Stochastic Dynamic Optimization for Trading and Dispatching on Intra-day Electricity Markets

Alberto Bemporad, Professor
IMT Lucca

EQuanT Bootcamp 2014

IMT INSTITUTE FOR ADVANCED STUDIES LUCCA

-E-QuantT- bootcamp

Quantitative Analysis and Modelling for Energy Trading and Risk Management

IMT Lucca, Tuscany (Italy), 13-17th October 2014

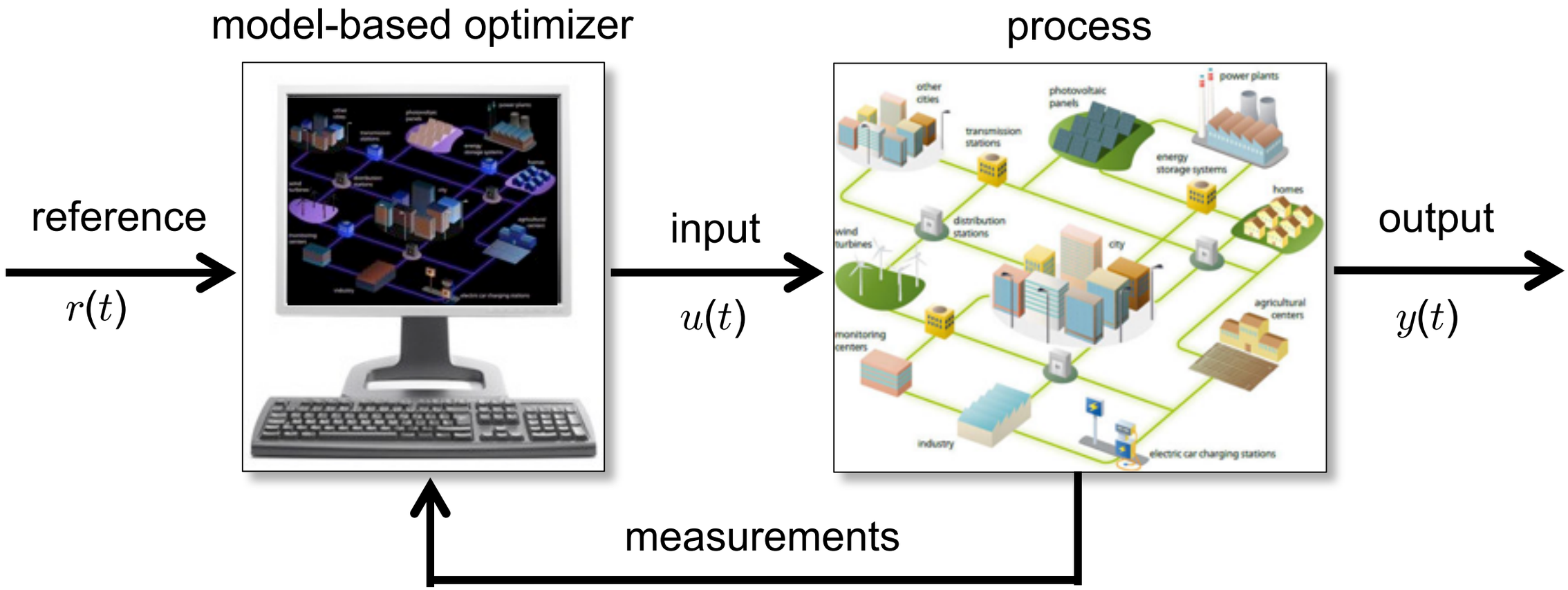
Outline

- Model Predictive Control (MPC): main concepts
- Stochastic Model Predictive Control (SMPC)
- SMPC for real-time market-based optimal power dispatch
- SMPC for optimal bidding on energy markets
- SMPC for dynamic option hedging

Hands-on demonstration of MATLAB tools for SMPC

Basic principles of Model Predictive Control

Model Predictive Control (MPC)



*Use a dynamical **model** of the process to **predict** its possible future evolution and choose the "best" **control** action*

Receding horizon control

- **At time t:** solve an **optimal control** problem over a future horizon of N steps

penalty on tracking error

penalty on actuation effort

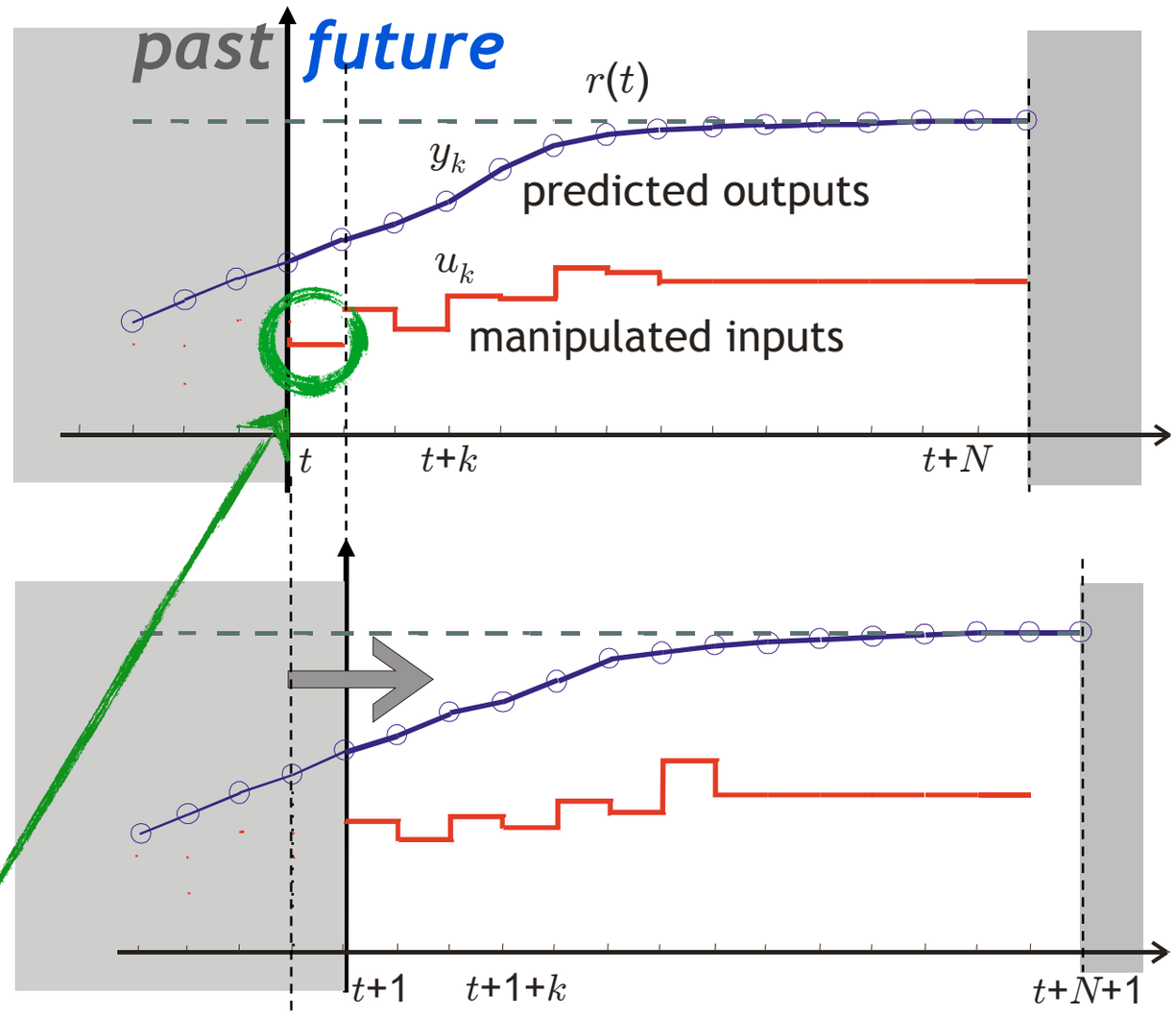
$$\min \quad \sum_{k=0}^{N-1} \ell(y_k - r(t+k), u_k)$$

$$\text{s.t.} \quad x_{k+1} = f(x_k, u_k)$$

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

constraints on u_k, x_k, y_k

$$x_0 = x(t)$$



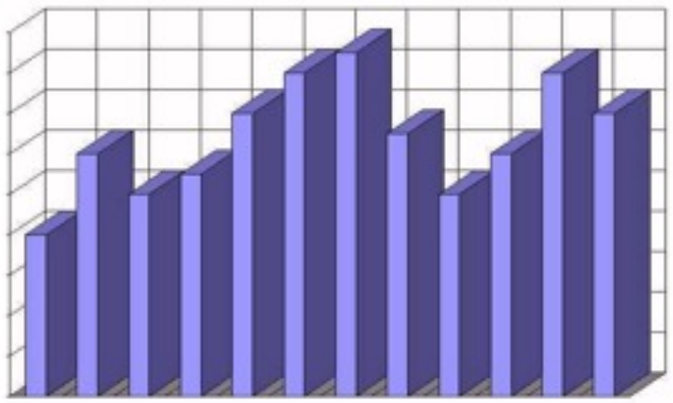
- Apply only the first optimal move $u^*(t)$, throw the rest of the sequence away
- **At time** $t+1$: Get new measurements, repeat the optimization. And so on …

MPC transforms open-loop optimal control into **feedback** control

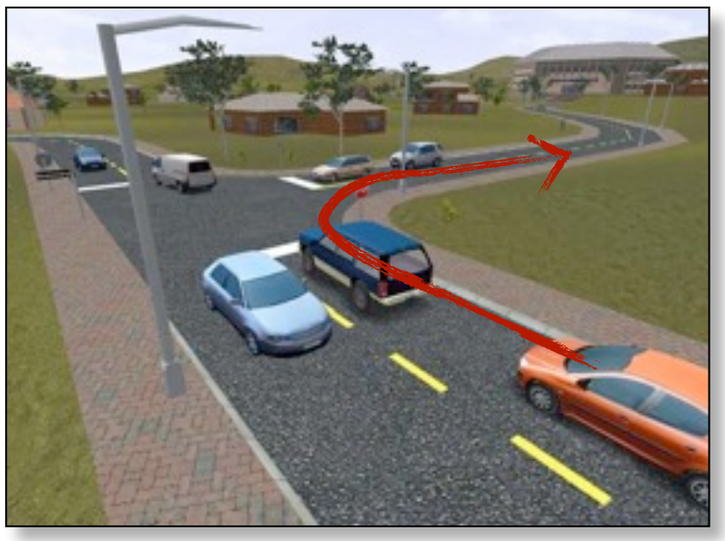
Receding horizon examples

- MPC is like **playing chess !**

- “Rolling horizon” policies are also used frequently **in finance**



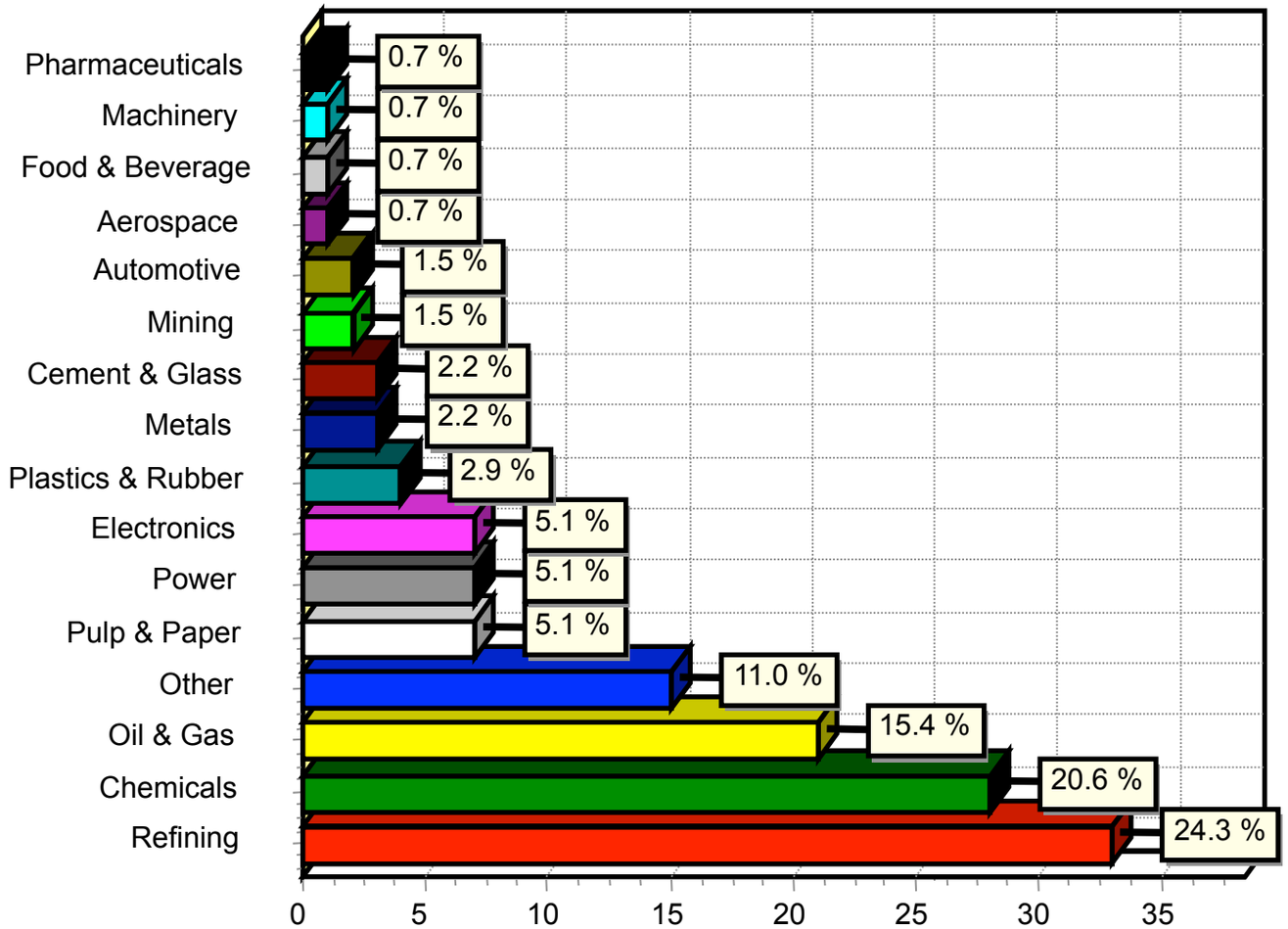
- Drivers use “receding horizon” control too!



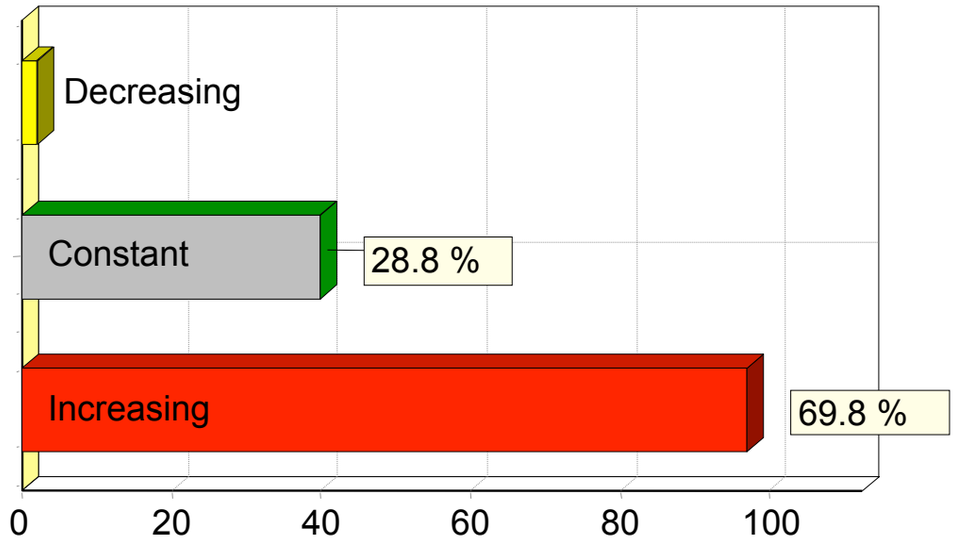
MPC in industry

Results from a survey (November 2005) about the use of MPC techniques / real-time optimization in a set of US industries:

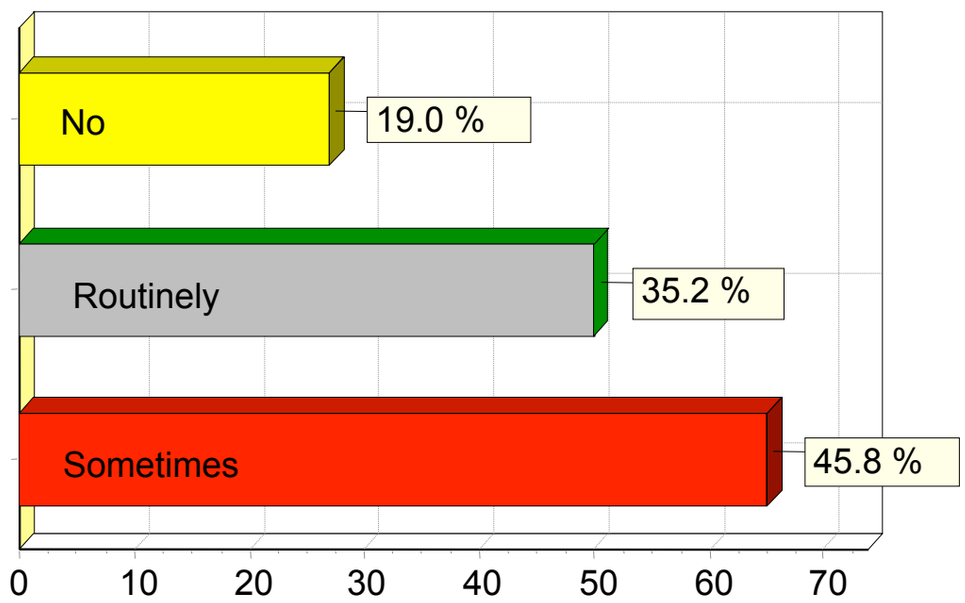
Industrial area of respondents to the survey:



Do you see your use of MPC accelerating, staying constant, or declining?



Does your company use MPC ?

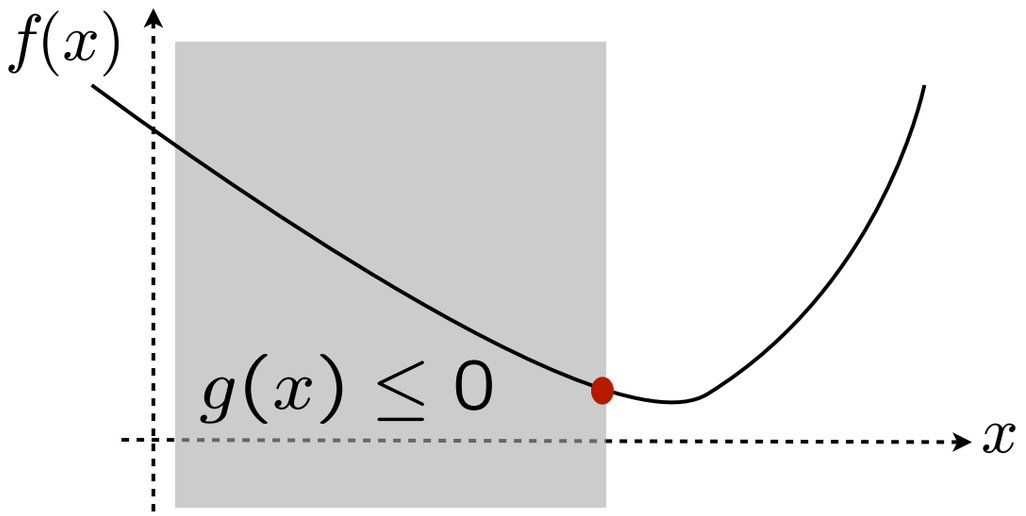


courtesy: ARC Advisory Group

Mathematical programming

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & g(x) \leq 0 \end{aligned}$$

$$x \in \mathbb{R}^n, \; f : \mathbb{R}^n \to \mathbb{R}, \; g : \mathbb{R}^n \to \mathbb{R}^m$$



$$\left(\begin{aligned} \max_x \quad & f(x) \\ \text{s.t.} \quad & g(x) \leq 0 \end{aligned} \right)$$

$$f(x) = f(x_1, \ldots, x_n) \qquad x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \qquad g(x) = \begin{bmatrix} g_1(x_1, \ldots, x_n) \\ g_2(x_1, \ldots, x_n) \\ \vdots \\ g_m(x_1, \ldots, x_n) \end{bmatrix}$$

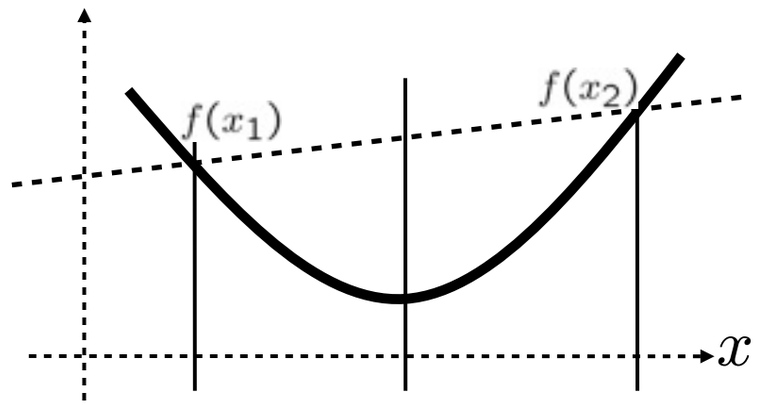
In general, problem is difficult to solve ⟹ **use software tools**

Convex optimization problem

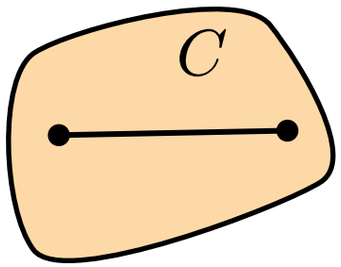
$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & x \in C \end{aligned}$$

f = convex function

C = convex set



- Very efficient numerical algorithms exist
- Global solution attained
- Extensive useful theory
- Often occurring in engineering problems
- Tractable in theory and in practice



Excellent reference textbook: “Convex Optimization” by S. Boyd and L. Vandenberghe

http://www.stanford.edu/~boyd/cvxbook/

Linear programming

$$\begin{array}{ll} \min & f'x \\ \text{s.t.} & Ax \leq b, \;\; x \in \mathbb{R}^n \end{array}$$

linear program (LP)

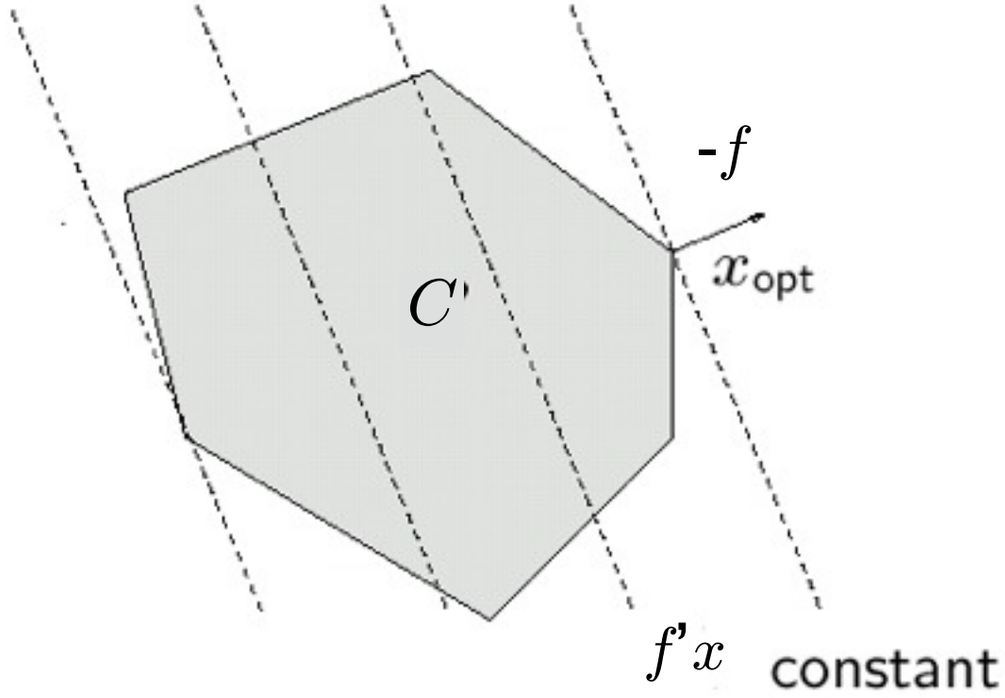
f = linear function

C = polyhedron

George Dantzig
(1914 - 2005)

$$\begin{array}{ll} \min & f'x \\ \text{s.t.} & Ax = b \\ & x \geq 0, \;\; x \in \mathbb{R}^n \end{array}$$

LP in standard form



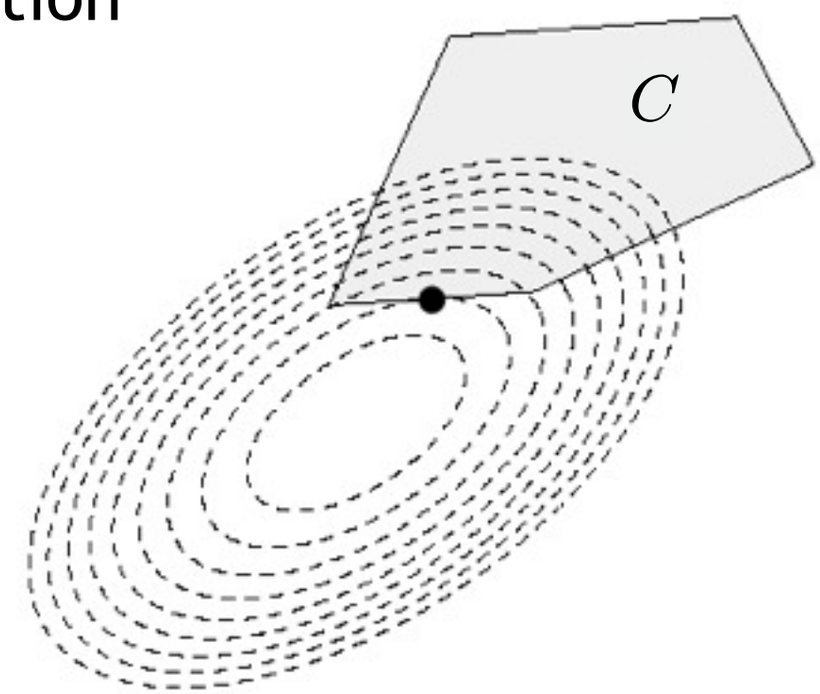
Quadratic programming

$$\begin{array}{ll} \min & \frac{1}{2}x'Px + f'x \\ \text{s.t.} & Ax \leq b, \;\; x \in \mathbb{R}^n \end{array}$$

quadratic program (QP)

f = quadratic function

C = polyhedron



- Convex optimization if $P \geq 0$ ($P =$ positive semidefinite matrix)
- Hard problem if $P \not\geq 0$ ($P =$ indefinite matrix)

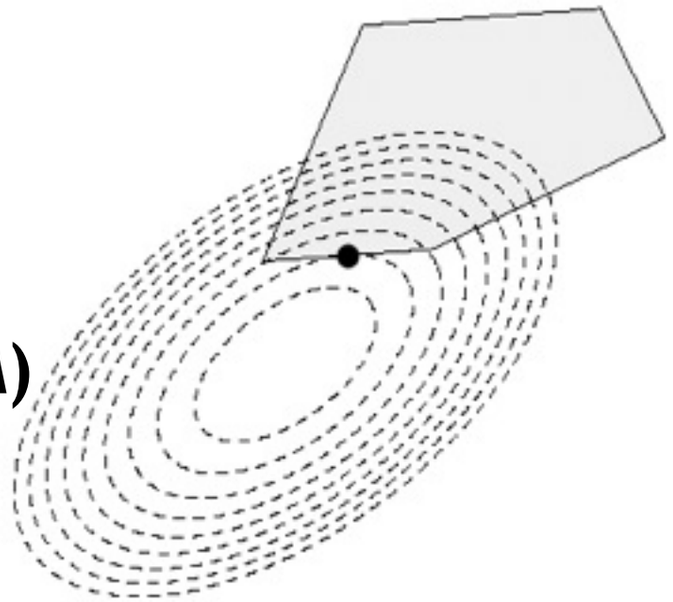
Solution methods for QP

Most used algorithms for solving QP problems:

- ***active set*** methods **(small/medium size)**
- ***interior point*** methods **(large scale/sparse)**
- ***gradient projection*** methods
- ***conjugate gradient*** methods
- ***alternating direction method of multipliers*** **(ADMM)**

$$\begin{aligned} \min_z \quad & \tfrac{1}{2} z' H z + x' F z \\ \text{s.t.} \quad & G z \leq W + S x \end{aligned}$$

Quadratic Program (QP)



A useful performance comparison of many solvers can be found at

http://plato.asu.edu/guide.html

Modeling languages

- **AMPL** (A Modeling Language for Mathematical Programming) most used modeling language, supports several solvers
- **OPL** (Optimization Programming Language), associated with commercial package IBM-CPLEX
- **MOSEL**, associated with commercial package FICO Xpress
- **GAMS** (General Algebraic Modeling System) is one of the first modeling languages
- **LINGO**, modeling language of Lindo Systems Inc.
- **GNU MathProg**, a subset of AMPL associated with the *free* package GLPK (GNU Linear Programming Kit)
- **FLOPC++** *open source* modeling language (C++ class library)

Modeling languages

- **YALMIP** MATLAB-based modeling language
- **CVX** MATLAB-based modeling language for convex problems
- **PYOMO** Python-based modeling language
- **PICOS** A Python interface to conic optimization solvers
- **PuLP** An linear programming modeler for Python
- **JuMP** A modeling language for linear, quadratic, and nonlinear constrained optimization problems embedded in **Julia** (`http://julialang.org`)

Linear MPC

- Linear prediction model:

$$\begin{cases} x_{k+1} &= Ax_k + Bu_k \\ y_k &= Cx_k + Du_k \end{cases}$$

$$x \in \mathbb{R}^n$$
$$u \in \mathbb{R}^m$$
$$y \in \mathbb{R}^p$$

$$x_0 = x(t)$$

- Constraints to enforce:

$$\begin{cases} u_{\min} \leq u_k \leq u_{\max} \\ y_{\min} \leq y_k \leq y_{\max} \end{cases}$$

- Constrained optimal control problem (quadratic performance index):

$$\begin{aligned} \min_z \quad & x'_N P x_N + \sum_{k=0}^{N-1} x'_k Q x_k + u'_k R u_k \\ \text{s.t.} \quad & u_{\min} \leq u_k \leq u_{\max}, \;\; k = 0, \ldots, N-1 \\ & y_{\min} \leq y_k \leq y_{\max}, \;\; k = 1, \ldots, N \end{aligned}$$

$$\begin{aligned} R &= R' \succ 0 \\ Q &= Q' \succeq 0 \\ P &= P' \succeq 0 \end{aligned}$$

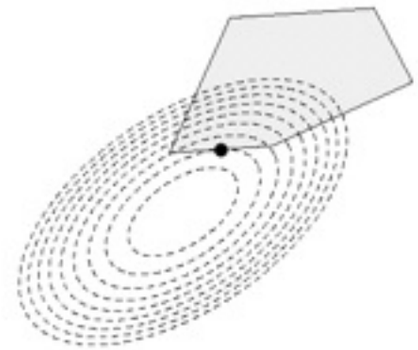
Linear MPC - Constrained case

- Optimization problem:

$$V(x_0) = \tfrac{1}{2}x_0' Y x_0 + \quad \min_z \quad \tfrac{1}{2}z'Hz + x_0'F'z \qquad \text{(quadratic)}$$

$$\text{s.t.} \quad Gz \leq W + Sx_0 \qquad \text{(linear)}$$

Convex QUADRATIC PROGRAM (QP)

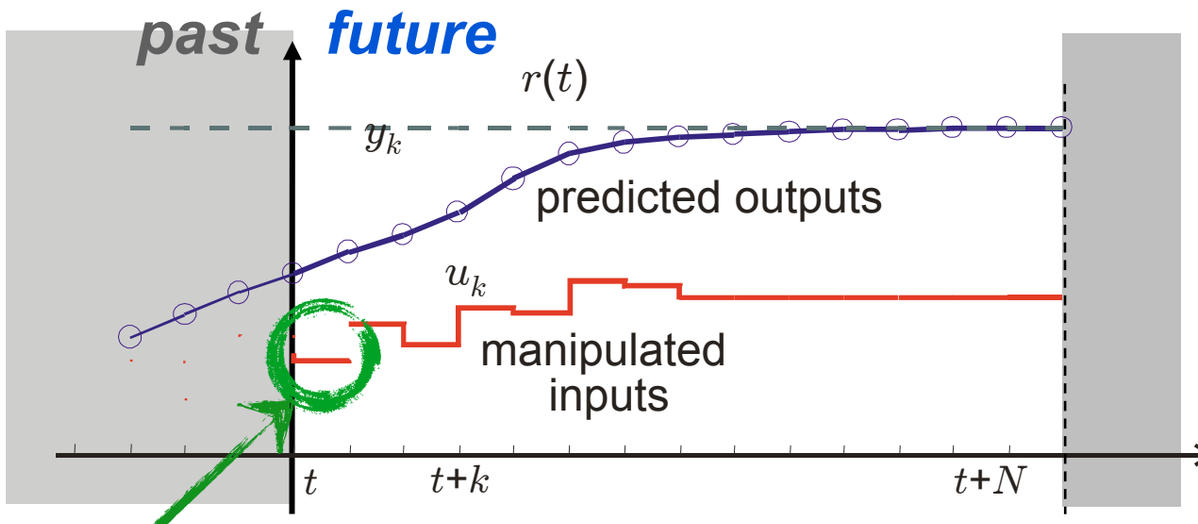


- $z = \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix} \in \mathbb{R}^s, \;\; s \triangleq Nm$ is the optimization vector

- $H = H' \succ 0$ and H, F, Y, G, W, S depend on weights Q, R, P, upper and lower bounds $u_{\min}$, $u_{\max}$, $y_{\min}$, $y_{\max}$, and model matrices A, B, C

Linear MPC algorithm

@ each sampling step t:



- Measure (or estimate) the current state $x(t)$
- Get the solution $z^* = \{u_0^*, \ldots, u_{N-1}^*\}$ of the QP

feedback !

$$\min_z \quad \frac{1}{2} z' H z + x'(t) F' z$$

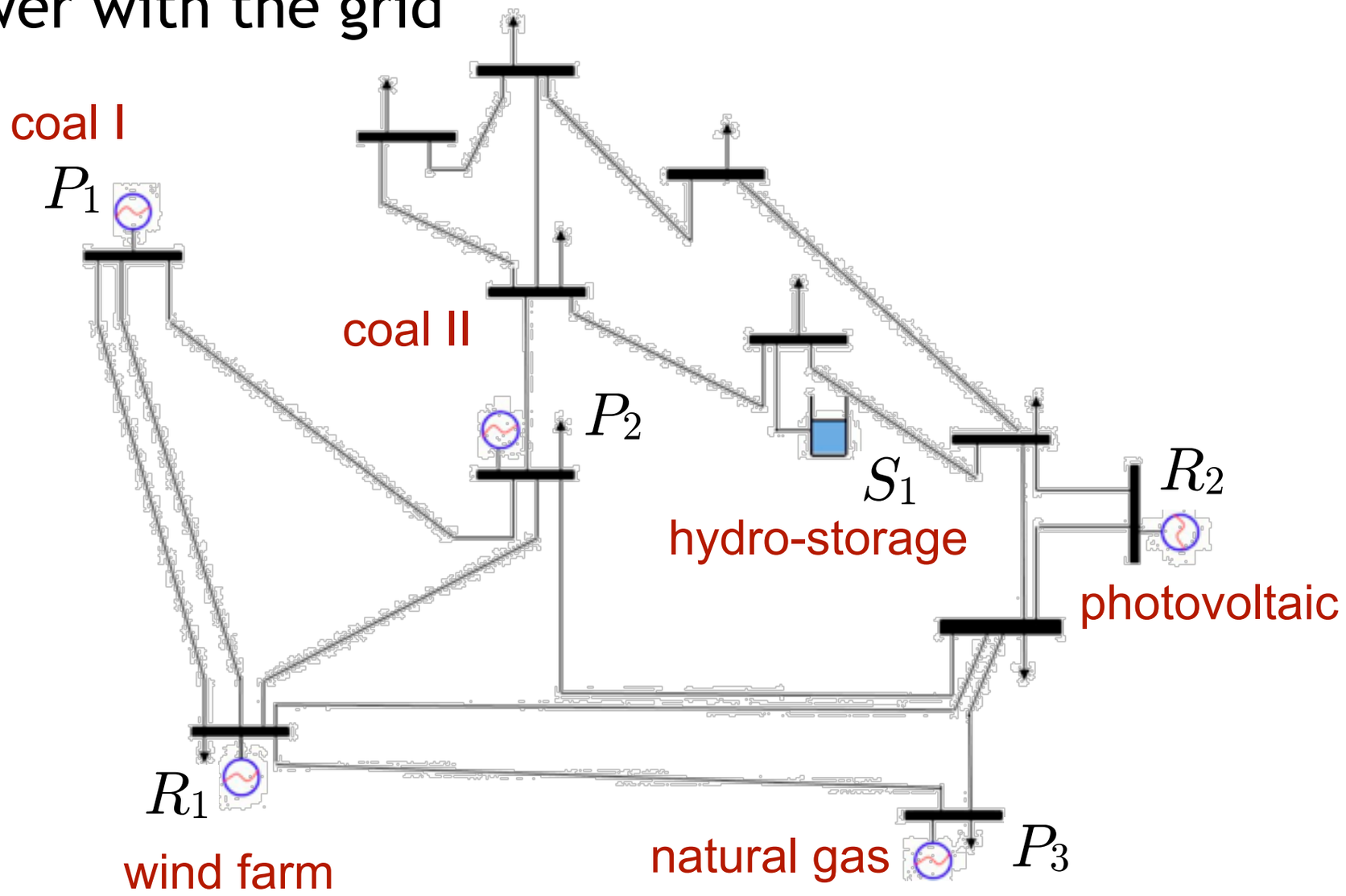
$$\text{s.t.} \quad Gz \leq W + Sx(t)$$

- Apply only $u(t) = u_0^*$, discard remaining optimal inputs $u_1^*, \ldots, u_{N-1}^*$

Example: Power dispatch model

(Patrinos, Trimboli, Bemporad, CDC'11)

- Microgrid with three conventional power generators (P1,P2,P3), two renewables (R1,R2), one storage system (S1), satisfying local load and exchanging power with the grid



http://www.e-price-project.eu/

Example: Power dispatch model

- Conventional generator model (i=1,2,3)

power generated at next time unit

$$P_{i,k+1} = P_{i,k} + \Delta P_{i,k}$$

constraints on generated power: $P_{i,\min} \le P_{i,k} \le P_{i,\max}$

constraints on power variation: $\Delta P_{i,\min} \le \Delta P_{i,k} \le \Delta P_{i,\max}$

- Storage model

$$S_{k+1} = \alpha S_k + \alpha_c u_{c,k} - \frac{1}{\alpha_d} u_{d,k}$$

α = self discharge loss

α_c = charge efficiency

α_d = discharge efficiency

constraints on charge/discharge: $S_{\min} \le S_k \le S_{\max}$

constraints on charge/discharge rate: $\Delta_{S,\min} \le S_{k+1} - S_k \le \Delta_{S,\max}$

constraints on power flows: $0 \le u_{c,k} \le u_{c,\max}, \;\; 0 \le u_{d,k} \le u_{d,\max}$

Example: Power dispatch model

- Power exchanged with the rest of the grid (=balance)

$$P_{\text{ex},k} = P_{1,k} + P_{2,k} + P_{3,k} - u_{c,k} + u_{d,k} + r_k - l_k$$

conventional power

storage charge/discharge

renewable power

load

- Overall linear model and constraints

$$x = \begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ S \end{bmatrix} \qquad u = \begin{bmatrix} \Delta P_1 \\ \Delta P_2 \\ \Delta P_3 \\ u_c \\ u_d \\ r - l \end{bmatrix} \qquad y = \begin{bmatrix} P_{\text{ex}} \; P_1 \\ P_2 \\ P_3 \\ S \\ \Delta S \end{bmatrix}$$

uncontrolled input

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \alpha \end{bmatrix} \qquad B = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \alpha_c & -\frac{1}{\alpha_d} & 0 \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \alpha - 1 \end{bmatrix} \qquad D = \begin{bmatrix} 0 & 0 & 0 & -1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \alpha_c & -\frac{1}{\alpha_d} & 0 \end{bmatrix}$$

Example: Power dispatch model

- Cost function: terms to penalize

$$\min \sum_{k=0}^{N-1} \gamma(P_{\text{ex},k} - E_k)^2 + (a_i P_{i,k}^2 + b_i P_{i,k} + c_i) - p_k[P_{\text{ex},k} - E_k]$$

electricity price on RT market

penalty on deviation from E-program

production cost from conventional plants

price to pay to get power from RT market

$E_k = 0$, $\gamma = 0$ if no E-Program is agreed on the day-ahead market

- The overall linear MPC problem maps into a QP:

$$\min_z \quad \frac{1}{2} z' H z + \left(F \begin{bmatrix} x_0 \\ r-l \\ E \end{bmatrix} + c \right) z + d$$

$$\text{s.t.} \quad Gz \leq W + S \begin{bmatrix} x_0 \\ r-l \\ E \end{bmatrix}$$

x_0 = current state
$r - l$ = predicted renewable power - load
E = E-program

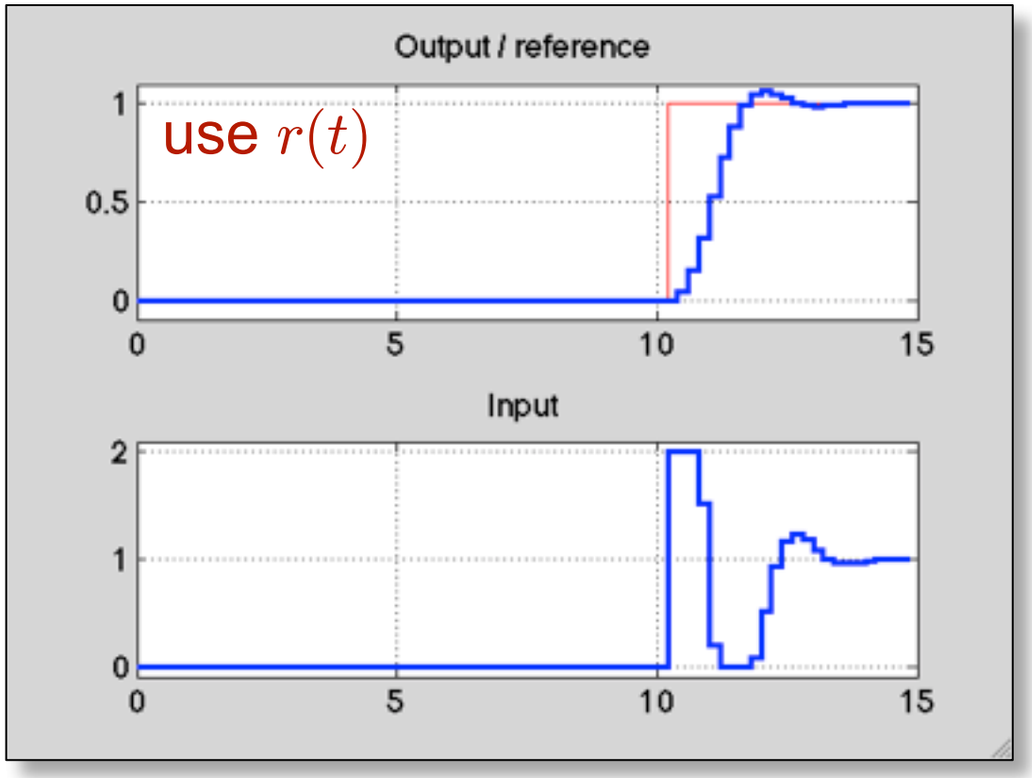
$$z = \left\{ \Delta P_{i,k}, u_{c,k}, u_{d,k} \right\}_{k=0}^{N-1}$$

Anticipative action (a.k.a. “preview”)

A main advantage of MPC is its ability to take future information of the reference/disturbance signals into account for better tracking error y_k-r_k:

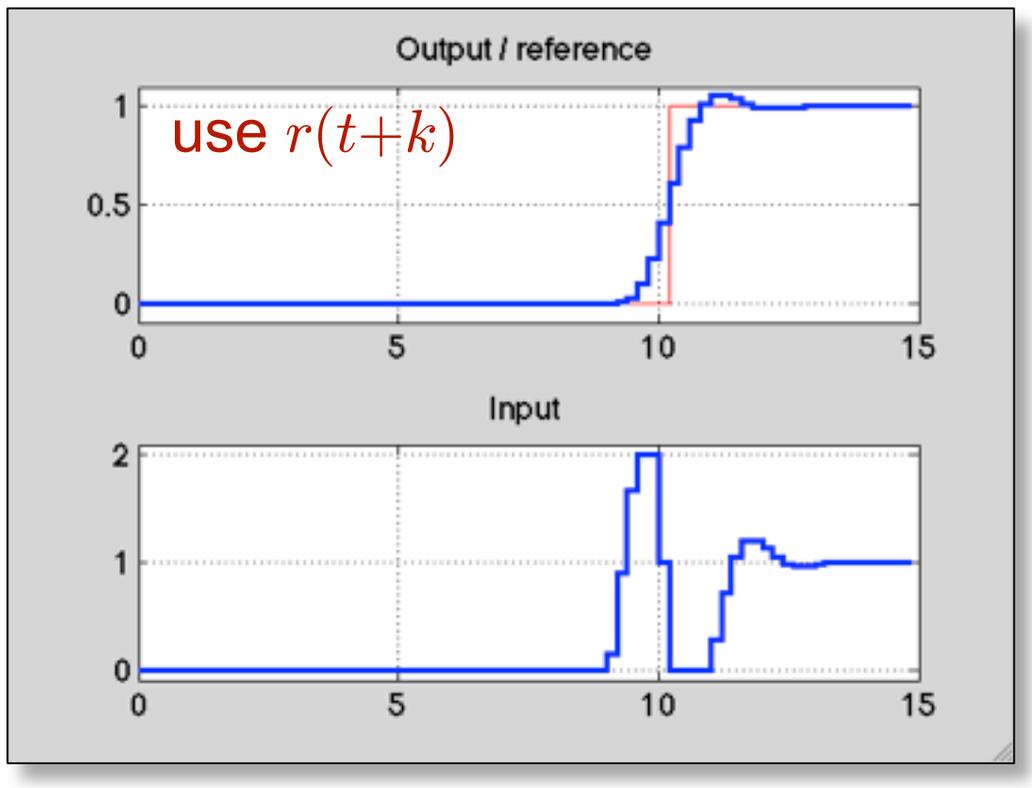
- Reference not known in advance:

$$r_k \equiv r(t), \forall k = 0, \ldots, N-1$$



- Future reference samples (partially) known in advance:

$$r_k = r(t+k), \forall k = 0, \ldots, N-1$$



go to demo **`mpcpreview.m`** (MPC-Tbx)

QP feasibility and soft constraints

- To prevent QP infeasibility, relax output constraints:

$$
\begin{aligned}
\min_{z,\epsilon} \quad & J(z, x_0) + \rho_\epsilon \epsilon^2 \\
\text{s.t.} \quad & y_{\min} - \epsilon V_{\min} \leq y_k \leq y_{\max} + \epsilon V_{\max}, \;\; k = 0, \ldots, N-1 \\
& u_{\min} \leq u_k \leq u_{\max}, \;\; k = 0, \ldots, N-1
\end{aligned}
\qquad \rho_\epsilon \gg 1
$$

ϵ = “panic” variable

$V_{\min}$, $V_{\max}$ = vectors with entries ≥ 0

the larger the i-th entry of vector $V_{\min}$ ($V_{\max}$), the relatively softer the corresponding constraint

- Infeasibility can be due to:
  - modeling errors, disturbances
  - wrong MPC setup (e.g., prediction horizon is too short)

Good models for (MPC) control

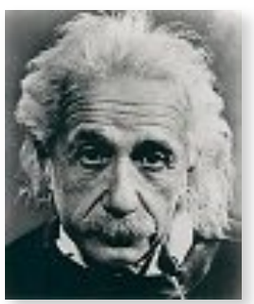
Computational complexity and performance properties depend on chosen model/objective/constraints

Good models for MPC:

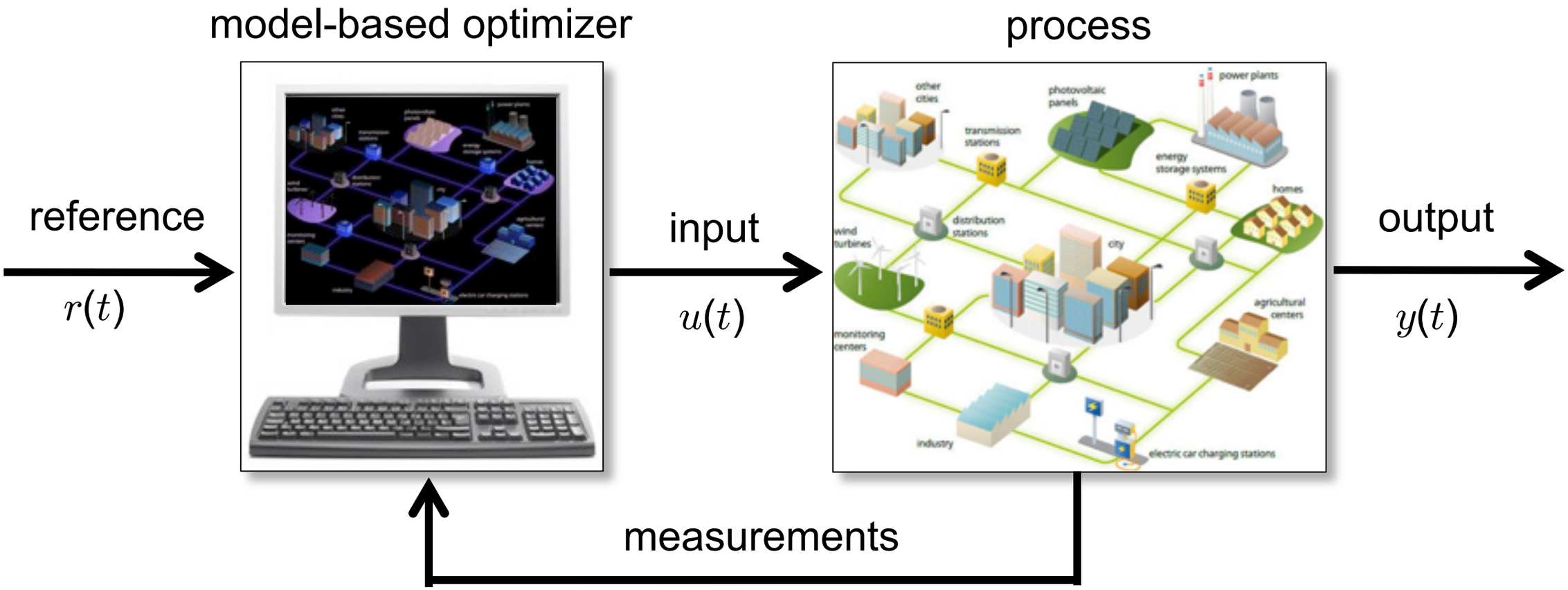
- **Descriptive** enough to capture the most significant dynamics of the system

- **Simple** enough for solving the optimization problem

"Make everything as simple as possible, but not simpler."
— Albert Einstein



Model Predictive Control (MPC)



rough

likely

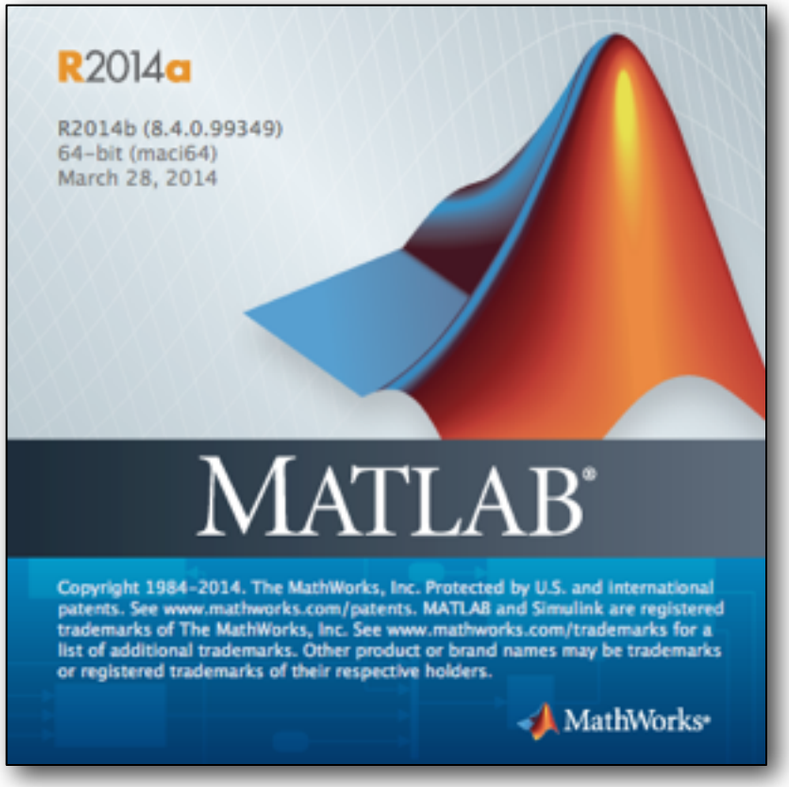
*Use a dynamical **model** of the process to **predict** its possible future evolution and choose the ~~"best"~~ **control** action*

a very good

Model Predictive Control Toolbox

(Bemporad, Ricker, Morari, 1998-2014)

- **MPC Toolbox 5.0** (The Mathworks, Inc.)
  - Linear MPC, explicit MPC, adaptive MPC
  - MPC Simulink library
  - MPC Graphical User Interface
  - Code generation [RTW, xPC Target, dSpace, etc.]
  - Linked to OPC Toolbox, System ID Toolbox, ...



http://www.mathworks.com/products/mpc/

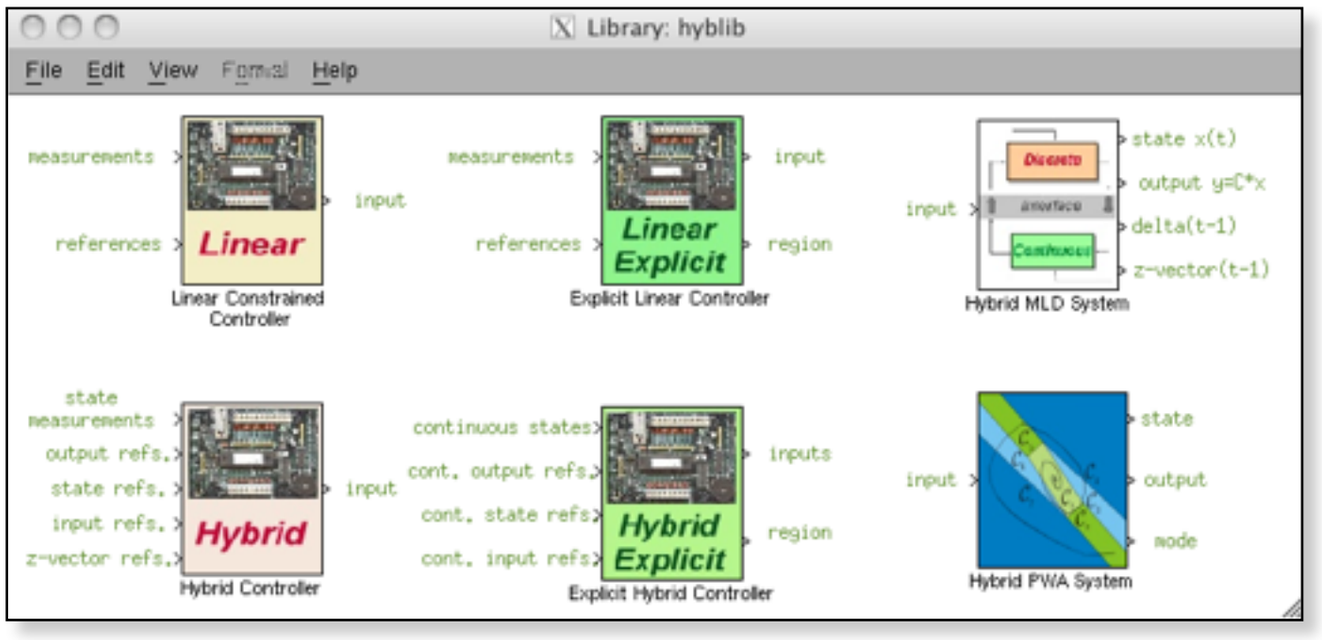
Hybrid Toolbox for MATLAB

(Bemporad, 2003-2014)

Features:

- **Explicit** MPC control (via multi-parametric programming) and C-code generation
- **Hybrid** models (binary variables and logic constraints): design, simulation, verification

Supported by



5700+ download requests since October 2004

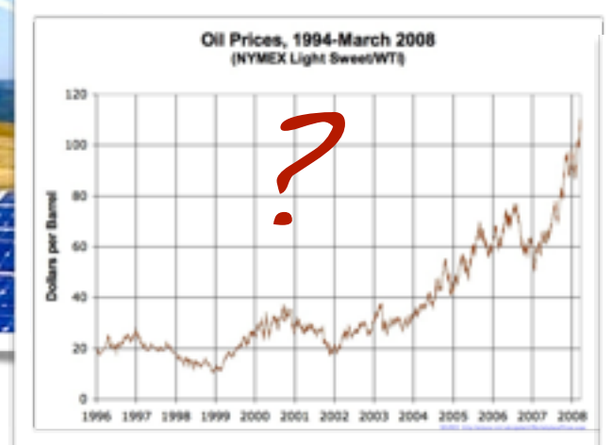
http://cse.lab.imtlucca.it/~bemporad/hybrid/toolbox/

Stochastic Model Predictive Control

Optimize decisions under uncertainty

- In many control problems decisions must be taken under **uncertainty**


renewable power



prices

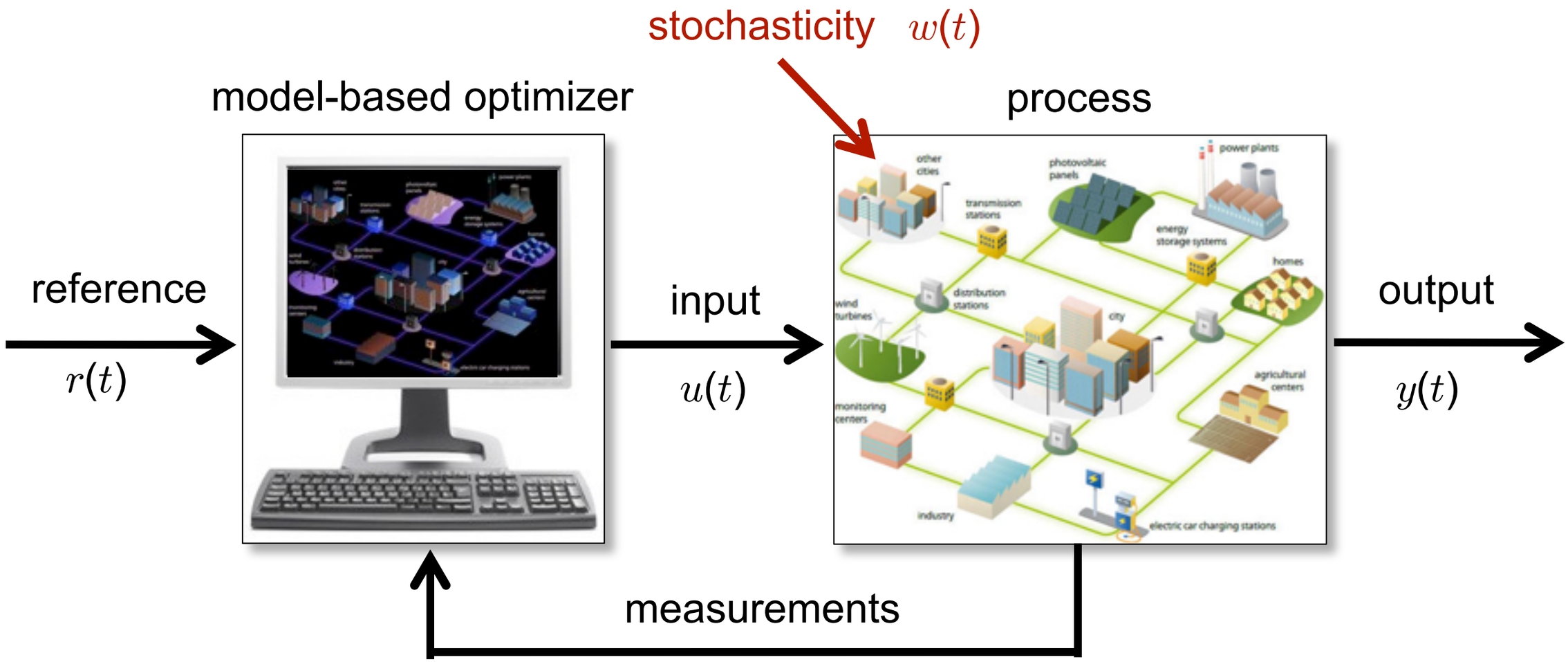

water


demand


human (inter)action

- **Robust** control approaches do not model uncertainty (only assume that is bounded) and pessimistically consider the worst case
- **Stochastic models** provide instead additional information about uncertainty
- **Optimality** often sought (ex: minimize expected economic cost)

Stochastic Model Predictive Control (SMPC)



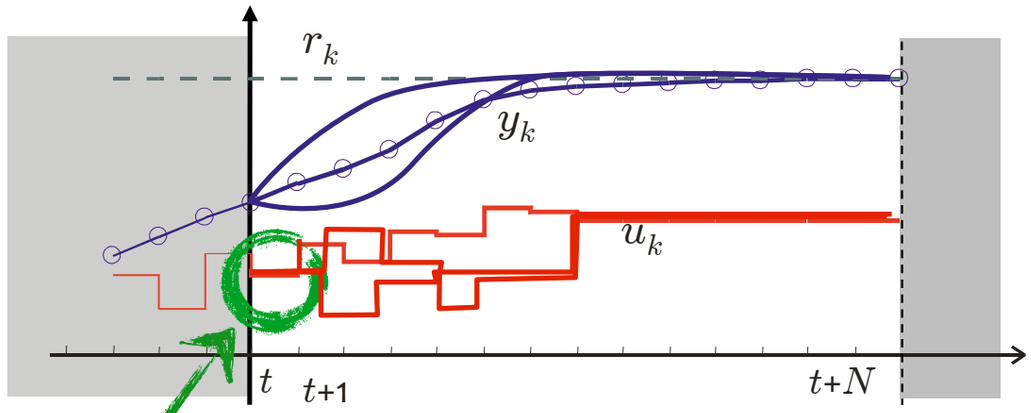
*Use a **stochastic** dynamical **model** of the process to **predict** its possible future evolutions and choose the "best" **control** action*

Stochastic Model Predictive Control

- At time t: solve **stochastic optimal control** problem over future horizon $[t,t+N]$

$$\min_{u} \quad E_w\left[\sum_{k=0}^{N-1} \ell(y_k, u_k, w_k)\right]$$

s.t.
$$x_{k+1} = A(w_k)x_k + B(w_k)u_k + f(w_k)$$
$$y_k = C(w_k)x_k + D(w_k)u_k + g(w_k)$$
$$u_{\min} \le u_k \le u_{\max}$$
$$y_{\min} \le y_k \le y_{\max}, \forall w \quad \text{robustness}$$
$$x_0 = x(t) \quad \text{feedback}$$



$x(t)$ = process state

$u(t)$ = manipulated vars

$y(t)$ = controlled output

$w(t)$ **= stochastic disturbances**

- Solve stochastic optimal control problem w.r.t. future input sequence
- Apply the first optimal move $u(t)=u_0^*$, throw the rest of the sequence away
- At time $t+1$: Get new measurements, repeat the optimization. And so on …

A few sample applications of SMPC

- **Energy systems**: power dispatch in smart grids, optimal bidding on electricity markets

  (Patrinos, Trimboli, Bemporad 2011)

  (Puglia, Bernardini, Bemporad 2011)

- **Financial engineering**: dynamic hedging of portfolios replicating synthetic options

  (Bemporad, Bellucci, Gabbriellini, 2009)

  (Bemporad, Gabbriellini, Puglia, Bellucci, 2010)

  (Bemporad,Puglia, Gabbriellini, 2011)

- **Water networks**: pumping control in urban drinking water networks, under uncertain demand & minimizing costs under varying electricity prices

  (Sampathirao, Sopasakis, Bemporad, 2014)

- **Automotive control**: energy management in HEVs, adaptive cruise control (human-machine interaction)

  (Di Cairano, Bernardini, Bemporad, Kolmanovsky, 2014)

- **Networked control**: improve robustness against communication imperfections

  (Bernardini, Donkers, Bemporad, Heemels, NECSYS 2010)

Linear stochastic MPC w/ discrete disturbance

- **Linear stochastic** prediction model

$$\begin{cases} x_{k+1} &= A(w_k)x_k + B(w_k)u_k + f(w_k) \\ y_k &= C(w_k)x_k + D(w_k)u_k + g(w_k) \end{cases}$$

- **Discrete disturbance** $w_k \in \{w^1, \ldots, w^s\}$ $p_j = \mathbf{Pr}[w_k = w^j]$

$$p_j \geq 0, \ \sum_{j=1}^{s} p_j = 1$$

Often w_k is low-dimensional (ex: electricity price, weather, etc.)

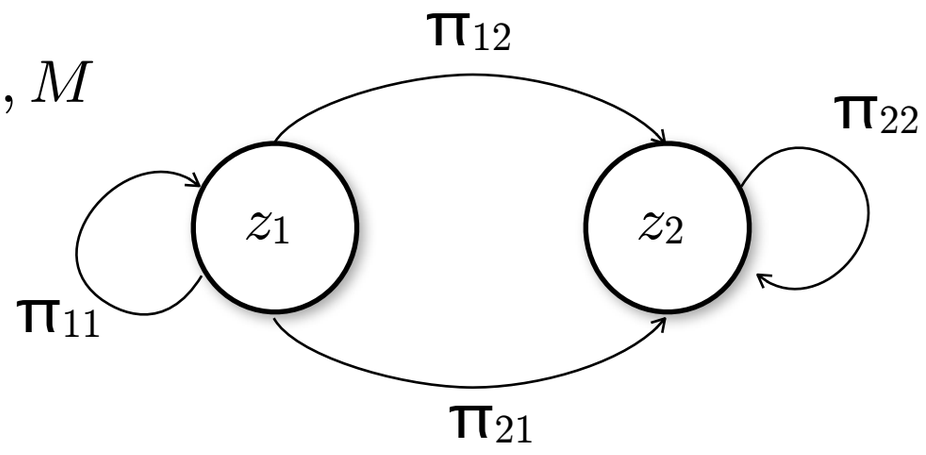
Linear stochastic MPC w/ discrete disturbance

- Probabilities $p_j(t)$ can have their own dynamics

  Example: Markov chain

$$\pi_{ih} = \mathbf{Pr}[z(t+1) = z_h \mid z(t) = z_i], \;\; i, h = 1, \ldots, M$$

$$p_j(t) = \begin{cases} e_{1j} & \text{if} \;\; z(t) = z_1 \\ \vdots & \vdots \\ e_{Mj} & \text{if} \;\; z(t) = z_M \end{cases}$$

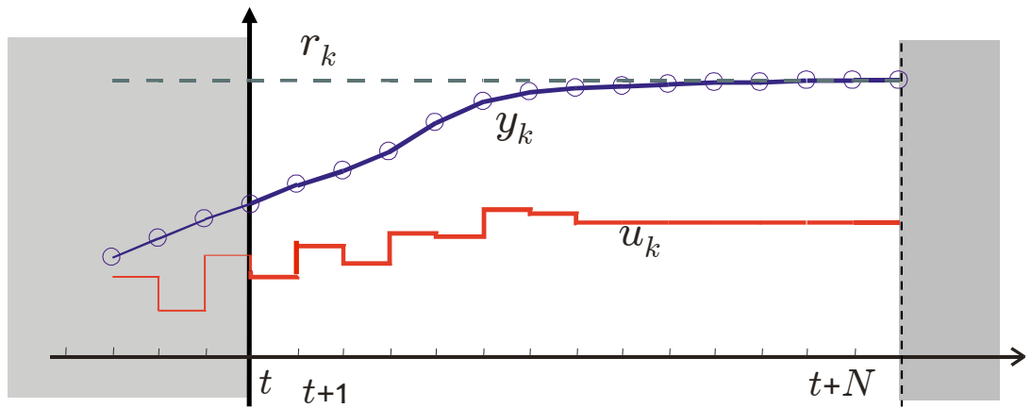


- Discrete distributions can be **estimated from historical data** (and adapted on-line)

Cost functions for SMPC to minimize

- **Expected performance**

$$\min_u \sum_{k=0}^{N-1} E_w \left[(y_k - r_k)^2 \right]$$



- Tradeoff between **expectation & risk**

$$\min_u \sum_{k=0}^{N-1} (E_w [y_k - r_k])^2 + \alpha \mathrm{Var}_w [y_k - r_k]$$

$$\alpha \geq 0$$

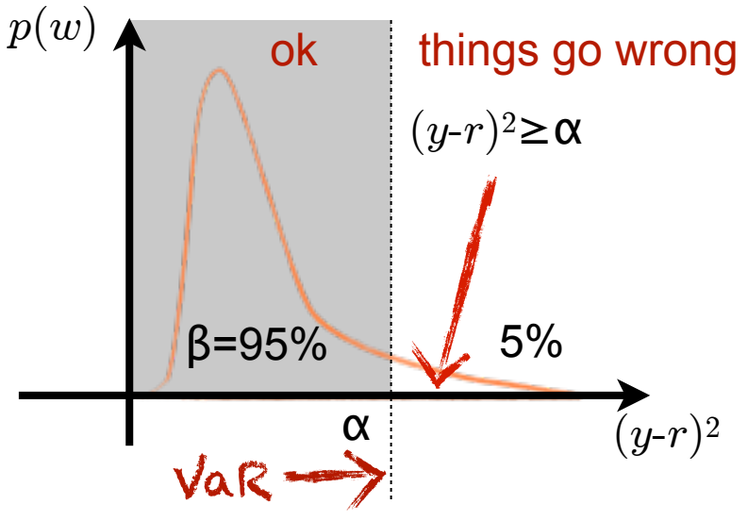
- Note that they coincide for α=1, since $\mathrm{Var}_w E [y_k - r_k] = E_w \left[(y_k - r_k)^2 \right] - (E_w [y_k - r_k])^2$

Cost functions for SMPC to minimize

- Conditional Value-at-Risk (**CVaR**) (Rockafellar, Uryasev, 2000)

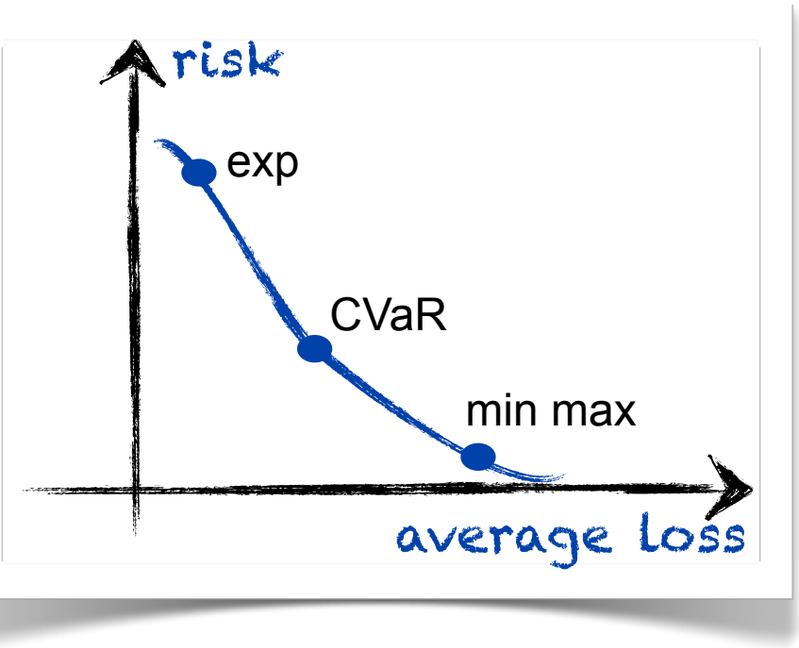
$$\min_{u,\alpha} \sum_{k=0}^{N-1} \alpha_k + \frac{1}{1-\beta} E_w \left[\max \left\{ (y_k - r_k)^2 - \alpha_k, 0 \right\} \right]$$

= minimize expected loss when things go wrong (convex !)



- **Min-max** = minimize worst case performance

$$\min_{u,\alpha} \sum_{k=0}^{N-1} \max_{w} (y_k - r(w_k))^2 + \rho u_k^2$$



Stochastic optimal control problem

- CVaR optimization

$$\min_{u,\alpha} \sum_{k=0}^{N-1} \alpha_k + \frac{1}{1-\beta} E_w \left[\max\left\{|y_k - r_k| - \alpha_k, 0\right\}\right]$$

$$\begin{aligned} \min_{u,z,\alpha} \quad & \sum_{k=0}^{N-1} \alpha_k + \frac{1}{1-\beta} \sum_{j=1}^{S} p^j z_k^j \\ \text{s.t.} \quad & z_k^j \geq y_k - r_k - \alpha_k \\ & z_k^j \geq r_k - y_k - \alpha_k \\ & z_k^j \geq 0 \end{aligned}$$

CVaR optimization becomes a linear programming problem

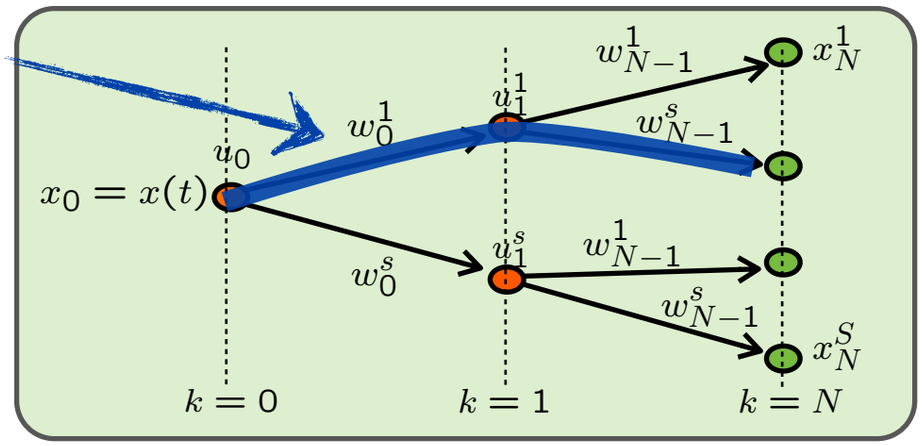
Stochastic optimal control problem

- Enumerate all possible scenarios $\{w_0^j, w_1^j, \ldots, w_{N-1}^j\},\ j = 1, \ldots, S$

  scenario = path on the tree

  number S of scenarios = number of leaf nodes

scenario #j



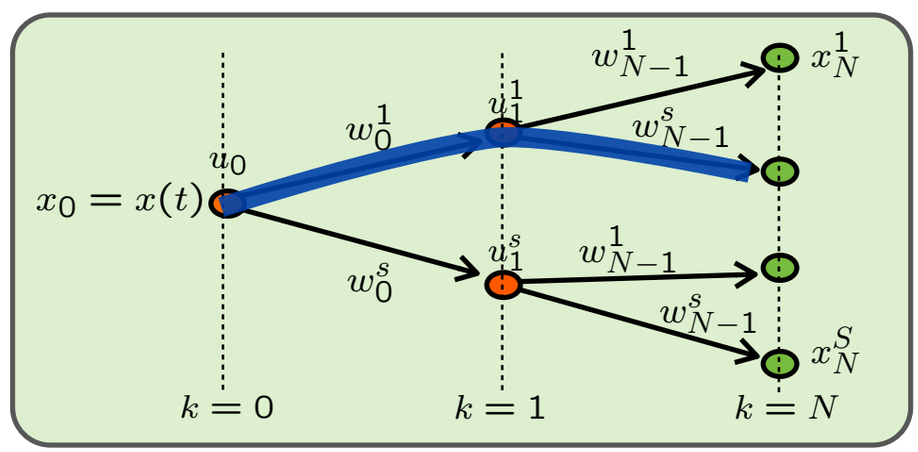
- Each scenario has probability $p^j = \prod_{k=0}^{N-1} \mathbf{Pr}[w_k = w_k^j]$

Stochastic optimal control problem

- Each scenario has its own evolution

$$x_{k+1}^j = A(w_k^j)x_k^j + B(w_k^j)u_k^j + f(w_k^j)$$

(=linear time-varying system)



- Expectations become simple sums !

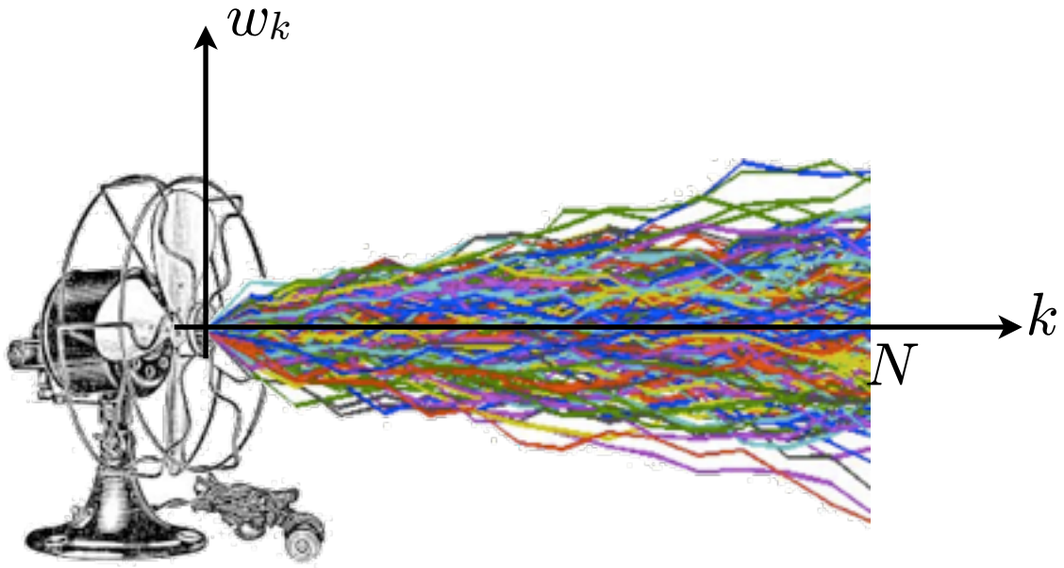
Ex: $\min E_w \left[x_N' P x_N + \sum_{k=0}^{N-1} x_k' Q x_k + u_k' R u_k \right]$

$$\min \sum_{j=1}^{S} p^j \left((x_N^j)' P x_N^j + \sum_{k=0}^{N-1} (x_k^j)' Q x_k^j + (u_k^j)' R u_k^j \right)$$

Expectations of quadratic costs remain quadratic costs

Scenario tree generation from data

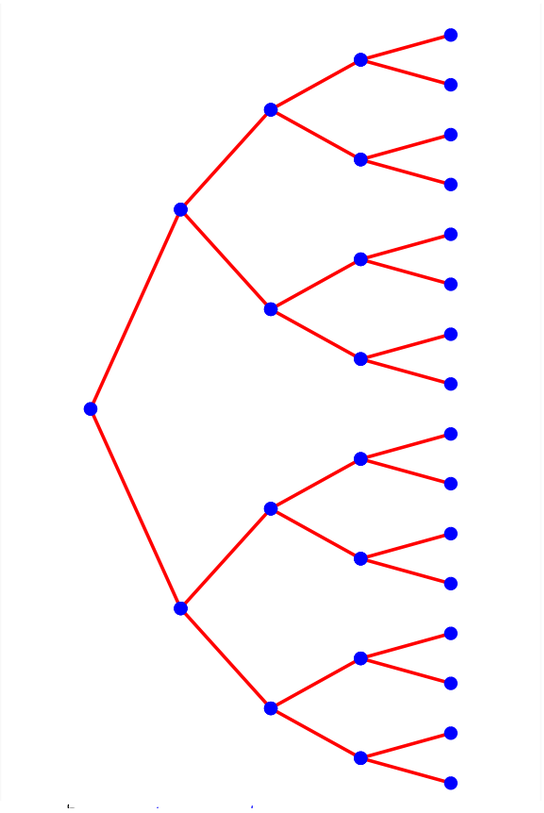
- Scenario trees can be generated by **clustering** sample paths
- Paths can be obtained by **Monte Carlo simulation** of (estimated) models, or from **historical data**
- The **number of nodes** can be decided a priori



scenario "fan" (collection of sample paths)

Heuristic
Multilevel
Clustering

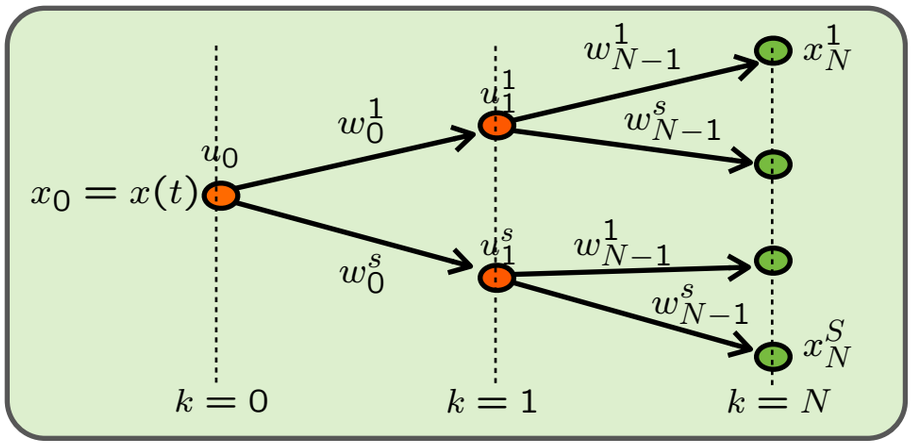
(Heitsch, Römisch, 2009)



scenario tree

- Alternative (simpler/less accurate) approach: **k-means** clustering

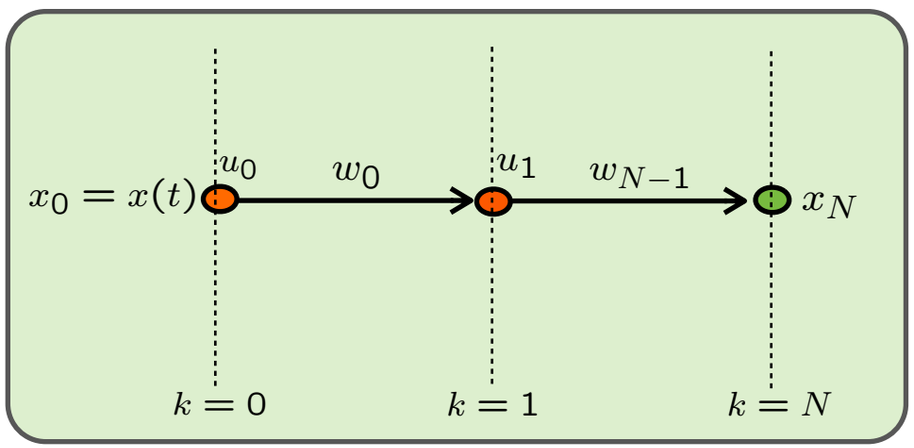
Free control variables



Stochastic optimal control

Causality constraint: $u_k^j = u_k^h$ when scenarios j and h share the same node at prediction time k (for example: $u_0^j \equiv u_0^h$ at root node $k{=}0$)

Decision u_k only depends on past disturbance realizations $\{w_0, w_1, \ldots, w_{k-1}\}$

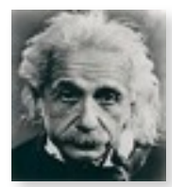


Deterministic control

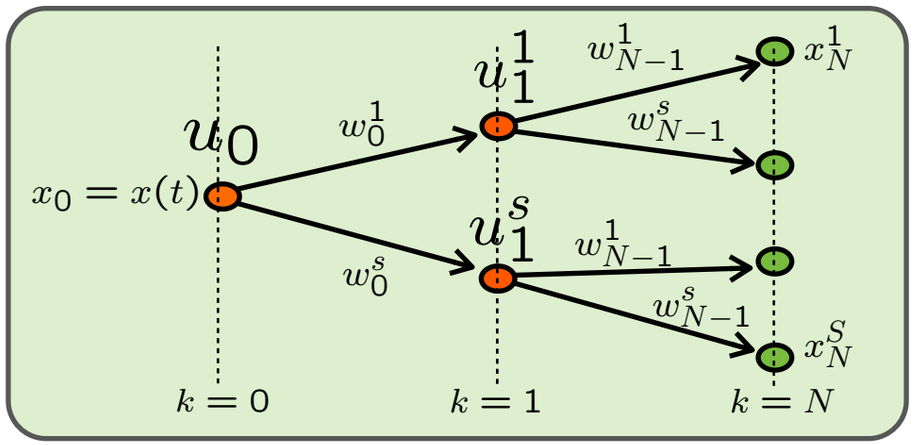
Only a sequence of disturbances is considered

- frozen-time: $w_k \equiv w(t), \ \forall k$ (causal prediction)
- prescient control: $w_k \equiv w(t+k)$ (non-causal)
- certainty equivalence: $w_k = E[w(t+k)|t]$ (causal)

Trade off between **complexity** of optimization problem (=number of nodes) and **performance** (=accuracy of stochastic modeling)

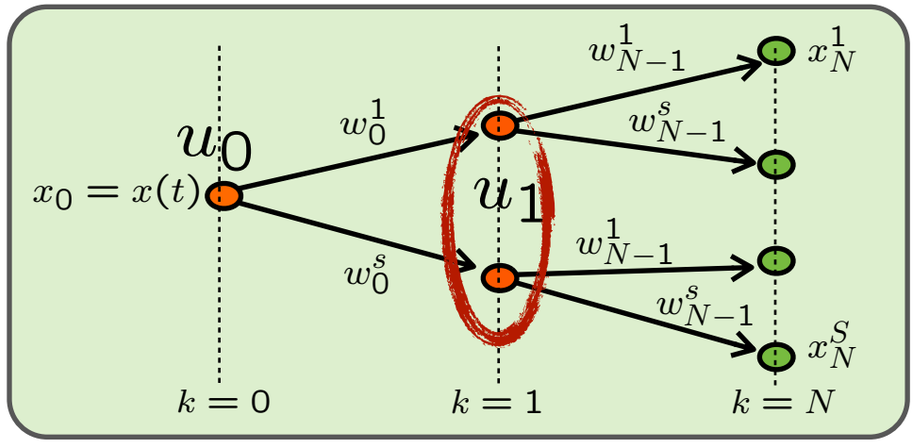


Open-loop vs. closed-loop prediction



closed-loop prediction

A proper move u is optimized to counteract each possible outcome of the disturbance w



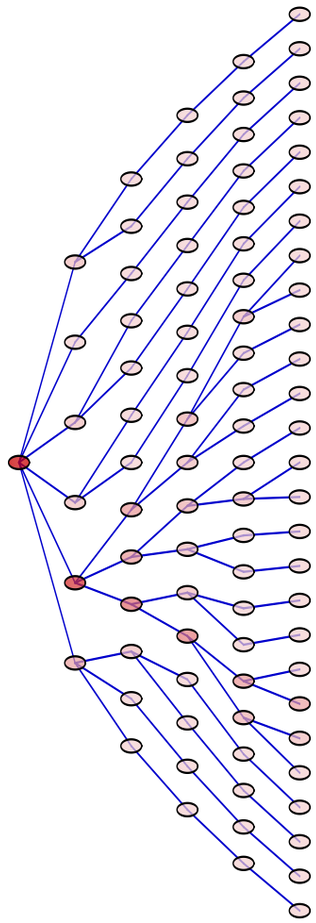
open-loop prediction

Only a sequence of inputs $\{u_0, u_1, \ldots, u_{N-1}\}$ is optimized, the same u must be good for all possible disturbance w

- Intuitively: OL prediction is more conservative than CL in handling constraints
- OL problem = CL problem + additional constraints $u^j \equiv u, \ \forall j = 1, \ldots, S$ (=less degrees of freedom)

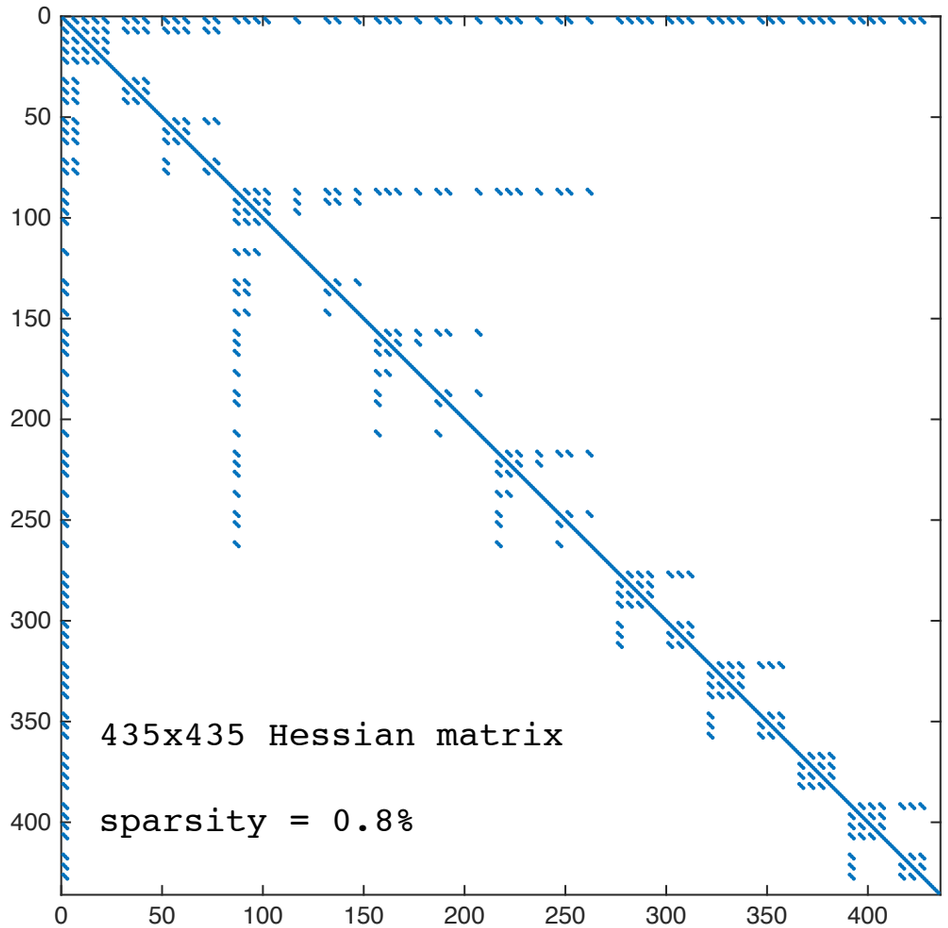
Complexity of stochastic optimization problem

- #optimization variables = #nodes x #inputs (in condensed version)
- Problems are **very sparse** (well exploited by **interior point methods**)
- Example: SMPC with quadratic cost and linear constraints

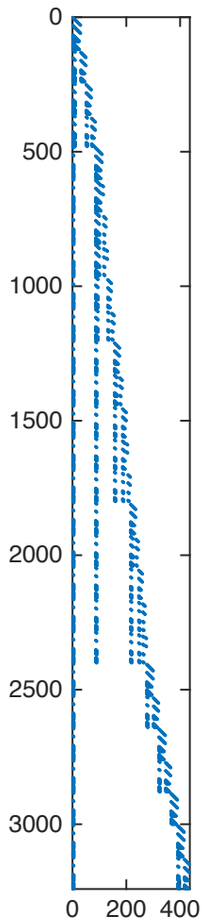


Tree=87 nodes

Branching factor M=[6 3 2 2 2]



435 free variables (5 inputs x node)



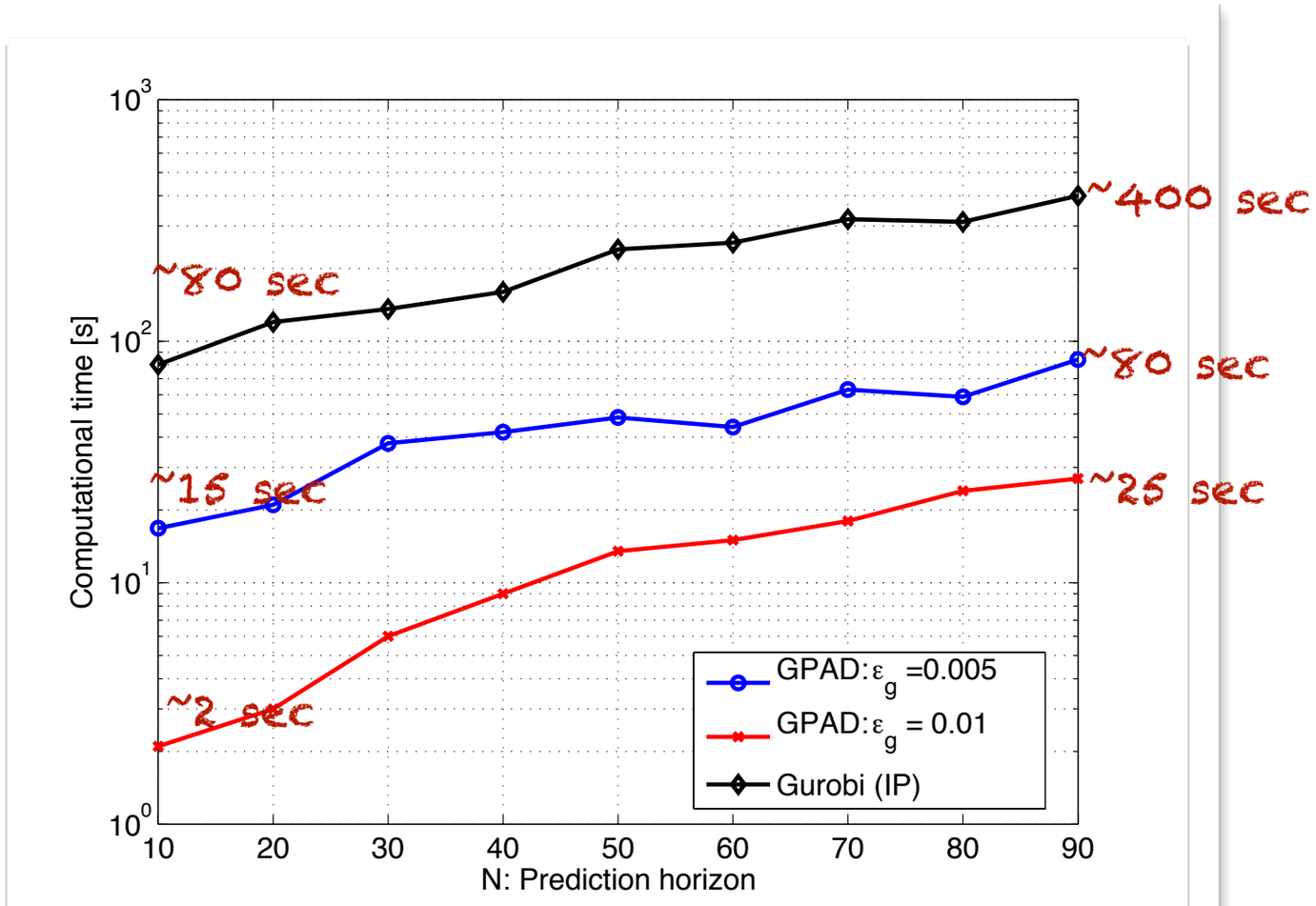
3240x435 constraint matrix
sparsity = 1.1%

Distributed GPAD for stochastic MPC

- A distributed (**parallelized**) variant of the **A**ccelerated **G**radient **P**rojection applied to **D**ual (GPAD) for solving SMPC problems is available

(Sampathirao, Sopasakis, Bemporad, 2014)

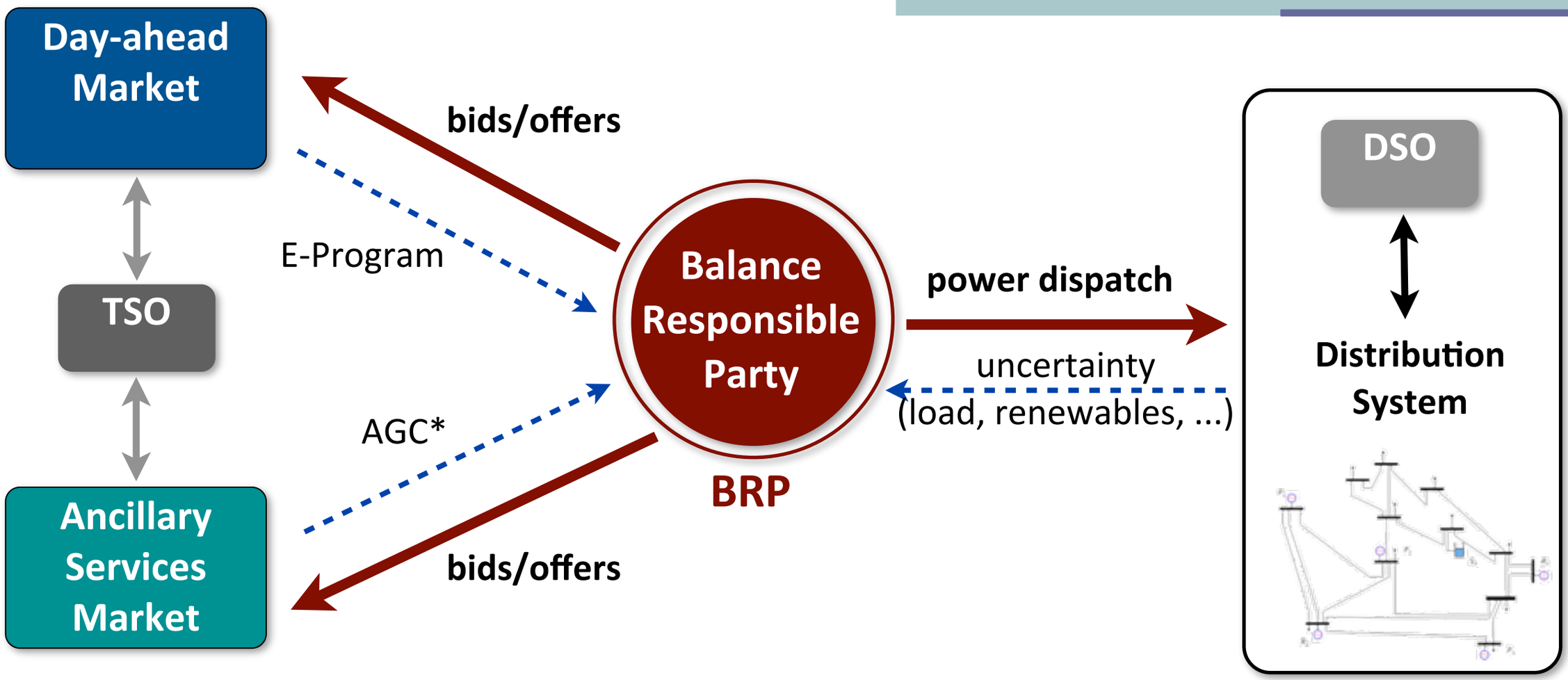
Example: stochastic MPC with **60 states, 25 inputs, 256 scenarios**



16x ÷ 40x faster
than commercial state-
of-the-art
Interior-Point method

Remark: For larger problems (*e.g.*, *50* states, *30* inputs, *9036* nodes) GUROBI gets stuck on a 4GB 4-core PC, while dGPAD can solve the problem

SMPC for operating on energy markets

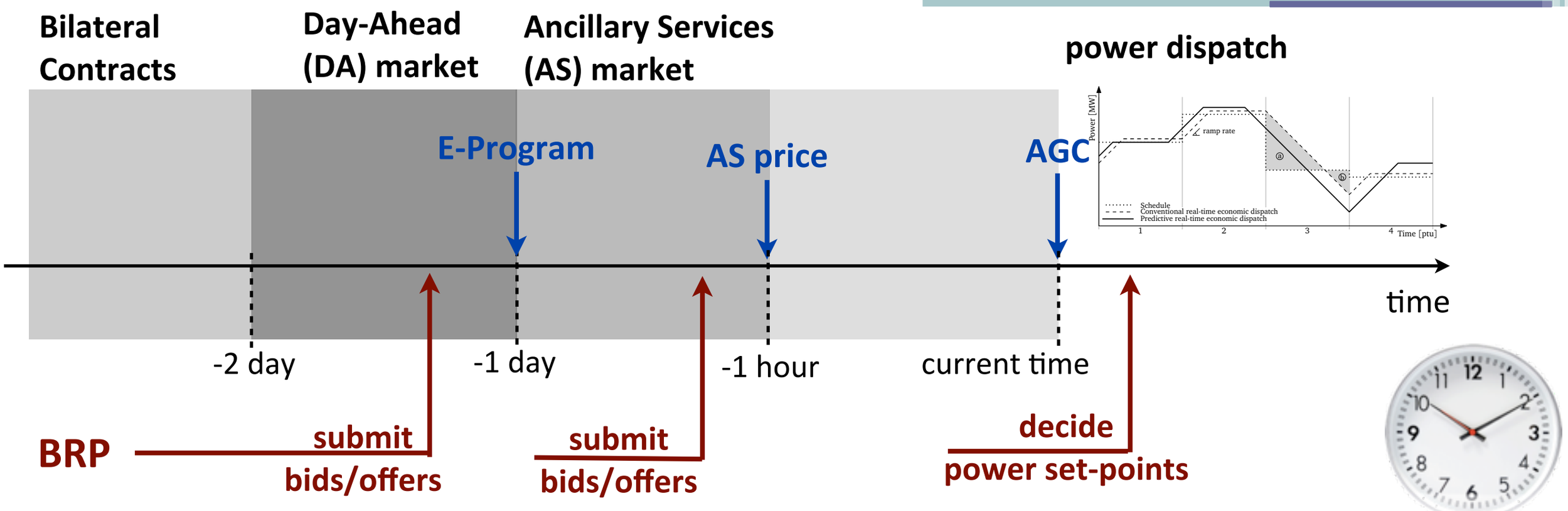


Objective: maximize BRP's profit !

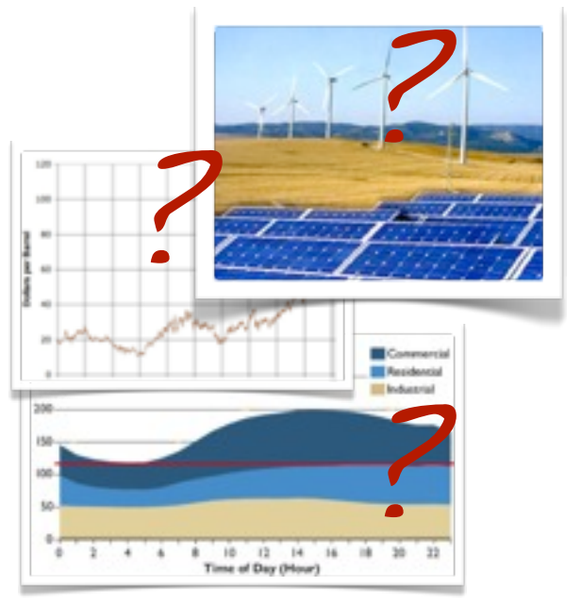
(while satisfying local power demand and all grid constraints)

*AGC = Automatic Generation Control

Operations of energy market players (BRPs)



- We can use SMPC strategies for solving:
  - The power dispatch problem
  - The two bidding problems

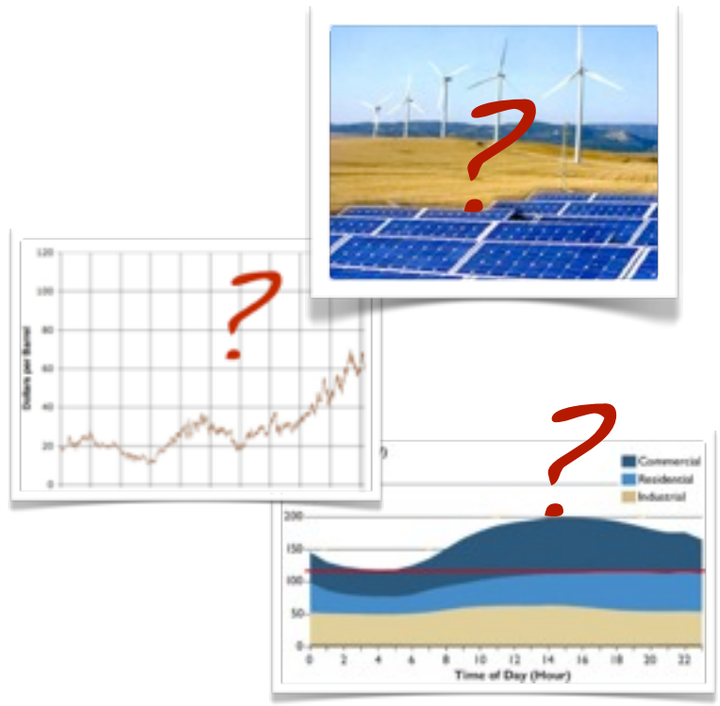


- The decision process is heavily affected by **uncertainty** (prices, load, renewables, ...)

Real-time power dispatch problem

- **Balance Responsible Parties (BRPs)** participate to the various energy markets and trade electricity to satisfy their loads and make profits

- Optimal control of BRPs is challenging, as in **real-time** a BRP must
  - **fulfill its E-Program**, despite perturbations induced by **uncertainties**
  - **intermittent generation** from renewable sources
  - **time-varying** internal loads
  - **react to signals** arriving from the TSO
  - **frequency deviations, AS bids** activated by the TSO
  - **minimize** generation and imbalance **costs**
  - **time-varying**, **stochastic** imbalance prices
  - consider plant **dynamics** and satisfy **constraints**
  - bounds on power output, **ramp-rate constraints**



SMPC for real-time market-based power dispatch

(Patrinos, Trimboli, Bemporad, CDC'11)

- We are a legal entity (BRP) trading on the energy (PX) and ancillary service (AS) markets

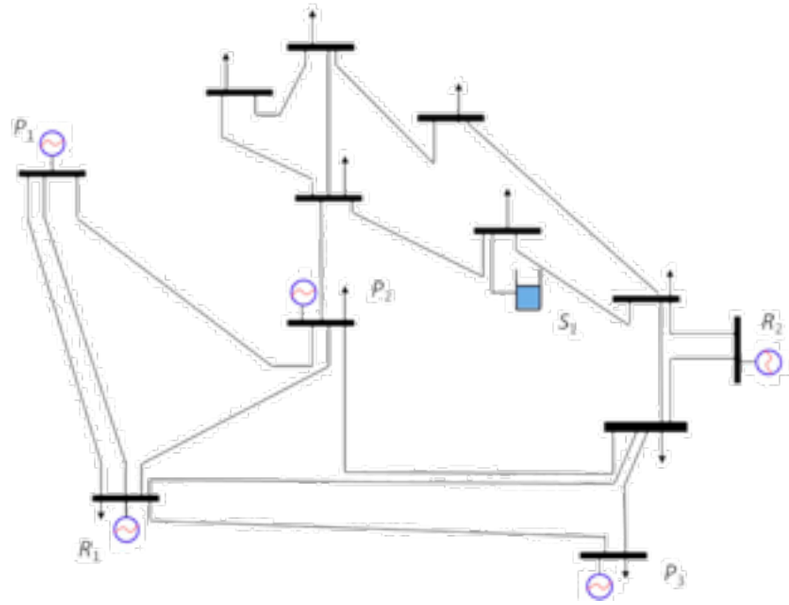
- **Objective**: Minimize costs via efficient use of intermittent resources, and maximize profits by trading on electricity (PX, AS) markets

- **Constraints**: Grid capacity, rate limits, load balancing, AS balancing

SMPC for market-based optimal power dispatch

- Power exchanged with the grid

$$P_{\text{ex},k} = P_{1,k} + P_{2,k} + P_{3,k} - u_{c,k} + u_{d,k} + (r_k - l_k)$$



stochastic

variables

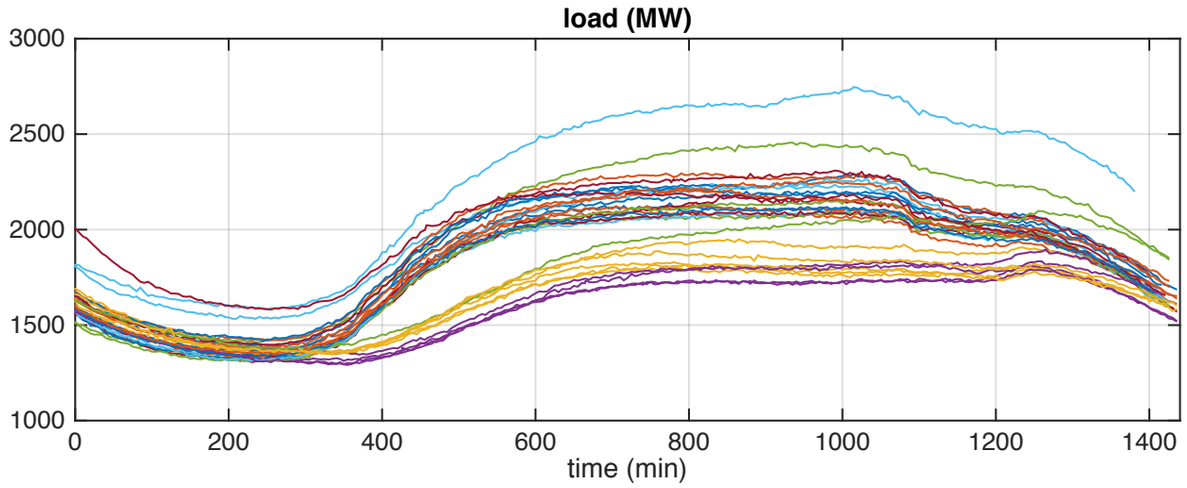
- Cost function to minimize

$$\min \sum_{k=0}^{N-1} \gamma(P_{\text{ex},k} - E_k)^2 + (a_i P_{i,k}^2 + b_i P_{i,k} + c_i) - p_k P_{\text{ex},k}$$

SMPC for market-based optimal power dispatch

- Historical data of load (MW)

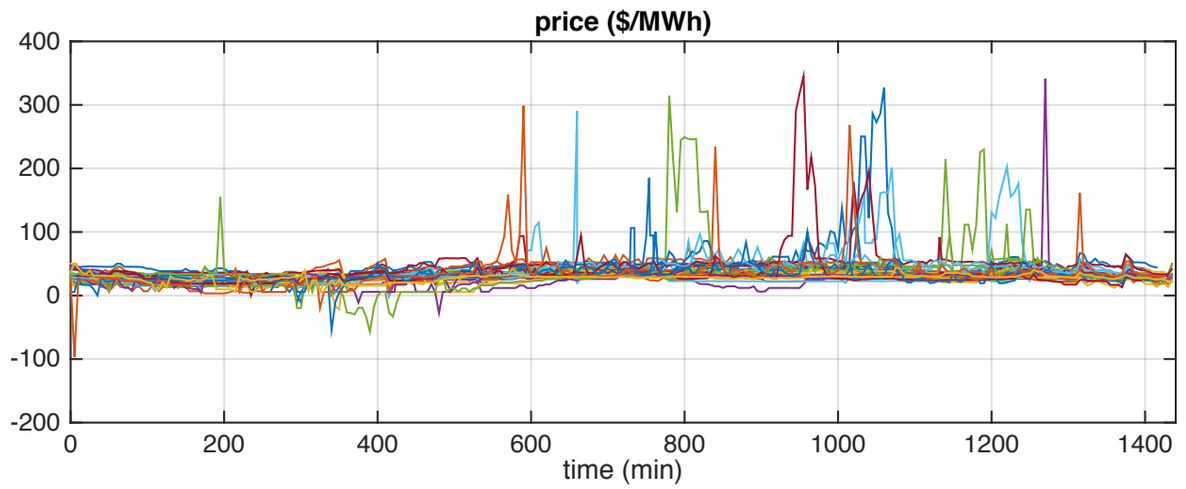
  load = 1/3 load of N.Y.C. district
  (daily data of 1-31 May 2014,
  sampling time = 5 min)



http://www.nyiso.com/public/markets_operations/market_data/load_data/index.jsp

- Historical data of price (MW)

  electricity price of N.Y.C. district
  (daily data of 1-31 May 2014,
  sampling time = 5 min)

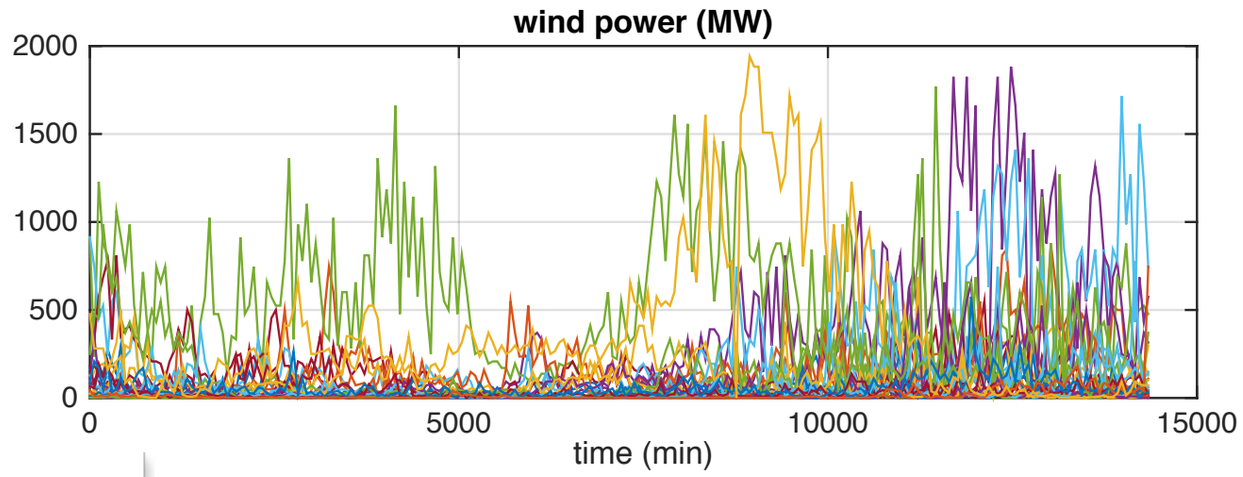


http://www.nyiso.com/public/markets_operations/market_data/pricing_data/index.jsp

SMPC for market-based optimal power dispatch

- Historical data of wind speed (m/s)

  Station BGNN4 (NY)
  (daily data of 1-31 May 2014,
  sampling time = 6 min)



wind power proportional to cubic wind velocity

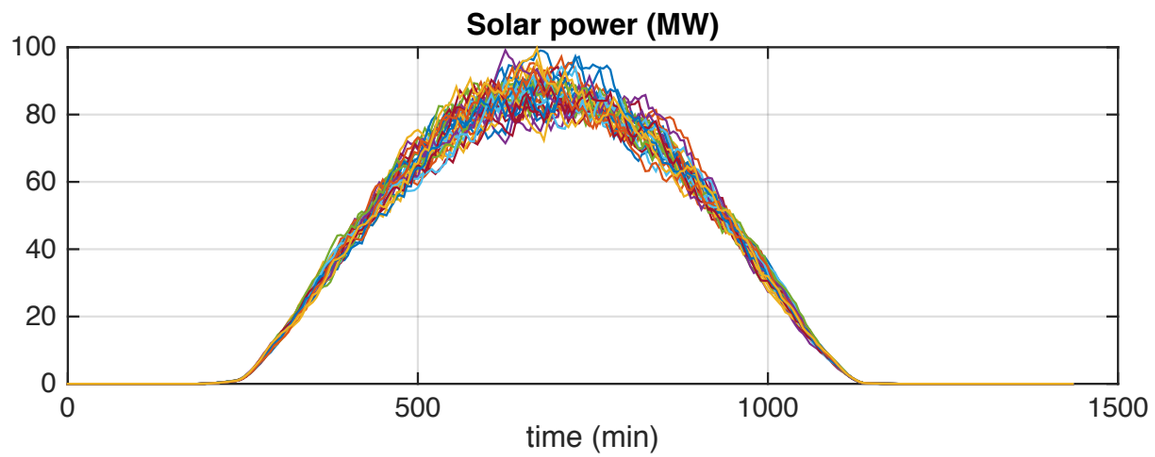
$$P_w = k v_w^3$$

http://www.ndbc.noaa.gov/station_history.php?station=bgnn4

- Historical data of solar irradiation (W/m^2)

  NY Central Park, daily data of
  1-31 May 1991-2005, sampling time = 1 h

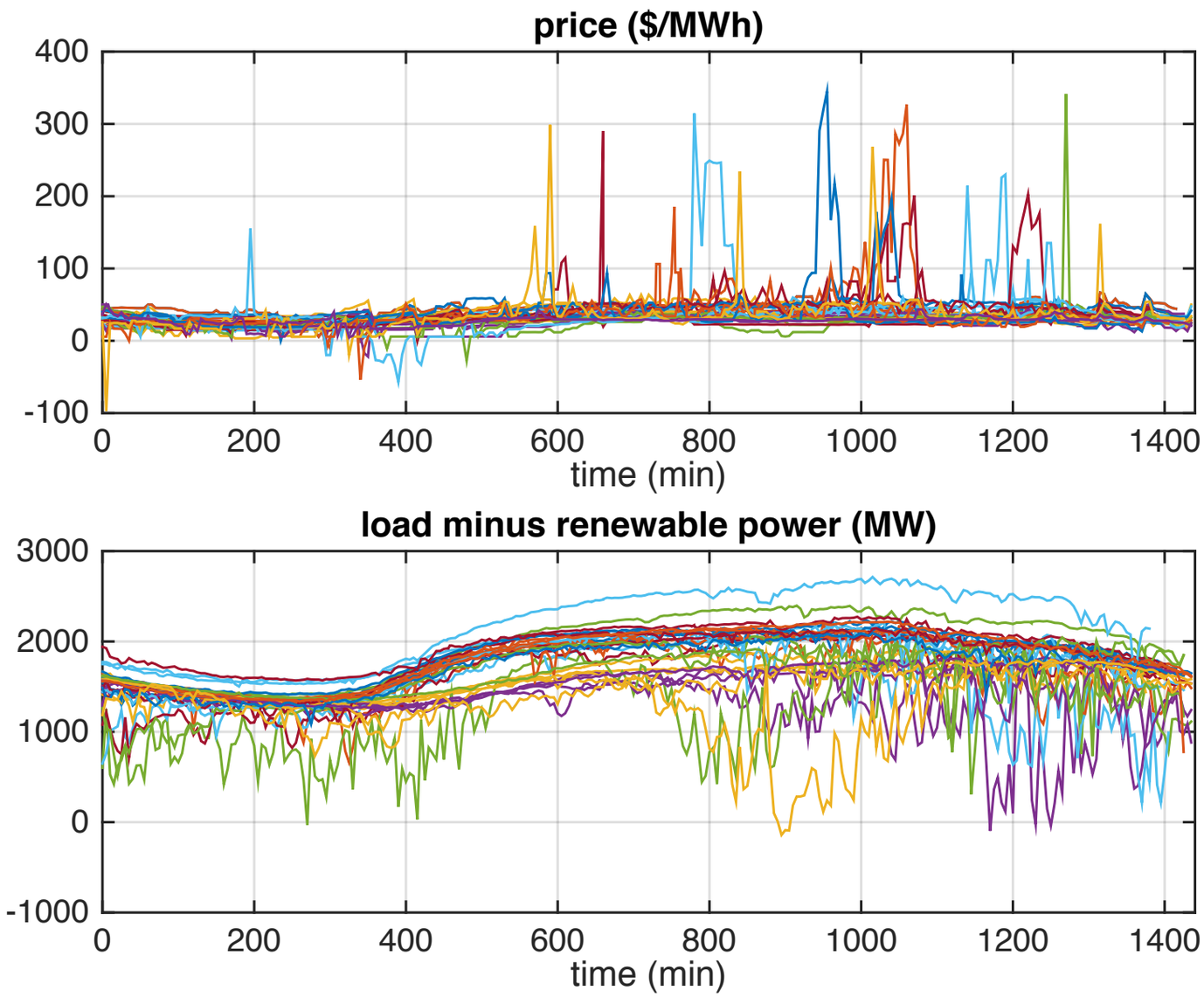
  Data perturbed by noise to mimic
  account cloud coefficient (unavailable)



http://en.openei.org/datasets/files/39/pub/725033.tar.gz

SMPC for market-based optimal power dispatch

- Historical data of overall uncertainty



SMPC for market-based optimal power dispatch

- Data used for scenario generation (31 days):

value at time t+k in scenario #j

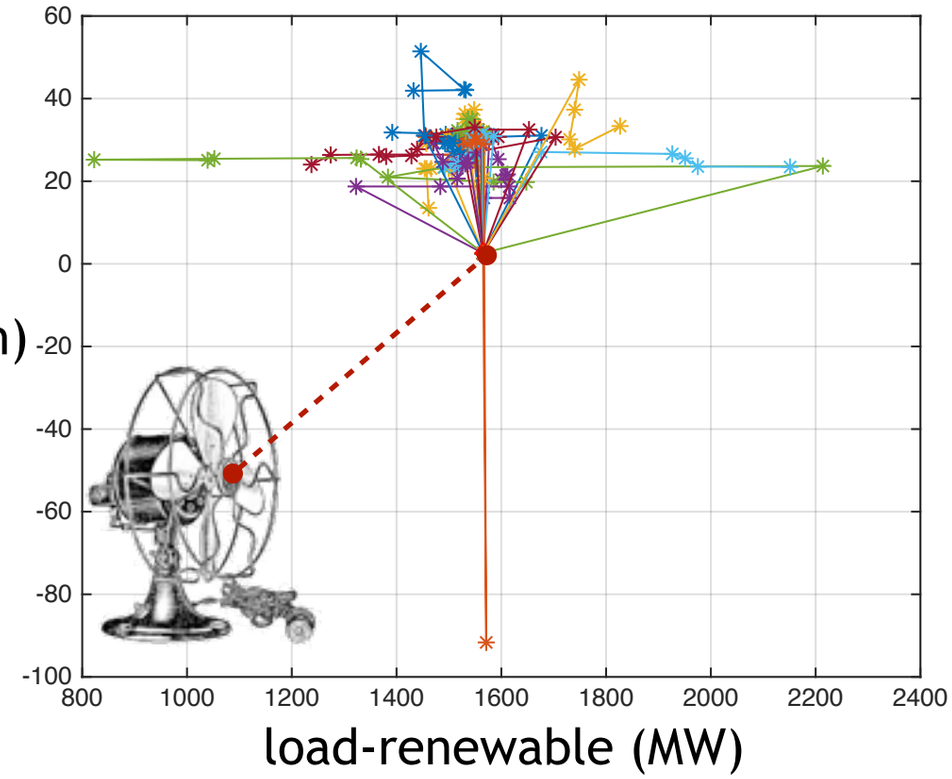
initial value at time t in scenario #j

$$w^j(t+k) = v^j(t+k) - v^j(t) + v(t)$$

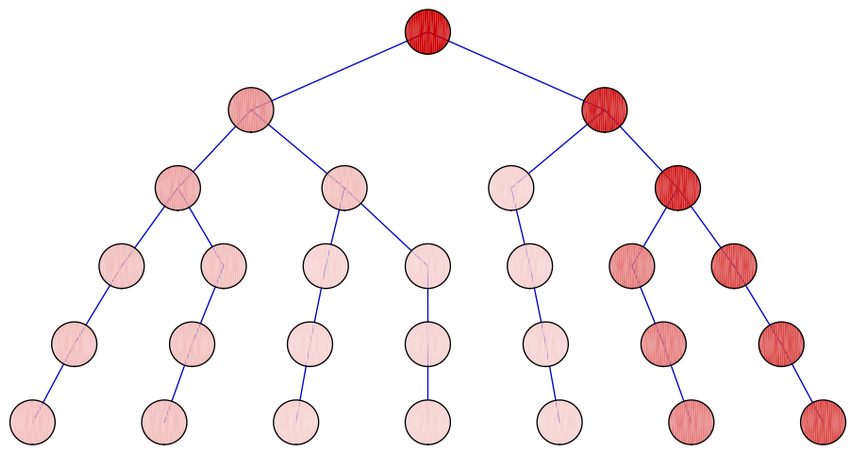
$$w^j(t) = v(t)$$

stochastic vector on scenario #j

actual value at time t



Heuristic Multilevel Clustering



28 nodes

- Tree obtained from 31 scenarios (branching factor M=[2 2 2 1 1])

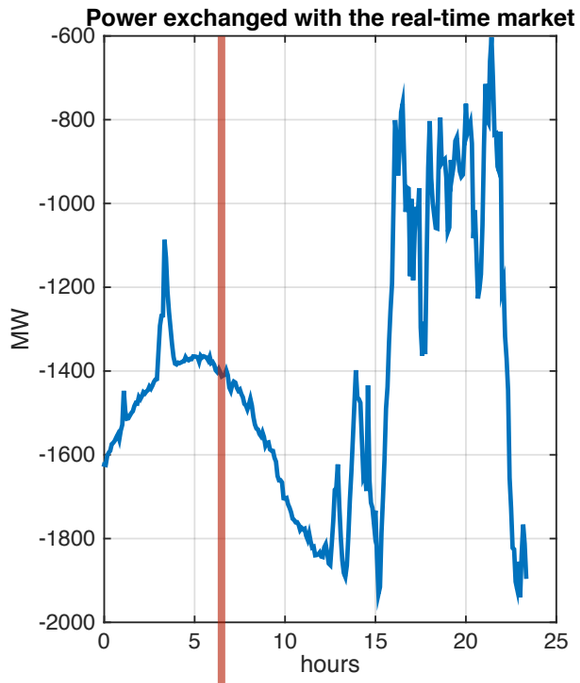
SMPC for market-based optimal power dispatch

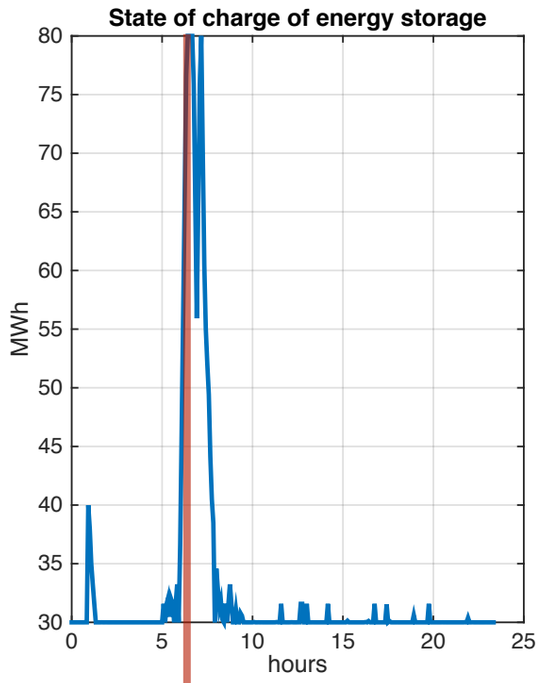
- MPC setup:
  - Sampling time: T_s=5 min
  - Prediction horizon: N=6 steps (=1/2 hour ahead)
  - Three controller options:

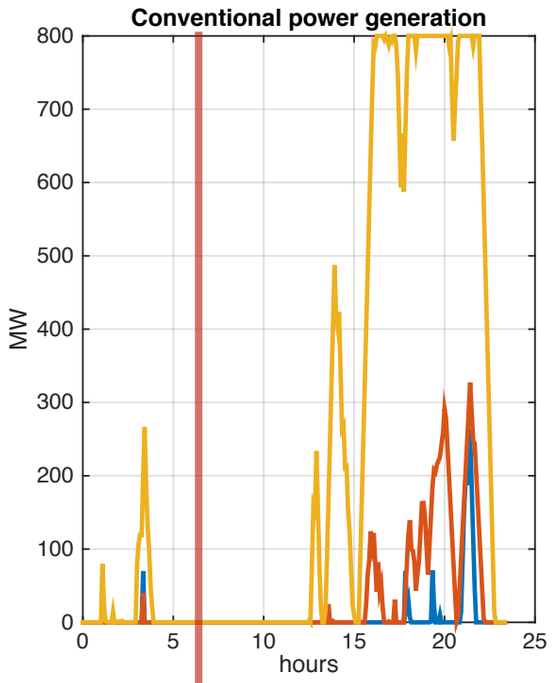
- **Stochastic MPC**, with branching factor M (ex: M=[4 3 2 2 1])
- **Average MPC**, that is deterministic MPC based on the **expected** (price, load-renewable) realization
- **Prescient MPC**, that is deterministic MPC based on the **exact** future (price, load-renewable) realization

SMPC for market-based optimal power dispatch

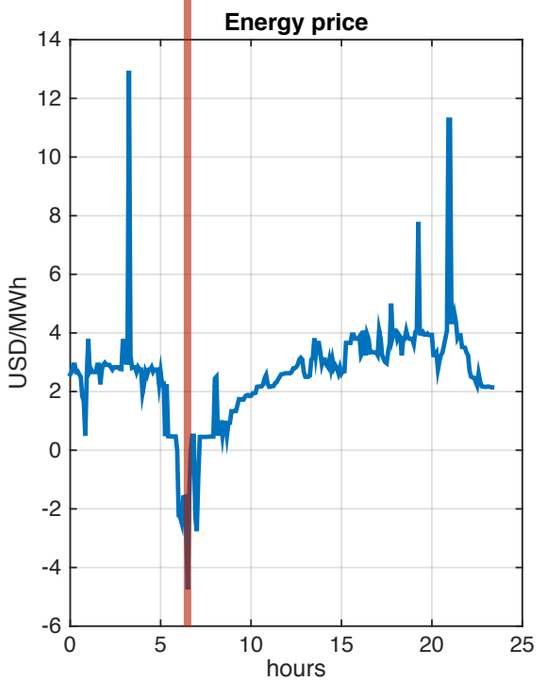
- Simulation results using SMPC, M=[2,2,2,1,1] (1 day, May 26, 2014)

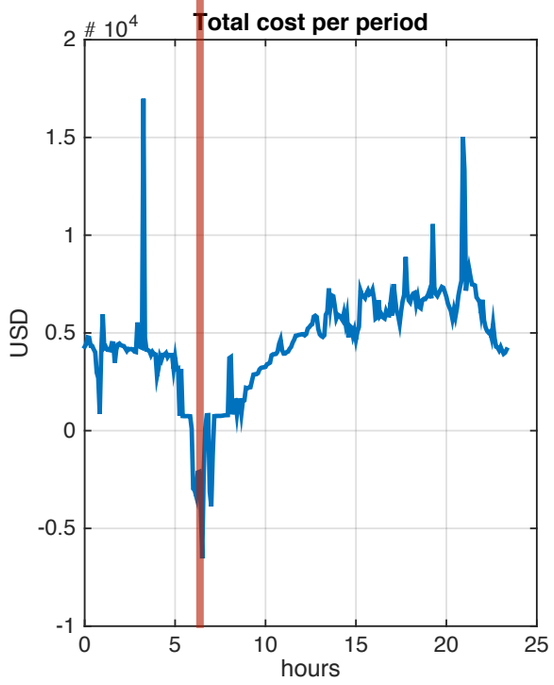


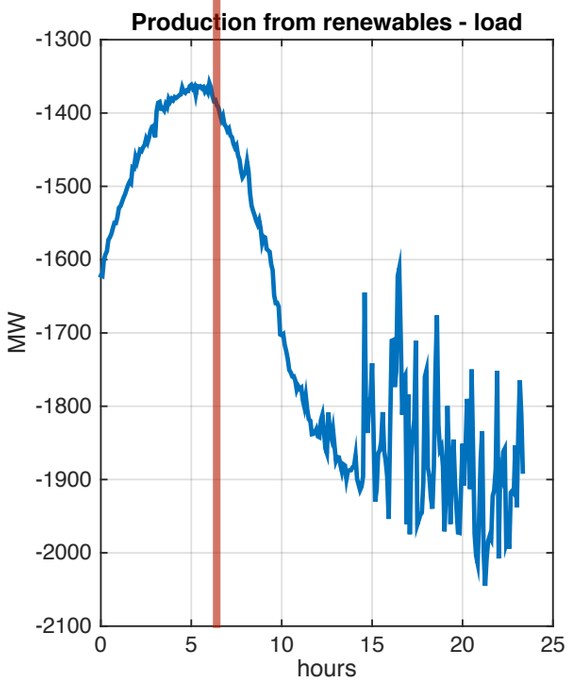




total cost = 1,266,099 USD







SMPC for market-based optimal power dispatch

- Compare simulation results wrt different tree complexity, prescient, and deterministic (1 day, May 26, 2014)

exact knowledge of future uncertainty

stochastic formulation

```
Prescient:   Total cost=  1,247,909 [USD],                    nvar=  30, CPUTIME =  14 [ms]
Stochastic:  Total cost=  1,266,099 [USD], M=[2,2,2,1,1], nvar= 105, CPUTIME =  43 [ms]
Stochastic:  Total cost=  1,266,123 [USD], M=[3,3,1,1,1], nvar= 140, CPUTIME =  50 [ms]
Stochastic:  Total cost=  1,266,214 [USD], M=[2,2,1,1,1], nvar=  95, CPUTIME =  30 [ms]
Stochastic:  Total cost=  1,266,701 [USD], M=[3,1,1,1,1], nvar=  80, CPUTIME =  27 [ms]
Stochastic:  Total cost=  1,267,069 [USD], M=[2,1,1,1,1], nvar=  55, CPUTIME =  22 [ms]
Average:     Total cost=  1,267,113 [USD], M=[1,1,1,1,1]  nvar=  30, CPUTIME =  14 [ms]
Frozen-time: Total cost=  1,267,401 [USD],                    nvar=  30, CPUTIME =  14 [ms]
```

deterministic: assume future disturbance = average of historical data

deterministic: assume future disturbance = current disturbance

nvar = number of variables in QP problem = 5*(# nodes), **CPUTIME** = time to build tree,build QP matrices, and solve

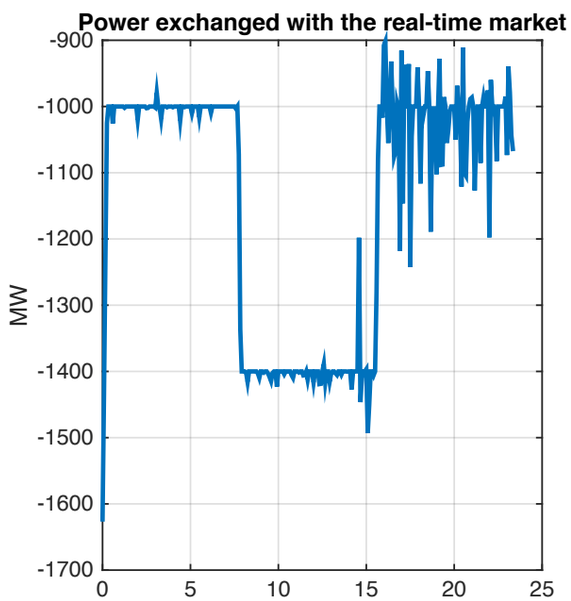
SMPC for market-based optimal power dispatch

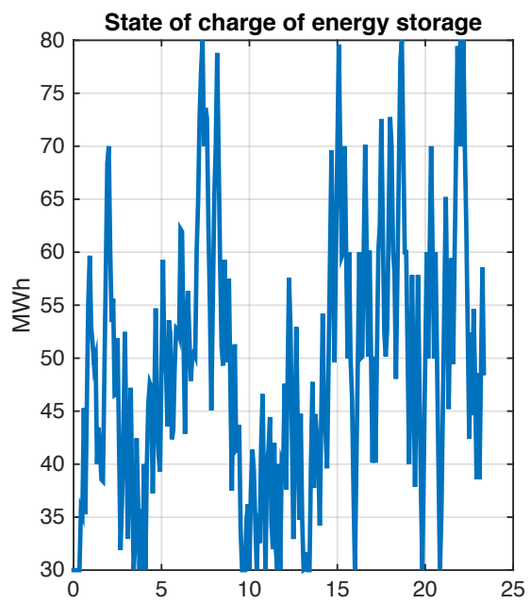
- Tracking an E-Program

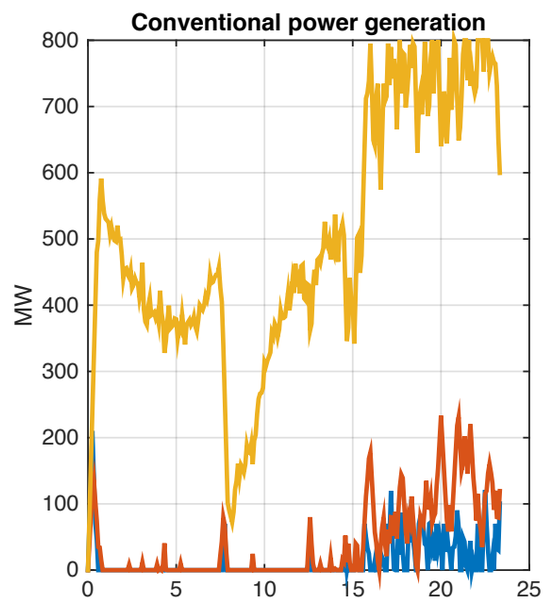
$$\min \sum_{k=0}^{N-1} \gamma(P_{\text{ex},k} - E_k)^2 + (a_i P_{i,k}^2 + b_i P_{i,k} + c_i) - p_k[P_{\text{ex},k} - E_k]$$

$\gamma = 10^3$

SMPC with branching factor M=[2 2 2 1 1]

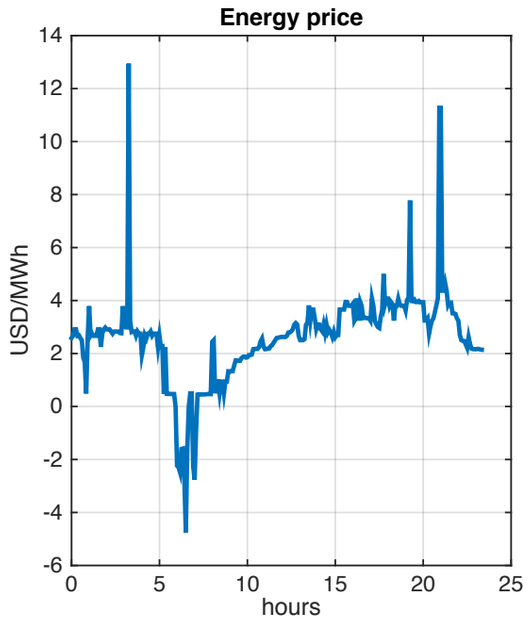


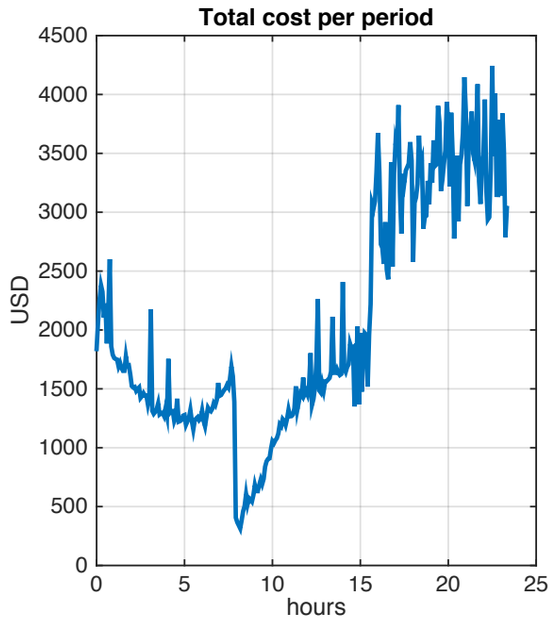


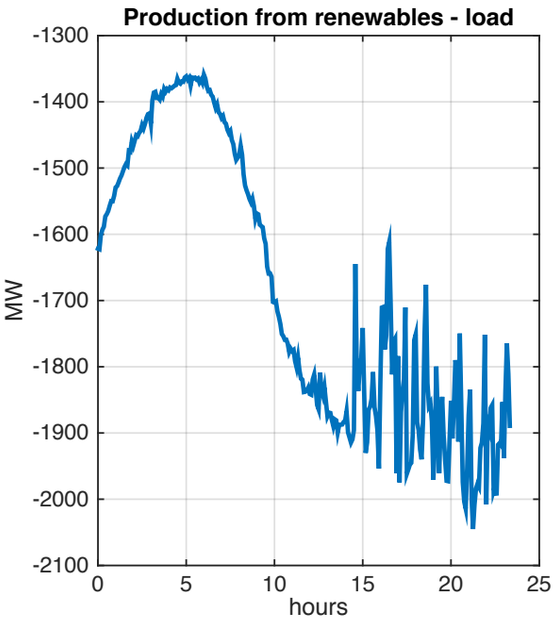


total cost = 574,388 USD

Revenues from day-ahead market are not counted



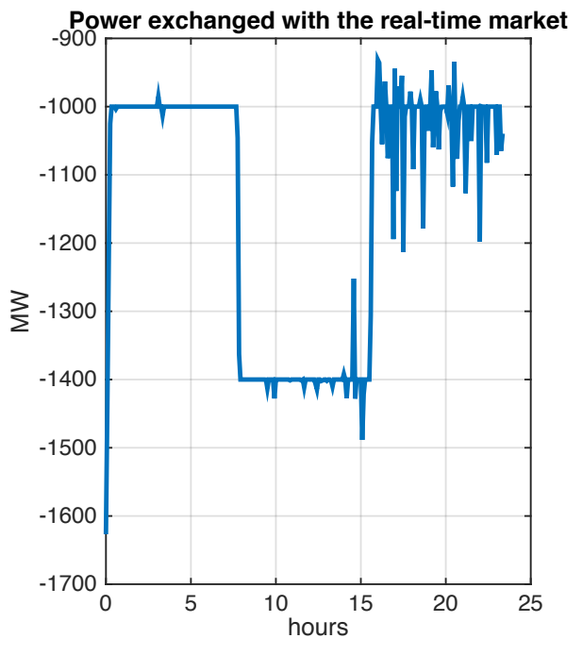


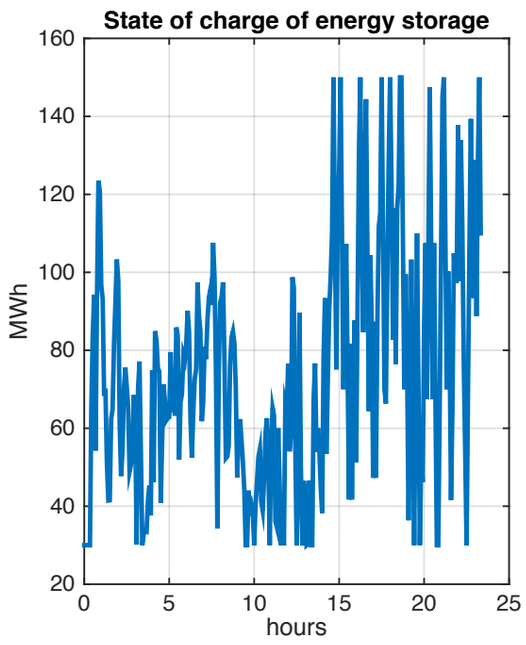


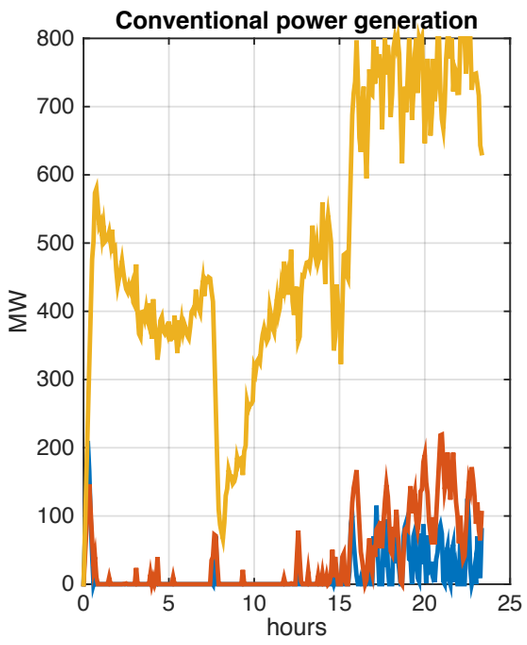
- Change storage type:

$S_{\min} \leq S_k \leq S_{\max}$ $\quad \Delta_{S,\min} \leq S_{k+1} - S_k \leq \Delta_{S,\max}$

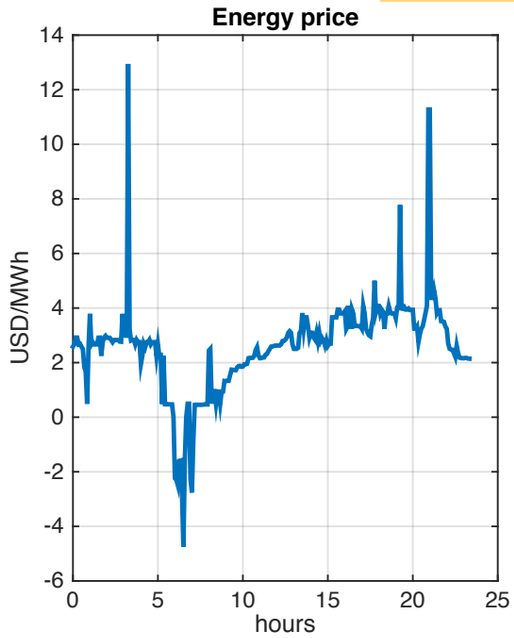
$S_{\min} = 30$
$S_{\max} = \cancel{80}$ 150 MWh
$\Delta_{S,\min} = \cancel{-10}$
$\Delta_{S,\max} = \cancel{10}$ ±40 MWh/PTU

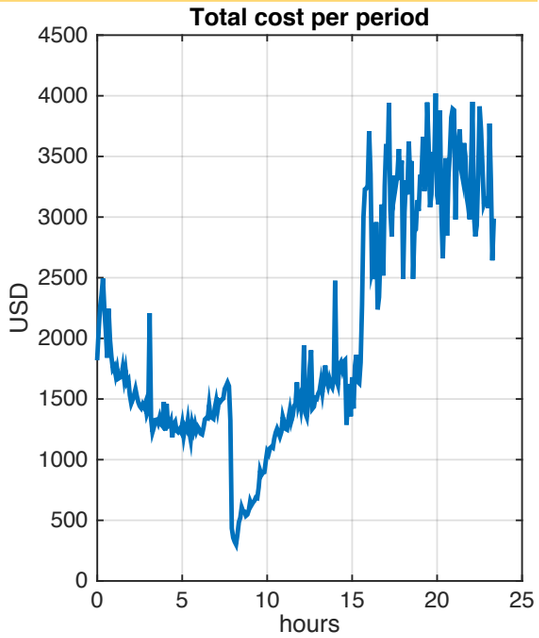


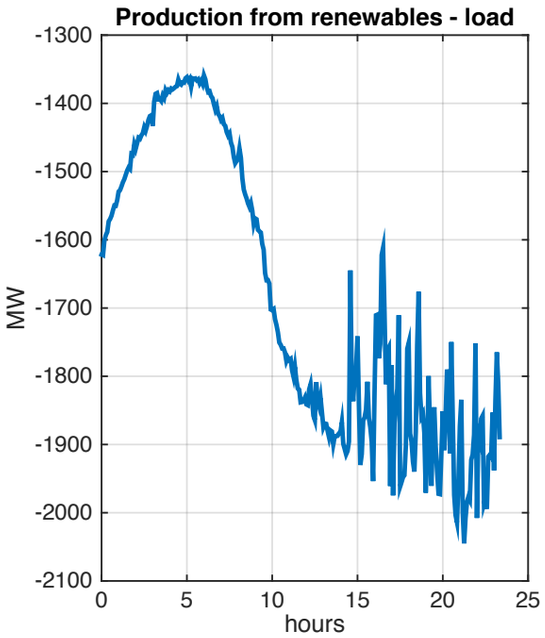




total cost = 563,283 USD



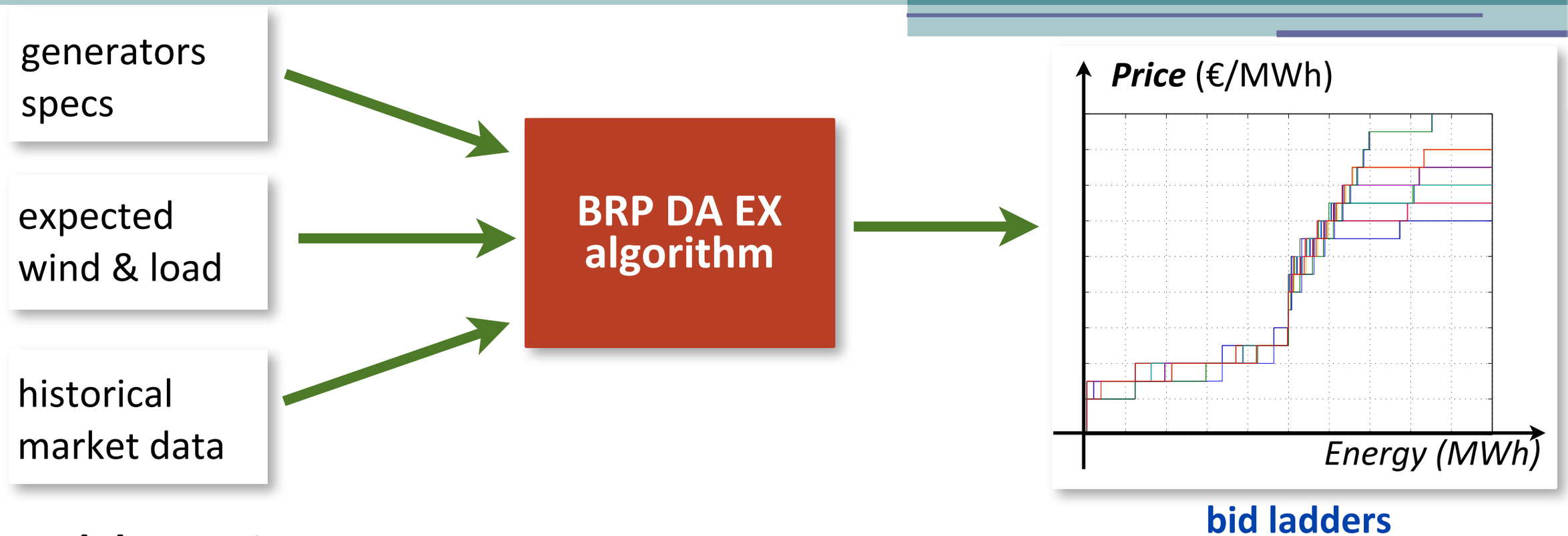




Revenues from day-ahead market are not counted

SMPC for optimal bidding on energy markets

Bidding on the Day-Ahead (EXchange) market



bid ladders

For each hour, given:

1. generator data (current power profile, min and max power, efficiency, etc.)
2. expected load and wind profiles for each hour of the following day
3. scenarios for AS prices, which are not disclosed yet (**stochastic disturbance**)

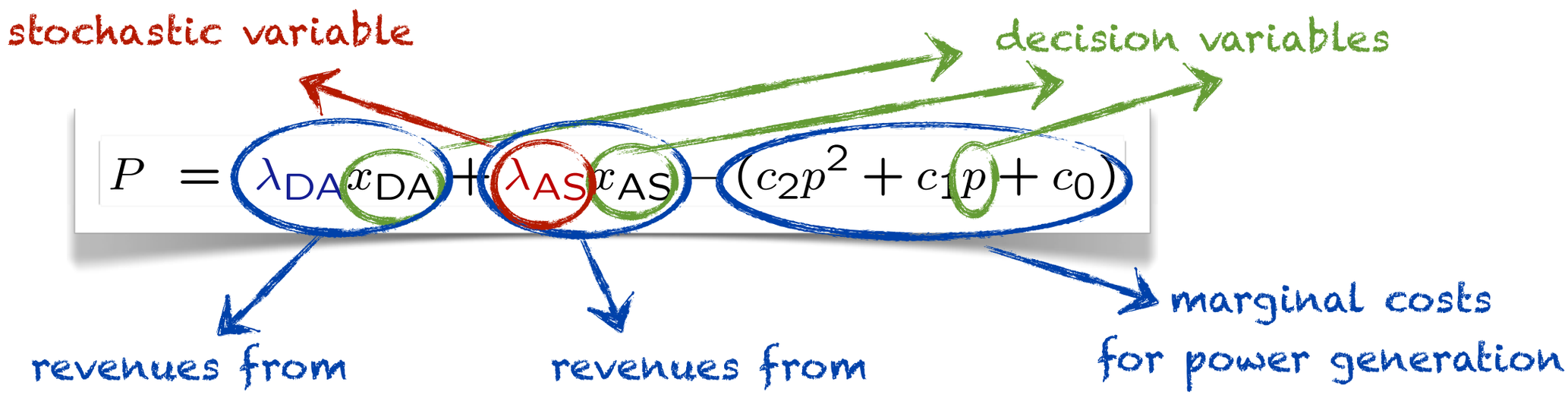
compute optimal allocation of energy on the EX market

that **maximizes profits** and **minimize imbalance risks**

Output: 24 bid curves (one for each hour of the following day)

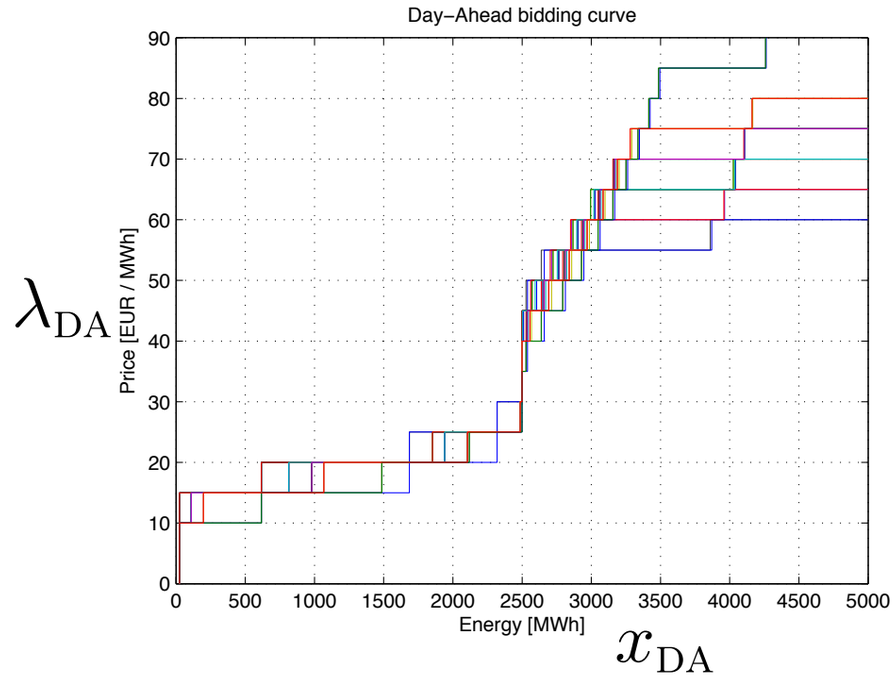
Bidding on the day-ahead (DA) market

- **Price-taking point of view**: the offer has no influence on the cleared price
- **Profit**: for each hour, **for each fixed price** λ_{DA}, the profit is



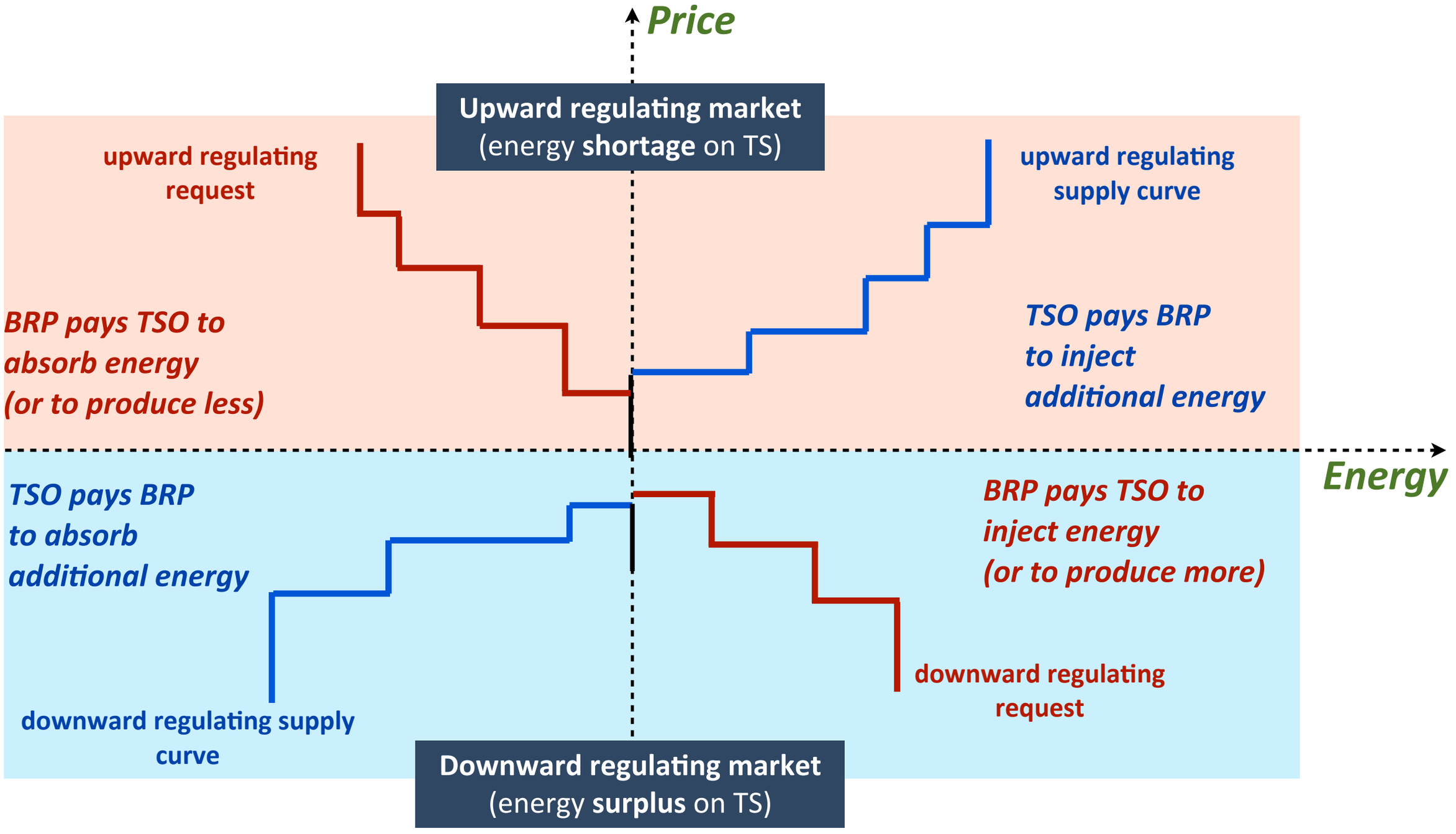
- **Objective**: minimize CVaR or maximize profit

- **Constraints:**
  - production limits, internal balancing



- **Uncertainty:** only AS price, average renewables/load considered (it's 1 day ahead!)

Bidding on AS market (double-sided market)



- **AS bidding algorithm:** for each (supply, request) price pair, **compute the best energy allocation** (considering stochastic scenarios of available wind power minus load)

Simulation results

- Economic performance metrics calculated to test the effectiveness of the E-Price market structure and algorithms
- **Dutch TN**: 7 BRPs consisting of gas/coal plants and wind farms
- Wind power forecast in E-program = actual wind power:

	Production Costs (€)	Imbalance Costs (€)	AS Profit (€)
Single-sided market	1,821,846	36,602	285,294
Double-sided market	1,824,743	16,899	366,895

- Simulation = 1 day
- Sum of all BRPs

- Imprecise wind forecast (actual wind power > expected during DA bidding):

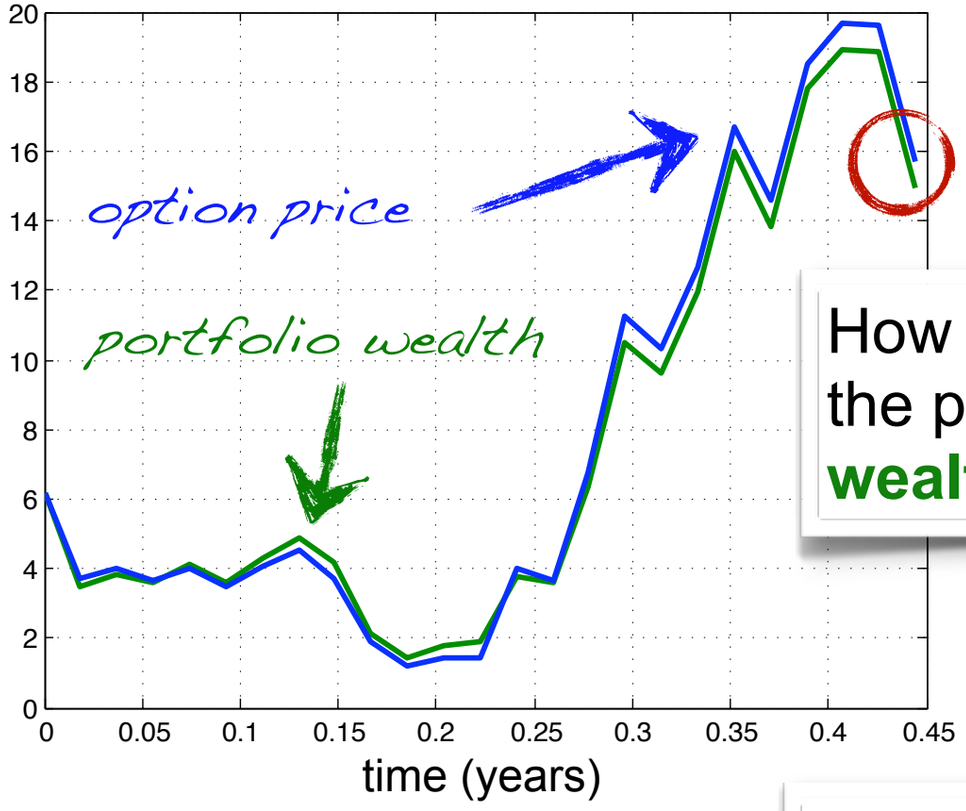
	Production Costs (€)	Imbalance Costs (€)	AS Profit (€)
Single-sided market	1,749,858	105,577	347,151
Double-sided market	1,750,263	98,024	452,491

- As creating imbalance is more expensive in the double-sided market, BRPs have higher incentives to efficiently allocate resources

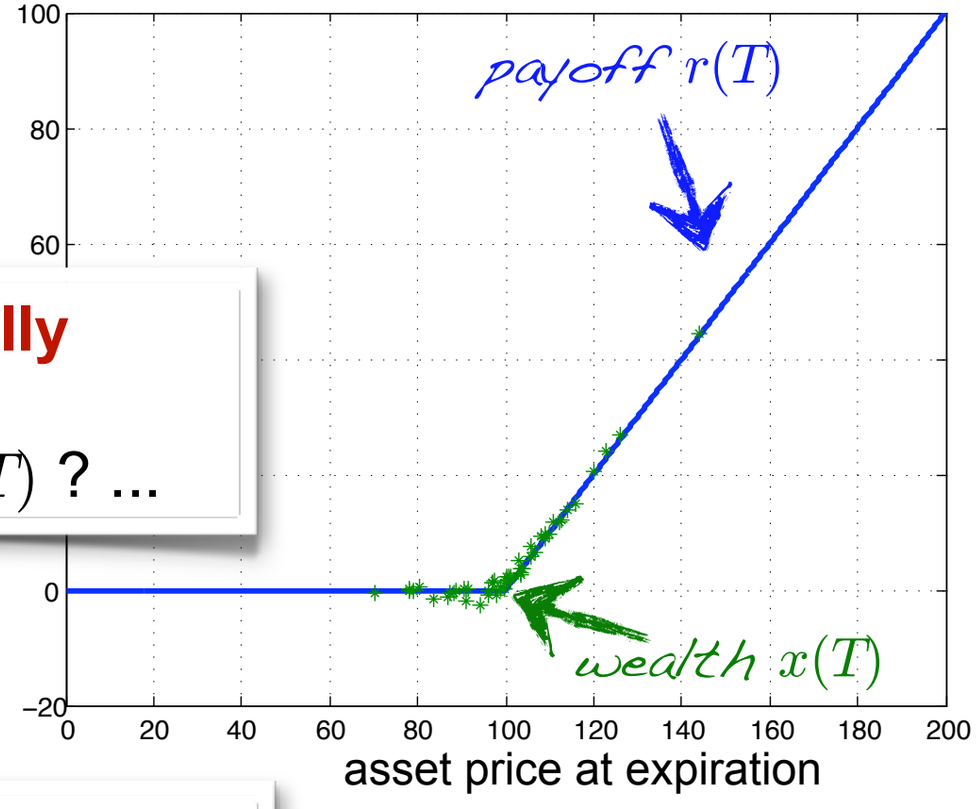
SMPC for dynamic hedging of financial options

Dynamic hedging problem for financial options

- The financial institution sells a synthetic option to a customer and gets $x(0)$ (€)
- Such money $x(0)$ is used to create a portfolio $x(t)$ of n underlying assets (e.g., stocks) whose prices at time t are $w_1(t)$, $w_2(t)$, ..., $w_n(t)$
- At the expiration date T, the option is worth the payoff $r(T)$ = wealth (€) to be returned to the customer



How to **adjust dynamically** the portfolio so that **wealth** $x(T) =$ **payoff** $r(T)$? ...



.. for any price realization $w_i(t)$?

Portfolio dynamics

- Portfolio wealth at time t:

$$x(t) = u_0(t) + \sum_{i=1}^{n} w_i(t) u_i(t)$$

money in bank account (risk-free asset)

price of asset #i (stochastic process)

number of assets #i

Example: $w_i(t) =$ **log-normal** model (used in Black-Scholes' theory)

$$dw_i = (\mu dt + \sigma dz_i) w_i$$

geometric Brownian motion

- Assets traded at **discrete-time** intervals under the *self-balancing constraint*:

$$x(t+1) = (1+r)x(t) + \sum_{i=0}^{n} b_i(t) u_i(t)$$

r = interest rate

$$b_i(t) \triangleq w_i(t+1) - (1+r) w_i(t)$$

Payoff function

- Let $w(t) = \begin{bmatrix} w_1(t) \\ \vdots \\ w_n(t) \end{bmatrix}$ = vector of assets, and $y(t) = \begin{bmatrix} y_1(t) \\ \vdots \\ y_n(t) \end{bmatrix}$

- Option price $p(t)$: $p(t) = f(w(t), y(t))$

$$p(t) = f(w(0), w(1), \ldots, w(t), w(t))$$ **path-dependent options**

- Payoff $p(T)$: $p(T) = f(w(0), w(1), \ldots, w(T))$

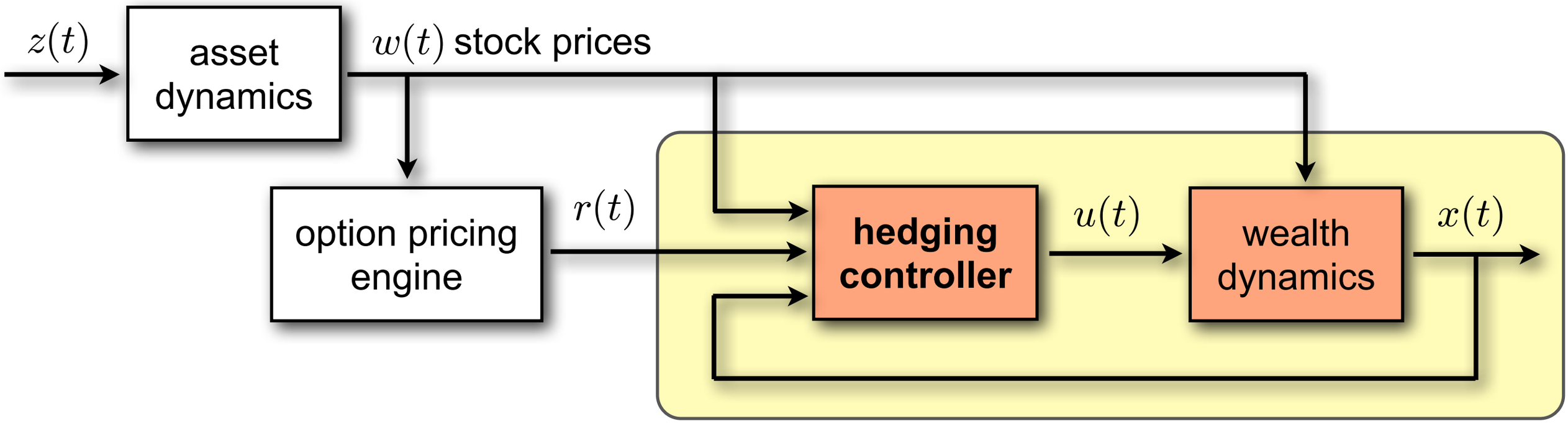
- Examples (n=1)

$$p(T) = \max\{w(T) - K, 0\}$$ **European call**

$$p(T) = \max\left\{0, C + \min_{i \in \{1,\ldots,N_{\text{fix}}\}} \frac{w(t_i) - w(t_{i-1})}{w(t_{i-1})}\right\}$$ **Napoleon cliquet** (t_i= fixing dates)

Option hedging = linear stochastic control

- Block diagram of dynamic option hedging problem:



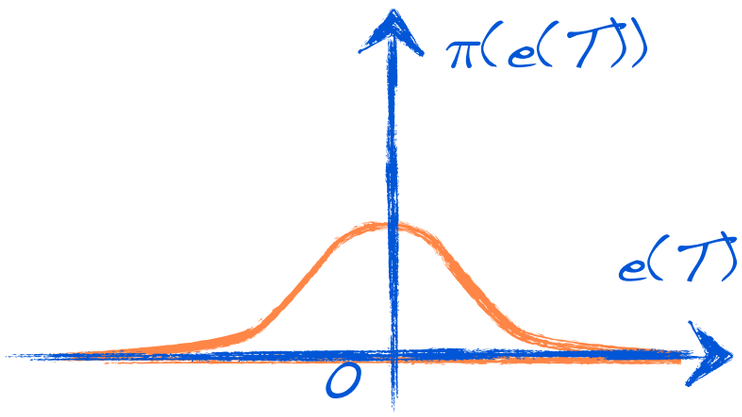
- Reference signal $r(t)$ = price $p(t)$ of hedged option

- **Control objective**: $x(T)$ should be as close as possible to $r(T)$, for any possible realization of the asset prices $w(t)$ *("tracking w/ disturbance rejection")*

Control objective

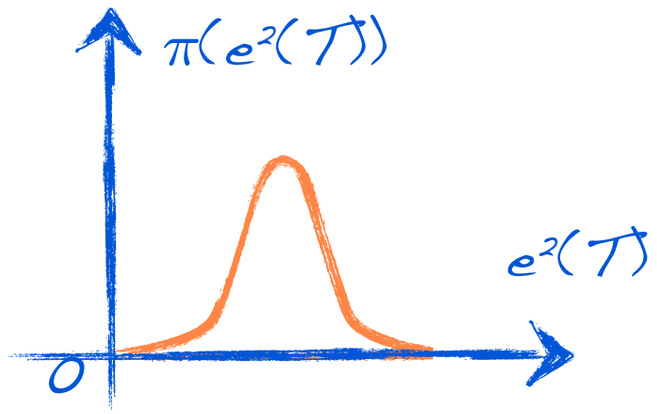
- Define **hedging error** $e(t) \triangleq w(t) - p(t)$

we want e(T) small !



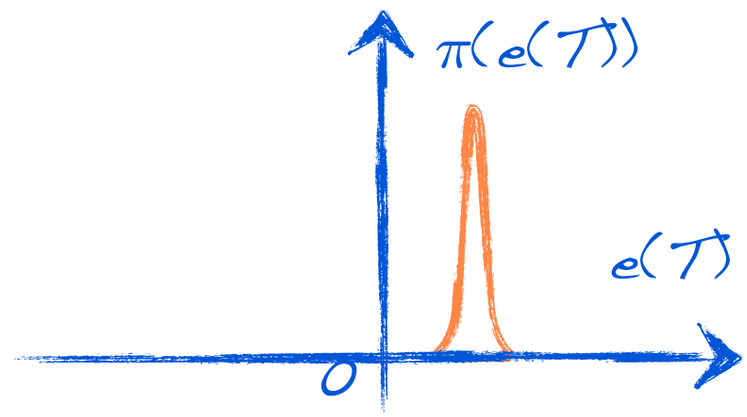
$$\min E\left[e(T)\right]^2$$

Very risky, variance of e(T) may be large !



$$\min E\left[e(T)^2\right]$$

If minimum is 0 then both mean and variance of hedging error are 0



$$\min E\left[(e(T) - E[e(T)])^2\right]$$

How about E[e(T)] ?

In fact: $E\left[e(T)^2\right] = E\left[(e(T) - E[e(T)])^2\right] + \alpha E\left[e(T)\right]^2$ for $\alpha = 1$

Var[e(T)]

Minimum variance control

- Let us minimize the **variance** of hedging error: $\min E\left[(e(T) - E[e(T)])^2\right]$

Theorem *Let* $w(0) = p(0)$. *Under* **non-arbitrage conditions**, *if a strategy exists such that* $Var[w(T) - p(T)] = 0$ *(=perfect hedging) then*

$$w(t) - p(t) = 0, \ \forall t = 0, 1, \ldots, T$$

and in particular $E[w(T) - p(T)] = 0$, *that is the final hedging error is deterministically 0.*

(cf. Black-Scholes theory)

Proof: By induction.

Note: $\mathrm{Var}[e(T)]=0$ means $e(T)=w(T)$-$p(T)$ is deterministic

$e(T)>0$ would imply $w(T)>p(T)$ ⇒ always **gain** wealth, which is impossible if no arbitrage exists, as

$$w(0) = (1 + r)^{-T} E[p(T)]$$

Existing approaches to dynamic option hedging

- **Analytical approaches**: choose $u(t)$ to "reject" exactly the effect of stochastic noise $z(t)$
  PROS: lots of insight !
  CONS: limited to simple stock models & payoff functions

(Black and Scholes, 1973)
(Merton, 1973)

- **Multi-stage stochastic programming**: choose $u(t)$ by solving a (large-scale) optimization problem.
  PROS: pricing & hedging in one shot, check arbitrage conditions
  CONS: rough discretization of probability space & trading dates

(Edirisinghe *et al.*, 1993)
(Gondzio *et al.*, 2003)
(Klaassen, 1998)
(Zhao and Ziemba, 1998)

- **Stochastic dynamic programming**:
  PROS: rather versatile
  CONS: rough discretization of probability space & trading dates

(Bertsimas *et al.*, 2001)

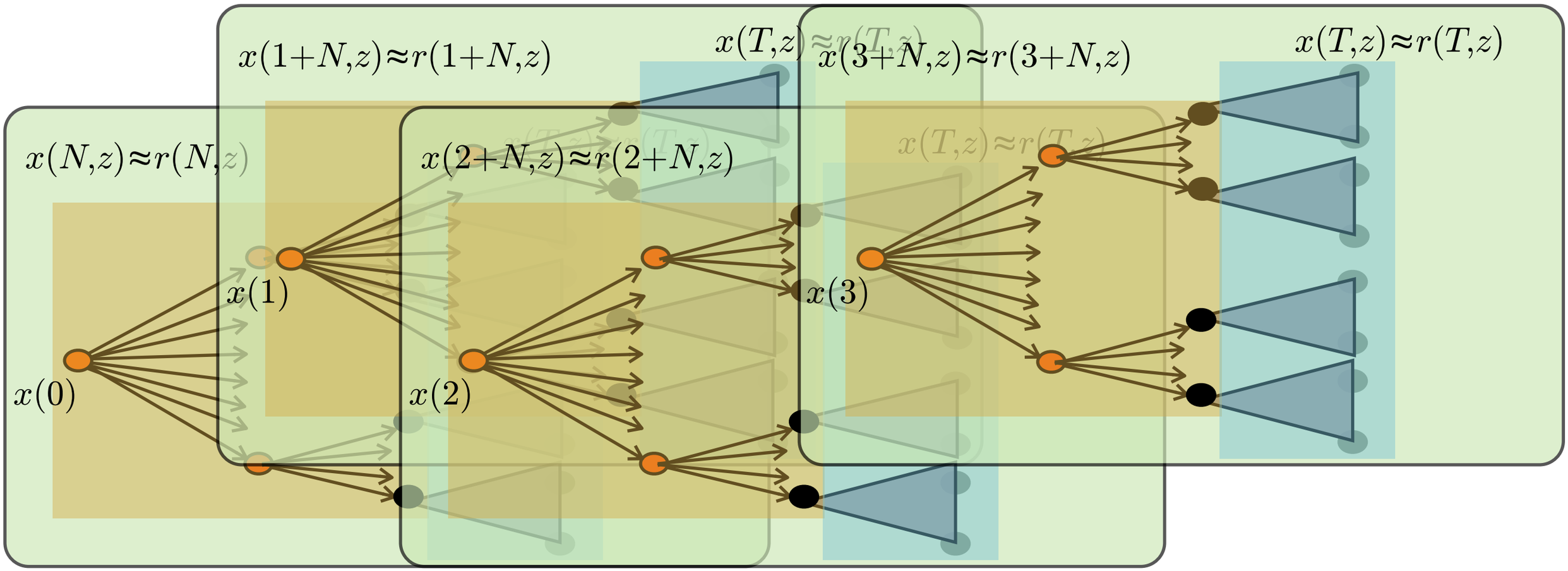
- **Stochastic model predictive control**: solve recursively and on-line an optimization problem over a short prediction horizon
  PROS: very versatile
  CONS: requires on-line optimization

(Primbs, 2009)
(Bemporad, Bellucci, Gabbriellini, 2012)

SMPC for dynamic option hedging

- Stochastic finite-horizon optimal control problem:

$$\begin{aligned} \min_{\{u(k,z)\}} \quad & \mathrm{Var}_z\left[x(t+N,z) - r(t+N,z)\right] \\ \text{s.t.} \quad & x(k+1,z) = (1+r)x(k,z) + \sum_{i=0}^{n} b_i(k,z)u_i(k,z), \;\; k = t,\ldots,t+N \\ & x(t,z) = x(t) \end{aligned}$$



SMPC for dynamic option hedging

- Drawback: the longer the horizon N, the largest the number of scenarios !
- Special case: use $N{=}1$!

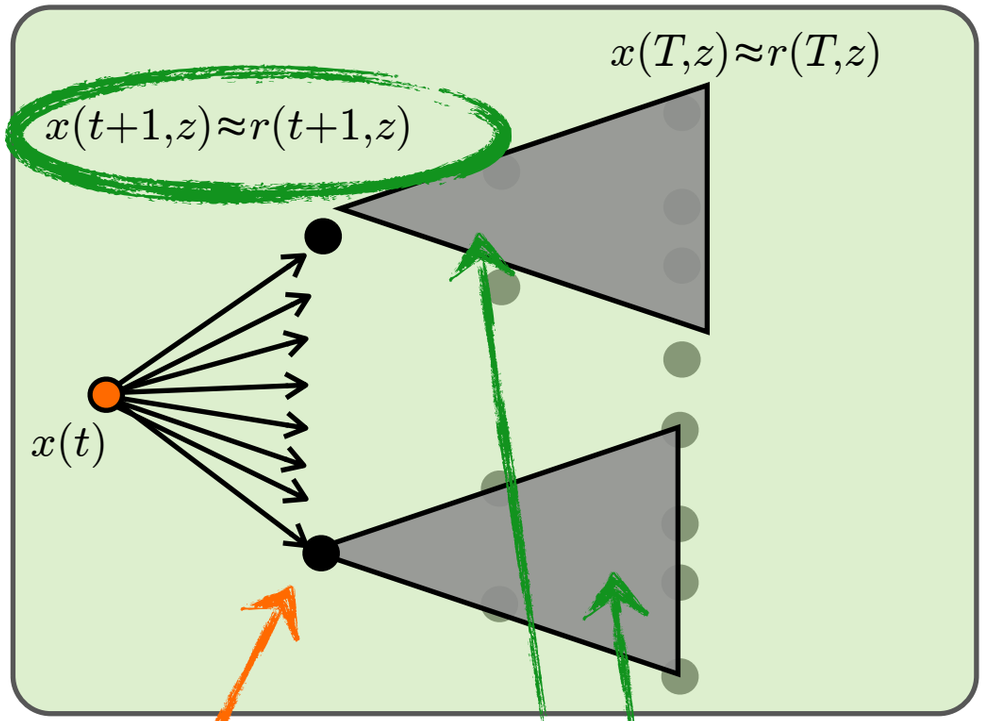
minimum variance control !

$$\begin{aligned} \min_{u(t)} \quad & \mathrm{Var}_z\left[x(t+1,z) - r(t+1,z)\right] \\ \text{s.t.} \quad & x(t+1,z) = (1+r)x(t) + \sum_{i=0}^{n} b_i(t,z)u_i(t) \end{aligned}$$

✓ **Only one** vector $u(t)$ to optimize

✓ No further branching, so we can generate **a lot** of scenarios for z! (example: 1000)

◉ Need to compute target wealth $r(t{+}1,z)$ for all z



On-line optimization: very simple **least squares** problem with n variables !

(n = number of traded assets)

SMPC hedging algorithm

- Let t=current hedging date, $w(t)$=wealth of portfolio, $x(t) \in \mathbb{R}^n$= asset prices
- Use **Monte Carlo simulation** to generate M scenarios of future asset prices

$$x^1(t+1),\ x^2(t+1),\ \ldots,\ x^M(t+1)$$
$$y^1(t+1),\ y^2(t+1),\ \ldots,\ y^M(t+1)$$

- Use a **pricing engine** to generate the corresponding future option prices

$$p^1(t+1),\ p^2(t+1),\ \ldots,\ p^M(t+1)$$

- **Optimize** *sample variance,* get new asset quantities $u(t) \in \mathbb{R}^n$, rebalance portfolio:

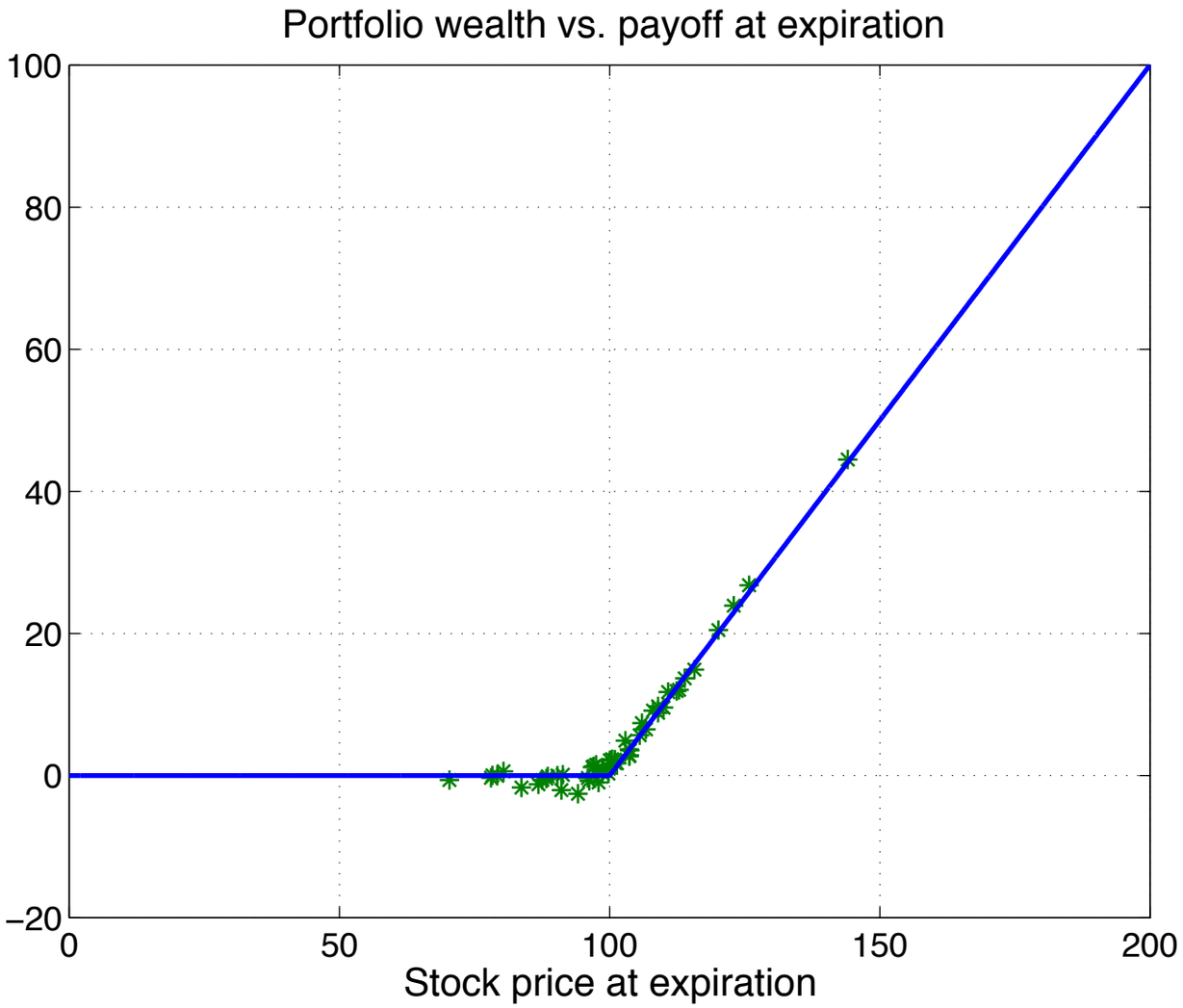
$$\min_{u(t)} \sum_{j=1}^{M} \left(w^j(t+1) - p^j(t+1) - \left(\frac{1}{M} \sum_{i=1}^{M} w^j(t+1) - p^j(t+1) \right) \right)^2$$

Least squares problem (n = number of traded assets)

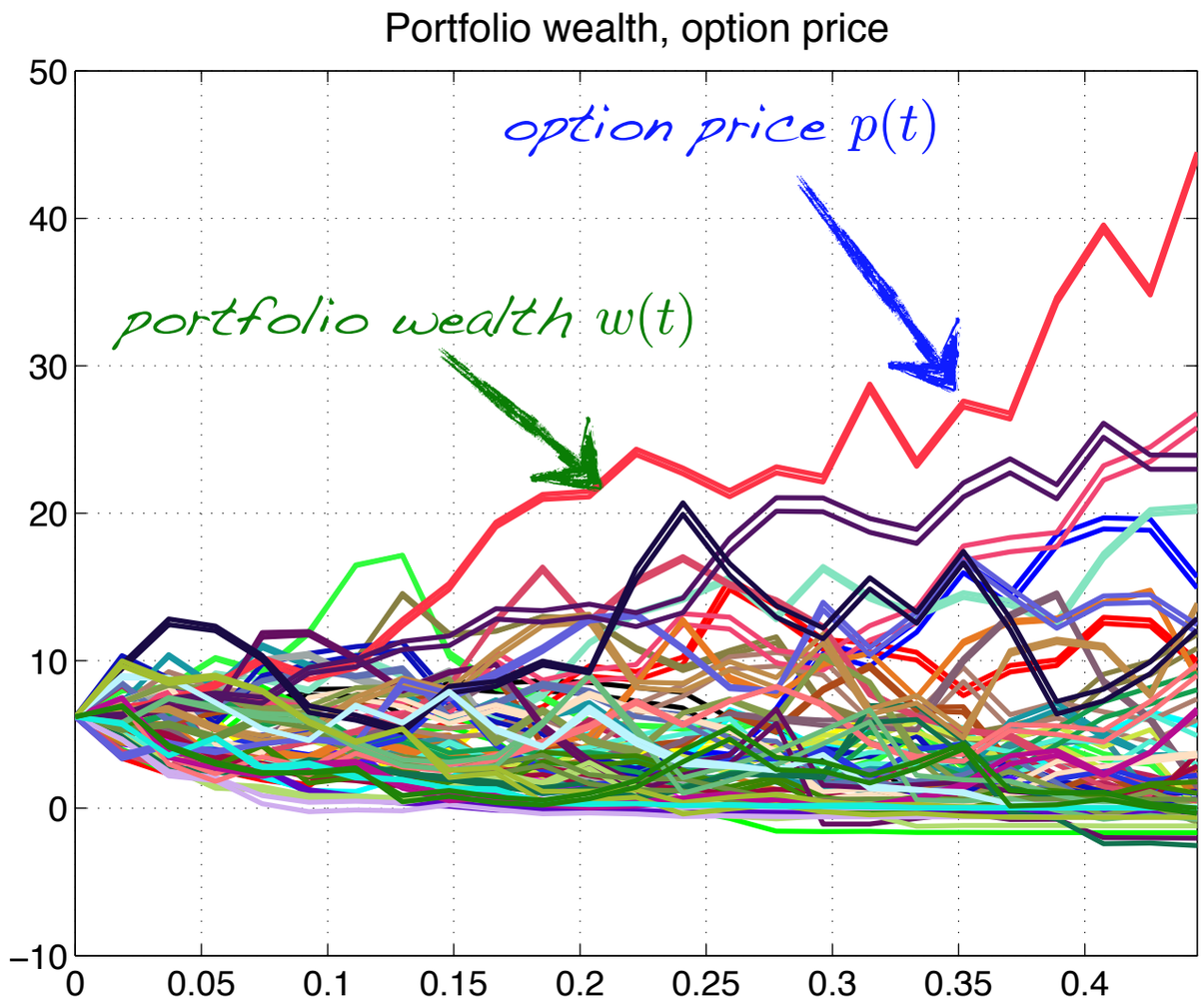
With transaction costs, the problem can be cast to a *quadratic program*

(Bemporad, Puglia, Gabbriellini, 2011)

Example: BS model, European call

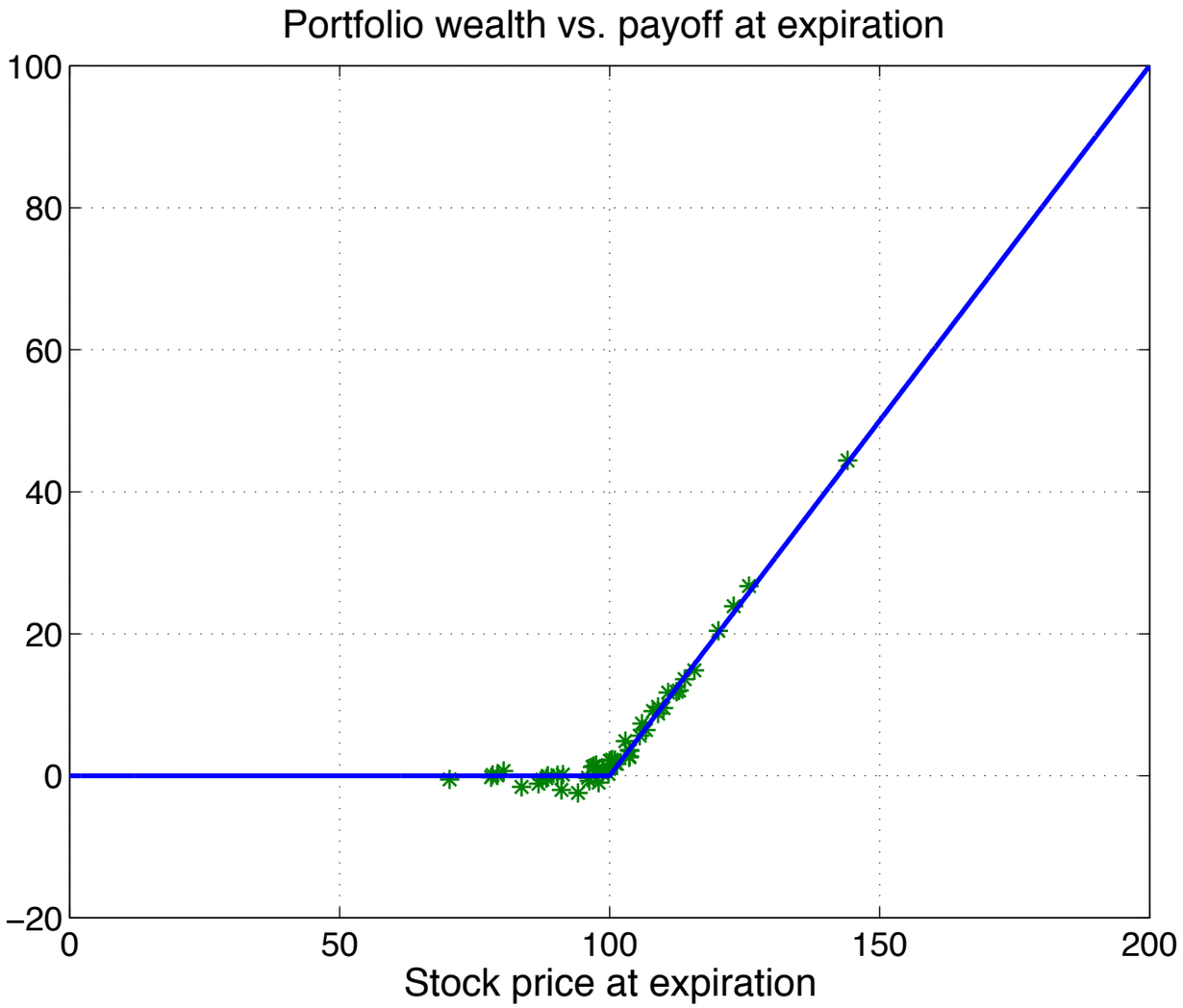


- Black-Scholes model (=log-normal)
- volatility$=0.2$, risk-free$=0.04$
- $T=24$ weeks ($\Delta t=1$ week)
- 50 simulations
- $M=100$ scenarios
- Pricing method: Monte Carlo sim.
- **SMPC**

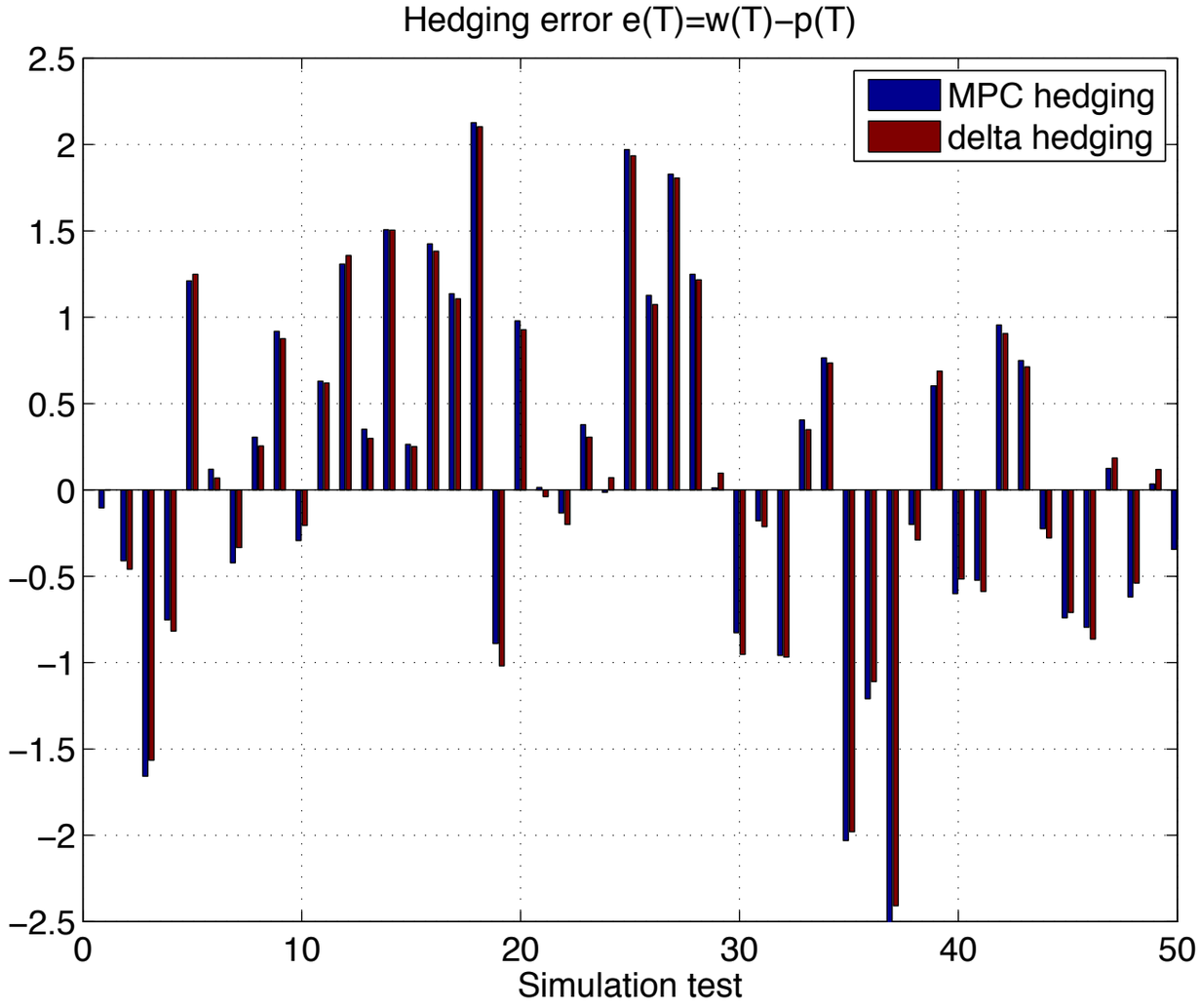


- CPU time $= 7.52$ ms per SMPC step
  (Matlab R2009 on 1.86GHz Intel Core 2 Duo)

Example: BS model, European call



- Black-Scholes model (=log-normal)
- volatility$=0.2$, risk-free$=0.04$
- $T=24$ weeks (hedging every week)
- 50 simulations
- $M=100$ scenarios
- **Delta-hedging**



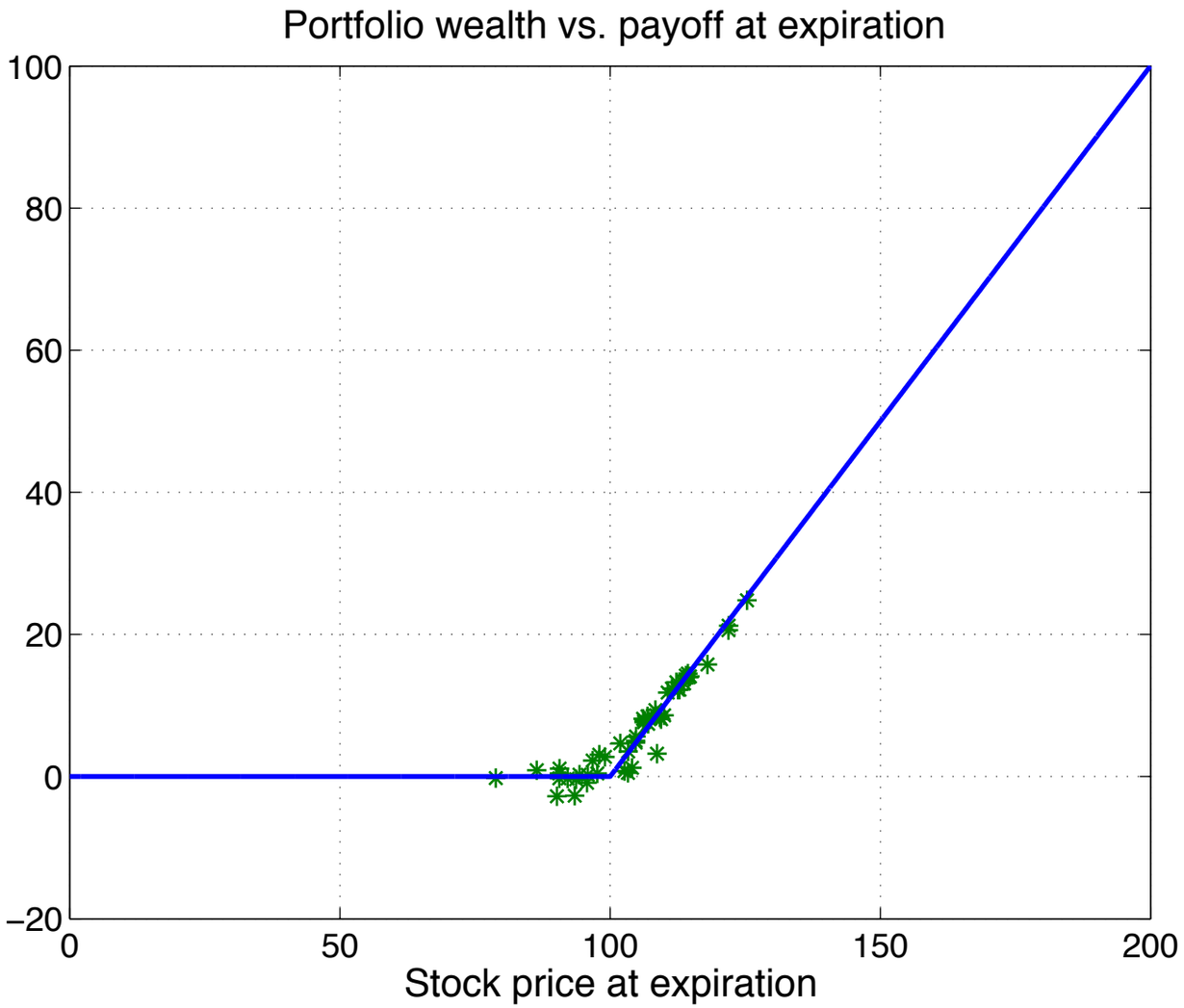
- CPU time $=0.2$ ms per SMPC step
  (Matlab R2009 on 1.86GHz Intel Core 2 Duo)

SMPC and delta-hedging are almost indistinguishable

Example: BS model, European call

TIME	x(t)	w(t)	p(t)	u0(t)	x(t)*u1(t)	x(t)*dp/dx(t)
t=0.0000:	S=100.000,	P= 6.196,	O= 6.196,	P(B)=-52.152,	P(S)= 58.348	(BS delta= 57.926)
t=0.0185:	S=101.367,	P= 6.955,	O= 6.865,	P(B)=-56.091,	P(S)= 63.046	(BS delta= 62.628)
t=0.0370:	S= 96.897,	P= 4.134,	O= 4.261,	P(B)=-42.629,	P(S)= 46.762	(BS delta= 46.307)
t=0.0556:	S= 94.582,	P= 2.985,	O= 3.108,	P(B)=-35.080,	P(S)= 38.065	(BS delta= 37.607)
t=0.0741:	S= 93.057,	P= 2.345,	O= 2.415,	P(B)=-29.877,	P(S)= 32.222	(BS delta= 31.771)
t=0.0926:	S= 93.371,	P= 2.431,	O= 2.395,	P(B)=-30.200,	P(S)= 32.632	(BS delta= 32.165)
t=0.1111:	S= 94.295,	P= 2.732,	O= 2.591,	P(B)=-32.518,	P(S)= 35.250	(BS delta= 34.760)
t=0.1296:	S= 88.192,	P= 0.426,	O= 0.859,	P(B)=-14.985,	P(S)= 15.411	(BS delta= 15.053)
t=0.1481:	S= 90.411,	P= 0.803,	O= 1.199,	P(B)=-19.776,	P(S)= 20.579	(BS delta= 20.147)
t=0.1667:	S= 88.586,	P= 0.373,	O= 0.754,	P(B)=-14.236,	P(S)= 14.609	(BS delta= 14.234)
t=0.1852:	S= 87.683,	P= 0.214,	O= 0.544,	P(B)=-11.312,	P(S)= 11.526	(BS delta= 11.186)
t=0.2037:	S= 90.998,	P= 0.641,	O= 1.000,	P(B)=-18.744,	P(S)= 19.385	(BS delta= 18.910)
t=0.2222:	S= 94.742,	P= 1.425,	O= 1.867,	P(B)=-30.734,	P(S)= 32.158	(BS delta= 31.555)
t=0.2407:	S= 99.890,	P= 3.149,	O= 3.945,	P(B)=-52.320,	P(S)= 55.469	(BS delta= 54.841)
t=0.2593:	S=102.720,	P= 4.682,	O= 5.466,	P(B)=-64.736,	P(S)= 69.418	(BS delta= 68.857)
t=0.2778:	S= 99.723,	P= 2.609,	O= 3.439,	P(B)=-51.468,	P(S)= 54.077	(BS delta= 53.379)
t=0.2963:	S= 99.591,	P= 2.499,	O= 3.147,	P(B)=-50.513,	P(S)= 53.012	(BS delta= 52.268)
t=0.3148:	S= 98.178,	P= 1.709,	O= 2.233,	P(B)=-42.460,	P(S)= 44.169	(BS delta= 43.336)
t=0.3333:	S=100.471,	P= 2.709,	O= 3.142,	P(B)=-55.135,	P(S)= 57.845	(BS delta= 57.034)
t=0.3519:	S=102.804,	P= 4.012,	O= 4.363,	P(B)=-69.359,	P(S)= 73.371	(BS delta= 72.719)
t=0.3704:	S= 97.457,	P= 0.144,	O= 1.202,	P(B)=-34.892,	P(S)= 35.037	(BS delta= 33.884)
t=0.3889:	S= 97.789,	P= 0.238,	O= 1.030,	P(B)=-34.692,	P(S)= 34.930	(BS delta= 33.564)
t=0.4074:	S= 98.881,	P= 0.602,	O= 1.089,	P(B)=-41.289,	P(S)= 41.891	(BS delta= 40.275)
t=0.4259:	S= 97.699,	P= 0.071,	O= 0.308,	P(B)=-22.850,	P(S)= 22.921	(BS delta= 20.300)
t=0.4444:	S= 96.002,	P= -0.344,	O= 0.000,	P(B)= -0.344,	P(S)= 0.000	(BS delta= 0.000)

Example: Heston model, European call



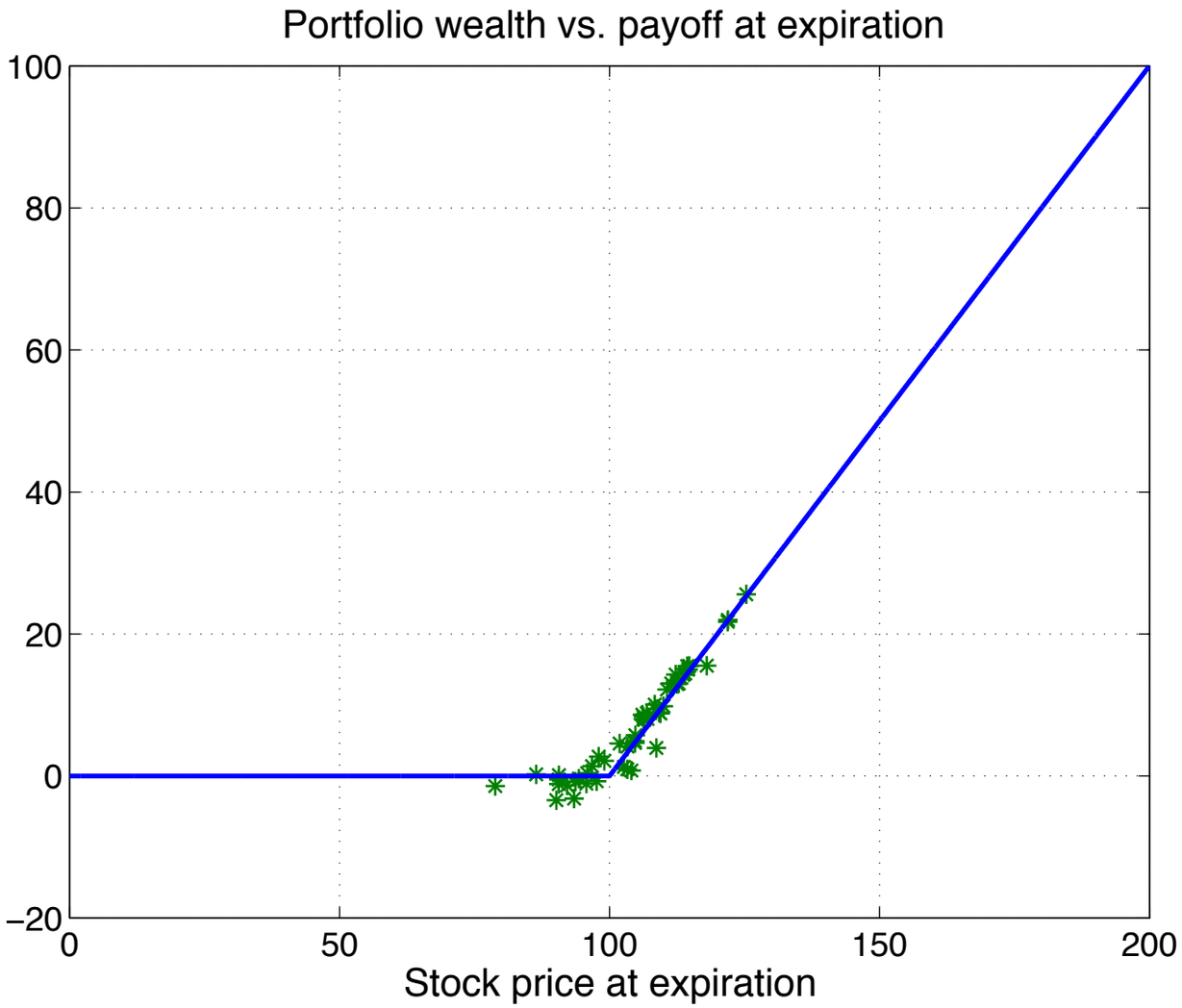
- **Heston's model**
- $T = 24$ weeks (hedging every week)
- 50 simulations
- $M = 100$ scenarios
- risk-free$=0.04$
- Pricing method: Monte Carlo sim.
- **SMPC**

- CPU time $= 85.5$ ms per SMPC step
  (Matlab R2009 on 1.86GHz Intel Core 2 Duo)

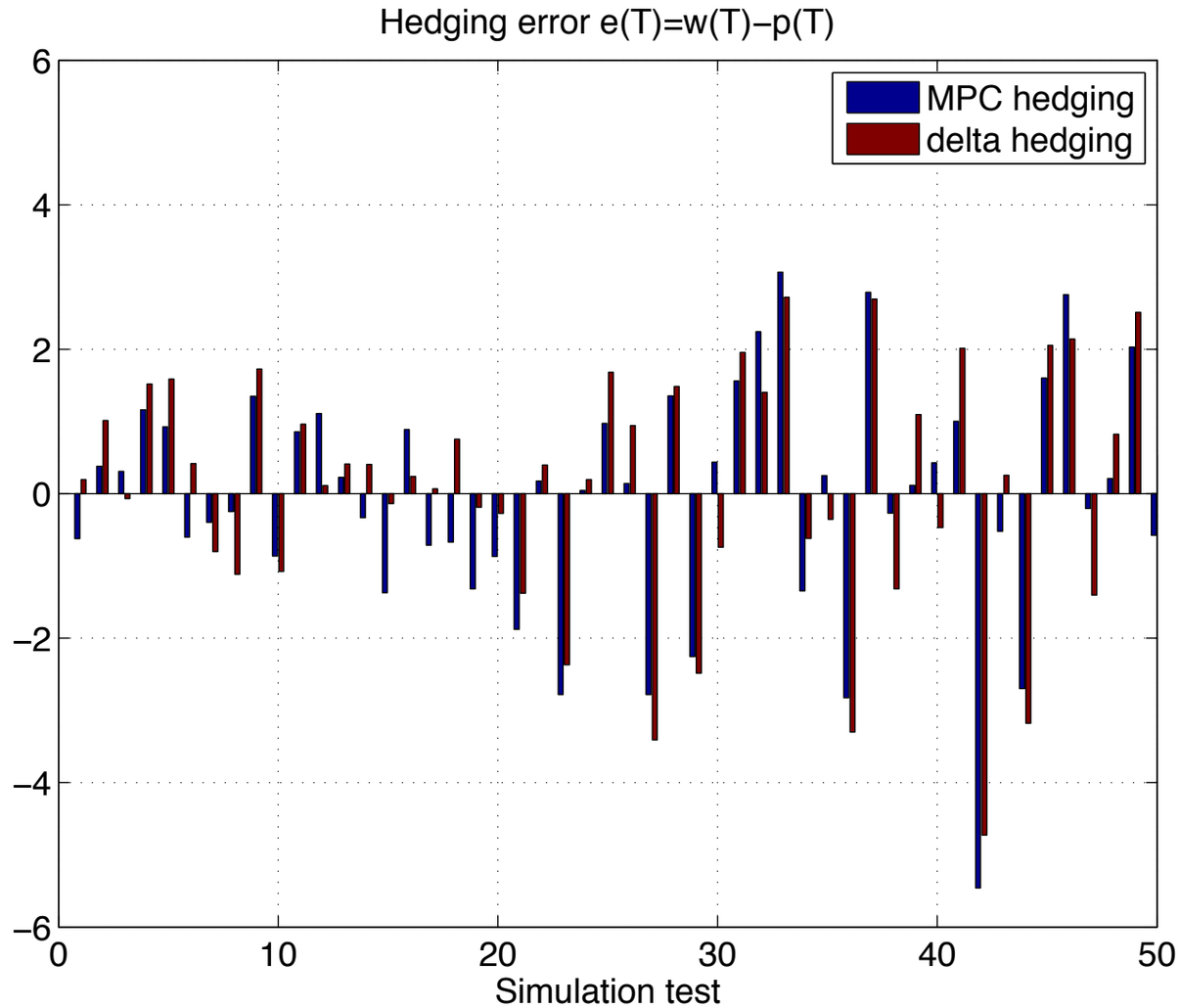
Heston's model

$$
\begin{aligned}
dx_i(\tau) &= (\mu_i^x d\tau + \sqrt{y_i(\tau)} dz_i^x) x_i(\tau) \\
dy_i(\tau) &= \theta_i (k_i - y_i(\tau)) d\tau + \omega_i \sqrt{y_i(\tau)} dz_i^y
\end{aligned}
$$

Example: Heston model, European call

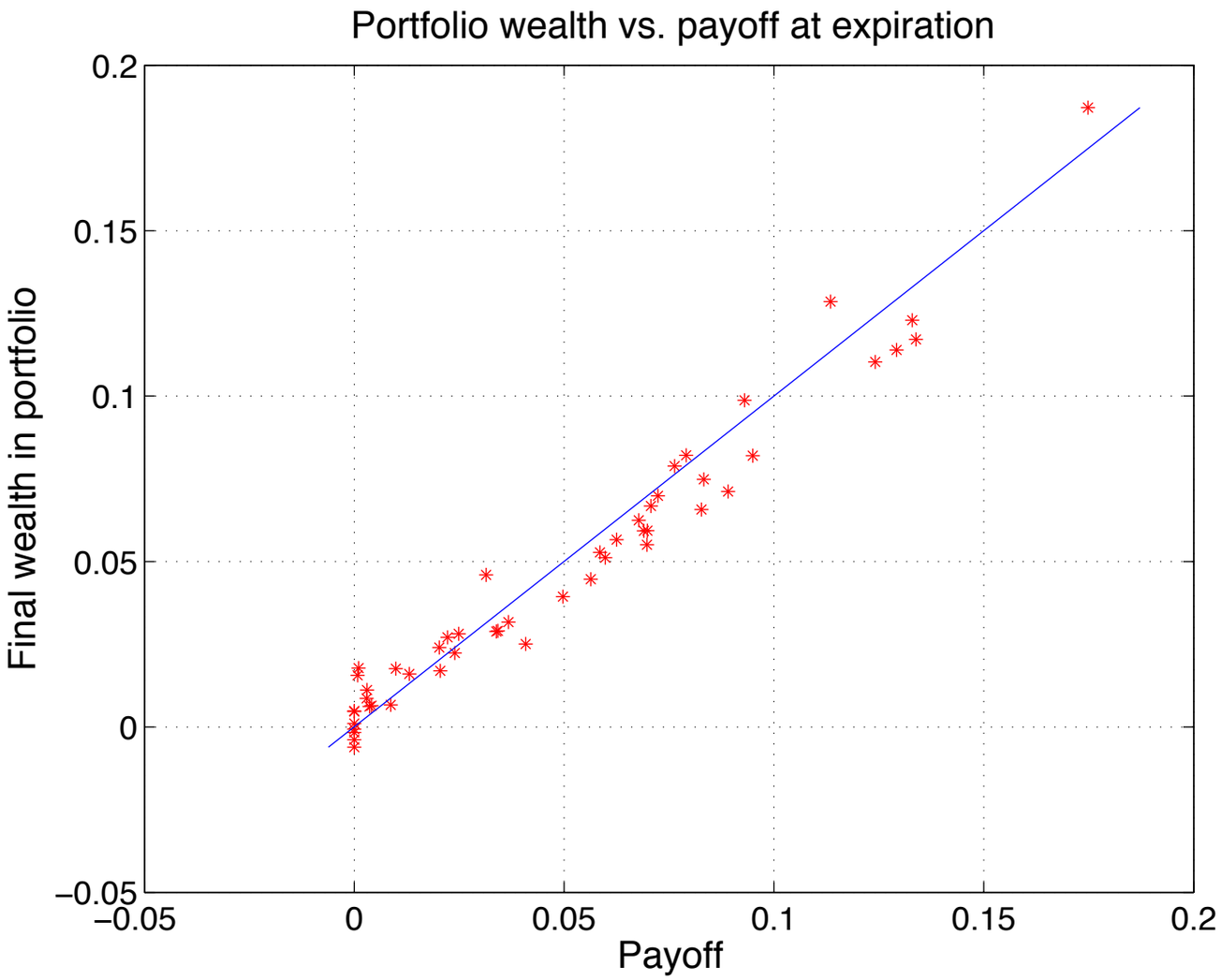


- Heston's model
- $T = 24$ weeks (hedging every week)
- 50 simulations
- $M = 100$ scenarios
- risk-free$= 0.04$
- **Delta hedging**



- CPU time $= 1.85$ ms per SMPC step
  (Matlab R2009 on 1.86GHz Intel Core 2 Duo)

Example: BS model, Napoleon cliquet



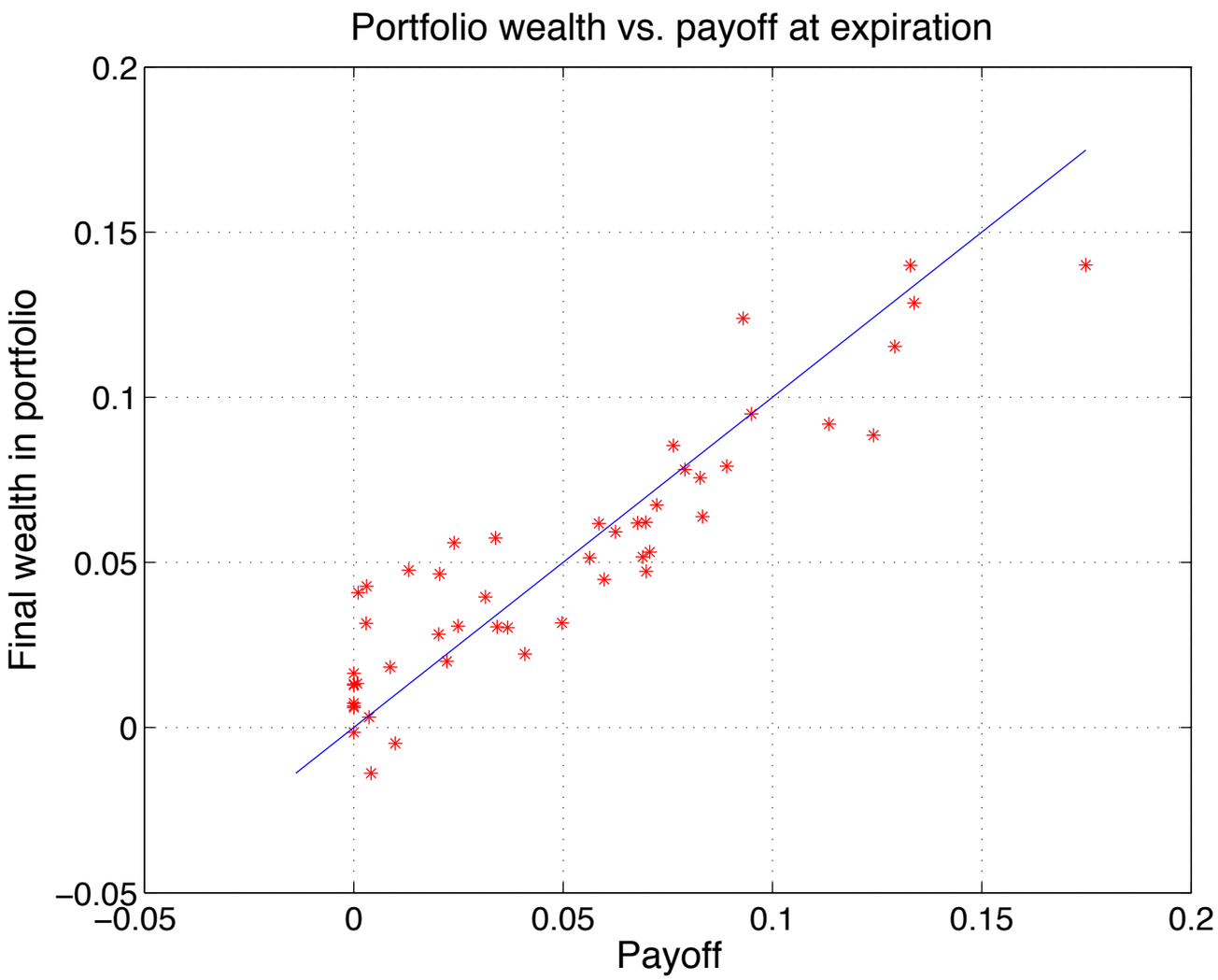
- Black-Scholes model (=log-normal)
- volatility$=0.2$
- $T=24$ weeks (hedging every week)
- 50 simulations
- $M=100$ scenarios
- risk-free$=0.04$
- Pricing method: Monte Carlo sim.
- **SMPC: only trade underlying stock**

$$p(T) = \max\left\{0, C + \min_{i\in\{1,\ldots,N_{\text{fix}}\}} \frac{x(t_i) - x(t_{i-1})}{x(t_{i-1})}\right\}$$

t_i=0,8,16,24 weeks

- CPU time $=1400$ ms per SMPC step
  (Matlab R2009 on 1.86GHz Intel Core 2 Duo)

Example: BS model, Napoleon cliquet



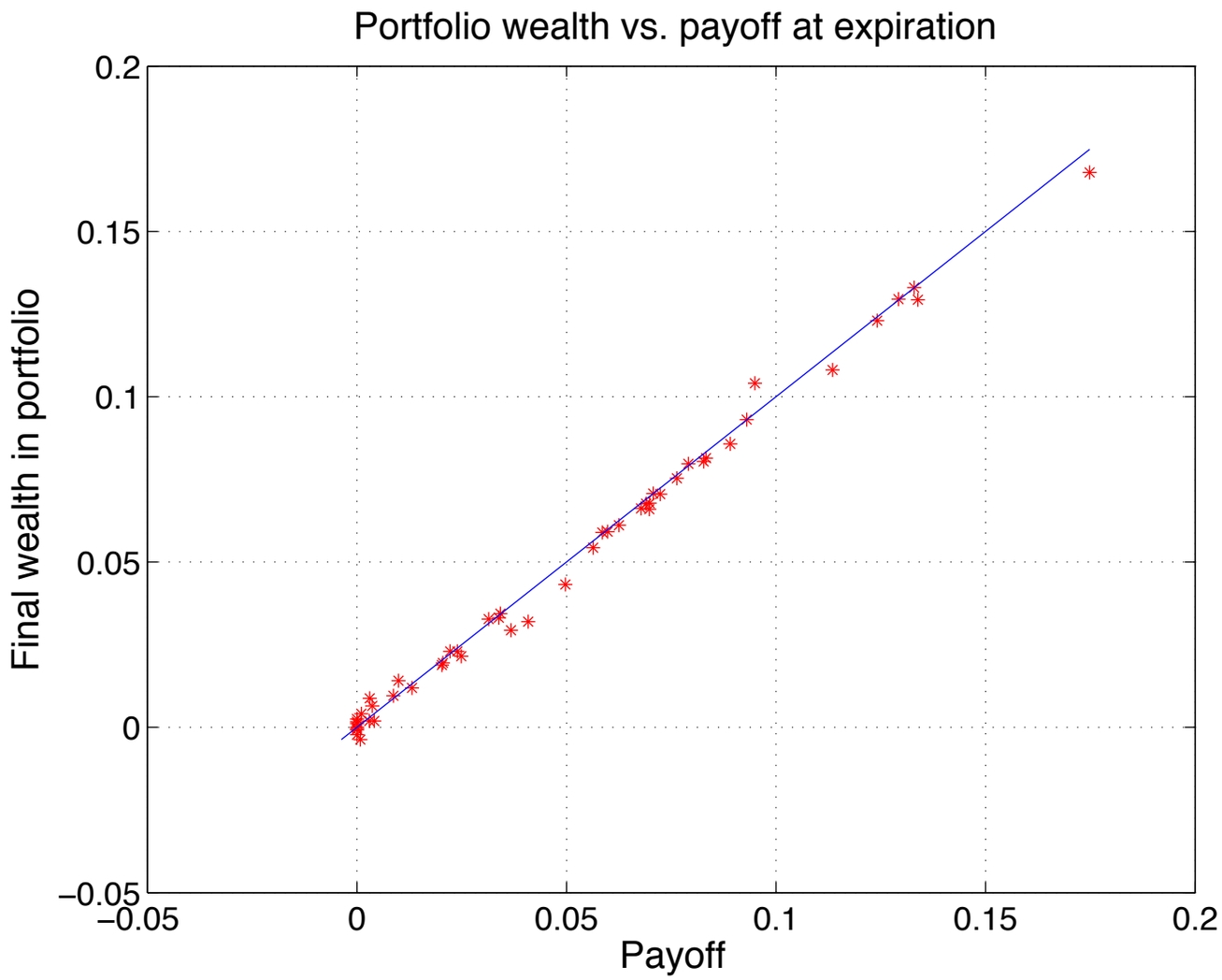
- Black-Scholes model (=log-normal)
- volatility$=0.2$
- $T=24$ weeks (hedging every week)
- 50 simulations
- $M=100$ scenarios
- risk-free$=0.04$
- **Delta hedging, only trade underlying stock**

$$p(T) = \max\left\{0, C + \min_{i\in\{1,\ldots,N_{\text{fix}}\}} \frac{x(t_i) - x(t_{i-1})}{x(t_{i-1})}\right\}$$

t_i=0,8,16,24 weeks

- CPU time $= 2.41$ ms per SMPC step
  (Matlab R2009 on 1.86GHz Intel Core 2 Duo)

Example: BS model, Napoleon cliquet



- Black-Scholes model (=log-normal)
- volatility$=0.2$
- $T=24$ weeks (hedging every week)
- 50 simulations
- $M=100$ scenarios
- risk-free$=0.04$
- Pricing method: Monte Carlo sim.
- **SMPC: Trade underlying stock & European call with maturity** $t+T$

$$p(T) = \max\left\{0, C + \min_{i\in\{1,\ldots,N_{\text{fix}}\}} \frac{x(t_i) - x(t_{i-1})}{x(t_{i-1})}\right\}$$

t_i=0,8,16,24 weeks

- CPU time $= 1625$ ms per SMPC step
  (Matlab R2009 on 1.86GHz Intel Core 2 Duo)

Approximate option pricing

- **Bottleneck** of the approach for exotic options: price M future option values

$$p^1(t+1),\ p^2(t+1),\ \ldots,\ p^M(t+1)$$

- **Monte Carlo pricing** can be time consuming: say L scenarios to evaluate a single option value ⇒ need to simulate MLpaths to build optimization problem

  (e.g.: M=100, L=10000, ML=10^6)

- Use off-line **function approximation** techniques to estimate $p(t)$ as a function of current asset parameters and other option-related parameters

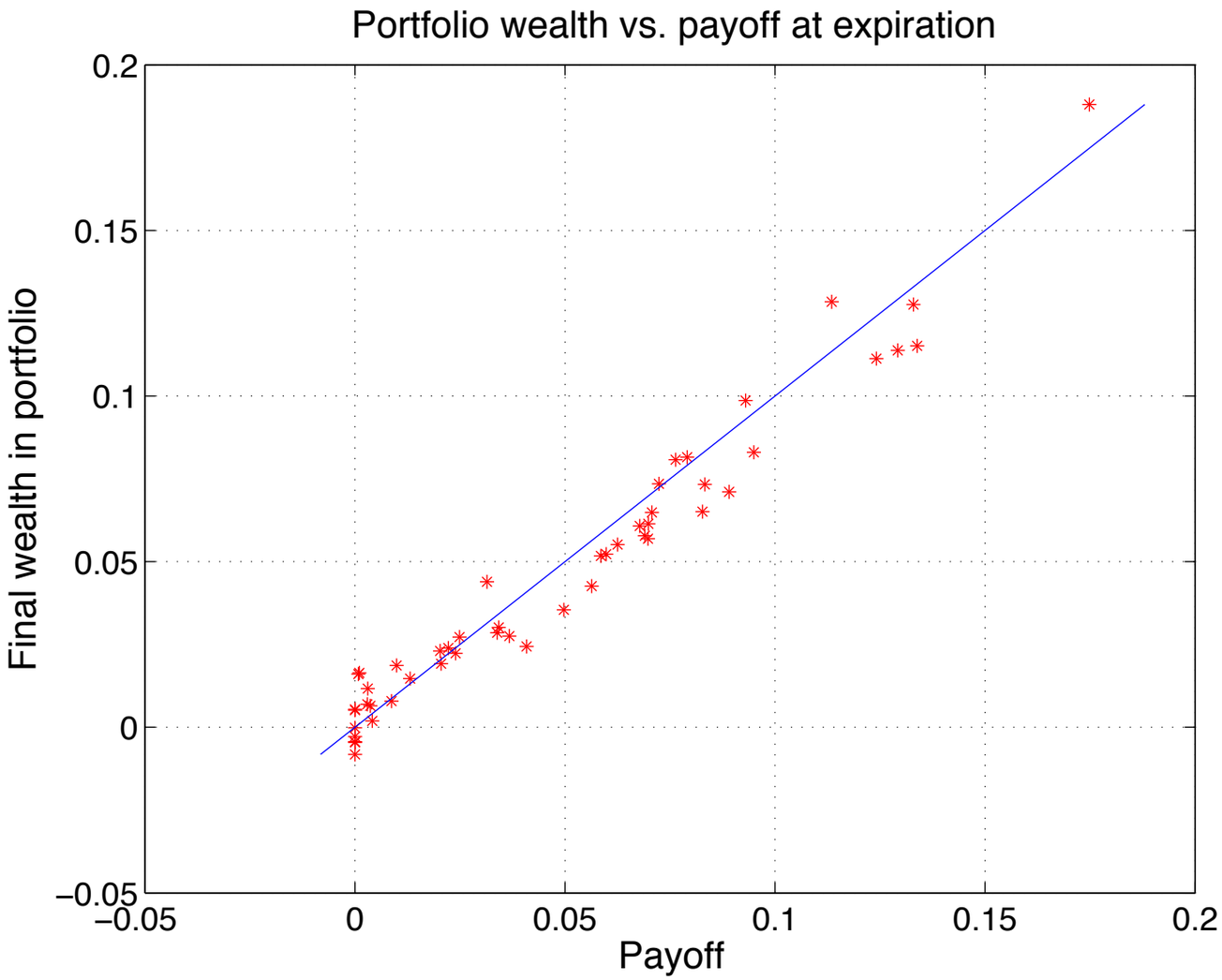
  Example: Napoleon cliquet, Heston model

$$p(t) = f(x(t), \sigma(t), x(t_1), \ldots, x(t_{N_{\text{fix}}})$$

- Most suitable method for estimating pricing function f:
  least-squares Monte Carlo approach based on polynomial approximations

(Longstaff, Schwartz, 2001)

Example: BS model, Napoleon cliquet



- Black-Scholes model (=log-normal)
- volatility$=0.2$, risk-free$=0.04$
- $T=24$ weeks (hedging every week)
- 50 simulations
- $M=100$ scenarios
- Pricing method: LS approximation
- **SMPC: only trade underlying stock**

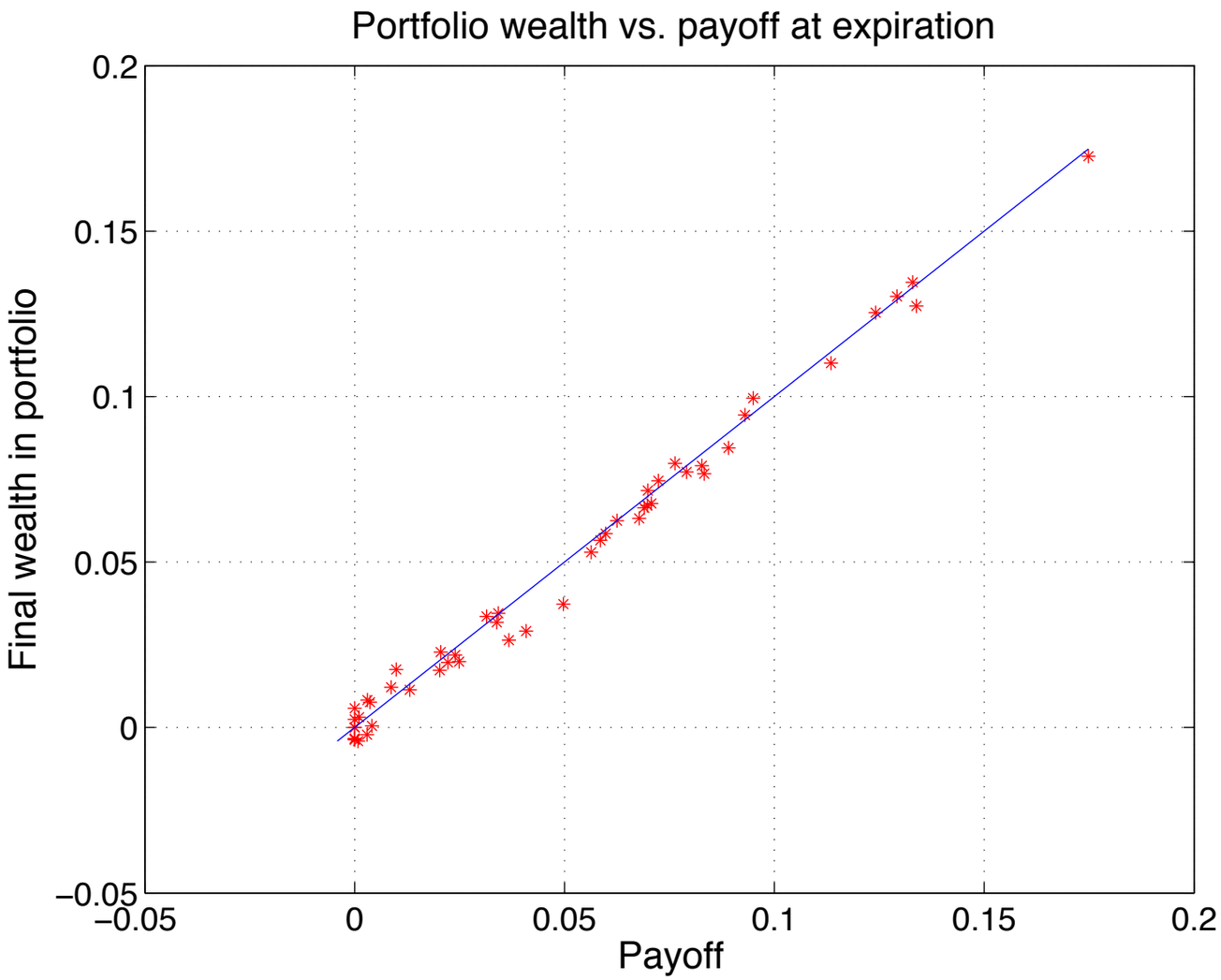
- ~~CPU time $= 1400$ ms per SMPC step~~
  ~~(Matlab R2009 on 1.86GHz Intel Core 2 Duo)~~

- CPU time $= 50.5$ ms per SMPC step
  (Matlab R2009 on 1.86GHz Intel Core 2 Duo)

- CPU time $= 76.7$ s to compute
  LS approximation (off-line)

Hedging quality very similar !

Example: BS model, Napoleon cliquet



- Black-Scholes model (=log-normal)
- volatility$=0.2$, risk-free$=0.04$
- $T=24$ weeks (hedging every week)
- 50 simulations
- $M=100$ scenarios
- Pricing method: LS approximation
- **SMPC: Trade underlying stock & European call with maturity** $t+T$

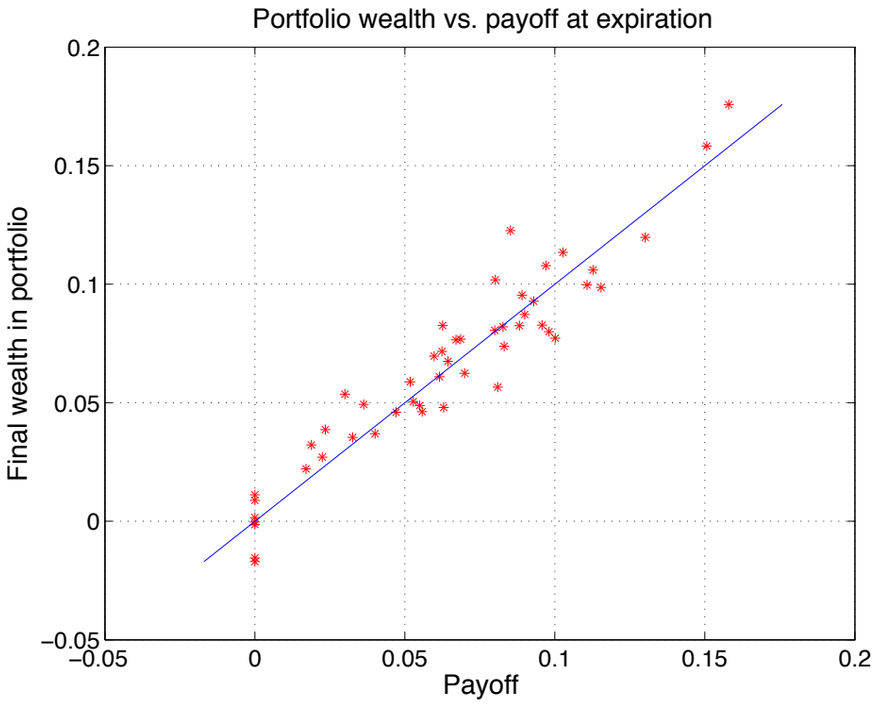
- ~~CPU time $=1625$ ms per SMPC step~~
  ~~(Matlab R2009 on 1.86GHz Intel Core 2 Duo)~~

- CPU time $=59.2$ ms per SMPC step
  (Matlab R2009 on 1.86GHz Intel Core 2 Duo)

- CPU time $=76.7$ s to compute LS approximation (off-line)

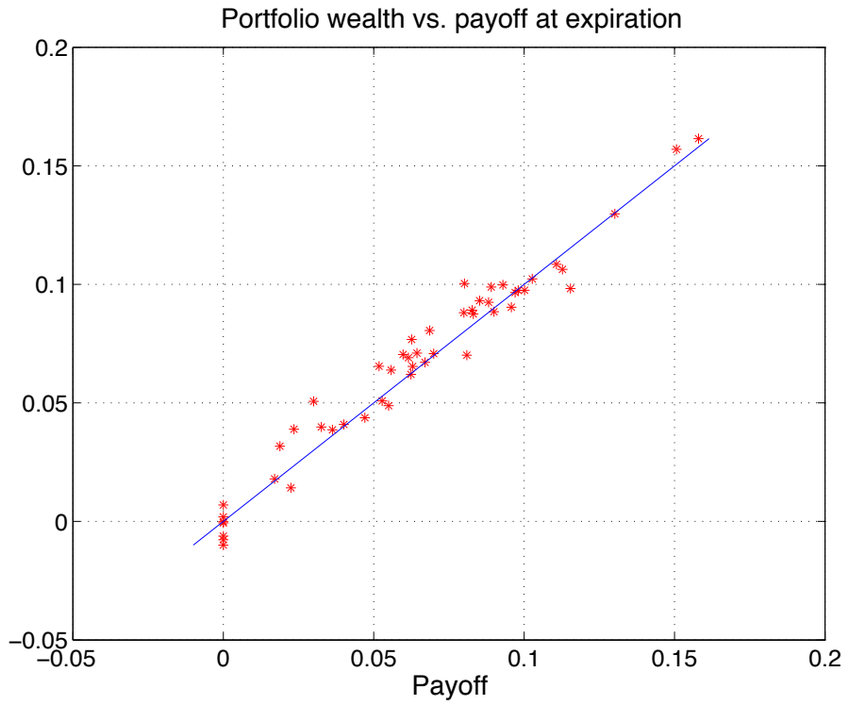
Hedging quality very similar !

Example: Heston model, Napoleon cliquet



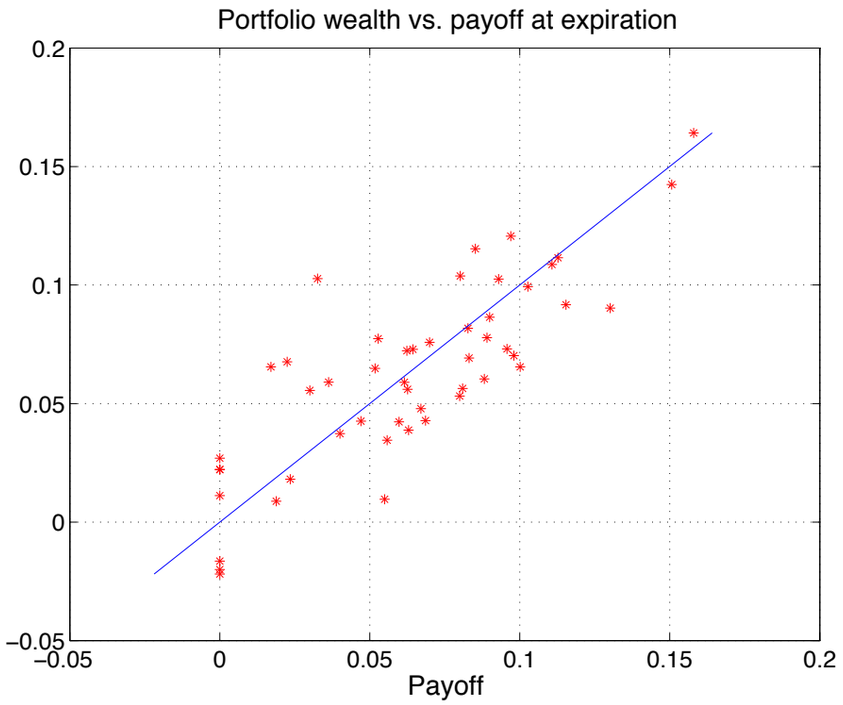
SMPC: only trade underlying stock

CPU time $=$ 220 ms per SMPC step



SMPC: Trade underlying stock & European call with maturity $t+T$

CPU time $=$ 277 ms per SMPC step



Delta hedging only trade underlying stock

CPU time $=$ 156 ms per SMPC step

CPU time $=$ 156 s to compute LS approximation (off-line)

Conclusions

- Stochastic MPC is an effective approach to address automatic decision problems related to energy markets (**real-time power dispatch** and optimal **bidding**) and financial engineering (**option hedging**)

- Pros and cons in using SMPC in energy markets:

  - ✖ Need dynamical **models**, historical **data** (renewable energy, prices, demand, ...)
  - ✖ Need to **gather data in real-time** from distributed energy resources to take decisions

  - ✓ Ability to smoothly **integrate renewables**
  - ✓ Minimize **generation costs** and **imbalance costs**
  - ✓ Handle operating constraints
  - ✓ Take care of **risk figures** in an optimal way
  - ✓ Useful decision support system for market operators

IMT INSTITUTE FOR ADVANCED STUDIES LUCCA

www.imtlucca.it

ODYS OPTIMIZATION & CONTROL ENGINEERING

www.odys.it

Bibliography

[1] A. Bemporad, L. Bellucci, and T. Gabbriellini, “Dynamic option hedging via stochastic model predictive control based on scenario simulation, ”Quantitative Finance, 2012.

[2] P. Patrinos, S. Trimboli, and A. Bemporad, “Stochastic MPC for real-time market-based optimal power dispatch,” in Proc. 50th IEEE Conf. on Decision and Control and European Control Conf., Orlando, FL, 2011, pp. 7111–7116.

[3] L. Puglia, D. Bernardini, and A. Bemporad, “A multi-stage stochastic optimization approach to optimal bidding on energy markets,” in Proc. 50th IEEE Conf. on Decision and Control and European Control Conf., Orlando, FL, 2011, pp. 1509–1514.

[4] H. Heitsch and W. Romisch, "Scenario tree generation for multi-stage stochastic programs”, Stochastic Optimization Methods in Finance and Energy, pp. 313-341, 2001.