CasADi: A Tool for Automatic Differentiation and Simulation-Based Nonlinear Programming

Moritz Diehl*

with Joel Andersson*, Joris Gillis*, Johan Akesson**

*OPTEC, KU Leuven, Belgium

**Lund University / Modelon

LCCC, Sept 20, 2012

KATHOLIEKE UNIVERSITEIT LEUVEN

OPTEC K.U.Leuven Optimization in Engineering Center

EMBOCON

ERC HIGHWIND SIMULATION, OPTIMIZATION & CONTROL OF HIGH-ALTITUDE WIND POWER GENERATORS

ITN SADCO Initial Training Network Sensitivity Analysis for Deterministic Controller Design

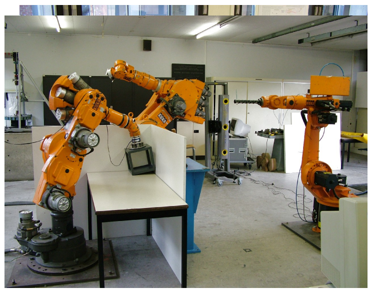
OPTEC - Optimization in Engineering Center

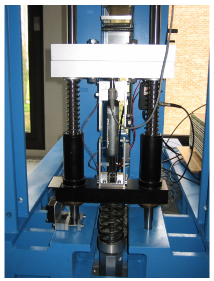
***Center of Excellence of KU Leuven**, since 2005*

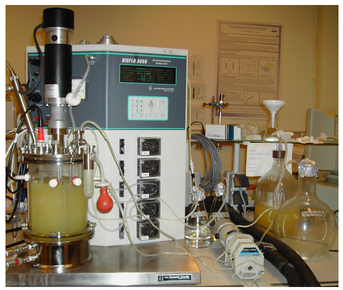
*70 people, working jointly on **methods and applications of optimization,** in 5 departments:*

- Electrical Engineering
- Mechanical Engineering
- Chemical Engineering
- Computer Science
- Civil Engineering

Many real world applications at OPTEC...







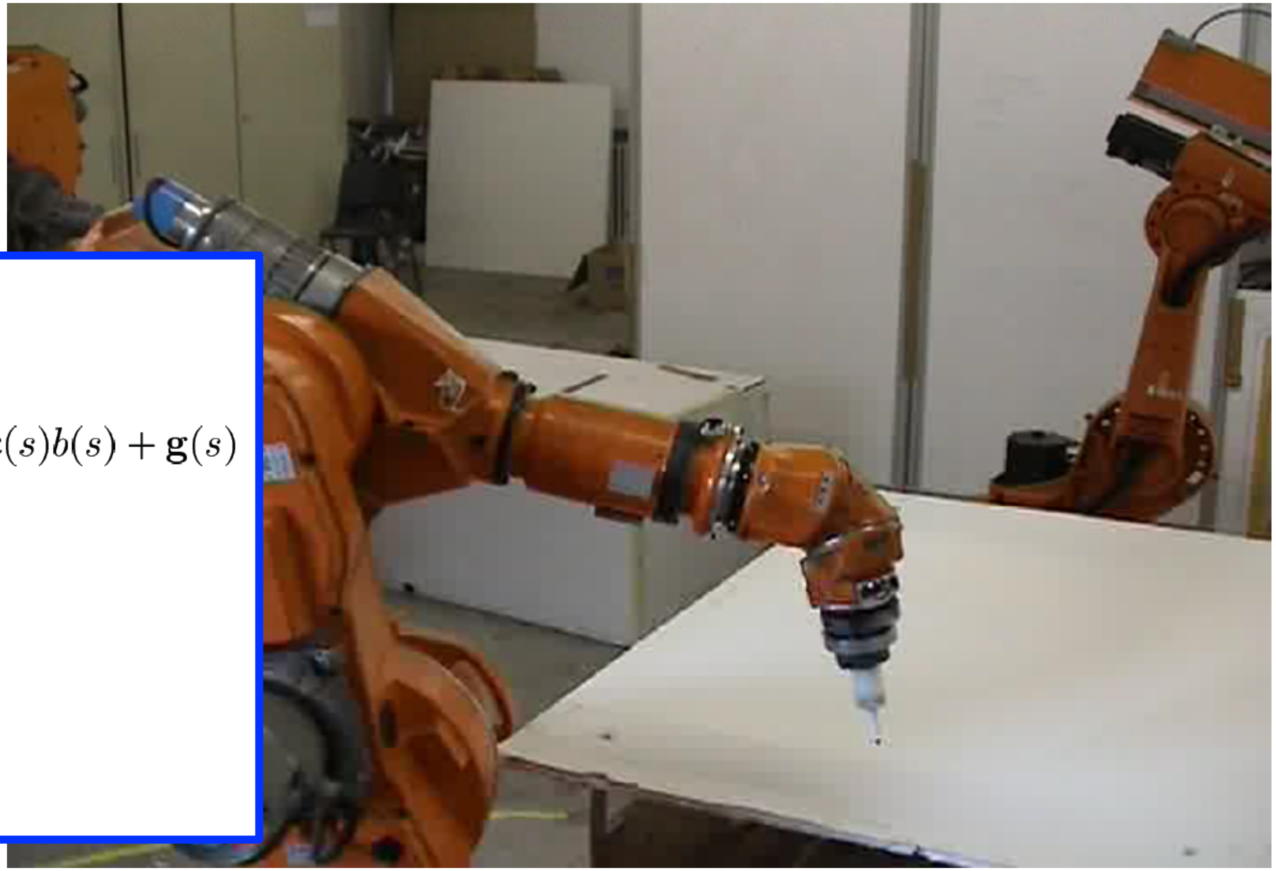
OPTEC Research Example: Time Optimal Robot Motion

Robot shall write as fast as possible.
Global solution found in 2 ms due to ***convex reformulation***

IEEE TRANSACTIONS ON AUTOMATIC CONTROL, VOL. 54, NO. 10, OCTOBER 2009

Time-Optimal Path Tracking for Robots: A Convex Optimization Approach

Diederik Verscheure, Bram Demeulenaere, Jan Swevers, Joris De Schutter, and Moritz Diehl



$$
\begin{aligned}
\min_{a(\cdot),b(\cdot),\boldsymbol{\tau}(\cdot)} \quad & \int_0^1 \frac{1}{\sqrt{b(s)}} ds \\
\text{subject to} \quad & \boldsymbol{\tau}(s) = \mathbf{m}(s)a(s) + \mathbf{c}(s)b(s) + \mathbf{g}(s) \\
& b(0) = \dot{s}_0^2 \\
& b(1) = \dot{s}_T^2 \\
& b'(s) = 2a(s) \\
& b(s) \geq 0 \\
& \underline{\boldsymbol{\tau}}(s) \leq \boldsymbol{\tau}(s) \leq \overline{\boldsymbol{\tau}}(s) \\
& \text{for } s \in [0,1].
\end{aligned}
$$

Overview

- Optimization in Engineering Center OPTEC
- **State of the Art in Optimal Control Algorithms (ACADO)**
- CasADi: A Framework to WRITE Optimal Control Algorithms

Optimal Control Problem in Continuous Time

$$\underset{x(\cdot),\,u(\cdot)}{\text{minimize}} \qquad \int_0^T L(x(t),u(t))\,dt \;+\; E\,(x(T))$$

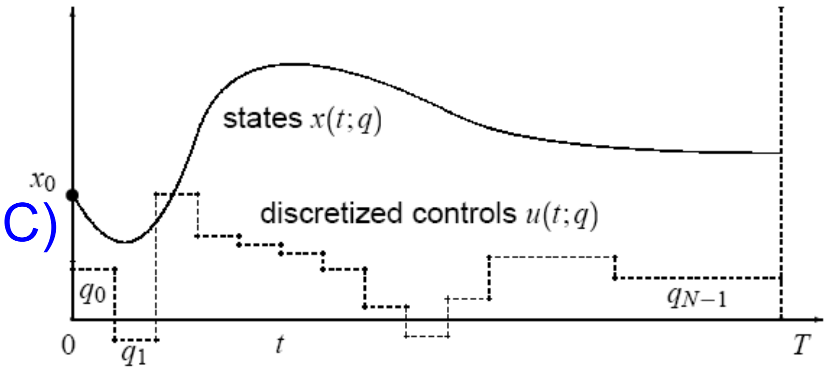
subject to

$$\begin{aligned}
x(0) - x_0 &= 0, && && \text{(fixed initial value)}\\
\dot{x}(t) - f(x(t),u(t)) &= 0, && t \in [0,T], && \text{(ODE model)}\\
h(x(t),u(t)) &\geq 0, && t \in [0,T], && \text{(path constraints)}\\
r\,(x(T)) &= 0 && && \text{(terminal constraints).}
\end{aligned}$$

How to solve these nonlinear problems reliably and fast?

Sequential Approach (Single Shooting): Eliminate States

$$\begin{array}{lrcl}\underset{u}{\text{minimize}} & \sum_{i=0}^{N-1} L_i(\tilde{x}_i(u), \tilde{z}_i(u), u_i) & + & E\left(\tilde{x}_N(u)\right) \\ \text{subject to} & h_i(\tilde{x}_i(u), \tilde{z}_i(u), u_i) & \leq & 0, \quad i = 0, \ldots, N-1, \\ & r\left(\tilde{x}_N(u)\right) & \leq & 0. \end{array}$$



Pros:

- Only control degrees of freedom (for NMPC)
- Can couple with "Vanilla NLP" solver

Cons:

- Sparsity of problem lost
- Unstable systems cannot be treated

Historically first "direct" approach ("single shooting", Sargent&Sullivan 1978)

Simultaneous Approach: Keep States in NLP

INTERNATIONAL FEDERATION OF AUTOMATIC CONTROL
9TH WORLD CONGRESS

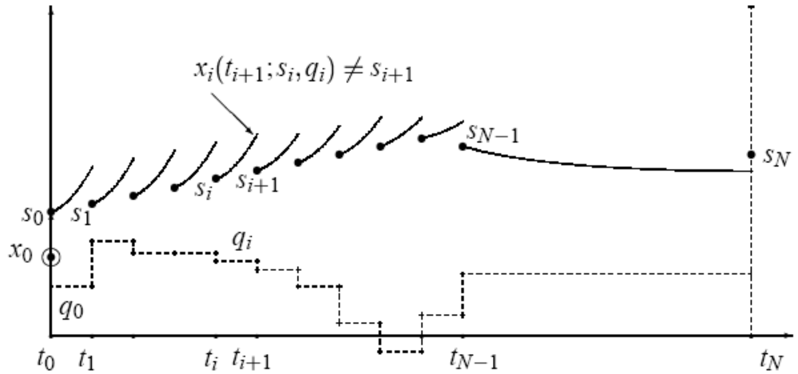
BUDAPEST, HUNGARY
JULY 2-6 1984

A MULTIPLE SHOOTING ALGORITHM FOR DIRECT SOLUTION OF OPTIMAL CONTROL PROBLEMS*

Hans Georg Bock and Karl J. Plitt

Institut für Angewandte Mathematik, SFB 72, Universität Bonn, 5300 Bonn, Federal Republic of Germany

Variants:
Direct Multiple Shooting and Collocation



Pros:

- Sparsity of problem kept
- Unstable systems can be treated, nonlinearity reduced

Cons:

- Large scale problems
- Need to develop (or use) structure exploiting NLP solver

Nonlinear Program (NLP) in Multiple Shooting

$$
\begin{array}{llcll}
\underset{x,\,z,\,u}{\text{minimize}} & \displaystyle\sum_{i=0}^{N-1} L_i(x_i, z_i, u_i) & + & E\,(x_N) & \\
\text{subject to} & x_0 - \bar{x}_0 & = & 0, & \\
& x_{i+1} - f_i(x_i, z_i, u_i) & = & 0, & i = 0, \ldots, N-1, \\
& g_i(x_i, z_i, u_i) & = & 0, & i = 0, \ldots, N-1, \\
& h_i(x_i, z_i, u_i) & \leq & 0, & i = 0, \ldots, N-1, \\
& r\,(x_N) & \leq & 0. &
\end{array}
$$

Structured parametric Nonlinear Program

- Initial Value $\bar{x}_0$ is often not known beforehand (“online data” in NMPC)
- Discrete time dynamics from ODE simulation (we will need sensitivities!)

Sequential Convex Programming (SCP)

- Summarize problem as
$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & f(x) \\ \text{s.t.} \quad & g(x) + M\xi = 0, \\ & x \in \Omega, \end{aligned}$$
with convex f and Ω

Step 1: Linearize nonlinear constraints at x^k to obtain convex problem:

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & f(x) \\ \text{s.t.} \quad & g(x^k) + g'(x^k)(x - x^k) + M\xi = 0, \\ & x \in \Omega. \end{aligned}$$

Step 2: Solve convex problem to obtain next iterate.

Obtain new value of parameter ξ and go to step 1)

- Convergence to (and tracking of) local minima under mild assumptions [1]

[1] Tran Dinh, Savorgnan, Diehl: Adjoint-based predictor-corrector SCP for parametric nonlinear optimization. *SIAM Journal on Optimization* (in print)

ACADO Toolkit [1]

- ACADO = Automatic Control and Dynamic Optimization

- Open source (LGPL) C++:   www. acadotoolkit. org
- Implements direct multiple shooting [2] and real-time iterations [3]
- User interface close to mathematical syntax
- *Automatic C-Code Export for Microsecond Nonlinear MPC* [4]

- Developed at OPTEC by B. Houska, H.J. Ferreau, M. Vukov, ...
- ~3000 downloads since first release in 2009

[1] Houska, Ferreau, D., *OCAM,* 2011
[2] Bock, Plitt, *IFAC WC*, 1984
[3] D., Bock, Schloder, Findeisen, Nagy, Allgower, *JPC*, 2002
[4] Houska, Ferreau, D., *Automatica,* 2011



Rocket Example in ACADO Language

Mathematical Formulation:

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

subject to

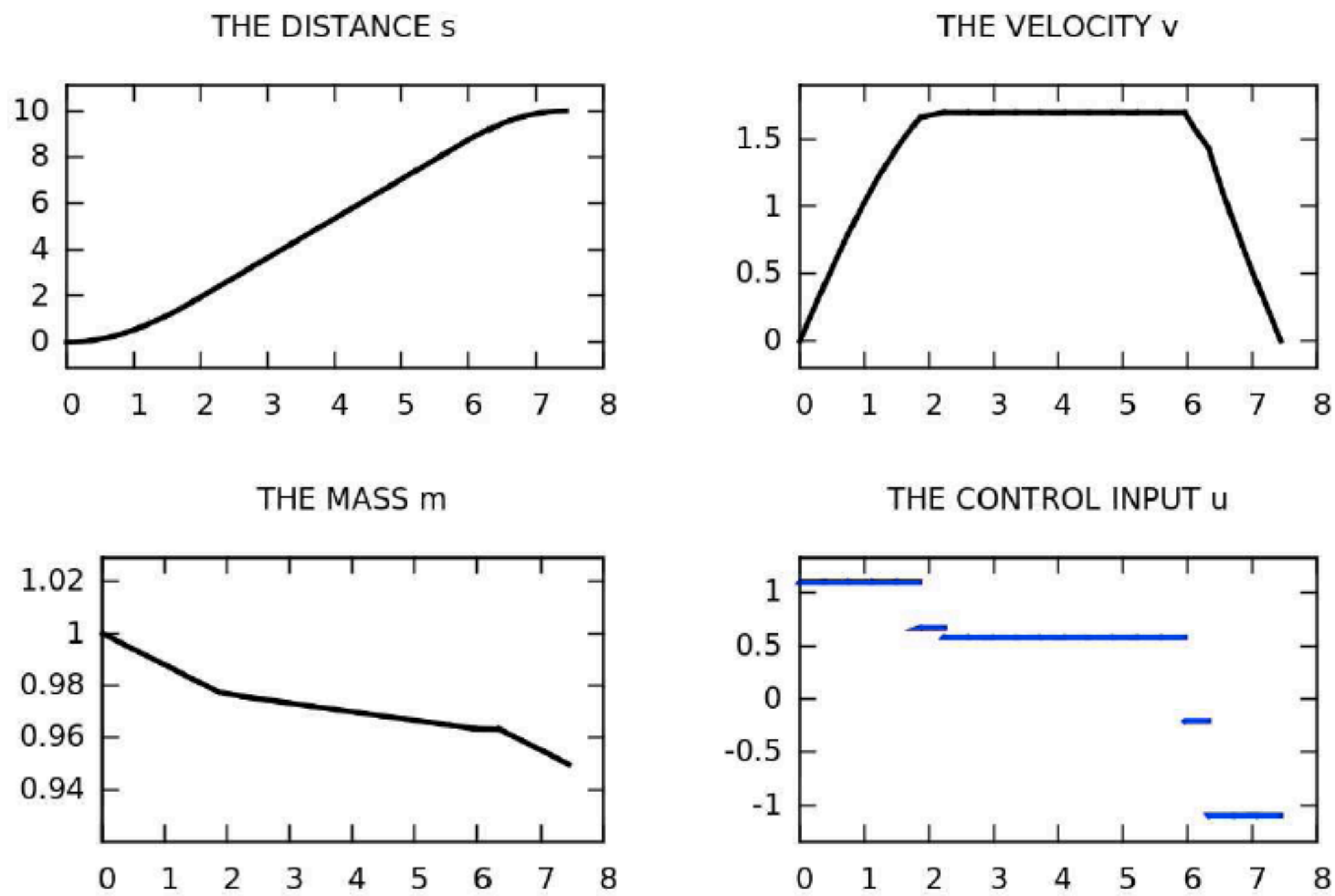
$$\begin{aligned}
\dot{s}(t) &= v(t) \\
\dot{v}(t) &= \frac{u(t)-0.2\,v(t)^2}{m(t)} \\
\dot{m}(t) &= -0.01\,u(t)^2
\end{aligned}$$

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

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

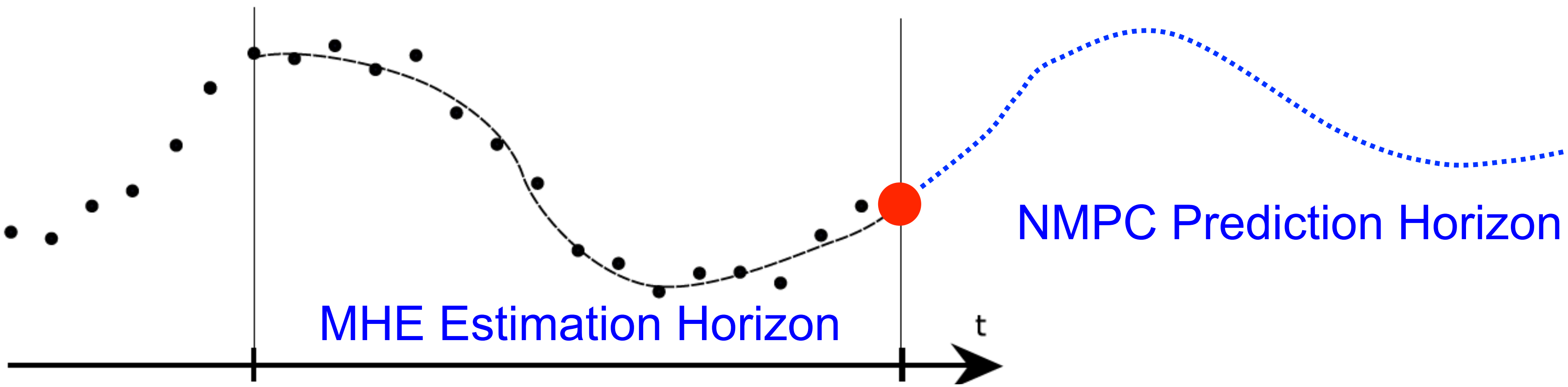
```
DifferentialState                    s,v,m;
Control                                  u;
Parameter                                T;
DifferentialEquation        f( 0.0, T );

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(  0.0 <= v <=  1.7   );
ocp.subjectTo( -1.1 <= u <=  1.1   );
ocp.subjectTo(  5.0 <= T <= 15.0   );

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

ACADO Results Plot (after few milliseconds)



NMPC Practice: Estimation AND Optimization

- Moving Horizon Estimation (MHE): Get State by Least Squares Optimization
- Nonlinear Model Predictive Control (NMPC): Solve Optimal Control Problem

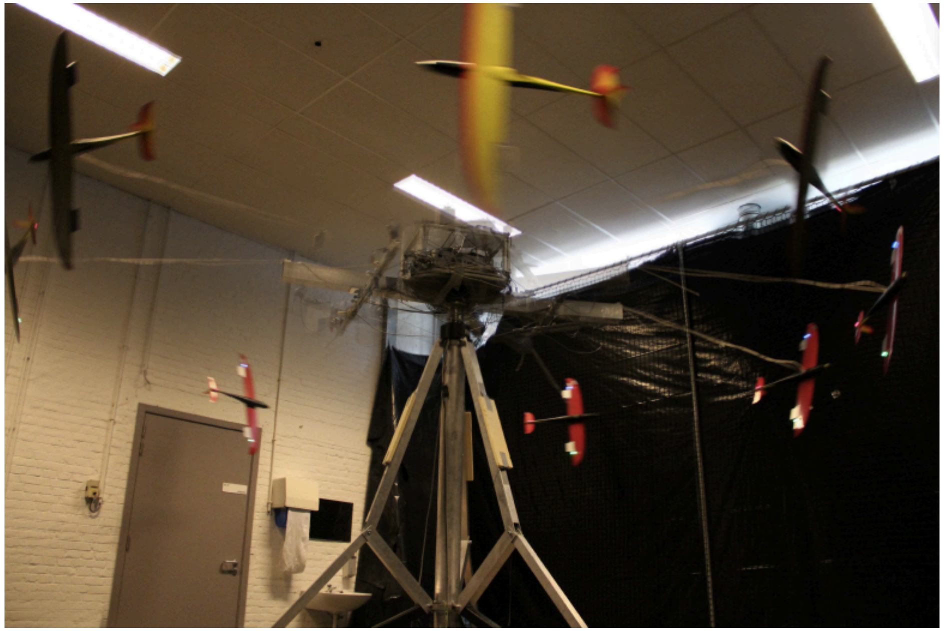


Gauss-Newton in ACADO:

ocp.minimizeMayerTerm()→ ocp.minimizeLSQ();

ACADO Code Generation for Tethered Airplanes

- 22 states, nonlinear, unstable
- 2 controls
- 1 s horizons in past / future

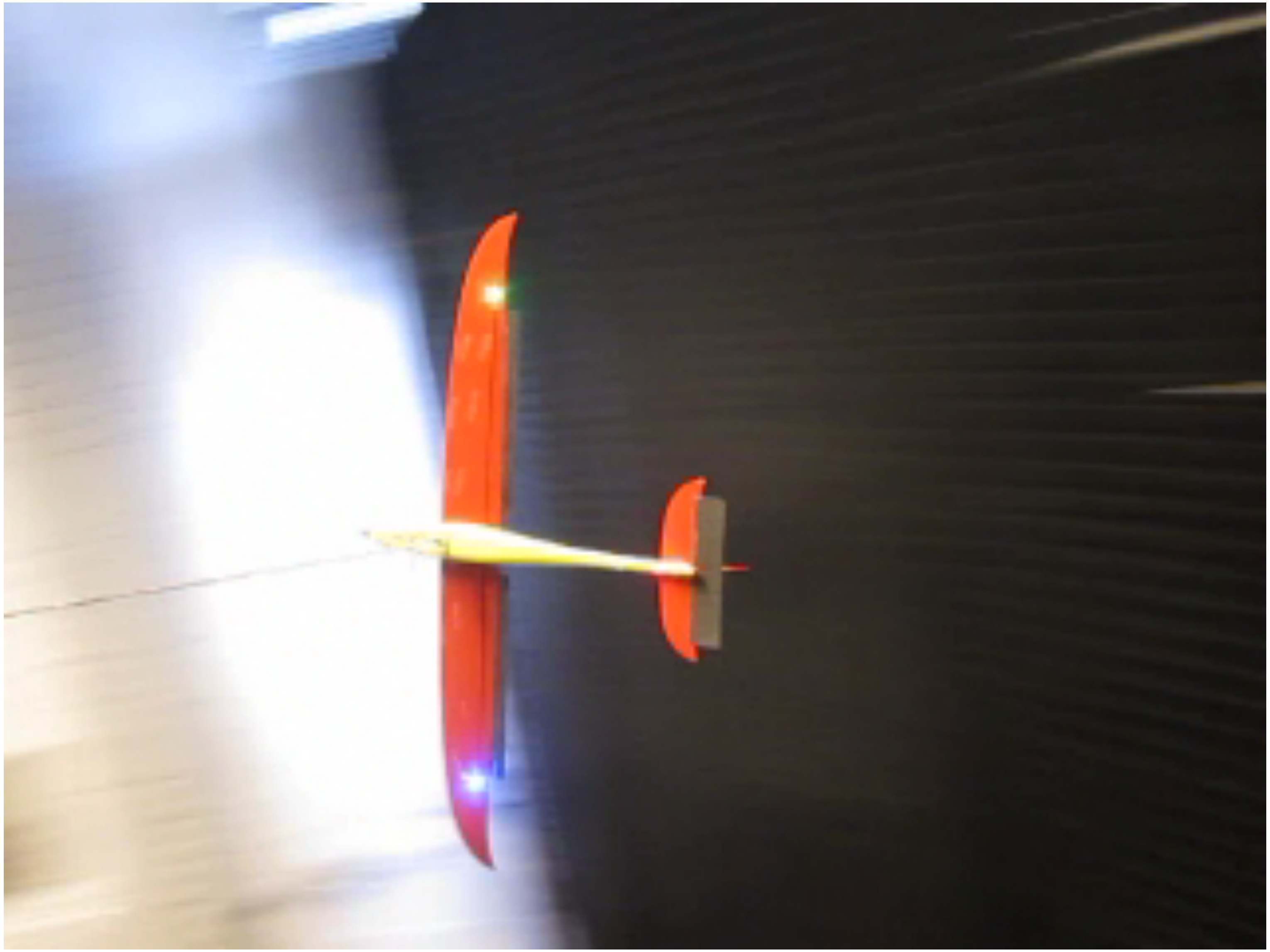


4 ms execution time for one optimization problem (on i7 2.5 GHz)

[Note: NMPC today 100 000x faster than 1997]

ERC HIGHWIND
SIMULATION, OPTIMIZATION & CONTROL OF
HIGH-ALTITUDE WIND POWER GENERATORS

MHE+NMPC Experiments (Aug 22, 2012)



Overview

- Optimization in Engineering Center OPTEC
- State of the Art in Optimal Control Algorithms (ACADO)
- **CasADi: A Framework to WRITE Optimal Control Algorithms**

Optimal Control Problem (OCP) Solvers

Two implementation approaches

- Write/use a general-purpose OCP solver
  - Examples: MUSCOD-II, ACADO Toolkit, DyOS, DIRCOL
  - + Easy to set up for the average user
  - + Can be very efficient for medium size problems
  - − Many OCPs cannot be formulated
- Write special-purpose OCP solvers
  - OCP→NLP using algebraic modelling language
  - + Full control of NLP formulation, easier to *extend*
  - − So far only for collocation methods
- Both approaches taken at OPTEC using two in-house software tools
  - ACADO Toolkit: A general-purpose OCP solver for NMPC
  - CasADi: A *framework for writing OCP solvers*

Computer Algebra System for Algorithmic Differentiation

CasADi

What is CasADi?

A framework for C++, Python and Octave for quick, yet efficient, implementation of algorithms for numeric optimization

In particular

Facilitates OCP→NLP transcription for collocation methods *and* shooting methods (e.g. single-shooting method in 30 lines of code)

Permissive open-source license (LGPL)

www.casadi.org

CasADi

Main components of CasADi

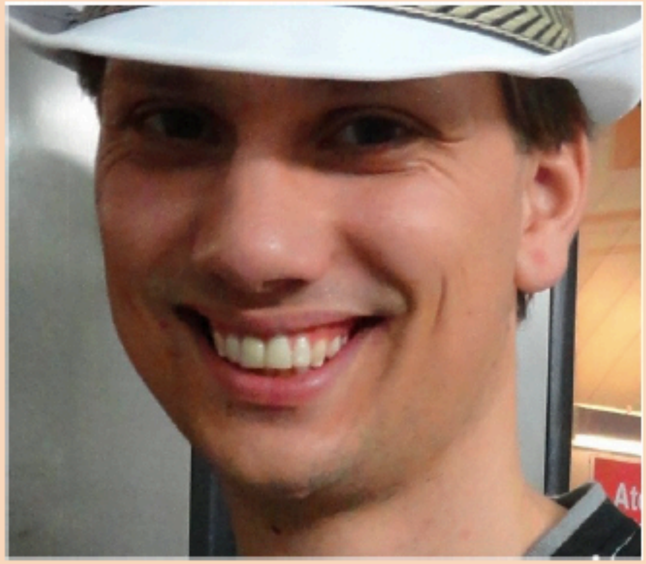
- A symbolic framework with state-of-the-art algorithmic differentiation (all eight flavours of AD)
- Interfaces to other tools; NLP solvers, ODE/DAE integrators, ...
- In-house tools; NLP solvers, ODE/DAE integrators, ...
- Framework for import and symbolic reformulation of OCPs from Modelica

Implementation

- Written in self-contained C++ code
- Full-featured front-ends to Python and Octave using SWIG

CasADi

Main developers



Joel Andersson

Joris Gillis

CasADi

An illustrating example

Drive a Van der Pol oscillator to the origin with minimal control effort:

$$\begin{aligned} \underset{v,p,u}{\text{minimize:}} \quad & \int_0^{t_f} u(t)^2 \, dt \\ \text{subject to:} \quad & \dot{x}(t) = \begin{bmatrix} \dot{v} \\ \dot{p} \end{bmatrix} = \begin{bmatrix} (1-p^2)\,v - p + u \\ v \end{bmatrix}, \quad t \in [0, t_f] \\ & v(0) = 0, \quad p(0) = 1, \\ & v(t_f) = 0, \quad p(t_f) = 0 \\ & -0.75 \le u(t) \le 1.0, \quad t \in [0, t_f] \end{aligned}$$

Solve with a direct-single shooting method.

CasADi

Step 1: Formulate symbolic expression ODE in CasADi

- The ODE:

$$\begin{bmatrix} \dot{v} \\ \dot{p} \end{bmatrix} = \begin{bmatrix} (1 - p^2)\, v - p + u \\ v \end{bmatrix},$$

- Can be formulated in CasADi–Python:

```
# Declare variables
u  = ssym("u")
v  = ssym("v")
p  = ssym("p")
```

```
# ODE right hand side
vdot = (1 - p*p)*v - p + u
pdot = v
```

- Syntax $\approx$ Matlab Symbolic Toolbox
- ODE can also be imported from **Modelica**

CasADi

Step 2: Create ODE function

- These expressions define the ODE rhs *function* $f : \mathbb{R}^2 \times \mathbb{R} \to \mathbb{R}^2$:

```
f = SXFunction( \
     daeIn(   x = vertcat([v,p]), p = u), \
     daeOut(ode = vertcat([vdot,pdot])))
```

- Creating a function means *topologically sorting* the expression graph
- Function can be evaluated:
  - In the CasADi interpretor: numerically *or* symbolically
  - By generating and compiling C-code
  - Through just-in-time compilation (using *LLVM* framework)
- Derivatives in CasADi are calculated by *automatic differentiation*

CasADi

Step 3: Formulate discrete time dynamics

- Assume a piecewise constant control with 20 intervals and let t_f be 10 s.

```
nk = 20    # Control discretization (uniform)
th = 10.0  # Length of the time horizon
```

- Get the discrete time dynamics by allocating an ODE integrator instance, e.g. using CasADi's interface to Sundials:

```
f_d = CVodesIntegrator(f)
f_d.setOption("tf",th/nk) # Interval length
f_d.init()
```

- Integrators in CasADi are differentiable functions in CasADi and can be differentiated an arbitrary number of times
- Derivatives calculated through *forward/adjoint sensitivity analysis*

CasADi

Step 4: Formulate NLP

The integrator allows us to form an expression for the state at the final time:

```
U = msym("U",nk)      # Controls for each interval
X0 = [0,1] # The initial state
# Build a graph of integrator calls
X  = X0
for k in range(nk):
  X,_,_,_ = I.call([X,U[k]])
```

this defines NLP objective functions and constraints:

```
# Objective function: ||U||^2
F = MXFunction([U],[mul(U.T,U)])

# Terminal constraints: x=[0,0]
G = MXFunction([U],[X])
```

CasADi

Step 5: Solve NLP

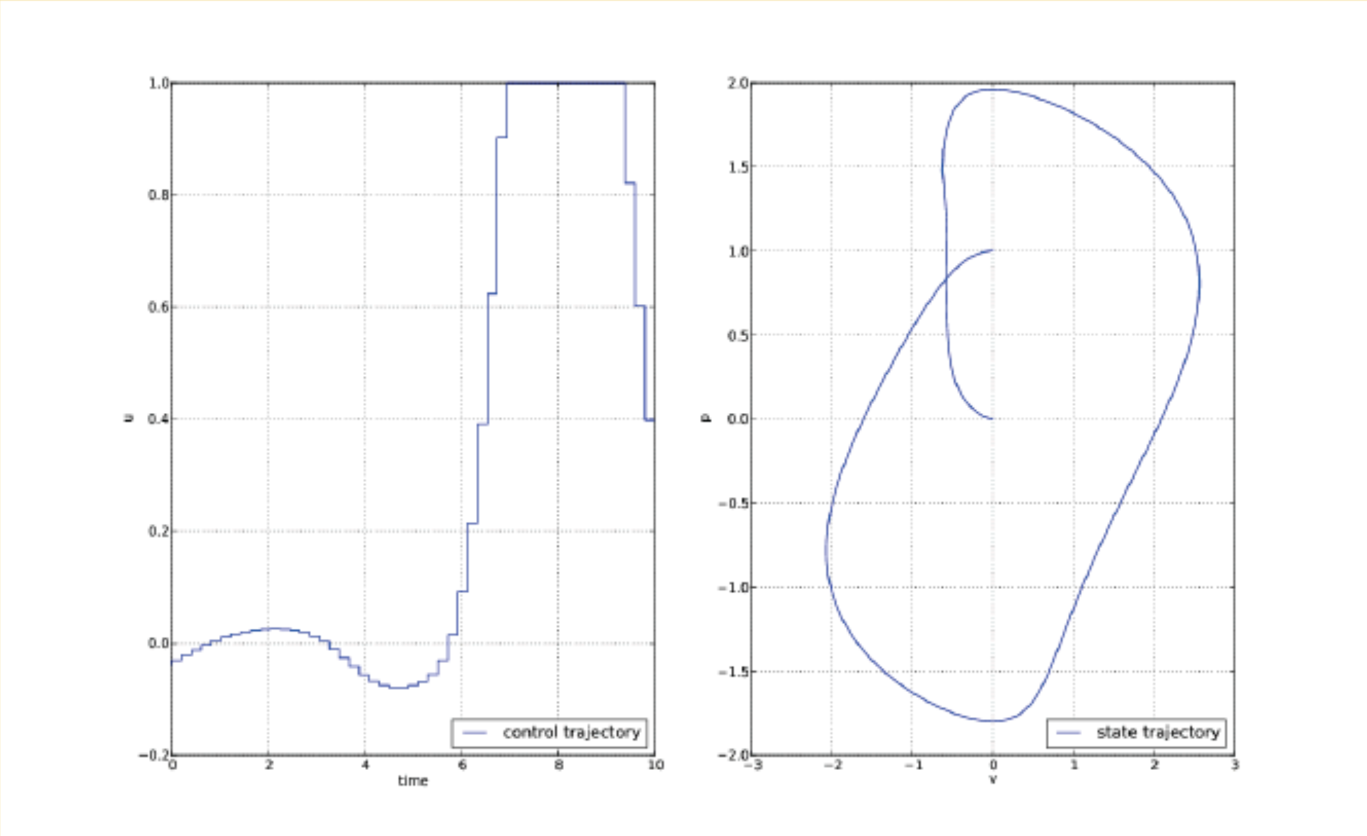
Solve NLP by using one of the interfaced NLP solvers, e.g. IPOPT:

```
import numpy        # Standard linear algebra routines

# Allocate an NLP solver
solver = IpoptSolver(F,G)
solver.init()

# Set bounds and initial guess
solver.setInput(-0.75*numpy.ones(nk), NLP_LBX)
solver.setInput(1.0*numpy.ones(nk), NLP_UBX)
solver.setInput(numpy.zeros(nk),NLP_X_INIT)
solver.setInput(numpy.zeros(2),NLP_LBG)
solver.setInput(numpy.zeros(2),NLP_UBG)

# Solve the problem
solver.solve()
```

CasADi

Step 6: Visualize solution

Use standard Python packages visualizing the solution:



CasADi Users

Other OCP methods successfully implemented using CasADi

- Direct collocation (J. Andersson, J. Åkesson & F. Magnusson, M. Zanon & S. Gross, J. Steinberg, J. Gillis . . .)
- Direct multiple-shooting (J. Andersson, K. Gevelen, J. Frasch)
- Distributed multiple-shooting (A. Kozma & C. Savorgnan)
- Pseudospectral optimization (C. Andersson)

Benchmarking CasADi vs AMPL Solver Library

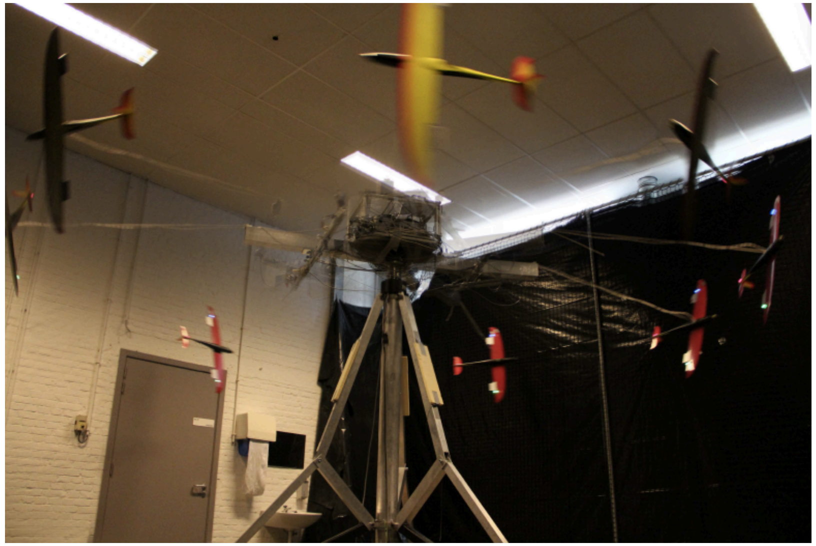
Problem	Dimensions		Time ASL [s]		Time CasADi [s]		Diff.
	#var	#con	Total	AD	Total	AD	
gpp	250	498	0.492	0.272	0.500	0.264	-3 %
reading1	10001	5000	0.712	0.408	0.306	0.104	-76 %
porous2	4900	4900	1.916	0.188	1.736	0.036	-81 %
orthrgds	10003	5000	0.949	0.568	0.512	0.164	-71 %
clnlbeam	1499	1000	0.776	0.184	0.784	0.184	0 %
svanberg	5000	5000	2.492	0.520	2.300	0.272	-48 %
orthregd	10003	5000	0.332	0.208	0.160	0.060	-71 %
trainh	20000	10002	3.932	1.984	2.804	0.896	-55 %
orthrgdm	10003	5000	0.328	0.208	0.156	0.068	-67 %
dtoc2	5994	3996	0.296	0.124	0.224	0.048	-61 %

Benchmarking

- CasADi VM outperformed ASL VM by a factor 2 on average
- Most of the time spent in linear solver anyway
- Note: ≈5x faster still with C-codegen or just-in-time

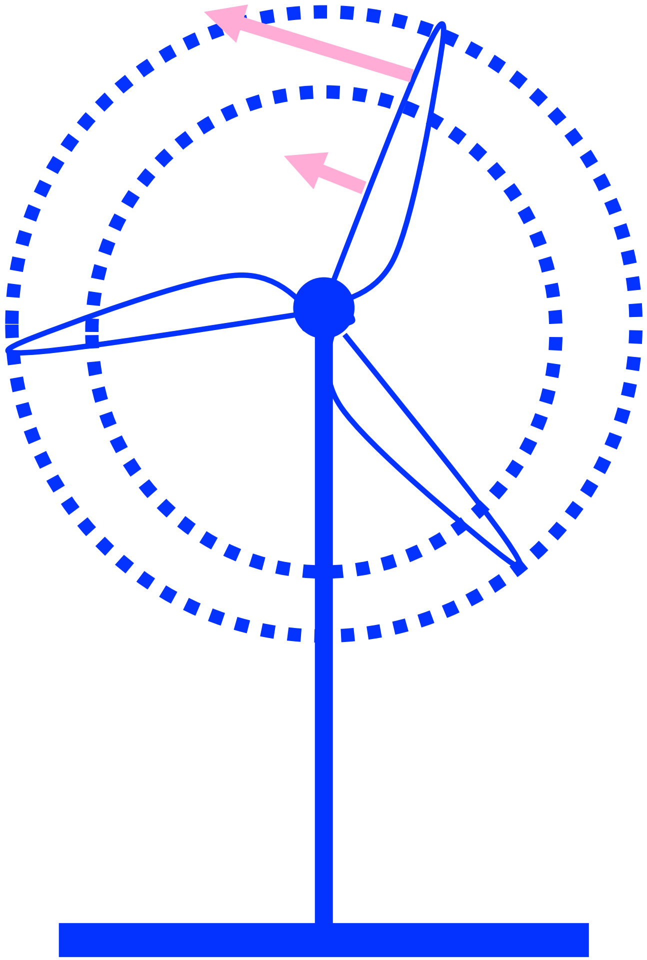
CasADi Usage in Leuven: Complex Plane Orbits

- Within ERC Project HIGHWIND, running from 2011-2016



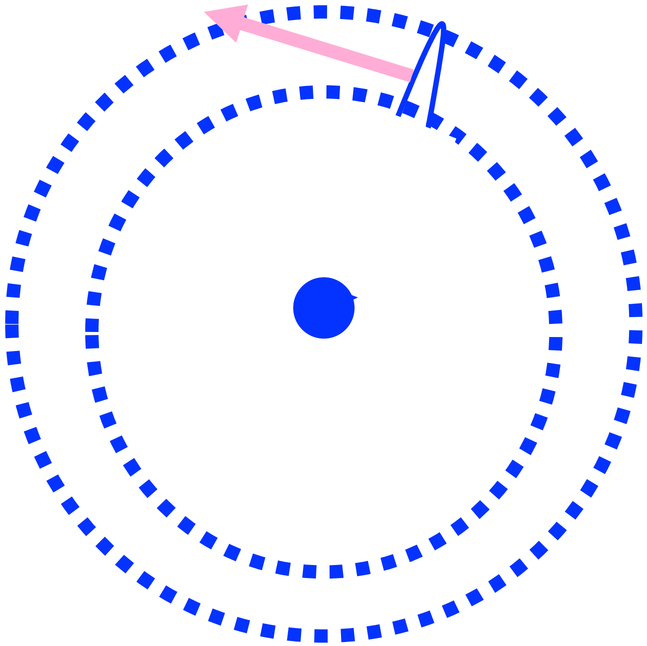
ERC
HIGHWIND
SIMULATION, OPTIMIZATION & CONTROL OF
HIGH-ALTITUDE WIND POWER GENERATORS

What is the Optimal Wind Turbine ?



- Due to high speed, wing tips are ***most efficient*** part of wing
- Best winds are in high altitudes

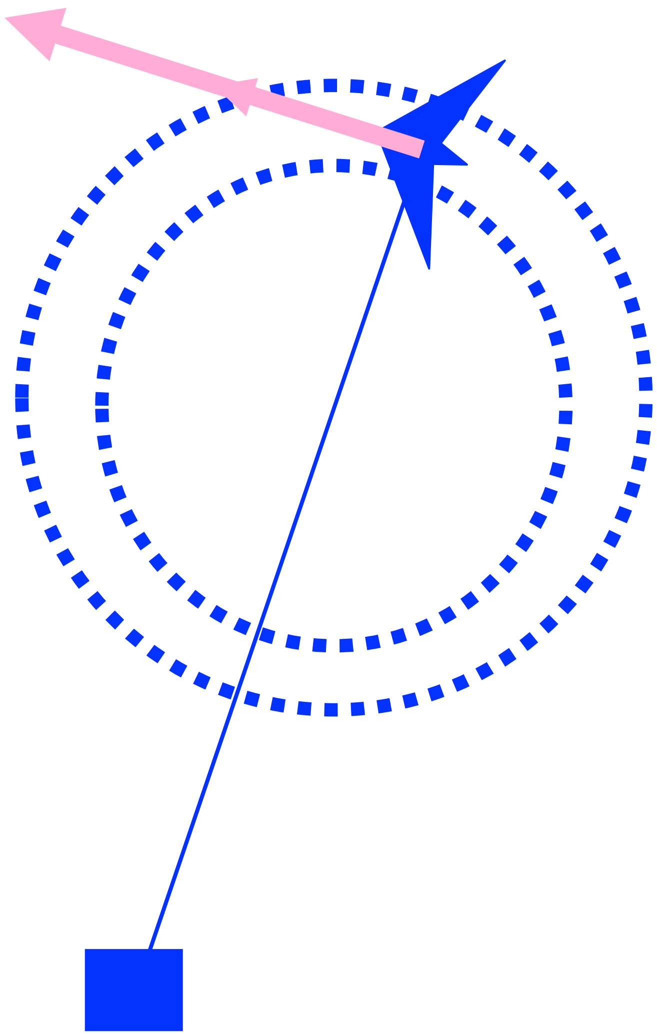
What is the Optimal Wind Turbine ?



- Due to high speed, wing tips are ***most efficient*** part of wing
- Best winds are in high altitudes

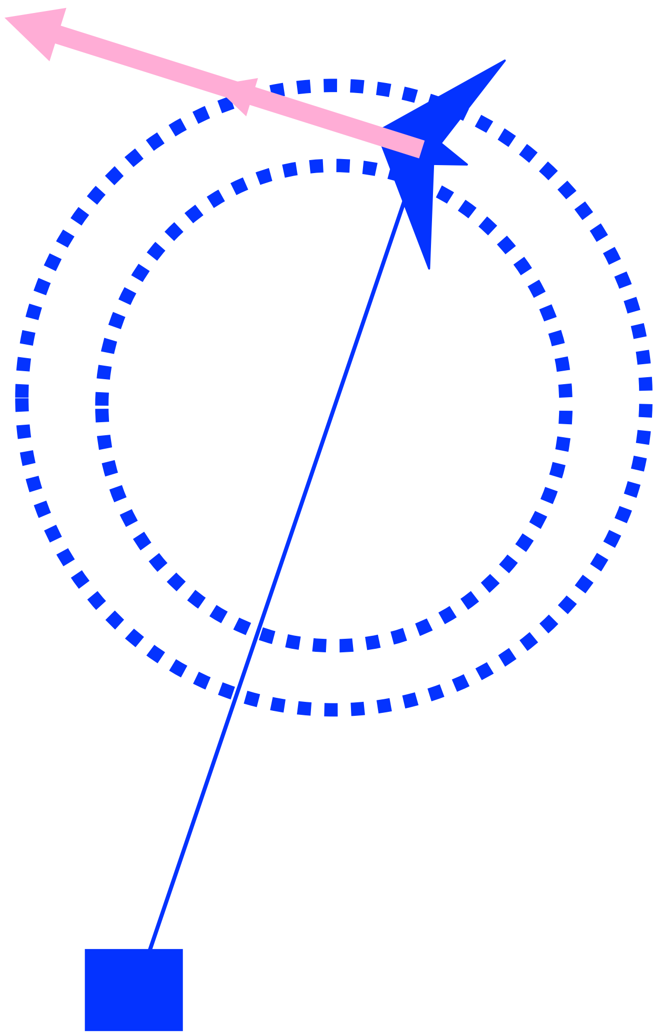
Could we construct a wind turbine with only wing tips and generator?

Crosswind Kite Power



- Fly kite fast in crosswind direction
- Very strong force

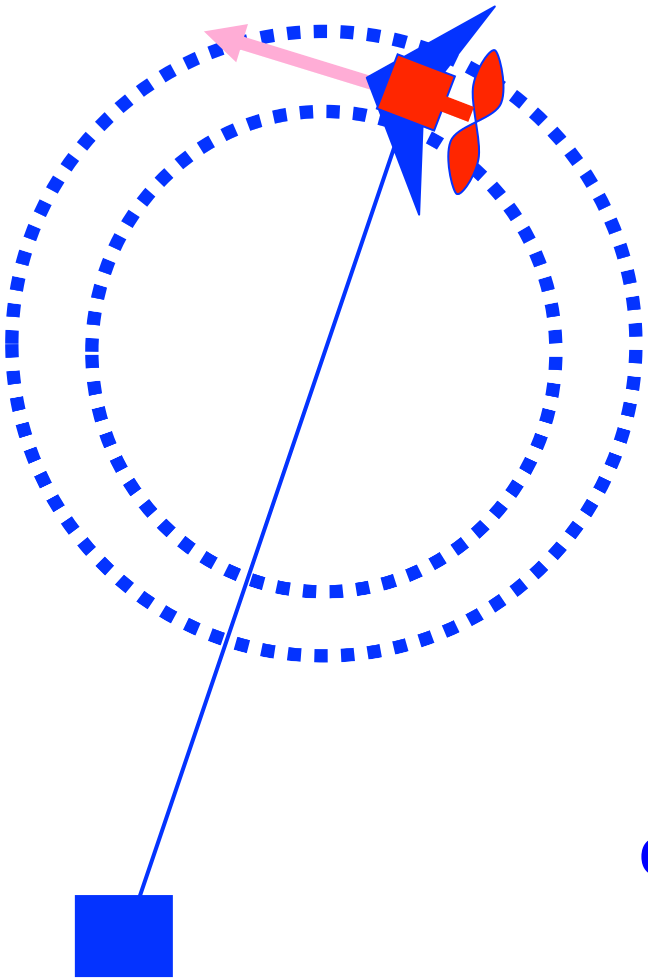
Crosswind Kite Power



- Fly kite fast in crosswind direction
- Very strong force

But where could a generator be driven?

One Variant: On-Board Generator



- attach ***small wind turbines*** to kite
- cable transmits power

Question:

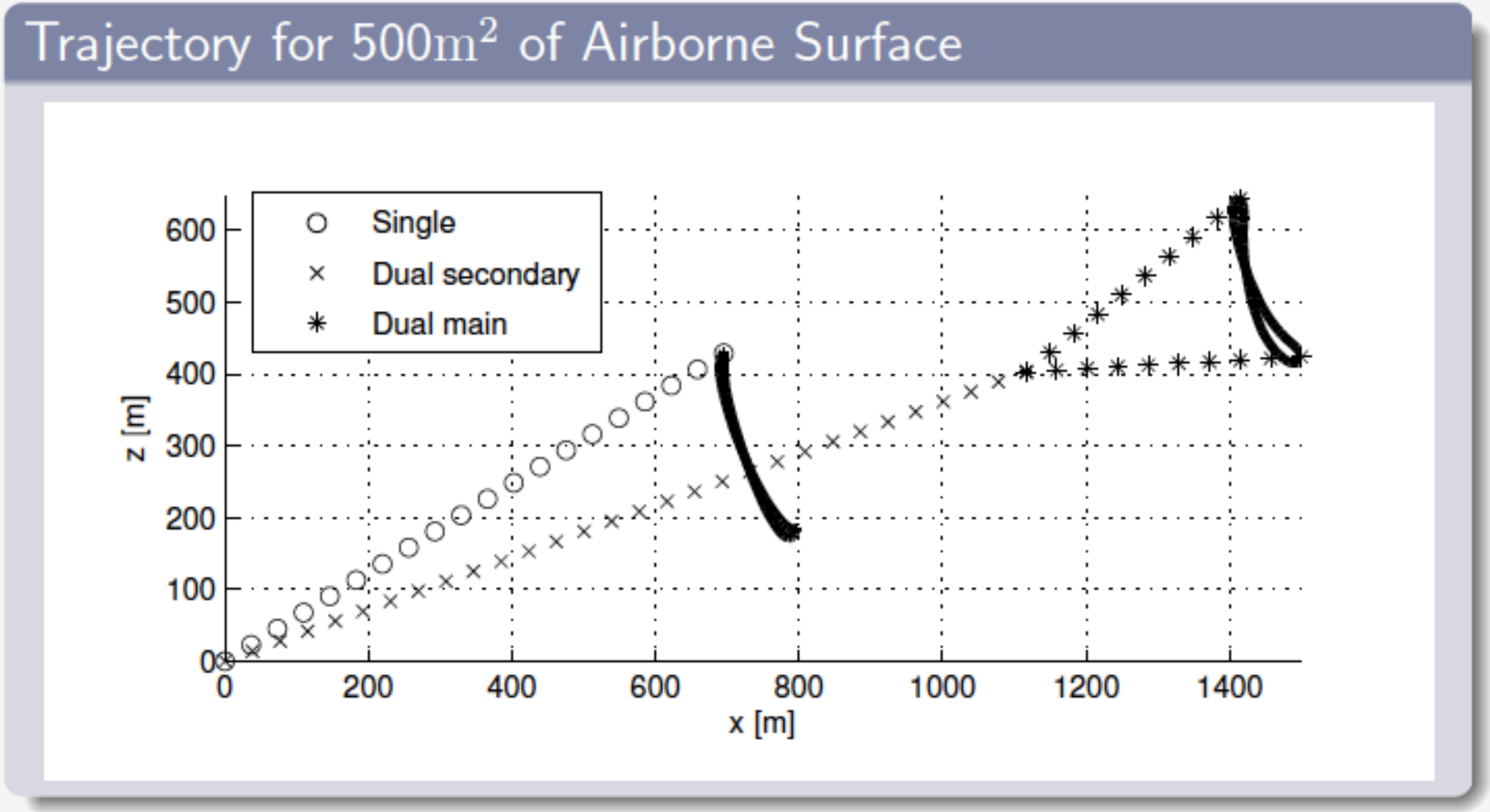
what are the optimal periodic orbits ?

CasADi Usage in Leuven: Complex Plane Orbits

- Complex aerodynamic models
- Periodic boundary conditions

- Connecting two tethers can increase the power output significantly…
- …but leads to even more complex models and optimal control problems

Single vs. Dual Airfoils: Optimal Large System



Complex OCPs solved with CasADi, Collocation, IPOPT, from [Zanon et al., submitted]

Visualization of Single vs. Dual Airfoils

Summary

- Optimal Control Tools now 100000x faster than 1997, and ACADO Code Generation is currently tested in a variety of fast real world applications (cranes, airplanes, vehicles, induction motors, …)

- But non-standard problems need non-standard solvers: CasADi allows the user to easily write competitive state-of-the-art optimal control algorithms specifically designed for one problem class

- CasADi distributed under permissive LGPL license and used by a growing number of people in and outside Leuven (e.g. Jmodelica)

CasADi www.casadi.org

Appendix

CasADi Performance

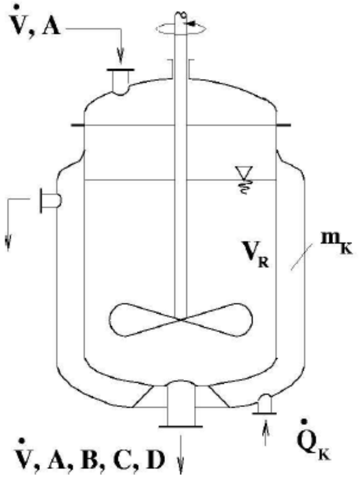
Benchmarking using CUTEr

- 10 NLPs from Bob Vanderbei's AMPL translation of CUTEr
- AMPL used to parse/pre-optimize AMPL models
- Solved using IPOPT 3.10 with MA27 as linear solver in two ways
  - Using AMPL Solver Library's (ASL) interface to IPOPT
  - Using CasADi's .nl import and interface to IPOPT
  - Only virtual machines (VM) for both tools, no codegen

Complete CasADi Code for OCP Solution

```
from casadi import *

nk = 50    # Control discretization
th = 10.0  # End time

# Declare variables
u  = ssym("u")
v  = ssym("v")
p  = ssym("p")
x  = vertcat([v,p])

# ODE right hand side
vdot = (1 - p*p)*v - p + u
pdot = v
xdot = vertcat([vdot,pdot])

# DAE residual function
f = SXFunction(daeIn(x=x,p=u),daeOut(ode=xdot))

# Create an integrator
I = CVodesIntegrator(f)
I.setOption("tf",th/nk) # final time
I.init()

# All controls (use matrix graph)
U = msym("U",nk) # nk-by-1 symbolic variable

# The initial state (x = [0,1])
X0 = [0,1]

# Build a graph of integrator calls
X  = X0
for k in range(nk):
  X,_,_,_ = I.call([X,U[k]])

# Objective function: ||U||^2
F = MXFunction([U],[mul(U.T,U)])

# Terminal constraints: x=[0,0]
G = MXFunction([U],[X])

# Allocate an NLP solver
solver = IpoptSolver(F,G)
solver.init()

# Set bounds and initial guess
solver.setInput(-0.75*ones(nk), NLP_LBX)
solver.setInput(1.0*ones(nk), NLP_UBX)
solver.setInput(zeros(nk),NLP_X_INIT)
solver.setInput(zeros(2),NLP_LBG)
solver.setInput(zeros(2),NLP_UBG)

# Solve the problem
solver.evaluate()
```

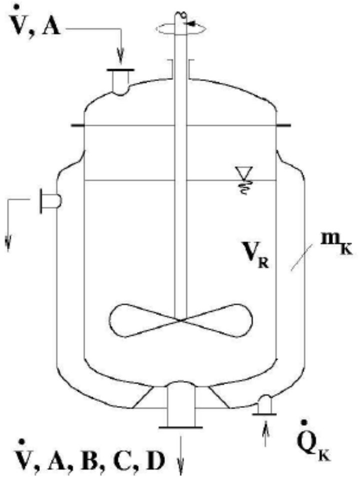
ACADO Code Generation for Benchmark CSTR



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

CSTR Benchmark by [Klatt, Engell, Kremling, Allgower 1995]

ACADO Code Generation for Benchmark CSTR



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

CSTR Benchmark by [Klatt, Engell, Kremling, Allgower 1995]

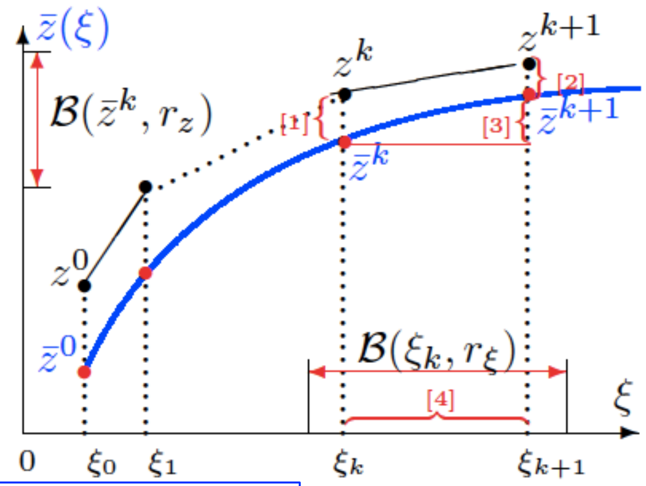
CPU Times for ACADO:

	CPU time (μs)	%
Integration & sensitivities	121	30
Condensing	98	24
QP solution (with qpOASES)[a]	180	44
Remaining operations	<5	<2
A complete real-time iteration	404	100

From [Houska, Ferreau, D., *Automatica,* 2011]

NMPC now 100 000x faster than 1997 (200x by CPU, 500x by algorithms)

(SCP Real-Time Iteration Contraction Estimate)



Contraction estimate for primal dual errors [1]:

$$\boxed{\begin{aligned}\left\|z^{k+1} - \bar{z}^{k+1}\right\| \leq \left(\alpha + c_1 \left\|z^k - \bar{z}^k\right\|\right) \left\|z^k - \bar{z}^k\right\| \\ + \left(c_2 + c_3 \left\|\xi_{k+1} - \xi_k\right\|\right) \left\|\xi_{k+1} - \xi_k\right\|\end{aligned}}$$

Depends only on nonlinearity of equalities, independent of active set changes!

[1] Tran Dinh, Savorgnan, Diehl: Adjoint-based predictor-corrector SCP for parametric nonlinear optimization. SIAM J. Opt. 2013 (in print)

Optimal Control Family Tree