MODEL PREDICTIVE CONTROL

LINEAR TIME-VARYING AND NONLINEAR MPC

Alberto Bemporad

`http://cse.lab.imtlucca.it/~bemporad`

COURSE STRUCTURE

- ✔ Linear model predictive control (MPC)
- Linear time-varying and nonlinear MPC
- MPC computations: quadratic programming (QP), explicit MPC
- Hybrid MPC
- Stochastic MPC
- Data-driven MPC

MATLAB Toolboxes:

- **MPC Toolbox** (linear/explicit/parameter-varying MPC)
- **Hybrid Toolbox** (explicit MPC, hybrid systems)

Course page:

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

LINEAR TIME-VARYING MODEL PREDICTIVE CONTROL

LPV MODELS

- **Linear Parameter-Varying (LPV)** model

$$\begin{cases} x_{k+1} = A(p(t))x_k + B(p(t))u_k + B_v(p(t))v_k \\ y_k = C(p(t))x_k + D_v(p(t))v_k \end{cases}$$

  that depends on a vector $p(t)$ of parameters

- The weights in the quadratic performance index can also be LPV

- The resulting optimization problem is still a QP

$$\begin{array}{ll} \min\limits_z & \frac{1}{2}z'H(p(t))z + \begin{bmatrix} x(t) \\ r(t) \\ u(t-1) \end{bmatrix}' F(p(t))'z \\ \text{s.t.} & G(p(t))z \leq W(p(t)) + S(p(t)) \begin{bmatrix} x(t) \\ r(t) \\ u(t-1) \end{bmatrix} \end{array}$$

- The QP matrices must be constructed online, contrarily to the LTI case

LINEARIZING A NONLINEAR MODEL: LPV CASE

- An LPV model can be obtained by linearizing the **nonlinear model**

$$\begin{cases} \dot{x}_c(t) & = & f(x_c(t), u_c(t), p_c(t)) \\ y_c(t) & = & g(x_c(t), p_c(t)) \end{cases}$$

- $p_c \in \mathbb{R}^{n_p}$ = a vector of exogenous signals (e.g., ambient conditions)
- At time t, consider **nominal values** $\bar{x}_c(t)$, $\bar{u}_c(t)$, $\bar{p}_c(t)$ and linearize

$$\frac{d}{d\tau}(x_c(t+\tau) - \bar{x}_c(t)) = \frac{d}{d\tau}(x_c(t+\tau)) \simeq \underbrace{\left.\frac{\partial f}{\partial x}\right|_{\bar{x}_c(t), \bar{u}_c(t), \bar{p}_c(t)}}_{A_c(t)} (x_c(t+\tau) - \bar{x}_c(t)) +$$

$$\underbrace{\left.\frac{\partial f}{\partial u}\right|_{\bar{x}_c(t), \bar{u}_c(t), \bar{p}_c(t)}}_{B_c(t)} (u_c(t+\tau) - \bar{u}_c(t)) + \underbrace{f(\bar{x}_c(t), \bar{u}_c(t), \bar{p}_c(t))}_{B_{vc}(t)} \cdot 1$$

- Convert $(A_c, [B_c \; B_{vc}])$ to discrete-time and get prediction model $(A, [B \; B_v])$
- Same thing for the output equation to get matrices C and D_v

LTV MODELS

- **Linear Time-Varying (LTV)** model

$$\begin{cases} x_{k+1} & = & A_k(t)x_k + B_k(t)u_k \\ y_k & = & C_k(t)x_k \end{cases}$$

- At each time t the model can also change over the prediction horizon k
- The measured disturbance is embedded in the model
- The resulting optimization problem is still a QP

$$\begin{array}{ll} \min\limits_{z} & \frac{1}{2}z'H(t)z + \begin{bmatrix} x(t) \\ r(t) \\ u(t-1) \end{bmatrix}' F(t)'z \\ \text{s.t.} & G(t)z \leq W(t) + S(t) \begin{bmatrix} x(t) \\ r(t) \\ u(t-1) \end{bmatrix} \end{array}$$

- As for LPV-MPC, the QP matrices must be constructed online

LINEARIZING A NONLINEAR MODEL: LTV CASE

- LPV/LTV models can be obtained by linearizing **nonlinear models**

$$\begin{cases} \dot{x}_c(t) & = & f(x_c(t), u_c(t), p_c(t)) \\ y_c(t) & = & g(x_c(t), p_c(t)) \end{cases}$$

- At time t, consider **nominal trajectories**

$$U = \{\bar{u}_c(t), \bar{u}_c(t+T_s), \ldots, \bar{u}_c(t+(N-1)T_s)\}$$

(example: U = shifted previous optimal sequence or input ref. trajectory)

$$P = \{\bar{p}_c(t), \bar{p}_c(t+T_s), \ldots, \bar{p}_c(t+(N-1)T_s)\}$$

(no preview: $\bar{p}_c(t+k) \equiv \bar{p}_c(t)$)

- **Integrate** the model and get nominal state/output trajectories

$$\begin{array}{rcl} X & = & \{\bar{x}_c(t), \bar{x}_c(t+T_s), \ldots, \bar{x}_c(t+(N-1)T_s)\} \\ Y & = & \{\bar{y}_c(t), \bar{y}_c(t+T_s), \ldots, \bar{y}_c(t+(N-1)T_s)\} \end{array}$$

- Examples: $\bar{x}_c(t) =$ current $x_c(t)$; $\bar{x}_c(t) =$ equilibrium; $\bar{x}_c(t) =$ reference

LINEARIZING A NONLINEAR MODEL: LTV CASE

- While integrating, also compute the **sensitivities**

$$
\begin{aligned}
A_k(t) &= \frac{\partial \bar{x}_c(t+(k+1)T_s)}{\bar{x}_c(t+kT_s)} \\
B_k(t) &= \frac{\partial \bar{x}_c(t+(k+1)T_s)}{\bar{u}_c(t+kT_s)} \\
C_k(t) &= \frac{\partial \bar{y}_c(t+kT_s)}{\bar{x}_c(t+kT_s)}
\end{aligned}
$$

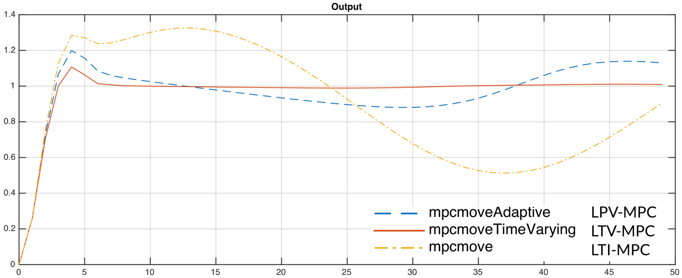
- Approximate the NL model as the LTV model

$$
\left\{
\begin{aligned}
\overbrace{x_c(k+1)-\bar{x}_c(k+1)}^{x_{k+1}} &= A_k(t)\overbrace{(x_c(k)-\bar{x}_c(k))}^{x_k}+B_k(t)\overbrace{(u_c(k)-\bar{u}_c(k))}^{u_k} \\
\underbrace{y_c(k)-\bar{y}_c(k)}_{y_k} &= C_k(t)\underbrace{(x_c(k)-\bar{x}_c(k))}_{x_k}
\end{aligned}
\right.
$$

LTV-MPC EXAMPLE

- Process model is LTV

$$\frac{d^3y}{dt^3} + 3\frac{d^2y}{dt^2} + 2\frac{dy}{dt} + (6+\sin(5t))y = 5\frac{du}{dt} + \left(5 + 2\cos\left(\frac{5}{2}t\right)\right)u$$



- LTI-MPC cannot track the setpoint, LPV-MPC tries to catch-up with time-varying model, LTV-MPC has preview on future model values

(See demo TimeVaryingMPCControlOfATimeVaryingLinearSystemExample in MPC Toolbox)

LTV-MPC EXAMPLE

- Define LTV model

```
Models = tf; ct = 1;
for t = 0:0.1:10
    Models(:,:,ct) = tf([5 5+2*cos(2.5*t)],[1 3 2 6+sin(5*t)]);
    ct = ct + 1;
end

Ts = 0.1; % sampling time
Models = ss(c2d(Models,Ts));
```

- Design MPC controller

```
sys = ss(c2d(tf([5 5],[1 3 2 6]),Ts)); % average model time
p = 3; % prediction horizon
m = 3; % control horizon
mpcobj = mpc(sys,Ts,p,m);

mpcobj.MV = struct('Min',-2,'Max',2); % input constraints
mpcobj.Weights = struct('MV',0,'MVRate',0.01,'Output',1);
```

LTV-MPC EXAMPLE

- Simulate LTV system with **LTI-MPC** controller

```
for ct = 1:(Tstop/Ts+1)
    real_plant = Models(:,:,ct); % Get the current plant
    y = real_plant.C*x;
    u = mpcmove(mpcobj,xmpc,y,1); % Apply LTI MPC
    x = real_plant.A*x + real_plant.B*u;
end
```

- Simulate LTV system with **LPV-MPC** controller

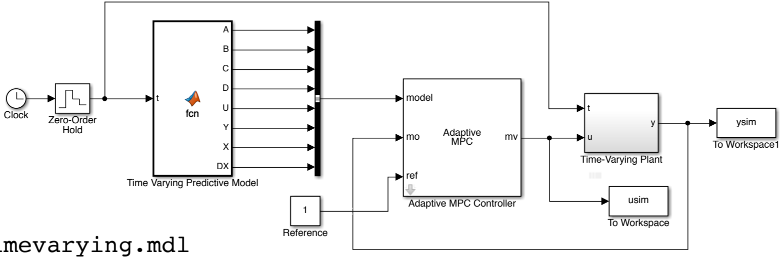
```
for ct = 1:(Tstop/Ts+1)
    real_plant = Models(:,:,ct); % Get the current plant
    y = real_plant.C*x;
    u = mpcmoveAdaptive(mpcobj,xmpc,real_plant,nominal,y,1);
    x = real_plant.A*x + real_plant.B*u;
end
```

LTV-MPC EXAMPLE

- Simulate LTV system with **LTV-MPC** controller

```
for ct = 1:(Tstop/Ts+1)
    real_plant = Models(:,:,ct); % Get the current plant
    y = real_plant.C*x;
    u = mpcmoveAdaptive(mpcobj,xmpc,Models(:,:,ct:ct+p),Nominals,y,1);
    x = real_plant.A*x + real_plant.B*u;
end
```

- Simulate in Simulink

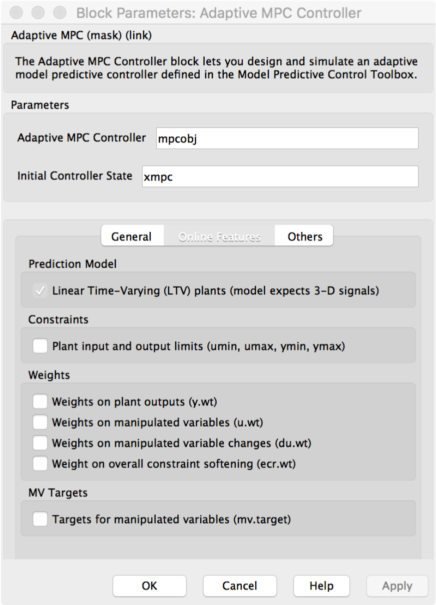


`mpc_timevarying.mdl`

LTV-MPC EXAMPLE

- Simulink block

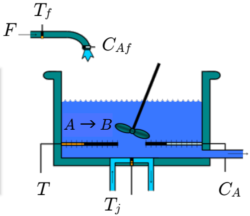
need to provide 3D array
of future models



`mpc_timevarying.mdl`

EXAMPLE: LPV-MPC OF A NONLINEAR CSTR SYSTEM

- MPC control of a diabatic **continuous stirred tank reactor (CSTR)**
- Process model is nonlinear



$$
\begin{aligned}
\frac{dC_A}{dt} &= \frac{F}{V}(C_{Af} - C_A) - C_A k_0 e^{-\frac{\Delta E}{RT}} \\
\frac{dT}{dt} &= \frac{F}{V}(T_f - T) + \frac{UA}{\rho C_p V}(T_j - T) - \frac{\Delta H}{\rho C_p} C_A k_0 e^{-\frac{\Delta E}{RT}}
\end{aligned}
$$

  - T : temperature inside the reactor $[K]$ (state)
  - C_A : concentration of the reactant in the reactor $[kgmol/m^3]$ (state)
  - T_j : jacket temperature $[K]$ (input)
  - T_f : feedstream temperature $[K]$ (measured disturbance)
  - C_{Af} : feedstream concentration $[kgmol/m^3]$ (measured disturbance)

- Objective: **manipulate** T_j to **regulate** C_A on desired setpoint

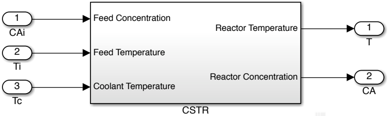
```
>> edit ampccstr_linearization
```

(MPC Toolbox)

EXAMPLE: LPV-MPC OF A NONLINEAR CSTR SYSTEM

Process model:

```
>> mpc_cstr_plant
```




```
% Create operating point specification.
plant_mdl = 'mpc_cstr_plant';
op = operspec(plant_mdl);

op.Inputs(1).u = 10; % Feed concentration known @initial condition
op.Inputs(1).Known = true;
op.Inputs(2).u = 298.15; % Feed concentration known @initial condition
op.Inputs(2).Known = true;
op.Inputs(3).u = 298.15; % Coolant temperature known @initial condition
op.Inputs(3).Known = true;

[op_point, op_report] = findop(plant_mdl,op); % Compute initial condition

x0 = [op_report.States(1).x;op_report.States(2).x];
y0 = [op_report.Outputs(1).y;op_report.Outputs(2).y];
u0 = [op_report.Inputs(1).u;op_report.Inputs(2).u;op_report.Inputs(3).u];

% Obtain linear plant model at the initial condition.
sys = linearize(plant_mdl, op_point);
sys = sys(:,2:3); % First plant input CAi dropped because not used by MPC
```

EXAMPLE: LPV-MPC OF A NONLINEAR CSTR SYSTEM

- MPC design

```
% Discretize the plant model
Ts = 0.5; % hours
plant = c2d(sys,Ts);

% Design MPC Controller

% Specify signal types used in MPC
plant.InputGroup.MeasuredDisturbances = 1;
plant.InputGroup.ManipulatedVariables = 2;
plant.OutputGroup.Measured = 1;
plant.OutputGroup.Unmeasured = 2;
plant.InputName = 'Ti','Tc';
plant.OutputName = 'T','CA';

% Create MPC controller with default prediction and control horizons
mpcobj = mpc(plant);

% Set nominal values in the controller
mpcobj.Model.Nominal = struct('X', x0, 'U', u0(2:3), 'Y', y0, 'DX', [0 0]);
```

EXAMPLE: LPV-MPC OF A NONLINEAR CSTR SYSTEM

- MPC design (cont'd)

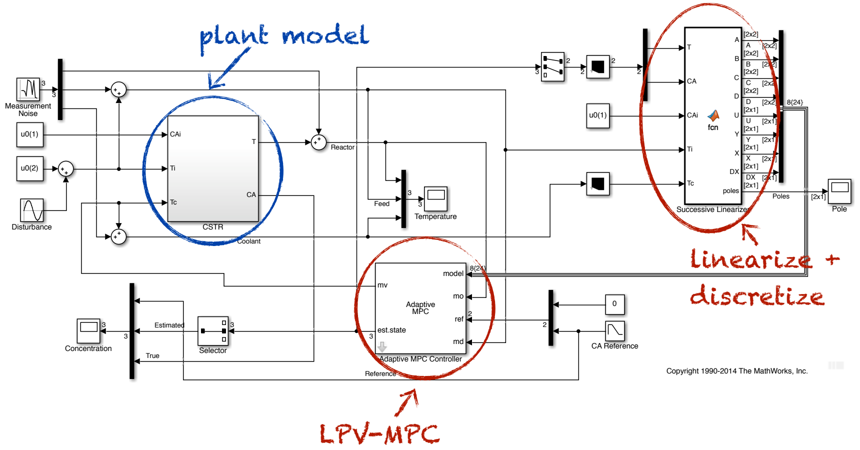
```
% Set scale factors because plant input and output signals have different
% orders of magnitude
Uscale = [30 50];
Yscale = [50 10];
mpcobj.DV(1).ScaleFactor = Uscale(1);
mpcobj.MV(1).ScaleFactor = Uscale(2);
mpcobj.OV(1).ScaleFactor = Yscale(1);
mpcobj.OV(2).ScaleFactor = Yscale(2);

% Let reactor temperature T float (i.e. with no setpoint tracking error
% penalty), because the objective is to control reactor concentration CA
% and only one manipulated variable (coolant temperature Tc) is available.
mpcobj.Weights.OV = [0 1];

% Due to the physical constraint of coolant jacket, Tc rate of change is
% bounded by degrees per minute.
mpcobj.MV.RateMin = -2;
mpcobj.MV.RateMax = 2;
```

EXAMPLE: LPV-MPC OF A NONLINEAR CSTR SYSTEM

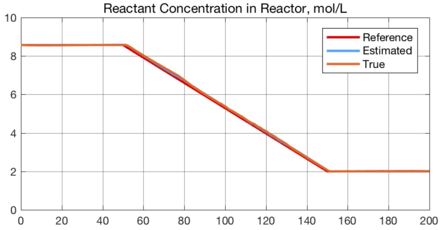
- Simulink diagram

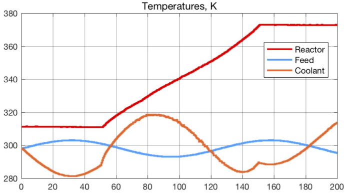


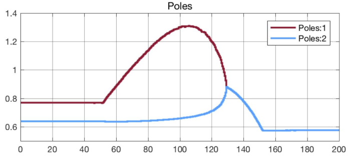

```
>> edit ampc_cstr_linearization
```

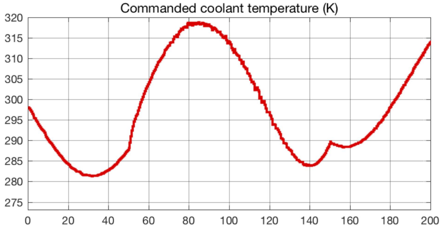
EXAMPLE: LPV-MPC OF A NONLINEAR CSTR SYSTEM

- Closed-loop results



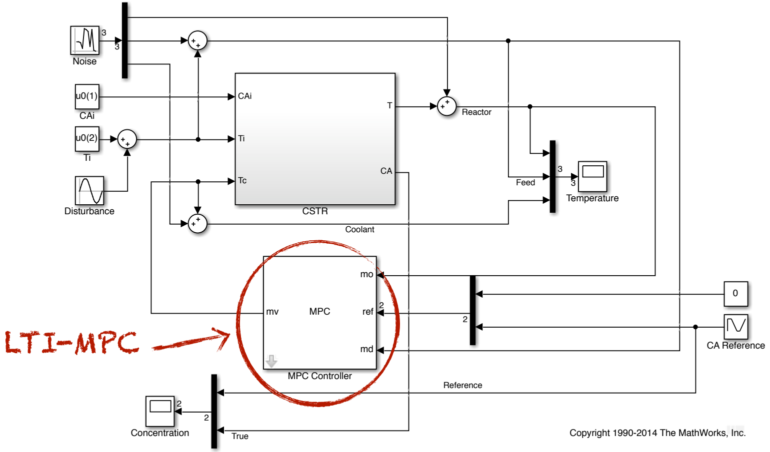






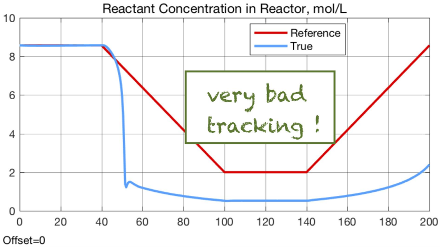
EXAMPLE: LTI-MPC OF A NONLINEAR CSTR SYSTEM

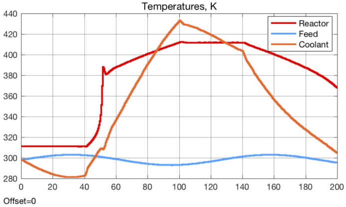
- Closed-loop results with **LTI-MPC**, same tuning

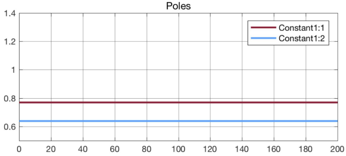


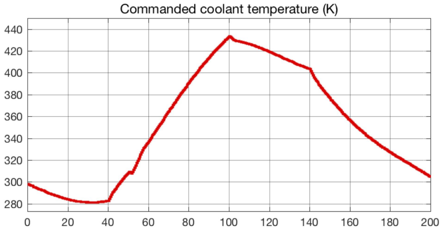
EXAMPLE: LTI-MPC OF A NONLINEAR CSTR SYSTEM

- Closed-loop results









LTV KALMAN FILTER

- Process model = **LTV model with noise**

$$\begin{aligned} x(k+1) &= A(k)x(k) + B(k)u(k) + G(k)\xi(k) \\ y(k) &= C(k)x(k) + \zeta(k) \end{aligned}$$

- $\xi(k) \in \mathbb{R}^q$ = zero-mean white **process noise** with covariance $Q(k) \succeq 0$
- $\zeta(k) \in \mathbb{R}^p$ = zero-mean white **measurement noise** with covariance $R(k) \succ 0$
- **measurement update**:

$$\begin{aligned} M(k) &= P(k|k-1)C(k)'[C(k)P(k|k-1)C(k)' + R(k)]^{-1} \\ \hat{x}(k|k) &= \hat{x}(k|k-1) + M(k)\left(y(k) - C(k)\hat{x}(k|k-1)\right) \\ P(k|k) &= (I - M(k)C(k))P(k|k-1) \end{aligned}$$

- **time update**:

$$\hat{x}(k+1|k) = A(k)\hat{x}(k|k) + B(k)u(k)$$

$$P(k+1|k) = A(k)P(k|k)A(k)' + G(k)Q(k)G(k)'$$

EXTENDED KALMAN FILTER

- Process model = **nonlinear model with noise**

$$\begin{array}{rcl} x(k+1) & = & f(x(k),u(k),\xi(k)) \\ y(k) & = & g(x(k),u(k))+\zeta(k) \end{array}$$

- **measurement update**:

$$\begin{array}{rcl} C(k) & = & \frac{\partial g}{\partial x}(\hat{x}_{k|k-1},u(k)) \\ M(k) & = & P(k|k-1)C(k)'[C(k)P(k|k-1)C(k)'+R(k)]^{-1} \\ \hat{x}(k|k) & = & \hat{x}(k|k-1)+M(k)\left(y(k)-g(\hat{x}(k|k-1),u(k))\right) \\ P(k|k) & = & (I-M(k)C(k))P(k|k-1) \end{array}$$

- **time update**:

$$\hat{x}(k+1|k) = f(\hat{x}(k|k),u(k))$$

$$A(k) = \frac{\partial f}{\partial x}(\hat{x}_{k|k},u(k),E[\xi(k)]),\ G(k) = \frac{\partial f}{\partial \xi}(\hat{x}_{k|k},u(k),E[\xi(k)])$$

$$P(k+1|k) = A(k)P(k|k)A(k)'+G(k)Q(k)G(k)'$$

NONLINEAR MODEL PREDICTIVE CONTROL

NONLINEAR MPC

(Mayne, Rawlings, Diehl, 2017)

- Nonlinear prediction model

$$\begin{cases} x_{k+1} & = & f(x_k, u_k) \\ y_k & = & g(x_k, u_k) \end{cases}$$

- Nonlinear constraints $h(x_k, u_k) \leq 0$
- Nonlinear performance index $\min \ \ell_N(x_N) + \sum_{k=0}^{N-1} \ell(x_k, u_k)$
- Optimization problem: **nonlinear programming problem (NLP)**

$$\begin{array}{ll} \min_z & F(z, x(t)) \\ \text{s.t.} & G(z, x(t)) \leq 0 \\ & H(z, x(t)) = 0 \end{array} \qquad z = \begin{bmatrix} u_0 \\ \vdots \\ u_{N-1} \\ x_1 \\ \vdots \\ x_N \end{bmatrix}$$

NONLINEAR OPTIMIZATION

(Nocedal, Wright, 2006)

- (Nonconvex) NLP is harder to solve than QP

- Convergence to a global optimum may not be guaranteed

- Several NLP solvers exist (such as **Sequential Quadratic Programming (SQP)**)

- NL-MPC is not used in practice so often, except for dealing with strong dynamical nonlinearities and slow processes (such as chemical processes)

FAST NONLINEAR MPC

(Lopez-Negrete, D'Amato, Biegler, Kumar, 2013)

- **Fast MPC**: exploit **sensitivity analysis** to compensate for the computational delay caused by solving the NLP
- **Key idea:** pre-solve the NLP between time $t-1$ and t based on the predicted state $x^*(t)=f(x(t-1),u(t-1))$ in background
- Get $u^*(t)$ and sensitivity $\left.\dfrac{\partial u^*}{\partial x}\right|_{x^*(t)}$ within sample interval $[(t-1)T_s, tT_s)$
- At time t, get $x(t)$ and compute

$$\boxed{u(t)=u^*(t)+\frac{\partial u^*}{\partial x}(x(t)-x^*(t))}$$

- Note that still one NLP must be solved within the sample interval

FROM LTV-MPC TO NL-MPC

- **Key idea:** Solve a **sequence of LTV-MPC** problems at the same time t

- Given the current state $x(t)$ and reference $\{r(t+k), u_r(t+k)\}$, initial guess $U_0 = \{u_0^0, \ldots, u_{N-1}^0\}$ and corresponding state trajectory $X_0 = \{x_0^0, \ldots, x_N^0\}$

- A good initial guess U_0, X_0 is the previous (shifted) optimal solution

- At a generic iteration i, **linearize** the NL model around U_i, X_i:

$$\begin{cases} x_{k+1} & = & f(x_k, u_k) \\ y_k & = & g(x_k) \end{cases}$$

$$A_i = \frac{\partial f(x_0^i, u_0^i)}{\partial x},\ B_i = \frac{\partial f(x_0^i, u_0^i)}{\partial u},\ C_i = \frac{\partial g(x_0^i, u_0^i)}{\partial x}$$

NONLINEAR MPC

For $h = 0$ to $h_{\max} - 1$ do:

1. Simulate from $x(t)$ with inputs U_h, parameter sequence P and get state trajectory X_h and output trajectory Y_h
2. Linearize around (X_h, U_h, P) and discretize in time with sample time T_s
3. Let U^*_{h+1} be the solution of the QP problem corresponding to LTV-MPC
4. Find optimal step size $\alpha_h \in (0, 1]$;
5. Set $U_{h+1} = (1 - \alpha_h)U_h + \alpha_h U^*_{h+1}$;

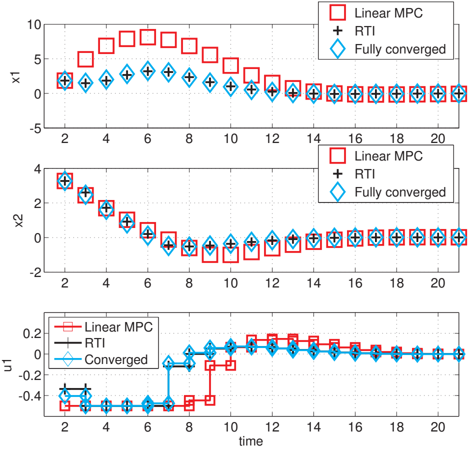
Return solution $U_{h_{\max}}$

- The method above is a **Gauss-Newton method** to solve the NL-MPC problem
- Special case: just solve one iteration with $\alpha = 1$ (a.k.a. **Real-Time Iteration**)

  (Diehl, Bock, Schloder, Findeisen, Nagy, Allgower, 2002)

(Gros, Zanon, Quirynen, Bemporad, Diehl, 2016)

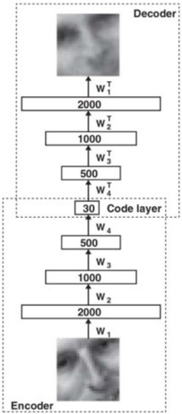
- Example



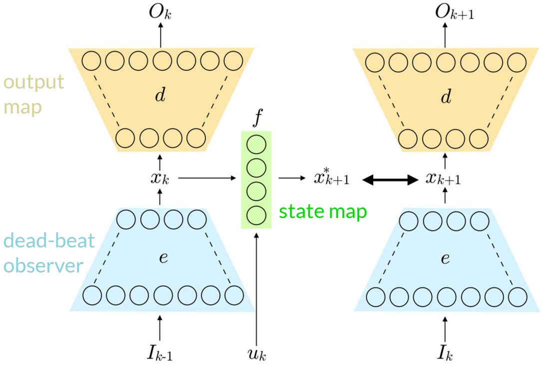
LEARNING NONLINEAR MODELS FOR MPC

(Masti, Bemporad, CDC 2018)

- **Idea:** use **autoencoders** and artificial neural networks to learn a **nonlinear state-space model** of **desired order** from input/output data



(Hinton, Salakhutdinov, 2006)

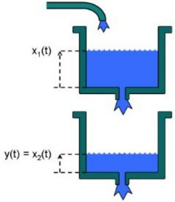


$$
\begin{aligned}
O_k &= [y_k' \; \dots \; y_{k-m}']' \\
I_k &= [y_k' \; \dots \; y_{k-n_a+1}' \; u_k' \; \dots \; u_{k-n_b+1}']'
\end{aligned}
$$

LEARNING NONLINEAR MODELS FOR MPC - AN EXAMPLE

(Masti, Bemporad, CDC 2018)

- System generating the data = nonlinear 2-tank benchmark

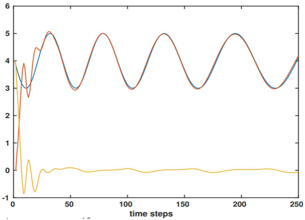


www.mathworks.com

$$\begin{cases} x_1(k+1) = x_1(k) - k_1\sqrt{x_1(k)} + k_2(u(k)+w(k)) \\ x_2(k+1) = x_2(k) + k_3\sqrt{x_1(k)} - k_4\sqrt{x_2(k)} \\ y(k) = x_2(k) + v(k) \end{cases}$$

Model is totally unknown to learning algorithm

- Artificial neural network (ANN): 3 hidden layers
60 exponential linear unit (ELU) neurons
- For given number of model parameters,
autoencoder approach is superior to NNARX
- **Jacobians** directly obtained from ANN structure
for Kalman filtering & MPC problem construction



LTV-MPC results