Embedded Optimization for Control and Signal Processing

Moritz Diehl
Optimization in Engineering Center (OPTEC) &
Electrical Engineering Department (ESAT)
K.U. Leuven
Belgium

KATHOLIEKE UNIVERSITEIT LEUVEN

Linkoping, June 16, 2011

OPTEC - Optimization in Engineering Center

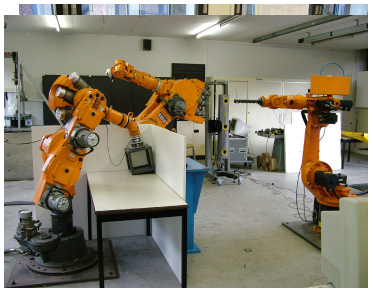
Center of Excellence of K.U. Leuven, from 2005-2010, 2010-2017

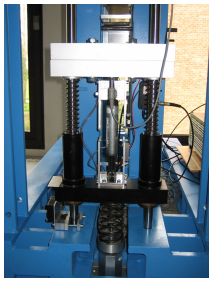
About 20 professors, 10 postdocs, and 40 PhD students involved in OPTEC research

Scientists in 5 divisions:

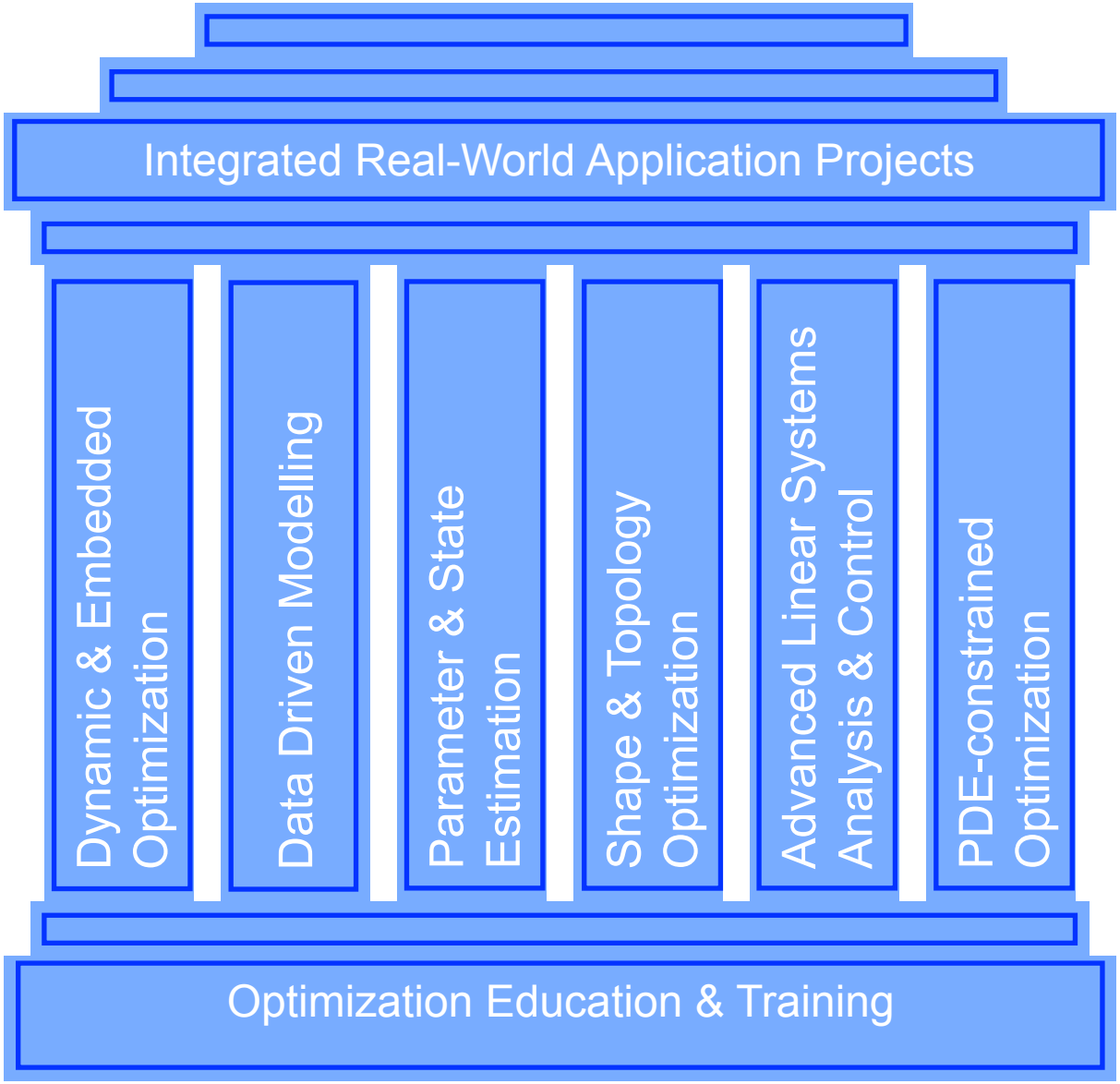
- Electrical Engineering
- Mechanical Engineering
- Chemical Engineering
- Computer Science
- Civil Engineering

Many real world applications at OPTEC...

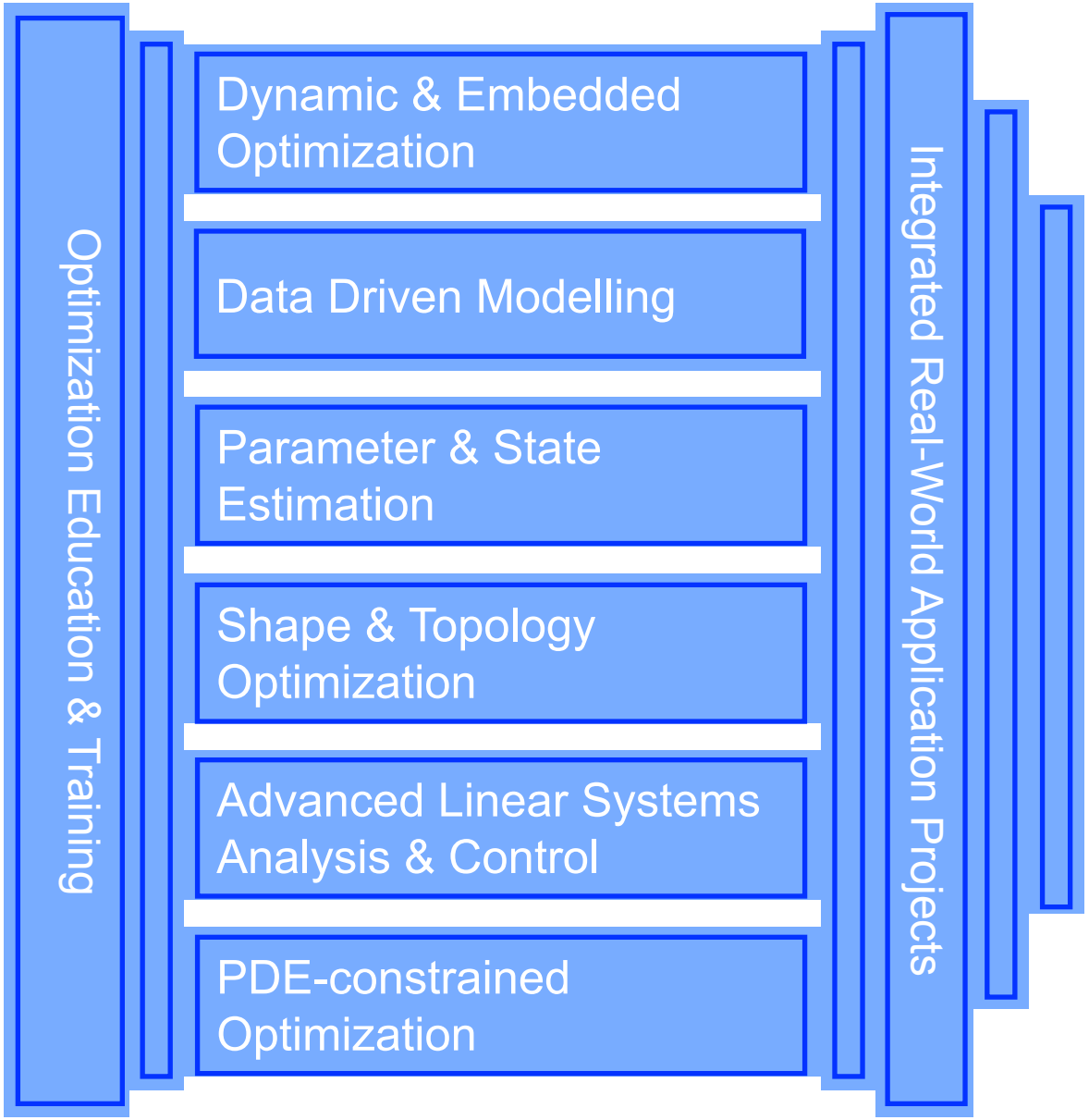




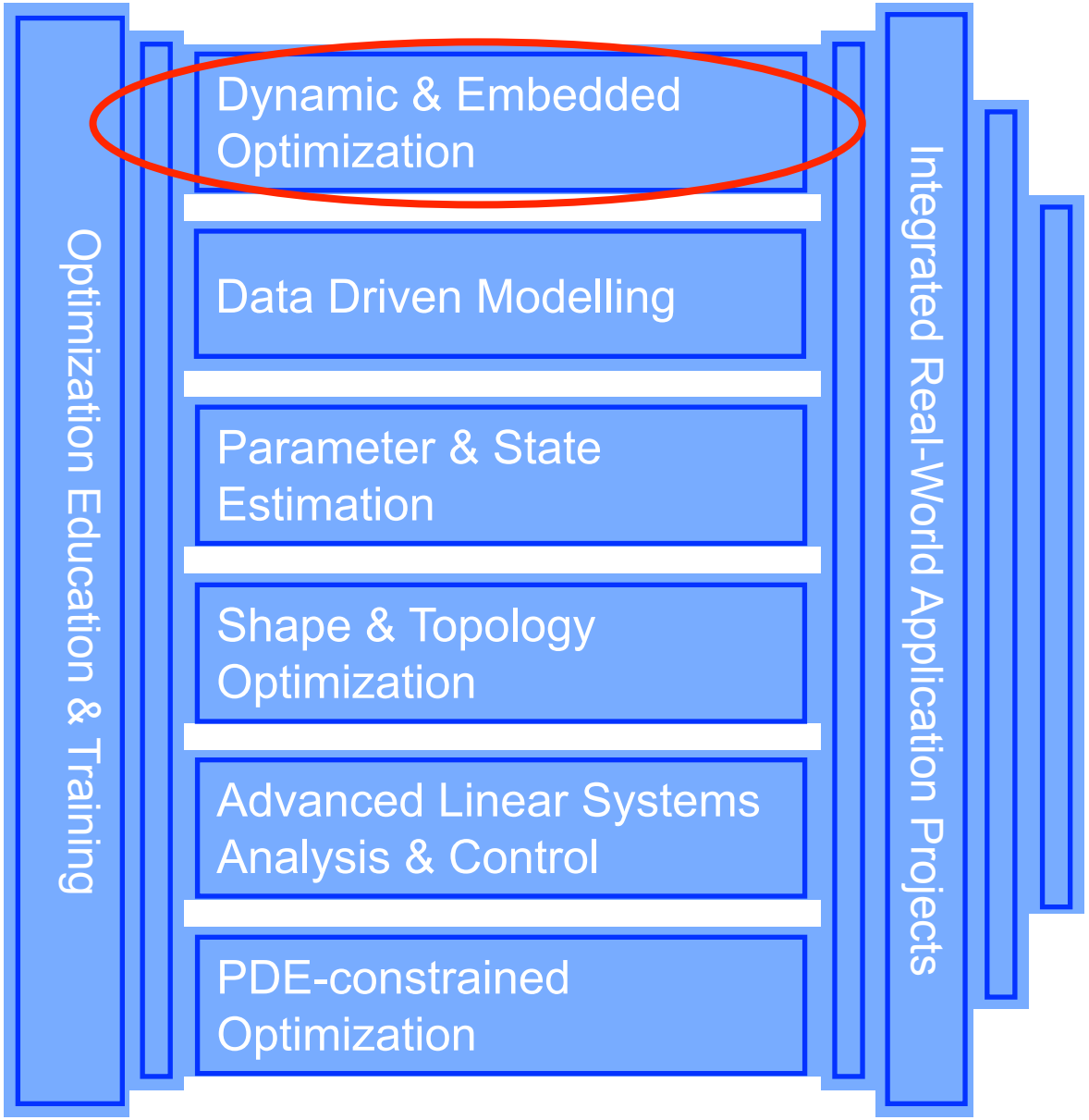
OPTEC: 70 people in six methodological working groups



OPTEC: 70 people in six methodological working groups



OPTEC: 70 people in six methodological working groups



Overview

- Idea of Embedded Optimization
- Perception-based Clipping of Audio Signals using Convex Optimization
- Time Optimal MPC of Machine Tools
- Optimal Control of Tethered Airplanes for Wind Power Generation

Classical Filters

We are interested in maps from one space of sequences:

$$\ldots, y_{i-1}, y_i, y_{i+1}, \ldots$$

into another:

$$\ldots, u_{i-1}, u_i, u_{i+1}, \ldots$$

Classical Filters

We are interested in maps from one space of sequences:

$$\ldots, y_{i-1}, y_i, y_{i+1}, \ldots$$

into another:

$$\ldots, u_{i-1}, u_i, u_{i+1}, \ldots$$

Important special case:

Linear *Time Invariant, Finite Impulse Response Filters:*

$$u_k = \sum_{i=0}^{N} a_i y_{k-i}$$

“output = linear combination of past inputs”

Classical Filters

We are interested in maps from one space of sequences:

$$\ldots, y_{i-1}, y_i, y_{i+1}, \ldots$$

into another:

$$\ldots, u_{i-1}, u_i, u_{i+1}, \ldots$$

Important special case:

Linear *Time Invariant, Finite Impulse Response Filters:*

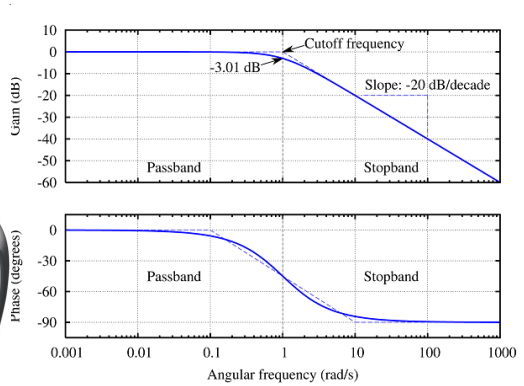
$$u_k = \sum_{i=0}^{N} a_i y_{k-i}$$

“output = linear combination of past inputs”

Linear Filters are Everywhere…

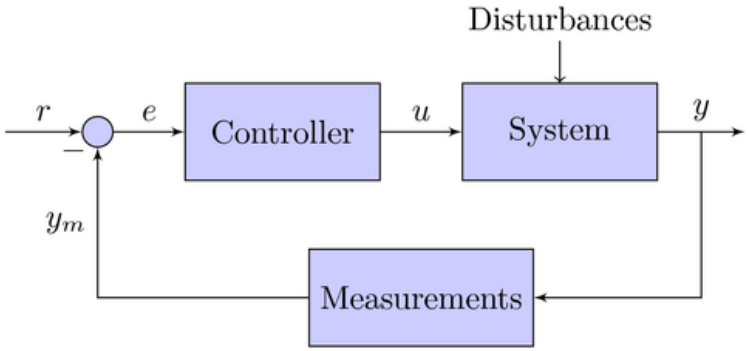
In audio processing:

- Dolby
- active noise cancelling
- echo and other sound effects



In control:

- Kalman filter
- PID
- LQR



…**but they often need lots online tuning to deal with constraint saturations, gain changes etc.**

Alternative: Embedded Optimization

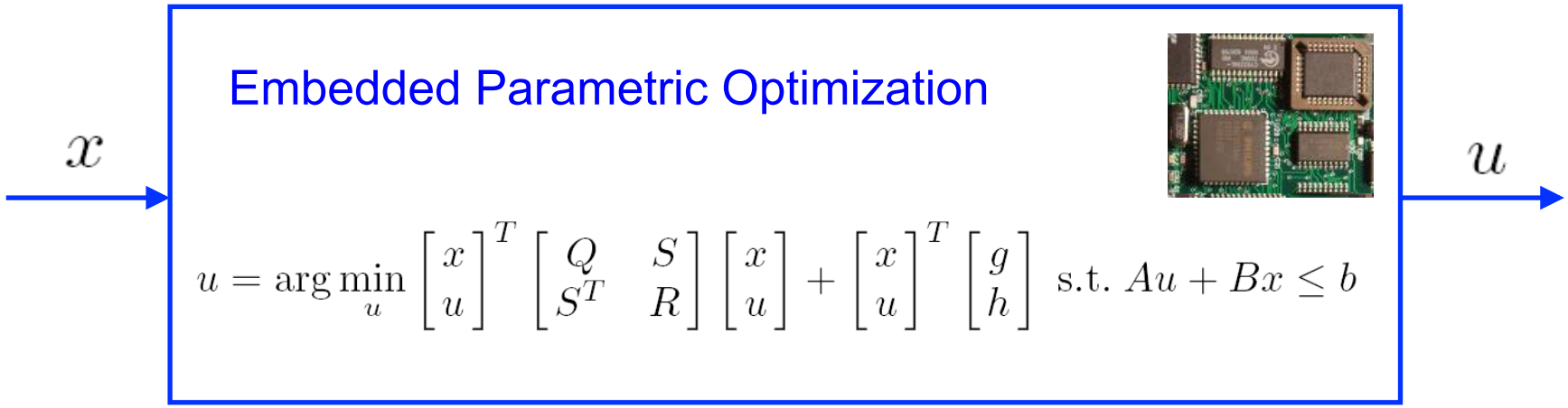
- Idea: obtain **NON-LINEAR** map by solving repeatedly a parametric optimization problem:

$$u = \arg\min_{u} g(u, x) \text{ s.t. } (u, x) \in \Gamma$$

- Example: parametric quadratic programming:

$$u = \arg\min_{u} \begin{bmatrix} x \\ u \end{bmatrix}^T \begin{bmatrix} Q & S \\ S^T & R \end{bmatrix} \begin{bmatrix} x \\ u \end{bmatrix} + \begin{bmatrix} x \\ u \end{bmatrix}^T \begin{bmatrix} g \\ h \end{bmatrix} \text{ s.t. } Au + Bx \leq b$$

Embedded Optimization = CPU Intensive, Nonlinear Map



Very powerful concept!

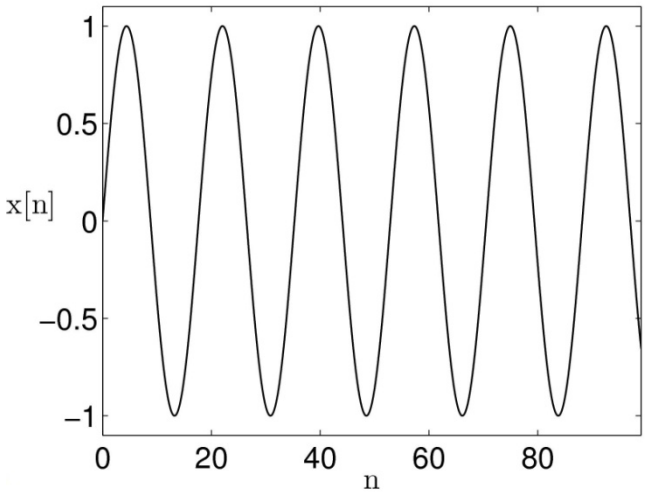
We can prove [Baes, D., Necoara 2008]:
"**Every continuous map** can be generated as solution map of a convex parametric program"

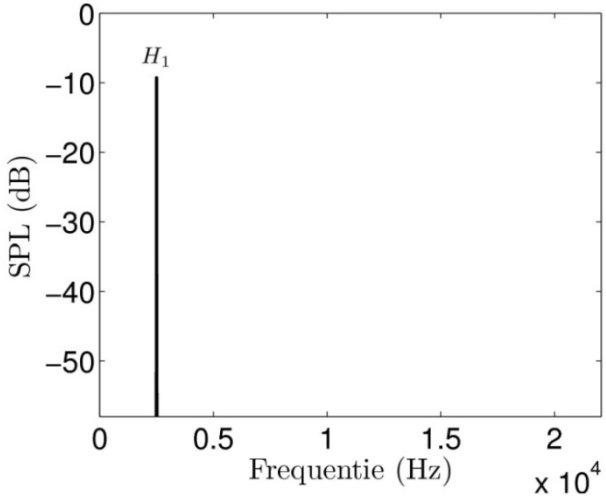
Real-time perception-based clipping of audio signals using convex optimization

Bruno Defraene, Toon van Waterschoot, Hans Joachim Ferreau, Marc Moonen & M.D.

Clipping Problem Statement

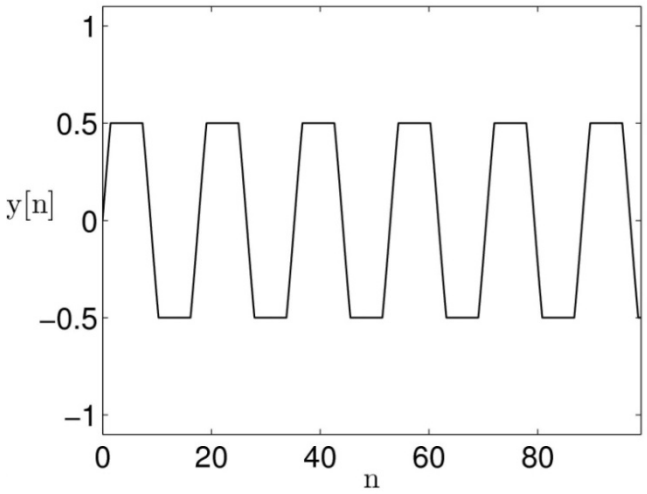
- Clipping = limit amplitude of digital audio signal to range [L,U]
- Real time audio applications (mobile phones, hearing aids…)
- Hard clipping has a large negative effect on perceptual sound quality (distortion)



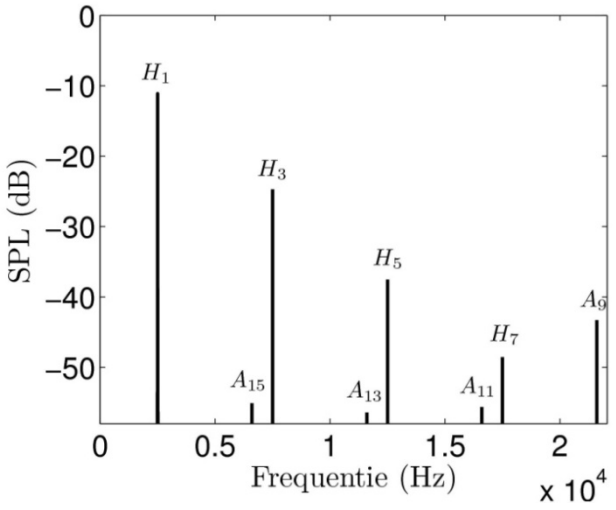


Clipping Problem Statement

- Clipping = limit amplitude of digital audio signal to range [L,U]
- Real time audio applications (mobile phones, hearing aids…)
- Hard clipping has a large negative effect on perceptual sound quality (distortion)

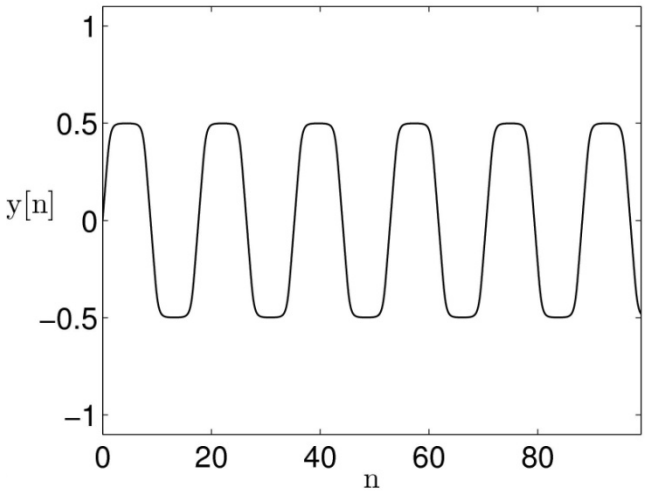


HARD CLIPPING

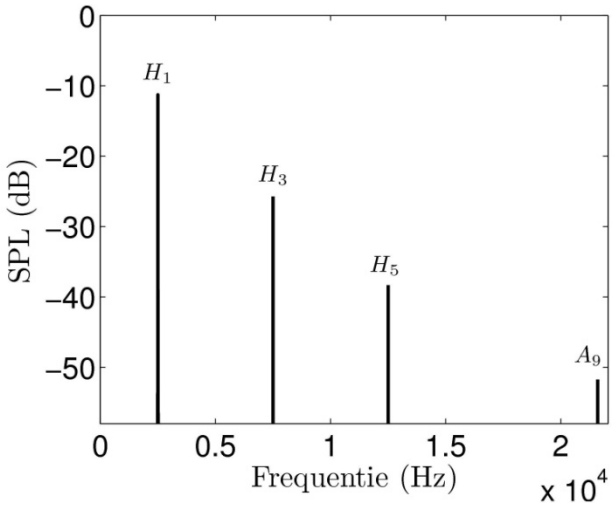


Clipping Problem Statement

- Clipping = limit amplitude of digital audio signal to range [L,U]
- Real time audio applications (mobile phones, hearing aids…)
- Hard clipping has a large negative effect on perceptual sound quality (distortion)
- Soft clipping does not help much

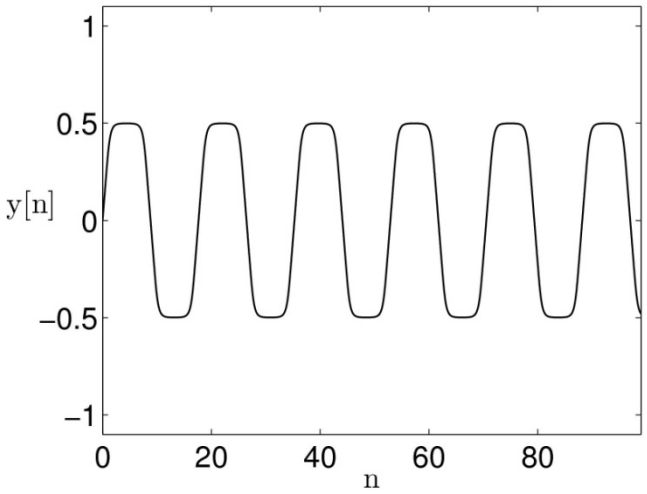


SOFT CLIPPING

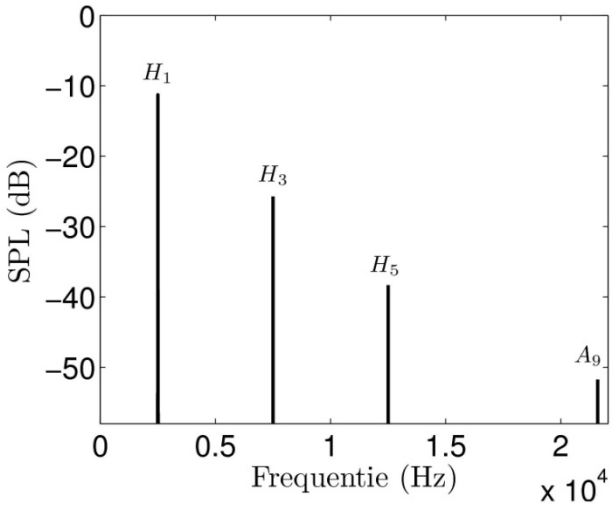


Clipping Problem Statement

- Clipping = limit amplitude of digital audio signal to range [L,U]
- Real time audio applications (mobile phones, hearing aids…)
- Hard clipping has a large negative effect on perceptual sound quality (distortion)
- Soft clipping does not help much



SOFT CLIPPING



What is the optimal way of clipping?

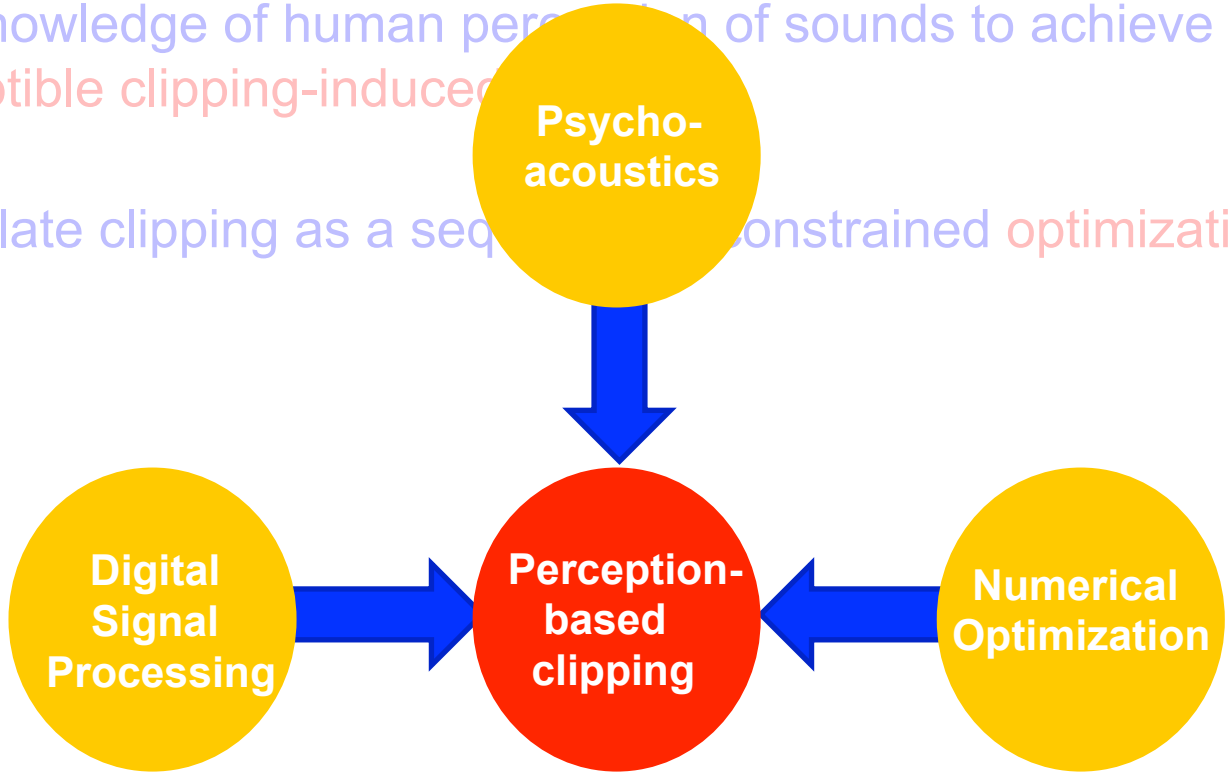
Perception-based clipping - a novel approach

- Use knowledge of human perception of sounds to achieve minimal perceptible clipping-induced distortion
- Formulate clipping as a sequence of constrained optimization problems

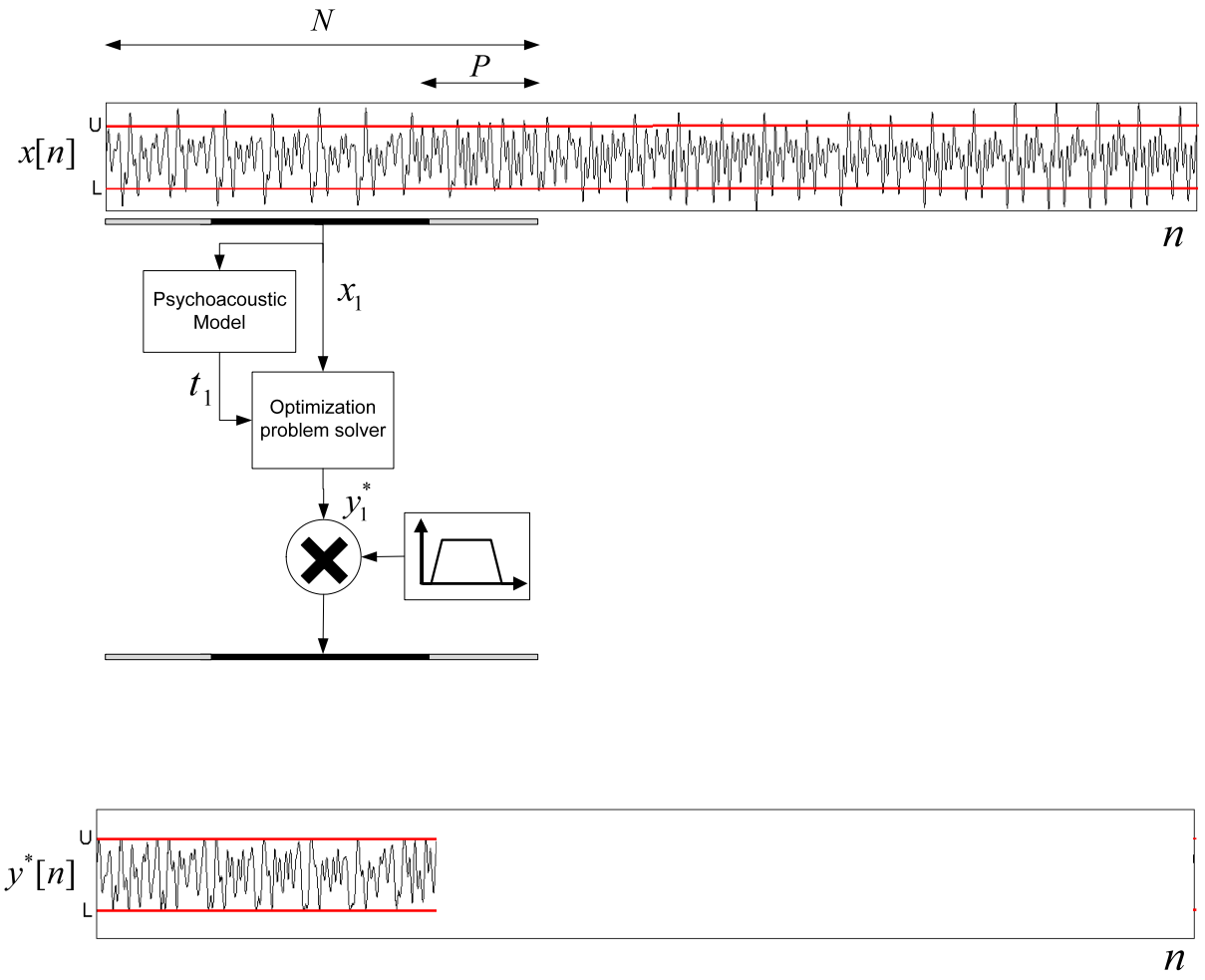
Perception-based clipping - a novel approach

Multidisciplinary approach

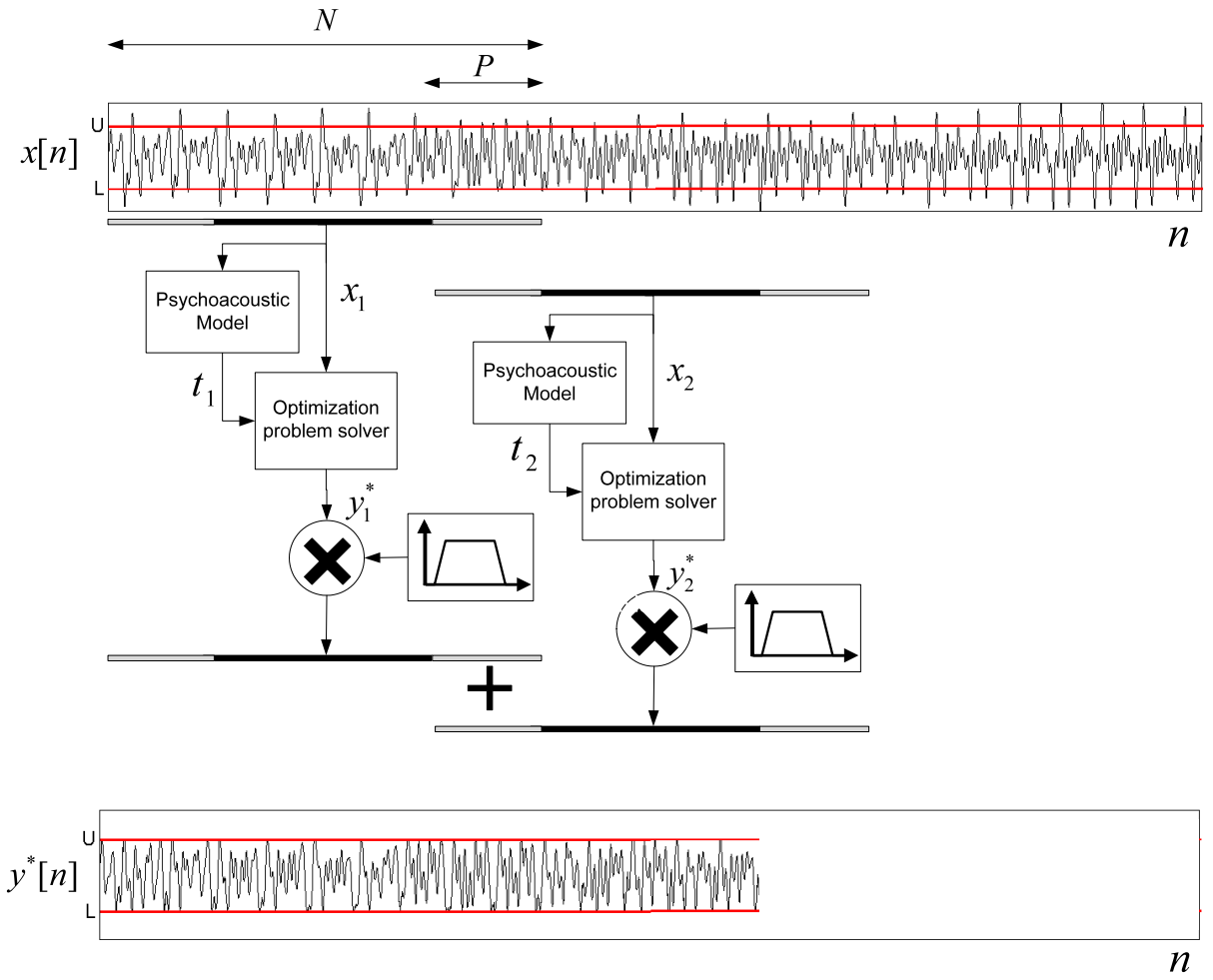
- Use knowledge of human perception of sounds to achieve minimal perceptible clipping-induced
- Formulate clipping as a seq constrained optimization problems



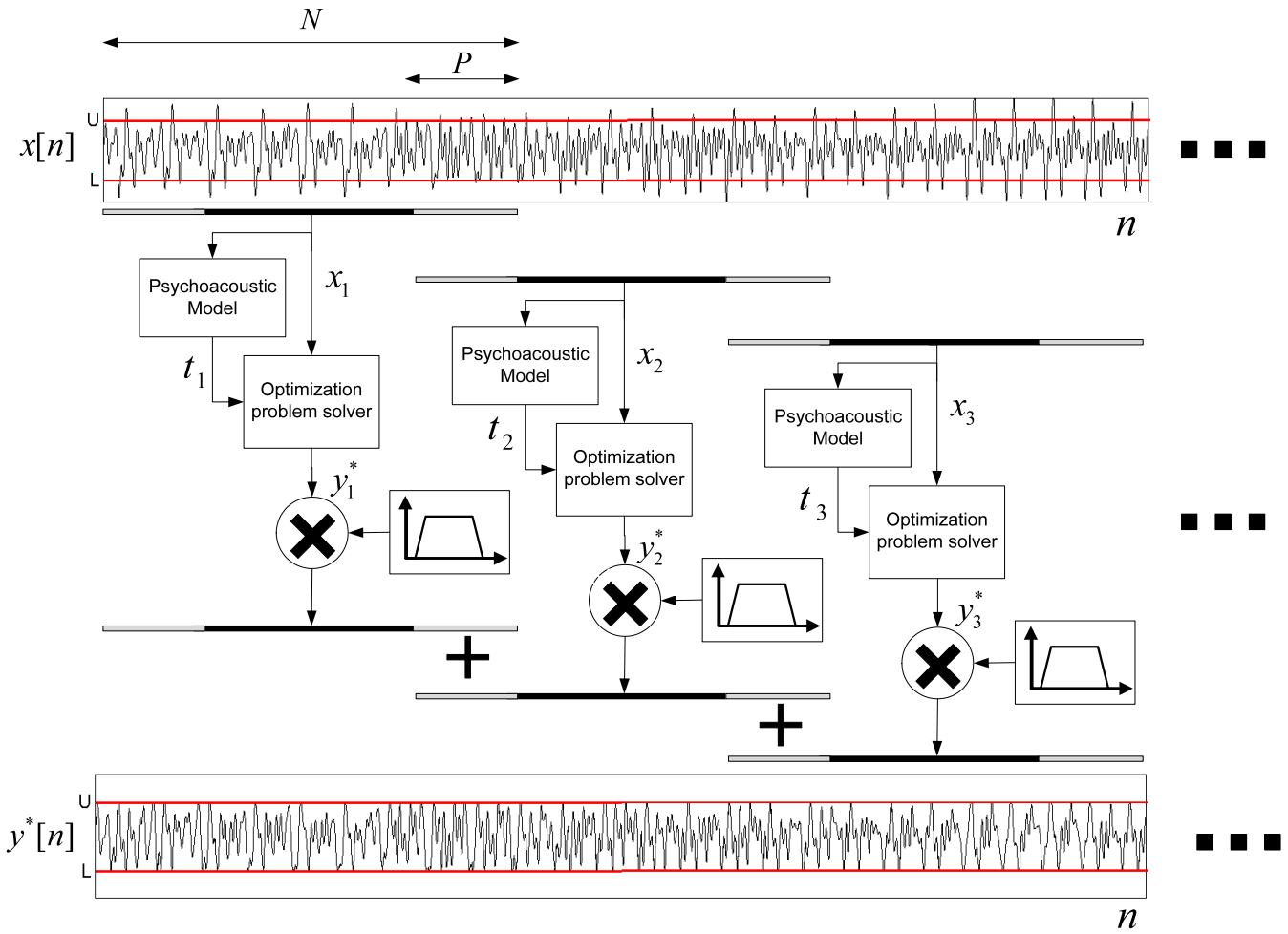
Perception-based clipping algorithm



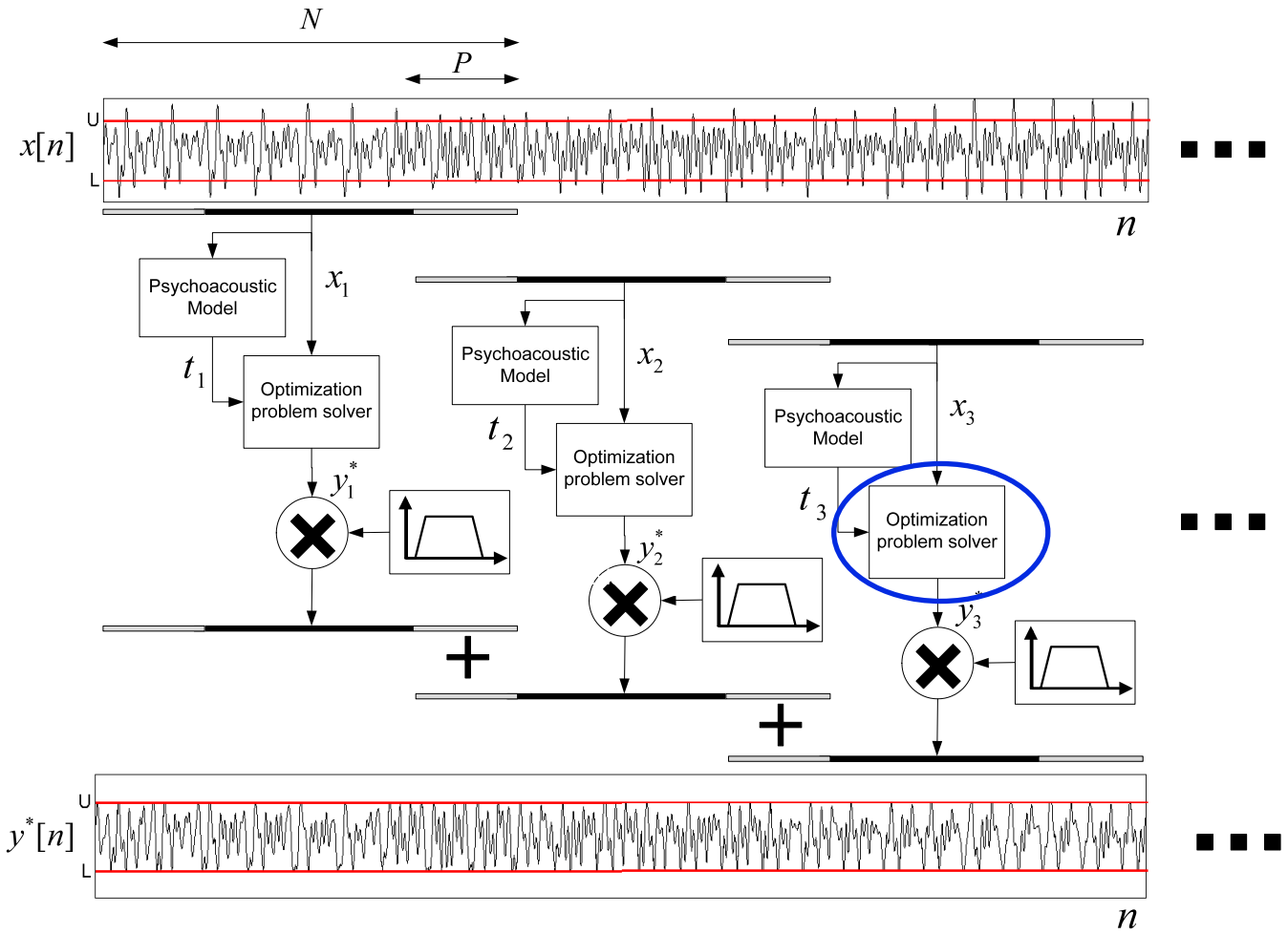
Perception-based clipping algorithm



Perception-based clipping algorithm

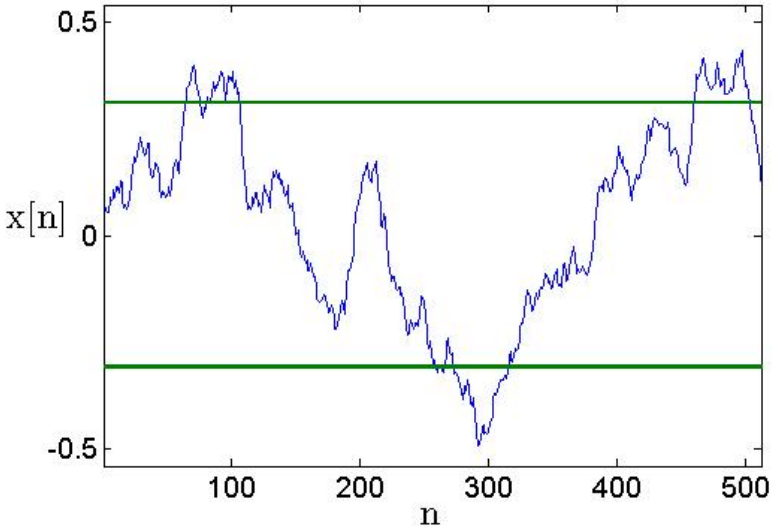


Perception-based clipping algorithm



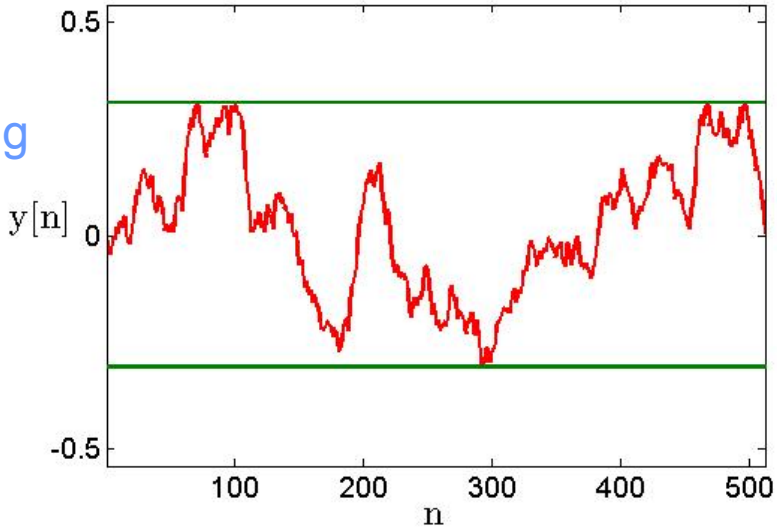
Embedded optimization problems = QPs

Input audio frame



Clipping

Output audio frame



Minimize perceptual difference in frequency space subject to clipping constraint:

$$\min_{y \in \mathbb{R}^N} \frac{1}{2}(y-x)^T D^H W D(y-x) \quad \text{s.t.} \quad L \leq y \leq U$$

dense **Fourier Matrix** D and diagonal weighting matrix $W = \begin{bmatrix} w_1 & & \\ & w_2 & \\ & & \ddots \end{bmatrix}$

w_i = inverse of *perceptual masking threshold* of current audio frame (not today's topic)

Medium Scale Quadratic Programs

$$\min_{y \in \mathbb{R}^N} \frac{1}{2}(y-x)^T D^H W D(y-x) \quad \text{s.t.} \quad L \leq y \leq U$$

QP with 512 variables, 1024 constraints, dense Hessian.

QP solution time using a general purpose QP solver : ~ 500 ms

[Intel CPU 2.8 GHz]

Real-time objective to achieve delay free CD-Quality: 8.7 ms

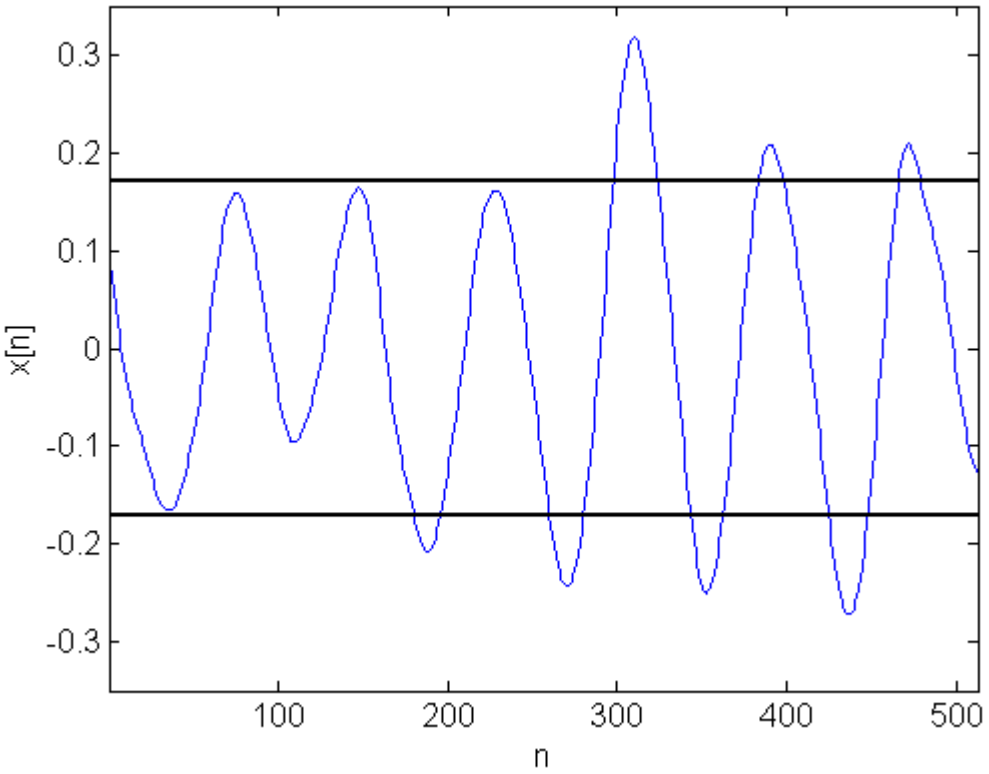
Three Tailored QP Solution Methods

- Method 1: External Active Set Strategy with Small Dual QPs
- Method 2: Projected Gradient
- Method 3: Nesterov's Optimal Gradient Scheme

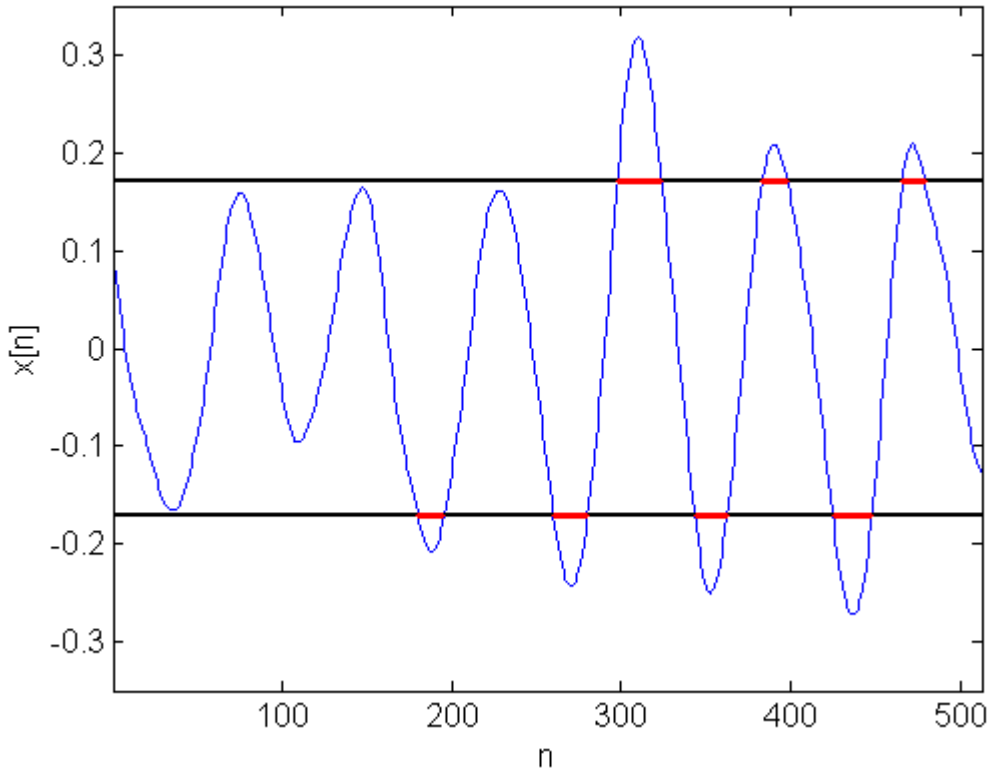
Three Tailored QP Solution Methods

- **Method 1: External Active Set Strategy with Small Dual QPs**
- Method 2: Projected Gradient
- Method 3: Nesterov's Optimal Gradient Scheme

External Active Set Strategy: Original Signal

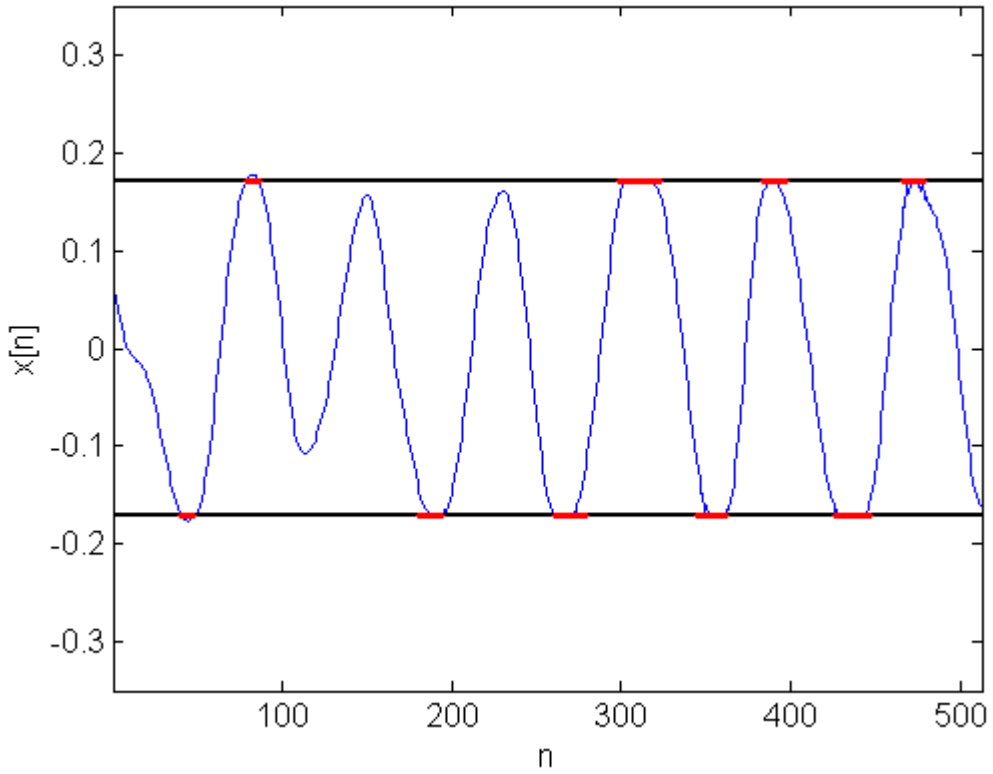


External Active Set Strategy: Original Signal

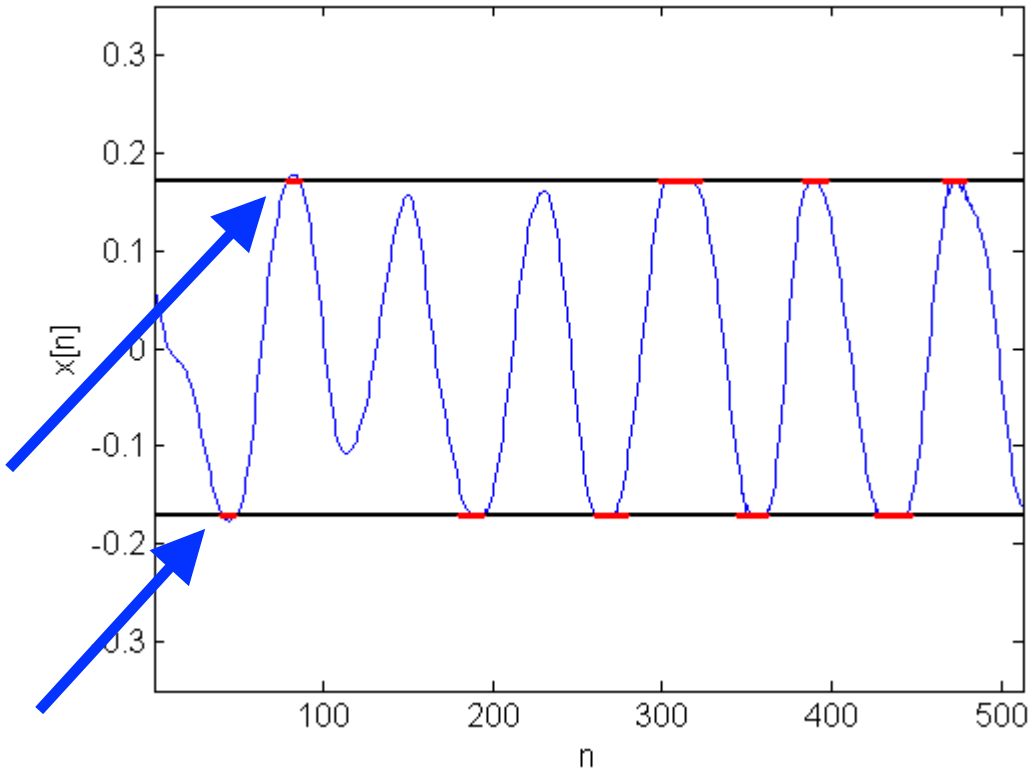


violated constraint indices = nonzero multipliers in small scale dual QP

External Active Set Strategy: 1st Iteration Result

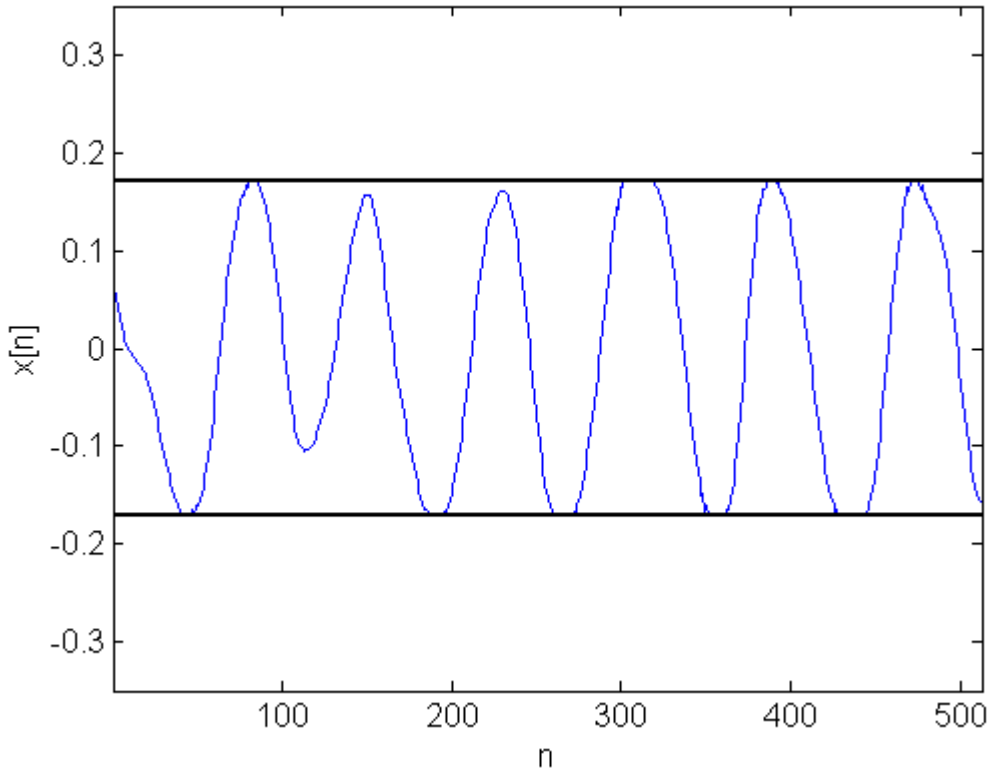


External Active Set Strategy: 1st Iteration Result

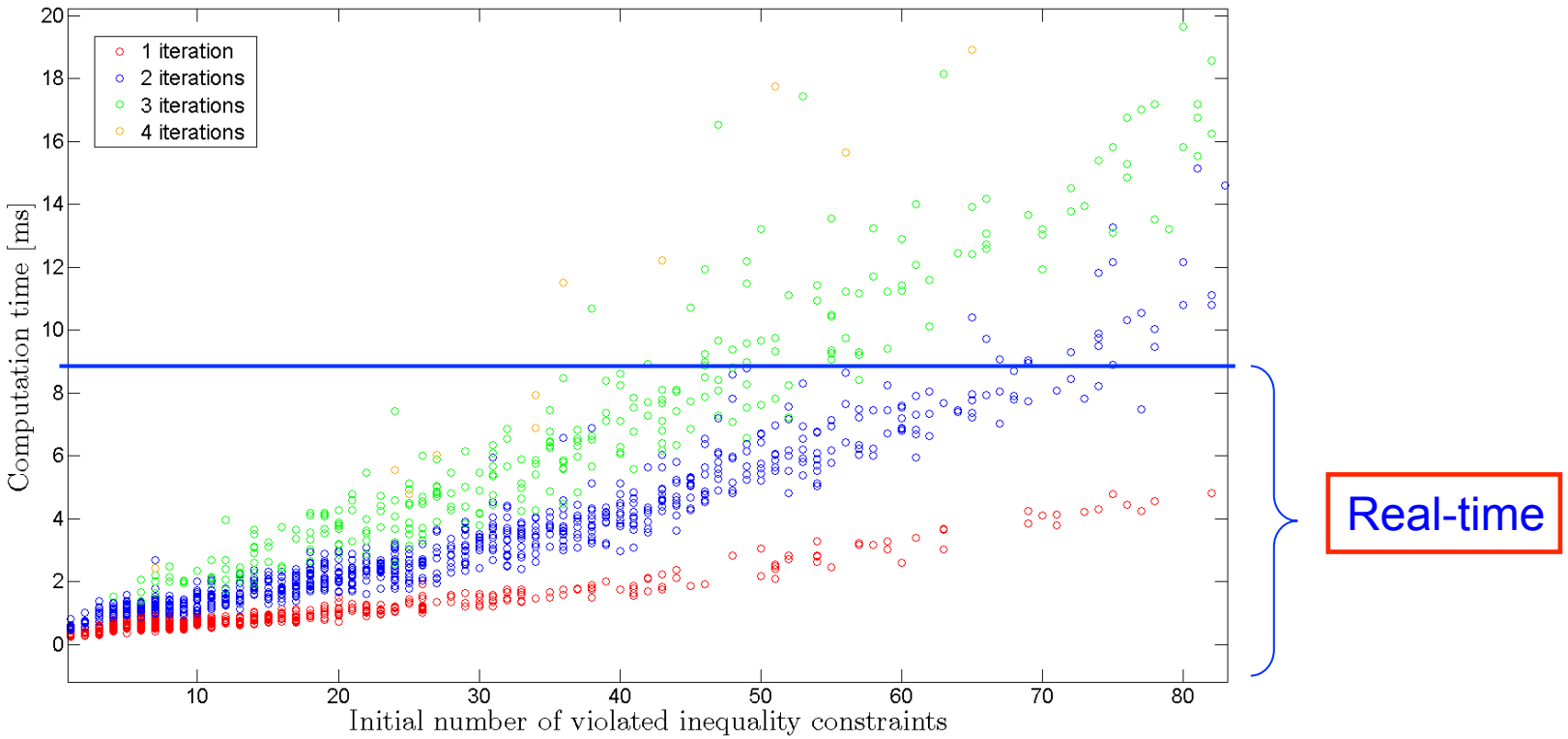


add the very few newly violated constraint indices to dual QP, solve again

External Active Set Strategy: 2nd Iteration = Solution



CPU Time Tests with External Active Set Strategy



[Intel CPU ~2.8 GHz]

- 40 x faster than standard QP solver, but not always real-time feasible

Three Tailored QP Solution Methods

- Method 1: External Active Set Strategy with Small Dual QPs
- **Method 2: Projected Gradient**
- Method 3: Nesterov's Optimal Gradient Scheme

Method 2: Projected Gradient

Algorithm 1 Projected gradient descent

Input $x \in \mathbb{R}^n$, $y^0 = rand$, Lipschitz constant L

Output $y^* \in \mathbb{R}^n$

1: $k = 0$

2: **while** stopping criterion is not met **do**

3: $\tilde{y}^{k+1} = y^k - \frac{1}{L}\nabla f(y^k)$ **Gradient step**

4: $y^{k+1} = \Pi_Q(\tilde{y}^{k+1})$

5: $k = k + 1$

6: **end while**

Method 2: Projected Gradient

Algorithm 1 Projected gradient descent

Input $x \in \mathbb{R}^n$, $y^0 = rand$, Lipschitz constant L

Output $y^* \in \mathbb{R}^n$

1: $k = 0$

2: **while** stopping criterion is not met **do**

3: $\tilde{y}^{k+1} = y^k - \frac{1}{L}\nabla f(y^k)$

4: $y^{k+1} = \Pi_Q(\tilde{y}^{k+1})$ **Projection on feasible set**

5: $k = k + 1$

6: **end while**

Method 2: Projected Gradient

Algorithm 1 Projected gradient descent

Input $x \in \mathbb{R}^n$, $y^0 = rand$, Lipschitz constant L

Output $y^* \in \mathbb{R}^n$

1: $k = 0$
2: **while** stopping criterion is not met **do**
3: $\quad \tilde{y}^{k+1} = y^k - \frac{1}{L}\nabla f(y^k)$
4: $\quad y^{k+1} = \Pi_Q(\tilde{y}^{k+1})$
5: $\quad k = k + 1$
6: **end while**

- Calculating the gradient is extremely cheap !

$$\nabla f(y) = D^H W D(y - x)$$ **= FFT – weighting - IFFT**

Method 2: Projected Gradient

Algorithm 1 Projected gradient descent

Input $x \in \mathbb{R}^n$, $y^0 = rand$, Lipschitz constant L

Output $y^* \in \mathbb{R}^n$

1: $k = 0$
2: **while** stopping criterion is not met **do**
3: $\quad \tilde{y}^{k+1} = y^k - \frac{1}{L}\nabla f(y^k)$
4: $\quad y^{k+1} = \Pi_Q(\tilde{y}^{k+1})$
5: $\quad k = k + 1$
6: **end while**

- Calculating the gradient is extremely cheap !

$$\nabla f(y) = D^H W D(y - x)$$ **= FFT – weighting - IFFT**

- Projecting onto feasible set is also extremely cheap !

$$\Pi_Q(y) := \arg\min_{y \in \mathbb{R}^N} \frac{1}{2}\|y - x\|_2^2 \quad \text{s.t.} \quad L \leq y \leq U$$ **= Hard clipping**

Three Tailored QP Solution Methods

- Method 1: External Active Set Strategy with Small Dual QPs
- Method 2: Projected Gradient
- **Method 3: Nesterov's Optimal Gradient Scheme**

Method 3 - Nesterov's Optimal Scheme

Algorithm 1 Projected gradient descent using Nesterov's optimal method

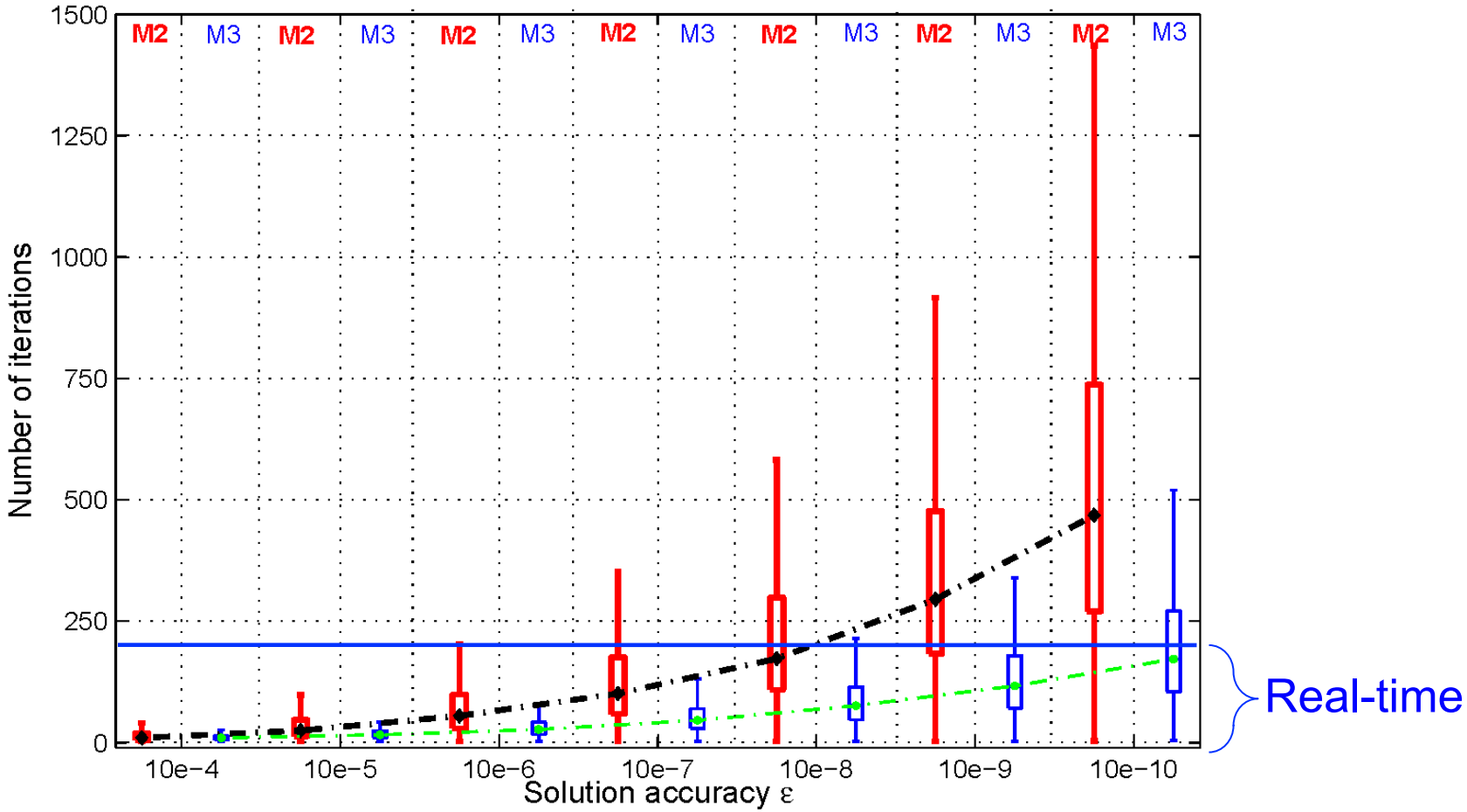
Input $x \in \mathbb{R}^n$, $y^{-1} = c^0 = rand$, $a^0 = 1$, Lipschitz constant L

Output $y^* \in \mathbb{R}^n$

1: $k = 0$
2: **while** stopping criterion is not met **do**
3: $\quad \tilde{y}^{k+1} = c^k - \frac{1}{L}\nabla f(c^k)$
4: $\quad y^{k+1} = \Pi_Q(\tilde{y}^{k+1})$
5: $\quad a^{k+1} = \frac{(1+\sqrt{4(a^k)^2+1})}{2}$
6: $\quad c^{k+1} = y^k + \frac{a^k-1}{a^{k+1}}(y^k - y^{k-1})$
7: $\quad k = k + 1$
8: **end while**

- Minor code modifications to standard projected gradient
- one extra vector addition: negligible extra cost per iteration
- faster convergence, provably with optimal rate $O(\frac{1}{\sqrt{\varepsilon}})$ [Nesterov 1983]

Audio CPU Test: Gradient (M2) vs. Nesterov (M3)

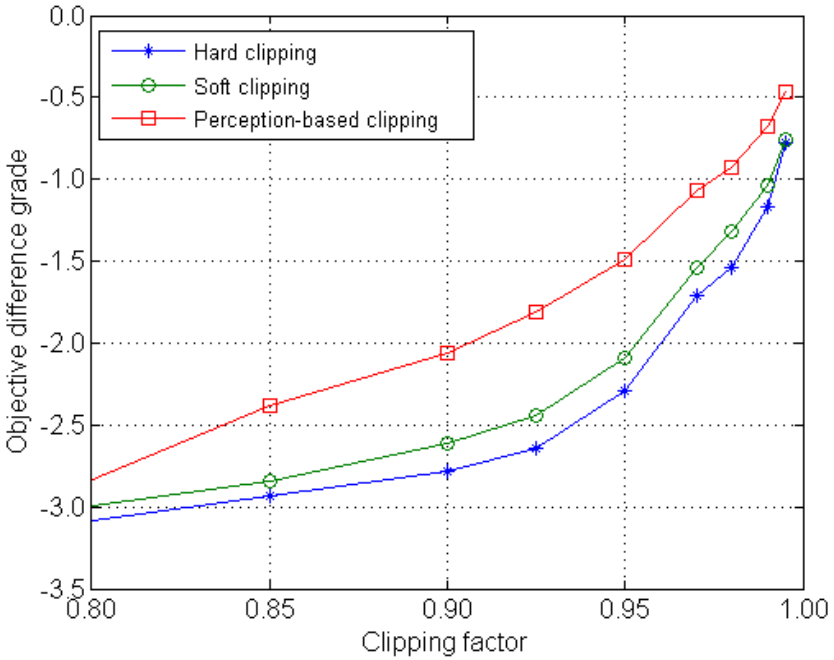


- Nesterov's scheme real-time feasible below accuracy 10^{-8}
- Already 10^{-6} delivers no perceptual difference to exact solution

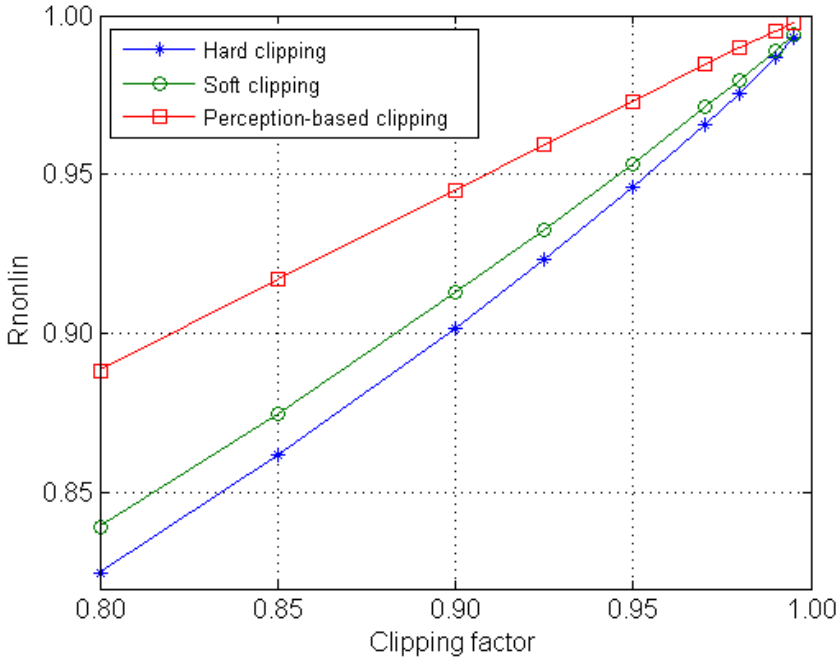
Comparative evaluation of sound quality

- Two objective measures of sound quality
- Averaged scores over eight audio signals
- Perception-based clipping results in significantly higher scores as compared to the other clipping techniques, for all clipping factors

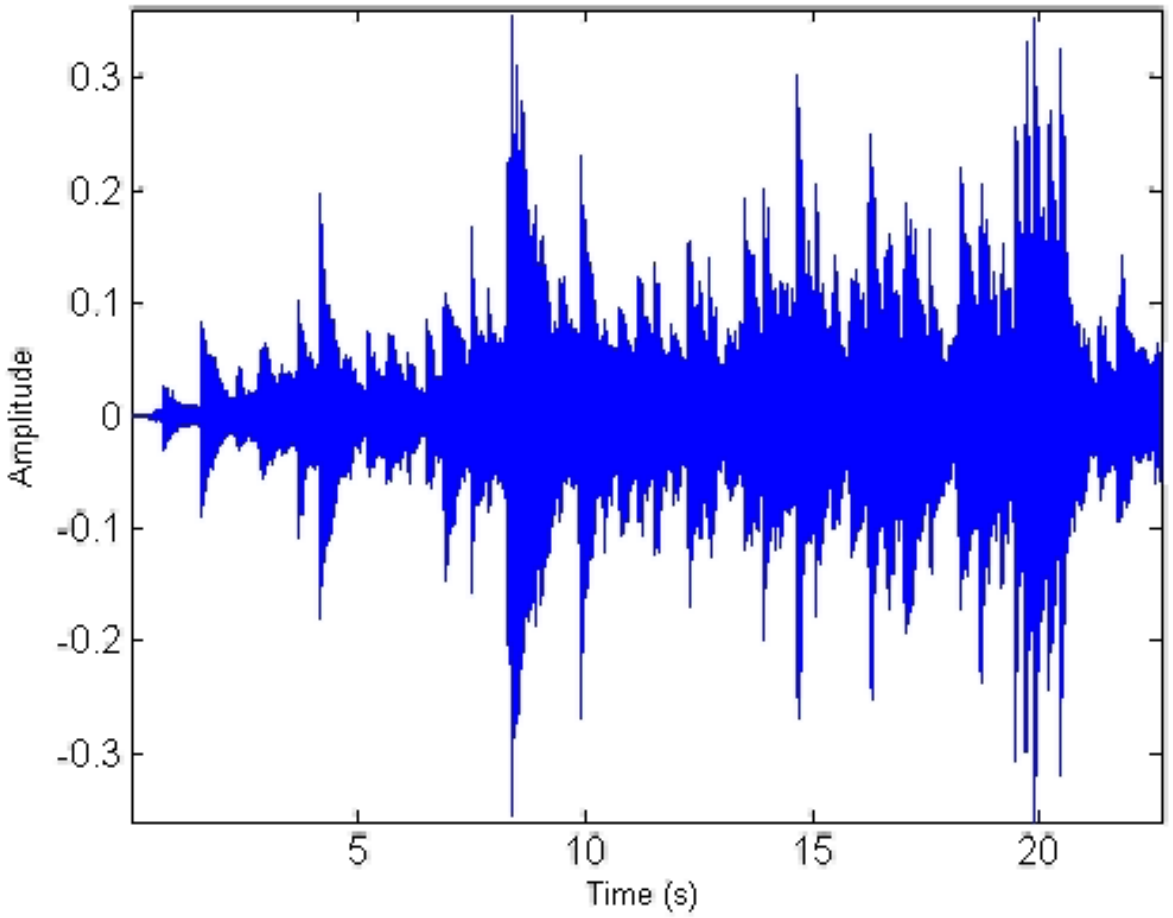
PEAQ



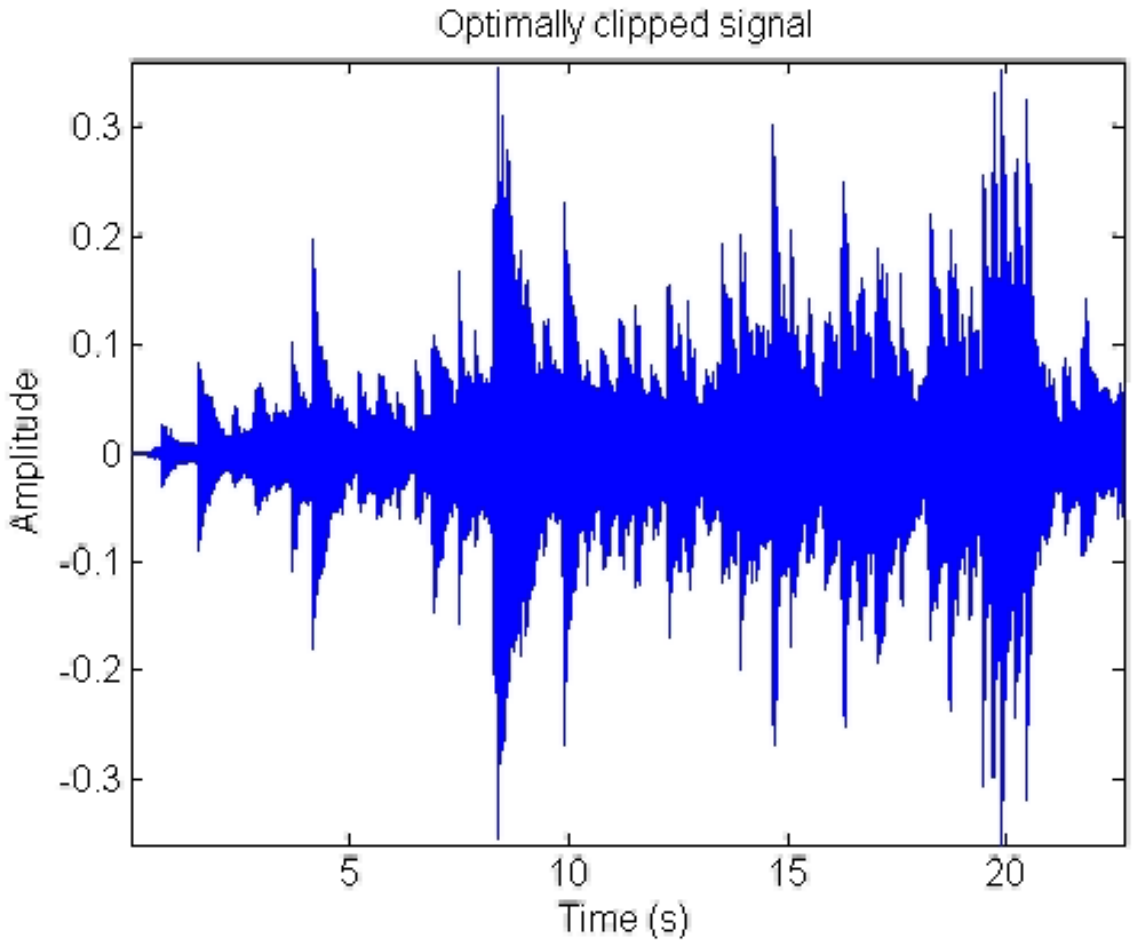
Rnonlin



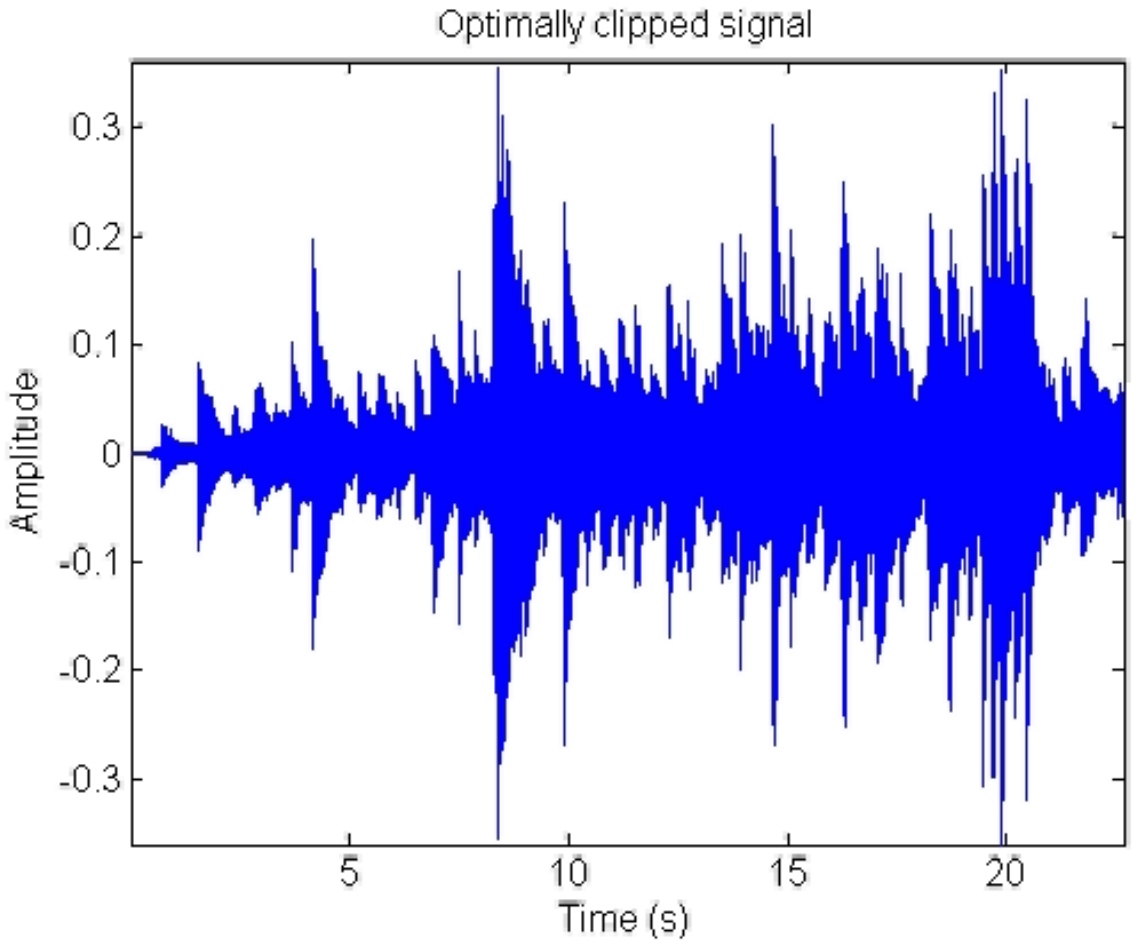
Hard Clipped Signal



Optimally Clipped by Nesterov's Gradient Scheme



Optimally Clipped by Nesterov's Gradient Scheme



Bruno's next steps:

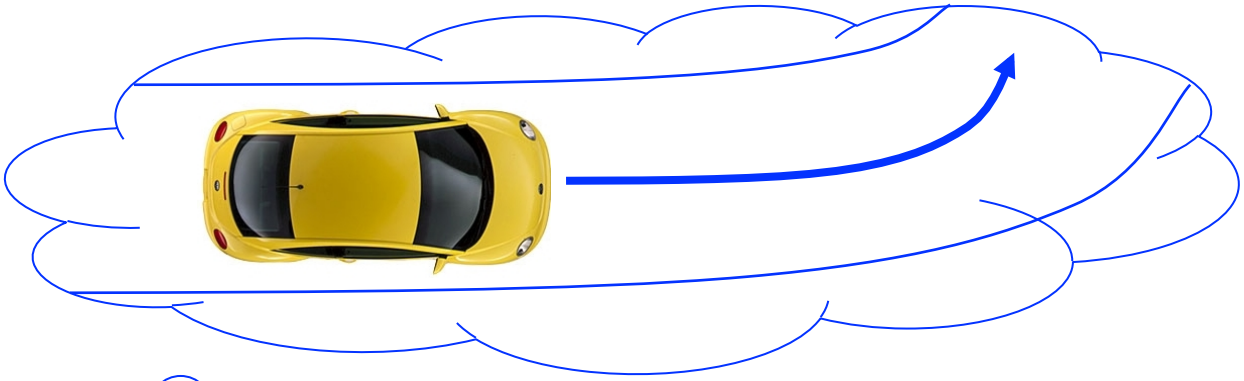
- implement perception-based clipping as slim, fast C-code
- explore its use within hearing aids

Time Optimal Model Predictive Control for Machine Tools

with **Lieboud Vanden Broeck, Hans Joachim Ferreau,** Jan Swevers, M. D.

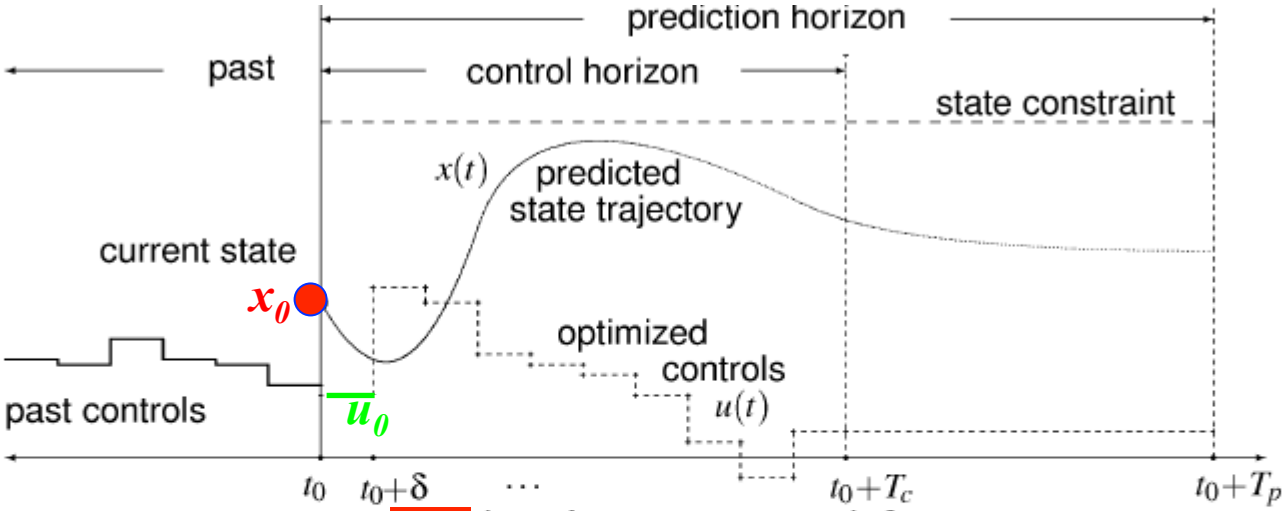
Model Predictive Control (MPC)

Always look a bit into the future.



Brain predicts and optimizes:
e.g. slow down **before** curve

Computations in Model Predictive Control (MPC)



1. Estimate current system state x_0 (and parameters) from measurements.
2. Solve *in real-time* an optimal control problem:

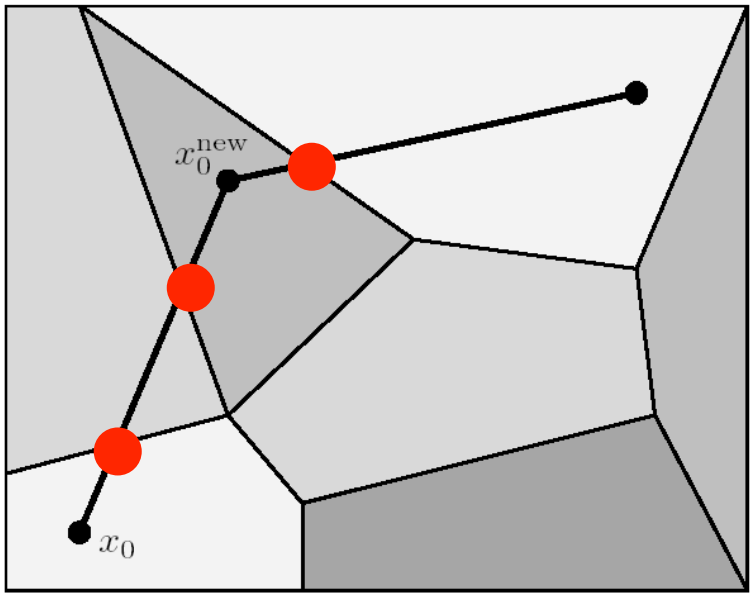
$$\min_{x,z,u} \int_{t_0}^{t_0+T_p} L(x,z,u)dt + E\left(x(t_0+T_p)\right) \; s.t. \begin{cases} x(t_0) - x_0 = 0, \\ \dot{x} - f(x,z,u) = 0, \; t \in [t_0, t_0+T_p] \\ g(x,z,u) = 0, \; t \in [t_0, t_0+T_p] \\ h(x,z,u) \geq 0, \; t \in [t_0, t_0+T_p] \\ r\left(x(t_0+T_p)\right) \geq 0. \end{cases}$$

3. Implement first control u_0 for time δ at real plant. Set $t_0 = t_0 + \delta$ and go to 1.

Main challenge for MPC: fast and reliable real-time optimization

qpOASES: Tailored QP Solver

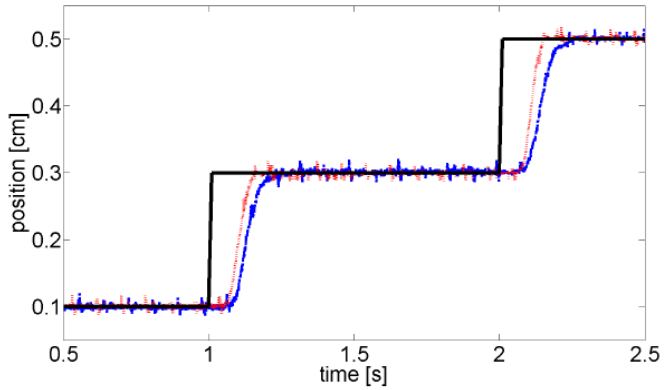
Solve p-QP via „Online Active Set Strategy“:

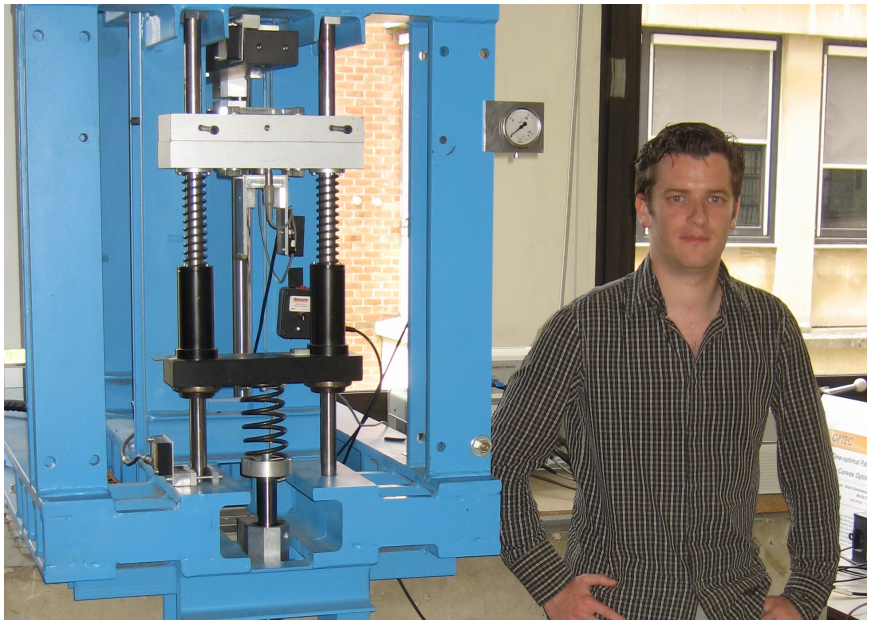


- go on straight line in parameter space from old to new problem data
- **solve each QP on path exactly (keep primal-dual feasibility)**
- Update matrix factorization at boundaries of critical regions
- Up to 10 x faster than standard QP

qpOASES: open source C++ code by Hans Joachim Ferreau

Time Optimal MPC: a 100 Hz Application

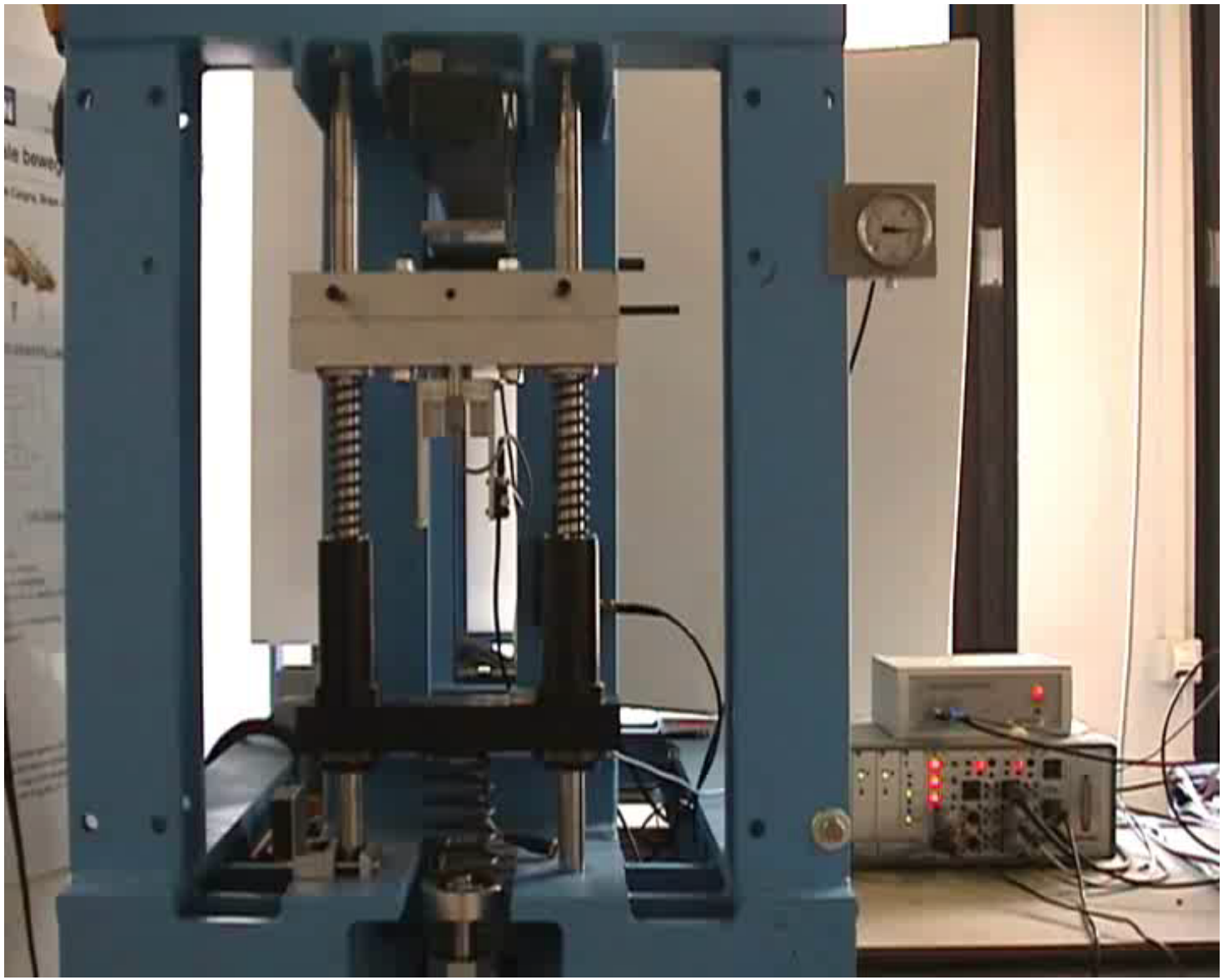




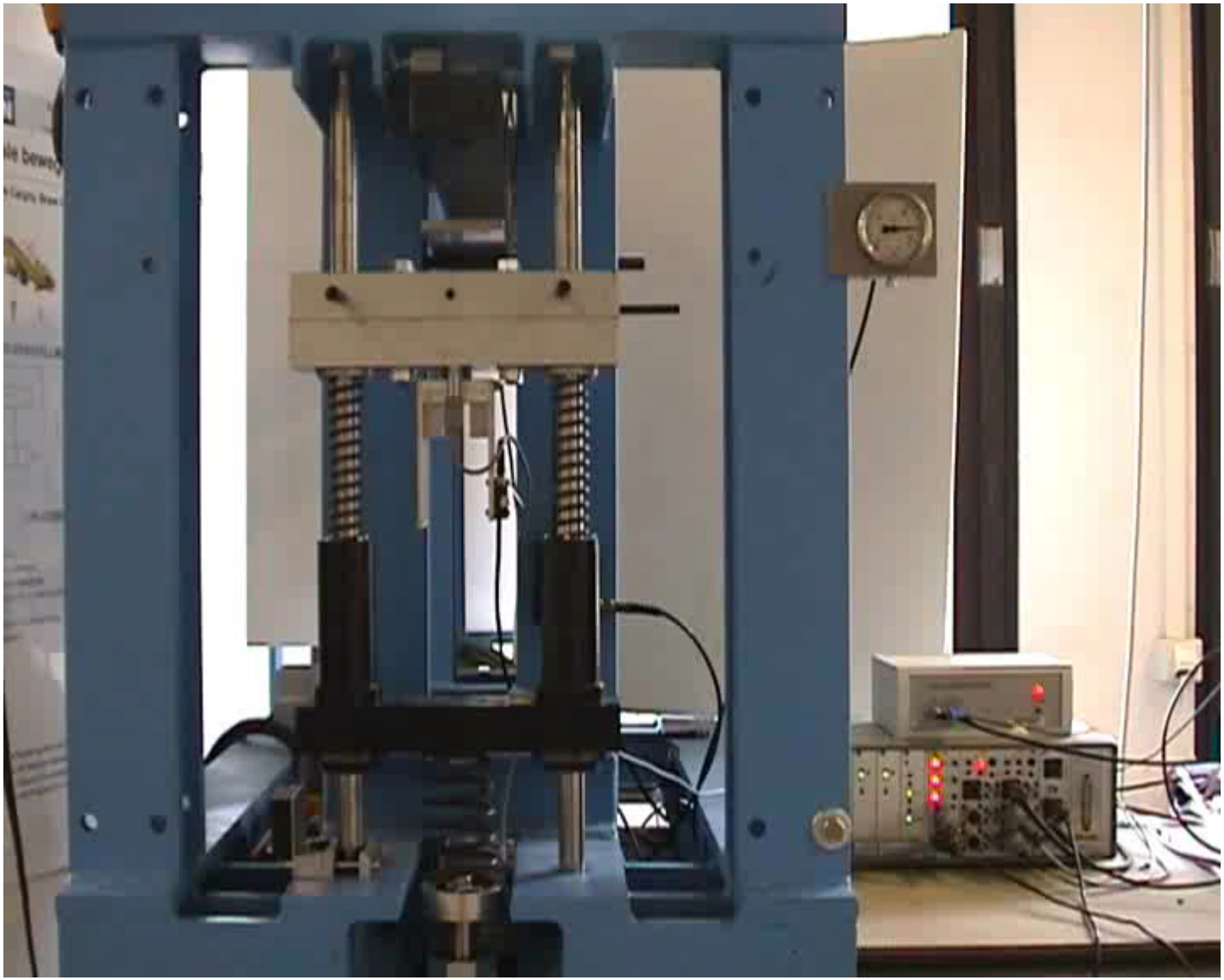
- Quarter car: oscillating spring damper system
- MPC Aim: settle at any new setpoint in ***in minimal time***
- Two level algorithm: MIQP
  - 6 online data
  - 40 variables + one integer
  - 242 constraints (in-&output)
- use qpOASES on dSPACE
- **CPU time: <10 ms**

Lieboud Van den Broeck in front of quarter car experiment

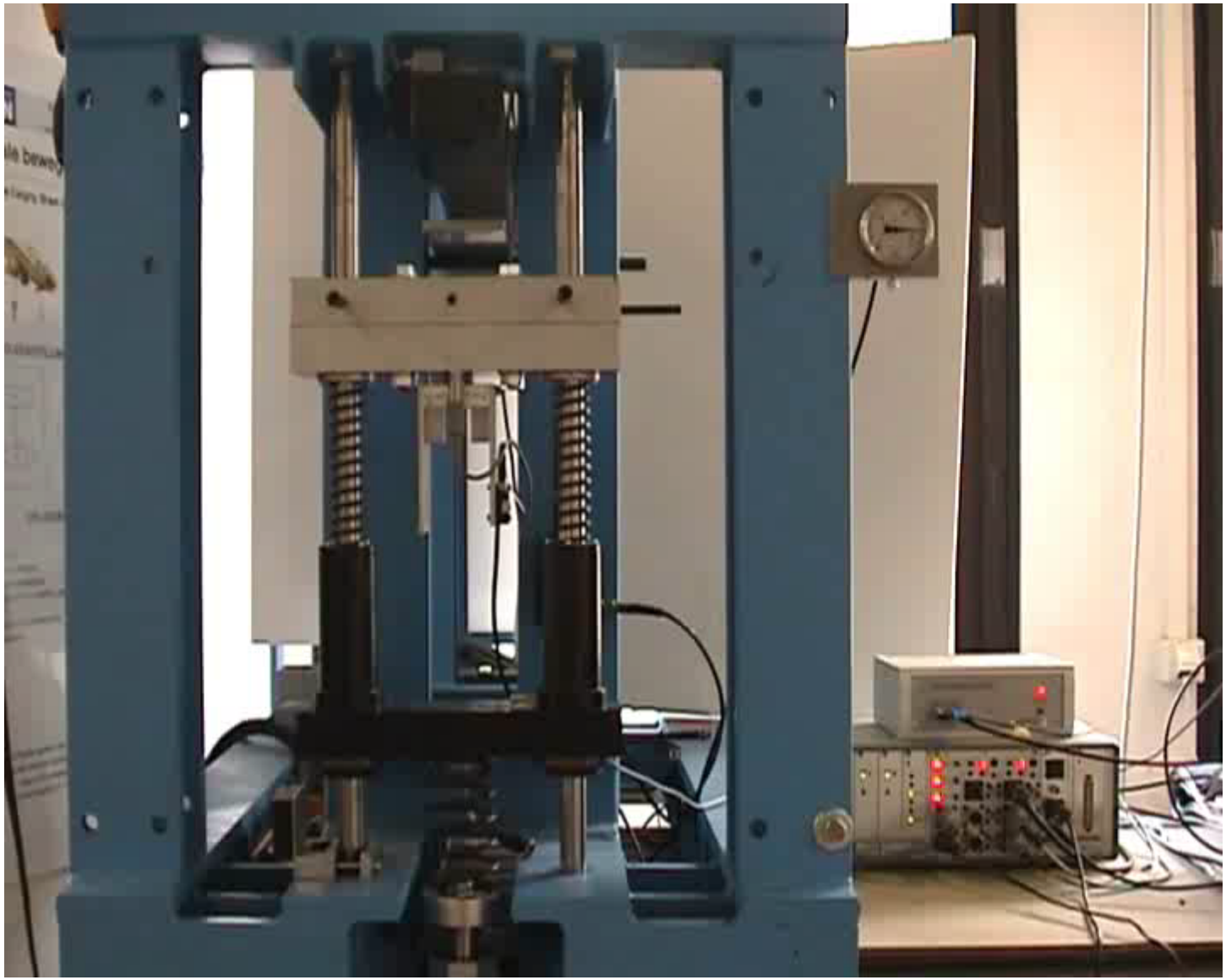
Setpoint change without control: oscillations



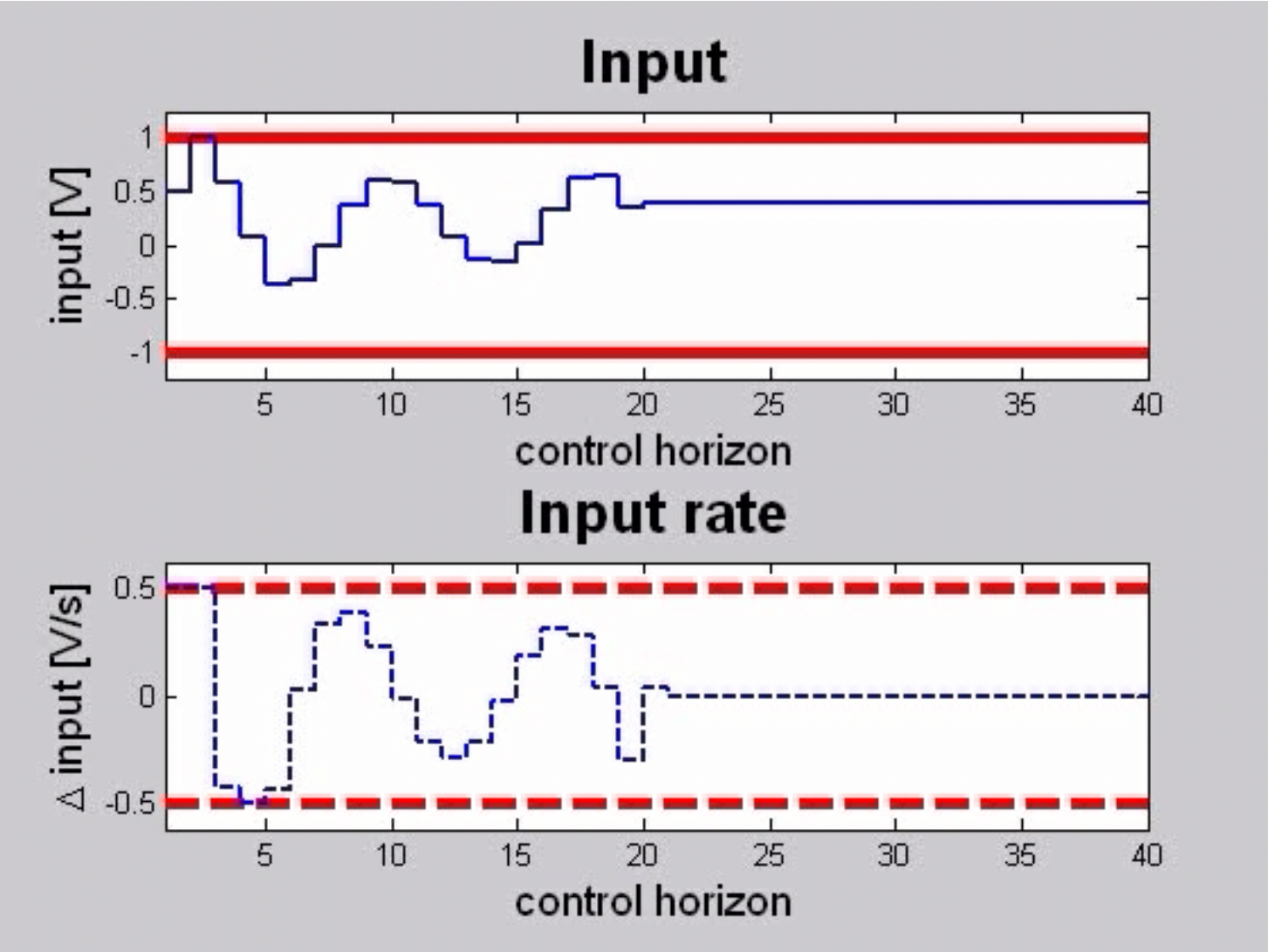
With LQR control: inequalities violated



With Time Optimal MPC



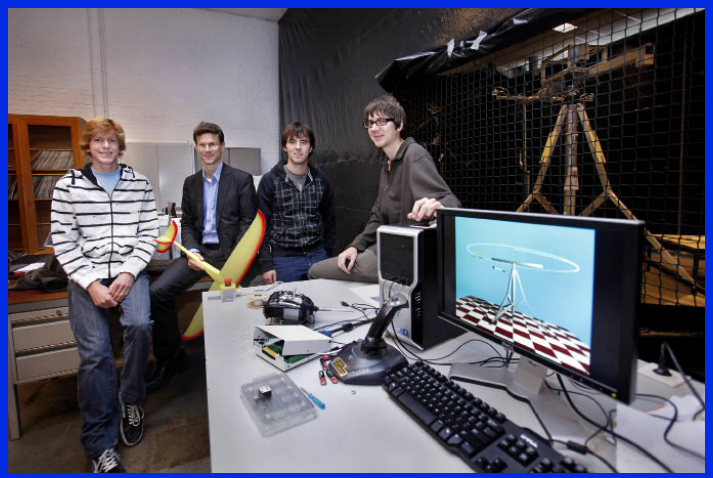
Time Optimal MPC: qpOASES Optimizer Contents



qpOASES running on Industrial Control Hardware (20 ms)

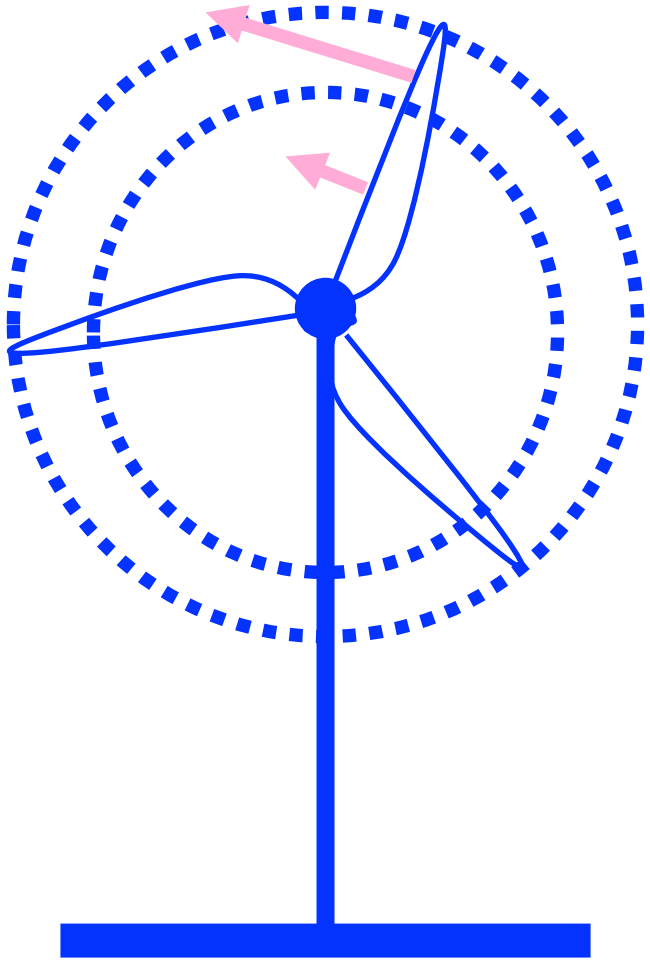
Project manager (Dec. 2008): “…we had NO problem at all with the qpOASES code. Your Software has throughout the whole project shown reliable and robust performance.”

Optimization and Control of Tethered Airplanes for Wind Power Generation



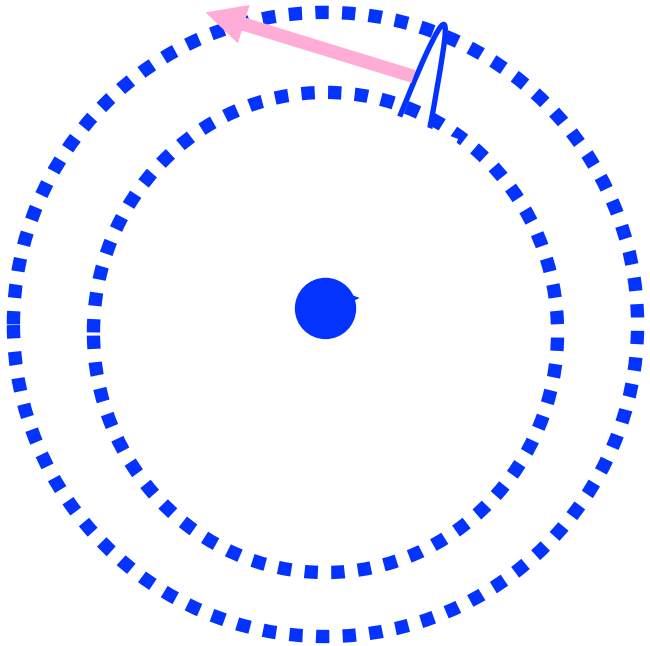
with **Kurt Geebelen, Reinhart Paelinck, Joris Gillis**, Boris Houska, Hans Joachim Ferreau, Jan Swevers, Dirk Vandepitte, …

What is the Optimal Wind Turbine ?



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

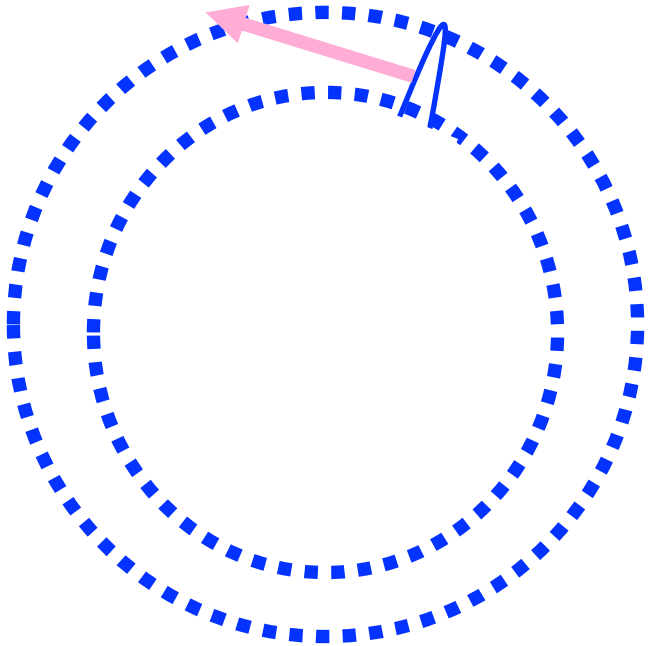
What is the Optimal Wind Turbine ?



- 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?

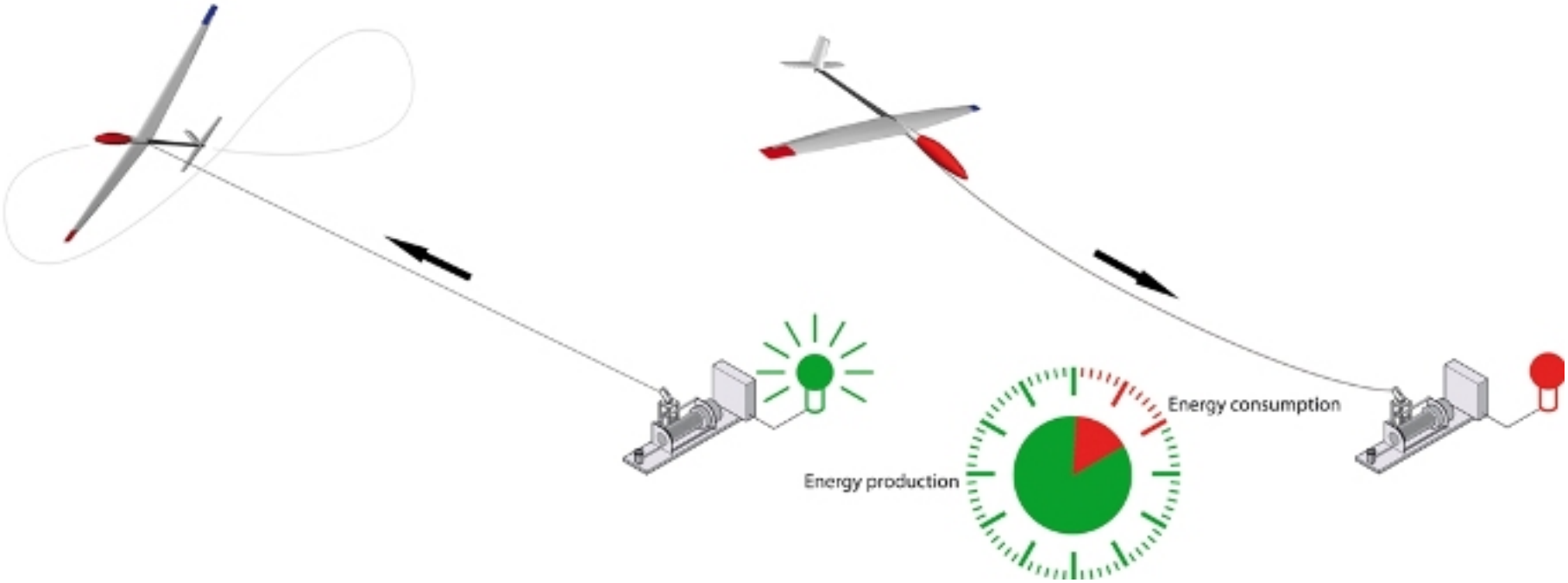
What is the Optimal Wind Turbine ?



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

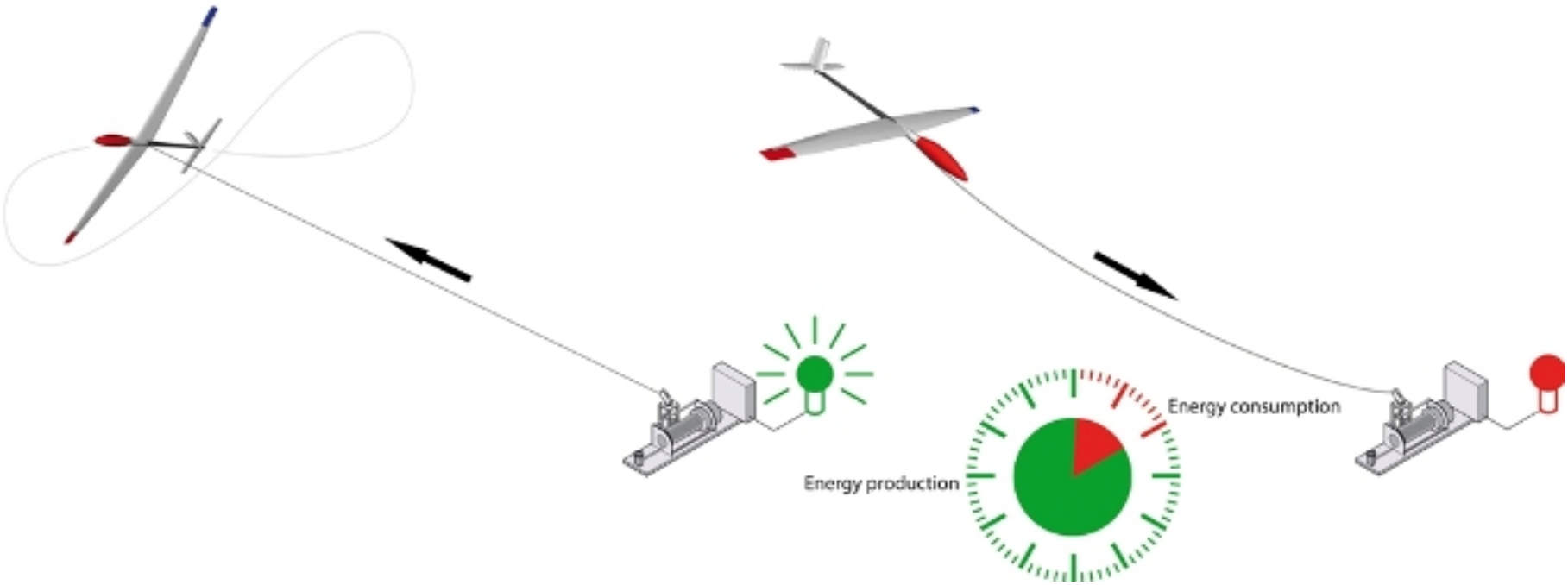
Could we construct a wind turbine with only wing tips and generator?

Idea: Wind Power by Tethered Planes



Enormous potential (e.g. 5 MW for 500 m^2 wing) . Investigated since 2005 by increasing number of people (~100 world wide, most in start-up companies)

Idea: Wind Power by Tethered Planes

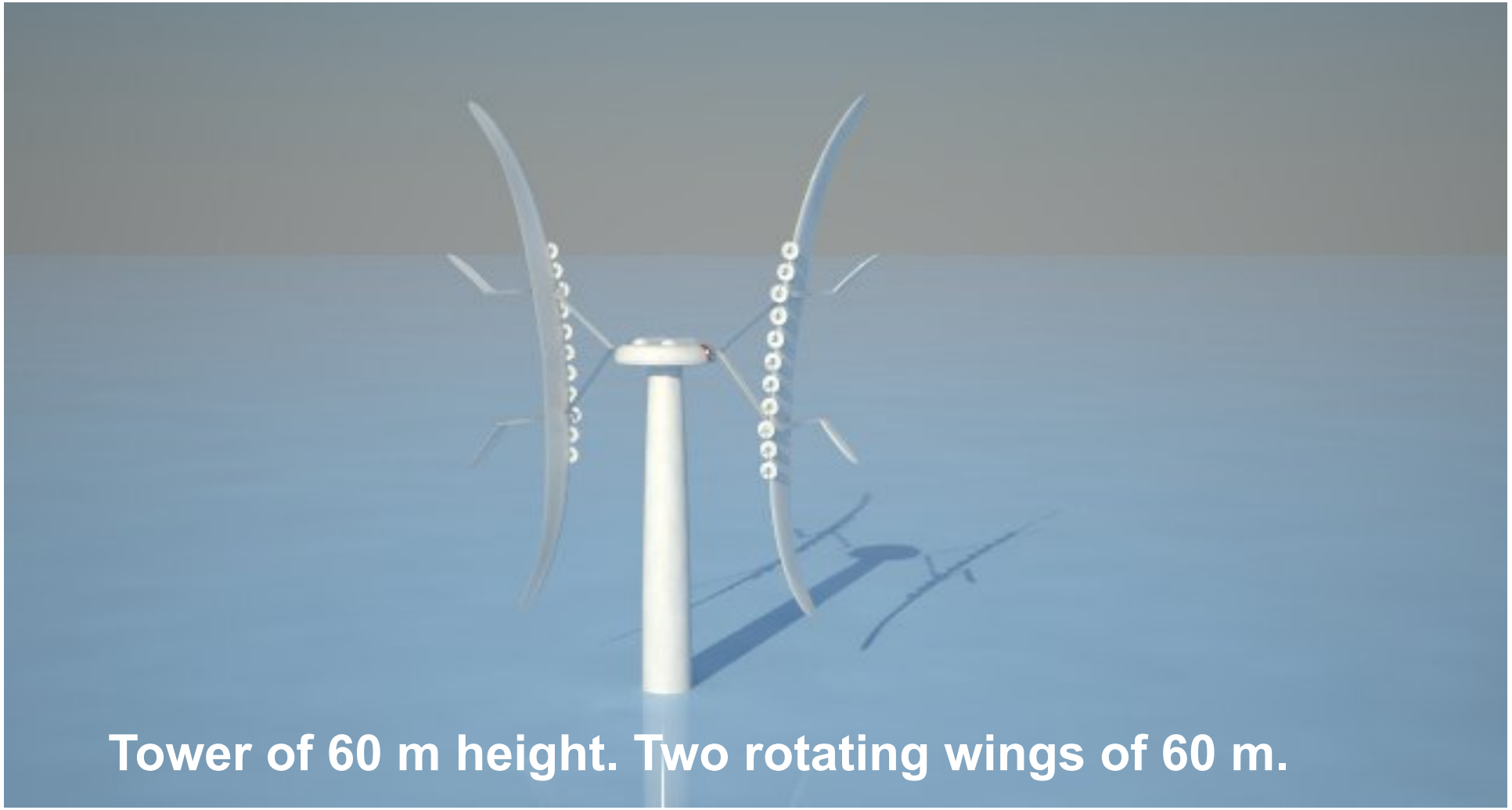


Enormous potential (e.g. 5 MW for 500 m^2 wing) . Investigated since 2005 by increasing number of people (~100 world wide, most in start-up companies)

Two major questions:

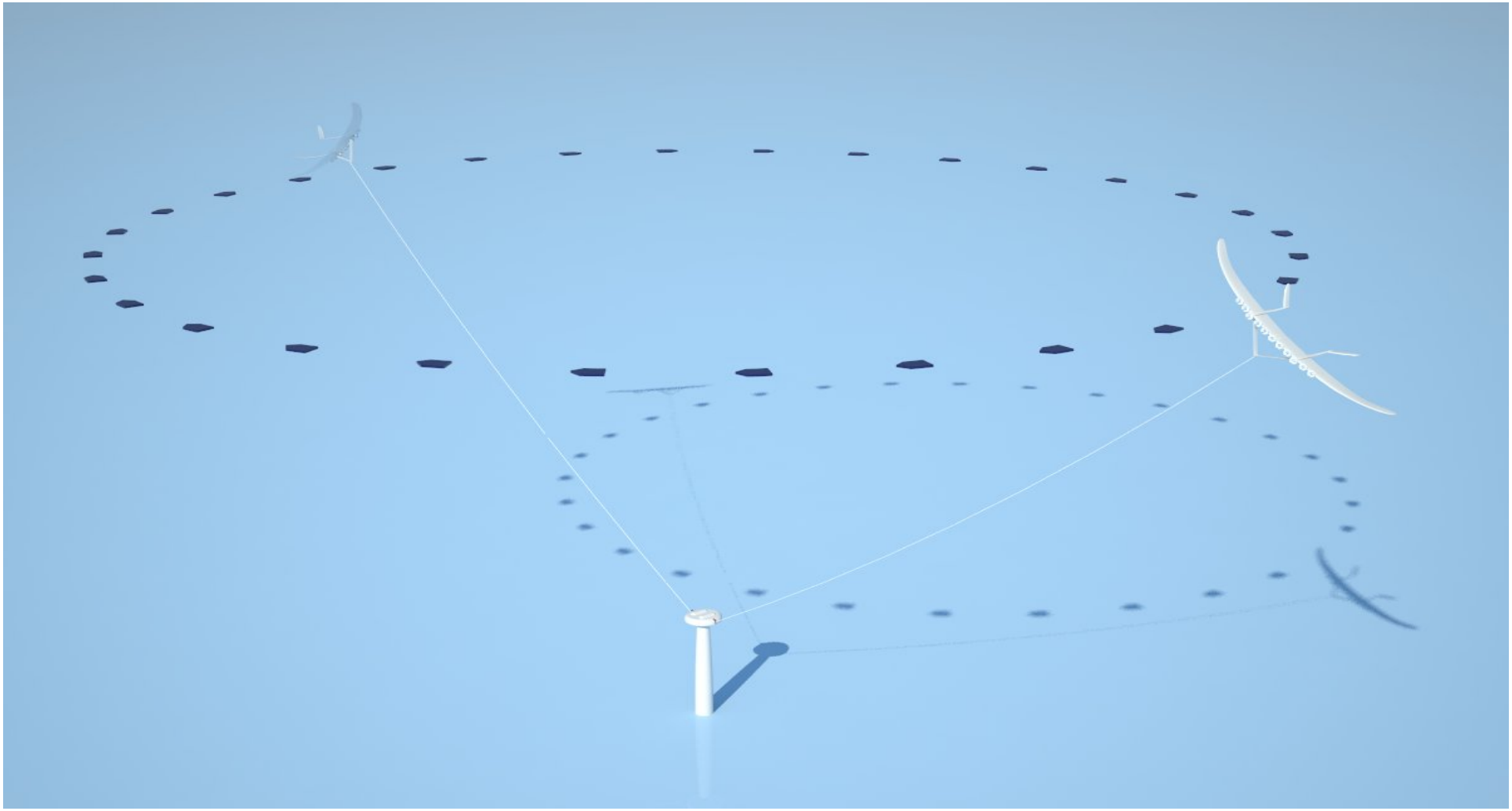
- How to start and land ?
- How to control automatically ?

Our vision: Start by Rotation (Visualized by Reinhart Paelinck)

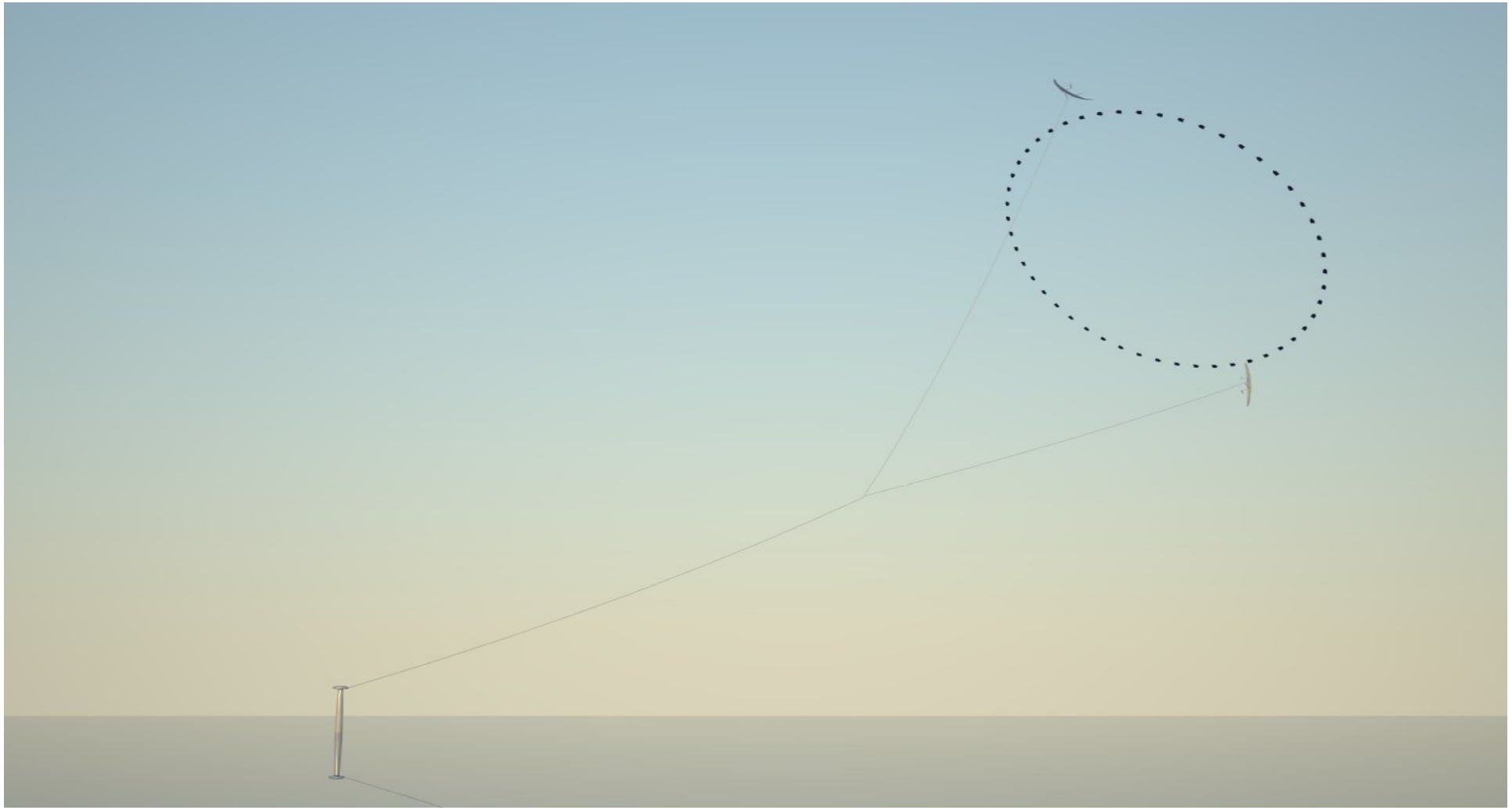


Centrifugal and lifting forces keep kites in the air

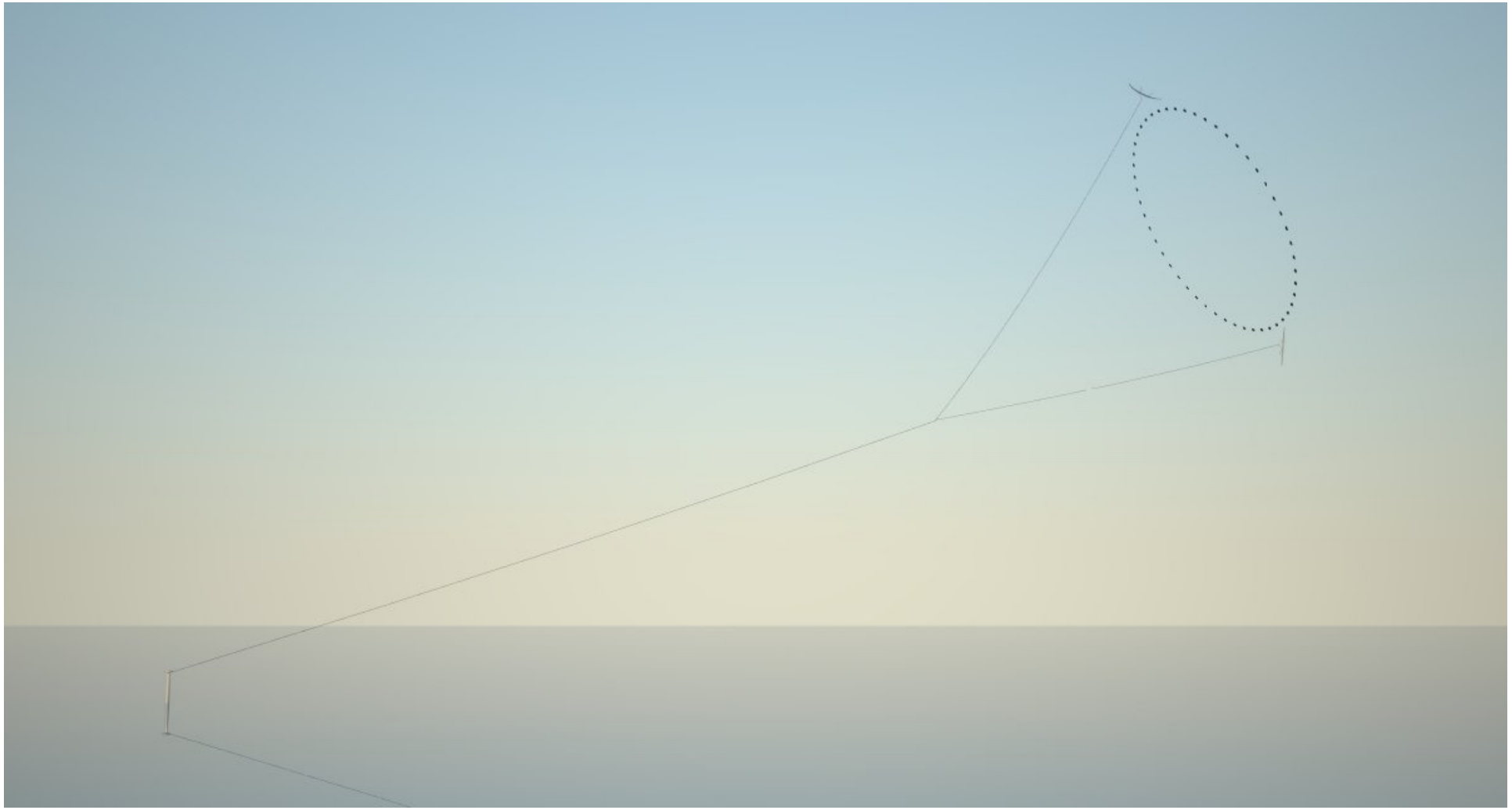
Later, lift forces dominate



Lines are connected to reduce line drag



Final orbit – A virtual 18 MW windmill is erected!

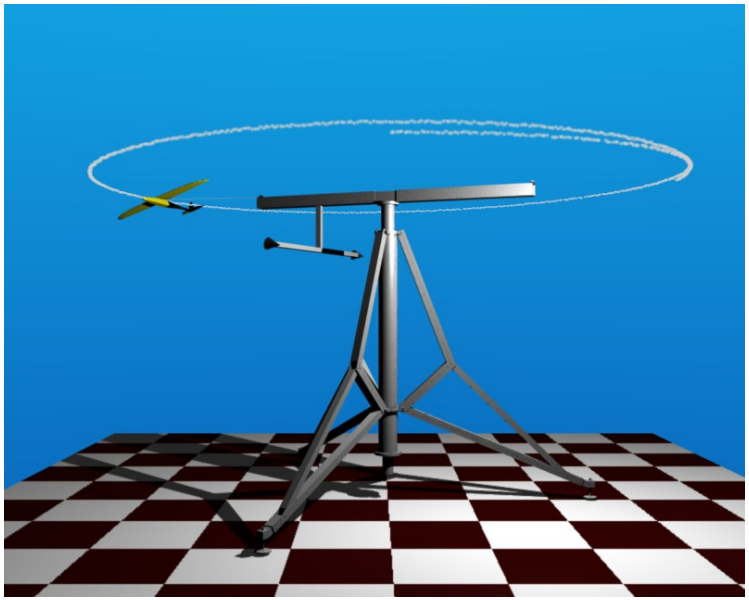


ERC Starting Grant "HIGHWIND" for 2011-2015

erc **HIGHWIND**

Simulation, Optimization, and Control of High Altitude Wind Power Generators

Aim: Guide the development of high altitude wind power, focus on *modeling, optimization, and control*, plus small scale experiments.



HIGHWIND: Optimization, Estimation, Control

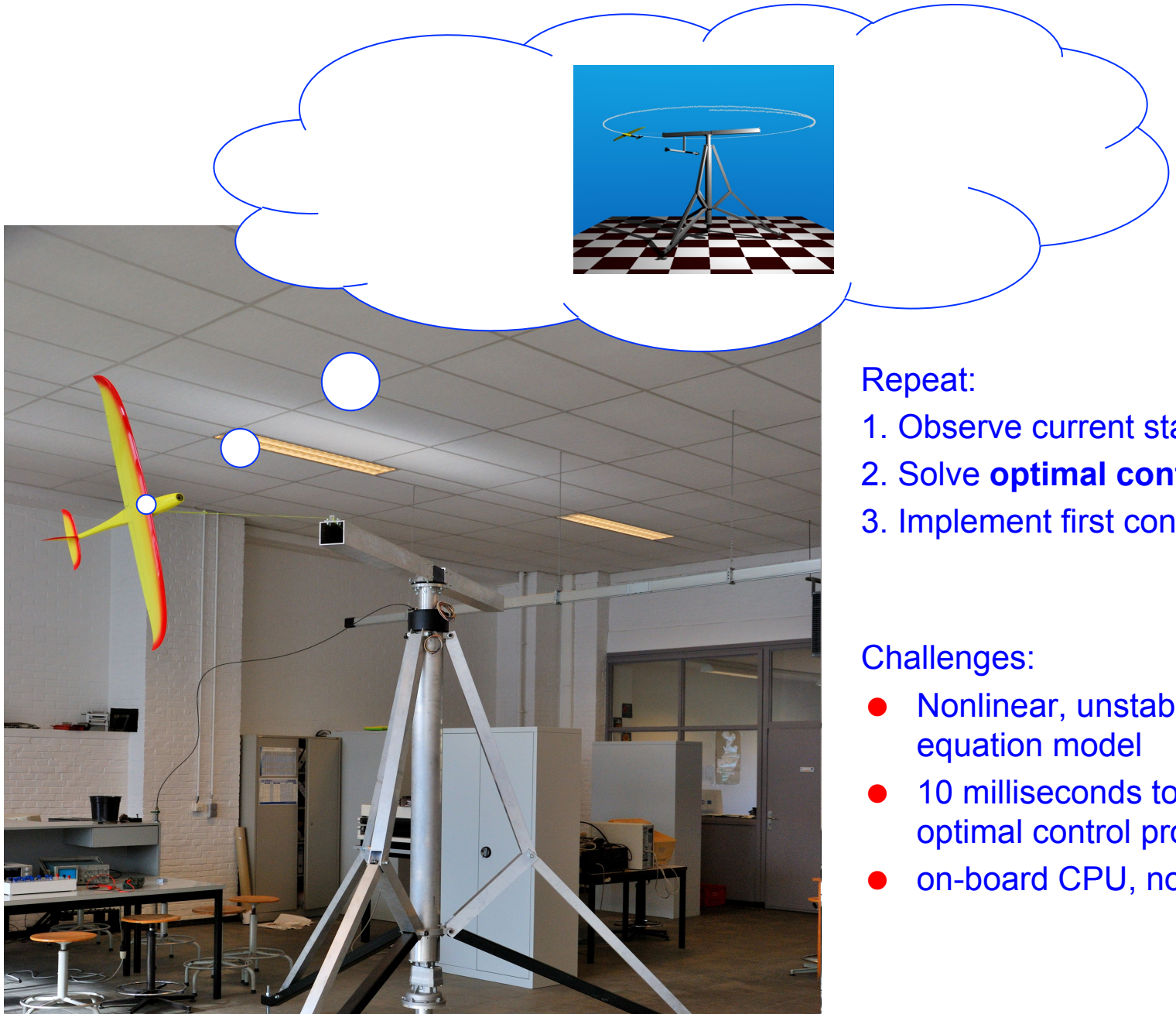
Many tasks for mathematical optimization:

- Estimate parameters of highly unstable differential equation models
- Maximize energy generated in pumping orbits
- Maximize stability and robustness of energy generating orbits
- Minimize power losses in “reverse pumping” when no wind blows

and in particular for embedded optimization:

- Estimate online system state given data from accelerometer, gyros, …
- Optimize online a predicted trajectory, minimizing tracking errors: Nonlinear Model Predictive Control (NMPC)

Nonlinear Model Predictive Control (NMPC)



Repeat:

1. Observe current state
2. Solve **optimal control** problem
3. Implement first control.

Challenges:

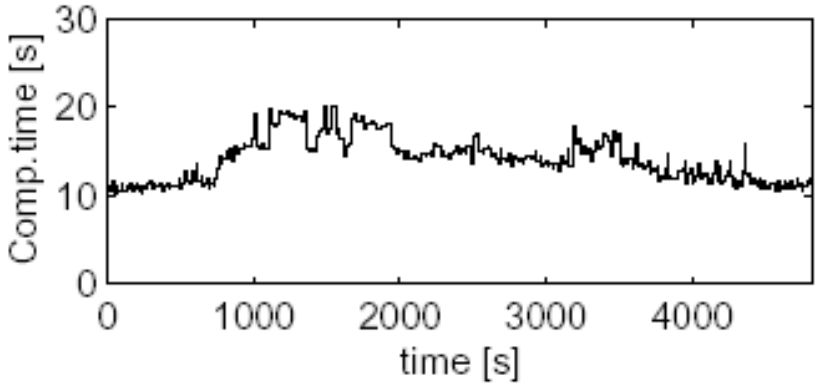
- Nonlinear, unstable differential equation model
- 10 milliseconds to solve each optimal control problem
- on-board CPU, not desktop PC

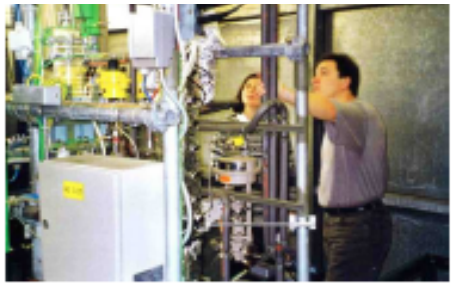
Basic Algorithm: Real-Time Iteration [D. 2001]

At each sampling time:

- simulate system on subintervals and compute sensitivities
- solve a structured quadratic program

In 2001, successfully tested at Distillation column of ISR, Stuttgart: 20 CPU seconds on high end Linux PC…





sti

[D.,
Schl

- Par
exp
- On
Filt
me
- Imp
Lin
- Co
via
- Sel

Automatic Code Generation with ACADO Toolkit

- A Toolkit for „Automatic Control and Dynamic Optimization“
- Open-source software (LGPL 3)
- Implements direct single and multiple shooting
- Developed at OPTEC by Boris Houska & Hans Joachim Ferreau

- Uses symbolic user syntax
  - to generate derivative code by automatic differentiation
  - to detect model sparsity
  - to **auto-generate C-code** for NMPC Real-Time Iterations...

Automatic Code Generation with ACADO Toolkit

- A Toolkit for „Automatic Control and Dynamic Optimization“
- Open-source software (LGPL 3)
- Implements direct single and multiple shooting
- Developed at OPTEC by Boris Houska & Hans Joachim Ferreau

- Uses symbolic user syntax
  - to generate derivative code by automatic differentiation
  - to detect model sparsity
  - to **auto-generate C-code** for NMPC Real-Time Iterations...

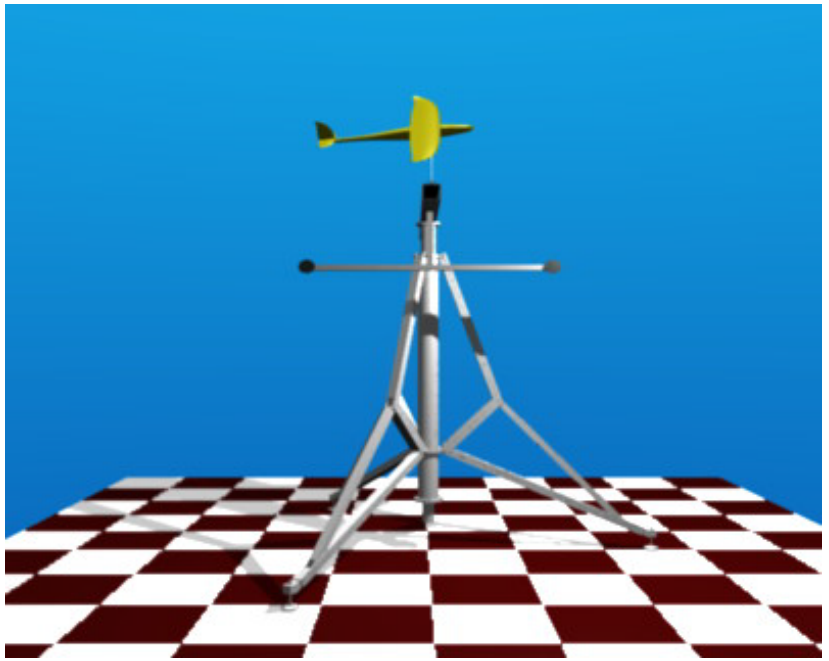
```
params.g[0] = acadoWorkspace.g[4] +
acadoWorkspace.H[4]*acadoWorkspace.deltaY[0] +
acadoWorkspace.H[18]*acadoWorkspace.deltaY[1] +
```

New auto-generated real-time iteration [Houska, Ferreau, D. 2010]

Tethered airplane optimal control problem:

- ODE model with 4 states & 2 controls
- 30 Runge-Kutta integrator steps
- 10 multiple shooting intervals
- NLP with 60 variables

Use ACADO Toolkit [Houska & Ferreau] with *automatic C-code generation*

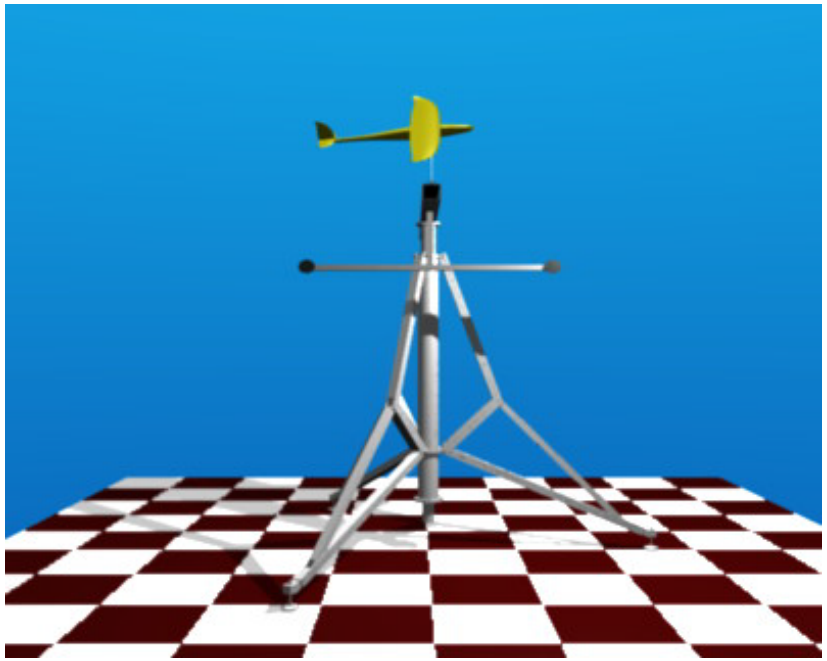


New auto-generated real-time iteration [Houska, Ferreau, D. 2010]

Tethered airplane optimal control problem:

- ODE model with 4 states & 2 controls
- 30 Runge-Kutta integrator steps
- 10 multiple shooting intervals
- NLP with 60 variables

Use ACADO Toolkit [Houska & Ferreau] with *automatic C-code generation*



	CPU time
Integration & sensitivities	711 μs
Condensing	71 μs
QP solution (with `qpOASES`)	34 μs
Remaining operations	27 μs
A complete real-time iteration	843 μs

20 000 times faster than Distillation NMPC [D. 2001]

300 000 times fast than similar NMPC problems in 1998 [Allgower et al 1998]

First Controlled Indoors Test Flights (May 2011)

Summary

- Embedded Optimization promises to revolutionize all aspects of control engineering and signal processing
- It needs sophisticated numerical methods
- We develop new embedded optimization algorithms and code them in open-source software
- Powerful tool in applications:
  - Optimal clipping for hearing aids (convex, medium, 100 Hz)
  - Time Optimal MPC for machine tools (series of QPs, small, 100 Hz)
  - Predictive control of tethered airplanes (non-convex, small, 1000 Hz)

Announcement:

- open postdoc position in ERC project HIGHWIND on automatic control of tethered UAVs

Appendix

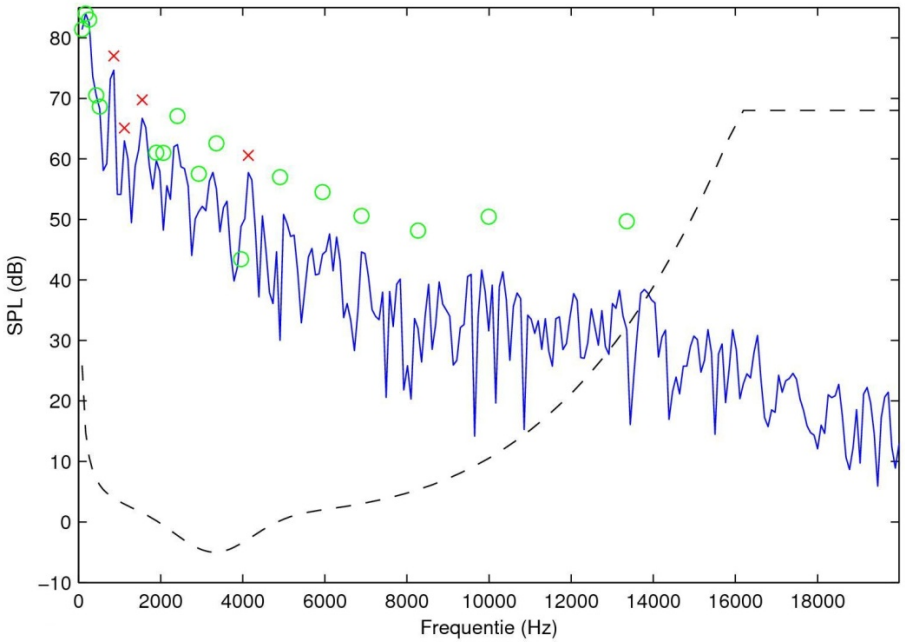
How to compute the frequency weights ?

- Use MPEG 1 Layer 1 Psychoacoustic Model
- Compute instantaneous global masking threshold of input frame, i.e. the "Amount of distortion energy (dB) at each frequency bin that can be masked by the input signal"
- Three steps:
  1. Identification of tonal and noise maskers
  2. Calculation of individual masking thresholds
  3. Calculation of global masking threshold

How to compute the frequency weights ?

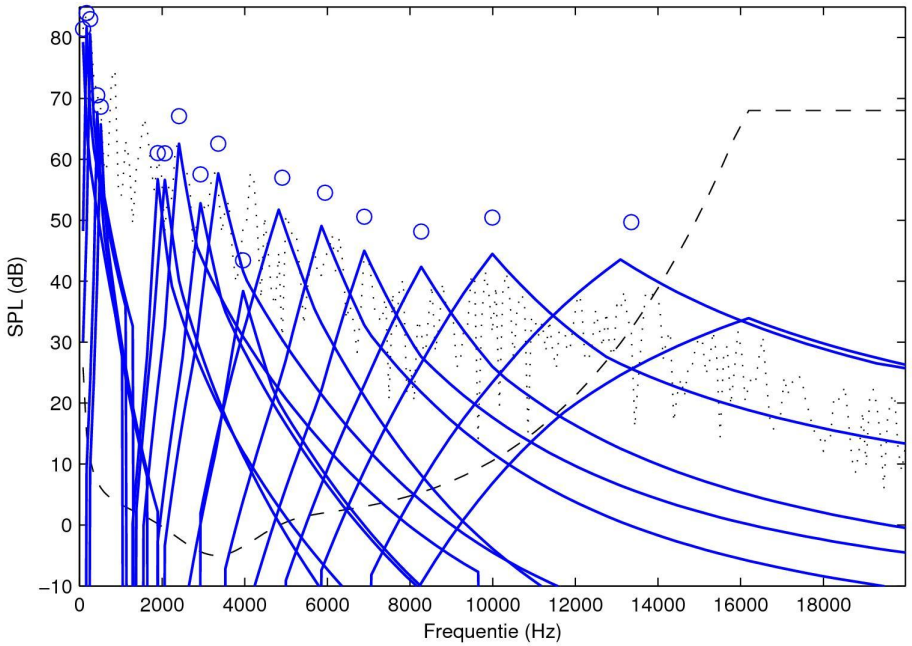
- Use MPEG 1 Layer 1 Psychoacoustic Model
- Compute instantaneous global masking threshold of input frame, i.e. the "Amount of distortion energy (dB) at each frequency bin that can be masked by the input signal"
- Three steps:

1. **Identification of tonal and noise maskers**
2. Calculation of individual masking thresholds
3. Calculation of global masking threshold



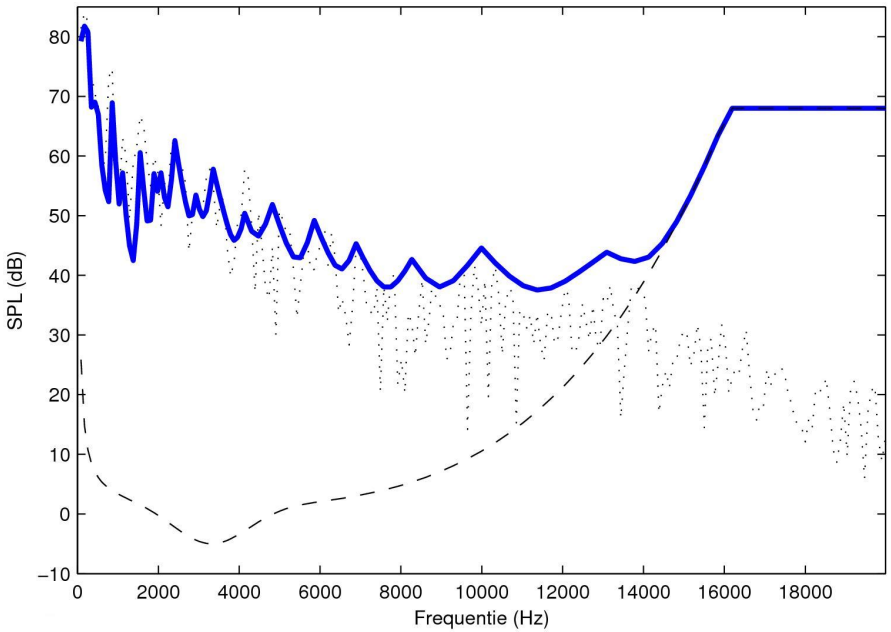
How to compute the frequency weights ?

- Use MPEG 1 Layer 1 Psychoacoustic Model
- Compute instantaneous global masking threshold of input frame, i.e. the "Amount of distortion energy (dB) at each frequency bin that can be masked by the input signal"
- Three steps:
  1. Identification of tonal and noise maskers
  2. **Calculation of individual masking thresholds**
  3. Calculation of global masking threshold



How to compute the frequency weights ?

- Use MPEG 1 Layer 1 Psychoacoustic Model
- Compute instantaneous global masking threshold of input frame, i.e. the "Amount of distortion energy (dB) at each frequency bin that can be masked by the input signal"
- Three steps:
  1. Identification of tonal and noise maskers
  2. Calculation of individual masking thresholds
  3. **Calculation of global masking threshold**



Method 2: Projected Gradient

- Regard $\min_{y \in \Omega} f(y)$ with $f(y) = \frac{1}{2}(y-x)^T D^H W D (y-x)$ &
$\Omega = \{y | L \leq y \leq U\}$

- compute Lipschitz constant of gradient = largest Eigenvalue of Hessian:

$$L := \lambda_{\max}(D^H W D) = \max_{1 \leq i \leq N} w_i$$

- Perform projected gradient steps:

$$y_{k+1} = \Pi_Q(y_k - \frac{1}{L} \nabla f(y_k))$$