SOLVING OPTIMAL CONTROL PROBLEMS WITH ACADO TOOLKIT

Boris Houska, Moritz Diehl

Optimization in Engineering Center (OPTEC)
K. U. Leuven

Trondheim, June 07, 2010
Overview

- Scope of ACADO Toolkit
- An Optimal Control Tutorial Example (with Software Demo)
- Algorithms and Modules in ACADO
- Outlook
Optimal Control Applications in OPTEC:

- Optimal Robot Control, Kite Control, Solar Power Plants, Bio-chemical reactions...
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](http://www.acadotoolkit.org)
- user interfaces close to mathematical syntax
- Code extensibility: use C++ capabilities
- Self-containedness: only need C++ compiler
• Optimal control of dynamic systems (ODE, DAE, hybrid models)
Implemented Problem Classes in ACADO Toolkit

- Optimal control of dynamic systems (ODE, DAE, hybrid models)
- Multi-objective optimization (joint work with Filip Logist)
Implemented Problem Classes in ACADO Toolkit

- Optimal control of dynamic systems (ODE, DAE, hybrid models)
- Multi-objective optimization (joint work with Filip Logist)
- State and parameter estimation (Niels Haverbeke, Ioanna Stamati, Boris Houska)
Implemented Problem Classes in ACADO Toolkit

- Optimal control of dynamic systems (ODE, DAE, hybrid models)
- Multi-objective optimization (joint work with Filip Logist)
- State and parameter estimation (Niels Haverbeke, Ioanna Stamati, Boris Houska)
- Feedback control based on real time optimization (with Hans Joachim Ferreau)
Implemented Problem Classes in ACADO Toolkit

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

- **Multi-objective optimization**
  (joint work with Filip Logist)

- **State and parameter estimation**
  (Niels Haverbeke, Ioanna Stamati, Boris Houska)

- **Feedback control based on real time optimization**
  (with Hans Joachim Ferreau)

- **Robust optimal control**
  (my PhD. thesis)
Overview

- Scope of ACADO Toolkit
- An Optimal Control Tutorial Example (with Software Demo)
- Algorithms and Modules in ACADO
- Outlook
Tutorial Example: Time Optimal Control of a Rocket

Mathematical Formulation:

\[
\begin{align*}
\text{minimize} & \quad T \\
\text{subject to} & \quad \dot{s}(t) = v(t) \\
& \quad \dot{v}(t) = \frac{u(t) - 0.2 \cdot v(t)^2}{m(t)} \\
& \quad \dot{m}(t) = -0.01 \cdot u(t)^2 \\
& \quad s(0) = 0 \quad s(T) = 10 \\
& \quad v(0) = 0 \quad v(T) = 0 \\
& \quad m(0) = 1 \\
& \quad -0.1 \leq v(t) \leq 1.7 \\
& \quad -1.1 \leq u(t) \leq 1.1 \\
& \quad 5 \leq T \leq 15
\end{align*}
\]
Tutorial Example: Time Optimal Control of a Rocket

Mathematical Formulation:

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

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

\[
\begin{align*}
f &\leftarrow \text{dot}(s) = v; \\
f &\leftarrow \text{dot}(v) = (u-0.2*v*v)/m; \\
f &\leftarrow \text{dot}(m) = -0.01*u*u; \\
ocp.&\text{subjectTo}( f ) \\
ocp.&\text{subjectTo}( \text{AT\_START}, s = 0.0 ); \\
ocp.&\text{subjectTo}( \text{AT\_START}, v = 0.0 ); \\
ocp.&\text{subjectTo}( \text{AT\_START}, m = 1.0 ); \\
ocp.&\text{subjectTo}( \text{AT\_END} , s = 10.0 ); \\
ocp.&\text{subjectTo}( \text{AT\_END} , v = 0.0 ); \\
ocp.&\text{subjectTo}( -0.1 <= v <= 1.7 ); \\
ocp.&\text{subjectTo}( -1.1 <= u <= 1.1 ); \\
ocp.&\text{subjectTo}( 5.0 <= T <= 15.0 ); \\
\text{OptimizationAlgorithm} &\text{algorithm}(ocp); \\
\text{algorithm.solve();}
\end{align*}
\]
Jojo Model

- Position/velocity of the hand: $x, v$
- Position/velocity of the hand: $y, w$
- Control the acceleration $u$ of the hand
Tutorial Example: Optimal Control of a Jojo

Jojo Model

• Position/velocity of the hand: $x, v$
• Position/velocity of the hand: $y, w$
• Control the acceleration $u$ of the hand

\[
\begin{pmatrix}
\dot{x}(t) \\
\dot{v}(t) \\
\dot{y}(t) \\
\dot{w}(t)
\end{pmatrix}
= \begin{pmatrix}
v(t) \\
u(t) \\
w(t) \\
-\mu g + ku(t) + a(w(t) - v(t))
\end{pmatrix}
\]

• damping ratio: $k = \frac{J}{mr^2 + J}$, effective mass ratio: $\mu = 1 - \frac{1}{k}$, roll friction coefficient $a$
Interesting aspect of the jojo model:

- At the time $t_1$ for which $x(t_1) - y(t_1) = L$:

$$w(t_1) = \lim_{t \to t_1^-} kv(t) + \sqrt{k^2 v(t)^2 + k(w(t)^2 - 2v(t)w(t))}$$
Interesting aspect of the jojo model:

- At the time $t_1$ for which $x(t_1) - y(t_1) = L$:

$$w(t_1) = \lim_{t \to t_1} k v(t) + \sqrt{k^2 v(t)^2 + k(w(t)^2 - 2v(t)w(t))}$$

Our aim:

- Minimize the control action $\Phi := \int_0^{t_2} u^2(t) \, dt$
- Satisfy all model equations and constraints:

\[
\begin{align*}
    x(0) &= y(0) = 0 \text{ m} & v(0) &= w(0) = 0 \text{ m/s} \\
    x(t_1) - y(t_1) &= L \\
    x(t_2) &= y(t_2) = -0.1 \text{ m} & v(t_2) &= w(t_2) = 0 \text{ m/s}
\end{align*}
\]
Formulation of multi-stage optimal control problems

\[
\begin{align*}
\text{minimize} & \quad \Phi(x(\cdot), u(\cdot), p, T) \\
\text{subject to:} & \\
\forall t \in \mathbb{T}_i : & \quad 0 = F_i(t, x(t), \dot{x}(t), u(t), p) \\
& \quad 0 = \lim_{t \to t_k^-} J_k(x(t), x(t_k), p) \\
\forall t \in \mathbb{T}_i : & \quad 0 \leq h_i(t, x(t), u(t), p) \\
& \quad 0 = r(x(t_1), \ldots, x(t_{n+1}), p) \\
\text{for all} & \quad i \in I, k \in K
\end{align*}
\]
Tutorial Example: Optimal Control of a Jojo

- Position of the Hands (x)
- Velocity of the Hands (v)
- Position of the Jojo (y)
- Velocity of the Jojo (ω)
- Control Input (u)

Solving Optimal Control Problems with ACADO Toolkit – p. 11/19
Overview

- Scope of ACADO Toolkit
- An Optimal Control Tutorial Example (with Software Demo)
- Algorithms and Modules in ACADO
- 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 12 additions to evaluate $f$.
- ACADO needs 3 multiplications and 6 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:

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

Implemented Solution Methods

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

Currently under development

• Collocation methods, Interior point methods
• SCP methods, Lifted Newton methods, ...
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
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

In Exercises:

• Simulation, Optimal Control, MPC, ...
• Please install from www.acadotoolkit.org.
Thank you for your attention!