ACADO TOOLKIT - AUTOMATIC CONTROL AND DYNAMIC OPTIMIZATION

Boris Houska, Hans Joachim Ferreau, Filip Logist, Moritz Diehl

Optimization in Engineering Center (OPTEC)

K. U. Leuven
Overview

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

Optimal Control Applications in OPTEC:

- Optimal Robot Control, Kite Control, Solar Power Plants, Bio-chemical reactions...
Motivation: Optimal Control and Engineering Applications

Optimal Control Applications in OPTEC:

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

Need to solve optimal control problems:

\[
\begin{align*}
\text{minimize} & \quad \int_{0}^{T} L(\tau, y(\tau), u(\tau), p) \, d\tau + M(y(T), p) \\
\text{subject to:} & \\
\forall t \in [0, T] : & 0 = f(t, \dot{y}(t), y(t), u(t), p) \\
& 0 = r(y(0), y(T), p) \\
\forall t \in [0, T] : & 0 \geq s(t, y(t), u(t), p)
\end{align*}
\]
What is ACADO Toolkit?

ACADO Toolkit:

• **Automatic Control And Dynamic Optimization**
• great variety of numerical optimization algorithms
• open framework for users and developers
What is ACADO Toolkit?

ACADO Toolkit:

- Automatic Control And Dynamic Optimization
- 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

- 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
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
- 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
- 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
- Feedback control (NMPC) and closed loop simulation tools
- Robust optimal control
- Real-Time MPC and Code Export
Overview

- 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:

\[
\begin{align*}
\text{minimize} & \quad T \\
\text{subject to} & \quad \dot{s}(t) = v(t) \\
& \quad \dot{v}(t) = \frac{u(t) - 0.2 v(t)^2}{m(t)} \\
& \quad \dot{m}(t) = -0.01 u(t)^2 \\
\end{align*}
\]

\[
\begin{align*}
s(0) &= 0 \quad s(T) = 10 \\
v(0) &= 0 \quad v(T) = 0 \\
m(0) &= 1 \\
-0.1 &\leq v(t) \leq 1.7 \\
-1.1 &\leq u(t) \leq 1.1 \\
5 &\leq T \leq 15
\end{align*}
\]
Tutorial Example: Time Optimal Control of a Rocket

Mathematical Formulation:

\[
\begin{align*}
\text{minimize} & \quad T \\
\text{subject to} & \quad \dot{s}(t) = v(t) \\
& \quad \dot{v}(t) = \frac{u(t) - 0.2v(t)^2}{m(t)} \\
& \quad \dot{m}(t) = -0.01u(t)^2 \\
& \quad s(0) = 0, \quad s(T) = 10 \\
& \quad v(0) = 0, \quad v(T) = 0 \\
& \quad m(0) = 1 \\
& \quad -0.1 \leq v(t) \leq 1.7 \\
& \quad -1.1 \leq u(t) \leq 1.1 \\
& \quad 5 \leq T \leq 15
\end{align*}
\]

DifferentialState \( s, v, m \);
Control \( u \);
Parameter \( T \);
DifferentialEquation \( f(0.0, T) \);
OCP \( ocp(0.0, T) \);
\( ocp.\text{minimizeMayerTerm}(T) \);
\( f \prec \dot{s}(t) = v \);
\( f \prec \dot{v}(t) = \frac{(u-0.2v^2)}{m} \);
\( f \prec \dot{m}(t) = -0.01u^2 \);
\( ocp.\text{subjectTo}(f) \);
\( ocp.\text{subjectTo}(\text{AT}\_\text{START}, s = 0.0) \);
\( ocp.\text{subjectTo}(\text{AT}\_\text{START}, v = 0.0) \);
\( ocp.\text{subjectTo}(\text{AT}\_\text{START}, m = 1.0) \);
\( ocp.\text{subjectTo}(\text{AT}\_\text{END}, s = 10.0) \);
\( ocp.\text{subjectTo}(\text{AT}\_\text{END}, v = 0.0) \);
\( ocp.\text{subjectTo}(-0.1 \leq v \leq 1.7) \);
\( ocp.\text{subjectTo}(-1.1 \leq u \leq 1.1) \);
\( ocp.\text{subjectTo}(5.0 \leq T \leq 15.0) \);
OptimizationAlgorithm \( \text{algorithm}(ocp) \);
\( \text{algorithm.solve}() \);
Optimization Results
Overview

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

Symbolic Functions allow:

• Dependency/Sparsity Detection
• Automatic Differentiation
• Symbolic Differentiation
• Convexity Detection
• Code Optimization
• C-code Generation
Symbolic Functions allow:

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

Example 1:

```c
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

DAE simulation and sensitivity generation:

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

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

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

\[
\begin{align*}
\text{minimize} & \quad \mathcal{J} = \int_0^T L(\tau, y(\tau), u(\tau), p) \, d\tau + M(y(T), p) \\
\text{subject to:} & \\
\forall t \in [0, T] : \quad 0 &= f(t, \dot{y}(t), y(t), u(t), p) \\
0 &= r(y(0), y(T), p) \\
\forall t \in [0, T] : \quad 0 &\geq s(t, y(t), u(t), p)
\end{align*}
\]
Nonlinear Optimization Algorithms

Implemented Solution Methods

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

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

Latest Feature:

- Automatic Code Export for NMPC
Overview

• Scope of ACADO Toolkit
• An Optimal Control Tutorial Example
• Algorithms and Modules in ACADO
• Code Generation
• Outlook
Main Idea:

- Automatically generate tailored C code for each specific application
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

- 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
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( -0.2 <= a <= 0.2 );
OptimizationAlgorithm algorithm(ocp);
algorithm.solve();
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( -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:

<table>
<thead>
<tr>
<th></th>
<th>CPU time</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integration &amp; sensitivities</td>
<td>34 µs</td>
<td>63 %</td>
</tr>
<tr>
<td>Condensing</td>
<td>11 µs</td>
<td>20 %</td>
</tr>
<tr>
<td>QP solution (with qpOASES)</td>
<td>5 µs</td>
<td>9 %</td>
</tr>
<tr>
<td>Remaining operations</td>
<td>&lt; 5 µs</td>
<td>&lt; 8 %</td>
</tr>
<tr>
<td>One complete real-time iteration</td>
<td>54 µs</td>
<td>100 %</td>
</tr>
</tbody>
</table>
Another Example: CSTR Benchmark

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

\[
\begin{align*}
\dot{c}_A(t) &= u_1(c_0 - c_A(t)) - k_1(\vartheta(t))c_A(t) - k_3(\vartheta(t))(c_A(t))^2 \\
\dot{c}_B(t) &= -u_1c_B(t) + k_1(\vartheta(t))c_A(t) - k_2(\vartheta(t))c_B(t) \\
\dot{\vartheta}(t) &= u_1(\vartheta_0 - \vartheta(t)) + \frac{k_w A_R}{\rho C_p V_R} (\vartheta_K(t) - \vartheta(t)) \\
&\quad - \frac{1}{\rho C_p} \left[ k_1(\vartheta(t))c_A(t)H_1 + k_2(\vartheta(t))c_B(t)H_2 + k_3(\vartheta(t))(c_A(t))^2 H_3 \right] \\
\dot{\vartheta}_K(t) &= \frac{1}{m_K C_{PK}} (u_2 + k_w A_R (\vartheta(t) - \vartheta_K(t)))
\end{align*}
\]

where

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

- 4 states, 2 control inputs, 10 control steps
Another Example: CSTR Benchmark (cont.)

- Concentration of A [mol/L]
- Concentration of B [mol/L]
- Temperature in the reactor [°C]
- Temperature of the jacket [°C]
- Scaled feed flow: $u_1$ [1/s]
- Heat removal rate: $u_2$ [kJ/s]
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**: 

<table>
<thead>
<tr>
<th></th>
<th>CPU time</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integration &amp; sensitivities</td>
<td>117 $\mu$s</td>
<td>65%</td>
</tr>
<tr>
<td>Condensing</td>
<td>31 $\mu$s</td>
<td>17%</td>
</tr>
<tr>
<td>QP solution (with qpOASES)</td>
<td>28 $\mu$s</td>
<td>16%</td>
</tr>
<tr>
<td>Remaining operations</td>
<td>$&lt; 5$ $\mu$s</td>
<td>$&lt; 2%$</td>
</tr>
<tr>
<td>A complete real-time iteration</td>
<td>181 $\mu$s</td>
<td>100%</td>
</tr>
</tbody>
</table>
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
Thank you for your attention!