Part 1:

- Scope of ACADO Toolkit
- An Optimal Control Tutorial Example (with Software Demo)
- Algorithms and Modules in ACADO
Overview

Part 1:
- Scope of ACADO Toolkit
- An Optimal Control Tutorial Example (with Software Demo)
- Algorithms and Modules in ACADO

Part 2:
- A Parameter Estimation Tutorial Example
- A Simple Model Predictive Control Simulation (with Software Demo)
- Outlook
Overview

Part 1:
- **Scope of ACADO Toolkit**
- An Optimal Control Tutorial Example (with Software Demo)
- Algorithms and Modules in ACADO

Part 2:
- A Parameter Estimation Tutorial Example
- A Simple Model Predictive Control Simulation (with Software Demo)
- Outlook
Optimal Control Applications in OPTEC:

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

#### Optimal Control Applications in OPTEC:
- Optimal Robot Control, Kite Control, Solar Power Plants
- Batch Distillation Processes, Bio-chemical reactions...

#### Ubiquitous Need for Nonlinear Optimal Control Software

Existing Packages:
- IPOPT (C++, open source, collocation, interior point method)
- MUSCOD (Fortran/C, proprietary, Multiple Shooting, SQP)
- PROPT (commercial Matlab software, collocation, SQP)
- DSOA (C/C++, open-source, single shooting, SQP)
- ...
Motivation: Optimal Control and Engineering Applications

Optimal Control Applications in OPTEC:

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

Umbiqous Need for Nonlinear Optimal Control Software

Existing Packages:

- IPOPT (C++, open source, collocation, interior point method)
- MUSCOD (Fortran/C, proprietary, Multiple Shooting, SQP)
- PROPT (commercial Matlab software, collocation, SQP)
- DSOA (C/C++, open-source, single shooting, SQP)
- ...

All packages have their particular strengths in a specific range of applications, but ...
Motivation for ACADO Toolkit

Most of the existing Packages are ...

- either not open-source or limited in their user-friendliness
- difficult to install - especially on embedded hardware
- not designed for closed loop MPC applications
- hard to extend with specialized algorithms
Motivation for ACADO Toolkit

Most of the existing Packages are ...

- either not open-source or limited in their user-friendliness
- difficult to install - especially on embedded hardware
- not designed for closed loop MPC applications
- hard to extend with specialized algorithms

Key Properties of ACADO Toolkit

- Open Source (LGPL)  www.acadotoolkit.org
- user friendly interfaces close to mathematical syntax
- Code extensibility: use C++ capabilities
- Self-containedness: only need C++ compiler
Problem Classes and the Scope of ACADO

Optimal Control of Dynamic Systems

- Objectives: Mayer and/or Lagrange terms.
- Differential and algebraic equations.
- Initial value-, terminal-, path- and boundary constraints.
### Optimal Control of Dynamic Systems

- Objectives: Mayer and/or Lagrange terms.
- Differential and algebraic equations.
- Initial value-, terminal-, path- and boundary constraints.

### State and Parameter Estimation

- Estimation of model parameters of DAE’s.
- A posteriori analysis: Computation of variance-covariances.
**Problem Classes and the Scope of ACADO**

**Optimal Control of Dynamic Systems**
- Objectives: Mayer and/or Lagrange terms.
- Differential and algebraic equations.
- Initial value-, terminal-, path- and boundary constraints.

**State and Parameter Estimation**
- Estimation of model parameters of DAE’s.
- A posteriori analysis: Computation of variance-covariances.

**Feedback control based on real-time optimization (MPC/MHE)**
- Computation of current process state using measurements.
- Computation of optimal control action in real-time.
Part 1:
- Scope of ACADO Toolkit
- An Optimal Control Tutorial Example (with Software Demo)
- Algorithms and Modules in ACADO

Part 2:
- A Parameter Estimation Tutorial Example
- A Simple Model Predictive Control Simulation (with Software Demo)
- Outlook
A tutorial example: time optimal control of a rocket

A simple rocket model

- Three differential states: $s$, $v$, and $m$.
- Control input: $u$
- Dynamic equations (model):

  \[
  \begin{align*}
  \dot{s}(t) &= v(t) \\
  \dot{v}(t) &= \left[u(t) - 0.2 \, v(t)^2\right] / m(t) \\
  \dot{m}(t) &= -0.01 \, u(t)^2 .
  \end{align*}
  \]
A tutorial example: time optimal control of a rocket

A simple rocket model

- Three differential states: \( s, v, \) and \( m \).
- Control input: \( u \)
- Dynamic equations (model):

\[
\begin{align*}
\dot{s}(t) &= v(t) \\
\dot{v}(t) &= \left[ u(t) - 0.2 \, v(t)^2 \right] / m(t) \\
\dot{m}(t) &= -0.01 \, u(t)^2.
\end{align*}
\]

Aim:

- Fly in minimum time \( T \) from \( s(0) = 0 \) to \( s(T) = 10 \).
- Start/land at rest: \( v(0) = 0, \quad v(T) = 0 \).
- Start with \( m(0) = 1 \) and satisfy \( v(t) \leq 1.7 \).
- Satisfy control constraints: \(-1.1 \leq u(t) \leq 1.1 \).
A tutorial example: time optimal control of a rocket

Mathematical Formulation:

minimize \[ T \]
\[ s(\cdot), v(\cdot), m(\cdot), u(\cdot) \]

subject to

\[ \dot{s}(t) = v(t) \]
\[ \dot{v}(t) = \frac{u(t) - 0.2 \, v(t)^2}{m(t)} \]
\[ \dot{m}(t) = -0.01 \, u(t)^2 \]

\[ s(0) = 0 \quad s(T) = 10 \]
\[ v(0) = 0 \quad v(T) = 0 \]
\[ m(0) = 1 \]
\[ -0.1 \leq v(t) \leq 1.7 \]
\[ -1.1 \leq u(t) \leq 1.1 \]
\[ 5 \leq T \leq 15 \]
A tutorial example: time optimal control of a rocket

Mathematical Formulation:

\[
\begin{align*}
\text{minimize} & \quad T \\
\text{subject to} & \quad s(t) = v(t) \\
& \quad \dot{v}(t) = \frac{u(t) - 0.2v(t)^2}{m(t)} \\
& \quad \dot{m}(t) = -0.01u(t)^2 \\
& \quad s(0) = 0, \quad s(T) = 10 \\
& \quad v(0) = 0, \quad v(T) = 0 \\
& \quad m(0) = 1 \\
& \quad -0.1 \leq v(t) \leq 1.7 \\
& \quad -1.1 \leq u(t) \leq 1.1 \\
& \quad 5 \leq T \leq 15
\end{align*}
\]
A 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 \( ocp(0.0, T) \); 
\( ocp\text{.minimizeMayerTerm}(T) \); 
\( f \ll \text{dot}(s) == v; \) 
\( f \ll \text{dot}(v) == (u-0.2*v^2)/m; \) 
\( f \ll \text{dot}(m) == -0.01*u^2; \) 
\( ocp\text{.subjectTo}(f) \);
A tutorial example: time optimal control of a rocket

Mathematical Formulation:

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

\[
\begin{align*}
s(0) &= 0 & s(T) &= 10 \\
v(0) &= 0 & v(T) &= 0 \\
m(0) &= 1 \\
-0.1 & \leq v(t) \leq 1.7 \\
-1.1 & \leq u(t) \leq 1.1 \\
5 & \leq T \leq 15 \\
\end{align*}
\]

DifferentialState \( s, v, m \);
Control \( u \);
Parameter \( T \);
DifferentialEquation \( f(0.0, T) \);
OCP ocp(0.0, T);
ocp.minimizeMayerTerm(T);
\[
\begin{align*}
f & \ll \text{dot}(s) = v; \\
f & \ll \text{dot}(v) = (u - 0.2 v v) / m; \\
f & \ll \text{dot}(m) = -0.01 u u; \\
\end{align*}
\]

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);
A tutorial example: time optimal control of a rocket

Mathematical Formulation:

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

DifferentialState $s,v,m$; 
Control $u$; 
Parameter $T$; 
DifferentialEquation $f(0.0, T)$; 
OCP $ocp(0.0, T)$; 
$ocp$.minimizeMayerTerm($T$); 
\begin{verbatim}
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();
\end{verbatim}
A tutorial example: time optimal control of a rocket

Graphical Output:

On the terminal:

<table>
<thead>
<tr>
<th>#</th>
<th>KKT tol.</th>
<th>Obj. Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.001e+03</td>
<td>1.000e+01</td>
</tr>
<tr>
<td>2</td>
<td>5.766e+00</td>
<td>9.950e+00</td>
</tr>
<tr>
<td>3</td>
<td>2.946e-02</td>
<td>9.932e+00</td>
</tr>
<tr>
<td>4</td>
<td>7.481e-02</td>
<td>9.906e+00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>8.740e-04</td>
<td>7.442e+00</td>
</tr>
<tr>
<td>13</td>
<td>3.308e-07</td>
<td>7.442e+00</td>
</tr>
</tbody>
</table>

convergence achieved.
Overview

Part 1:
  - Scope of ACADO Toolkit
  - An Optimal Control Tutorial Example (with Software Demo)
  - **Algorithms and Modules in ACADO**

Part 2:
  - A Parameter Estimation Tutorial Example
  - A Simple Model Predictive Control Simulation (with Software Demo)
  - Outlook
Symbolic Functions allow:

- Dependency/Sparsity Detection
- Automatic Differentiation
- Symbolic Differentiation
- Convexity Detection
- Code Optimization
- C-code Generation
The Power of Symbolic Functions

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() == BT_TRUE )
printf("f is convex. ");
```
Symbolic Functions

Example 2 (code optimization):

Matrix A(3,3);
Vector b(3);
DifferentialStateVector 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;

- We would expect 12 flops to evaluate f.
- ACADO Toolkit needs only 6 flops.
Integration Algorithms

DAE simulation and sensitivity generation

- ACADO provides several Runge Kutta and a BDF integrator.
- All integrators provide first and second order numeric and automatic internal numerical differentiation.
DAE simulation and sensitivity generation

- ACADO provides several Runge Kutta and a BDF integrator.
- All integrators provide first and second order numeric and automatic internal numerical differentiation.
- BDF integrator uses diagonal implicit Runge Kutta starter.
- The BDF routine can deal with fully implicit index 1 DAE’s:

\[ \forall t \in [0, T] : \quad F(\dot{y}(t), y(t), u(t), p, T) = 0. \]
DAE simulation and sensitivity generation

- ACADO provides several Runge Kutta and a BDF integrator.
- All integrators provide first and second order numeric and automatic internal numerical differentiation.
- BDF integrator uses diagonal implicit Runge Kutta starter.
- The BDF routine can deal with fully implicit index 1 DAE's:

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

- The Integrators are also available as a stand alone package.
- Sparse LA solvers can be linked.
Nonlinear Optimal Control Problem

ACADO solves problem of the general form:

\[
\begin{align*}
\text{minimize} & \quad y(\cdot), u(\cdot), p, T \\
& \quad \int_0^T L(\tau, y(\tau), u(\tau), p) \, d\tau + M(y(T), p) \\
\text{subject to:} & \\
\forall t \in [0, T]: & \quad 0 = f(t, \dot{y}(t), y(t), u(t), p) \\
& \quad 0 = r(y(0), y(T), p) \\
\forall t \in [0, T]: & \quad 0 \geq s(t, y(t), u(t), p)
\end{align*}
\]
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
- Sequential convex optimization techniques
- Lifted Newton methods
- ...
Using ACADO Toolkit for Parameter Estimation and Model Predictive Control

Boris Houska, Hans Joachim Ferreau, Moritz Diehl

Electrical Engineering Department
K.U. Leuven

OPTEC Seminar, 2/9/2009
Overview

Part 1:
- Scope of ACADO Toolkit
- An Optimal Control Tutorial Example (with Software Demo)
- Algorithms and Modules in ACADO

Part 2:
- A Parameter Estimation Tutorial Example
- A Simple Model Predictive Control Simulation (with Software Demo)
- Outlook
ACADO Toolkit can solve parameter estimation problems of the following form:

\[
\begin{align*}
\text{minimize} & \quad \sum_{i=1}^{N} \| h(t_i, x(t_i), z(t_i), u(t_i), p) - \eta_i \|^2_{S_i} \\
\text{subject to:} & \\
\forall t \in [0, T] : & \quad \dot{x}(t) = f(t, x(t), z(t), u(t), p) \\
\forall t \in [0, T] : & \quad 0 = g(t, x(t), z(t), u(t), p) \\
\forall t \in [0, T] : & \quad 0 = r(x(0), z(0), x(T), z(T), p) \\
\forall t \in [0, T] : & \quad 0 \geq s(t, x(t), z(t), u(t), p)
\end{align*}
\]
Tutorial Example: A Simple Pendulum

- **Simple pendulum** model describing the excitation angle $\phi$ governed by the following ODE:

$$\ddot{\phi}(t) = -\frac{g}{l}\phi(t) - \alpha \dot{\phi}(t)$$

where $l$ is the length of the line, $\alpha$ the friction coefficient and $g$ the gravitational constant.
Simple pendulum model describing the excitation angle $\phi$ governed by the following ODE:

$$\ddot{\phi}(t) = -\frac{g}{l}\phi(t) - \alpha \dot{\phi}(t)$$

where $l$ is the length of the line, $\alpha$ the friction coefficient and $g$ the gravitational constant.

Aim is to estimate $l$ and $\alpha$ from ten measurements of the state $\phi$. 
Tutorial Example: A Simple Pendulum

Mathematical Formulation:

\[
\begin{align*}
\text{minimize} \quad & \sum_{i=1}^{10} (\phi(t_i) - \eta_i)^2 \\
\text{subject to:} \quad & \forall t \in [0, 2] : \ddot{\phi}(t) = -\frac{g}{l} \phi(t) - \alpha \dot{\phi}(t) \\
& \quad 0 \leq \alpha \leq 4 \\
& \quad 0 \leq l \leq 2
\end{align*}
\]
## Mathematical Formulation:

\[
\begin{align*}
\text{minimize} & \quad \sum_{i=1}^{10} (\phi(t_i) - \eta_i)^2 \\
\text{subject to:} & \quad \forall t \in [0, 2] : \ddot{\phi}(t) = -\frac{g}{l} \phi(t) - \alpha \dot{\phi}(t) \\
& \quad 0 \leq \alpha \leq 4 \\
& \quad 0 \leq l \leq 2
\end{align*}
\]

## C++ Implementation:

```cpp
DifferentialState phi, dphi;
Parameter l, alpha;
const double g = 9.81;
DifferentialEquation f;
Function h;

OCP ocp(0.0, 2.0);
h << phi;
ocp.minimizeLSQ(h, "data.txt");

f << dot(phi) == dphi;
f << dot(dphi) == -(g/l) * sin(phi) - alpha * dphi;

ocp.subjectTo(f);
ocp.subjectTo(0.0 <= alpha <= 4.0);
ocp.subjectTo(0.0 <= l <= 2.0);

ParameterEstimationAlgorithm alg(ocp);
alg.solve();
```
Tutorial Example: A Simple Pendulum

Mathematical Formulation:

\[
\begin{align*}
\text{minimize} & \quad \sum_{i=1}^{10} (\phi(t_i) - \eta_i)^2 \\
\text{subject to:} & \\
\forall t \in [0, 2] : & \quad \ddot{\phi}(t) = -\frac{g}{l} \phi(t) - \alpha \dot{\phi}(t) \\
& \quad 0 \leq \alpha \leq 4 \\
& \quad 0 \leq l \leq 2
\end{align*}
\]

C++ Implementation:

```cpp
DifferentialState phi, dphi;
Parameter l, alpha;
const double g = 9.81;
DifferentialEquation f;
Function h;
OCP ocp(0.0, 2.0);
h << phi;
ocp.minimizeLSQ(h, "data.txt");
f << dot(phi) == dphi;
f << dot(dphi) == -(g/l) * sin(phi) - alpha * dphi;
ocp.subjectTo(f);
ocp.subjectTo(0.0 <= alpha <= 4.0);
ocp.subjectTo(0.0 <= l <= 2.0);
ParameterEstimationAlgorithm alg(ocp);
alg.solve();
```
**Tutorial Example: A Simple Pendulum**

**Mathematical Formulation:**

\[
\begin{align*}
\text{minimize} & \quad \phi(\cdot), \alpha, l \\
& \quad \sum_{i=1}^{10} (\phi(t_i) - \eta_i)^2 \\
\text{subject to:} & \\
& \forall t \in [0, 2] : \quad \ddot{\phi}(t) = -\frac{g}{l} \phi(t) - \alpha \dot{\phi}(t) \\
& \quad 0 \leq \alpha \leq 4 \\
& \quad 0 \leq l \leq 2
\end{align*}
\]

**C++ Implementation:**

```cpp
#include <Optimization/ACADO.hpp>

DifferentialState phi, dphi;
Parameter l, alpha;
const double g = 9.81;
DifferentialEquation f;
Function h;
OCP ocp(0.0, 2.0);
    h << phi;
ocp.minimizeLSQ(h, "data.txt");
    f << dot(phi) == dphi;
    f << dot(dphi) == -(g/l) * sin(phi) - alpha * dphi;
oxp.subjectTo(f);
oxp.subjectTo(0.0 <= alpha <= 4.0);
oxp.subjectTo(0.0 <= l <= 2.0);
ParameterEstimationAlgorithm alg(xp);
    alg.solve();
```

ACADO Toolkit Introduction — Boris Houska, Hans Joachim Ferreau, Moritz Diehl
Parameter estimation problems are (nonlinear) least-square problems with objective function $\frac{1}{2} \| F(x) \|_2^2$.

Parameter estimation problems are solved using the constrained Gauss-Newton method.

Newton-type method where the Hessian matrix is approximated by

$$
\left( \frac{\partial F(x)}{\partial x} \right)^T \left( \frac{\partial F(x)}{\partial x} \right)
$$

The constrained Gauss-Newton method works well for:

- small residual problems
- almost linear problems
**Data file** `data.txt`:

<table>
<thead>
<tr>
<th>TIME POINTS</th>
<th>MEASUREMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00000e+00</td>
<td>1.00000e+00</td>
</tr>
<tr>
<td>2.72321e-01</td>
<td>nan</td>
</tr>
<tr>
<td>3.72821e-01</td>
<td>5.75146e-01</td>
</tr>
<tr>
<td>7.25752e-01</td>
<td>-5.91794e-02</td>
</tr>
<tr>
<td>9.06107e-01</td>
<td>-3.54347e-01</td>
</tr>
<tr>
<td>1.23651e+00</td>
<td>-3.03056e-01</td>
</tr>
<tr>
<td>1.42619e+00</td>
<td>nan</td>
</tr>
<tr>
<td>1.59469e+00</td>
<td>-9.64208e-02</td>
</tr>
<tr>
<td>1.72029e+00</td>
<td>-1.97671e-02</td>
</tr>
<tr>
<td>2.00000e+00</td>
<td>9.35138e-02</td>
</tr>
</tbody>
</table>

**Fitting Results:**

- $l = 1.001e+00 \pm 1.734e-01$
- $\alpha = 1.847e+00 \pm 4.059e-01$
Overview

Part 1:
- Scope of ACADO Toolkit
- An Optimal Control Tutorial Example (with Software Demo)
- Algorithms and Modules in ACADO

Part 2:
- A Parameter Estimation Tutorial Example
- A Simple Model Predictive Control Simulation (with Software Demo)
- Outlook
Simulation Environment of ACADO Toolkit

ACADO Toolkit Introduction — Boris Houska, Hans Joachim Ferreau, Moritz Diehl
ACADO Toolkit can solve **model predictive control problems** of the following form:

\[
\begin{align*}
\text{minimize} & \quad \int_{0}^{T} \left\| y(t) - y_{\text{ref}}(t) \right\|_{Q}^{2} + \left\| u(t) - u_{\text{ref}}(t) \right\|_{R}^{2} \, \text{d}t \\
& \quad + \left\| y(T) - y_{\text{ref}}(T) \right\|_{P}^{2} \\
\text{subject to:} & \quad x(0) = x_{0} \\
& \forall t \in [0, T]: \quad \dot{x}(t) = f(t, x(t), z(t), u(t), p) \\
& \forall t \in [0, T]: \quad 0 = g(t, x(t), z(t), u(t), p) \\
& \forall t \in [0, T]: \quad y(t) = h(t, x(t), z(t), u(t), p) \\
& \forall t \in [0, T]: \quad 0 \geq s(t, x(t), z(t), u(t), p)
\end{align*}
\]
Each MPC problem might be solved till convergence

Preferably, the **real-time iteration scheme** is employed:
- Only one real-time SQP step per MPC loop
- Initial value embedding
- Division into feedback and preparation phase
Each MPC problem might be solved till convergence

Preferably, the **real-time iteration scheme** is employed:
- Only one real-time SQP step per MPC loop
- Initial value embedding
- Division into feedback and preparation phase

Model based feedback control often requires an online **state estimator**

Moving Horizon Estimation (MHE) and Kalman filters will be implemented
A Simple MPC Simulation

- First principle **quarter car** model **with active suspension**, four states $x_b, x_w, v_b, v_w$ describing vertical position/velocity of body/wheel
- Control input: limited damping force $F$ to act between body and wheel
- External disturbance: road excitation $R$
First principle **quarter car** model **with active suspension**, four states $x_b, x_w, v_b, v_w$ describing vertical position/velocity of body/wheel

- Control input: limited damping force $F$ to act between body and wheel

- External disturbance: road excitation $R$

- **Simulation scenario:** road excitation set to zero, body has initial displacement of 1 cm

- **Aim:** Bring body and wheel back to rest with zero displacement
Mathematical Formulation:

\[
\begin{align*}
\text{minimize} & \quad \mathbf{x}_b, \mathbf{x}_w, \mathbf{v}_b, \mathbf{v}_w, F \\
& \quad \int_0^1 \|y(t)\|_Q^2 \, d\tau \\
\text{subject to:} & \\
\forall t \in [0, 1]: & \quad \dot{x}_b(t) = v_b(t) \\
\forall t \in [0, 1]: & \quad \dot{x}_w(t) = v_w(t) \\
\forall t \in [0, 1]: & \quad \dot{v}_b(t) = f_1(x_b(t), x_w(t), F(t)) \\
\forall t \in [0, 1]: & \quad \dot{v}_w(t) = f_2(x_b(t), x_w(t), F(t), R(t)) \\
\forall t \in [0, 1]: & \quad y(t) = (x_b(t), x_w(t), v_b(t), v_w(t))^T \\
\forall t \in [0, 1]: & \quad -500 \leq u(t) \leq 500
\end{align*}
\]
A Simple MPC Simulation

**Mathematical Formulation:**

\[
\begin{align*}
\text{minimize} & \quad \int_{0}^{1} \left\| y(t) \right\|_Q^2 \, dt \\
\text{subject to:} & \quad \forall t \in [0, 1] : \dot{x}_b(t) = v_b(t) \\
& \quad \forall t \in [0, 1] : \dot{x}_w(t) = v_w(t) \\
& \quad \forall t \in [0, 1] : \dot{v}_b(t) = f_1(x_b(t), x_w(t), F(t)) \\
& \quad \forall t \in [0, 1] : \dot{v}_w(t) = f_2(x_b(t), x_w(t), F(t), R(t)) \\
& \quad \forall t \in [0, 1] : y(t) = (x_b(t), x_w(t), v_b(t), v_w(t))^T \\
& \quad \forall t \in [0, 1] : -500 \leq u(t) \leq 500 \\
\end{align*}
\]

**C++ Implementation:**

```cpp
DifferentialState xB, xW, vB, vW;
Control F;
Disturbance R;
DifferentialEquation f;
//...
Function y;
y << xB;
y << xW;
y << vB;
y << vW;
Matrix Q(4,4);
Q.setIdentity();
OCP ocp( 0.0, 1.0, 20 );
ocp.minimizeLSQ( Q, y );
ocp.subjectTo( f );
ocp.subjectTo( -500 <= F <= 500 );
ocp.subjectTo( R == 0.0 );
```

ACADO Toolkit Introduction — Boris Houska, Hans Joachim Ferreau, Moritz Diehl
A Simple MPC Simulation

Mathematical Formulation:

\[
\begin{align*}
\text{minimize} & \quad \int_{0}^{1} \| y(t) \|_{Q}^{2} \, d\tau \\
\text{subject to:} & \\
\forall t \in [0, 1]: & \quad \dot{x}_b(t) = v_b(t) \\
\forall t \in [0, 1]: & \quad \dot{x}_w(t) = v_w(t) \\
\forall t \in [0, 1]: & \quad \dot{v}_b(t) = f_1(x_b(t), x_w(t), F(t)) \\
\forall t \in [0, 1]: & \quad \dot{v}_w(t) = f_2(x_b(t), x_w(t), F(t), R(t)) \\
\forall t \in [0, 1]: & \quad y(t) = (x_b(t), x_w(t), v_b(t), v_w(t))^T \\
\forall t \in [0, 1]: & \quad -500 \leq u(t) \leq 500 
\end{align*}
\]

C++ Implementation:

```cpp
DifferentialState xB, xW, vB, vW;
Control F;
Disturbance R;
DifferentialEquation f;

//...

Function y;
y << xB;
y << xW;
y << vB;
y << vW;

Matrix Q(4,4);
Q.setIdentity();

OCP ocp( 0.0, 1.0, 20 );
ocp.minimizeLSQ( Q, y );

ocp.subjectTo( f );
ocp.subjectTo( -500 <= F <= 500 );
ocp.subjectTo( R == 0.0 );
```
A Simple MPC Simulation

Mathematical Formulation:

\[
\begin{align*}
\text{minimize} & \quad \int_0^1 \|y(t)\|^2_Q \, dt \\
\text{subject to:} & \\
\forall t \in [0, 1]: & \quad \dot{x}_b(t) = v_b(t) \\
\forall t \in [0, 1]: & \quad \dot{x}_w(t) = v_w(t) \\
\forall t \in [0, 1]: & \quad \dot{v}_b(t) = f_1(x_b(t), x_w(t), F(t)) \\
\forall t \in [0, 1]: & \quad \dot{v}_w(t) = f_2(x_b(t), x_w(t), F(t), R(t)) \\
\forall t \in [0, 1]: & \quad y(t) = (x_b(t), x_w(t), v_b(t), v_w(t))^T \\
\forall t \in [0, 1]: & \quad -500 \leq u(t) \leq 500
\end{align*}
\]

C++ Implementation:

```cpp
DifferentialState xB, xW, vB, vW;
Control F;
Disturbance R;
DifferentialEquation f;

Matrix Q(4,4);
Q.setIdentity();

OCP ocp( 0.0, 1.0, 20 );
ocp.minimizeLSQ( Q, y );

ocp.subjectTo( f );
ocp.subjectTo( -500 <= F <= 500 );
ocp.subjectTo( R == 0.0 );
```

ACADO Toolkit Introduction — Boris Houska, Hans Joachim Ferreau, Moritz Diehl
A Simple MPC Simulation

Simulation Setup:

C++ Implementation:

```cpp
OutputFcn identity;
DynamicSystem dynamicSystem( f, identity );
Process process( dynamicSystem, INT_RK45 );
```
Simulation Setup:

C++ Implementation:

```cpp
OutputFcn identity;
DynamicSystem dynamicSystem( f,identity );
Process process( dynamicSystem,INT_RK45 );
RealTimeAlgorithm alg( ocp );
DynamicFeedbackLaw feedbackLaw( alg,0.05 );
Estimator trivialEstimator;
StaticReferenceTrajectory zeroReference;
Controller controller( feedbackLaw,trivialEstimator,zeroReference );
```
Simulation Setup:

C++ Implementation:

OutputFcn identity;
DynamicSystem dynamicSystem( f, identity );

Process process( dynamicSystem, INT_RK45 );

RealTimeAlgorithm alg( ocp );
DynamicFeedbackLaw feedbackLaw( alg, 0.05 );

Estimator trivialEstimator;
StaticReferenceTrajectory zeroReference;

Controller controller( feedbackLaw, trivialEstimator, zeroReference );

SimulationEnvironment sim( 0.0, 3.0, process, controller );

Vector x0(4);
x0(0) = 0.01;
sim.init( x0 );
sim.run( );
Overview

Part 1:
- Scope of ACADO Toolkit
- An Optimal Control Tutorial Example (with Software Demo)
- Algorithms and Modules in ACADO

Part 2:
- A Parameter Estimation Tutorial Example
- A Simple Model Predictive Control Simulation (with Software Demo)
- Outlook
Outlook

**Algorithmic extensions** currently under development:
- Collocation schemes
- Convex optimization algorithms
- Nonlinear interior point solver for solving NLPs
- Sequential convex programming algorithms
- State estimators for feedback control (MHE/Kalman)
**Outlook**

- **Algorithmic extensions** currently under development:
  - Collocation schemes
  - Convex optimization algorithms
  - Nonlinear interior point solver for solving NLPs
  - Sequential convex programming algorithms
  - State estimators for feedback control (MHE/Kalman)

- **Matlab interfaces** for Integrators and Optimal Control Problems
Outlook

- Additional **problem classes**:  
  - Multi-stage formulations  
  - Robust optimization  
  - Multi-objective problems  
  - Optimum experimental design

- Modular design of ACADO Toolkit allows for easy combination of different algorithmic features