



**HAL**  
open science

# Tutorial: Introduction to Interior Point Methods for Predictive Control

Eric Kerrigan

► **To cite this version:**

Eric Kerrigan. Tutorial: Introduction to Interior Point Methods for Predictive Control. SADC Summer School 2011 - Optimal Control, Sep 2011, London, United Kingdom. inria-00629539

**HAL Id: inria-00629539**

**<https://inria.hal.science/inria-00629539>**

Submitted on 6 Oct 2011

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# Introduction to Interior Point Methods for Predictive Control

**Eric Kerrigan**

Department of Aeronautics

Department of Electrical and Electronic Engineering



# Overview

## Introduction and motivation

- Why control & why feedback?
- Predictive control as a feedback strategy

## Constrained LQR (CLQR) to Quadratic Program (QP)

- Sampled-data CLQR and equivalent discrete-time CLQR
- Non-condensed and condensed QP formulations

## Algorithms for solving QP

- Interior point methods
- Exploiting structure of QP

## Hardware issues

- Interaction between algorithm choice and hardware
- Parallel computation and pipelining

## Wrap-up and questions

# Motivation for Control

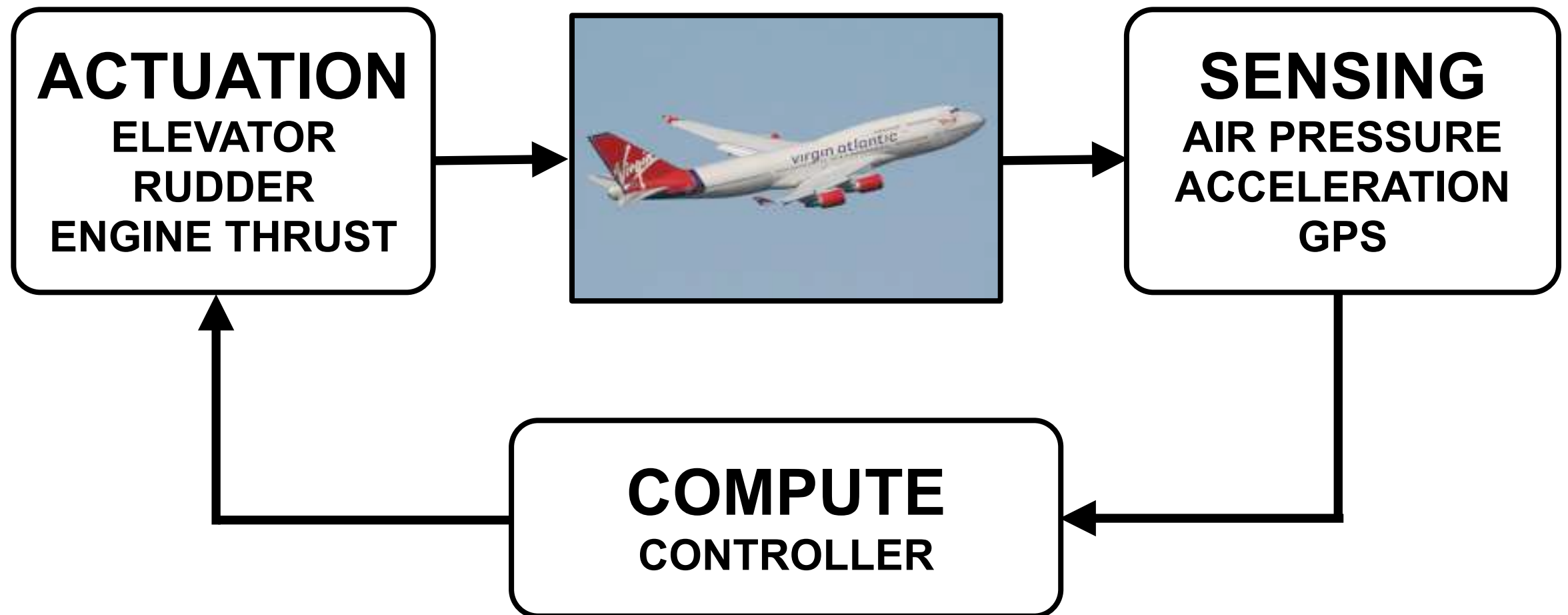
Predict behaviour of a dynamical system in an **uncertain environment**

**Uncertain Environment** = unmodelled dynamics + disturbances + measurement noise + other



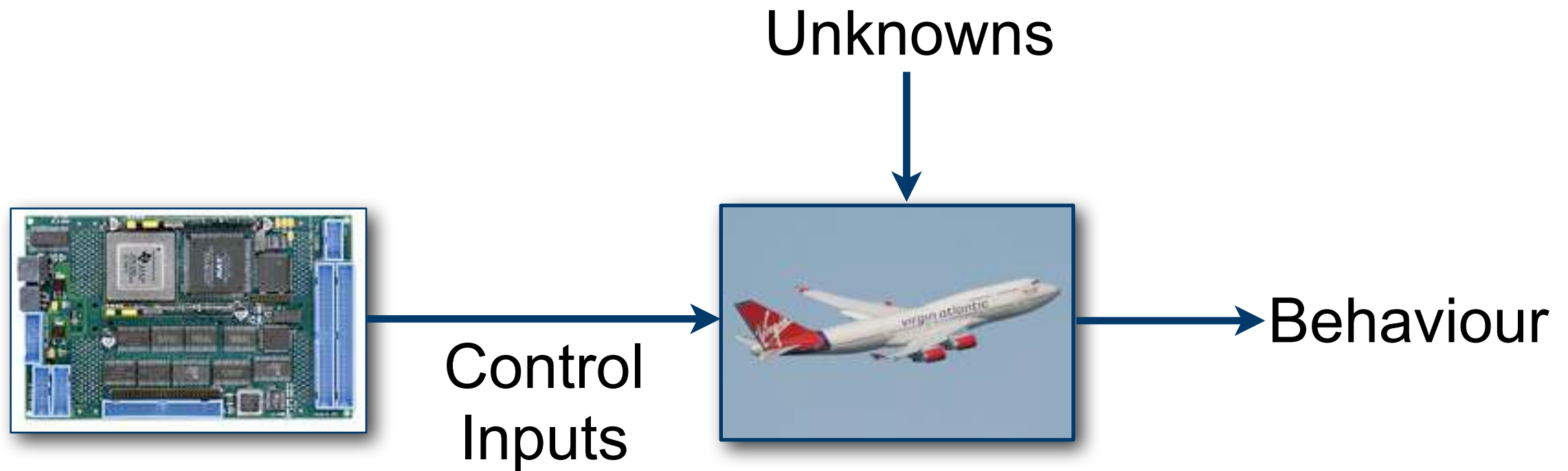
# What is Control?

Control = sensing + computation + actuation

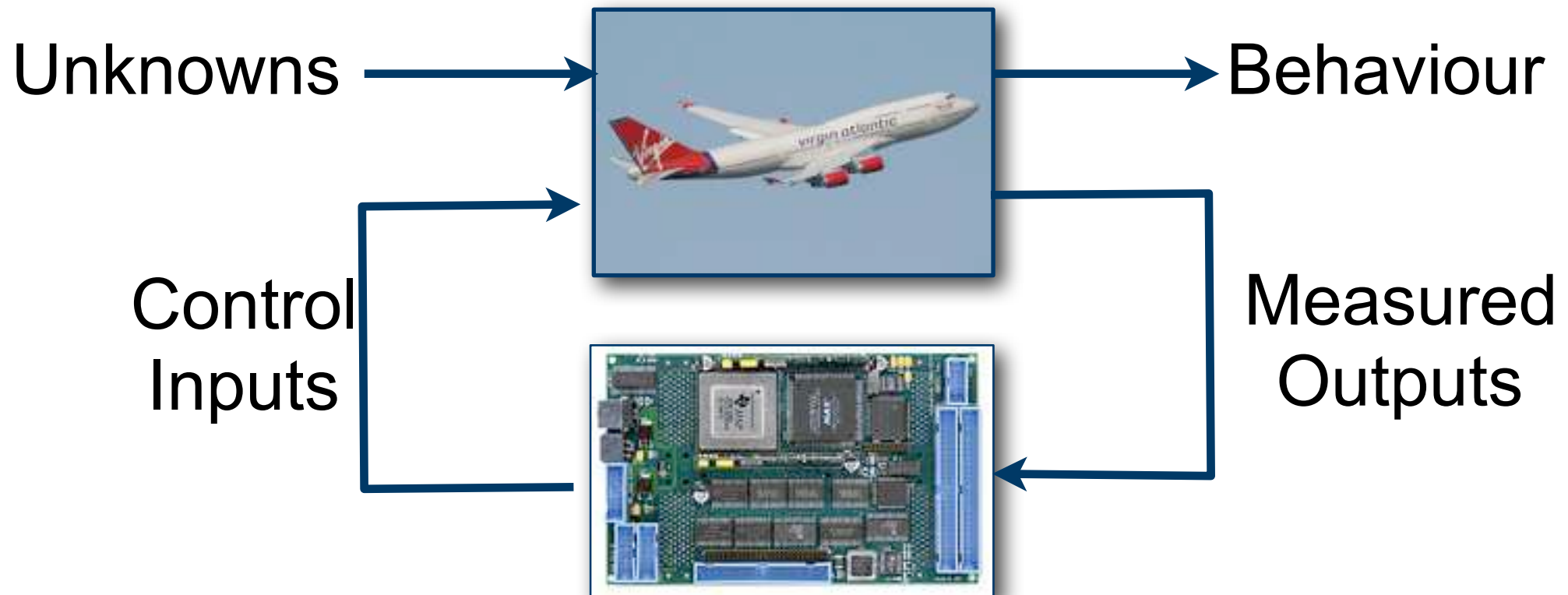


Use feedback **if and only** if there is uncertainty  
uncertainty includes noise, disturbances, modelling error

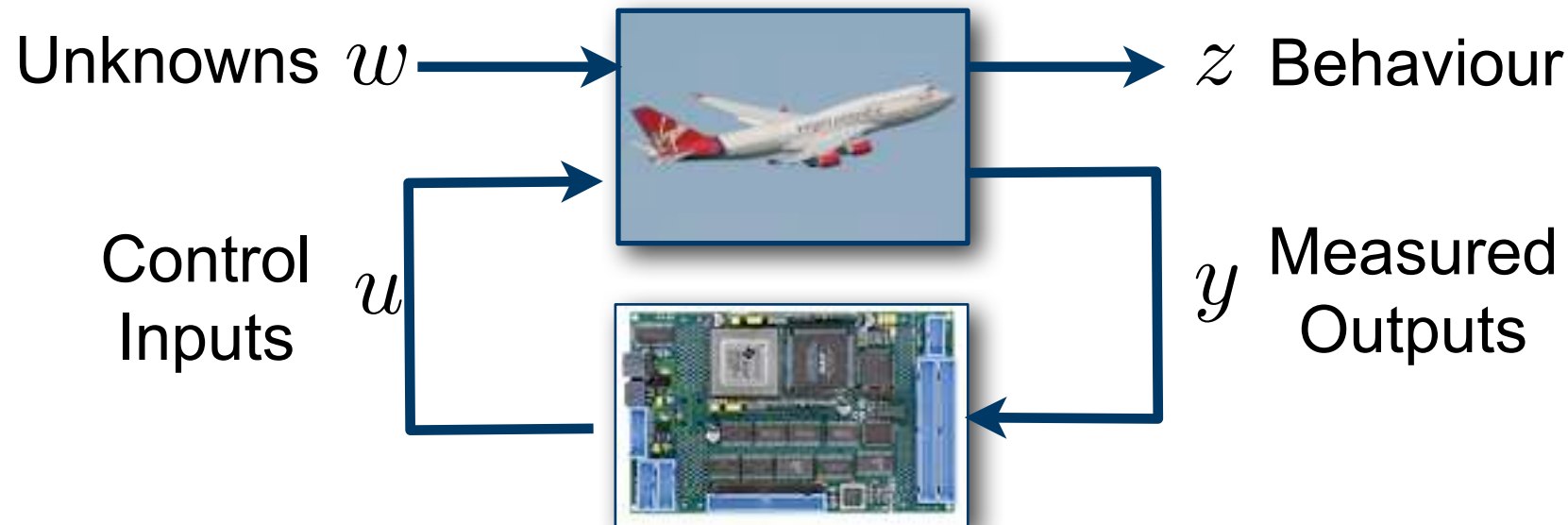
# Open-loop or Closed-loop?



or



# Why Closed-loop?



Linear Dynamics

$$z = Fu + Gw$$

$$y = Hu + Jw$$

$$u = Ky$$

No control:  $z = Gw \quad (u = 0)$

Open-loop:  $z = Fu + Gw$

Feedback:  $z = \left[ FK (I - HK)^{-1} J + G \right] w$

Feedback is the ONLY way to attenuate unknown  $w$



# Drawbacks of Feedback

Can destabilise a system

- Example: JAS 39 Gripen crash
- Open-loop unstable
- Input constraints saturated
- Outside controllable region

Couples different parts of the system

- Might inject measurement noise

Complicates manufacture and commissioning

This is why we use feedback ***only if*** there is uncertainty



# Drawbacks of Feedback

## Can destabilise a system

- Example: JAS 39 Gripen crash
- Open-loop unstable
- Input constraints saturated
- Outside controllable region



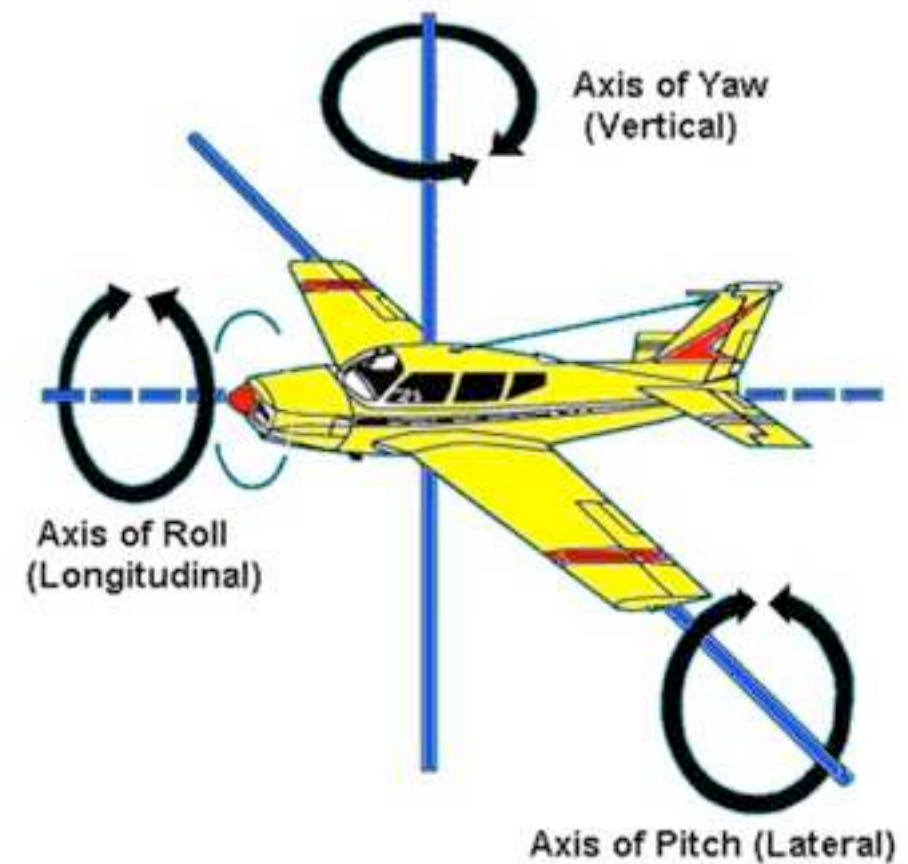
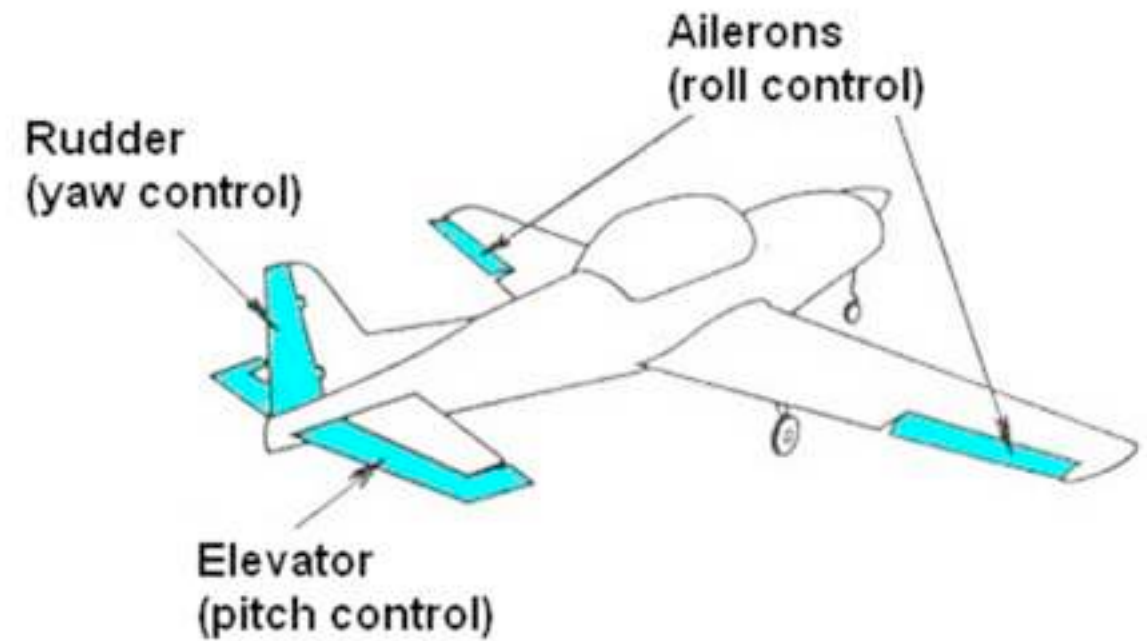
## Couples different parts of the system

- Might inject measurement noise

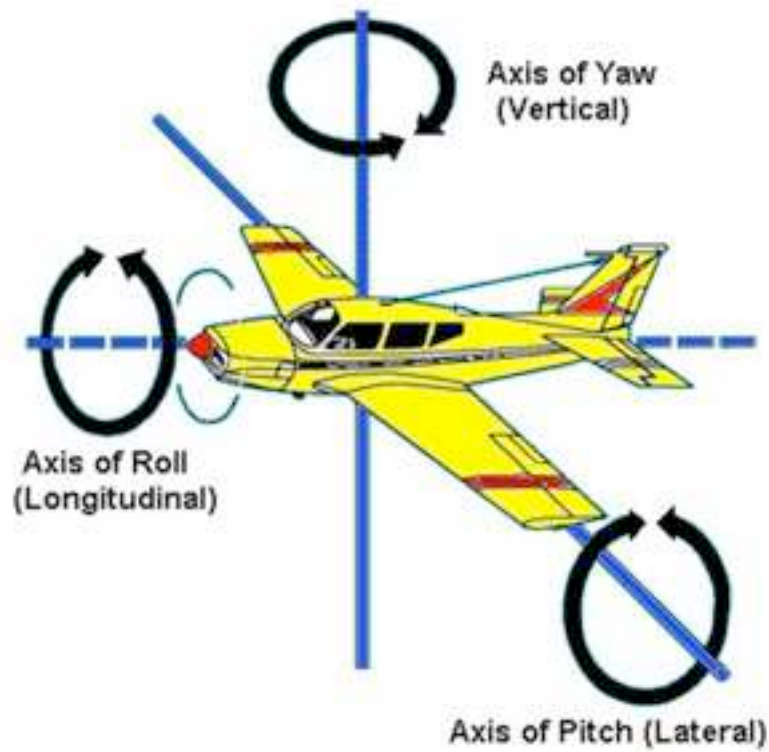
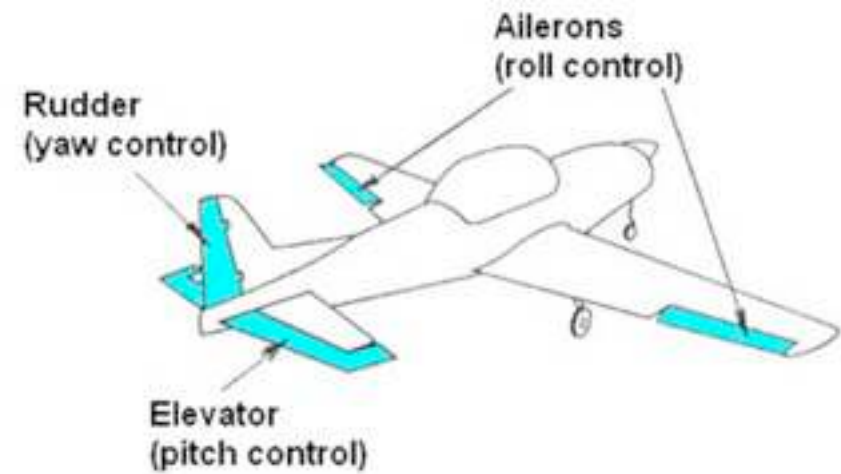
## Complicates manufacture and commissioning

This is why we use feedback ***only if*** there is uncertainty

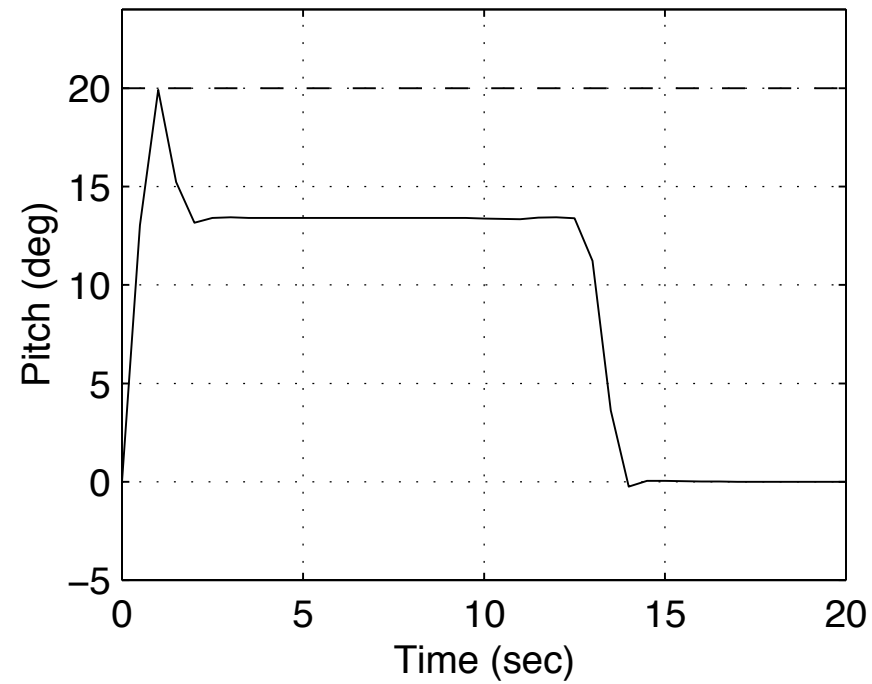
# Example: Cessna Citation



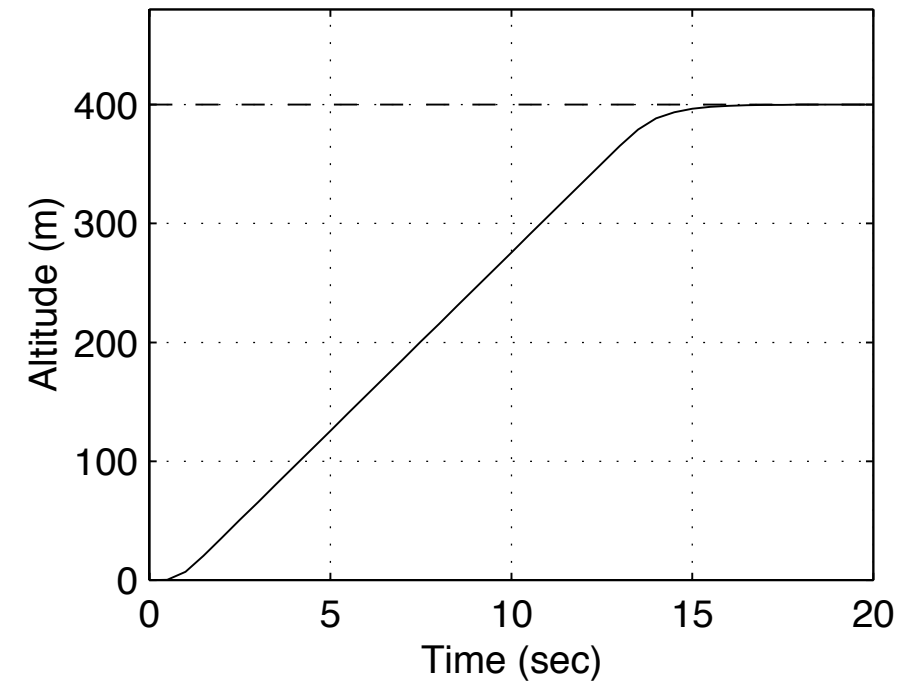
# Example: What, not How



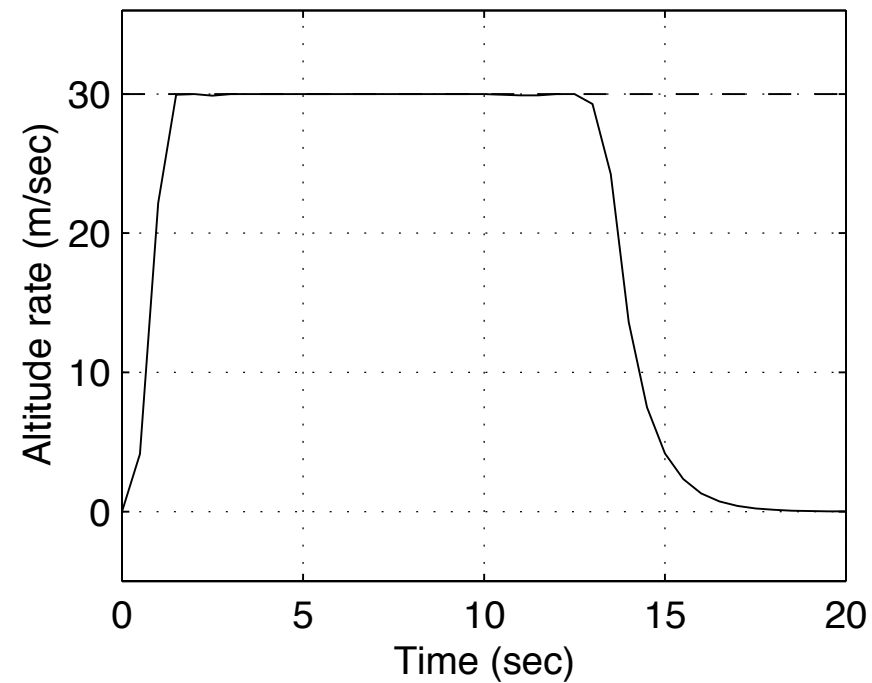
Pitch angle and constraint



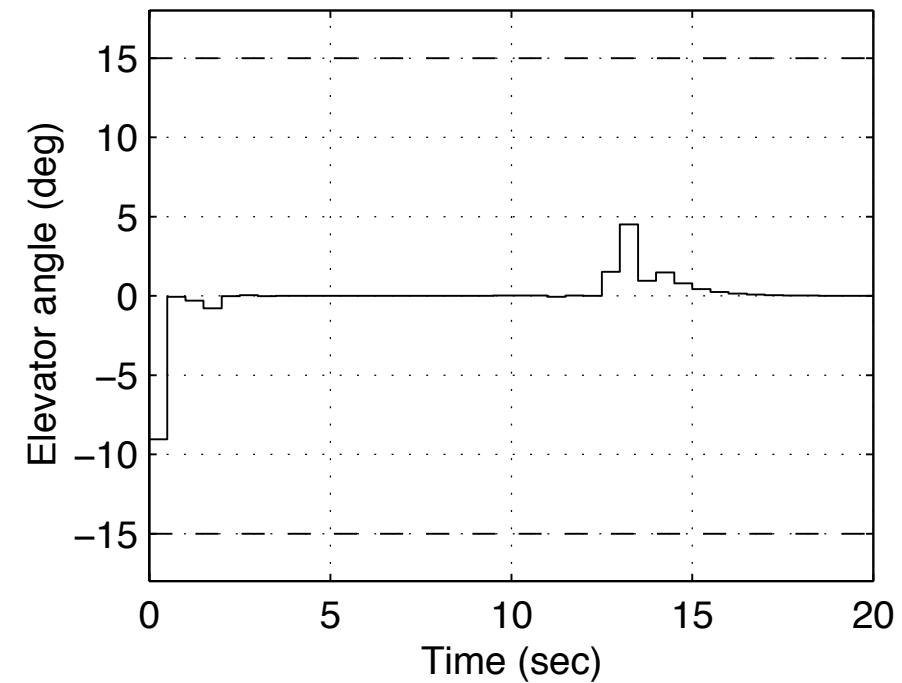
Altitude and set-point



Altitude rate and constraint



Elevator angle and constraint



# Constraints in Control

All physical systems have constraints:

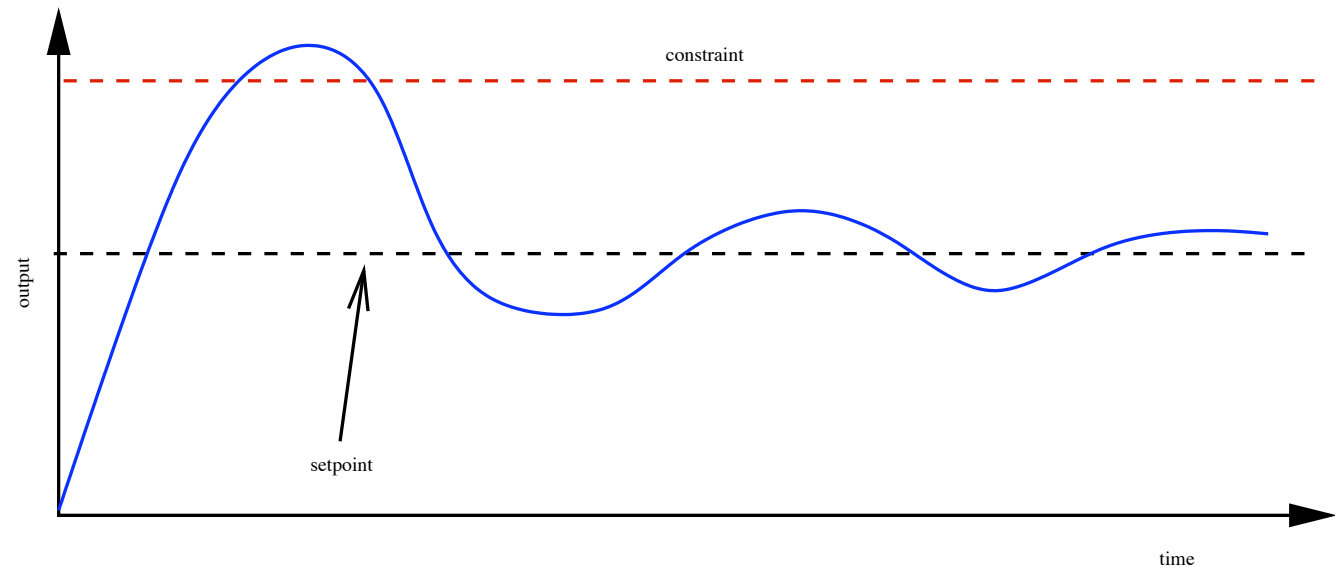
- Physical constraints, e.g. actuator limits
- Safety constraints, e.g. temperature/pressure limits
- Performance constraints, e.g. overshoot

Optimal operating points are often near constraints

Most control methods address constraints a posteriori:

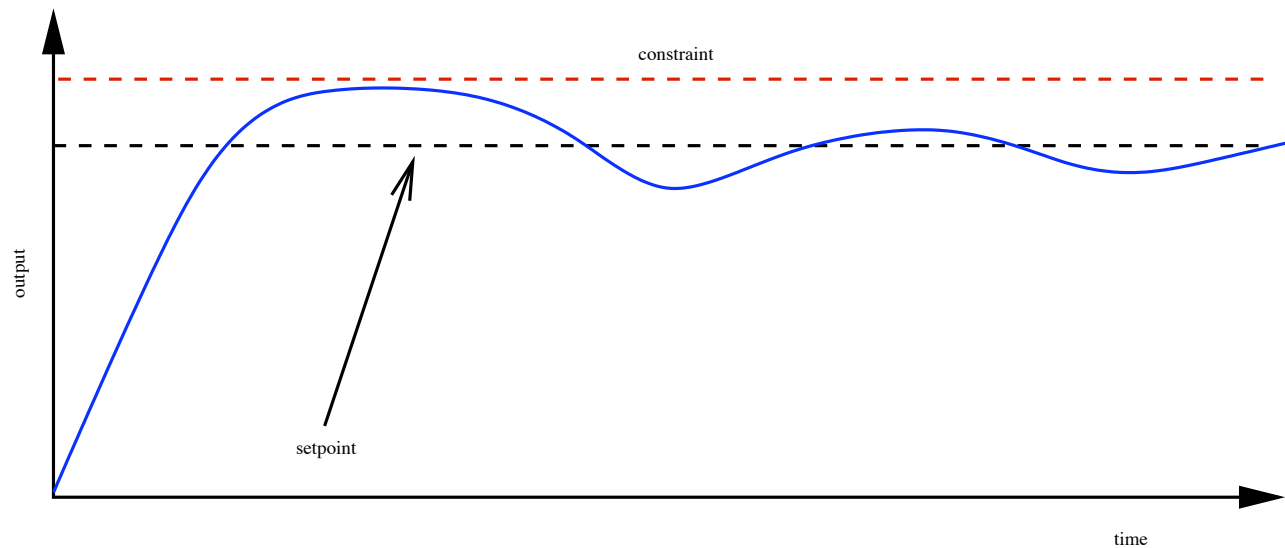
- Anti-windup methods, trial and error

# Optimal Operation and Constraints



## Classical Control

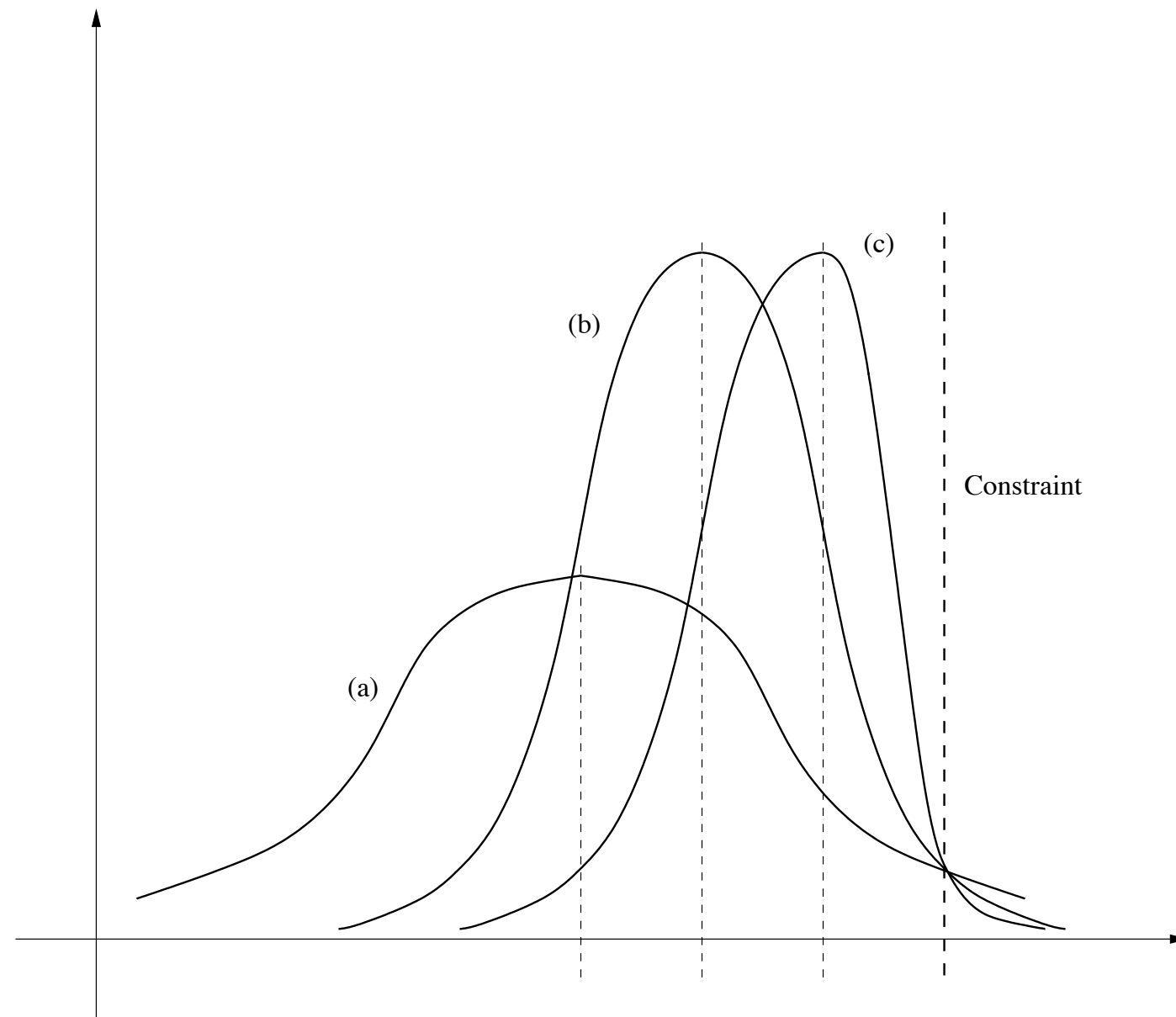
- No knowledge of constraints
- Setpoint far from constraints
- Suboptimal plant operation



## Predictive Control

- Constraints included in design
- Setpoint closer to optimal
- Improved plant operation

# Getting Closer to Constraints



- (a) PID control
- (b) LQG control
- (c) Predictive Control

# Receding Horizon Principle





# Receding Horizon Principle



# Receding Horizon Principle

1. Take measurement



# Receding Horizon Principle



1. Take measurement
2. Solve optimal control problem

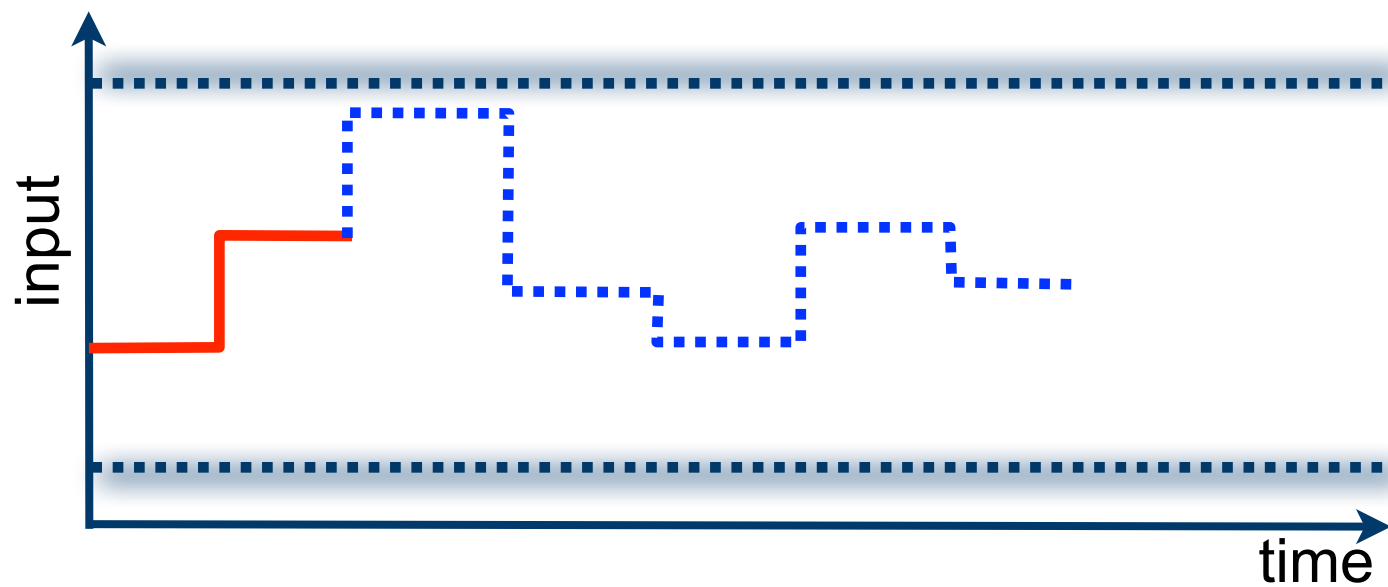
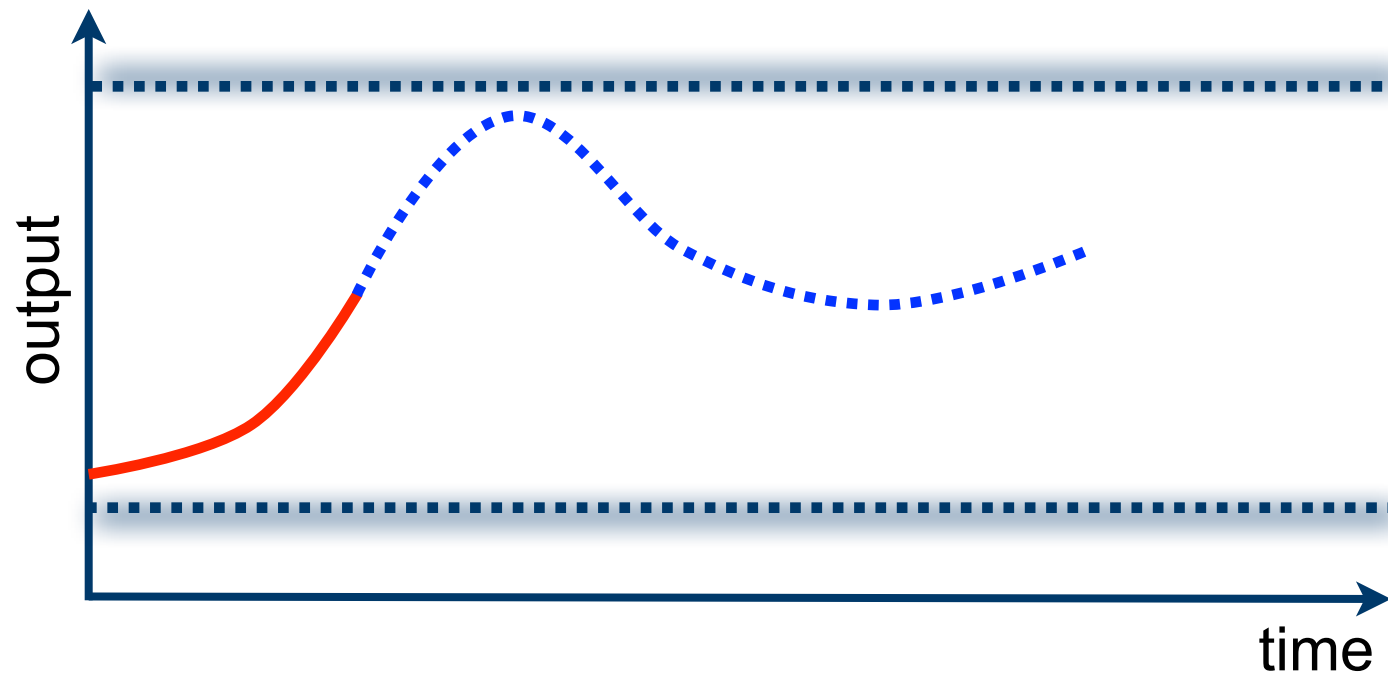
$$\min_{x(\cdot), u(\cdot)} V(x(\cdot), u(\cdot))$$

$$x(0) = \hat{x},$$

$$\dot{x}(t) = f(x(t), u(t))$$

$$c(x(\cdot), u(\cdot)) \leq 0$$

# Receding Horizon Principle



1. Take measurement
2. Solve optimal control problem

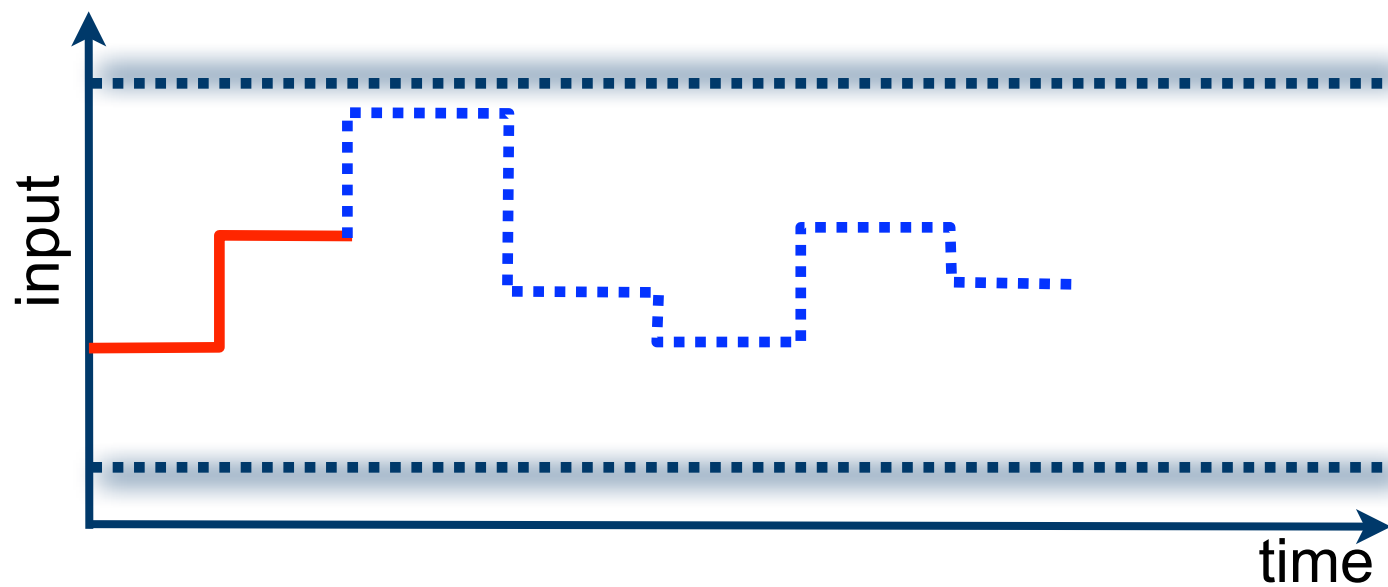
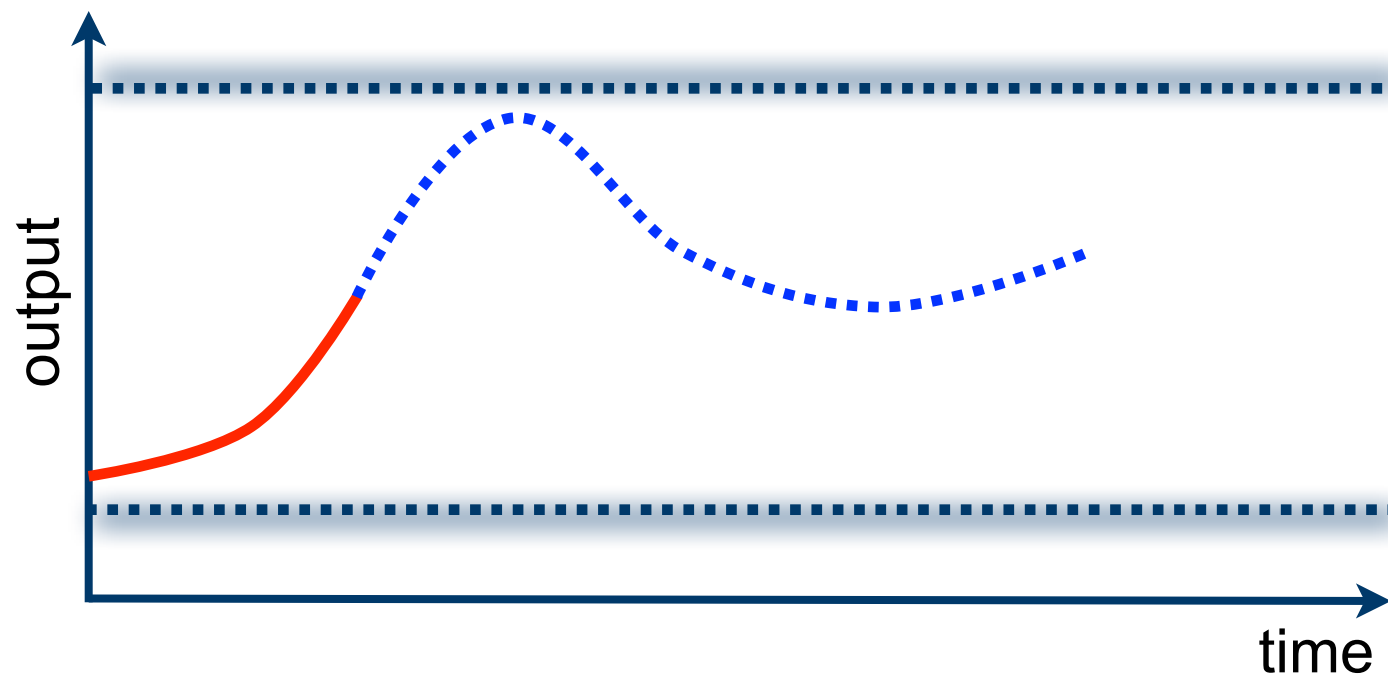
$$\min_{x(\cdot), u(\cdot)} V(x(\cdot), u(\cdot))$$

$$x(0) = \hat{x},$$

$$\dot{x}(t) = f(x(t), u(t))$$

$$c(x(\cdot), u(\cdot)) \leq 0$$

# Receding Horizon Principle



1. Take measurement
2. Solve optimal control problem
3. Implement only first part

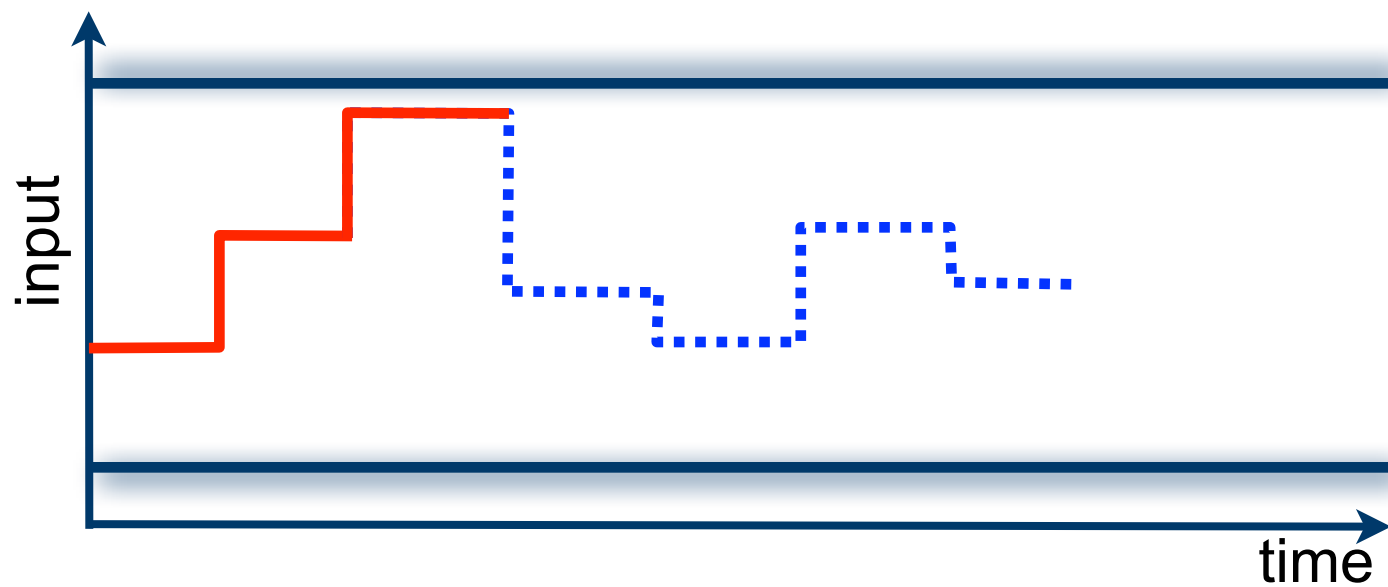
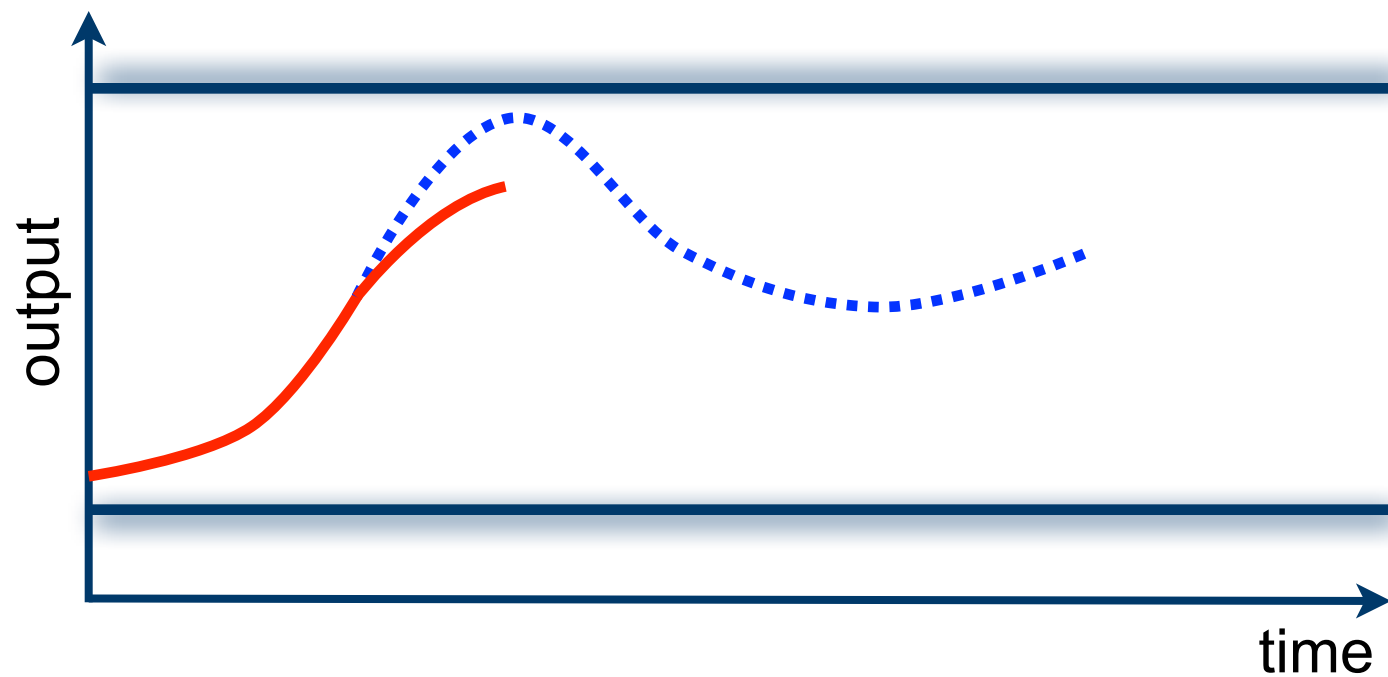
$$\min_{x(\cdot), u(\cdot)} V(x(\cdot), u(\cdot))$$

$$x(0) = \hat{x},$$

$$\dot{x}(t) = f(x(t), u(t))$$

$$c(x(\cdot), u(\cdot)) \leq 0$$

# Receding Horizon Principle



1. Take measurement
2. Solve optimal control problem
3. Implement only first part

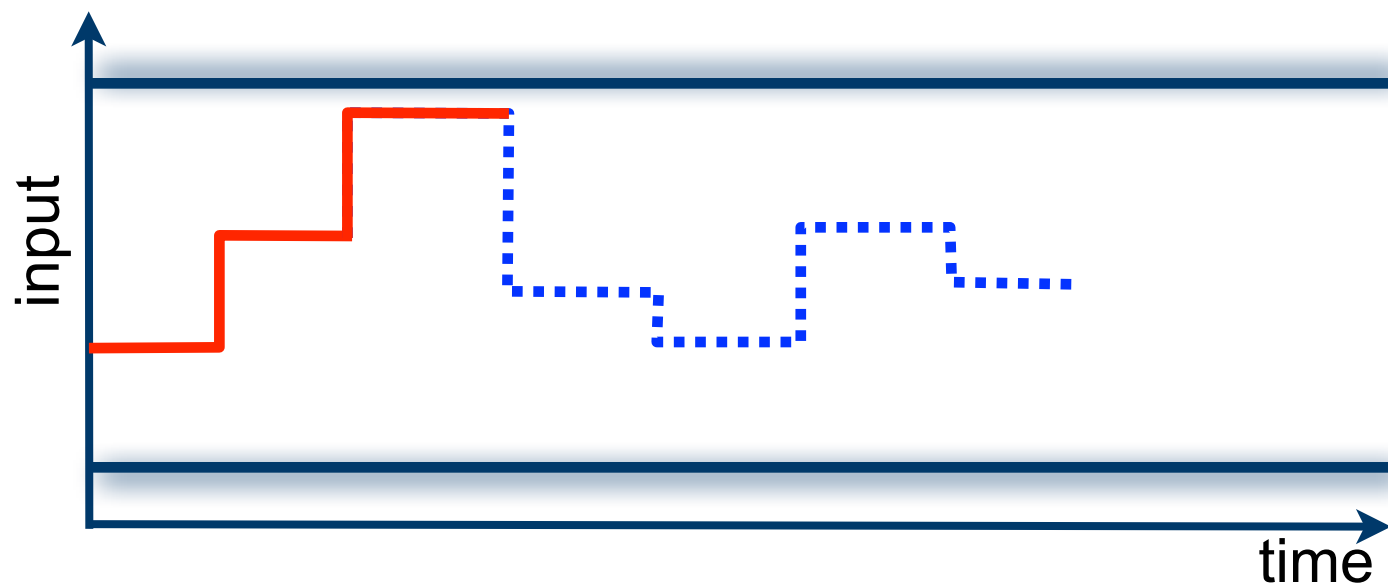
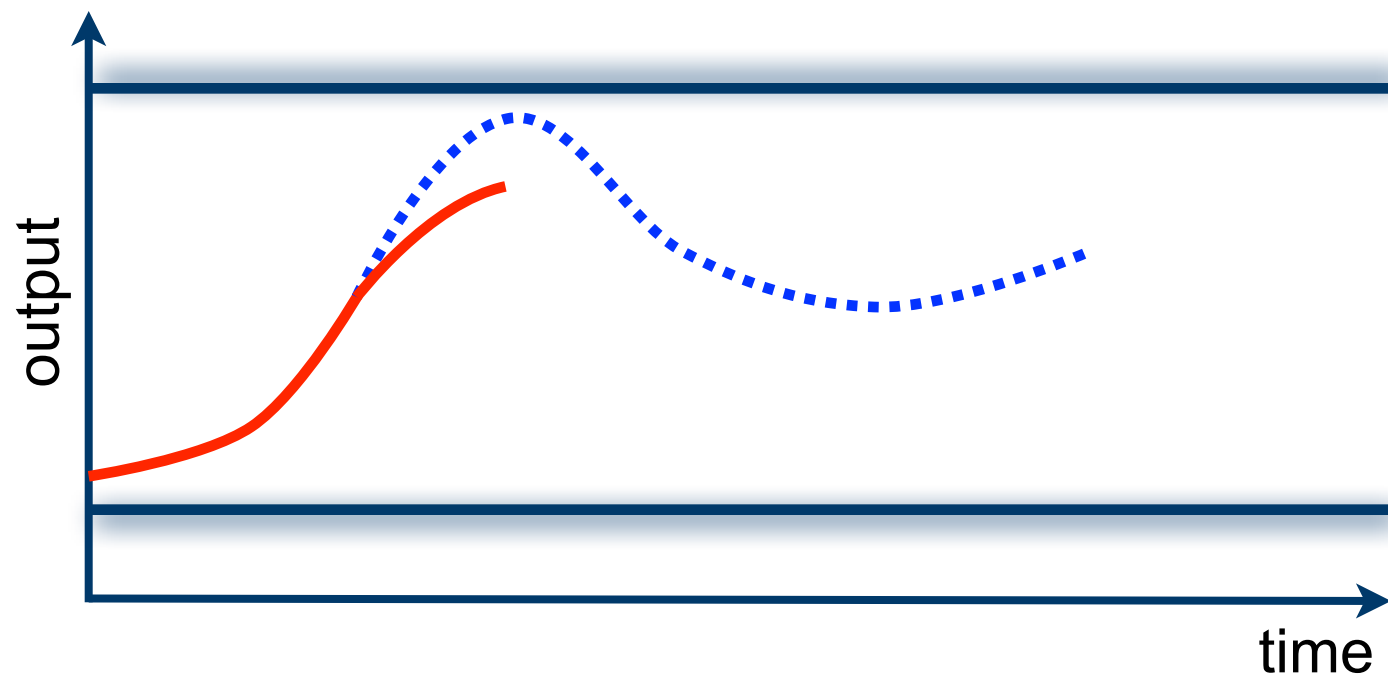
$$\min_{x(\cdot), u(\cdot)} V(x(\cdot), u(\cdot))$$

$$x(0) = \hat{x},$$

$$\dot{x}(t) = f(x(t), u(t))$$

$$c(x(\cdot), u(\cdot)) \leq 0$$

# Receding Horizon Principle



1. Take measurement
2. Solve optimal control problem
3. Implement only first part
4. Go to step 1

$$\min_{x(\cdot), u(\cdot)} V(x(\cdot), u(\cdot))$$

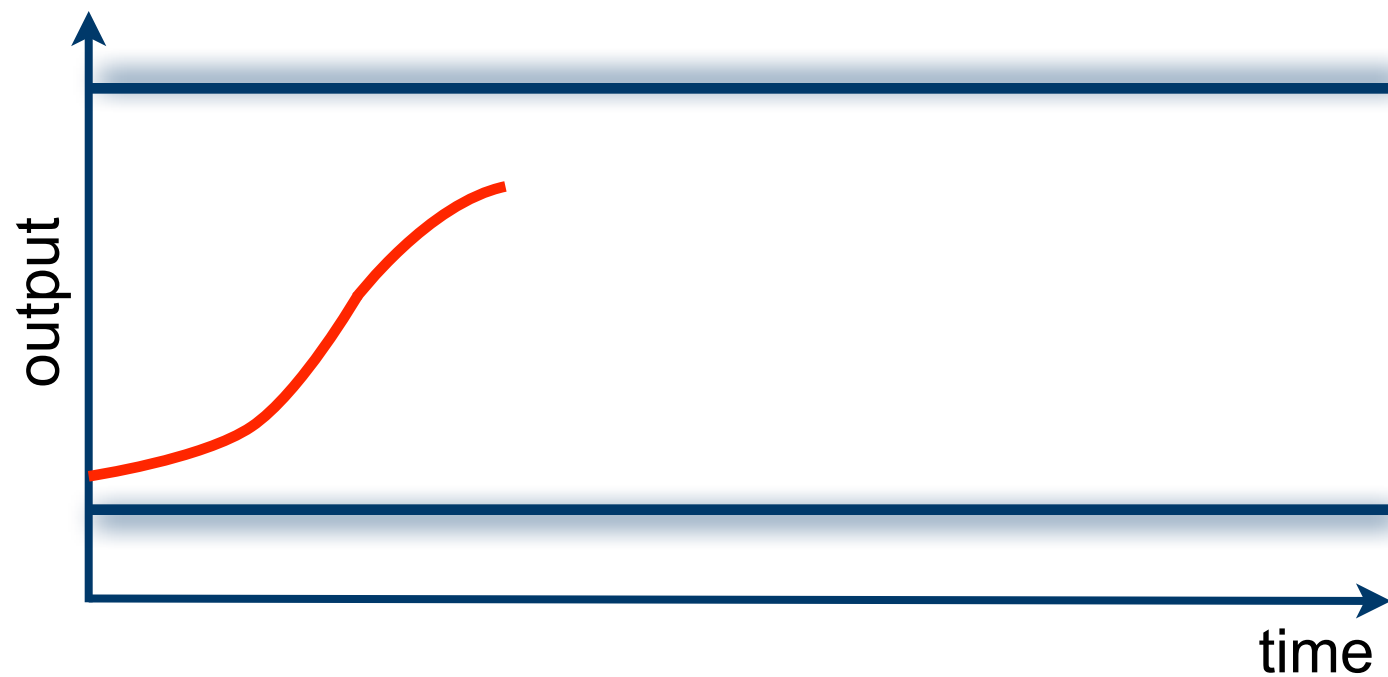
$$x(0) = \hat{x},$$

$$\dot{x}(t) = f(x(t), u(t))$$

$$c(x(\cdot), u(\cdot)) \leq 0$$



# Receding Horizon Principle



1. Take measurement
2. Solve optimal control problem
3. Implement only first part
4. Go to step 1

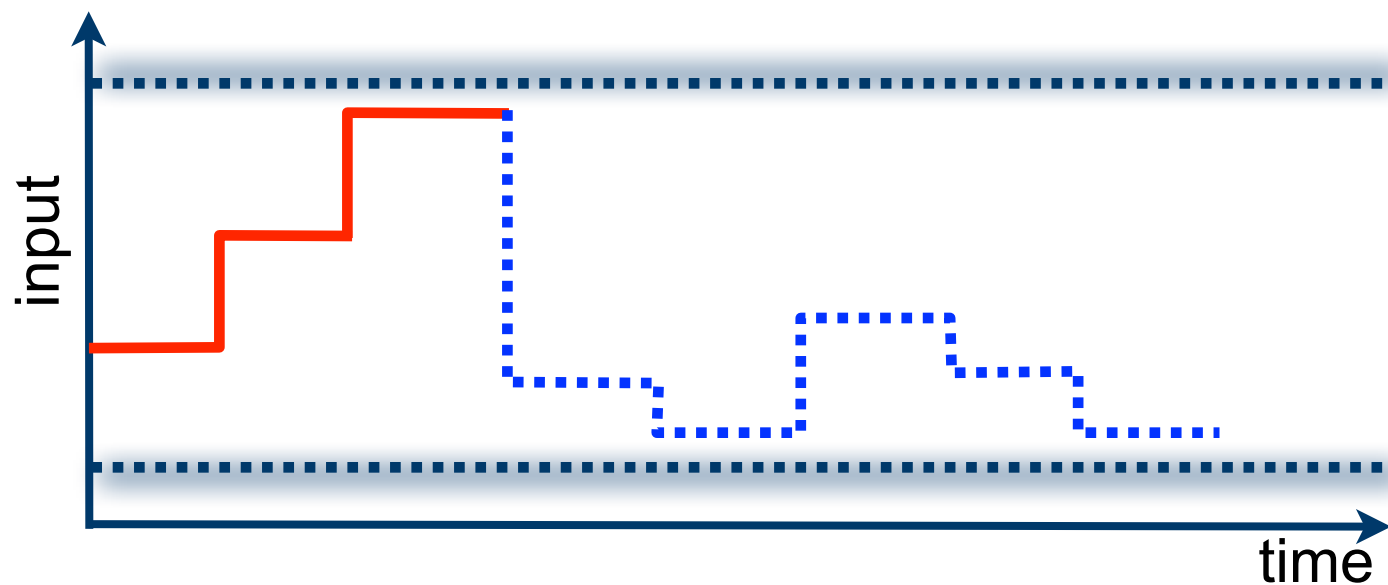
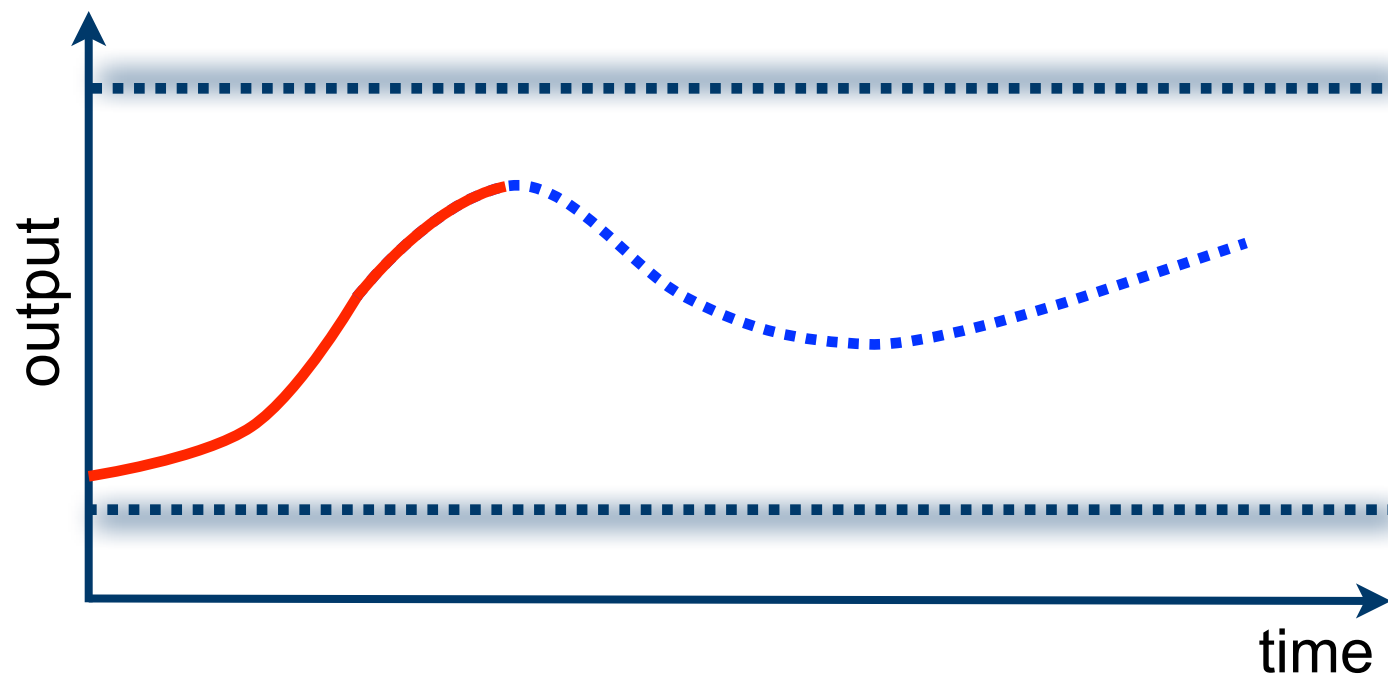
$$\min_{x(\cdot), u(\cdot)} V(x(\cdot), u(\cdot))$$

$$x(0) = \hat{x},$$

$$\dot{x}(t) = f(x(t), u(t))$$

$$c(x(\cdot), u(\cdot)) \leq 0$$

# Receding Horizon Principle



1. Take measurement
2. Solve optimal control problem
3. Implement only first part
4. Go to step 1

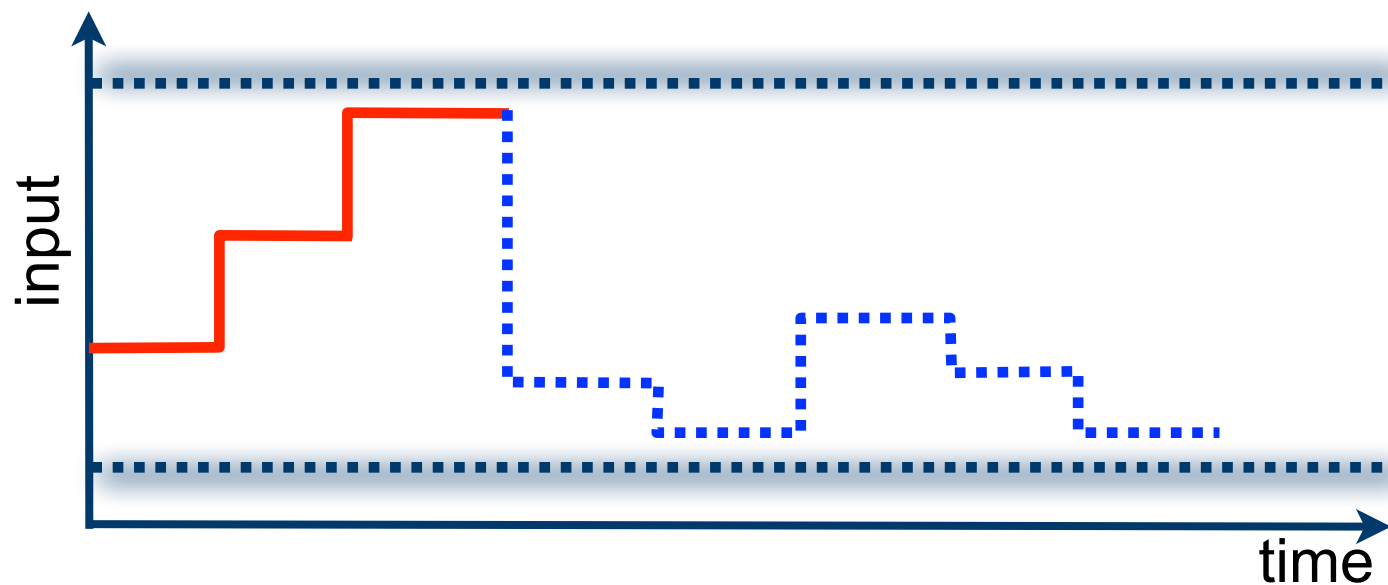
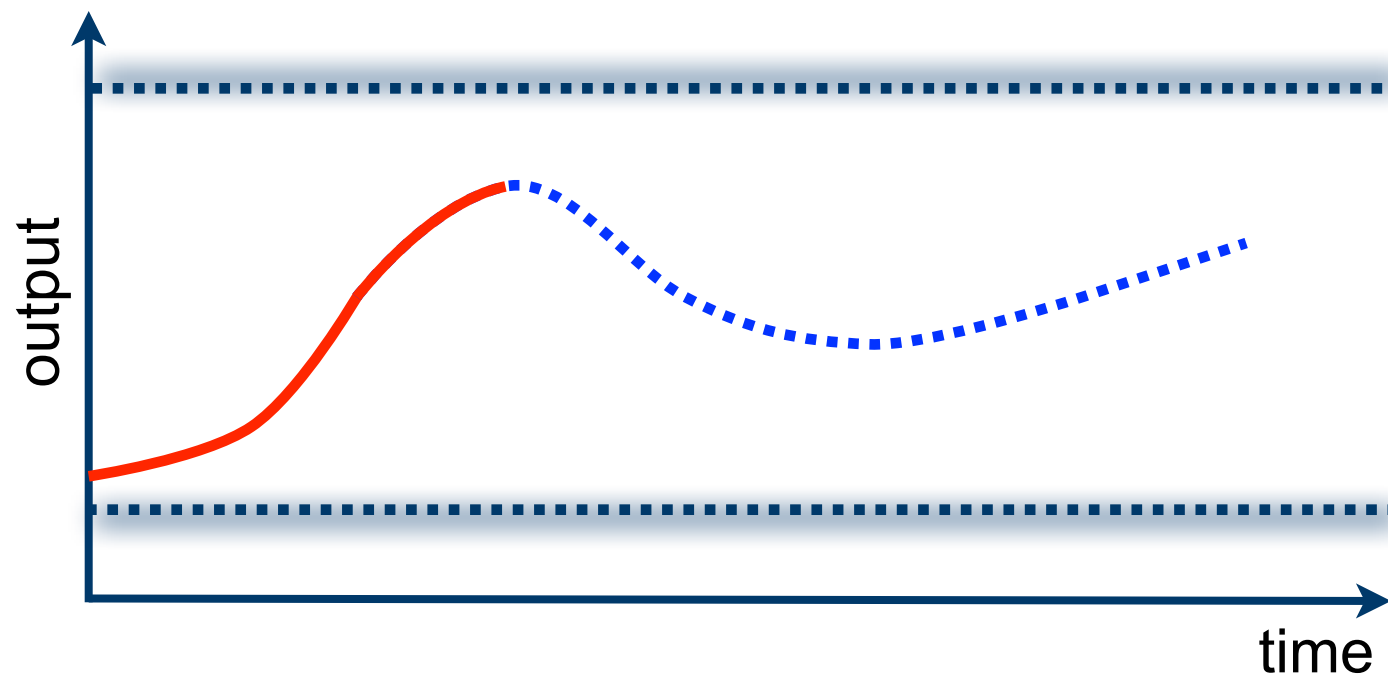
$$\min_{x(\cdot), u(\cdot)} V(x(\cdot), u(\cdot))$$

$$x(0) = \hat{x},$$

$$\dot{x}(t) = f(x(t), u(t))$$

$$c(x(\cdot), u(\cdot)) \leq 0$$

# Receding Horizon Principle



1. Take measurement
2. Solve optimal control problem
3. Implement only first part
4. Go to step 1

$$\min_{x(\cdot), u(\cdot)} V(x(\cdot), u(\cdot))$$

$$x(0) = \hat{x},$$

$$\dot{x}(t) = f(x(t), u(t))$$

$$c(x(\cdot), u(\cdot)) \leq 0$$

Solution is a function of current state, hence we have **feedback**

# Properties of Predictive Control

Is this a 'new' idea?

- Standard finite horizon optimal control
- Since the 1950s (Coates and Noton, 1956)
- Optimization in the loop, in 'real time'
- Repeatedly solving an optimal control problem

The main problems:

- Optimization needs to be fast enough
- The resulting control law might not be stable

The main advantages:

- Systematic method for handling constraints
- Flexible performance specifications
- Easy to understand

# Also Known As...

## Other Names in Industry and Academia:

- Dynamic Matrix Control (DMC)
- Generalised Predictive Control (GPC)
- ...

## Generic names:

- Model Predictive Control (MPC)
- Model-Based Predictive Control (MBPC)

## Predictive Control $\neq$ Receding Horizon Control (RHC)

- Decreasing Horizon Control (DHC)
- Variable Horizon Control (VHC)

# Computational Speed and Applications

Historically, MPC has been used on ‘slow’ processes:

- Petrochemical and process industries, pulp and paper
- Sample time of seconds to hours

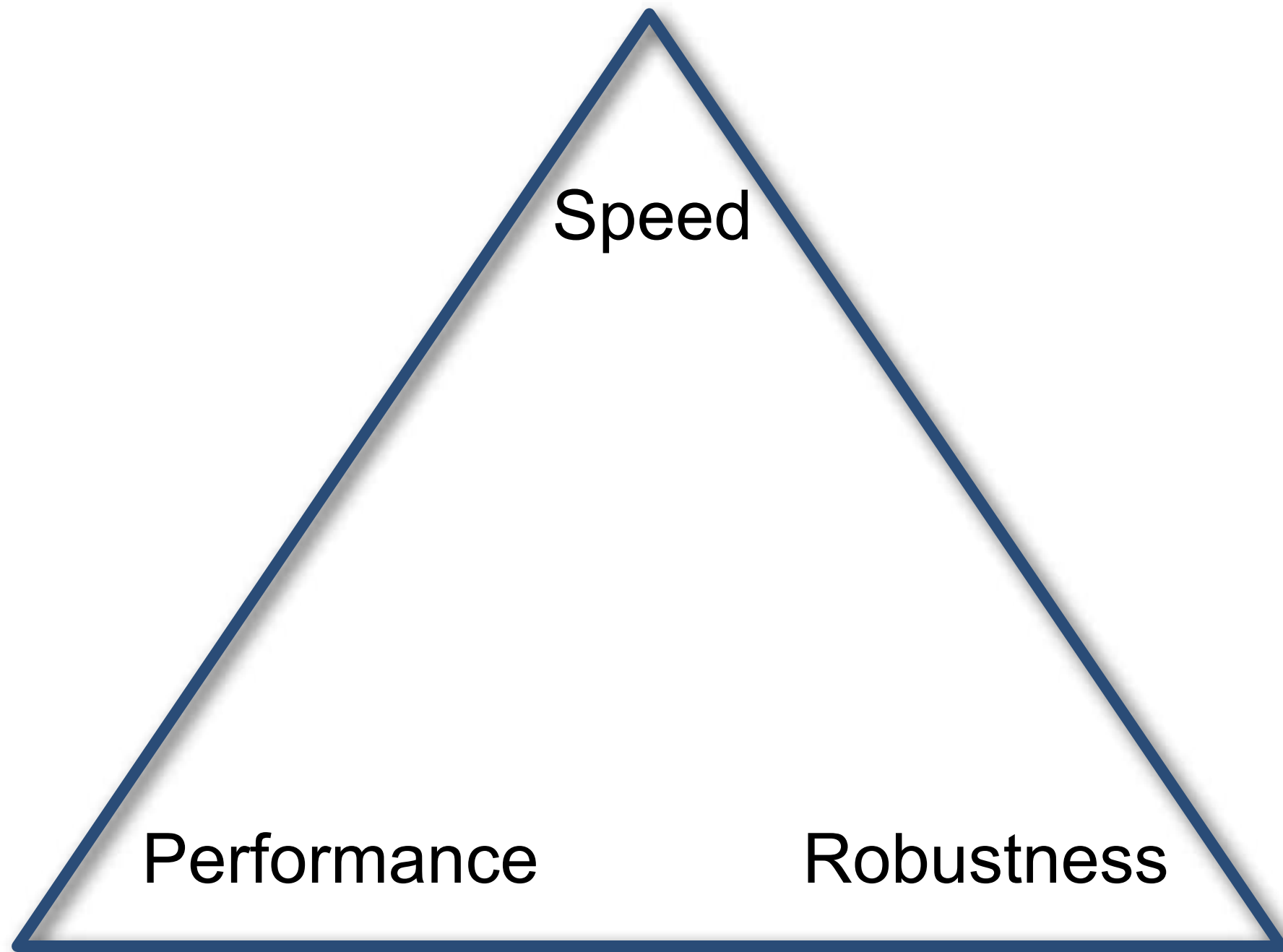
Major advances in hardware and algorithms

- Computation of 1 minute in 1990  $\Rightarrow$  now less than 1s

MPC now being proposed for “fast” processes:

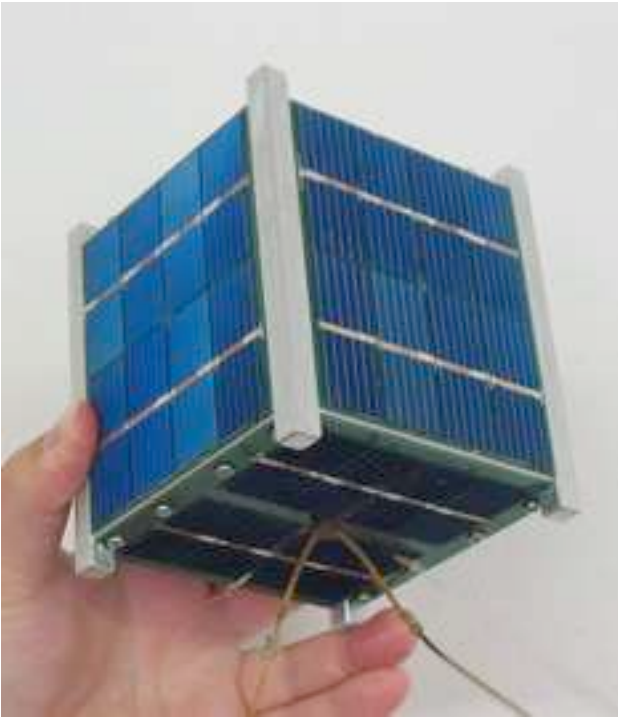
- Automotive traction and engine control
- Aerospace applications
- Autonomous vehicles
- Electricity generation and distribution

# Trade-Offs in MPC

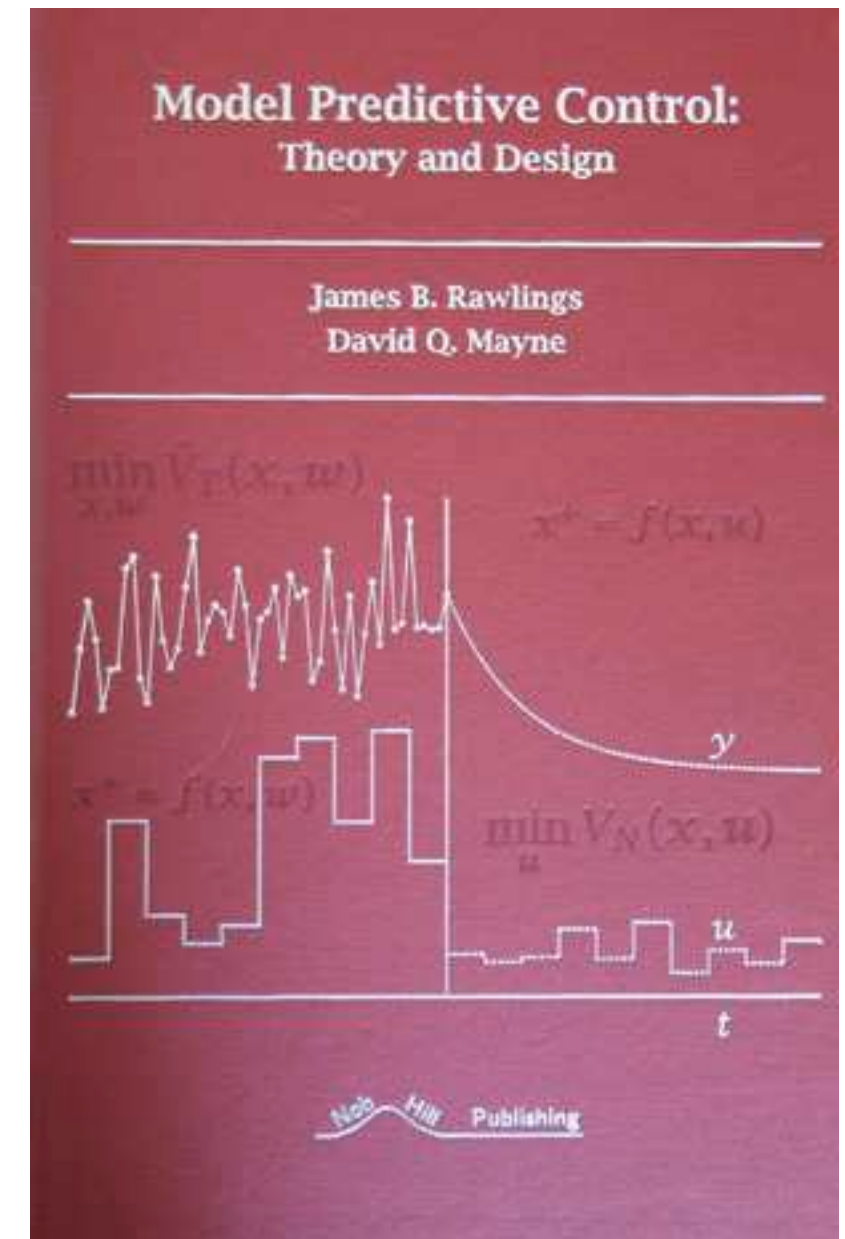
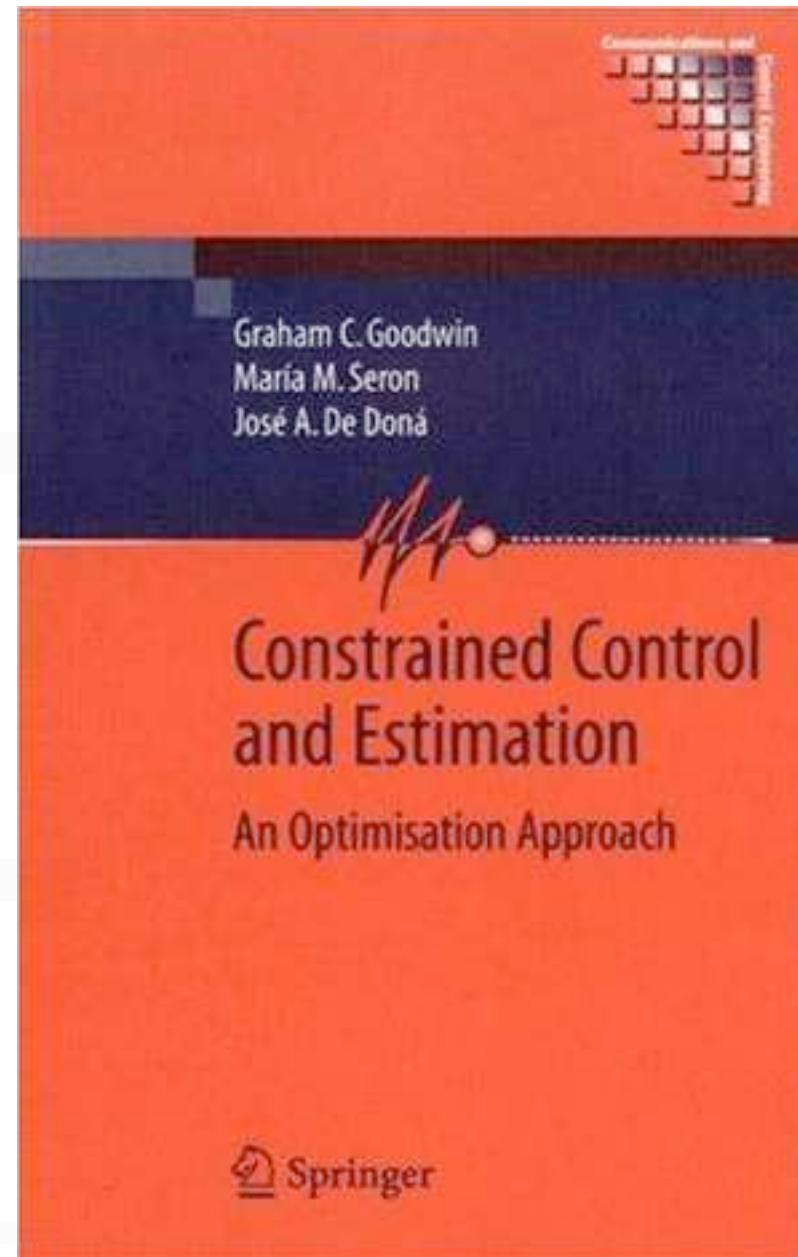
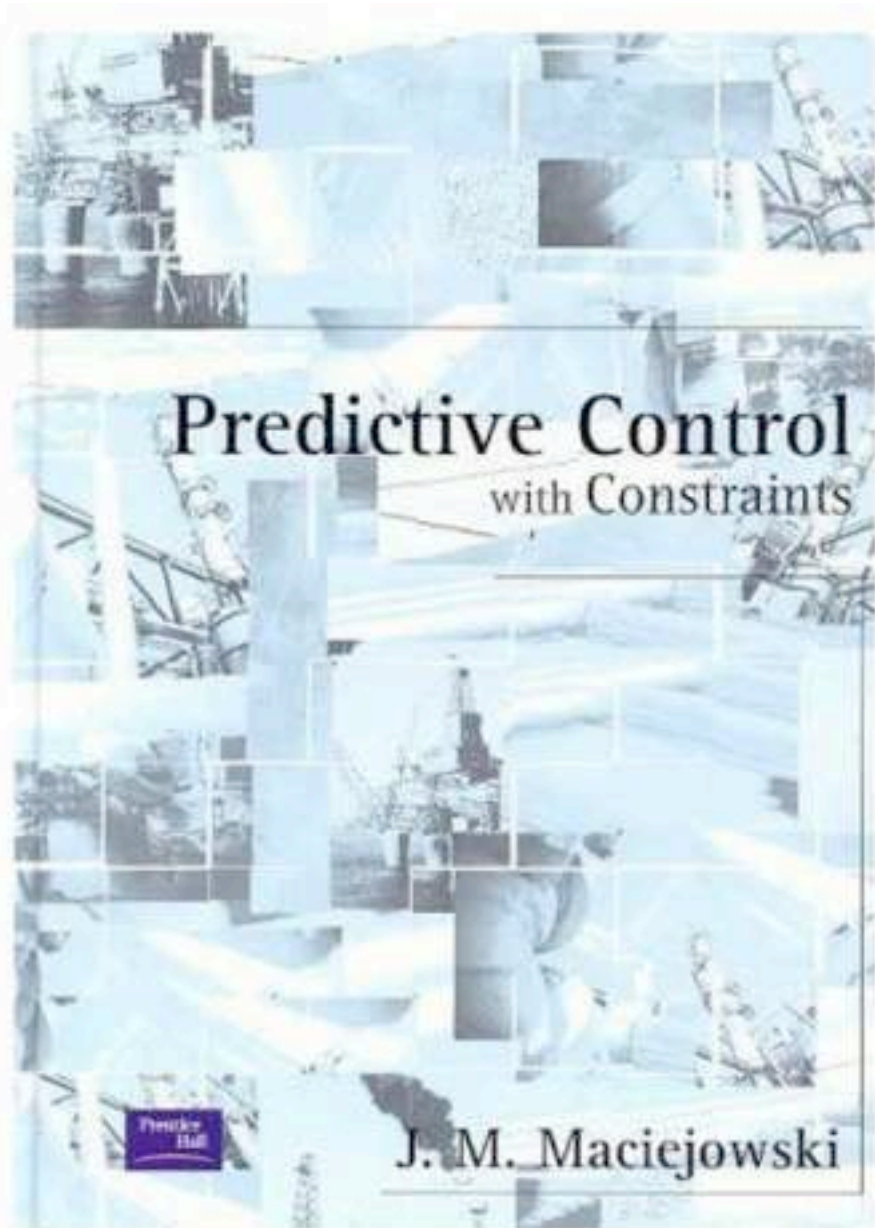




# Future Applications of MPC



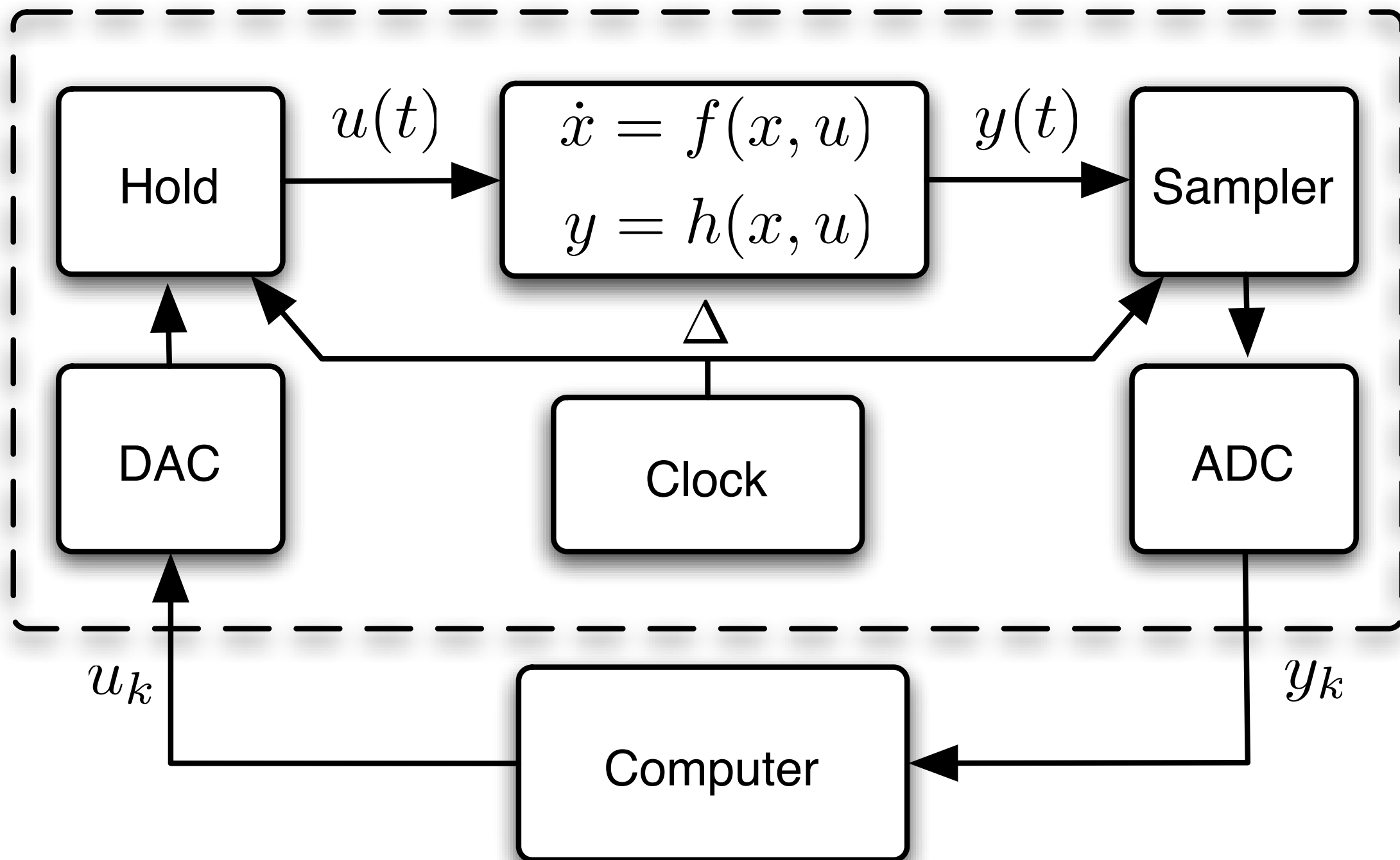
# Books on Predictive Control



# Sampled-data Control

$$x_{k+1} = g(x_k, u_k)$$

$$y_k = h(x_k, u_k)$$





# Continuous-time CLQR

$$\min_{x(\cdot), u(\cdot)} x(T)' P_c x(T) + \int_0^T x(t)' Q_c x(t) + u(t)' R_c u(t) dt$$

$$x(0) = \hat{x}, \quad \dot{x}(t) = A_c x(t) + B_c u(t), \quad \forall t \in (0, T)$$

$$E_c x(t) + J_c u(t) \leq c_c, \quad \forall t \in [0, T)$$

$$x(\cdot), u(\cdot) \in \mathcal{L}_2[0, T]$$

Standing assumptions:

- Solution exists
- Convex terminal and stage cost

# Sampled-data CLQR

$$\min_{x(\cdot), u(\cdot)} x(T)' P_c x(T) + \int_0^T x(t)' Q_c x(t) + u(t)' R_c u(t) dt$$

$$x(0) = \hat{x}, \quad \dot{x}(t) = A_c x(t) + B_c u(t), \quad \forall t \in (0, T)$$

$$E_\Delta x(t) + J_\Delta u(t) \leq c_\Delta, \quad \forall t \in \{0, \Delta, 2\Delta, \dots, (N-1)\Delta\}$$

Zero-order hold (ZOH) constraint commonly employed:

$$u(t) = u(k\Delta), \quad \forall t \in [k\Delta, (k+1)\Delta), \quad k = 0, 1, \dots, N-1$$

Conditions for convergence to continuous-time solution as sample period tends to zero:

J. Yuz, G. Goodwin, A. Feuer, and J. De Doná, "Control of constrained linear systems using fast sampling rates," *Systems & Control Letters*, vol. 54, no. 10, pp. 981–990, 2005.

# Equivalent Discrete-time CLQR

$$\min_{\substack{u_0, u_1, \dots, u_{N-1} \\ x_0, x_1, x_2, \dots, x_N}} \frac{1}{2} x'_N P x_N + \sum_{k=0}^{N-1} \frac{1}{2} (x'_k Q x_k + u'_k R u_k + 2x'_k S u_k)$$

$$x_0 = \hat{x}, \quad x_{k+1} = A x_k + B u_k, \quad k = 0, 1, \dots, N - 1$$

$$E x_k + J u_k \leq c, \quad k = 0, 1, \dots, N - 1$$

Computing various matrices (all functions of sample period, in general):

- Inside Matlab's `c2d` and `lqrd` function (*not* `dlqr`)
- K. Astrom and B. Wittenmark, *Computer-Controlled Systems*, 3rd ed., Prentice-Hall, 1997.
- C. Van Loan, "Computing integrals involving the matrix exponential", *IEEE Trans. Automatic Control*, vol. 23, no. 3, pp. 395–404, 1978.
- J. Yuz, G. Goodwin, A. Feuer, and J. De Doná, "Control of constrained linear systems using fast sampling rates," *Systems & Control Letters*, vol. 54, no. 10, pp. 981–990, 2005.
- G. Pannocchia, J. B. Rawlings, D. Q. Mayne, and W. Marquardt, "On computing solutions to the continuous time constrained linear quadratic regulator," *IEEE Trans. Automatic Control*, vol. 55, no. 9, pp. 2192–2198, 2010.

# Discrete Minimum Principle = KKT for QP

Pearson & Sridhar, IEEE Trans. Automat. Control, 1966:  
Stage cost and terminal cost *convex*, dynamics *linear* and  
inequality constraints *convex*, so **discrete minimum principle**  
gives *necessary and sufficient* conditions for the solution:

$$A' \nu_{k+1} = Qx_k + Su_k + \nu_k + E' \lambda_k, \quad k = 1, \dots, N - 1$$

$$\nu_N + Px_N = 0,$$

$$B' \nu_{k+1} = J' \lambda_k + Ru_k + S' x_k, \quad k = 0, \dots, N - 1$$

$$x_0 = \hat{x}, \quad x_{k+1} = Ax_k + Bu_k, \quad k = 0, \dots, N - 1$$

$$\lambda_k \geq 0, \quad Ex_k + Ju_k \leq c, \quad k = 0, \dots, N - 1$$

$$\text{diag}(\lambda_k)(Ex_k + Ju_k - c) = 0, \quad k = 0, \dots, N - 1$$

Same as **KKT** conditions for a suitably-defined quadratic  
program (**QP**), where the  $\nu_k$ ,  $\lambda_k$  are the Lagrange multipliers.



# Quadratic Program (QP)

$$\begin{aligned} \min_{\theta} \quad & \frac{1}{2} \theta' H \theta + h' \theta \\ & F \theta = f \\ & G \theta \leq g \end{aligned}$$

Matlab: QUADPROG

Octave: QP

Scilab: QPSOLVE

Free: OOQP, BPMPD, CLP, QPC, qpOASES, CVX/CVXGEN

Other: CPLEX, MOSEK, NAG, QPOPT, TOMLAB, XPRESS

Free SOCP/SDP solvers (also solves QPs): SDPT3, SEDUMI, SDPA, SDPNAL, CSDP, DSDP

Interfaces/modelling: YALMIP, AMPL, CVX/CVXGEN, TOMLAB

# Solving the CLQR Problem as a QP

Optimal input sequence is nonlinear function of current state

Formulate as a multi-parametric quadratic program (QP):

- Off-line: Compute solution as a the piecewise-affine expression for all possible values of state
- On-line: Implement solution as lookup table
- Limited to easy/small problems
- Bemporad et al. (Automatica, 2002)
- Multi-Parametric Toolbox (MPT), <http://control.ee.ethz.ch/~mpt/>

Philosophy of Predictive Control:

- Off-line: Don't compute explicit expression for solution
- On-line: Solve a QP in 'real-time' at each sample instant
- Gives *exactly* the same input sequence as explicit expression
- Faster for complex/medium/large problems
- More flexible than computing explicit expression

# Formulating CLQR Problem as a QP

Non-condensed

$$\theta := \begin{bmatrix} x_0 \\ u_0 \\ x_1 \\ u_1 \\ \vdots \\ u_{N-1} \\ x_N \end{bmatrix}$$

$$\Rightarrow \min_{\theta} \frac{1}{2} \theta' H \theta$$

$$F \theta = f(\hat{x})$$

$$G \theta \leq g$$

Condensed

$$\theta := \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix}$$

$$\Rightarrow \min_{\theta} \frac{1}{2} \theta' H \theta + h(\hat{x})' \theta$$

$$G \theta \leq g(\hat{x})$$

# Non-condensed Formulation

$$\theta := \begin{bmatrix} x_0 \\ u_0 \\ x_1 \\ u_1 \\ \vdots \\ u_{N-1} \\ x_N \end{bmatrix} \Rightarrow \begin{aligned} \min_{\theta} & \frac{1}{2} \theta' H \theta \\ & F \theta = f(\hat{x}) \\ & G \theta \leq g \end{aligned}$$

## **NB (why formulation and choice of QP solver matters):**

- Matrices  $(F, G, H)$  are *sparse and banded*.

Use a QP solver that exploits sparsity and bandedness:

- SJ Wright, *Parallel Computing*, 1990
- SJ Wright, *J. Optimization Theory Applications*, 1993

Tailor-made QP solvers can do better than general-purpose QP solvers:

- CV Rao, SJ Wright and JB Rawlings, *J. Optimization Theory Applications*, 1998

# Why Order of Unknowns Matters

$$\begin{array}{ccc} & \text{YES} & \\ & & \text{NO} \\ \theta := & \begin{bmatrix} x_0 \\ u_0 \\ x_1 \\ u_1 \\ \vdots \\ u_{N-1} \\ x_N \end{bmatrix} & \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_N \\ u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix} \end{array}$$

QP solvers rely on linear algebra solvers, which do not always perform well at detecting that re-ordering rows and/or columns results in a banded matrix.

# Matlab Example: Why Order Matters

```
>> n = 1e7;  
>> b1 = rand(n,1);  
>> permrows = randperm(n);  
>> b2 = b1(permrows); % randomly permute rows of b1  
>> A1 = speye(n); % identity matrix stored as sparse matrix  
>> A2 = A1(permrows,:); % randomly permute rows of A1  
  
>> % solve linear system A1*x1 = b1, with A1 diagonal  
>> tic; x1 = A1\b1; toc  
Elapsed time is 0.230493 seconds.  
  
>> % solve same linear system, but with rows permuted  
>> tic; x2 = A2\b2; toc  
Elapsed time is 9.231288 seconds.
```



# Using the Kronecker Product

$$M \in \mathbb{R}^{m \times n}$$

$$I_N \otimes M = \begin{bmatrix} M & & \\ & \ddots & \\ & & M \end{bmatrix} \in \mathbb{R}^{Nm \times Nn}$$

$$\mathbf{1}_N \otimes M = \begin{bmatrix} M \\ \vdots \\ M \end{bmatrix} \in \mathbb{R}^{Nm \times n}$$

$\mathbf{1}_N$  column vector of  $N$  ones



# Inequality Constraints (Non-condensed)

$$Ex_k + Ju_k \leq c, \quad k = 0, 1, \dots, N - 1$$

$\Leftrightarrow$

$$\begin{bmatrix} I_N \otimes [E \ J] & 0 \end{bmatrix} \begin{bmatrix} x_0 \\ u_0 \\ x_1 \\ u_1 \\ \vdots \\ u_{N-1} \\ x_N \end{bmatrix} \leq \mathbf{1}_N \otimes c$$

$\mathbf{1}_N$  column vector of  $N$  ones

# Cost Function (Non-condensed)

$$\begin{aligned}
 & \frac{1}{2} x'_N P x_N + \sum_{k=0}^{N-1} \frac{1}{2} (x'_k Q x_k + u'_k R u_k + 2x'_k S u_k) \\
 &= \frac{1}{2} \begin{bmatrix} x_0 \\ u_0 \\ x_1 \\ u_1 \\ \vdots \\ x_{N-1} \\ u_{N-1} \\ x_N \end{bmatrix}' \begin{bmatrix} I_N \otimes \begin{bmatrix} Q & S \\ S' & R \end{bmatrix} & 0 \\ & P \end{bmatrix} \begin{bmatrix} x_0 \\ u_0 \\ x_1 \\ u_1 \\ \vdots \\ x_{N-1} \\ u_{N-1} \\ x_N \end{bmatrix}
 \end{aligned}$$

# Homework: Non-condensed Formulation

$$\theta := \begin{bmatrix} x_0 \\ u_0 \\ x_1 \\ u_1 \\ \vdots \\ u_{N-1} \\ x_N \end{bmatrix} \Rightarrow \begin{aligned} \min_{\theta} & \frac{1}{2} \theta' H \theta \\ & F \theta = f(\hat{x}) \\ & G \theta \leq g \end{aligned}$$

Write Matlab/Octave/Scilab code that uses the previous three slides to compute the QP matrices  $(F, G, H)$  and vectors  $(f(\hat{x}), g)$  *without* using any for- or while-loops in your code.

*Hint:* Rewrite the matrix in the equality constraints using Kronecker products.

# Condensed Formulation

$$\theta := \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix} \Rightarrow \begin{aligned} \min_{\theta} & \frac{1}{2} \theta' H \theta + h(\hat{x})' \theta \\ & G \theta \leq g(\hat{x}) \end{aligned}$$

## **NB (why smaller is not always better):**

- Matrices  $(G, H)$  are *not* sparse and banded, in general, hence not always faster to solve than non-condensed case, even though less unknowns.
- Numerically ill-conditioned QP if dynamic system is unstable.

For a suitable change of variables, which introduces sparsity and bandedness and allows one to work with unstable systems, see:

- JL Jerez, EC Kerrigan and GA Constantinides. A Condensed and Sparse QP Formulation for Predictive Control, *Proc. IEEE CDC-ECC*, Dec. 2011

# Inequality Constraints (Condensed)

$$Ex_k + Ju_k \leq c, \quad k = 0, 1, \dots, N - 1$$

$\Leftrightarrow$

$$[I_N \otimes E \ 0] \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_N \end{bmatrix} + [I_N \otimes J] \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix} \leq \mathbf{1}_N \otimes c$$

$\mathbf{1}_N$  column vector of  $N$  ones

# Cost Function (Condensed)

$$\begin{aligned}
 & \frac{1}{2} x'_N P x_N + \sum_{k=0}^{N-1} \frac{1}{2} (x'_k Q x_k + u'_k R u_k + 2x'_k S u_k) \\
 &= \frac{1}{2} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_N \end{bmatrix}' \begin{bmatrix} I_N \otimes Q & 0 \\ 0 & P \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_N \end{bmatrix} + \frac{1}{2} \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix}' I_N \otimes R \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix} \\
 &+ \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_N \end{bmatrix}' \begin{bmatrix} I_N \otimes S \\ 0 \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix}
 \end{aligned}$$

# Dynamic System Equations (Condensed)

$$x_0 = \hat{x}, \quad x_{k+1} = Ax_k + Bu_k, \quad k = 0, 1, \dots, N-1$$

$$\Leftrightarrow$$

$$\begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_N \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ I_N \otimes A & 0 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_N \end{bmatrix} + \begin{bmatrix} 0 \\ I_N \otimes B \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix} + \begin{bmatrix} \hat{x} \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

$$\Leftrightarrow$$

$$\begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_N \end{bmatrix} = \left[ I - \begin{bmatrix} 0 & 0 \\ I_N \otimes A & 0 \end{bmatrix} \right]^{-1} \begin{bmatrix} 0 \\ I_N \otimes B \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix} + \left[ I - \begin{bmatrix} 0 & 0 \\ I_N \otimes A & 0 \end{bmatrix} \right]^{-1} \begin{bmatrix} I \\ 0 \\ \vdots \\ 0 \end{bmatrix} \hat{x}$$

# Homework: Condensed Formulation

$$\theta := \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix} \Rightarrow \begin{aligned} \min_{\theta} & \frac{1}{2} \theta' H \theta + h(\hat{x})' \theta \\ & G \theta \leq g(\hat{x}) \end{aligned}$$

Write Matlab/Octave/Scilab code that uses the previous three slides to compute the QP matrices  $(G, H)$  and vectors  $(g(\hat{x}), h(\hat{x}))$  *without* using any for- or while-loops in your code, or computing inverses of matrices.

*Hint:* Start by computing the matrices in the expression for the state sequence, which is a linear function of the input sequence and current state  $\hat{x}$ , and substituting the expression for the state sequence into the expressions for the cost and inequality constraints.



# Algorithms for Solving a Convex QP

$$\min_{\theta} \frac{1}{2} \theta' H \theta + h' \theta$$

$$F \theta = f$$

$$G \theta \leq g$$

Assume convex QP, i.e.  $H \geq 0$

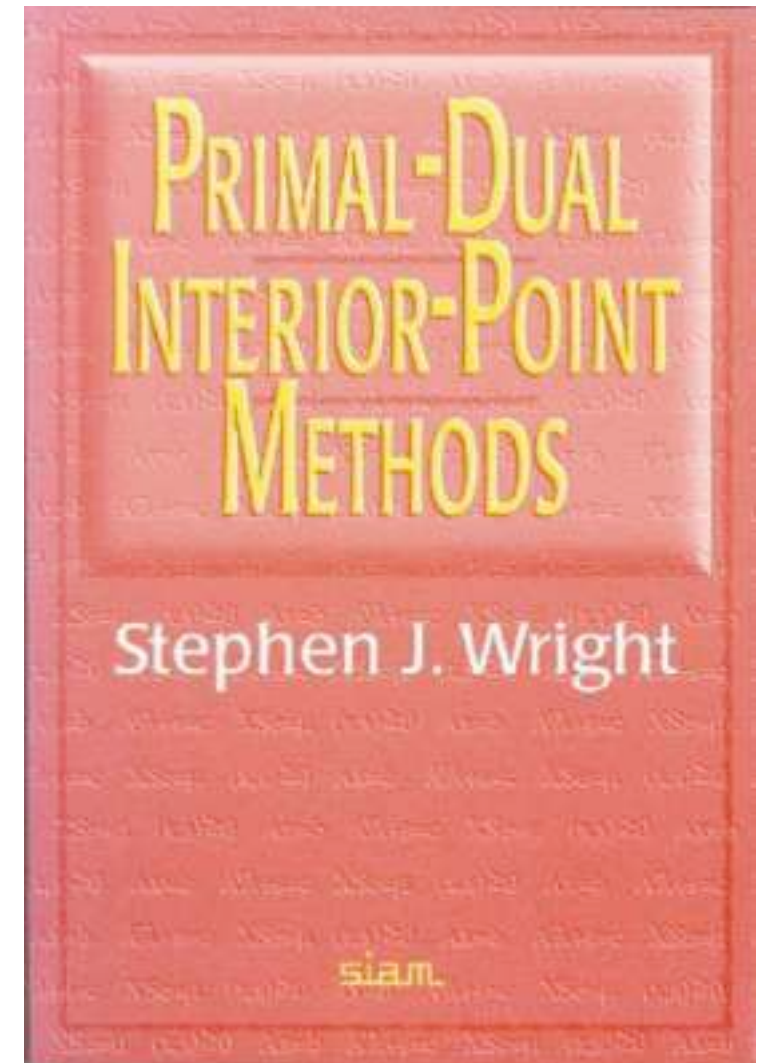
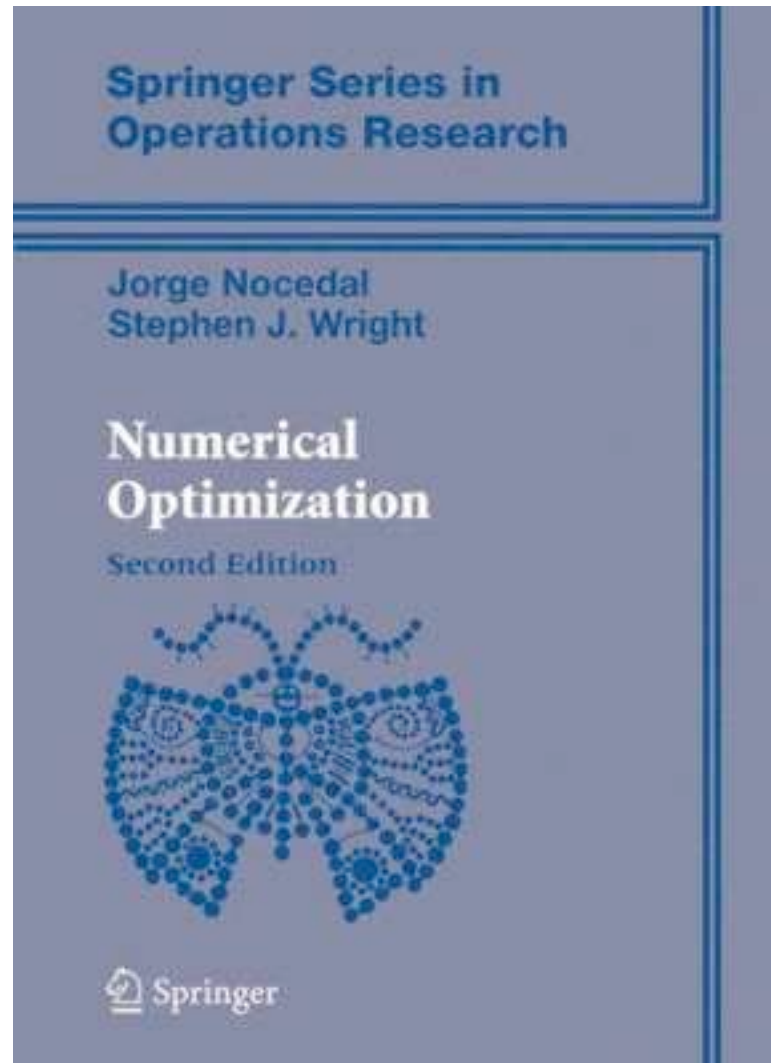
## Interior Point Methods:

- Easy to code
- Easy to design custom hardware/circuits
- Numerical ill-conditioning an issue for some problems
- (Conservative) polynomial-time bounds
- “Predictable” run-time

## Active Set Methods:

- Not so easy to code
- Not so easy to design custom hardware/circuits
- Exponential run-time in worst case
- “Not-so-predictable” run-time
- Very fast for some problems

# Books on Optimization



# Software for Solving a Convex QP

$$\begin{aligned} \min_{\theta} \quad & \frac{1}{2} \theta' H \theta + h' \theta \\ & F \theta = f \\ & G \theta \leq g \end{aligned}$$

Matlab: QUADPROG

Octave: QP

Scilab: QPSOLVE

Free: OOQP, BPMPD, CLP, QPC, qpOASES, CVX/CVXGEN

Other: CPLEX, MOSEK, NAG, QPOPT, TOMLAB, XPRESS

Free SOCP/SDP solvers (also solves QPs): SDPT3, SEDUMI, SDPA, SDPNAL, CSDP, DSDP

Interfaces/modelling: YALMIP, AMPL, CVX/CVXGEN, TOMLAB

# Optimality Conditions for a Convex QP

Karush-Kuhn-Tucker (KKT) optimality conditions are *necessary and sufficient*: there exist  $(\nu, \lambda, s)$  s.t.

$$H\theta + F'\nu + G'\lambda + h = 0$$

$$F\theta - f = 0$$

$$G\theta + s - g = 0 \quad \Leftrightarrow$$

$$\Lambda S \mathbf{1} = 0$$

$$s \geq 0, \lambda \geq 0$$

$$\theta = \arg \min_{\vartheta} \frac{1}{2} \vartheta' H \vartheta + h' \vartheta$$

$$F\vartheta = f$$

$$G\vartheta \leq g$$

---

$$\Lambda := \text{diag}(\lambda), S := \text{diag}(s)$$

$s$  is a slack vector, i.e.  $G\theta \leq g \Leftrightarrow \exists s \geq 0 : G\theta + s - g = 0$

Lagrange multipliers:

$\nu$  associated with equality constraints

$\lambda$  associated with inequality constraints

# Discrete Minimum Principle = KKT for QP

Pearson & Sridhar, IEEE Trans. Automat. Control, 1966:  
Stage cost and terminal cost *convex*, dynamics *linear* and  
inequality constraints *convex*, so **discrete minimum principle**  
gives *necessary and sufficient* conditions for the solution:

$$A' \nu_{k+1} = Qx_k + Su_k + \nu_k + E' \lambda_k, \quad k = 1, \dots, N - 1$$

$$\nu_N + Px_N = 0,$$

$$B' \nu_{k+1} = J' \lambda_k + Ru_k + S' x_k, \quad k = 0, \dots, N - 1$$

$$x_0 = \hat{x}, \quad x_{k+1} = Ax_k + Bu_k, \quad k = 0, \dots, N - 1$$

$$\lambda_k \geq 0, \quad Ex_k + Ju_k \leq c, \quad k = 0, \dots, N - 1$$

$$\text{diag}(\lambda_k)(Ex_k + Ju_k - c) = 0, \quad k = 0, \dots, N - 1$$

*Exactly the same* as the **KKT** conditions for the non-condensed  
**QP** formulation, where  $\nu_k$ ,  $\lambda_k$  are the Lagrange multipliers.

# Modified KKT Equations

Most infeasible (start) interior point methods use Newton's method applied to a sequence of linearizations of the *modified* KKT equations:

$$H\theta + F'\nu + G'\lambda + h = 0$$

$$F\theta - f = 0$$

$$G\theta + s - g = 0$$

$$\Lambda S \mathbf{1} = \mathbf{1}\tau$$

where the scalar  $\tau > 0$  is driven to zero, while ensuring the *interior point condition* is satisfied at each iteration:

$$(\lambda, s) > 0$$

**NB:** Just setting  $\tau = 0$  doesn't work in practice!

# Logarithmic Barrier Methods

$$\min_{\theta, s} \frac{1}{2} \theta' H \theta + h' \theta - \tau \mathbf{1}' \ln(s)$$

$$F \theta = f$$

$$G \theta + s = g$$

Provided that  $s > 0$  one can show that the stationary points of the Lagrangian of the above satisfy:

$$H \theta + F' \nu + G' \lambda + h = 0$$

$$F \theta - f = 0$$

$$G \theta + s - g = 0$$

$$\Lambda S \mathbf{1} = \mathbf{1} \tau$$

Use Newton's method to solve above, then reduce  $\tau > 0$  and repeat process till KKT conditions are satisfied

# Newton Step and Max Step Length

$$\text{Residual: } \rho := \begin{bmatrix} \rho_{\text{dual}} \\ \rho_{\text{eq}} \\ \rho_{\text{in}} \\ \rho_{\text{cent}} \end{bmatrix} := \begin{bmatrix} H\theta + F'\nu + G'\lambda + h \\ F\theta - f \\ G\theta + s - g \\ \Lambda S\mathbf{1} - \mathbf{1}\tau \end{bmatrix}$$

The *Newton step*  $(\Delta\theta, \Delta\nu, \Delta\lambda, \Delta s)$  is computed by linearizing the residual about the current iterate and solving the linear system:

$$\begin{bmatrix} H & F' & G' & 0 \\ F & 0 & 0 & 0 \\ G & 0 & 0 & I \\ 0 & 0 & S & \Lambda \end{bmatrix} \begin{bmatrix} \Delta\theta \\ \Delta\nu \\ \Delta\lambda \\ \Delta s \end{bmatrix} = -\rho,$$

$$\text{Max step length: } \alpha_{\max} := \max_{\alpha \in (0,1]} \{ \alpha \mid (\lambda, s) + \alpha(\Delta\lambda, \Delta s) \geq 0 \}$$



# Primal-dual Methods

Do not solve the modified KKT equations exactly.

1. Compute Newton step by solving the linear system:

$$\begin{bmatrix} H & F' & G' & 0 \\ F & 0 & 0 & 0 \\ G & 0 & 0 & I \\ 0 & 0 & S & \Lambda \end{bmatrix} \begin{bmatrix} \Delta\theta \\ \Delta\nu \\ \Delta\lambda \\ \Delta s \end{bmatrix} = - \begin{bmatrix} H\theta + F'\nu + G'\lambda + h \\ F\theta - f \\ G\theta + s - g \\ \Lambda S\mathbf{1} - \mathbf{1}\tau \end{bmatrix}$$

2. Compute step length  $\alpha$  according to some criterion

3. Compute new iterate

$$(\theta, \nu, \lambda, s) \leftarrow (\theta, \nu, \lambda, s) + \alpha(\Delta\theta, \Delta\nu, \Delta\lambda, \Delta s)$$

4. Decrease  $\tau$  according to some criterion

5. If KKT conditions not satisfied, go to step 1

Faster than logarithmic barrier methods in practice.

# Computational Bottleneck

The Newton step is a descent direction for

$$(\theta, \nu, \lambda, s) \mapsto \|\rho\|_2^2$$

Most interior point methods don't just use the residual for determining the step length

$$\alpha \in (0, \alpha_{\max}]$$

The criteria used for computing the step length and reducing  $\tau$  are important, since they determine the number of iterations.

**However:** Computing the *Newton step* takes up most of the time and resources per iteration.

# Computing a Newton Step

Solve the linear system:

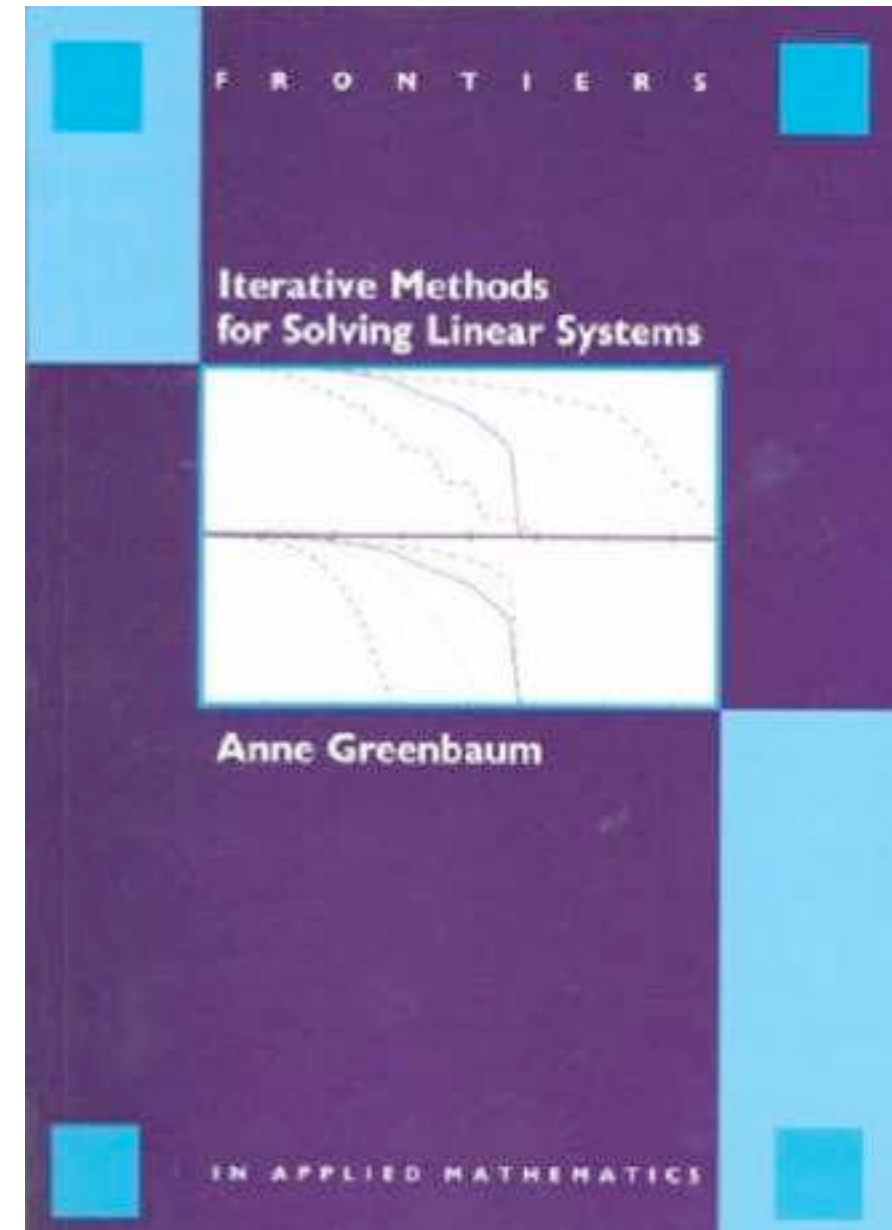
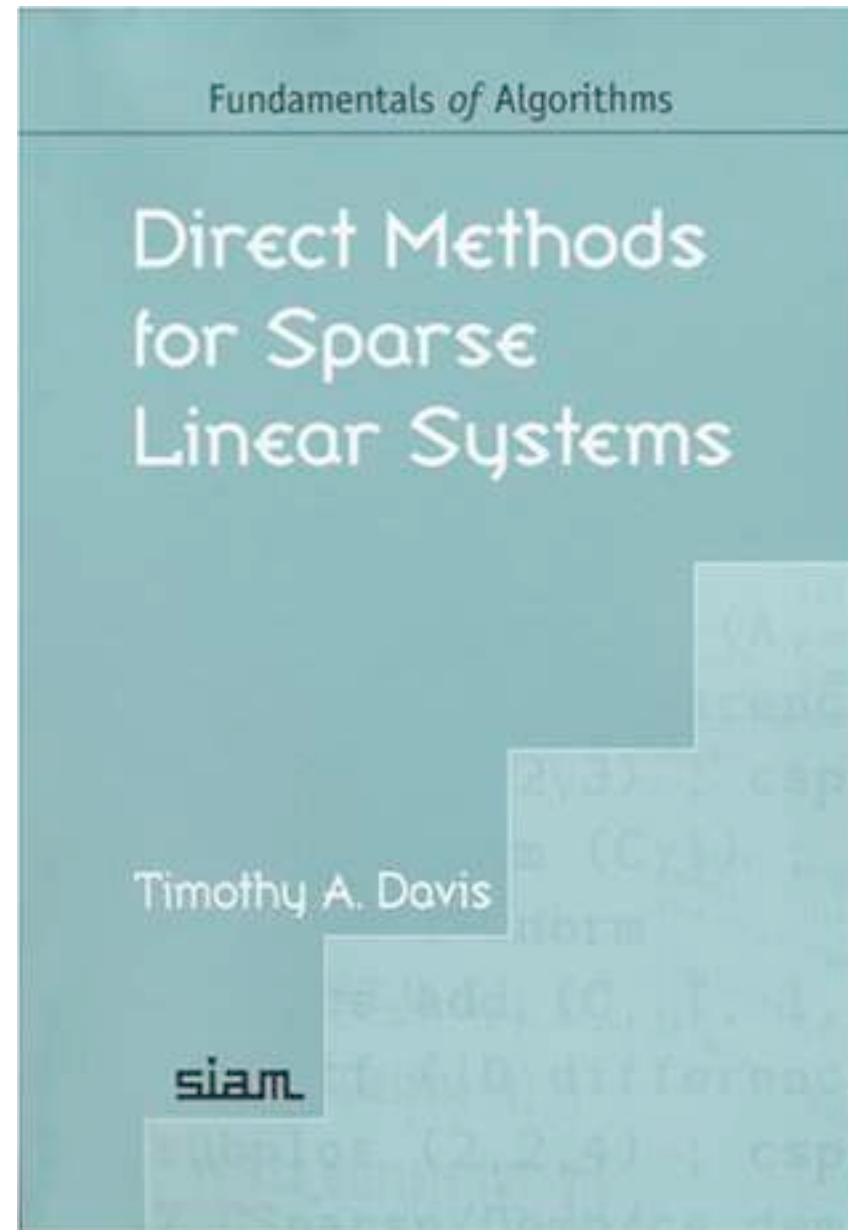
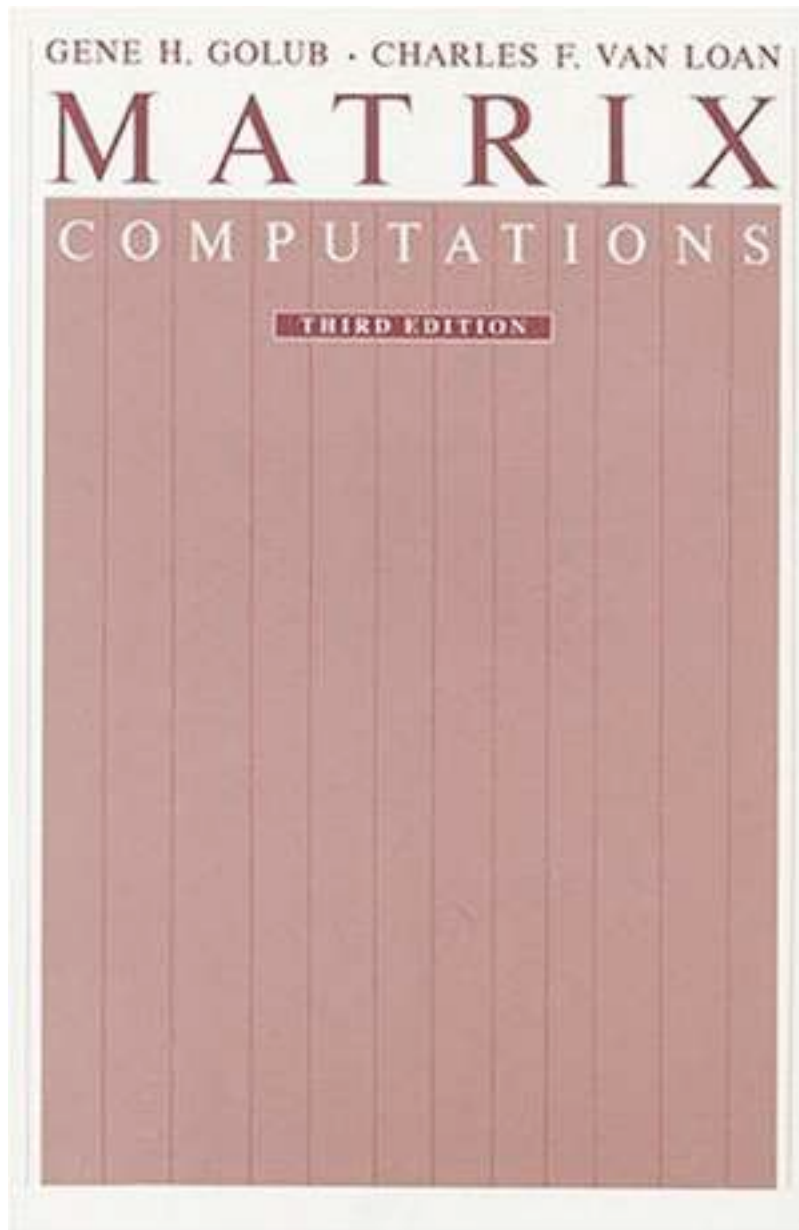
$$\begin{bmatrix} H & F' & G' & 0 \\ F & 0 & 0 & 0 \\ G & 0 & 0 & I \\ 0 & 0 & S & \Lambda \end{bmatrix} \begin{bmatrix} \Delta\theta \\ \Delta\nu \\ \Delta\lambda \\ \Delta s \end{bmatrix} = - \begin{bmatrix} \rho_{\text{dual}} \\ \rho_{\text{eq}} \\ \rho_{\text{in}} \\ \rho_{\text{cent}} \end{bmatrix}$$

The *KKT matrix* is:

- Non-symmetric
- Indefinite
- Can be very sparse and structured
- Can be ill-conditioned

Solution method extremely important.

# Books on Solving Linear Systems



# Methods for Solving Linear Systems

	<b>Non-symmetric matrix</b>	<b>Symmetric and indefinite matrix</b>	<b>Symmetric and positive definite matrix</b>
<b>Direct methods</b>	LU factorization	LDL factorization	Cholesky factorization
<b>Iterative methods</b>	Generalized Minimal Residual (GMRES)	Minimal Residual (MINRES)	Conjugate Gradient (CG)

# Important Questions to Ask

Block eliminate some variables before solving?

What is the numerical stability of the method?

Direct or iterative solver?

Sparse or dense solver?

Is there structure that can be exploited, e.g. can re-ordering variables result in a banded KKT matrix?

What is the computational hardware architecture?

# Which Block Eliminations?

Solve for some variables first using block elimination.

Many possible choices - order can affect numerical conditioning and efficiency of computation.

A popular choice, which introduces symmetry:

$$\begin{bmatrix} H & F' & G' \\ F & 0 & 0 \\ G & 0 & -\Lambda^{-1}S \end{bmatrix} \begin{bmatrix} \Delta\theta \\ \Delta\nu \\ \Delta\lambda \end{bmatrix} = - \begin{bmatrix} \rho_{\text{dual}} \\ \rho_{\text{eq}} \\ \rho_{\text{in}} - \Lambda^{-1}\rho_{\text{cent}} \end{bmatrix}$$

$$\Delta s = -\Lambda^{-1} (\rho_{\text{cent}} + S\Delta\lambda)$$

# Further Block Eliminations

One can reduce the system further, at potential expense of reducing sparsity:

$$\begin{bmatrix} H + G' \Lambda S^{-1} G & F' \\ F & 0 \end{bmatrix} \begin{bmatrix} \Delta \theta \\ \Delta \nu \end{bmatrix} = - \begin{bmatrix} \rho_{\text{dual}} + G' \Lambda S^{-1} \rho_{\text{in}} - G' S \rho_{\text{cent}} \\ \rho_{\text{eq}} \end{bmatrix}$$

$$\Delta \lambda = \Lambda S^{-1} G \Delta \theta + \Lambda S^{-1} \rho_{\text{in}} - S \rho_{\text{cent}}$$

$$\Delta s = -\Lambda^{-1} (\rho_{\text{cent}} + S \Delta \lambda)$$

If there are no equality constraints ( $F = 0$ ), then the reduced KKT matrix is positive semi-definite:

$$\left[ H + G' \Lambda S^{-1} G \right] \Delta \theta = -\rho_{\text{dual}} - G' \Lambda S^{-1} \rho_{\text{in}} + G' S \rho_{\text{cent}}$$



# What About Numerical Stability?

Original, non-symmetric KKT matrix:

- Ill-conditioned in later iterations if QP is degenerate, i.e. when Lagrange multipliers are zero for active constraints

Reduced, symmetric KKT matrices:

- Ill-conditioned in later iterations, since some components of  $\Lambda^{-1}S$  tend to zero and some to infinity.

Iterative methods need good pre-conditioners if KKT matrix is ill-conditioned.

Active area of research.

# Direct or Iterative Solvers?

## Direct, e.g. LU

### Advantages:

- Mature theory
- Numerically stable
- Large choice of software

### Disadvantages:

- Not necessarily good for sparse or large matrices
- Cannot terminate early
- Not easily parallelizable

## Iterative, e.g. GMRES

### Advantages:

- Good for sparse or large matrices
- Early termination possible
- Easily parallelizable

### Disadvantages:

- Theory not mature
- Numerically sensitive
- Small choice of software

# Sparse or Dense Solvers?

Big and sparse often faster and requires less memory than small and dense.

- Non-condensed formulation could be better than condensed formulation if care is taken.

Iterative methods often preferred for sparse matrices.

Some direct methods can minimize amount of “fill-in” during factorization of sparse matrices.

# Matlab Example: Why Size Isn't Everything

```
>> n=4e3; % Form a dense matrix
```

```
>> A2=randn(n,n);
```

```
>> b2=randn(n);
```

```
>> tic;x2=A2\b2;toc
```

Elapsed time is 9.021262 seconds.

```
>> n=1e6; % Form larger, but sparse (banded) matrix
```

```
>> A1=diag(sprandn(n,1,1))+[spalloc(n-1,1,0)  
diag(sprandn(n-1,1,1)); spalloc(1,n,0)];
```

```
>> b1=randn(n,1);
```

```
>> tic;x1=A1\b1;toc
```

Elapsed time is 1.048625 seconds.

# Why Structure, and Not Just Size, Matters

Suppose we want to solve a linear system  $Ax = b$ , where  $A$  is an  $n \times n$  positive definite matrix.

One can use Cholesky factorization to solve for  $x$ .

The number of floating point operations needed grows with  $O(n^3)$ .

If the matrix is banded with width  $k$ , then the number of floating point operations grows with  $O(k^2n)$ .

# Is There Structure to Exploit?

**Non-condensed:** Interleave input and state variables with Lagrange multipliers to get a **banded**, reduced KKT matrix:

- SJ Wright, *Parallel Computing*, 1990
- SJ Wright, *J. Optimization Theory Applications*, 1993
- CV Rao, SJ Wright and JB Rawlings, *J. Optimization Theory Applications*, 1998

**Condensed:** Same, but need a change of variables:

- JL Jerez, EC Kerrigan and GA Constantinides. A Condensed and Sparse QP Formulation for Predictive Control, *Proc. IEEE CDC-ECC*, Dec. 2011

# Growth of # Floating Point Operations

If upper and lower bounds on all inputs and states

	Horizon Length $N$	# inputs $m$	# states $n$
Condensed Dense Cholesky	<b>Cubic</b>	<b>Cubic</b>	<b>Linear</b>
Non-condensed Sparse $LDL'$	<b>Linear</b>	<b>Cubic</b>	<b>Cubic</b>

# Relation to Unconstrained LQR Control

The reduced linear system:

$$\begin{bmatrix} H + G' \Lambda S^{-1} G & F' \\ F & 0 \end{bmatrix} \begin{bmatrix} \Delta \theta \\ \Delta \nu \end{bmatrix} = - \begin{bmatrix} \rho_{\text{dual}} + G' \Lambda S^{-1} \rho_{\text{in}} - G' S \rho_{\text{cent}} \\ \rho_{\text{eq}} \end{bmatrix}$$

can be shown to be the necessary and sufficient optimality conditions of a time-varying LQR problem without inequality constraints and suitably-defined cost:

$$\min_{\substack{u_0, u_1, \dots, u_{N-1} \\ x_0, x_1, x_2, \dots, x_N}} \frac{1}{2} x'_N \tilde{Q}_N x_N + \sum_{k=0}^{N-1} \frac{1}{2} \left( x'_k \tilde{Q}_k x_k + u'_k \tilde{R}_k u_k + 2x'_k \tilde{S}_k u_k \right)$$

$$x_0 = \hat{x}, \quad x_{k+1} = Ax_k + Bu_k, \quad k = 0, 1, \dots, N-1$$

Rao, Wright, Rawlings, *J. Optimization Theory Applications*, 1998



# Discrete Minimum Principle = KKT for QP

Pearson & Sridhar, IEEE Trans. Automat. Control, 1966:  
Stage cost and terminal cost *convex*, dynamics *linear* and  
inequality constraints *convex*, so **discrete minimum principle**  
gives *necessary and sufficient* conditions for the solution:

$$A' \nu_{k+1} = Qx_k + Su_k + \nu_k + E' \lambda_k, \quad k = 1, \dots, N - 1$$

$$\nu_N + Px_N = 0,$$

$$B' \nu_{k+1} = J' \lambda_k + Ru_k + S' x_k, \quad k = 0, \dots, N - 1$$

$$x_0 = \hat{x}, \quad x_{k+1} = Ax_k + Bu_k, \quad k = 0, \dots, N - 1$$

$$\lambda_k \geq 0, \quad Ex_k + Ju_k \leq c, \quad k = 0, \dots, N - 1$$

$$\text{diag}(\lambda_k)(Ex_k + Ju_k - c) = 0, \quad k = 0, \dots, N - 1$$

*Exactly the same* as the **KKT** conditions for the non-condensed  
**QP** formulation, where  $\nu_k$ ,  $\lambda_k$  are the Lagrange multipliers.

# Which Hardware is Faster?



Concorde



Shinkansen

# Which Hardware is Faster?



100 passengers  
Mach 2 (2160 km/h)

Concorde

1630 passengers  
300 km/h



Shinkansen



# Which Hardware is Faster?



Concorde

100 passengers  
Mach 2 (2160 km/h)

300km:  
100 passengers = 8-9 min  
1630 passengers = 6hr 35 min

1630 passengers  
300 km/h

300km:  
100 passengers = 1hr  
1630 passengers = 1hr



Shinkansen

# What About the Hardware?

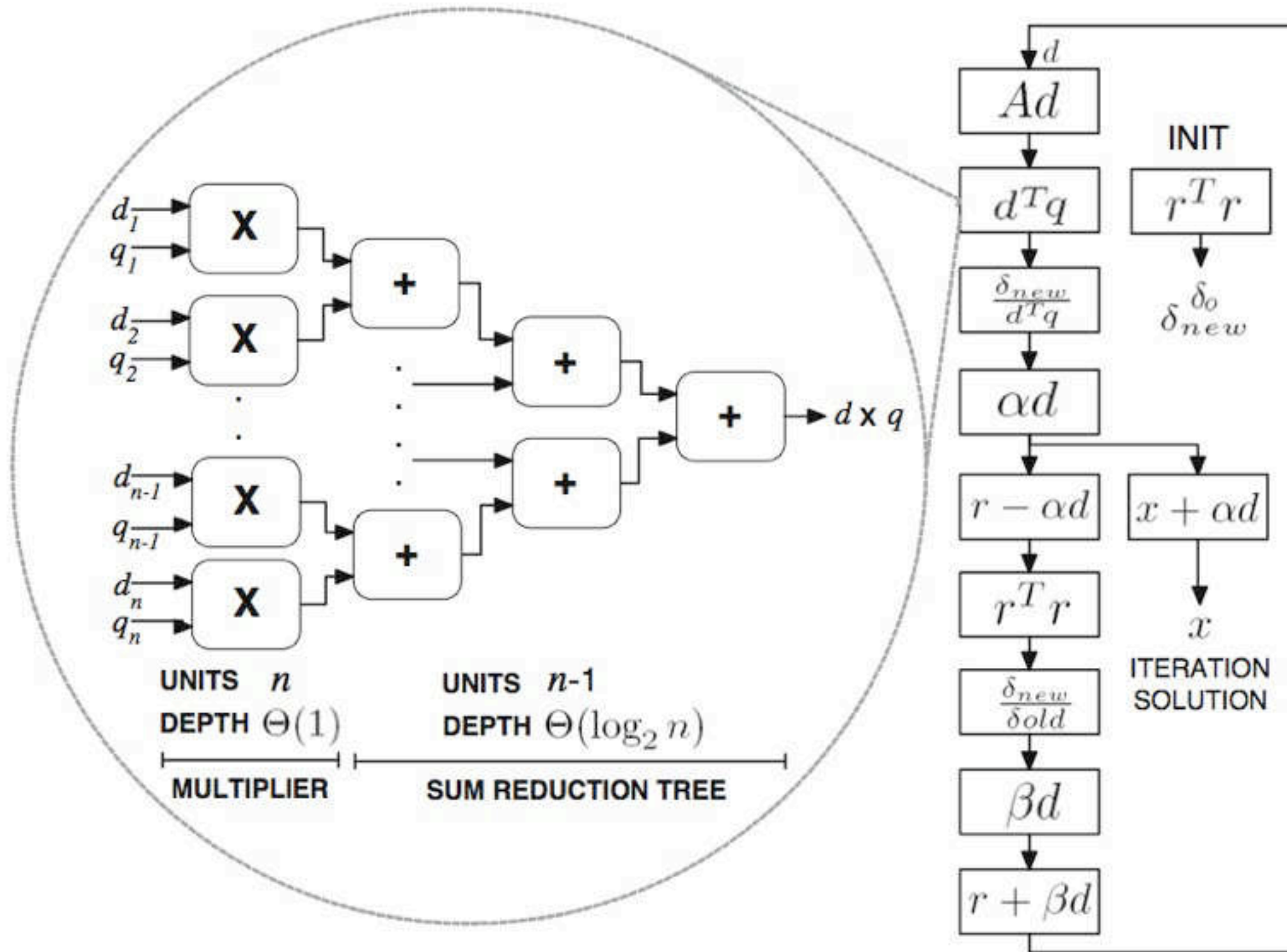
It's not just about:

- the software
- how fast the clock is
- the number of transistors
- how much memory there is
- the energy available
- the computational delay
- the sample rate
- the accuracy of the computation
- the closed-loop performance
- the robustness of the closed-loop

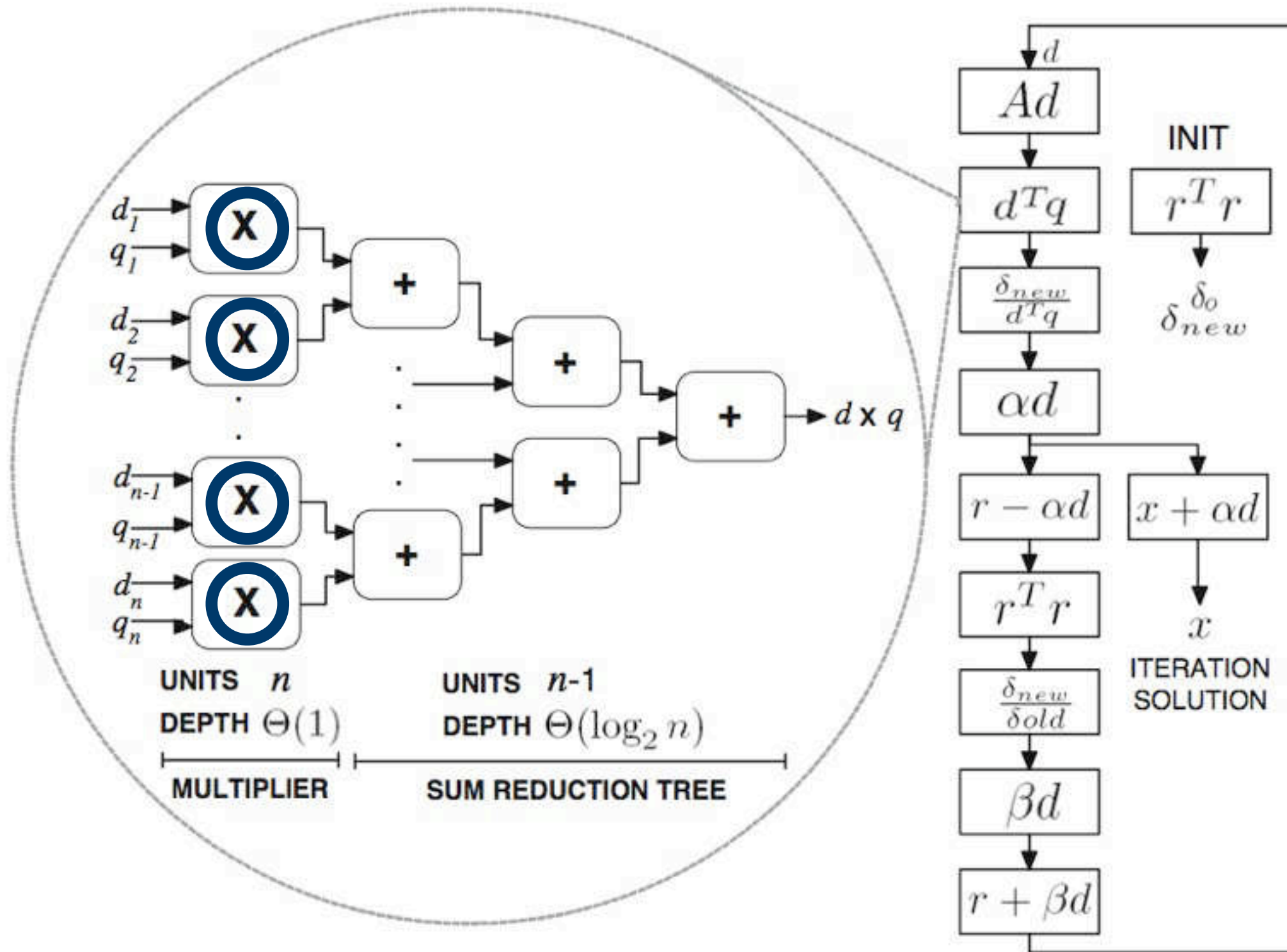
There is no optimal hardware implementation.

There is always a trade-off - “good enough” is possible

# Pipelining in CG Method

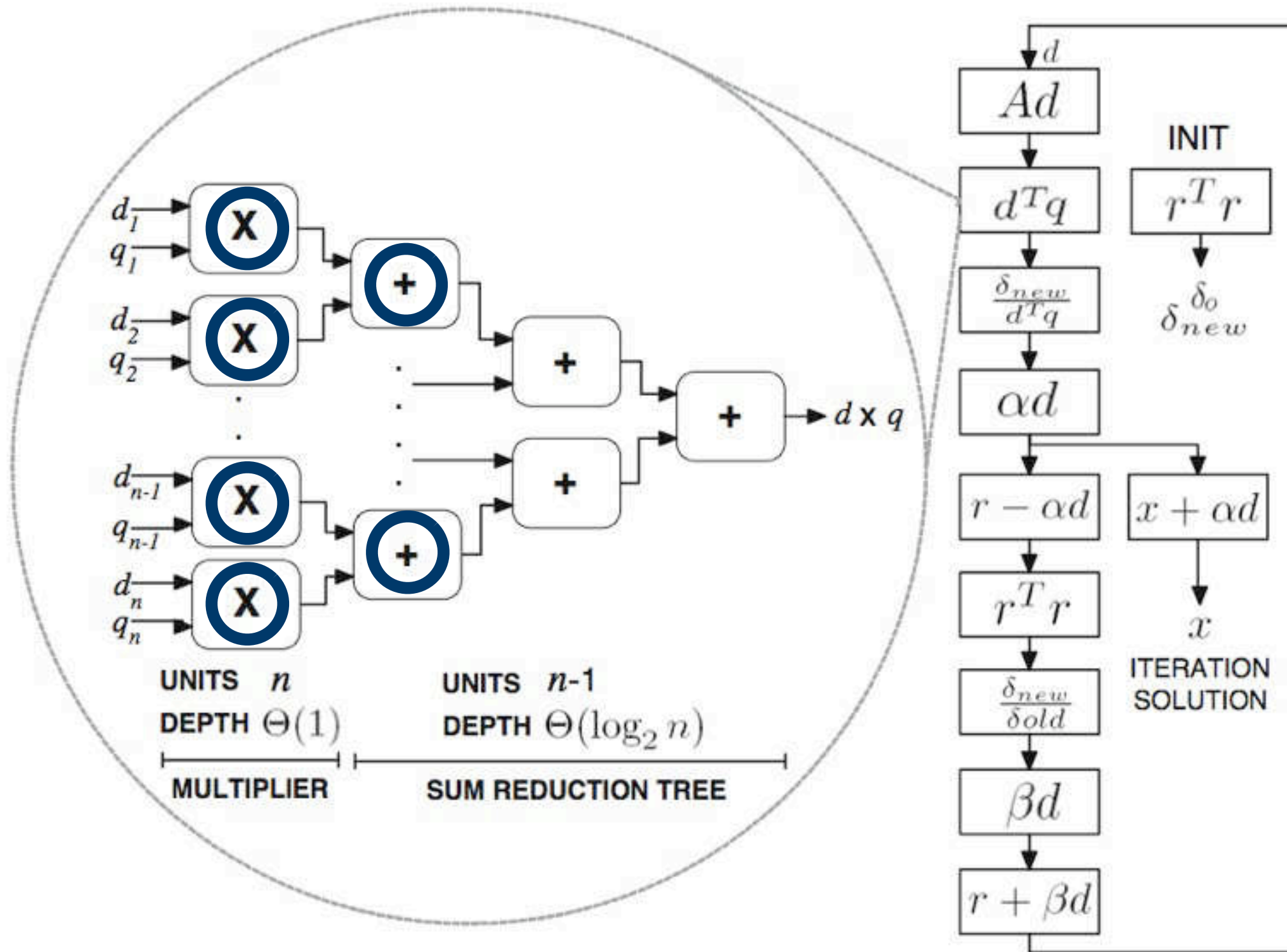


# Pipelining in CG Method



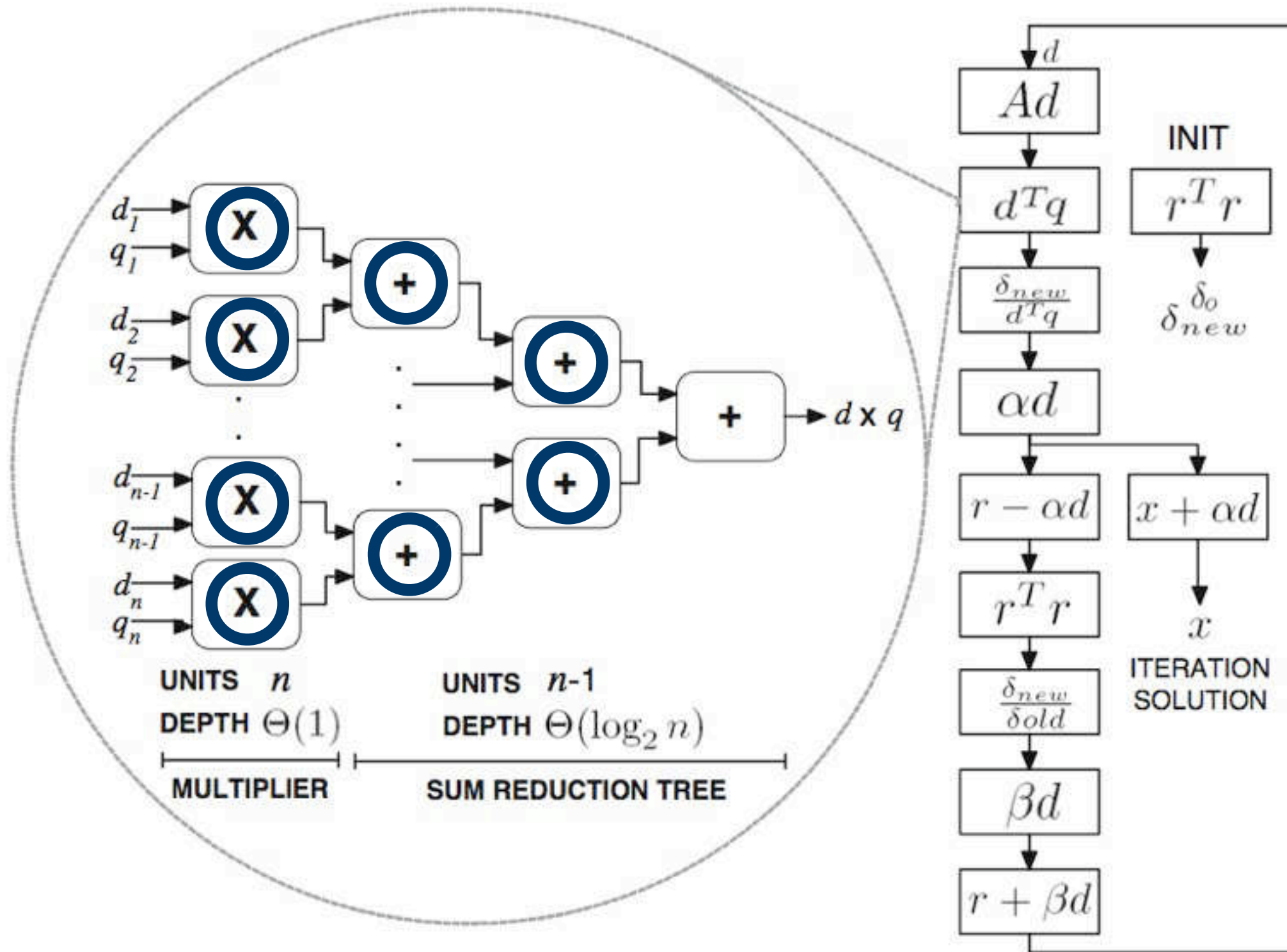


# Pipelining in CG Method

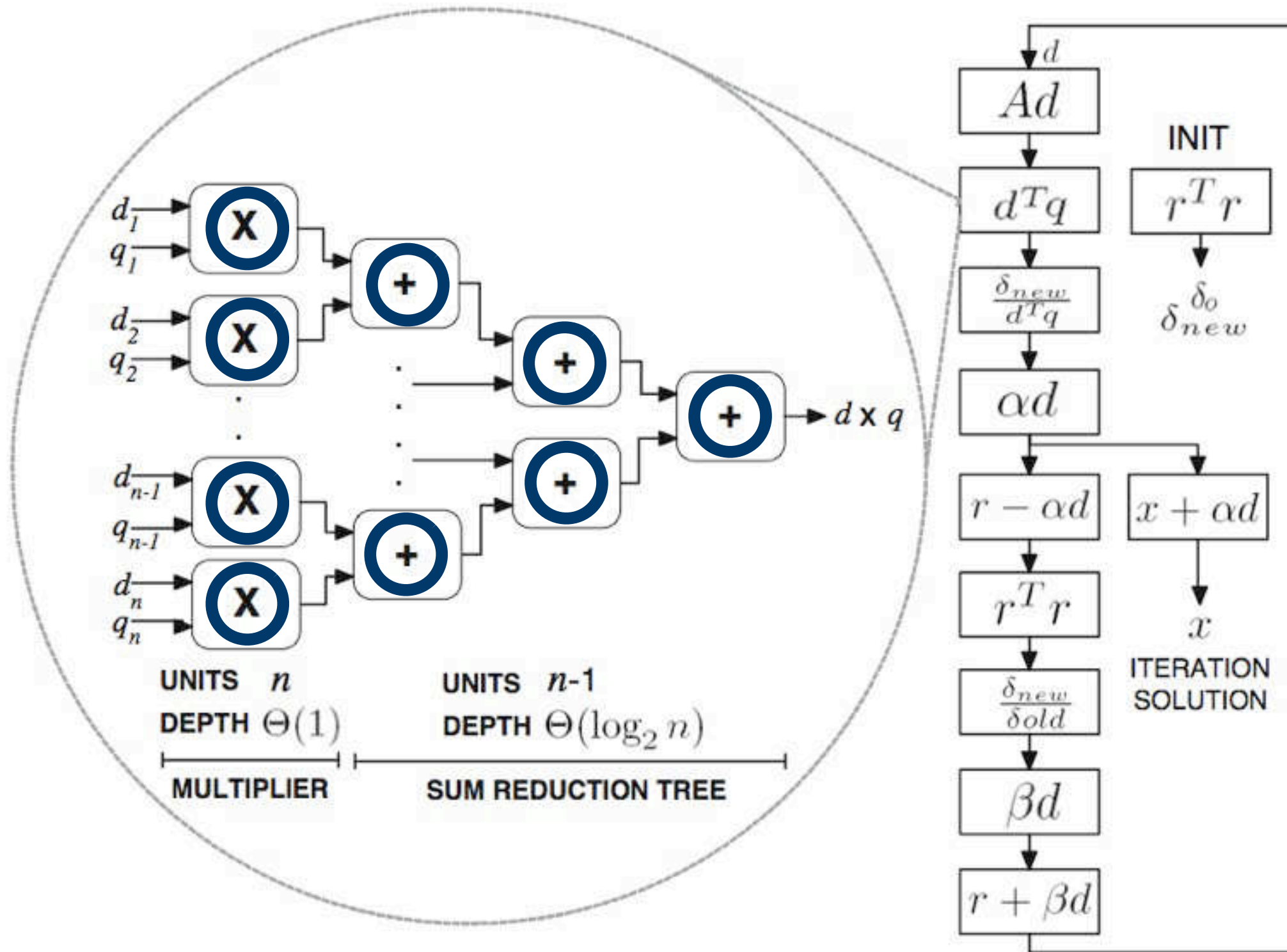




# Pipelining in CG Method

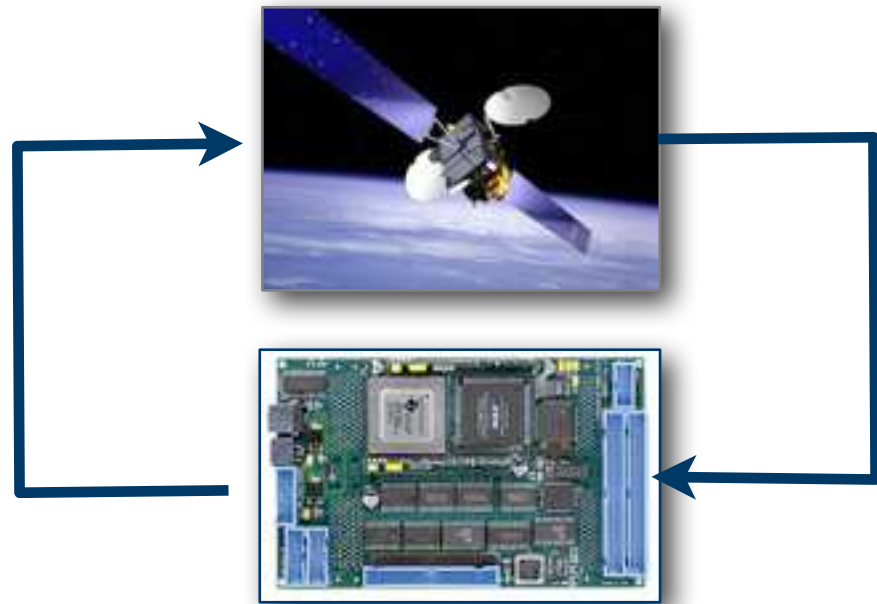


# Pipelining in CG Method



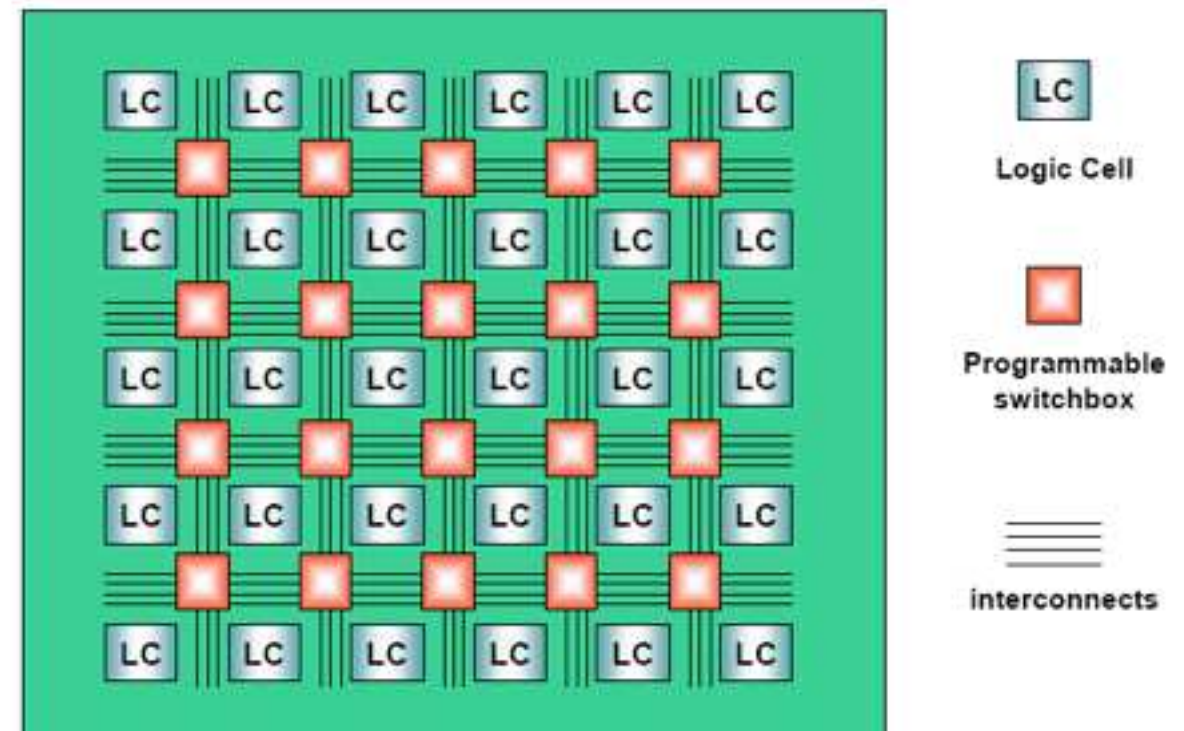
# Embedded Optimization and Control

Algorithm Design  $\longleftrightarrow$  Silicon Implementation



Field-Programmable Gate Array (FPGA)

- Parallel and pipelined computations
- Flexible number representation
- Design memory subsystem
- Predictable
- Energy efficient (<10mW-10W)
- Budget to match demand (<\$10-\$20k)

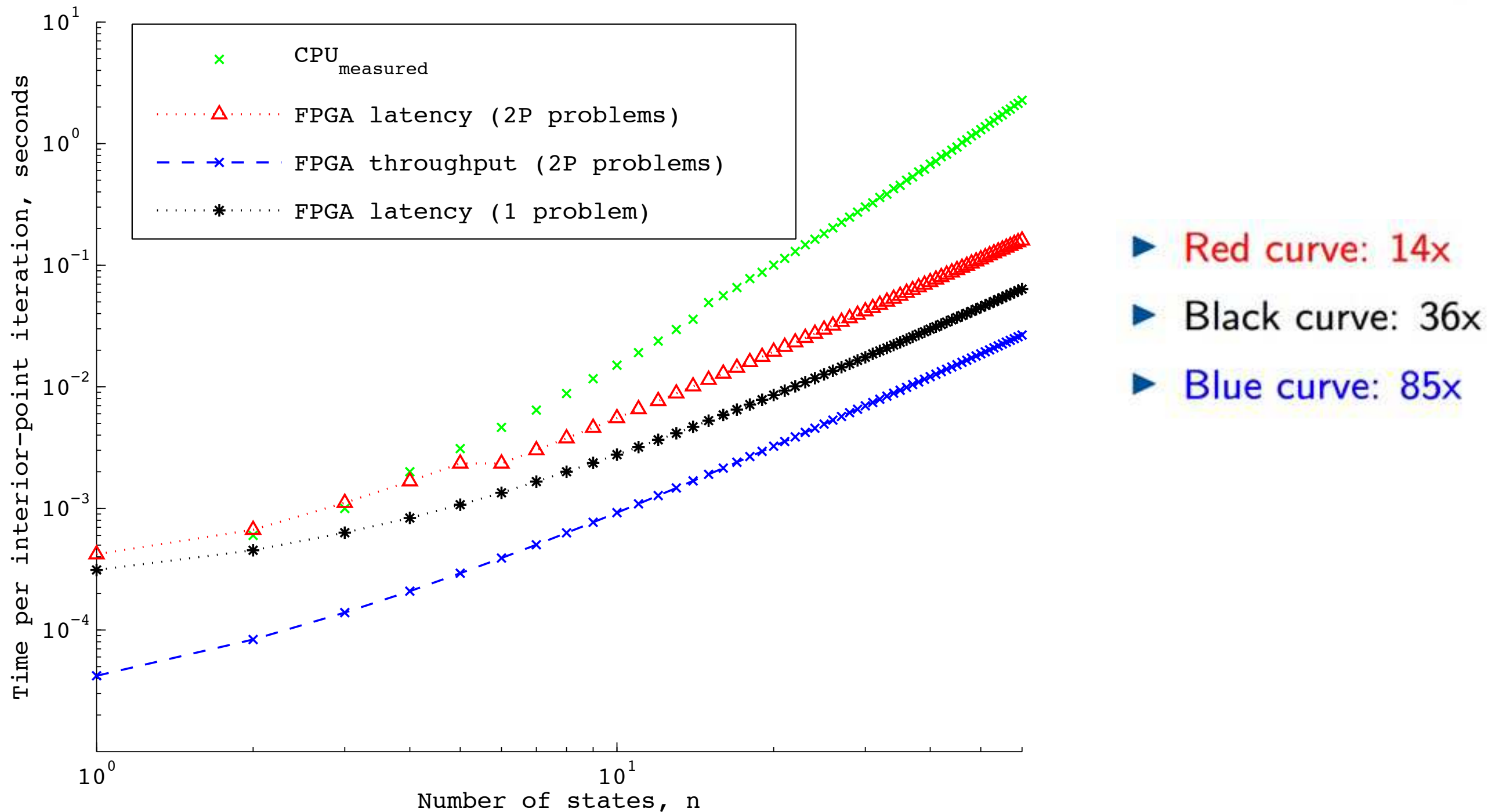




# Time: Non-condensed QP Formulation

Hardware : Xilinx Virtex 6 SX 475T @ 250MHz (40nm)

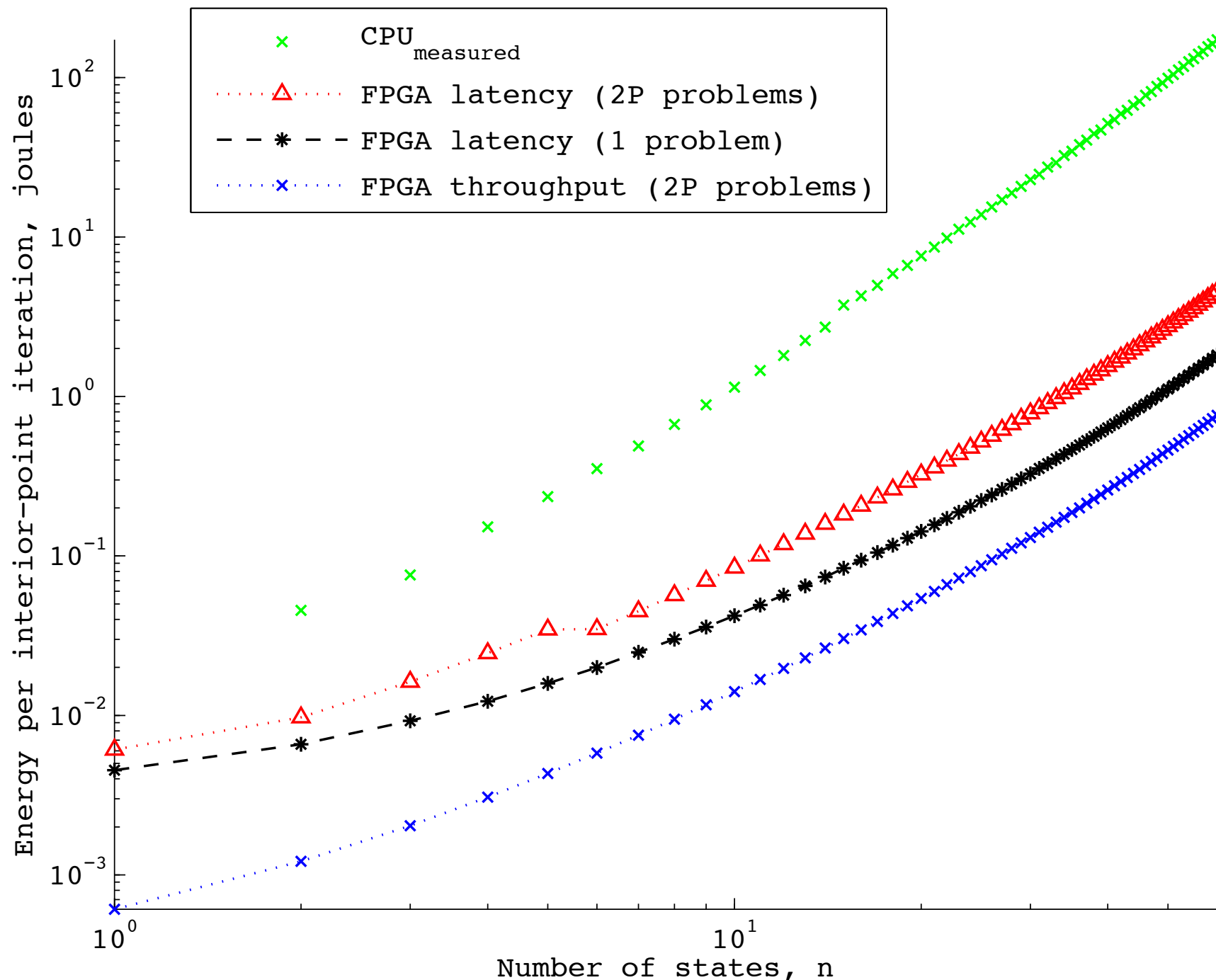
Software : Intel Core2 Q8300 @ 2.5GHz, 3GB RAM, 4MB L2 Cache (45nm)



# Energy: Non-condensed QP Formulation

Hardware : Xilinx Virtex 6 SX 475T @ 250MHz (40nm)

Software : Intel Core2 Q8300 @ 2.5GHz, 3GB RAM, 4MB L2 Cache (45nm)



CPU power ~ 76W

FPGA board power ~ 14-28W

For the largest problem, the improvement is:

▶ Red curve: 38x

▶ Black curve: 95x

▶ Blue curve: 227x

# Selected Papers

GA Constantinides. Tutorial Paper: Parallel Architectures for Model Predictive Control. *Proc. European Control Conference 2009*.

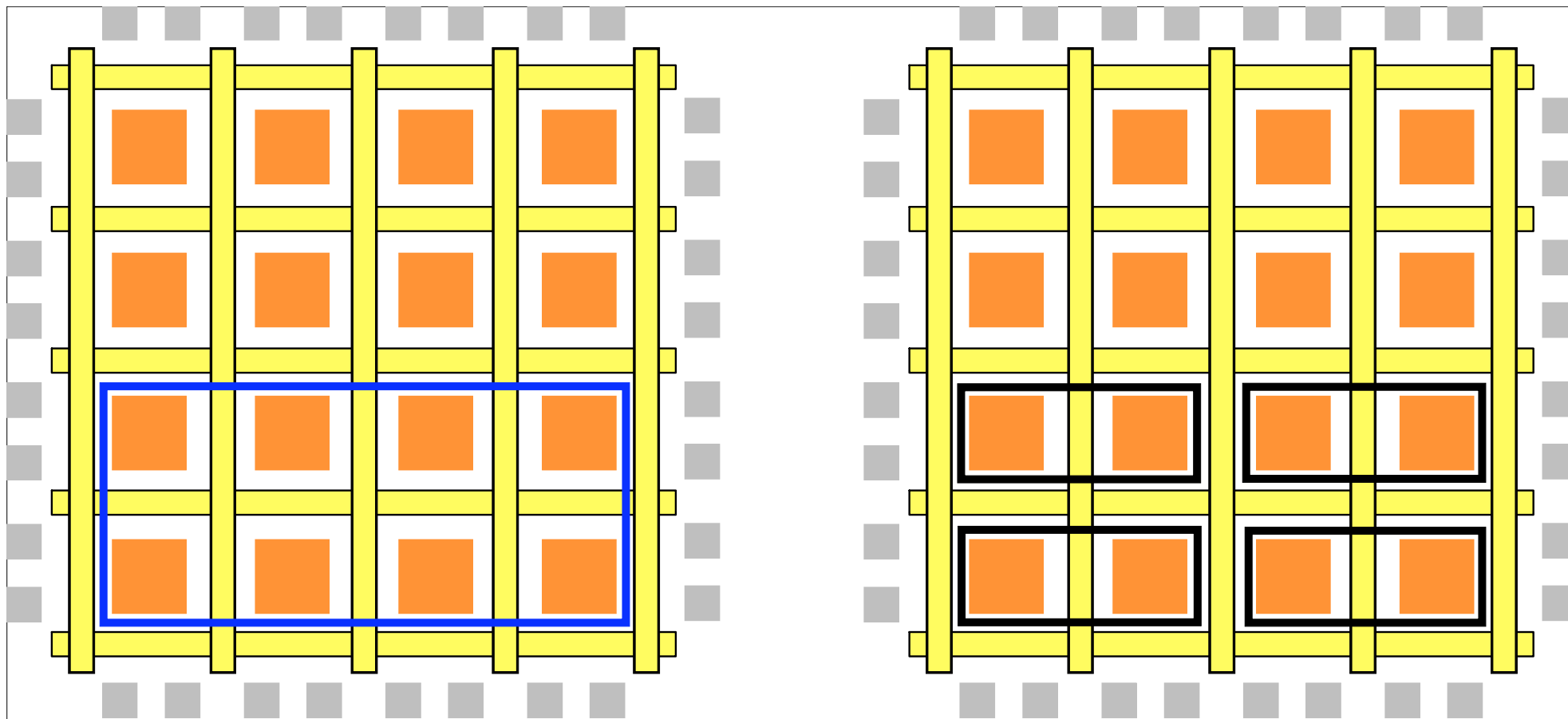
AG Wills, G Knagge, B Ninness. Fast Linear Model Predictive Control Via Custom Integrated Circuit Architecture. *IEEE Trans. Control Systems Technology* 2011.

JL Jerez, GA Constantinides, EC Kerrigan, K-V Ling. Parallel MPC for Real-Time FPGA-based Implementation. *Proc. 18th IFAC World Congress, 2011*.

# Precision/speed Trade-offs in Silicon

Silicon area used grows quadratically with precision

Less precision => more parallelism => potential speed up

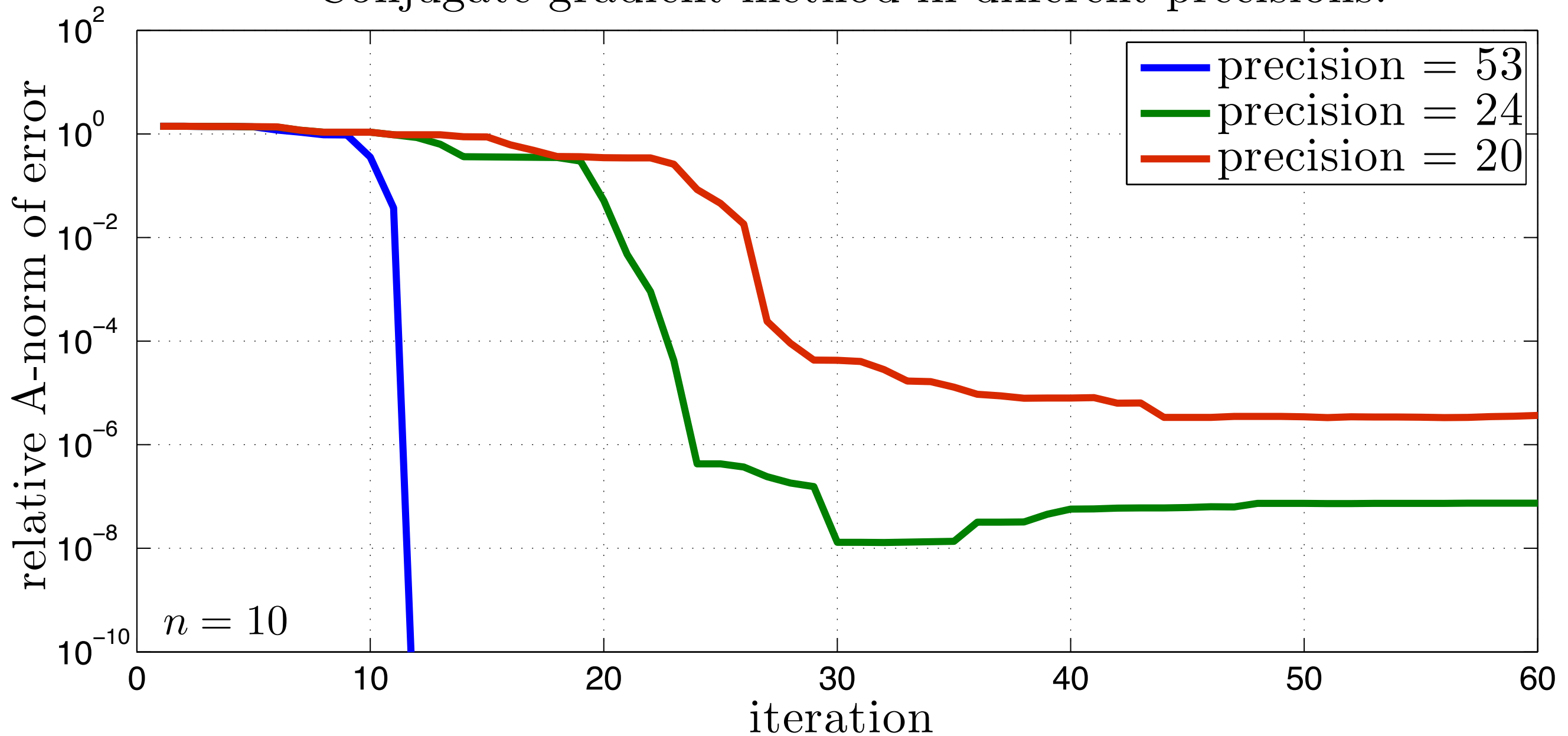


1x high precision circuit

4x low (half) precision circuits

# Finite Precision Effects

Conjugate gradient method in different precisions.





# Summary

## Motivation

- Use feedback if and only if uncertainty is present
- Predictive control is a constrained feedback strategy

## Repeatedly solve an open-loop CLQR problem

- Solve sampled-data CLQR problem by solving a QP
- Non-condensed and condensed QP formulations

## Interior point methods

- Use Newton's method to solve modified KKT conditions
- Computing Newton step uses most of the resources

## Hardware issues

- Hardware should inform choice of algorithm and vice versa
- Trade-offs always have to be made

# Extensions

Nonlinear dynamics

Time-varying dynamics and constraints

Non-quadratic costs, e.g. 1-norm, infinity-norm, time

Robust formulations, e.g. min-max, stochastic

State estimation with constraints and nonlinearities

# Further Reading

