

MODEL PREDICTIVE CONTROL

STOCHASTIC MPC

Alberto Bemporad

imt.lu/ab

COURSE STRUCTURE

- ✓ Basic concepts of model predictive control (MPC) and linear MPC
- ✓ Linear time-varying and nonlinear MPC
- ✓ Quadratic programming (QP) and explicit MPC
- ✓ Hybrid MPC
 - Stochastic MPC
 - Learning-based MPC

Course page:

http://cse.lab.imtlucca.it/~bemporad/mpc_course.html

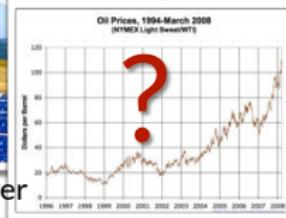
STOCHASTIC MODEL PREDICTIVE CONTROL

OPTIMIZE DECISIONS UNDER UNCERTAINTY

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



renewable power



prices



water



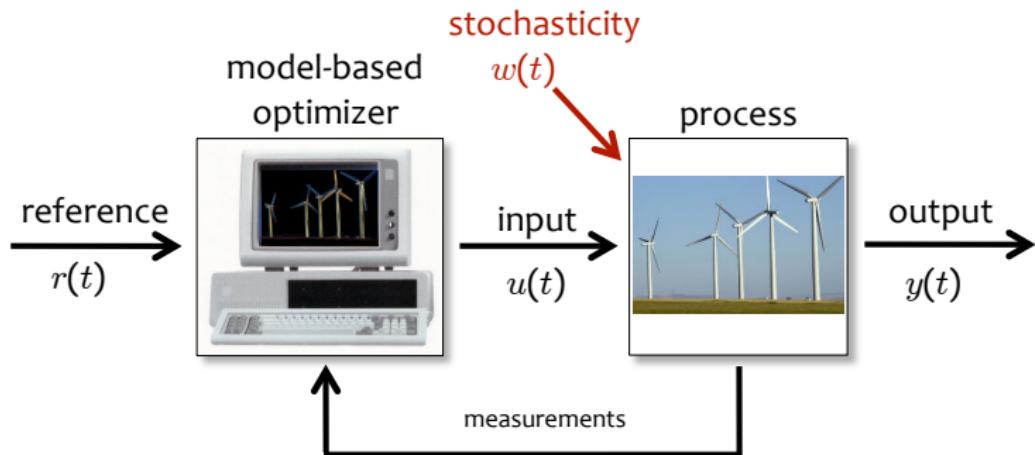
demand



human interaction

- 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** is often sought (ex: minimize expected economic cost)

STOCHASTIC MODEL PREDICTIVE CONTROL (SMPC)

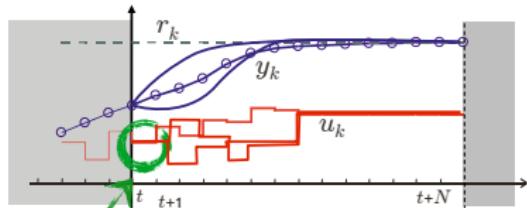


Use a **stochastic** dynamical **model** of the process to **predict** its possible future evolutions and choose the “best” **control** action

STOCHASTIC MODEL PREDICTIVE CONTROL

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

$$\begin{aligned} \min_u \quad & E_w \left[\sum_{k=0}^{N-1} \ell(y_k, u_k, w_k) \right] \\ \text{s.t.} \quad & x_{k+1} = A(w_k)x_k + B(w_k)u_k + f(w_k) \\ & y_k = C(w_k)x_k + D(w_k)u_k + g(w_k) \\ & u_{\min} \leq u_k \leq u_{\max} \\ & y_{\min} \leq y_k \leq y_{\max}, \quad \forall w \quad \text{robustness} \\ & x_0 = x(t) \quad \text{feedback} \end{aligned}$$



$x(t)$ = 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 ...

LINEAR STOCHASTIC MODEL W/ DISCRETE DISTURBANCE

- Linear stochastic prediction model

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

possibly subject to stochastic output constraints $y_{\min}(\mathbf{w}_k) \leq y_k \leq y_{\max}(\mathbf{w}_k)$

- Stochastic discrete disturbance

$$w_k \in \{w^1, \dots, w^s\}$$

with discrete probabilities $p_j = \Pr [w_k = w^j], p_j \geq 0, \sum_{j=1}^s p_j = 1$

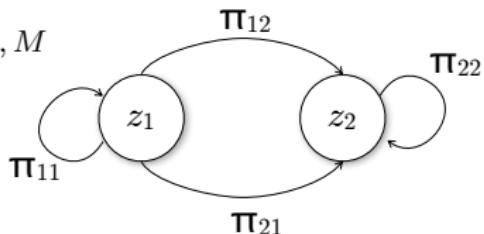
- (A, B, C) can be sparse matrices (e.g., network of interacting subsystems)
- Often w_k is low-dimensional (e.g., driver's power request, obstacle velocities, electricity price, weather, ...)

LINEAR STOCHASTIC MODEL W/ DISCRETE DISTURBANCE

- Probabilities p_j can be time varying, $p_j(t)$, and have their own dynamics

Example: Markov chain

$$\pi_{ih} = \Pr[z(t+1) = z_h \mid z(t) = z_i], \quad i, h = 1, \dots, M$$
$$p_j(t) = \begin{cases} e_{1j} & \text{if } z(t) = z_1 \\ \vdots & \vdots \\ e_{Mj} & \text{if } z(t) = z_M \end{cases}$$

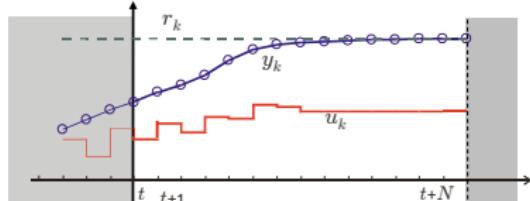


- Discrete distributions can be **estimated from historical data** (and adapted on-line)

COST FUNCTIONS FOR SMPC TO MINIMIZE

- Expected performance

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



- Tradeoff between expectation & risk

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

$$\alpha \geq 0$$

- Note that they coincide for $\alpha=1$, since

$$\text{Var}_w [y_k - r_k] = E_w [(y_k - r_k)^2] - (E_w [y_k - r_k])^2$$

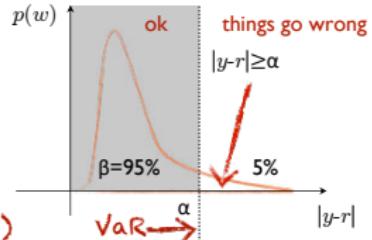
COST FUNCTIONS FOR SMPC TO MINIMIZE

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

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

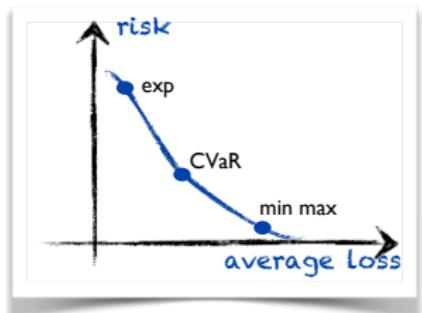
= minimize expected loss when things go wrong (convex !)

= expected shortfall



- Min-max = minimize worst-case performance

$$\min_u \sum_{k=0}^{N-1} \max_w |y_k - r_k|$$



COST FUNCTIONS FOR SMPC TO MINIMIZE

- CVaR optimization (Rockafellar, Uryasev, 2000)

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

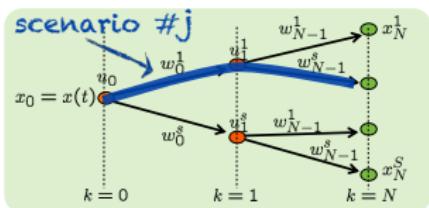


$$\begin{aligned} & \min_{u, z, \alpha} \sum_{k=0}^{N-1} \alpha_k + \frac{1}{1-\beta} \sum_{j=1}^S p^j z_k^j \\ \text{s.t.} \quad & z_k^j \geq y_k^j - r_k^j - \alpha_k \\ & z_k^j \geq r_k^j - y_k^j - \alpha_k \\ & z_k^j \geq 0 \end{aligned}$$

CVaR optimization becomes a linear programming problem

STOCHASTIC OPTIMAL CONTROL PROBLEM

- Enumerate all possible scenarios $\{w_0^j, w_1^j, \dots, w_{N-1}^j\}, j = 1, \dots, S$
- Scenario = path on the tree
- Number S of scenarios = number of leaf nodes



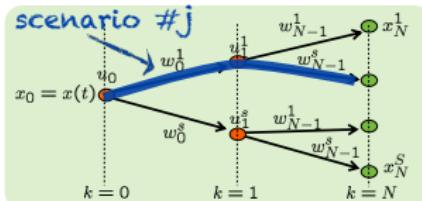
- Each scenario has probability $p_j = \prod_{k=0}^{N-1} \Pr[w_k = w_k^j]$

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!

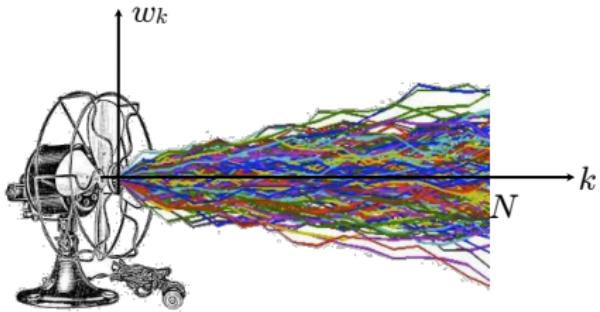
Example: $\min E_w \left[x_N' P x_N + \sum_{k=0}^{N-1} x_k' Q x_k + u_k' R u_k \right]$

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

Expectations of quadratic costs remain quadratic costs

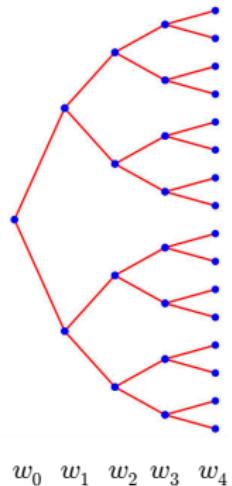
SCENARIO TREE GENERATION FROM DATA

- 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



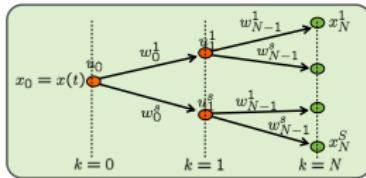
**Heuristic
Multilevel
Clustering**

(Heitsch, Römisch, 2009)



- **Alternatives** (simpler but less accurate): use histograms (only for $w_k \in \mathbb{R}$) or **K-means** (also in higher dimensions), within a recursive algorithm

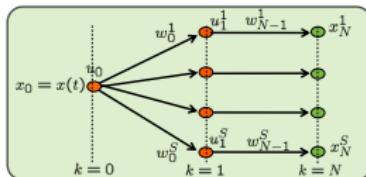
FREE CONTROL VARIABLES



Stochastic control (scenario tree)

Causality constraints: $u_k^j = u_k^h$ when scenarios j and h share the same node at prediction time k (in particular, $u_0^j \equiv u_0$ at root node $k = 0$)

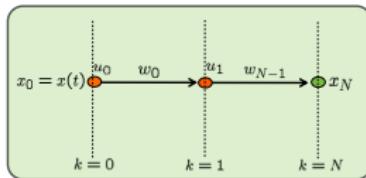
Decision u_k only depends on past disturbance realizations w_0, \dots, w_{k-1}



Stochastic control (scenario fan)

No causality in prediction: only $u_0^j \equiv u_0$ at root node.

Decision u_k depends on future disturbance realizations.

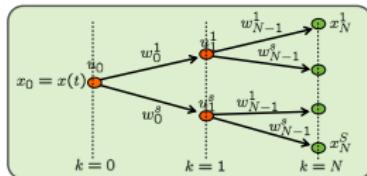


Deterministic control (single disturbance sequence)

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

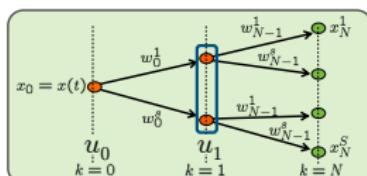
Tradeoff between **complexity** (=number of nodes) and **performance** (=accuracy of stochastic modeling)

OPEN-LOOP VS CLOSED-LOOP PREDICTION



closed-loop prediction

A different move u_k is optimized to counteract each outcome of the disturbance w_k



open-loop prediction

Only a sequence of inputs u_0, \dots, u_{N-1} is optimized, the same u_k must be good for all possible disturbances w_k

- Intuitively: OL prediction is more conservative than CL in handling constraints
- OL problem = CL problem + additional constraints (=less degrees of freedom)

LINEAR STOCHASTIC MPC FORMULATION

- A rich literature on stochastic MPC is available

(Schwarze, Nikolaou, 1999) (Munoz de la Pena, Bemporad, Alamo, 2005) (Primbs, 2007)
(Oldewurtel, Jones, Morari, 2008) (Wendt, Wozny, 2000) (Couchman, Cannon, Kouvaritakis, 2006)
(Ono, Williams, 2008) (Batina, Stoorvogel, Weiland, 2002) (van Hessem, Bosgra 2002)
(Bemporad, Di Cairano, 2005) (Bernardini, Bemporad, 2012)

See also the survey paper (Mesbah, 2016)

- Performance index: $\min E_w \left[x'_N P x_N + \sum_{k=0}^{N-1} x'_k Q x_k + u'_k R u_k \right]$
- Goal: ensure **mean-square convergence** $\lim_{t \rightarrow \infty} E[x'(t)x(t)] = 0 \quad (f(w(t)) = 0)$
- Mean-square stability ensured by **stochastic Lyapunov function** $V(x) = x'Px$

$$E_{w(t)} [V(x(t+1))] - V(x(t)) \leq -x'(t)Lx(t), \forall t \geq 0$$

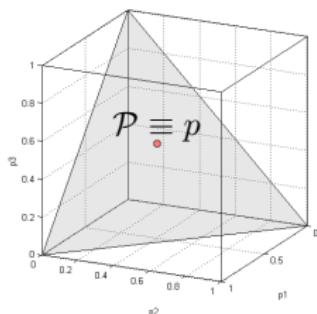
$$\begin{aligned} P &= P' \succ 0 \\ L &= L' \succ 0 \end{aligned}$$

(Morozan, 1983)

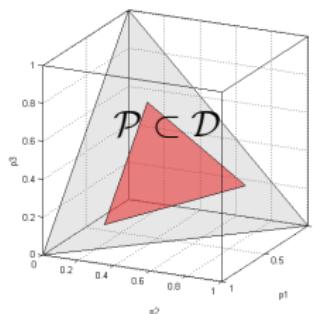
ROBUST LINEAR STOCHASTIC STABILIZATION

- The approach can be generalized to uncertain probabilities $p(t) \in \mathcal{P}$
(Example: time-varying probabilities)

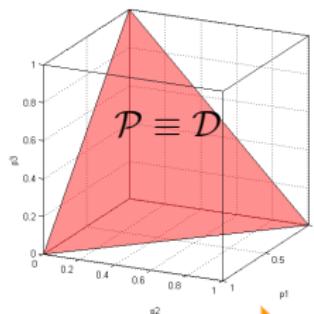
probability distribution is known



uncertain or time-varying probability distribution



no probability distribution is known, $w(t)$ can vary arbitrarily



Conservativeness

- If $\mathcal{P} \equiv \mathcal{D}$ we have a **robust** control problem (*robust convergence*)
- The more information we have about the probability distribution $p(t)$ of $w(t)$ the less conservative is the control action

STABILIZING STOCHASTIC MPC

(Bernardini, Bemporad, 2012)

- Impose stochastic stability constraint in SMPC problem
(=quadratic constraint w.r.t. u_0)

$$\begin{aligned} \min_u \quad & E_w \left[\sum_{k=0}^{N-1} \ell(x_k, u_k) \right] \\ \text{s.t.} \quad & x_{k+1} = A(w_k)x_k + B(w_k)u_k \\ & E[V(A(w_0)x_0 + B(w_0)u_0)] \leq x_0'(Q^{-1} - L)x_0 \\ & x_0 = x(t) \end{aligned}$$

performance and stability are decoupled

- SMPC approach:

- Solve LMI problem off-line to find stochastic Lyapunov fcn $V(x) = x'Q^{-1}x$
- Optimize stochastic performance based on scenario tree

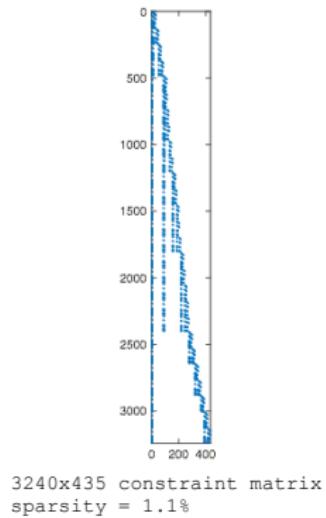
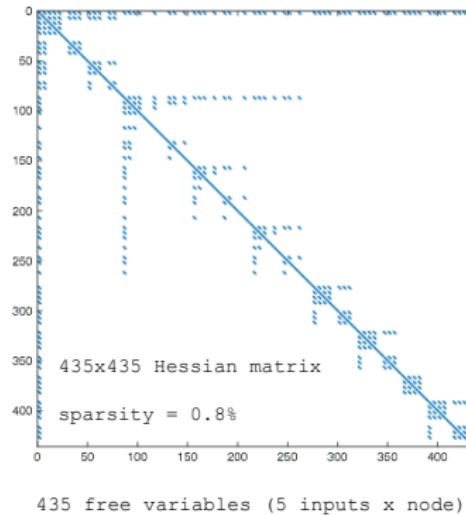
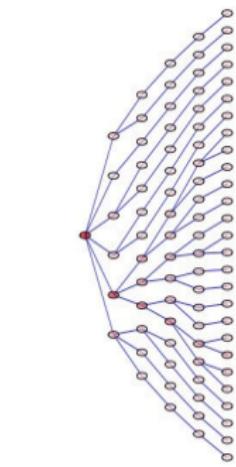
Theorem: The closed-loop system is as. stable in the mean-square sense

- SMPC can be generalized to handle **input and state constraints**

Note: recursive feasibility guaranteed by backup solution $u(k) = Kx(k)$

COMPLEXITY OF STOCHASTIC OPTIMIZATION PROBLEM

- In condensed form: #opt. vars = (# non-leaf nodes) \times (#inputs)
- Problems are **very sparse** (well exploited by **interior-point methods**)
- **Example:** SMPC with quadratic cost and linear constraints

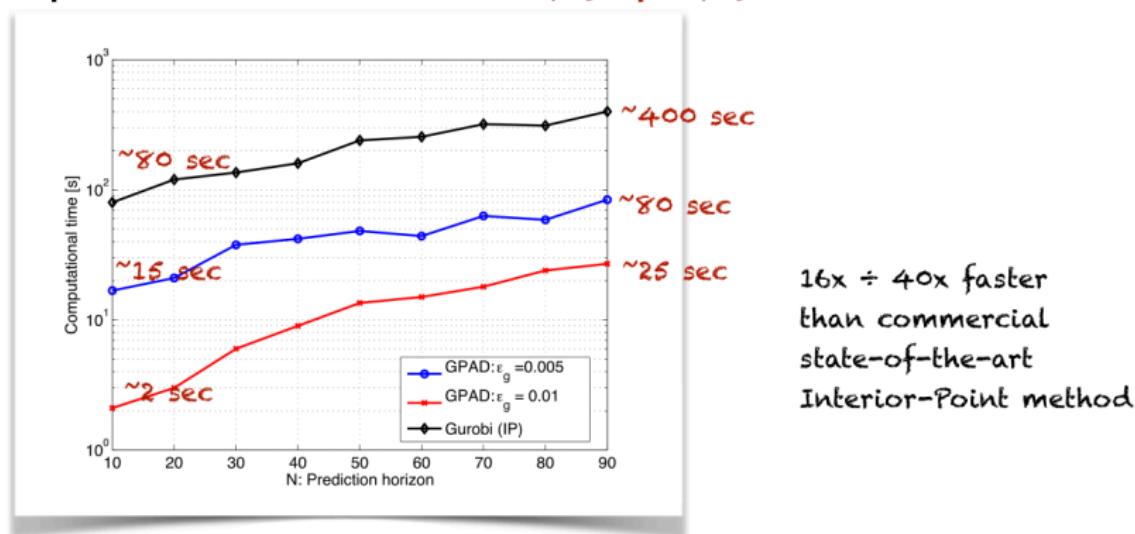


DISTRIBUTED GPAD FOR STOCHASTIC MPC

- A distributed (parallelized) variant of the Accelerated Gradient Projection applied to Dual (GPAD) for solving SMPC problems is available

(Sampathirao, Sopasakis, Bemporad, 2014)

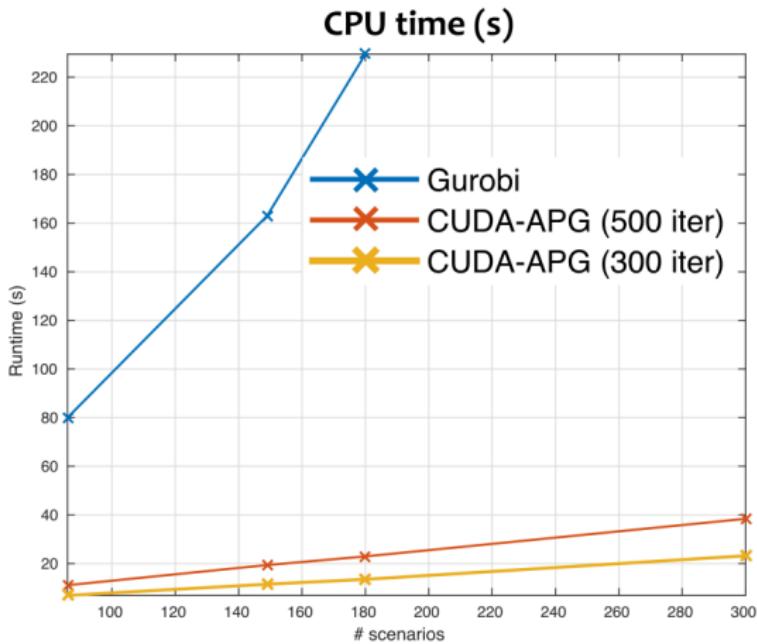
Example: stochastic MPC with **60 states, 25 inputs, 256 scenarios**



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

DISTRIBUTED GPAD FOR STOCHASTIC MPC

(Sampathirao, Sopasakis, Bemporad, 2015)



APG = Accelerated Proximal Gradient, parallel implemented on NVIDIA Tesla 2075 CUDA platform

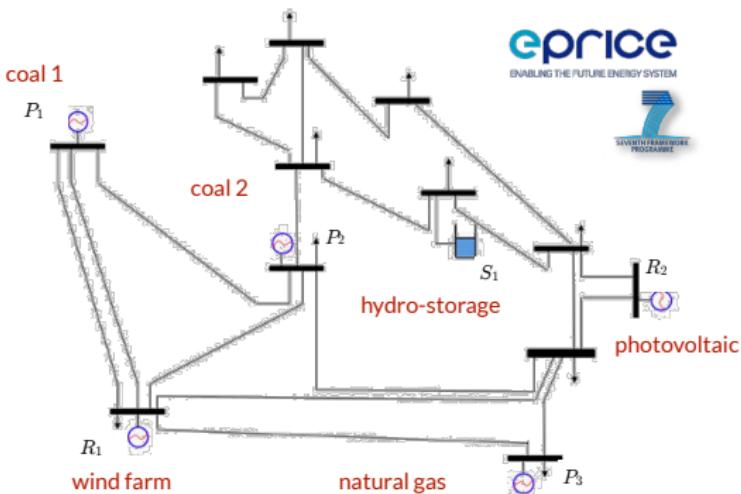
A FEW SAMPLE APPLICATIONS OF SMPC

- **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, Gabbiellini, 2009)
(Bemporad, Gabbiellini, Puglia, Bellucci, 2010)
(Bemporad, Puglia, Gabbiellini, 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)

SMPC FOR REAL-TIME OPTIMAL POWER DISPATCH

REAL-TIME POWER DISPATCH PROBLEM

(Patrinos, Trimboli, Bemporad, 2011)



- **Microgrid:** 3 conventional generators, 2 renewables, 1 storage + load
- **Goal:** minimize costs/maximize profits by trading on real-time **energy market**
- Energy demand and energy prices are **stochastic**

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:

$$P_{i,\min} \leq P_{i,k} \leq P_{i,\max}$$

constraints on power variation:

$$\Delta P_{i,\min} \leq \Delta P_{i,k} \leq \Delta P_{i,\max}$$

- Storage model

α = self discharge loss

α_c = charge efficiency

α_d = discharge efficiency

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



constraints on charge/discharge:

$$S_{\min} \leq S_k \leq S_{\max}$$

constraints on charge/discharge rate:

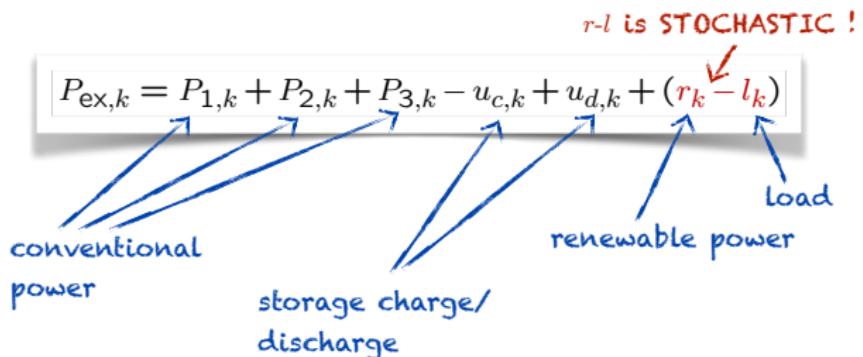
$$\Delta S_{\min} \leq S_{k+1} - S_k \leq \Delta S_{\max}$$

constraints on power flows:

$$0 \leq u_{c,k} \leq u_{c,\max}, \quad 0 \leq u_{d,k} \leq u_{d,\max}$$

POWER DISPATCH MODEL

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



- Overall linear model and constraints

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

uncontrolled input

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

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

POWER DISPATCH COST FUNCTION

- Cost function: terms to penalize

electricity price on
RT market (STOCHASTIC)

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

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:

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

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

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

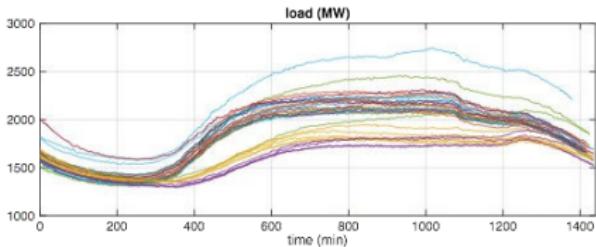
$$z = \{ \Delta P_{i,k}, u_{c,k}, u_{d,k} \}_{k=0}^{N-1}$$

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)

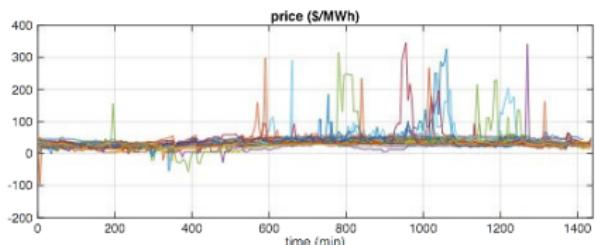


(source: <http://www.nyiso.com>)

- Historical data of price (MW)

electricity price of N.Y.C. district

(daily data of 1-31 May 2014,
sampling time = 5 min)

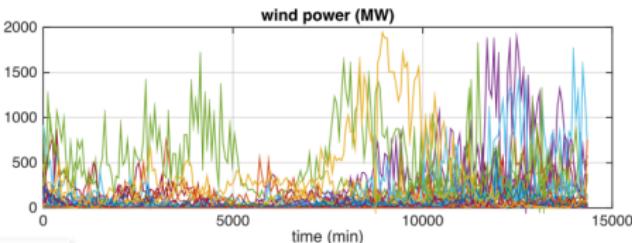


(source: <http://www.nyiso.com>)

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

$$P_w = k v_w^3$$

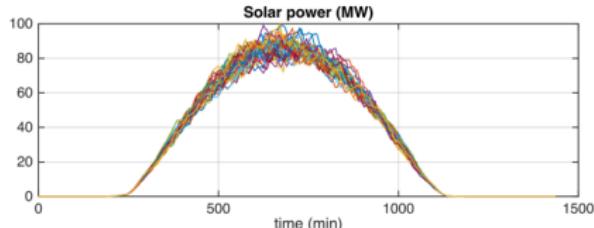
http://www.ndbc.noaa.gov/station_history.php?station=bgnn4

- Historical data of solar irradiation (W/m²)

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

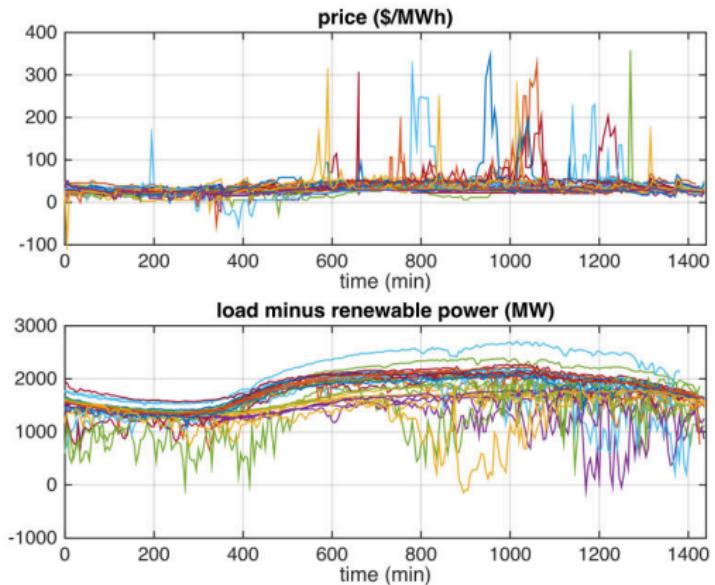
Data perturbed by noise to mimic
account cloud coefficient (unavailable)

<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):

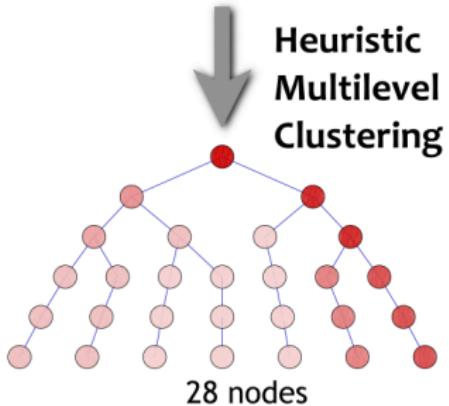
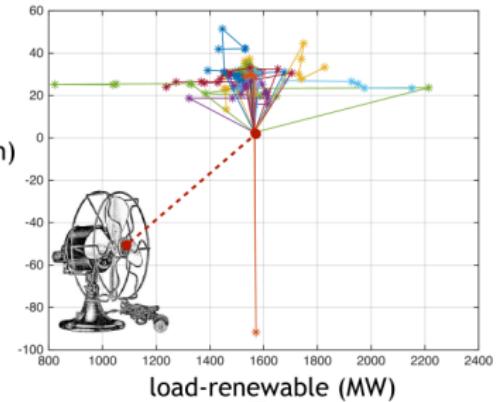
$$w^j(t+k) = v^j(t+k) - v^j(t) + v(t)$$

value at time $t+k$ in scenario # j

initial value at time t in scenario # j

stochastic vector on scenario # j

actual value at time t



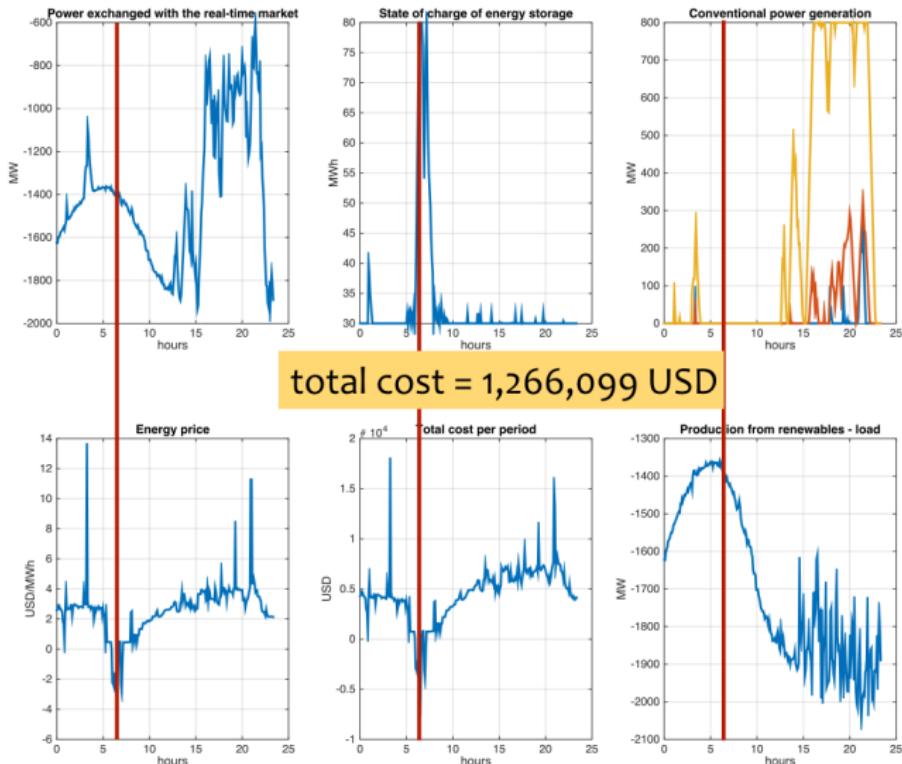
- Tree obtained from 31 scenarios (branching factor $M=[2\ 2\ 2\ 1\ 1]$)

SMPC FOR MARKET-BASED OPTIMAL POWER DISPATCH

- MPC setup:
 - Sampling time: $T_s = 5$ min
 - Prediction horizon: $N = 6$ steps ($= \frac{1}{2}$ hour ahead)
- Three controller options:
 - **Stochastic MPC**, with branching factor M (e.g., $M = [2, 2, 2, 1, 1]$)
 - **Average MPC** = deterministic MPC based on the **expected** realization (price, load minus renewable)
 - **Prescient MPC** = deterministic MPC based on the **exact** future realization (price, load minus renewable)

SMPC FOR MARKET-BASED OPTIMAL POWER DISPATCH

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



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= 1,266,095 [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

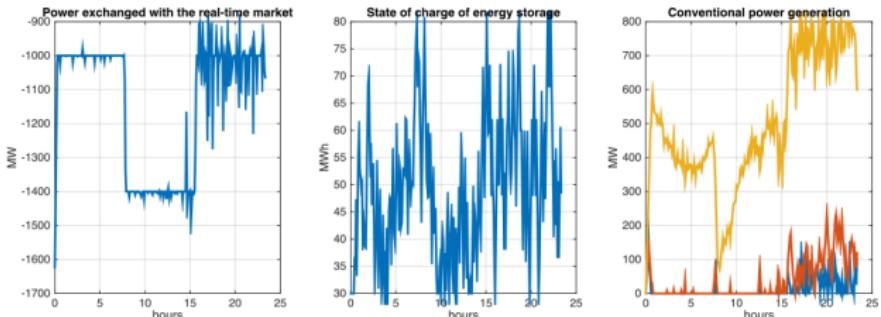
SMPC FOR MARKET-BASED OPTIMAL POWER DISPATCH

- Tracking an E-Program

$$\gamma = 10^3$$

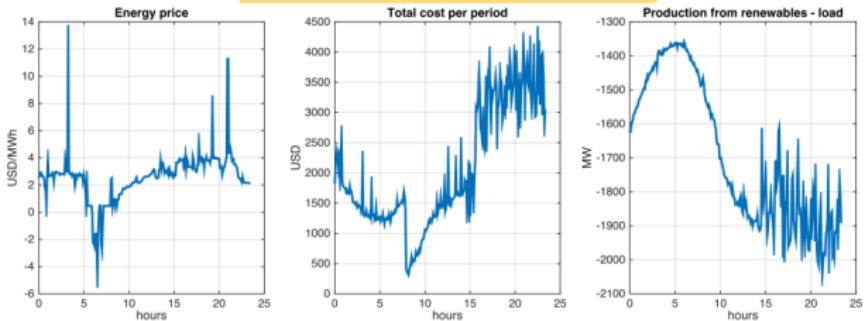
SMPC with branching factor $M=[2\ 2\ 2\ 1\ 1]$

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



total cost = 574,388 USD

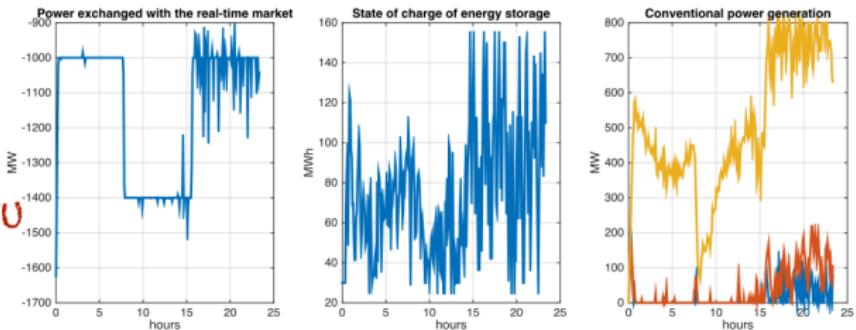
Revenues from day-ahead market are not counted



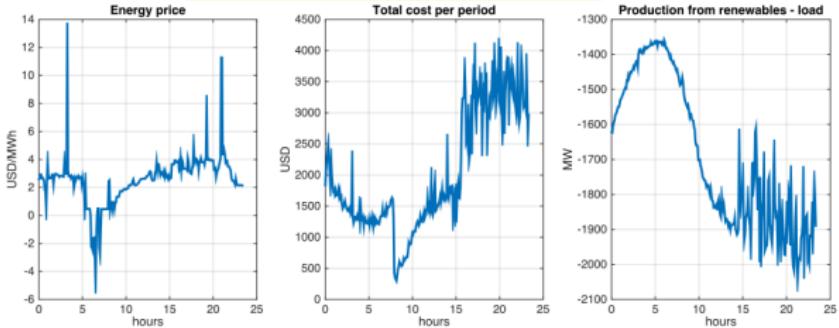
SMPC FOR MARKET-BASED OPTIMAL POWER DISPATCH

- Change storage type: $S_{\min} \leq S_k \leq S_{\max}$ $\Delta_{S,\min} \leq S_{k+1} - S_k \leq \Delta_{S,\max}$

$$\begin{aligned}S_{\min} &= 30 \\S_{\max} &= 80 \quad 150 \text{ MWh} \\ \Delta_{S,\min} &= -10 \quad \pm 40 \\ \Delta_{S,\max} &= 10 \quad \text{MWh/PTU}\end{aligned}$$



total cost = 563,283 USD

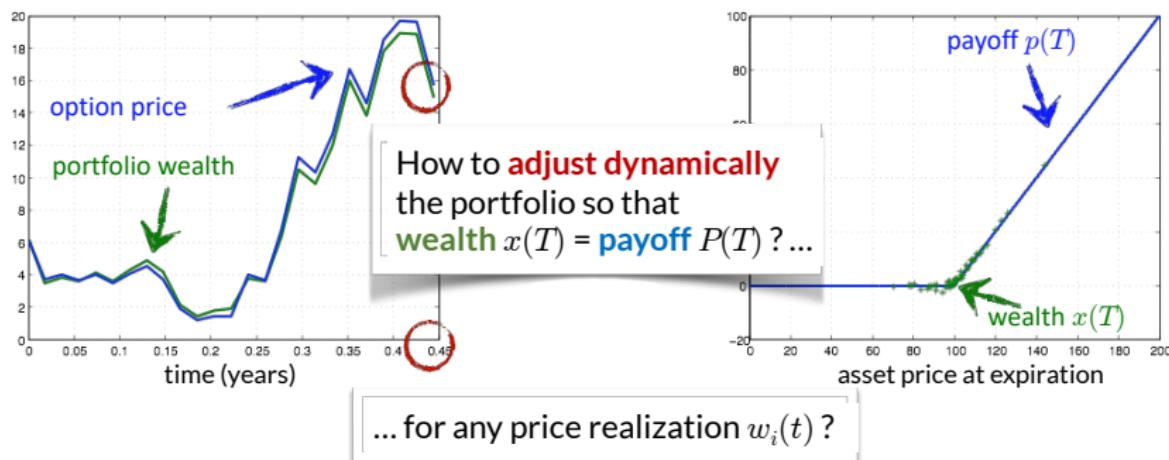


Revenues from day-ahead market are not counted

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 $x(0)$ (€)
- $x(0)$ is used to create a **portfolio** $x(t)$ of n underlying **assets** (e.g., stocks) whose prices at time t are $w_1(t), w_2(t), \dots, w_n(t)$
- At the expiration date T , the option is worth the **payoff** $p(T)$ = wealth (€) to be returned to the customer



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: $w_i(t) = \text{log-normal}$ model (used in Black-Scholes' theory)

$$dw_i = (\mu dt + \sigma dz_i)w_i \quad \text{geometric Brownian motion}$$

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



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

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

PAYOUT FUNCTION

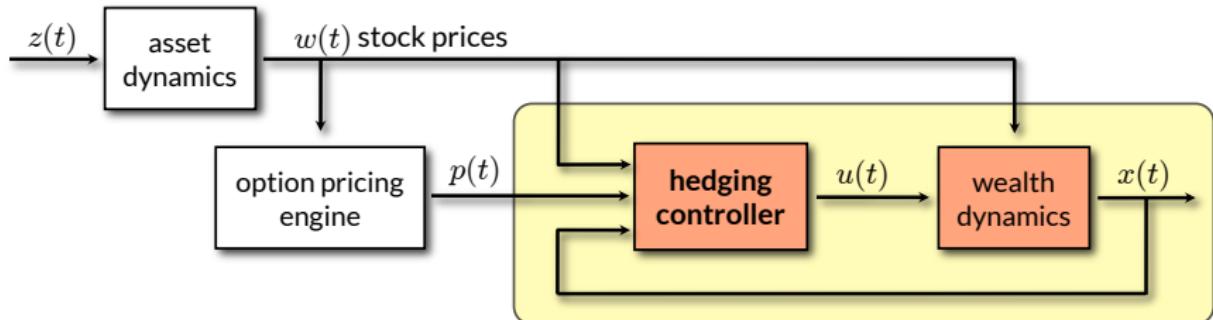
- **Stochastic disturbance:** $w(t) = [w_1(t) \dots w_n(t)]'$ (=asset prices)
- **Reference signal:** option price $p(t)$. This may depend only on $w(t)$, or also on previous prices $w(0), \dots, w(t-1)$
- Similarly $p(T)$ may either depend only on $w(T)$ or also on previous prices.
For example:
 - European call
 - Napoleon cliquet

$$p(T) = \max\{w(T), 0\}$$

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

OPTION HEDGING = LINEAR STOCHASTIC CONTROL

- Block diagram of dynamic option hedging problem:

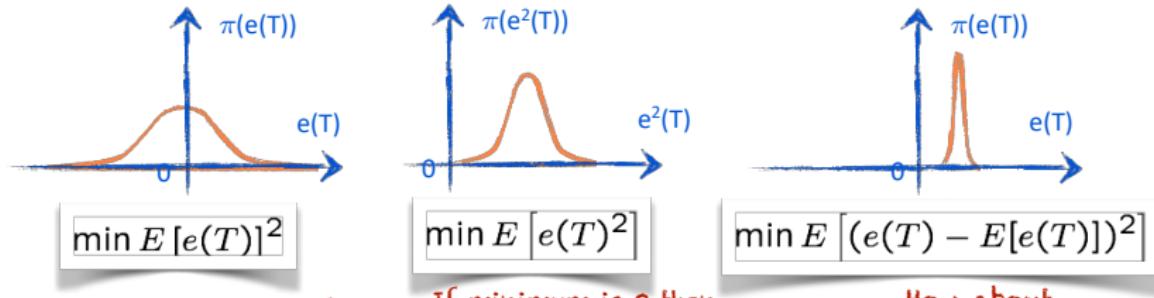


- Reference signal $p(t)$ = price of hedged option
- Control objective:** $x(T)$ should be as close as possible to $p(T)$, for any possible realization of the asset prices $w(t)$ ("tracking w/ disturbance rejection")

CONTROL OBJECTIVE

- **Tracking error** = hedging error $e(t) = x(t) - p(t)$

we want $e(T)$ small !

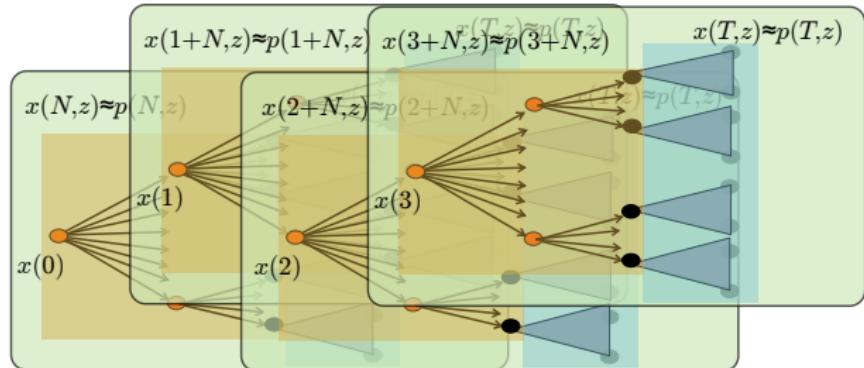


- Recall: $E[e(T)^2] = \overbrace{E[(e(T) - E[e(T)])^2]}^{\text{Var}[e(T)]} + \alpha E[e(T)]^2$ for $\alpha = 1$
- Under non-arbitrage conditions, if variance is minimized and the minimum is zero then $E[e(T)] = 0$ (Bemporad, Bellucci, Gabbriellini, 2014)

SMPC FOR DYNAMIC OPTION HEDGING

- Stochastic finite-horizon optimal control problem:

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



SMPC FOR DYNAMIC OPTION HEDGING

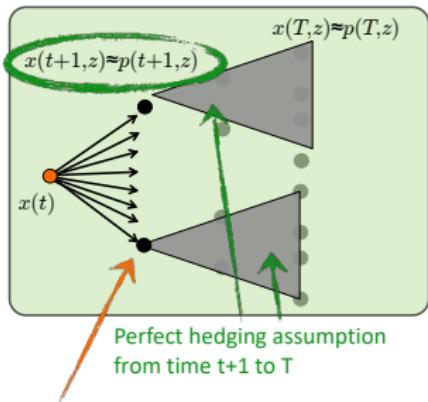
- Drawback: the longer the horizon N , the largest the number of scenarios !
- Special case: $N = 1$

minimum variance
control



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

- 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 $p(t+1, z)$ for all z
- Online optimization: very simple **least squares** problem with n variables ! ($n = \text{number of assets}$)



SMPC HEDGING ALGORITHM - SCENARIO-BASED SOLUTION

- Let t =current hedging date, $x(t)$ =wealth of portfolio, $w(t) \in \mathbb{R}^n$ = asset prices
- Use Monte Carlo simulation to generate M scenarios of future asset prices $x^1(t+1), \dots, x^M(t+1)$
- Use a pricing engine to generate the corresponding future option prices $p^1(t+1), \dots, p^M(t+1)$
- Optimize **sample variance** to get new asset quantities $u(t) \in \mathbb{R}^n$

$$\begin{aligned} \min_{u(t)} \quad & \sum_{j=1}^M \left(x^j(t+1) - p^j(t+1) - \frac{1}{M} \sum_{i=1}^M x^i(t+1) - p^i(t+1) \right)^2 \\ \text{s.t.} \quad & x^j(t+1) = (1+r)x(t) + \sum_{h=0}^n b_h^j(t)u_h(t) \end{aligned}$$

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

(Bemporad, Puglia, Gabbiellini, 2011) (Graf Plessen, Puglia, Gabbiellini, Bemporad, 2019)

SMPC HEDGING ALGORITHM - MINIMUM VARIANCE SOLUTION

- Alternatively, we can compute a covariance matrix before solving the problem

$$\begin{aligned}\text{Var}_z[x(t+1, z) - p(t+1, z)] &= \text{Var}_z \left[\underbrace{(1+r)x(t)}_{\text{deterministic}} + \sum_{i=0}^n b_i(t, z)u_i(t) - p(t+1, z) \right] \\ &= \text{Var}_z \left[\begin{bmatrix} b(t, z) \\ p(t+1, z) \end{bmatrix}' \begin{bmatrix} u(t) \\ -1 \end{bmatrix} \right] = \begin{bmatrix} u(t) \\ -1 \end{bmatrix}' \Sigma_t \begin{bmatrix} u(t) \\ -1 \end{bmatrix} \\ \Sigma_t &= \text{Var}_z \left[\begin{bmatrix} b(t, z) \\ p(t+1, z) \end{bmatrix} \right] = \begin{bmatrix} \Sigma_t^b & s_t \\ s_t' & \sigma_t^p \end{bmatrix}\end{aligned}$$

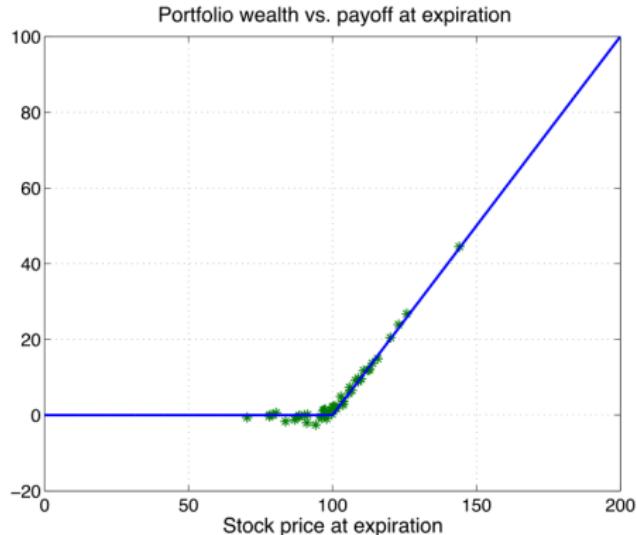
- Then, assuming Σ_t is invertible, the optimal allocation $u^*(t)$ is

$$u^*(t) = \arg \min_u \{u' \Sigma_t^b u - 2s_t' u + \sigma_t^p\} = (\Sigma_t)^{-1} s_t$$

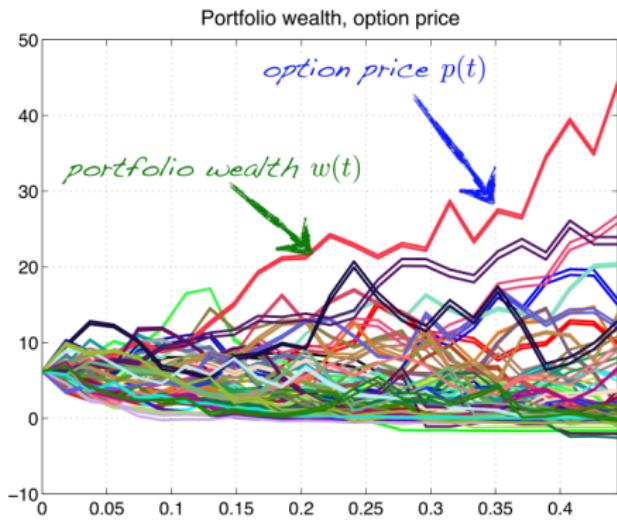
(if Σ_t not invertible, any u satisfying $\Sigma_t u = s_t$ is optimal)

- Note:** it is not always possible to compute the objective function analytically.
The scenario-based method is a more general approach

EXAMPLE: BS MODEL, EUROPEAN CALL

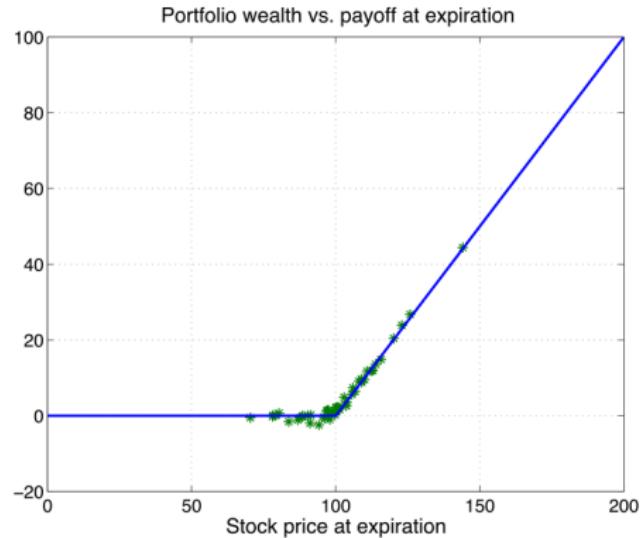


- Black-Scholes model (=log-normal)
- volatility=0.2, risk-free=0.04
- $T=24$ weeks ($\Delta t=1$ week)
- 50 simulations
- $M=100$ scenarios
- Pricing method: Monte Carlo sim.
- SMPC



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

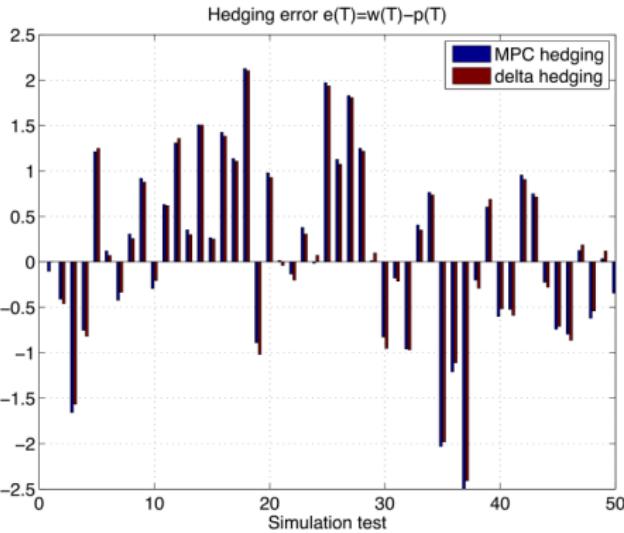
EXAMPLE: BS MODEL, EUROPEAN CALL



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

SMPC and delta-hedging are almost indistinguishable

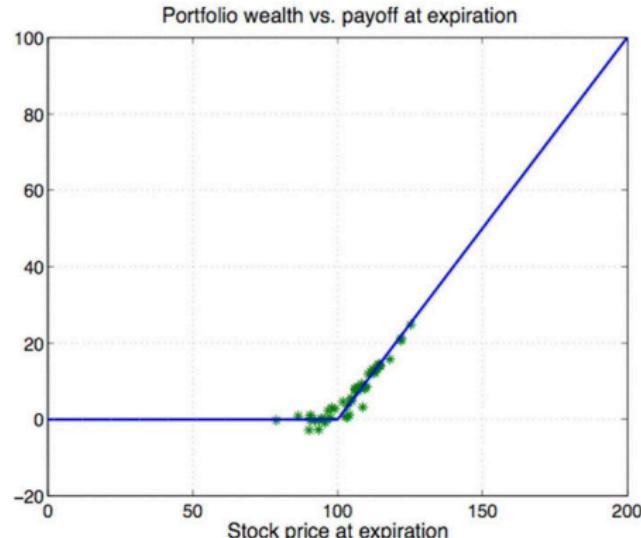
- Black-Scholes model (=log-normal)
- volatility=0.2, risk-free=0.04
- $T=24$ weeks (hedging every week)
- 50 simulations
- $M=100$ scenarios
- **Delta-hedging**



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)=-52.15, 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)= 63.016 (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



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

- **Heston's model**

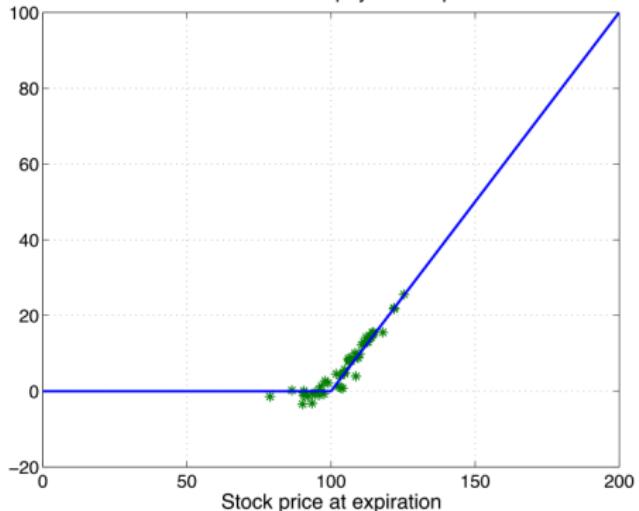
- $T=24$ weeks (hedging every week)
- 50 simulations
- $M=100$ scenarios
- risk-free=0.04
- Pricing method: Monte Carlo sim.
- **SMPC**

Heston's model

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

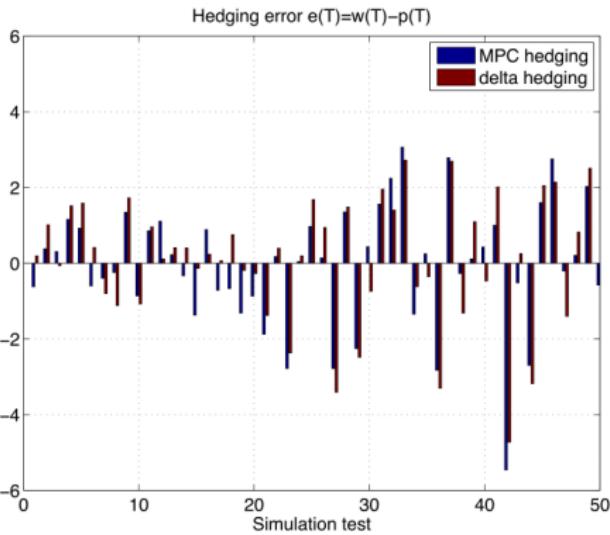
EXAMPLE: HESTON MODEL, EUROPEAN CALL

Portfolio wealth vs. payoff at expiration

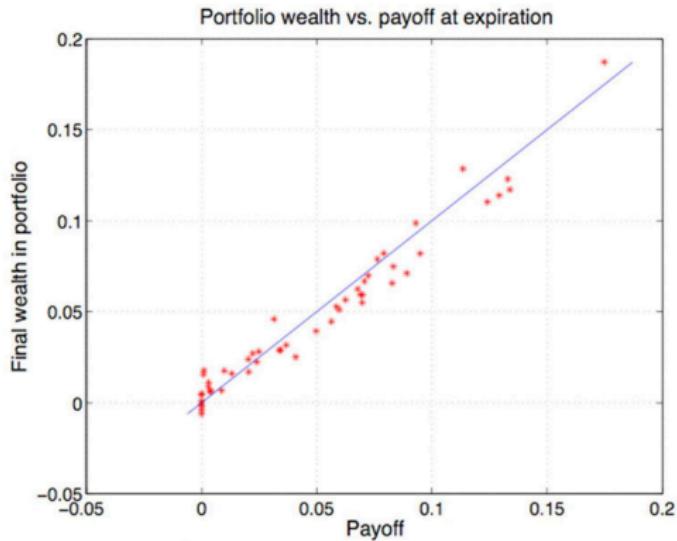


- CPU time = 1.85 ms per SMPC step
(Matlab R2009 on 1.86GHz Intel Core 2 Duo)

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



EXAMPLE: BS MODEL, NAPOLEON CLIQUET



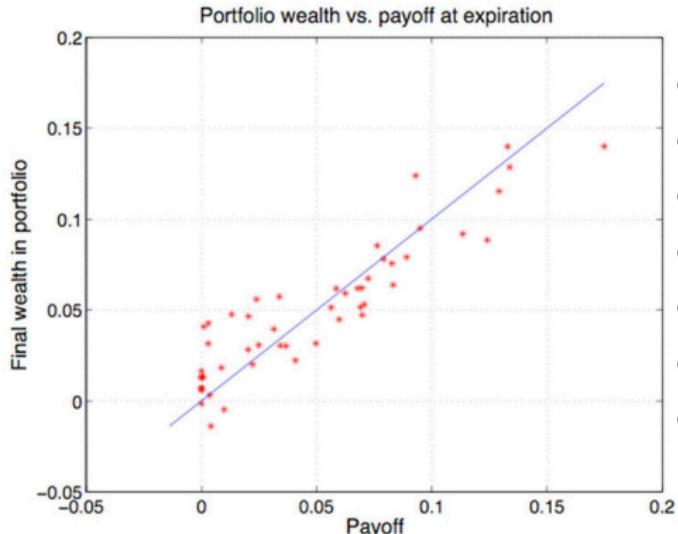
- Black-Scholes model (=log-normal)
- volatility=0.2
- $T=24$ weeks (hedging every week)
- 50 simulations
- $M=100$ scenarios
- risk-free=0.04
- Pricing method: Monte Carlo sim.
- **SMPC: only trade underlying stock**

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

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

$t_i = 0, 8, 16, 24$ weeks

EXAMPLE: BS MODEL, NAPOLEON CLIQUET



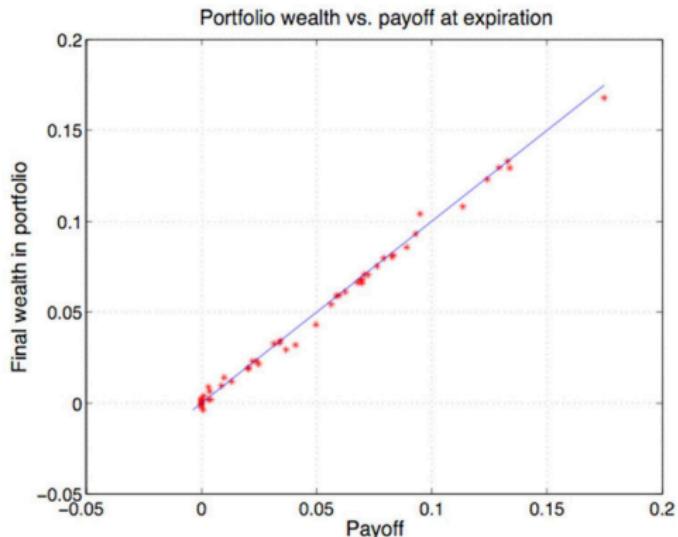
- Black-Scholes model (=log-normal)
- volatility=0.2
- $T=24$ weeks (hedging every week)
- 50 simulations
- $M=100$ scenarios
- risk-free=0.04
- **Delta hedging, only trade underlying stock**

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

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

$$t_i = 0, 8, 16, 24 \text{ weeks}$$

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 $t+T$**

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

$$t_i = 0, 8, 16, 24 \text{ weeks}$$

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

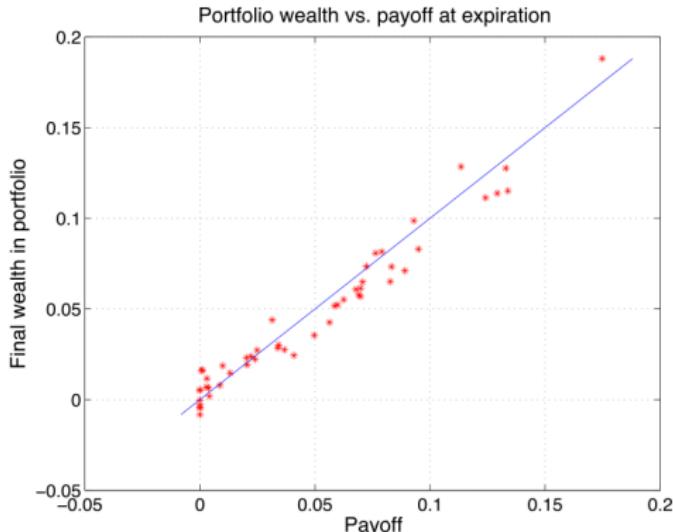
APPROXIMATE OPTION PRICING

- Bottleneck of the approach for exotic options: price M future option values $p^1(t+1), \dots, p^M(t+1)$
- Monte Carlo pricing can be time consuming: say L scenarios to evaluate a single option value \Rightarrow need to simulate ML paths to build the optimization problem (e.g.: $M = 100, L = 10000, ML = 10^6$)
- **Idea:** Use **offline function approximation** techniques to estimate $p(t)$ as a function of current asset parameters and other option-related parameters
- Example: Napoleon cliquet, Heston model

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

- A suitable method for estimating pricing function f is a least-squares Monte Carlo approach based on polynomial approximations (Longstaff, Schwartz, 2001)

EXAMPLE: BS MODEL, NAPOLEON CLIQUET



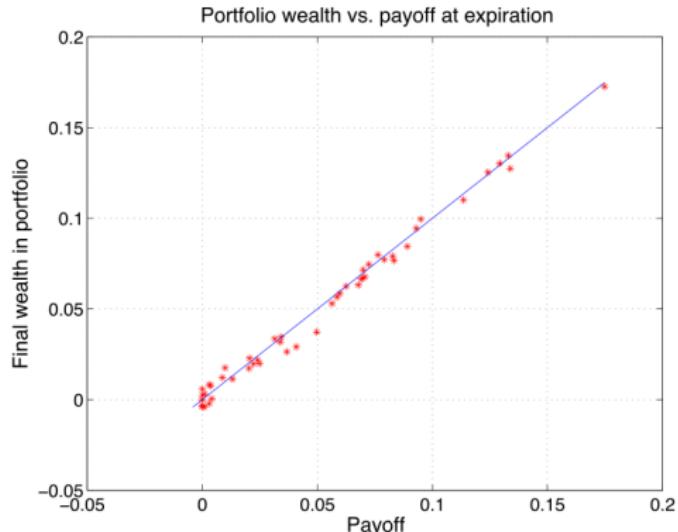
- ~~• CPU time = 1400 ms per SMPC step
(Matlab R2009 on 1.86GHz Intel Core 2 Duo)~~



- 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 is very similar !

EXAMPLE: BS MODEL, NAPOLEON CLIQUET



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

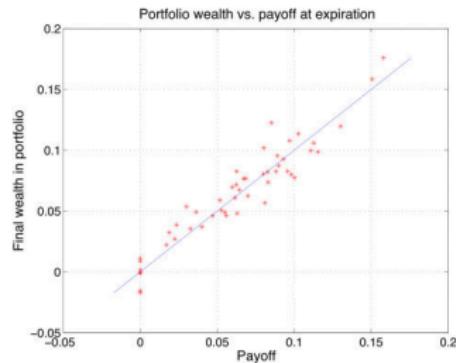


- Black-Scholes model (=log-normal)
- volatility=0.2, risk-free=0.04
- $T=24$ weeks (hedging every week)
- 50 simulations
- $M=100$ scenarios
- Pricing method: LS approximation
- **SMPC: Trade underlying stock & European call with maturity $t+T$**

- CPU time = **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)

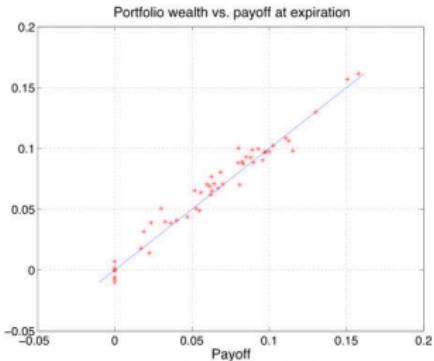
Hedging quality is very similar !

EXAMPLE: HESTON MODEL, NAPOLEON CLIQUET



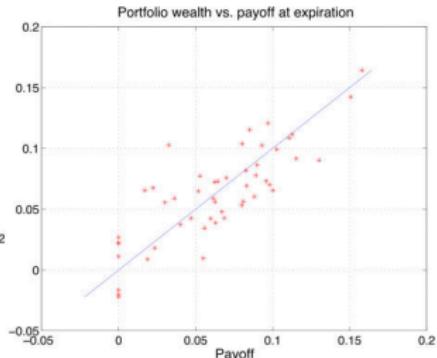
SMPC: only trade underlying stock

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)

OPTION HEDGING UNDER TRANSACTION COSTS

(Bemporad, Puglia, Gabbriellini, 2011) (Graf Plessen, Puglia, Gabbriellini, Bemporad, 2019)

- We can extend the approach to handle **transaction costs** proportional to the traded quantity $|u_i(t) - u_i(t-1)|$

$$h_i(t) = \epsilon_i |u_i(t) - u_i(t-1)| w_i(t)$$

- The portfolio dynamics becomes (Primbs, Yamada, 2008)

$$x(t+1) = (1+r) \left(x(t) - \sum_{i=1}^n h_i(t) \right) + \sum_{i=1}^n b_i(t) u_i(t)$$

- To handle $|u_i(t) - u_i(t-1)|$ we split $u_i(t) - u_i(t-1) = v_i^+(t) - v_i^-(t)$, with $v(t) = \begin{bmatrix} v^+(t) \\ v^-(t) \end{bmatrix} > 0$
- Then, the transaction cost is $h(t) = \epsilon_i (v_i^+(t) + v_i^-(t-1)) w_i(t)$
- Constraints on traded assets $u(t)$ can be translated into constraints on $v(t)$

OPTION HEDGING UNDER TRANSACTION COSTS

- It is easy to show that the **variance** of the hedging error

$e(t+1) = x(t+1) - p(t+1)$ is not affected by transaction cost, so we optimize

$$\min_{v(t)} \text{Var}[e(t+1)] + \alpha E^2[e(t+1)]$$

- In a **scenario-based** setting, we can also minimize the **CVaR** of $e(t+1)$

$$\begin{aligned} \min_{v(t), \ell(t), \{z_j(t)\}_{j=1}^M} \quad & \ell(t) + \frac{1}{1-\beta} \sum_{j=1}^M \pi_j z_j(t) \\ \text{s.t.} \quad & z_j(t) \geq \pm(x^j(t+1) - p^j(t+1)) - \ell(t) \quad j = 1, \dots, M \\ & z(t), v(t) \geq 0 \end{aligned}$$

or, still by **linear programming**, the **worst-case** hedging error

$$\begin{aligned} \min_{v(t), \ell(t)} \quad & \ell(t) \\ \text{s.t.} \quad & \ell(t) \geq \pm(x^j(t+1) - p^j(t+1)) \quad j = 1, \dots, M \\ & \ell(t), v(t) \geq 0 \end{aligned}$$

HEDGING RESULTS

- Stock prices generated by **log-normal** model in discrete-time

$$w_i(t+1) = w_i(t) e^{(\mu_i - \frac{1}{2} \sigma_i^2) T_s + \sigma_i \sqrt{T_s} \eta_i(t)}$$

T_s = trading interval, $\eta_i(t) \sim \mathcal{N}(0, 1)$, $\forall i = 1, \dots, n$

- $M = 100$ or 1000 scenarios with equal probability $\pi_i = \frac{1}{M}$, or $M = 5$ scenarios with π_i obtained from sampling a Gaussian distribution

Model	Monte Carlo $M = 100$					Monte Carlo $M = 1000$					discretized Gaussian $M = 5$				
	$E[e(T)]$	$E[e(T)]$	$\min(e(T))$	$\text{Var}[e(T)]$	CPU(s)	$E[e(T)]$	$E[e(T)]$	$\min(e(T))$	$\text{Var}[e(T)]$	CPU(s)	$E[e(T)]$	$E[e(T)]$	$\min(e(T))$	$\text{Var}[e(T)]$	CPU(s)
QP-Var	-2.30	2.73	-14.91	12.13	0.0247	-1.81	2.56	-12.30	10.81	0.0453	-1.64	2.69	-12.87	11.42	0.022
LP-CVaR	-1.34	2.58	-7.55	8.18	0.0067	-1.14	2.38	-5.99	7.13	0.0671	-1.31	2.65	-7.00	8.68	0.001
LP-MinMax	-2.67	3.83	-12.13	16.69	0.0067	-1.02	2.42	-6.49	7.93	0.31	-1.31	2.65	-7.00	8.68	0.001
Delta Hedging	-0.1312	1.77	-5.4	4.84	0.00012	-0.1312	1.77	-5.4	4.84	0.00012	-0.1312	1.77	-5.4	4.84	0.00012

European call option

Model	LS $M = 100$					LS $M = 5$					LS $M = 1000$				
	$E[e(T)]$	$E[e(T)]$	$\min(e(T))$	$\text{Var}[e(T)]$	CPU(s)	$E[e(T)]$	$E[e(T)]$	$\min(e(T))$	$\text{Var}[e(T)]$	CPU(s)	$E[e(T)]$	$E[e(T)]$	$\min(e(T))$	$\text{Var}[e(T)]$	CPU(s)
QP-Var	-2.19	6.85	-42.76	104.78	0.09	-1.06	3.82	-11.30	25.44	0.08	-1.65	10.82	-40.14	228.19	0.2001
LP-CVaR	-0.72	1.29	-12.16	7.14	0.38	-0.70	1.37	-13.63	8.55	0.08	-0.65	1.27	-11.73	6.85	0.1203
LP-MinMax	-0.72	1.29	-12.16	7.14	0.38	-0.70	1.37	-13.63	8.55	0.08	-0.72	1.33	-12.44	7.40	0.1223
Delta Hedging	-0.70	1.79	-16.14	13.61	0.0041	-0.70	1.79	-16.14	13.61	0.0041	-0.70	1.79	-16.14	13.61	0.0041

barrier option

SMPC FOR AUTOMOTIVE CONTROL

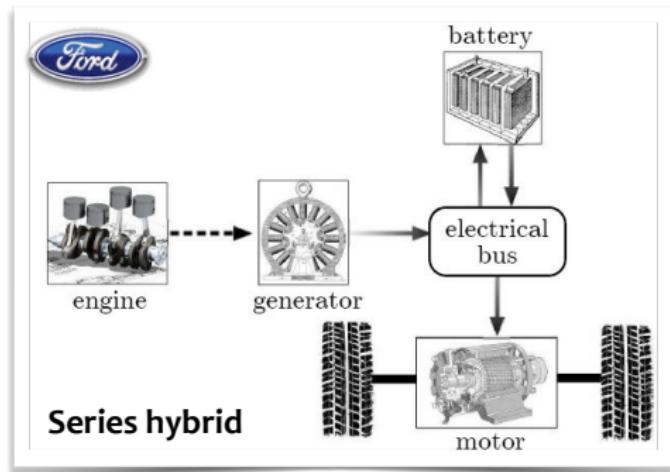
SMPC FOR HYBRID ELECTRIC VEHICLES (HEVs)

Control problem:

(Bichi, Ripaccioli, Di Cairano, Bernardini, Bemporad, Kolmanovsky, CDC 2010)

Decide optimal generation of **mechanical power** (from engine) and **electrical power** (from battery) to satisfy **driver's power request**

What will the future power request from the driver be ?



$P_{req}(w(t))$ = driver's power request

$$P_{req}(k) = P_{el}(k) + P_{mec}(k) - P_{br}(k)$$

LEARNING A STOCHASTIC MODEL OF THE DRIVER

- The driver action on the vehicle is modeled by the **stochastic** process $w(k)$
- Assume that the realization $w(k)$ can be **measured** at every time step k
- Depending on the **application**, $w(k)$ may represent different quantities (e.g., power request in an HEV, acceleration, velocity, steering wheel angle, ...)

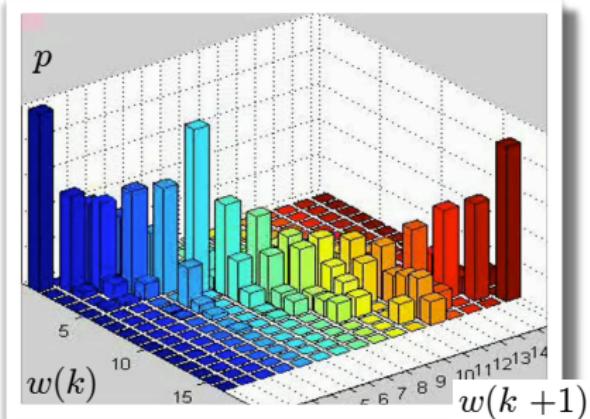
Good model for control purposes: $w(k)$ = **Markov chain**

$$[T]_{ij} = \mathbf{P}[w(k+1) = w_j | w(k) = w_i]$$

Number of states in Markov chain determines the **trade-off** between complexity and accuracy

Transition probability matrix T is easily estimated from driver's data

Several model improvements are possible (e.g., multiple Markov chains)



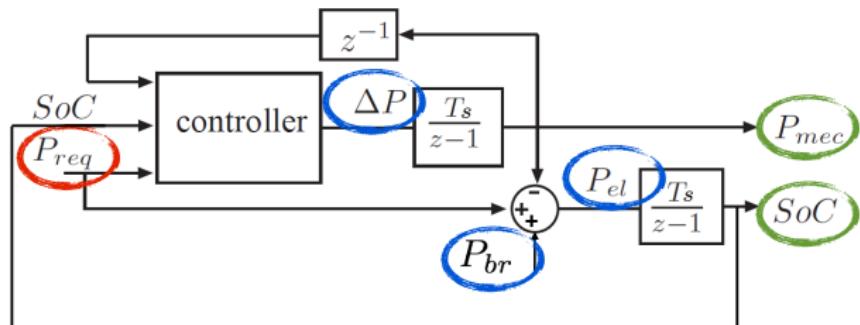
SMPC PROBLEM FOR HEV POWER MANAGEMENT

Manipulates inputs

$\Delta P(k)$, $P_{el}(k)$, $P_{br}(k)$

Uncertainty

$P_{req}(w(k))$



Controlled output

sample time $T_s = 1$ s

$$P_{req}(k) = P_{el}(k) + P_{mec}(k) - P_{br}(k)$$

Constraints

State-space equations

$$SoC(k+1) = SoC(k) - KT_s P_{el}(k)$$

$$P_{mec}(k+1) = P_{mec}(k) + \Delta P(k)$$

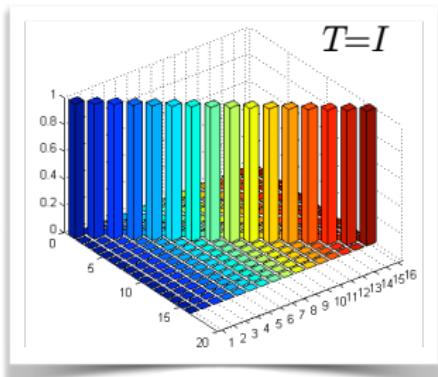
$$\begin{array}{lclclclcl} SoC_{min} & \leq & SoC(k) & \leq & SoC_{max} \\ 0 & \leq & P_{mec}(k-1) & \leq & P_{mec,max} \\ P_{el,min} & \leq & P_{el}(k) & \leq & P_{el,max} \\ \Delta P_{min} & \leq & \Delta P & \leq & \Delta P_{max} \\ 0 & \leq & P_{br}(k) & & \end{array}$$

COMPARISON WITH DETERMINISTIC MPC

“Frozen-time” MPC (FTMPC)

No stochastic disturbance model,
simply ZOH along prediction horizon

$$P_{req}(w(t+k|k)) = P_{req}(w(k))$$



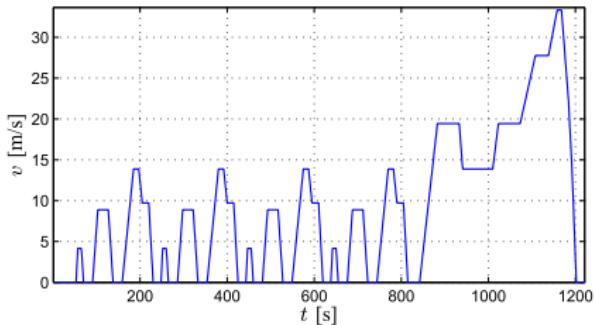
“Prescient” MPC (PMPC)

Future disturbance sequence $P_{req}(w(t+k|k))$
known in advance

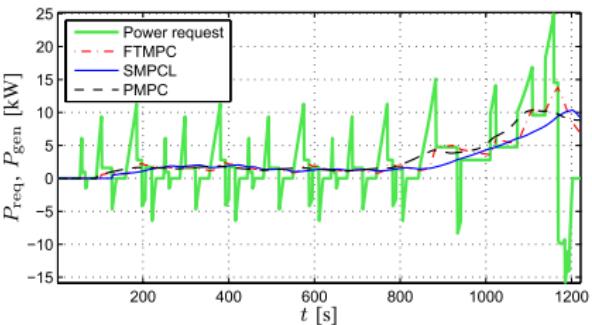


SIMULATION RESULTS

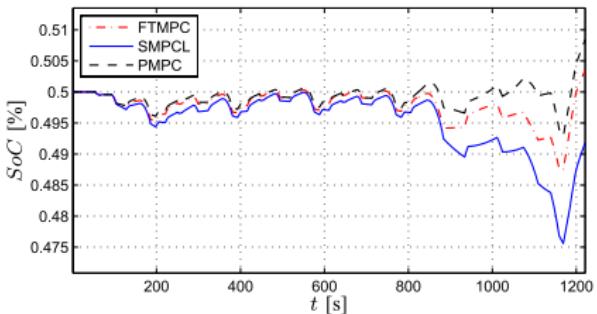
velocity



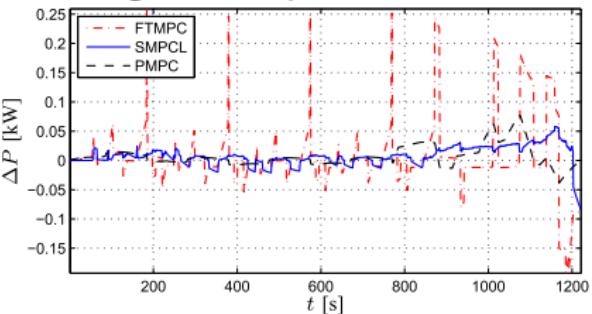
power request/generator power



state of charge



generator power variations



Results obtained on New European Driving Cycle (NEDC)

SIMULATION RESULTS: CONTROLLER COMPARISON

Comparison on different driving cycles

SHEV ENERGY MANAGEMENT SIMULATION RESULTS ON
STANDARD DRIVING CYCLES

	$\ \Delta P\ $	Fuel cons.	ΔSoC gain/loss	Equiv. fuel cons.	impr. wrt FTMPC
NEDC					
FTMPC	37.57kW	204g	0.35%	197g	—
→ SMPCL	16.28kW	166g	-0.82%	184g	6.45%
PMPC	15.25kW	196g	0.84%	177g	9.97%
FTP-75					
FTMPC	89.28kW	348g	0.64%	334g	—
→ SMPCL	26.07kW	292g	0.08%	290g	13.10%
PMPC	32.30kW	307g	0.89%	286g	14.20%
FTP-Highway					
FTMPC	39.33kW	267g	0.64%	253g	—
→ SMPCL	16.84kW	281g	2.12%	235g	7.26%
PMPC	16.33kW	254g	0.91%	234g	7.32%



pretty close to having
the crystal ball.

But we don't, we just
model uncertainty carefully

SIMULATION RESULTS: CONTROLLER COMPARISON

Comparison on different driving cycles - Real driving data

TABLE II
SIMULATION RESULTS ON REAL-WORLD DRIVING CYCLES

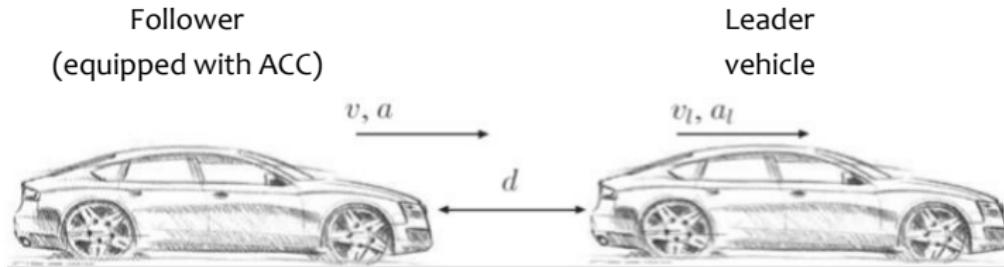
	$\ \Delta P\ $	Fuel cons.	<i>SoC</i> gain/loss	Equiv. fuel cons.	impr. wrt FTMPC
Trace #1 - smooth accelerations					
FTMPC	37.84kW	243g	-0.05%	244g	—
→ SMPCL	14.32kW	244g	0.90%	225g	8.04%
PMPC	14.08kW	223g	-0.08%	224g	8.19%
Trace #2 - steep accelerations					
FTMPC	80.61kW	327g	0.11%	323g	—
→ SMPCL	35.74kW	320g	1.16%	287g	11.34%
PMPC	30.67kW	287g	0.17%	282g	12.73%

TABLE III
PERCENTAGE IMPROVEMENT OF SMPCL STRATEGY DUE TO
ONLINE LEARNING OF THE MARKOV CHAIN

Standard cycle	Learning Improvement	Real-word Driving	Learning Improvement
NEDC	12.7%	Trace #1	1.3%
FTP-75	16.5%	Trace #2	13.4%
FTP-H.	1.1%		

STOCHASTIC MPC FOR ADAPTIVE CRUISE CONTROL

Problem setup



Goals

Control the follower acceleration variation (jerk) in order to:

- Improve safety (constraint on minimum distance)
- Improve comfort (reduce acceleration / deceleration)
- Track reference velocity



SMPC FOR ACC: PREDICTION MODEL

States

$a(k)$ acceleration

$v(k)$ velocity

$d(k)$ distance

$v_l(k)$ leader velocity

Dynamical Model

$$a(k+1) = a(k) + T_s u(k)$$

$$v(k+1) = v(k) + T_s a(k)$$

$$v_l(k+1) = v_l(k) + T_s a_l(k)$$

Inputs

$u(k)$ jerk

$a_l(k)$ leader acceleration

Uncertainty

Stochastic leader acceleration

$$a_l(k) = w(k)$$

Constraints

$$d(k) \geq d_{min}(k) = \delta + \gamma v(k) \quad \text{safety}$$

$$u_{min} \leq u \leq u_{max} \quad \text{comfort}$$

References

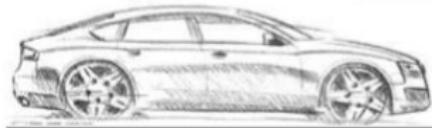
$$d_{ref}(k) = \delta_{ref} + \gamma_{ref} v(k)$$

$$v_{ref} = 26 \text{m/s}$$

SMPC FOR ACC: STOCHASTIC LEADER MODEL

Follower vehicle
(equipped with ACC)

v, a



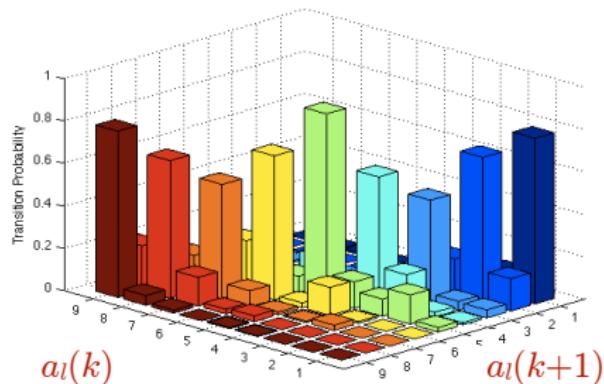
d

v_l, a_l



Leader vehicle

Leader acceleration a_l modeled by a Markov Chain (quantized in **9 states**)



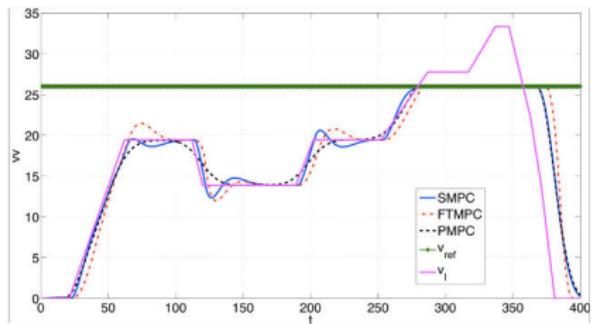
The Markov Chain is:

- Trained off-line on a collection of driving cycles (FTP, NEDC, 10-15 Mode)
- Adapted on-line by means of the learning algorithm

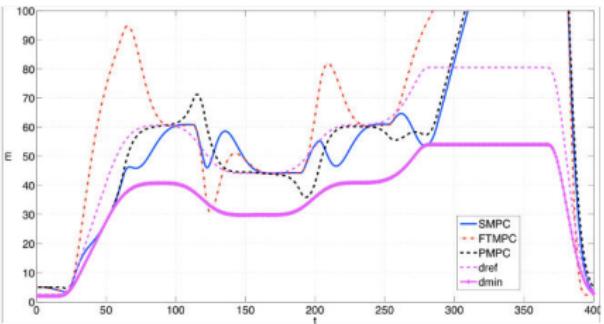


SMPC FOR ACC: SIMULATION RESULTS

Speed



Distance



Stochastic MPC (blue solid line)

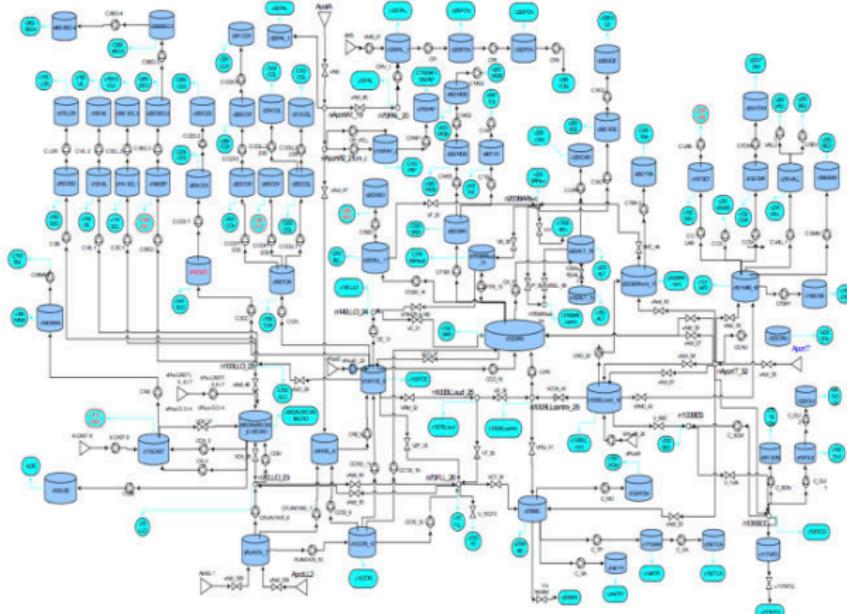
Frozen Time MPC (red dashed line)

Prescient MPC (black dashed line)

Simulation results on European Urban Driving Cycle (EUDC)

SMPC OF DRINKING WATER NETWORKS

DRINKING WATER NETWORK OF BARCELONA (SPAIN)



- General overview;

Municipalities supplied	23
Supply area	424 km ²
Population supplied	2,922,773
Average demand	7 m ³ /s

- Network parameters:

Pipes length	4.645 km
Pressure floors	113
Sectors	218

- Facilities

Remote stations	98
Water storage tanks	81
Valves	64
Flow meters	92
Pumps / Pumping stations	180 / 84
Chlorine dosing devices	23
Chlorine analyzers	74



European FP7-ICT project WIDE "DEcentralized and WIreless Control of Large-Scale Systems"

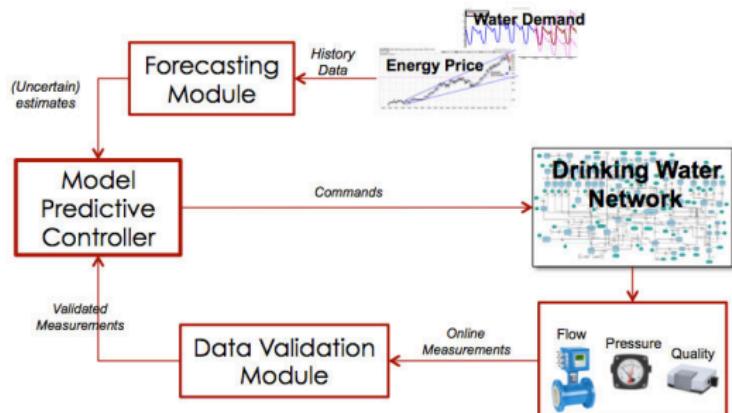


**European FP7-ICT project EFFINET
"EFFicient Integrated Real-time
Monitoring and Control of Drinking
Water NETworks"**

CONTROL OF THE DRINKING WATER NETWORK OF BCN

Main Goals:

- Reduce **electricity consumption** for pumping (€€€)
- Meet **demand** requirements
- Deliver **smooth** control actions
- Keep storage tanks above safety **limits**
- Respect the technical **limitations**: pressure limits, overflow limits & pumping capabilities



CONTROL OF THE DRINKING WATER NETWORK OF BCN

- The control objectives are translated into cost functions:

Expected total squared
water production cost
(ETSWPC) = **economic** cost

$$J^{ws} = W_\alpha^2 \sum_{l=1}^K \sum_{i=0}^{H_p-1} p^l (\alpha_1 + \alpha_{2,k})^2 (u_{k+i|k}^l)^2$$



Expected total smooth
operation cost (ETSOC)

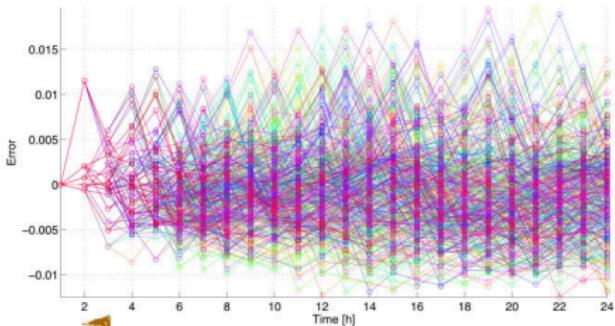
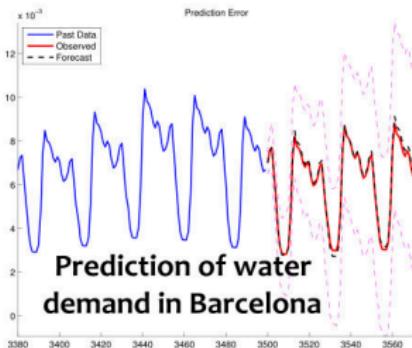
$$J^\Delta = \sum_{l=1}^K \sum_{i=1}^{H_p-1} p^l \ell^\Delta (\Delta u_{k+i|k}^l)$$

expected total **safety**
storage cost (ETSSC)

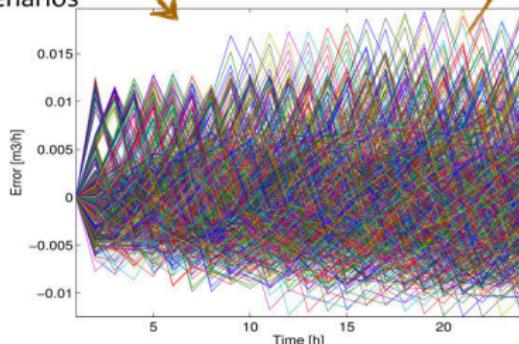
$$J^S = \sum_{l=1}^K \sum_{i=1}^{H_p} p^l \ell^s (x_{k+i|k}^l)$$

Need to minimize the total operating cost $V = J^{ws} + J^\Delta + J^S$

CONTROL OF THE DRINKING WATER NETWORK OF BCN



Uncertainty represented as a fan of scenarios



$$d_{k+i|k} = \hat{d}_{k+i|k} + \epsilon_{k+i|k}$$

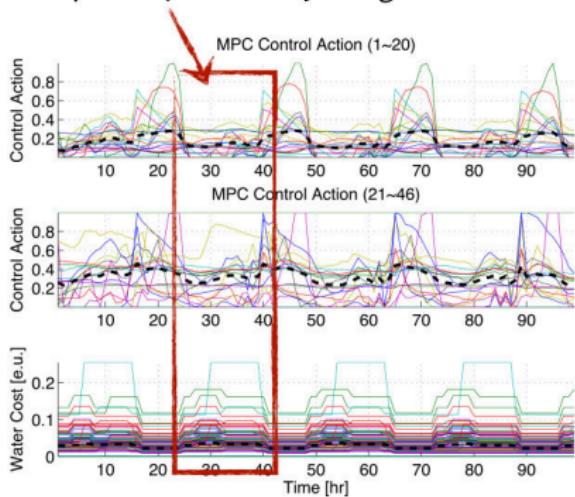
Uncertainty: demand prediction error



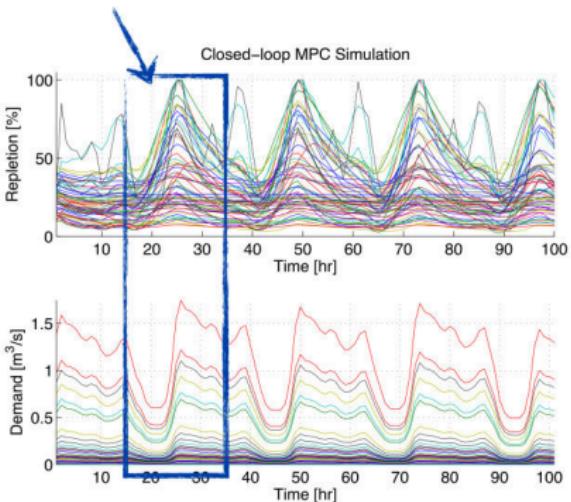
Agbar

CONTROL OF THE DRINKING WATER NETWORK OF BCN

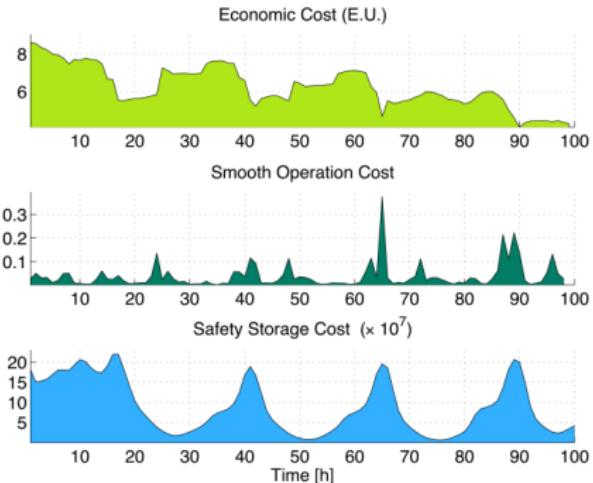
Economic: Avoid pumping when the price of electricity is high



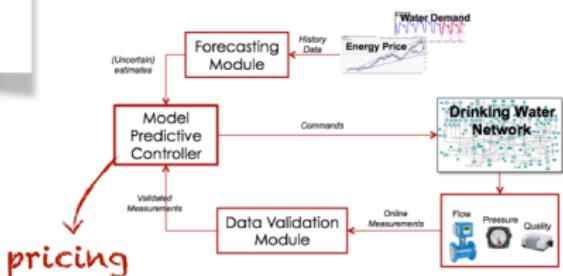
Foresight: tanks start loading up before the consumers ask for water



CONTROL OF THE DRINKING WATER NETWORK OF BCN



SMPC: The network operator has online information about the current and predicted operating cost in real time

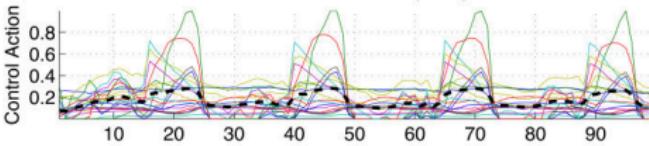
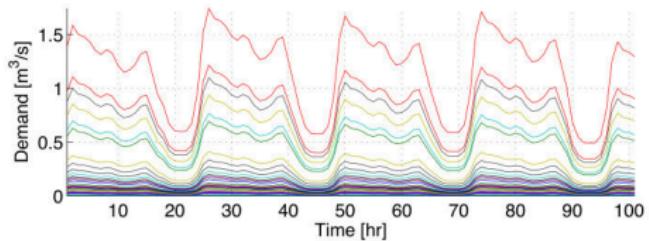


SCALABILITY OF THE SMPC APPROACH

How does the approach **scale** with the dimension of the system ?

- The dGPAD algorithm scales-up well with the size of the scenario tree (thanks to heavy parallelization)
- Scalable alternatives:
 - **Decentralized SMPC**: divide into subsystems and control each of them in parallel, exchanging some decisions **after** computations (Bemporad, Barcelli, 2010)
(others' decisions = measured disturbances)
 - **Distributed SMPC**: exchange some global variables **during** computations (Negenborn, Maestre, IEEE CSM, 2014)
- The same dGPAD algorithm can be used for decentralized SMPC (immediately), or for distributed SMPC by relaxing also the constraints that (weakly) couple the subsystems

STOCHASTIC MPC AND PARALLEL COMPUTATIONS ON GPU

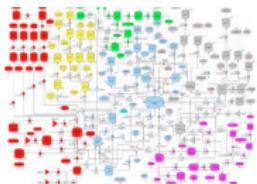


MPC-controlled network:

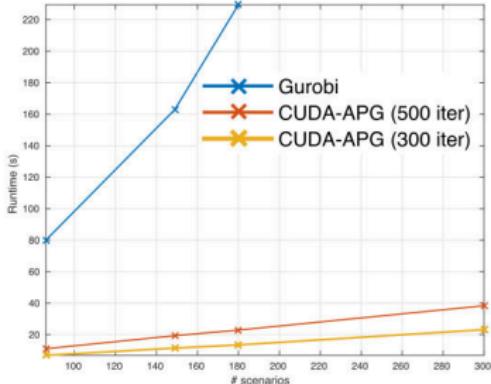
- Minimum pressure requirement hardly violated
- ~5% savings on energy cost w.r.t. current practice
- Smooth control actions
- sampling time = 1 hour

Drinking water network
of Barcelona:

63 tanks
114 controlled flows
17 mixing nodes



CPU time (s)



APG = Accelerated Proximal Gradient,
parallel implemented on NVIDIA Tesla
2075 CUDA platform

FP7-ICT project "EFFINET - Efficient Integrated Real-time Monitoring and Control of Drinking Water Networks" (2012-2015)

"Model Predictive Control" - © 2023 A. Bemporad. All rights reserved.

(Sampathirao, Sopasakis, Bemporad, 2015)

