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
• In many control problems decisions must be taken under uncertainty

• 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)
Use a **stochastic** dynamical **model** of the process to **predict** its possible future evolutions and choose the “best” **control** action.
• **At time** $t$: solve a **stochastic optimal control** problem over a finite future horizon of $N$ steps:

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

• 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

\[
\begin{align*}
    x_{k+1} &= A(w_k)x_k + B(w_k)u_k + f(w_k) \\
    y_k &= C(w_k)x_k + g(w_k)
\end{align*}
\]

possibly subject to stochastic output constraints \( y_{\text{min}}(w_k) \leq y_k \leq y_{\text{max}}(w_k) \)

- **Stochastic discrete disturbance**

\[ w_k \in \{w^1, \ldots, w^s\} \]

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, ...)

"Model Predictive Control" - © 2023 A. Bemporad. All rights reserved.
• Probabilities $p_j$ can be time varying, $p_j(t)$, and have their own dynamics.

Example: Markov chain

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

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

• Discrete distributions can be estimated from historical data (and adapted on-line)
Cost Functions for SMPC to Minimize

- Expected performance

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

- 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

\[
\text{Var}_w [y_k - r_k] = E_w [(y_k - r_k)^2] - (E_w [y_k - r_k])^2
\]
• **Conditional Value-at-Risk (CVaR)** (Rockafellar, Uryasev, 2000)

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

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

= expected shortfall

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

\[
\min_u \sum_{k=0}^{N-1} \max_w |y_k - r_k|
\]
• CVaR optimization
  
  \[
  \min_{u,\alpha} \sum_{k=0}^{N-1} \alpha_k + \frac{1}{1 - \beta} \mathbb{E}_w \left[ \max \{ |y_k - r_k| - \alpha_k, 0 \} \right]
  \]

\[
\min_{u,z,\alpha} \sum_{k=0}^{N-1} \alpha_k + \frac{1}{1 - \beta} \sum_{j=1}^{S} p^j z^j_k \\
\text{s.t.} \quad z^j_k \geq y^j_k - r^j_k - \alpha_k \\
\quad z^j_k \geq r^j_k - y^j_k - \alpha_k \\
\quad z^j_k \geq 0
\]

CVaR optimization becomes a linear programming problem
- Enumerate all possible scenarios \( \{w_j^0, w_j^1, \ldots, w_j^{N-1}\}, j = 1, \ldots, S \)
- Scenario = path on the tree
- Number \( S \) of scenarios = number of leaf nodes
- Each scenario has probability \( p_j = \prod_{k=0}^{N-1} \Pr[w_k = w_j^k] \)
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:

\[
\min \mathbb{E}_w \left[ x_N'Px_N + \sum_{k=0}^{N-1} x_k'Qx_k + u_k'Ru_k \right]
\]

\[
\min \sum_{j=1}^{S} p^j \left( (x_N^j)'Px_N^j + \sum_{k=0}^{N-1} (x_k^j)'Qx_k^j + (u_k^j)'Ru_k^j \right)
\]

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.

- **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.

(Heitsch, Römisch, 2009)
Stochastic control (scenario tree)

Causality constraints: \( u_j^k = u_h^k \) 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 \( w_0, \ldots, w_{k-1} \)

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)
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 $u_0, \ldots, u_{N-1}$ 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
  (Schwarme, Nikolaou, 1999) (Munoz de la Pena, Bemporad, Alamo, 2005) (Primbs, 2007)
  (Bemporad, Di Cairano, 2005) (Bernardini, Bemporad, 2012)

  See also the survey paper (Mesbah, 2016)

- Performance index: \( \min E_w \left[ x_N^T P x_N + \sum_{k=0}^{N-1} x_k^T Q x_k + u_k^T R u_k \right] \)

- Goal: ensure mean-square convergence \( \lim_{t \to \infty} E[x'(t)x(t)] = 0 \) \((f(w(t)) = 0)\)

- Mean-square stability ensured by stochastic Lyapunov function \( V(x) = x' P x \)

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

\[
P = P' > 0
\]

\[
L = L' > 0
\]

(Morozan, 1983)
• 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

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

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

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

• SMPC approach:
  1. Solve LMI problem off-line to find stochastic Lyapunov fcn \( V(x) = x'Q^{-1}x \)
  2. 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 \( u(k) = Kx(k) \)
In condensed form: \( \# \text{opt. vars} = (\# \text{non-leaf nodes}) \times (\# \text{inputs}) \)

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)

435x435 Hessian matrix

sparsity = 0.8%

3240x435 constraint matrix

sparsity = 1.1%
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**

![Graph showing computational time vs. prediction horizon](image)

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.
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, 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)
SMPC FOR REAL-TIME OPTIMAL POWER DISPATCH
• **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 \((i=1,2,3)\)

\[ P_{i,k+1} = P_{i,k} + \Delta P_{i,k} \]

- Constraints on generated power:
  \[ P_{i,\text{min}} \leq P_{i,k} \leq P_{i,\text{max}} \]

- Constraints on power variation:
  \[ \Delta P_{i,\text{min}} \leq \Delta P_{i,k} \leq \Delta P_{i,\text{max}} \]

• Storage model

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

- \( \alpha \) = self discharge loss
- \( \alpha_c \) = charge efficiency
- \( \alpha_d \) = discharge efficiency

- Constraints on charge/discharge:
  \[ S_{\text{min}} \leq S_k \leq S_{\text{max}} \]

- Constraints on charge/discharge rate:
  \[ \Delta S_{\text{min}} \leq S_{k+1} - S_k \leq \Delta S_{\text{max}} \]

- Constraints on power flows:
  \[ 0 \leq u_{c,k} \leq u_{c,\text{max}}, \quad 0 \leq u_{d,k} \leq u_{d,\text{max}} \]
• 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
- Renewable power
- Storage charge/discharge
- Load

- Overall linear model and constraints

\[
\begin{align*}
x & = \begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ S \end{bmatrix} \\
u & = \begin{bmatrix} \Delta P_1 \\ \Delta P_2 \\ \Delta P_3 \\ u_c \\ u_d \\ r - l \end{bmatrix} \\
y & = \begin{bmatrix} P_{\text{ex}} \\ P_1 \\ P_2 \\ P_3 \\ S \\ \Delta S \end{bmatrix}
\end{align*}
\]

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

\[
C = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \alpha - 1 \end{bmatrix} \\
D = \begin{bmatrix} 0 & 0 & 0 & -1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \alpha_c & -\frac{1}{\alpha_d} \end{bmatrix}
\]
Cost function: terms to penalize

\[
\begin{aligned}
\min \quad & \sum_{k=0}^{N-1} \gamma (P_{\text{ex},k} - E_k)^2 + (a_i P_{i,k}^2 + b_i P_{i,k} + c_i) p_k P_{\text{ex},k} \\
\text{s.t.} \quad & Gz \leq W + S \begin{bmatrix} x_0 \\ r-l \\ E \end{bmatrix} \\
\end{aligned}
\]

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

The overall linear MPC problem maps into a QP:
• 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)

• Historical data of price (MW)

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

• Historical data of overall uncertainty
• Data used for scenario generation (31 days):

\[ w^j(t+k) = v^j(t+k) - v^j(t) + v(t) \]

\[ w^j(t) = v(t) \]

Initial value at time t in scenario #j

Value at time t+k 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])

Heuristic Multilevel Clustering

28 nodes
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)
• Simulation results using SMPC, \( M=[2,2,2,1,1] \) (1 day, May 26, 2014)

Total cost = 1,266,099 USD
• Compare simulation results wrt different tree complexity, prescient, and deterministic (1 day, May 26, 2014)

<table>
<thead>
<tr>
<th>Method</th>
<th>Total cost [USD]</th>
<th>M</th>
<th>nvar</th>
<th>CPUTIME [ms]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prescient</td>
<td>1,247,909</td>
<td></td>
<td>30</td>
<td>14</td>
</tr>
<tr>
<td>Stochastic</td>
<td>1,266,099</td>
<td>[2,2,2,1,1]</td>
<td>105</td>
<td>43</td>
</tr>
<tr>
<td>Stochastic</td>
<td>1,266,123</td>
<td>[3,3,1,1,1]</td>
<td>140</td>
<td>50</td>
</tr>
<tr>
<td>Stochastic</td>
<td>1,266,214</td>
<td>[2,2,1,1,1]</td>
<td>95</td>
<td>30</td>
</tr>
<tr>
<td>Stochastic</td>
<td>1,266,701</td>
<td>[3,1,1,1,1]</td>
<td>80</td>
<td>27</td>
</tr>
<tr>
<td>Stochastic</td>
<td>1,267,069</td>
<td>[2,1,1,1,1]</td>
<td>55</td>
<td>22</td>
</tr>
<tr>
<td>Average</td>
<td>1,267,113</td>
<td>[1,1,1,1,1]</td>
<td>30</td>
<td>14</td>
</tr>
<tr>
<td>Frozen-time</td>
<td>1,267,401</td>
<td></td>
<td>30</td>
<td>14</td>
</tr>
</tbody>
</table>

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

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

\[\gamma = 10^3\]

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: 
  \[ S_{\text{min}} \leq S_k \leq S_{\text{max}} \]
  \[ \Delta S_{\text{min}} \leq S_{k+1} - S_k \leq \Delta S_{\text{max}} \]

- \( S_{\text{min}} = 30 \)
- \( S_{\text{max}} = 80 \)
- \( \Delta S_{\text{min}} = -10 \)
- \( \Delta S_{\text{max}} = 40 \) MWh/PTU

- Total cost = 563,283 USD

Revenues from day-ahead market are not counted
SMPC FOR DYNAMIC HEDGING OF 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 $w_1(t), w_2(t), \ldots, w_n(t)$

At the expiration date $T$, the option is worth the **payoff** $p(T) = \text{wealth (€)}$ to be returned to the customer

**How to adjust dynamically the portfolio so that**

$\text{wealth } x(T) = \text{payoff } P(T) \ ? \ ...$

... 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)  
number of assets \# \( i \)  
price of asset \# \( i \) (stochastic process)

Example: \( w_i(t) = \text{log-normal} \) 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**:  

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

\( r = \text{interest rate} \)

\[
b_i(t) \triangleq w_i(t + 1) - (1 + r)w_i(t)
\]
• **Stochastic disturbance:** $w(t) = [w_1(t) \ldots w_n(t)]'$ (=asset prices)

• **Reference signal:** option price $p(t)$. This may depend only on $w(t)$, or also on previous prices $w(0), \ldots, w(t-1)$

• Similarly $p(T)$ may either depend only on $w(T)$ or also on previous prices. For example:

  - **European call**
    \[
    p(T) = \max\{w(T), 0\}
    \]
  
  - **Napoleon cliquet**
    \[
    p(T) = \max\left\{0, C + \min_{i \in \{1, \ldots, N_{\text{fix}}\}} \frac{w(t_i) - w(t_{i-1})}{w(t_{i-1})}\right\}
    \]
    ($t_i$ = fixing dates)
Option Hedging = Linear Stochastic Control

- Block diagram of dynamic option hedging problem:

- Reference signal $p(t) = \text{price of hedged option}$

- 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”)

"Model Predictive Control" - © 2023 A. Bemporad. All rights reserved.
• **Tracking error** = hedging error \( e(t) = x(t) - p(t) \)

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

we want \( e(T) \) small !

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

How about \( E[e(T)] \) ?

\[
\min E[e(T)]^2
\]

\[
\min E[e(T)^2]
\]

\[
\min E[(e(T) - E[e(T)])^2] + \alpha E[e(T)]^2 \quad \text{for } \alpha = 1
\]

Under non-arbitrage conditions, if variance is minimized and the minimum is zero then \( E[e(T)] = 0 \) (Bemporad, Bellucci, Gabbriellini, 2014)

• Recall:

\[
E[e(T)^2] = E[(e(T) - E[e(T)])^2] + \alpha E[e(T)]^2
\]
SMPC FOR DYNAMIC OPTION HEDGING

- Stochastic finite-horizon optimal control problem:

$$\min \{ u(k, z) \} \quad \text{Var}_z [x(t + N, z) - p(t + N, z)]$$

s.t. $$x(k + 1, z) = (1 + r)x(k, z) + \sum_{i=0}^{n} b_i(k, z)u_i(k, z)$$

$$k = t, \ldots, t + N$$

$$x(t, z) = x(t)$$
SMPC FOR DYNAMIC OPTION HEDGING

- Drawback: the longer the horizon $N$, the largest the number of scenarios!

- Special case: $N = 1$

  \[
  \min_{u(t)} \text{Var}_z [x(t + 1, z) - p(t + 1, z)]
  \]

  minimum variance control

- 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!
  ($n = \text{number of assets}$)

"Model Predictive Control" - © 2023 A. Bemporad. All rights reserved.
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 $x^1(t + 1), \ldots, x^M(t + 1)$

- Use a pricing engine to generate the corresponding future option prices $p^1(t + 1), \ldots, p^M(t + 1)$

- Optimize sample variance to get new asset quantities $u(t) \in \mathbb{R}^n$

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

s.t. $x^j(t + 1) = (1 + r)x(t) + \sum_{h=0}^{n} b^j_h(t)u_h(t)$

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

(Bemporad, Puglia, Gabbriellini, 2011) (Graf Plessen, Puglia, Gabbriellini, Bemporad, 2019)
Alternatively, we can compute a covariance matrix before solving the problem

\[
\text{Var}_z [x(t + 1, z) - p(t + 1, z)] = \text{Var}_z \left[ \begin{array}{c}
(1 + r)x(t) + \sum_{i=0}^{n} b_i(t, z)u_i(t) - p(t + 1, z)
\end{array} \right]
\]

\[
= \text{Var}_z \left[ \begin{array}{c}
b(t, z) \\
\frac{1}{p(t + 1, z)}
\end{array} \right] ' \begin{bmatrix} u(t) \\ -1 \end{bmatrix} = \begin{bmatrix} u(t) \\ -1 \end{bmatrix} ' \Sigma_t \begin{bmatrix} u(t) \\ -1 \end{bmatrix}
\]

\[
\Sigma_t = \text{Var}_z \left[ \begin{array}{c}
b(t, z) \\
\frac{1}{p(t + 1, z)}
\end{array} \right] = \begin{bmatrix} \Sigma_t^b & s_t \\ s'_t & \sigma_t^p \end{bmatrix}
\]

Then, assuming \( \Sigma_t \) 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 \( \Sigma_t \) 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 ($\text{log-normal}$)
- 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**

**Portfolio wealth vs. payoff at expiration**

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

SmPC and delta-hedging are almost indistinguishable
<table>
<thead>
<tr>
<th>TIME</th>
<th>x(t)</th>
<th>w(t)</th>
<th>p(t)</th>
<th>u0(t)</th>
<th>x(t)*u1(t)</th>
<th>x(t)*dp/dx(t)</th>
</tr>
</thead>
<tbody>
<tr>
<td>t=0.0000:</td>
<td>S=100.000, P= 6.196, O= 6.196, P(B)=-52.152, P(S)= 58.348 (BS delta= 57.926)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t=0.00185:</td>
<td>S=101.367, P= 6.955, O= 6.865, P(B)=-56.091, P(S)= 38.065 (BS delta= 37.607)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t=0.00370:</td>
<td>S= 96.897, P= 4.134, O= 4.261, P(B)=-42.629, P(S)= 46.762 (BS delta= 46.307)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t=0.00556:</td>
<td>S= 94.582, P= 2.985, O= 3.108, P(B)=-35.080, P(S)= 38.065 (BS delta= 37.607)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t=0.00741:</td>
<td>S= 93.057, P= 2.345, O= 2.415, P(B)=-29.877, P(S)= 32.222 (BS delta= 31.771)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t=0.00926:</td>
<td>S= 93.371, P= 2.431, O= 2.395, P(B)=-30.200, P(S)= 32.632 (BS delta= 32.165)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t=0.01111:</td>
<td>S= 94.295, P= 2.732, O= 2.591, P(B)=-32.518, P(S)= 35.250 (BS delta= 34.760)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t=0.01296:</td>
<td>S= 88.192, P= 0.426, O= 0.859, P(B)=-14.985, P(S)= 15.411 (BS delta= 15.053)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t=0.01481:</td>
<td>S= 90.411, P= 0.803, O= 1.199, P(B)=-19.776, P(S)= 20.579 (BS delta= 20.147)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t=0.01667:</td>
<td>S= 88.586, P= 0.373, O= 0.754, P(B)=-14.236, P(S)= 14.609 (BS delta= 14.234)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t=0.01852:</td>
<td>S= 87.683, P= 0.214, O= 0.544, P(B)=-11.312, P(S)= 11.526 (BS delta= 11.186)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t=0.02037:</td>
<td>S= 90.998, P= 0.641, O= 1.000, P(B)=-18.744, P(S)= 19.385 (BS delta= 18.910)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t=0.02222:</td>
<td>S= 94.742, P= 1.425, O= 1.867, P(B)=-30.734, P(S)= 32.158 (BS delta= 31.555)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t=0.02407:</td>
<td>S= 99.890, P= 3.149, O= 3.945, P(B)=-52.320, P(S)= 55.469 (BS delta= 54.841)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t=0.02593:</td>
<td>S=102.720, P= 4.682, O= 5.466, P(B)=-64.736, P(S)= 69.418 (BS delta= 68.857)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t=0.02778:</td>
<td>S= 99.723, P= 2.609, O= 3.439, P(B)=-51.468, P(S)= 54.077 (BS delta= 53.379)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t=0.02963:</td>
<td>S= 99.591, P= 2.499, O= 3.147, P(B)=-50.513, P(S)= 53.012 (BS delta= 52.268)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t=0.03148:</td>
<td>S= 98.178, P= 1.709, O= 2.233, P(B)=-42.460, P(S)= 44.169 (BS delta= 43.336)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t=0.03333:</td>
<td>S=100.471, P= 2.709, O= 3.142, P(B)=-55.135, P(S)= 57.845 (BS delta= 57.034)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t=0.03519:</td>
<td>S=102.804, P= 4.012, O= 4.363, P(B)=-69.359, P(S)= 73.371 (BS delta= 72.719)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t=0.03704:</td>
<td>S= 97.457, P= 0.144, O= 1.202, P(B)=-34.892, P(S)= 35.037 (BS delta= 33.884)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t=0.03889:</td>
<td>S= 97.789, P= 0.238, O= 1.030, P(B)=-34.692, P(S)= 34.930 (BS delta= 33.564)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t=0.04074:</td>
<td>S= 98.881, P= 0.602, O= 1.089, P(B)=-41.289, P(S)= 41.891 (BS delta= 40.275)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t=0.04259:</td>
<td>S= 97.699, P= 0.071, O= 0.308, P(B)=-22.850, P(S)= 22.921 (BS delta= 20.300)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t=0.4444:</td>
<td>S= 96.002, P= -0.344, O= 0.000, P(B)=-0.344, P(S)= 0.000 (BS delta= 0.000)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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

\[
\begin{align*}
    dw_i(\tau) &= (\mu_i^w d\tau + \sqrt{y_i(\tau)} d z_i^w) w_i(\tau) \\
    dy_i(\tau) &= \theta_i (k_i - y_i(\tau)) d\tau + \omega_i \sqrt{y_i(\tau)} d z_i^y
\end{align*}
\]
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)
• 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>
\[
p(T) = \max \left\{ 0, C + \min_{i \in \{1, \ldots, N_{\text{fix}}\}} \frac{w(t_i) - w(t_{i-1})}{w(t_{i-1})} \right\}
\]
</p>

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

$t_i=0.8, 16, 24$ weeks
• 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

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

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

$t_i = 0, 8, 16, 24$ weeks
• 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 \( t + T \)

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

• CPU time = 1625 ms per SMPC step
  (Matlab R2009 on 1.86GHz Intel Core 2 Duo)
• Bottleneck of the approach for exotic options: price $M$ future option values $p^1(t+1), \ldots, p^M(t+1)$

• 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

\[
p(t) = f(w(t), \sigma(t), w(t_1), \ldots, w(t_{N_{fix}}))
\]

• 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

- 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 is 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

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

Hedging quality is very similar!
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)
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 $v(t) = \begin{bmatrix} v_i^+(t) \\ v_i^-(t) \end{bmatrix} > 0$

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 $v(t)$
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 $e(t + 1)$

$$\min_{v(t), \ell(t), \{z_j(t)\}_{j=1}^M} \ell(t) + \frac{1}{1 - \beta} \sum_{j=1}^M \pi_j z_j(t)$$

subject to

$$z_j(t) \geq \pm (x^j(t + 1) - p^j(t + 1)) - \ell(t) \quad j = 1, \ldots, M$$

$$z(t), v(t) \geq 0$$

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

$$\min_{v(t), \ell(t)} \ell(t)$$

subject to

$$\ell(t) \geq \pm (x^j(t + 1) - p^j(t + 1)) \quad j = 1, \ldots, M$$

$$\ell(t), v(t) \geq 0$$
HEDGING RESULTS

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

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

\[ T_s = \text{trading interval}, \eta_i(t) \sim \mathcal{N}(0, 1), \forall i = 1, \ldots, n \]

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

<table>
<thead>
<tr>
<th>Model</th>
<th>Monte Carlo ( M = 100 )</th>
<th>Monte Carlo ( M = 1000 )</th>
<th>discretized Gaussian ( M = 5 )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( E[e(T)] ) ( E[</td>
<td>e(T)</td>
<td>] )</td>
</tr>
<tr>
<td>QP-Var</td>
<td>-2.30</td>
<td>2.73</td>
<td>-14.91</td>
</tr>
<tr>
<td>LP-CVaR</td>
<td>-1.34</td>
<td>2.58</td>
<td>-7.55</td>
</tr>
<tr>
<td>LP-MinMax</td>
<td>-2.67</td>
<td>3.83</td>
<td>-12.13</td>
</tr>
<tr>
<td>Delta Hedging</td>
<td>-0.1312</td>
<td>1.77</td>
<td>-5.4</td>
</tr>
</tbody>
</table>

European call option

<table>
<thead>
<tr>
<th>Model</th>
<th>LS ( M = 100 )</th>
<th>LS ( M = 5 )</th>
<th>LS ( M = 1000 )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( E[e(T)] ) ( E[</td>
<td>e(T)</td>
<td>] ) ( \min(e(T)) ) ( \text{Var}(e(T)) ) CPU(s)</td>
</tr>
<tr>
<td>QP-Var</td>
<td>-2.19</td>
<td>6.85</td>
<td>-42.76</td>
</tr>
<tr>
<td>LP-CVaR</td>
<td>-0.72</td>
<td>1.29</td>
<td>-12.16</td>
</tr>
<tr>
<td>LP-MinMax</td>
<td>-0.72</td>
<td>1.29</td>
<td>-12.16</td>
</tr>
<tr>
<td>Delta Hedging</td>
<td>-0.70</td>
<td>1.79</td>
<td>-16.14</td>
</tr>
</tbody>
</table>

barrier option
SMPC FOR AUTOMOTIVE CONTROL
Control problem: 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?

\[ P_{req}(w(t)) = \text{driver's power request} \]

\[ P_{req}(k) = P_{el}(k) + P_{mec}(k) - P_{br}(k) \]
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:** $w(k) = \text{Markov chain}$

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

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

\[ P_{req}(k) = P_{el}(k) + P_{mec}(k) - P_{br}(k) \]

State-space equations

\[ SoC(k + 1) = SoC(k) - KT_s P_{el}(k) \]

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

Constraints

\[ SoC_{min} \leq SoC(k) \leq SoC_{max} \]

\[ 0 \leq P_{mec}(k - 1) \leq P_{mec,max} \]

\[ P_{el,min} \leq P_{el}(k) \leq P_{el,max} \]

\[ \Delta P_{min} \leq \Delta P \leq \Delta P_{max} \]

\[ 0 \leq P_{br}(k) \]

sample time \( T_s = 1 \text{ s} \)
“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
Results obtained on New European Driving Cycle (NEDC)
Comparison on different driving cycles

<table>
<thead>
<tr>
<th>SHEV Energy Management Simulation Results on Standard Driving Cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td>-------------------------------</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>NEDC</td>
</tr>
<tr>
<td>FTMPC</td>
</tr>
<tr>
<td>→ SMPCL</td>
</tr>
<tr>
<td>PMPC</td>
</tr>
<tr>
<td>FTP-75</td>
</tr>
<tr>
<td>FTMPC</td>
</tr>
<tr>
<td>→ SMPCL</td>
</tr>
<tr>
<td>PMPC</td>
</tr>
<tr>
<td>FTP-Highway</td>
</tr>
<tr>
<td>FTMPC</td>
</tr>
<tr>
<td>→ SMPCL</td>
</tr>
<tr>
<td>PMPC</td>
</tr>
</tbody>
</table>

pretty close to having the crystal ball.
But we don’t, we just model uncertainty carefully.
Comparison on different driving cycles - Real driving data

### TABLE II
Simulation Results on Real-World Driving Cycles

<table>
<thead>
<tr>
<th>Trace #1 - smooth accelerations</th>
<th>FTMPC 37.84kW</th>
<th>SMPCL 14.32kW</th>
<th>PMPC 14.08kW</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fuel cons.</td>
<td>243g</td>
<td>244g</td>
<td>223g</td>
</tr>
<tr>
<td>$\Delta P$</td>
<td>80.61kW</td>
<td>327g</td>
<td>30.67kW</td>
</tr>
<tr>
<td>$S_oC$ gain/loss</td>
<td>0.11%</td>
<td>1.16%</td>
<td>0.17%</td>
</tr>
<tr>
<td>Equiv. fuel cons.</td>
<td>323g</td>
<td>287g</td>
<td>282g</td>
</tr>
<tr>
<td>impr. wrt FTMPC</td>
<td>-</td>
<td>11.34%</td>
<td>12.73%</td>
</tr>
</tbody>
</table>

### Trace #2 - steep accelerations

<table>
<thead>
<tr>
<th>Trace #2 - steep accelerations</th>
<th>FTMPC 80.61kW</th>
<th>SMPCL 35.74kW</th>
<th>PMPC 30.67kW</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fuel cons.</td>
<td>327g</td>
<td>320g</td>
<td>287g</td>
</tr>
<tr>
<td>$\Delta P$</td>
<td>30.67kW</td>
<td>287g</td>
<td>30.67kW</td>
</tr>
<tr>
<td>$S_oC$ gain/loss</td>
<td>0.11%</td>
<td>1.16%</td>
<td>0.17%</td>
</tr>
<tr>
<td>Equiv. fuel cons.</td>
<td>323g</td>
<td>287g</td>
<td>282g</td>
</tr>
<tr>
<td>impr. wrt FTMPC</td>
<td>-</td>
<td>11.34%</td>
<td>12.73%</td>
</tr>
</tbody>
</table>

### TABLE III
Percentage Improvement of SMPCL Strategy Due to Online Learning of the Markov Chain

<table>
<thead>
<tr>
<th>Standard cycle</th>
<th>Learning Improvement</th>
<th>Real-word Driving</th>
<th>Learning Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>NEDC</td>
<td>12.7%</td>
<td>Trace #1</td>
<td>1.3%</td>
</tr>
<tr>
<td>FTP-75</td>
<td>16.5%</td>
<td>Trace #2</td>
<td>13.4%</td>
</tr>
<tr>
<td>FTP-H.</td>
<td>1.1%</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Problem setup

Follower (equipped with ACC)  Leader vehicle

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**

\[
al(k) \quad \text{acceleration} \\
v(k) \quad \text{velocity} \\
d(k) \quad \text{distance} \\
v_l(k) \quad \text{leader velocity}
\]

**Inputs**

\[
u(k) \quad \text{jerk} \\
a_l(k) \quad \text{leader acceleration}
\]

**Dynamical Model**

\[
a(k + 1) = a(k) + T_s u(k) \\
v(k + 1) = v(k) + T_s a(k) \\
v_l(k + 1) = v_l(k) + T_s a_l(k)
\]

**Uncertainty**

Stochastic leader acceleration

\[a_l(k) = w(k)\]

**Constraints**

\[
d(k) \geq d_{\text{min}}(k) = \delta + \gamma v(k) \quad \text{safety} \\
u_{\text{min}} \leq u \leq u_{\text{max}} \quad \text{comfort}
\]

**References**

\[
d_{\text{ref}}(k) = \delta_{\text{ref}} + \gamma_{\text{ref}} v(k) \\
v_{\text{ref}} = 26 \text{m/s}
\]
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
**General overview:**

- Municipalities supplied: 23
- Supply area: 424 km²
- Population supplied: 2,922,773
- Average demand: 7 m³/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"
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
The control objectives are translated into cost functions:

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

\[
J^{ws} = W_\alpha^2 \sum_{l=1}^{K} \sum_{i=0}^{H_p-1} p^l (\alpha_1 + \alpha_2,k)^2 (u_{k+i|k})^2 \quad €
\]

Expected total smooth operation cost (ETSOC)

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

Expected total safety storage cost (ETSSSC)

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

Need to minimize the total operating cost

\[
V = J^{ws} + J^\Delta + J^S
\]
CONTROL OF THE DRINKING WATER NETWORK OF BCN

Prediction of water demand in Barcelona

Uncertainty represented as a fan of scenarios

Reduction to a scenario tree

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

Uncertainty: demand prediction error

"Model Predictive Control" - © 2023 A. Bemporad. All rights reserved.
**Economic:** Avoid pumping when the price of electricity is high

**Foresight:** tanks start loading up before the consumers ask for water
**SMPC:** The network operator has online information about the current and predicted operating cost in real time.
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 (others' decisions = measured disturbances) (Bemporad, Barcelli, 2010)
  - **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)