, y(t), u(t), p)
\end{aligned}$$

# *Nonlinear Optimization Algorithms*

OPTEC

## Implemented Solution Methods

- Discretization: Single- or Multiple Shooting
- NLP solution: several SQP type methods e.g. with
  - BFGS Hessian Approximations or
  - Gauss-Newton Hessian Approximations
- Globalization: based on line search
- QP solution: active set methods (qp0ASES)

# *Nonlinear Optimization Algorithms*

**Implemented Solution Methods**

- Discretization: Single- or Multiple Shooting
- NLP solution: several SQP type methods e.g. with
  - BFGS Hessian Approximations or
  - Gauss-Newton Hessian Approximations
- Globalization: based on line search
- QP solution: active set methods (qp0ASES)

**Latest Feature:**

- Automatic Code Export for NMPC

# *Overview*

OPTEC

- Scope of ACADO Toolkit
- An Optimal Control Tutorial Example
- Algorithms and Modules in ACADO
- **Code Generation**
- Outlook

# *ACADO Code Generation*

OPTEC

## Main Idea:

- Automatically generate tailored C code for each specific application

# *ACADO Code Generation*

OPTEC

**Main Idea:**

- Automatically generate tailored C code for each specific application

**Advantages:**

- Faster execution as all overhead is avoided
- Fixing problem dimensions avoids dynamic memory allocation
- Plain C code is highly platform-independent

# *ACADO Code Generation in Detail*

OPTEC

- Export ODE system and its derivatives as optimized C-code
- Generate a tailored Runge-Kutta method with constant stepsizes

# *ACADO Code Generation in Detail*

- Export ODE system and its derivatives as optimized C-code
- Generate a tailored Runge-Kutta method with constant stepsizes
- Generate a discretization algorithm (single- or multiple-shooting)
- Generate a real-time iteration Gauss-Newton method and employ CVXGEN or adapted variant of qpOASES

# ACADO Code Generation in Detail (cont.)

```
DifferentialState    p,v,phi,omega;
Control              a;

Matrix Q = eye( 4 );
Matrix R = eye( 1 );

DifferentialEquation f;
f « dot(p)     == v;
f « dot(v)     == a;
f « dot(phi)   == omega;
f « dot(omega) == -g*sin(phi)
                  -a*cos(phi)-b*omega;

OCP ocp( 0.0, 5.0 ,10 );
ocp.minimizeLSQ( Q, R );

ocp.subjectTo( f );
ocp.subjectTo( AT_START, p     == 0.5 );
ocp.subjectTo( AT_START, v     == 0.0 );
ocp.subjectTo( AT_START, phi   == 0.0 );
ocp.subjectTo( AT_START, omega == 0.0 );
ocp.subjectTo( -0.2 <= a <=  0.2 );

OptimizationAlgorithm algorithm(ocp);
algorithm.solve();
```

# ACADO Code Generation in Detail (cont.)

```
DifferentialState    p,v,phi,omega;
Control              a;

Matrix Q = eye( 4 );
Matrix R = eye( 1 );

DifferentialEquation f;
f « dot(p)     == v;
f « dot(v)     == a;
f « dot(phi)   == omega;
f « dot(omega) == -g*sin(phi)
                  -a*cos(phi)-b*omega;

OCP ocp( 0.0, 5.0 ,10 );
ocp.minimizeLSQ( Q, R );

ocp.subjectTo( f );
ocp.subjectTo( AT_START, p     == 0.5 );
ocp.subjectTo( AT_START, v     == 0.0 );
ocp.subjectTo( AT_START, phi   == 0.0 );
ocp.subjectTo( AT_START, omega == 0.0 );
ocp.subjectTo( -0.2 <= a <=  0.2 );

MPCexport mpc( ocp );
mpc.exportCode( "name" );
```

# Run-Time of the Auto-Generated NMPC Algorithm

- We simulate the simple crane ODE model: 4 states, 1 control input, 10 control steps
- **One real-time iteration** of the auto-generated NMPC algorithm **takes less than 0.1 ms**:

| | CPU time | Percentage |
|---|---|---|
| Integration & sensitivities | $34\ \mu$s | $63\,\%$ |
| Condensing | $11\ \mu$s | $20\,\%$ |
| QP solution (with qp0ASES) | $5\ \mu$s | $9\,\%$ |
| Remaining operations | $< 5\ \mu$s | $< 8\,\%$ |
| One complete real-time iteration | $54\ \mu$s | $100\,\%$ |

# Another Example: CSTR Benchmark

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

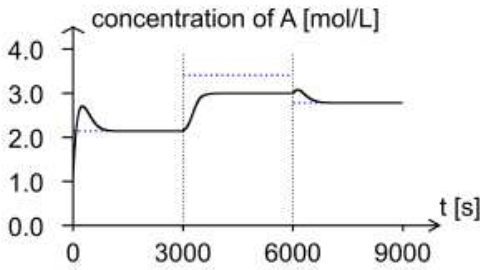
$$
\begin{aligned}
\dot{c}_A(t) &= u_1(c_{A0} - c_A(t)) - k_1(\vartheta(t))c_A(t) - k_3(\vartheta(t))(c_A(t))^2 \\
\dot{c}_B(t) &= -u_1 c_B(t) + k_1(\vartheta(t))c_A(t) - k_2(\vartheta(t))c_B(t) \\
\dot{\vartheta}(t) &= u_1(\vartheta_0 - \vartheta(t)) + \frac{k_w A_R}{\rho C_p V_R}(\vartheta_K(t) - \vartheta(t))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

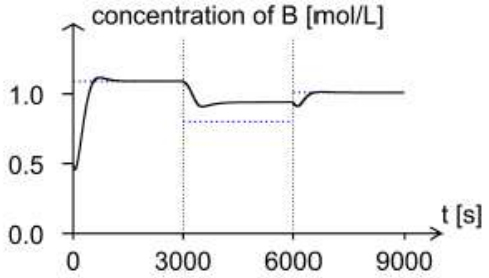
where

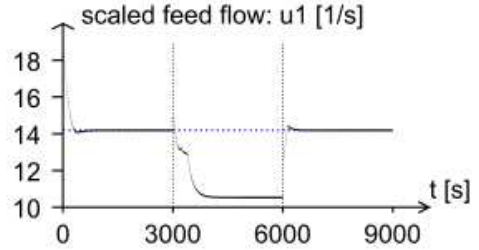
$$
k_i(\vartheta(t)) = k_{i0} \cdot \exp\left(\frac{E_i}{\vartheta(t)/^\circ\mathrm{C} + 273.15}\right), \quad i = 1, 2, 3
$$

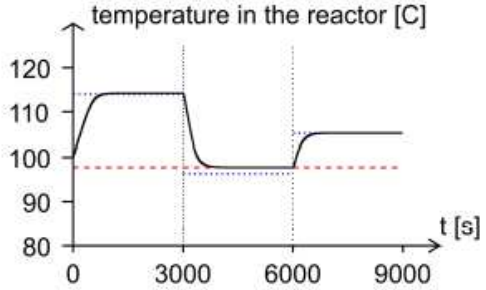
- 4 states, 2 control inputs, 10 control steps

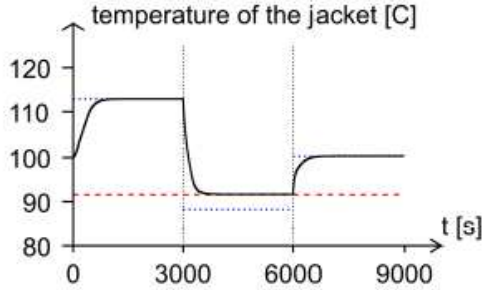
# Another Example: CSTR Benchmark (cont.)

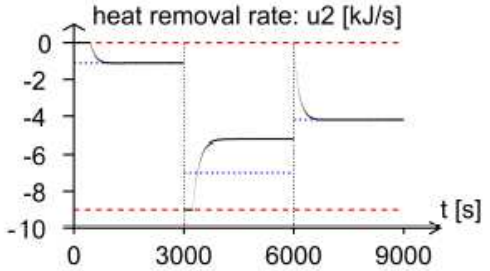


# Run-Time of the Auto-Generated NMPC Algorithm

- 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 qp0ASES) | $28\ \mu$s | $16\,\%$ |
| Remaining operations | $< 5\ \mu$s | $< 2\,\%$ |
| A complete real-time iteration | $181\ \mu$s | $100\,\%$ |

# *Summary*


## Highlights of ACADO

- Self contained C++ code
- Automatic differentiation, integration routines
- Single- and multiple shooting
- SQP methods (exact Hessian, GN, BFGS,...)
- Easy setup of optimal control problems
- Real-time iteration algorithms and code export

OPTEC

Thank you for your attention!