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

✔ MPC computations: quadratic programming (QP), explicit MPC

✔ Hybrid MPC

• Stochastic MPC

• Data-driven MPC

Course page:
http://cse.lab.imtlucca.it/~bemporad/mpc_course.html
STOCHASTIC MODEL PREDICTIVE CONTROL
• 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** 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.
• **At time $t$:** solve a **stochastic optimal control** problem over a finite future horizon of $N$ steps:

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

• 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 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 \( y_{\min}(w_k) \leq y_k \leq y_{\max}(w_k) \)

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

Example: Markov chain

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

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

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

- **Expected performance**

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

- **Tradeoff between expectation & risk**

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

\[\alpha \geq 0\]

- **Note that they coincide for \( \alpha = 1 \), since**

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

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

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

= expected shortfall

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

\[
\min_u \sum_{k=0}^{N-1} \max_w |y_k - r_k|
\]
• Enumerate all possible scenarios \( \{w_0^j, w_1^j, \ldots, w_{N-1}^j\}, \ j = 1, \ldots, S \)

• Scenario = path on the tree

• Number \( S \) of scenarios = number of leaf nodes

• 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

\[ x^j_{k+1} = A(w^j_k)x^j_k + B(w^j_k)u^j_k + f(w^j_k) \]

(=linear time-varying system)

- Expectations become simple sums!

Example:

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

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

Expectations of quadratic costs remain quadratic costs
STOCHASTIC OPTIMAL CONTROL PROBLEM

• CVaR optimization  (Rockafellar, Uryasev, 2000)

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

\[
\min_{u,z,\alpha} \sum_{k=0}^{N-1} \alpha_k + \frac{1}{1 - \beta} \sum_{j=1}^{S} p^j z^j_k
\]

s.t.
\[
\begin{align*}
    z^j_k & \geq y^j_k - r^j_k - \alpha_k \\
    z^j_k & \geq r^j_k - y^j_k - \alpha_k \\
    z^j_k & \geq 0
\end{align*}
\]

CVaR optimization becomes a linear programming problem
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.

Heuristic Multilevel Clustering (Heitsch, Römisch, 2009)
**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, \( u_k^0 \equiv u_0 \) 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)
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 recent survey (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

$$\lim_{t \to \infty} E[x'(t)x(t)] = 0 \quad (f(w(t)) = 0)$$

Mean-square stability ensured by stochastic Lyapunov function

$$V(x) = x' P x$$

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

$$P = P' \succ 0$$

$$L = L' \succ 0$$

(Morozan, 1983)
The approach can be generalized to uncertain probabilities $p(t) \in \mathcal{P}$ (Example: time-varying probabilities)

- If $\mathcal{P} \equiv \mathcal{P}$ 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

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

\[
\min_u \mathbb{E}_w \left[ \sum_{k=0}^{N-1} \ell(x_k, u_k) \right]
\]

\[
\text{s.t. } x_{k+1} = A(w_k)x_k + B(w_k)u_k \\
\mathbb{E} [V(A(w_0)x_0 + B(w_0)u_0)] \leq x'_0(Q^{-1} - L)x_0 \\
x_0 = x(t)
\]

• 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 $u(k) = Kx(k)$
Complexity of Stochastic Optimization Problem

- #optimization variables = #nodes x #inputs (in condensed version)

- Problems are very sparse (well exploited by interior point methods)

- Example: SMPC with quadratic cost and linear constraints
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)

- GPAD: $\epsilon_g = 0.005$
- GPAD: $\epsilon_g = 0.01$
- Gurobi (IP)

**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.

"Model Predictive Control" - © A. Bemporad. All rights reserved.
DISTRIBUTED GPAD FOR STOCHASTIC MPC

(Sampathirao, Sopasakis, Bemporad, 2015)

CPU time (s)

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
**Objective:** maximize BRP’s profit!

while satisfying local power demand and all grid constraints

*AGC = Automatic Generation Control*
Balance Responsible Parties (BRPs) participate to the various energy markets and trade electricity to satisfy their loads and make profits.

Optimal control of BRPs is challenging, as in real-time a BRP must:

- fulfill its E-Program, despite perturbations induced by uncertainties
- intermittent generation from renewable sources
- time-varying internal loads
- react to signals arriving from the TSO
- frequency deviations, AS bids activated by the TSO
- minimize generation and imbalance costs
- time-varying, stochastic imbalance prices
- consider plant dynamics and satisfy constraints
- bounds on power output, ramp-rate constraints
• We are a legal entity (BRP) trading on the energy (PX) and ancillary service (AS) markets

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

• **Constraints:** Grid capacity, rate limits, load balancing, AS balancing
- Microgrid with three conventional power generators (P1, P2, P3), two renewables (R1, R2), one storage system (S1), satisfying local load and exchanging power with the grid.
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

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

\[\alpha = \text{self discharge loss}\]
\[\alpha_c = \text{charge efficiency}\]
\[\alpha_d = \text{discharge efficiency}\]

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 DISPATCH MODEL

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

\[ P_{\text{ex},k} = P_{1,k} + P_{2,k} + P_{3,k} - u_c,k + u_d,k + (r_k - l_k) \]

\[ x = \begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ S \end{bmatrix}, \quad u = \begin{bmatrix} \Delta P_1 \\ \Delta P_2 \\ \Delta P_3 \\ u_c \\ u_d \end{bmatrix}, \quad y = \begin{bmatrix} P_{\text{ex}} \\ P_1 \\ P_2 \\ P_3 \\ S \end{bmatrix} \]

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

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

"Model Predictive Control" - © A. Bemporad. All rights reserved.
Power dispatch cost function

- Cost function: terms to penalize

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

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

- The overall linear MPC problem maps into a QP:

\[
\begin{align*}
\min_z & \quad \frac{1}{2} z' H z + \left( \frac{F}{r-l} \right) c z + d \\
\text{s.t.} & \quad G z \leq W + S \left[ \begin{array}{c} x_0 \\ \frac{r-l}{E} \end{array} \right]
\end{align*}
\]

- \( x_0 \) = current state
- \( r - l \) = predicted renewable power - load
- \( E \) = E-program

\[
z = \{ \Delta P_{i,k}, u_{c,k}, u_{d,k} \}_{k=0}^{N-1}
\]
• 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):

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

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

- 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 \( M=[2 2 2 1 1] \))

Heuristic Multilevel Clustering

28 nodes
• MPC setup:

  – Sampling time: $T_s=5$ min

  – Prediction horizon: $N=6$ steps (=1/2 hour ahead)

  – Three controller options:

    • **Stochastic MPC**, with branching factor $M$ (ex: $M=[4 \ 3 \ 2 \ 2 \ 1]$)

    • **Average MPC**, that is deterministic MPC based on the **expected** (price, load-renewable) realization

    • **Prescient MPC**, that is deterministic MPC based on the **exact** future (price, load-renewable) realization
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>CPU TIME [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 (1)</td>
<td>1,266,099</td>
<td>[2,2,2,1,1]</td>
<td>105</td>
<td>43</td>
</tr>
<tr>
<td>Stochastic (2)</td>
<td>1,266,123</td>
<td>[3,3,1,1,1]</td>
<td>140</td>
<td>50</td>
</tr>
<tr>
<td>Stochastic (3)</td>
<td>1,266,214</td>
<td>[2,2,1,1,1]</td>
<td>95</td>
<td>30</td>
</tr>
<tr>
<td>Stochastic (4)</td>
<td>1,266,701</td>
<td>[3,1,1,1,1]</td>
<td>80</td>
<td>27</td>
</tr>
<tr>
<td>Stochastic (5)</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>

- exact knowledge of future uncertainty
- stochastic formulation
- deterministic: assume future disturbance = average of historical data
- deterministic: assume future disturbance = current disturbance

nvar = number of variables in QP problem = 5* (# nodes), CPU TIME = time to build tree, build QP matrices, and solve
SMPC FOR MARKET-BASED OPTIMAL POWER DISPATCH

• Tracking an E-Program

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

\[
\gamma = 10^3
\]

SMPC with branching factor \( M = [2 2 2 1 1] \)

Revenues from day-ahead market are not counted

total cost = 574,388 USD
SMPC FOR MARKET-BASED OPTIMAL POWER DISPATCH

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

- \( S_{\text{min}} = 30 \)
- \( S_{\text{max}} = 80 \)
- \( \Delta S_{\text{min}} = -10 \)
- \( \Delta S_{\text{max}} = 10 \)

- Power exchanged with the real-time market
- State of charge of energy storage
- Conventional power generation

- Energy price
- Total cost per period
- Production from renewables - load

- 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 $x(0)$ (€).

Such money $x(0)$ is used to create a portfolio $x(t)$ of $n$ underlying assets (e.g., stocks) whose prices at time $t$ are $w_1(t), w_2(t), \ldots, w_n(t)$.

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

How to adjust dynamically the portfolio so that wealth $x(T) = \text{payoff } r(T)$? ...

.. for any price realization $w_i(t)$?
**Portfolio Dynamics**

- Portfolio wealth at time $t$:

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

- Money in bank account (risk-free asset)
- 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**:

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

$r = \text{interest rate}$

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

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

- Block diagram of dynamic option hedging problem:

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

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

we want $e(T)$ small!

- $\min E[e(T)]^2$
- $\min E[e(T)^2]$ 
- $\min E[(e(T) - E[e(T)])^2] + \alpha E[e(T)]^2$ for $\alpha = 1$

• **Recall**: $E[e(T)^2] = E[(e(T) - E[e(T)])^2] + \alpha E[e(T)]^2$

• 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:

\[
\min_{\{u(k, z)\}} \text{Var}_z [x(t + N, z) - r(t + N, z)]
\]

s.t. \[x(k + 1, z) = (1 + r)x(k, z) + \sum_{i=0}^{n} b_i(k, z)u_i(k, z), \quad k = t, \ldots, t + N\]

\[x(t, z) = x(t)\]
• Drawback: the longer the horizon $N$, the largest the number of scenarios!

• Special case: use $N=1$!

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

✓ Only one vector $u(t)$ to optimize

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

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

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

$n = \text{number of traded assets}$

Perfect hedging assumption from time $t+1$ to $T$

Optimize up to time $t+1$
SMPC HEDGING ALGORITHM

• Let $t =$ current hedging date, $w(t) =$ wealth of portfolio, $x(t) \in \mathbb{R}^n =$ asset prices

• Use **Monte Carlo simulation** to generate $M$ scenarios of future asset prices
  
  $x^1(t + 1), x^2(t + 1), \ldots, x^M(t + 1)$

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

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

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

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

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

  **Least squares problem**

  $(n = \text{number of traded assets})$
EXAMPLE: BS MODEL, EUROPEAN CALL

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

- CPU time = 7.52 ms per SMPC step
  (Matlab R2009 on 1.86GHz Intel Core 2 Duo)
**EXAMPLE: BS MODEL, EUROPEAN CALL**

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

**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
Example: BS model, European call

<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=0.0185:</td>
<td>S=101.367, P= 6.955, O= 6.865, P(B)= -56.091, P(S)= 39.316 (BS delta= 62.628)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t=0.0370:</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=0.0556:</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=0.0741:</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=0.0926:</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=0.1111:</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=0.1296:</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=0.1481:</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=0.1667:</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=0.1852:</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=0.2037:</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=0.2222:</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=0.2407:</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=0.2593:</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=0.2778:</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=0.2963:</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=0.3148:</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=0.3333:</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=0.3519:</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=0.3704:</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=0.3889:</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=0.4074:</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=0.4259:</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>
Example: Heston Model, European Call

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

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

Heston's model

\[
\begin{align*}
    dx_i(\tau) &= \left(\mu_{i}^x d\tau + \sqrt{y_i(\tau)} dz_i^x\right) x_i(\tau) \\
    dy_i(\tau) &= \theta_i(k_i - y_i(\tau)) d\tau + \omega_i \sqrt{y_i(\tau)} dz_i^y
\end{align*}
\]
**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)

**Portfolio wealth vs. payoff at expiration**

**Hedging error** $e(T) = w(T) - p(T)$

"Model Predictive Control" - © A. Bemporad. All rights reserved.
EXAMPLE: BS MODEL, NAPOLEON CLIQUET

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

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

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

\( t_i = 0, 8, 16, 24 \) weeks
• Black-Scholes model (log-normal)
• volatility=0.2
• $T=24$ weeks (hedging every week)
• 50 simulations
• $M=100$ scenarios
• risk-free=0.04
• Delta hedging, only trade underlying stock

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

• CPU time = 2.41 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

Portion of portfolio wealth vs. payoff at expiration

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

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

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

"Model Predictive Control" - © A. Bemporad. All rights reserved.
**Approximate Option Pricing**

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

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

- **Monte Carlo pricing** can be time consuming: say $L$ scenarios to evaluate a single option value $\Rightarrow$ need to simulate $ML$ paths to build optimization problem

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

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

  Example: Napoleon cliquet, Heston model

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

- Most suitable method for estimating pricing function $f$:

  **least-squares Monte Carlo** approach based on polynomial approximations

  (Longstaff, Schwartz, 2001)
EXAMPLE: BS MODEL, NAPOLEON CLIQUET

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

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

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

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

Hedging quality 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
  (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)
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?

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

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

Series hybrid

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

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

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)
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

\[ P_{\text{req}}(w(t+k|k)) = P_{\text{req}}(w(k)) \]

“Prescient” MPC (PMPC)

Future disturbance sequence \( P_{\text{req}}(w(t+k|k)) \)
known in advance
Results obtained on New European Driving Cycle (NEDC)
Comparison on different driving cycles

### SHEV Energy Management Simulation Results on Standard Driving Cycles

|                  | \(|\|\Delta P\|\) | Fuel cons. | \(\Delta SoC\) gain/loss | Equivalent 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.
Comparison on different driving cycles - Real driving data

TABLE II
SIMULATION RESULTS ON REAL-WORLD DRIVING CYCLES

<table>
<thead>
<tr>
<th></th>
<th>$|\Delta P|$</th>
<th>Fuel cons.</th>
<th>$SoC$ gain/loss</th>
<th>Equiv. fuel cons.</th>
<th>impr. wrt FTMPCE</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Trace #1 - smooth accelerations</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FTMPC</td>
<td>37.84kW</td>
<td>243g</td>
<td>-0.05%</td>
<td>244g</td>
<td>–</td>
</tr>
<tr>
<td>→ SMPCL</td>
<td>14.32kW</td>
<td>244g</td>
<td>0.90%</td>
<td>225g</td>
<td>8.04%</td>
</tr>
<tr>
<td>PMPC</td>
<td>14.08kW</td>
<td>223g</td>
<td>-0.08%</td>
<td>224g</td>
<td>8.19%</td>
</tr>
<tr>
<td><strong>Trace #2 - steep accelerations</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FTMPC</td>
<td>80.61kW</td>
<td>327g</td>
<td>0.11%</td>
<td>323g</td>
<td>–</td>
</tr>
<tr>
<td>→ SMPCL</td>
<td>35.74kW</td>
<td>320g</td>
<td>1.16%</td>
<td>287g</td>
<td>11.34%</td>
</tr>
<tr>
<td>PMPC</td>
<td>30.67kW</td>
<td>287g</td>
<td>0.17%</td>
<td>282g</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
**States**

- $a(k)$: acceleration
- $v(k)$: velocity
- $d(k)$: distance
- $v_l(k)$: leader velocity

**Dynamical Model**

\[
\begin{align*}
a(k+1) &= a(k) + T_s u(k) \\
v(k+1) &= v(k) + T_s a(k) \\
v_l(k+1) &= v_l(k) + T_s a_l(k)
\end{align*}
\]

**Inputs**

- $u(k)$: jerk
- $a_l(k)$: leader acceleration

**Uncertainty**

Stochastic leader acceleration

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

**Constraints**

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

**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

Stochastic MPC (blue solid line)
Frozen Time MPC (red dashed line)
Prescient MPC (black dashed line)

Simulation results on European Urban Driving Cycle (EUDC)
**DRINKING WATER NETWORK OF BARCELONA (SPAIN)**

**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

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

Expected total smooth operation cost (ETSOC)

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

Expected total safety storage cost (ETSSC)

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

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
**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** (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
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" - © A. Bemporad. All rights reserved.