Low level implementation of MPC

José M. Maestre



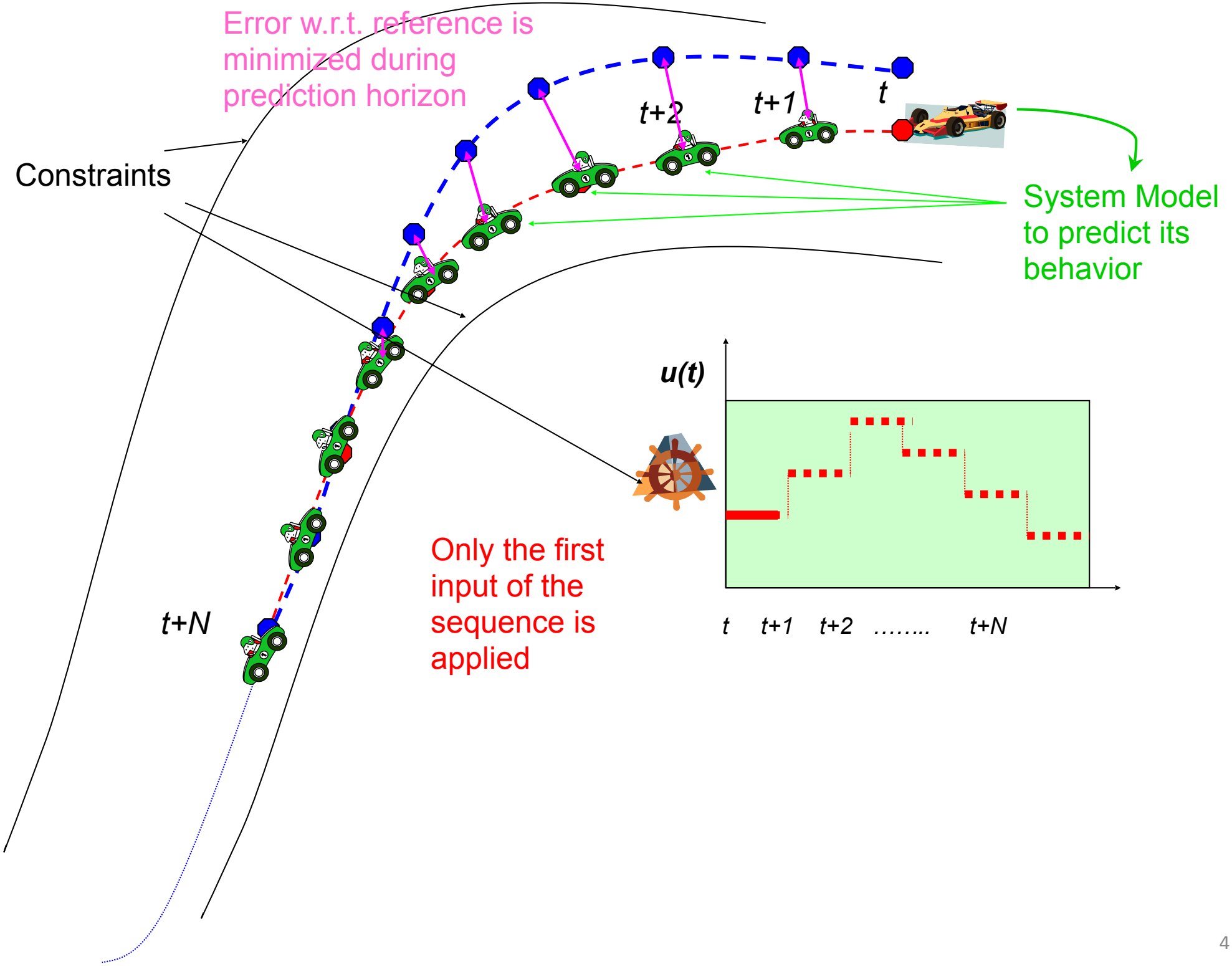
Previously…

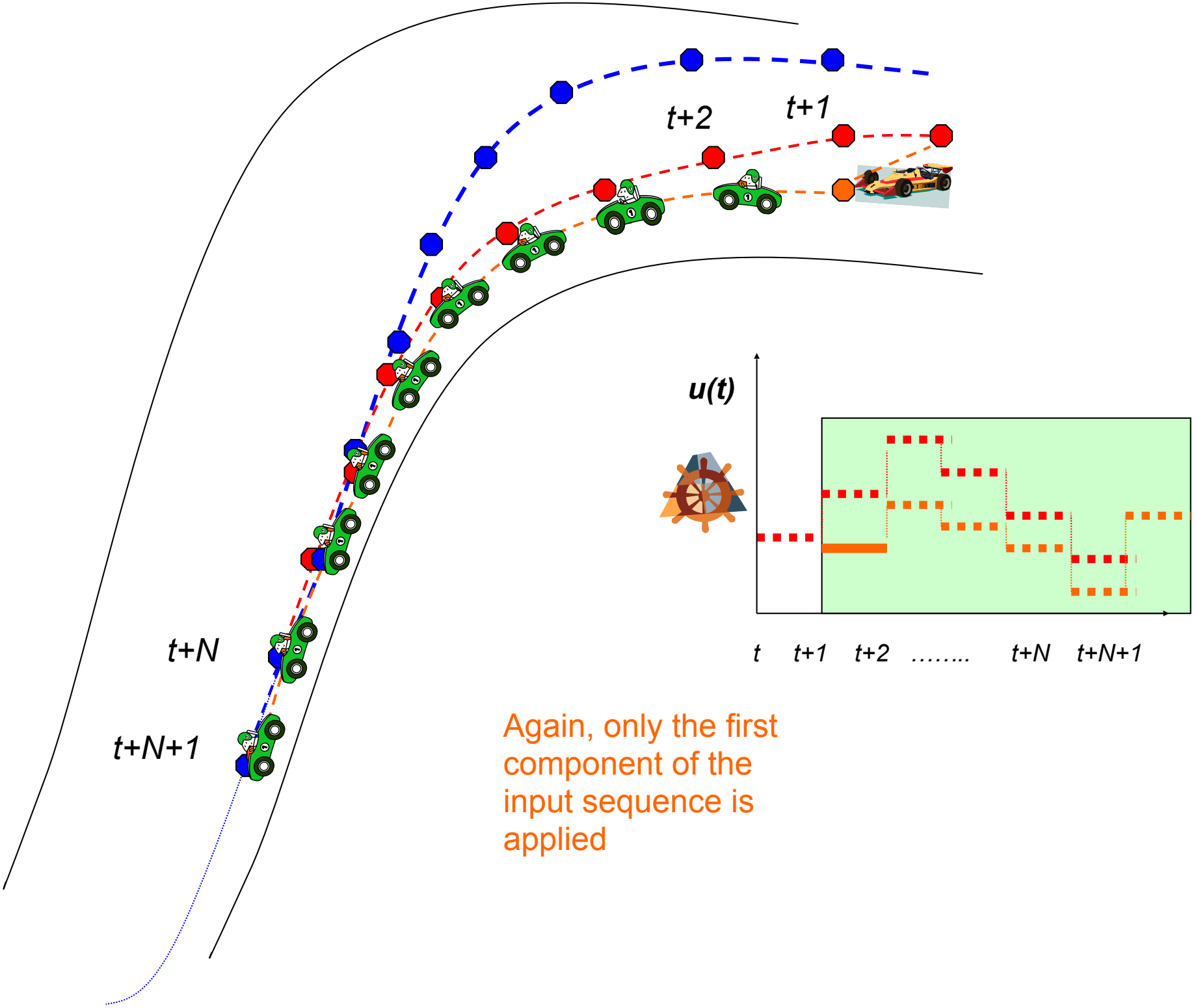
Model Predictive Control

- MPC is an advanced computer-based control approach that uses a mathematical model to predict the system evolution and calculate its optimal inputs according to a cost function during an N step prediction horizon

- It allows dealing explicitly complex phenomena such as constraints on the system variables, delays, disturbances, uncertainties, multiple optimization objectives, etc.

- Due to its versatility there are many applications, e.g.: solar plants, water systems, traffic control,…

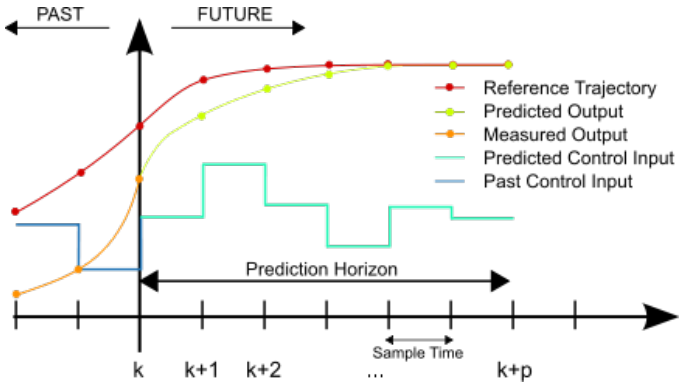

Error w.r.t. reference is minimized during prediction horizon
Constraints
t+2
t+1
t
System Model to predict its behavior
u(t)
Only the first input of the sequence is applied
t+N
t
t+1
t+2
……..
t+N



Again, only the first component of the input sequence is applied

Model Predictive Control

- At each time step, optimal inputs are calculated for the next N time steps according to a cost function
- The first component of the optimal input sequence is actually applied and the rest are discarded
- The process is repeated at the next time step



Model Predictive Control

What is it needed to apply MPC?

- System model available
- A cost function that involves the controlled and manipulated variables
- Prediction horizon (tuning parameter)
- Optimizer with additional information to take into account: disturbance forecast, constraints



Today…

Setpoint!!!



t+1

t+2

t+3

t+4

t+5

t+6

t+7

t+8

t+9



Lecture 2

Let us focus on a single iteration of the MPC controller:

1. Measure the system state
2. Compute sequence of control actions
3. Apply the element of the sequence that corresponds

We will design a Matlab function to do this

Toy Example

Consider a shop (integrator)

$$x(k+1)=x(k)+u(k)-w(k)$$

where

- x(k) is the number of units in our stock
- u(k) is the number of new units that we received from the distributor
- w(k) stands for the demand, i.e., how many units we sold every day



Toy Example

Toy example:

$$x(k+1)=x(k)+u(k)-w(k)$$

For simplicity:

- x(k), u(k) and w(k) are assumed to be real in this example.



Toy Example

Exercise 1:

We want to be as general as possible, so if we assume that any model can be written as:

$$x(k+1)=Ax(k)+Bu(k)+Dw(k)$$

Our model is: $x(k+1)=x(k)+u(k)-w(k)$

- A, B, D?

Model Predictive Control

Exercise 1:

We want to be as general as possible, so if we assume that any model can be written as:

$$x(k+1)=Ax(k)+Bu(k)+Dw(k)$$

Our model is: $x(k+1)=x(k)+u(k)-w(k)$

- Sol. A=1, B=1, D=-1

Matlab script

```
% This command is to clean the screen
clc;

% Clean the memory -> start from scratch
clear all;

% Define model matrices
A=1;
B=1;
D=-1;

% Initial state x(0)
x=5;
```

Model Predictive Control

I want to predict the evolution of the system

Assume I know my initial state x(k) at time k=0

What will be my state tomorrow?

$$x(1)=Ax(0)+Bu(0)+Dw(0)$$

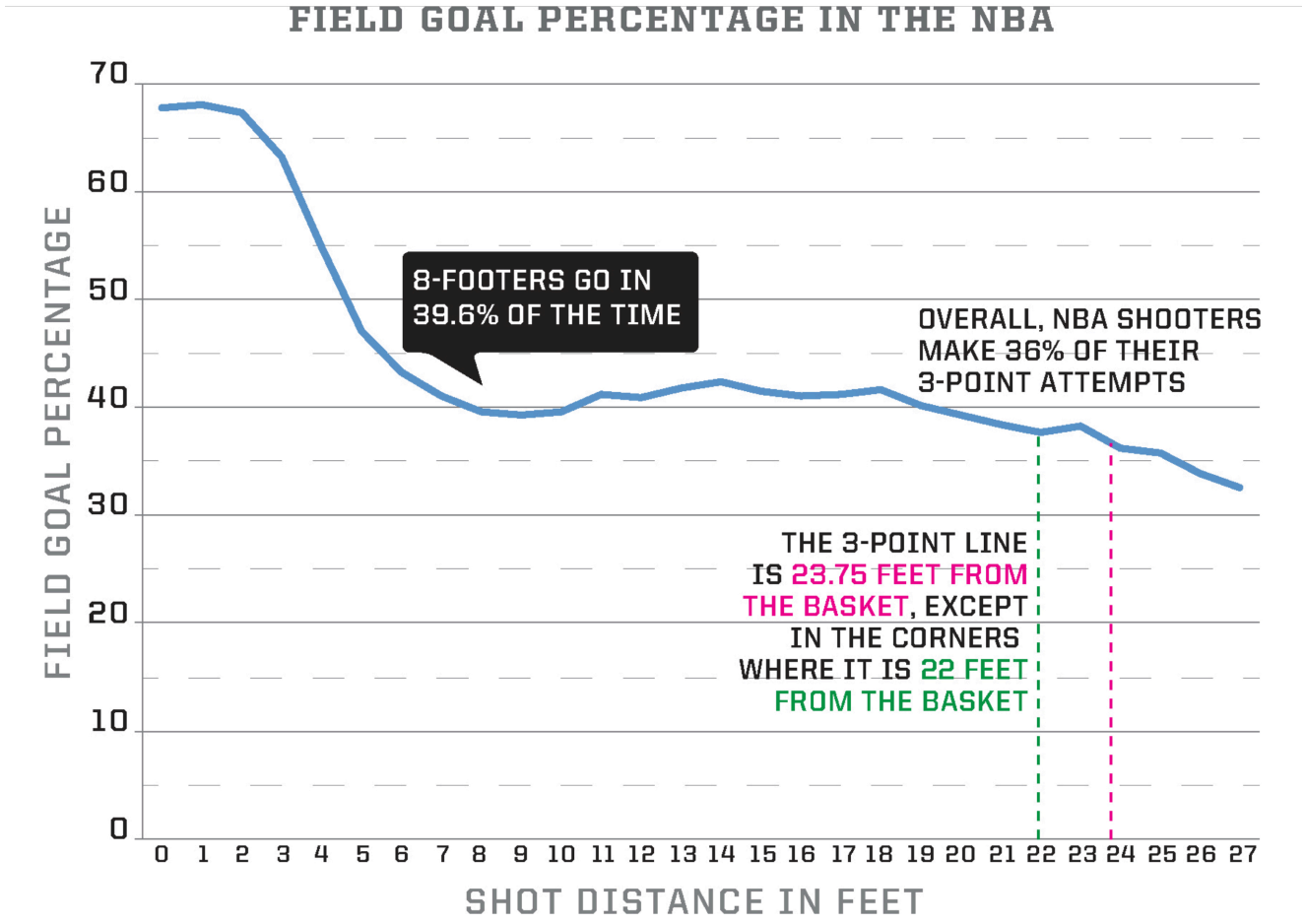
And the day after?

$$x(2)=Ax(1)+Bu(1)+Dw(1)$$



I can predict the state at k+1 as long as I am given the state, the actions and the demand at k

Your Daily Reminder



All models are wrong, but some are useful

You lose precision as you look into the future

Model Predictive Control

Exercise 2:

I would like to know these states:
$x(1), x(2), x(3),\ldots, x(N)$

What do you need taking into account that
$x(k+1)=Ax(k)+Bu(k)+Dw(k)$?

Model Predictive Control

Exercise 2:

I would like to know these states:
$x(1), x(2), x(3),\ldots, x(N)$

What do you need taking into account that
$x(k+1)=Ax(k)+Bu(k)+Dw(k)$?

Sol: $x(0), u(0), u(1), \ldots, u(N-1), w(0), w(1), \ldots, w(N-1)$

Model Predictive Control

Exercise 2 (cont.):

I want to know: x(1), x(2), x(3),…, x(N)

I need:
x(0), u(0), u(1), …, u(N-1), w(0), w(1), …, w(N-1)

What can you choose here?

Model Predictive Control

Exercise 2 (cont.):

I want to know: x(1), x(2), x(3),…, x(N)

I need:
x(0), u(0), u(1), …, u(N-1), w(0), w(1), …, w(N-1)

What can you choose here? **u(0), u(1), …, u(N-1)**
x(0) is given (number of items in stock)
w(0), w(1), …, w(N-1) has to be estimated

Model Predictive Control

Exercise 3:

Let:

$$X = \begin{bmatrix} x(1) \\ x(2) \\ \vdots \\ x(N) \end{bmatrix} \in \mathbb{R}^{N \cdot n_x \times 1},\ U = \begin{bmatrix} u(0) \\ u(1) \\ \vdots \\ u(N-1) \end{bmatrix} \in \mathbb{R}^{N \cdot n_u \times 1}, W = \begin{bmatrix} w(0) \\ w(1) \\ \vdots \\ w(N-1) \end{bmatrix} \in \mathbb{R}^{N \cdot n_w \times 1}$$

It is possible to write:

$$X = G_x x(0) + G_u U + G_w W \ \text{ with } G_x \in \mathbb{R}^{N \cdot n_x \times N \cdot n_x},\ G_u \in \mathbb{R}^{N \cdot n_x \times N \cdot n_u},\ G_w \in \mathbb{R}^{N \cdot n_x \times N \cdot n_w}$$

Calculate G_x, G_u, G_w.

Model Predictive Control

$$X = \begin{bmatrix} x(1) \\ x(2) \\ x(3) \\ \vdots \\ x(N) \end{bmatrix} = \underbrace{\begin{bmatrix} A \\ A^2 \\ A^3 \\ \vdots \\ A^N \end{bmatrix}}_{G_x} x(0) + \underbrace{\begin{bmatrix} B & 0 & 0 & 0 & 0 \\ AB & B & 0 & 0 & 0 \\ A^2B & AB & B & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 \\ A^{N-1}B & A^{N-2}B & \cdots & AB & B \end{bmatrix}}_{G_u} \begin{bmatrix} u(0) \\ u(1) \\ u(2) \\ \vdots \\ u(N-2) \\ u(N-1) \end{bmatrix} + \underbrace{\begin{bmatrix} D & 0 & 0 & 0 & 0 \\ AD & D & 0 & 0 & 0 \\ A^2D & AD & D & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 \\ A^{N-1}D & A^{N-2}D & \cdots & AD & D \end{bmatrix}}_{G_w} \begin{bmatrix} w(0) \\ w(1) \\ w(2) \\ \vdots \\ w(N-2) \\ w(N-1) \end{bmatrix}$$

```
N = 5;

% Useful variables
nx = size(A, 2); % number of states
nu = size(B, 2); % number of inputs
nw = size(D, 2); % number of disturbances

% Build Gx: i-th row block is A^(i)
Gx = [];
for i = 1:N
    Gx = [Gx; A^i];
end

% Build Gu: i-th row block and j-th column block
% Gw is built analogously
Gu = [];
for i = 1:N
  temp = [];
  for j = 1:N
     if j <= i
        temp = [temp, A^(i-j) * B];
     else
        temp = [temp, zeros(nx, nu)];
     end
  end
  Gu = [Gu; temp];
end
```

```
% Define horizon
N = 5;

% Useful variables
nx = size(A, 2); % number of states
nu = size(B, 2); % number of inputs
nw = size(D, 2); % number of disturbances

% Initialize Gx, Gu, and Gw
Gx = zeros(N * nx, nx);
Gu = zeros(N * nx, N * nu);
Gw = zeros(N * nx, N * nw);

% Build Gx: i-th row block is A^(i)
for i = 1:N
    Gx((i - 1) * nx + 1:i * nx, :) = A^i;
end

% Build Gu and Gw: i-th row block and j-th column block
for i = 1:N
    for j = 1:i
        % Gu: i-th row block and j-th column block is A^(i-j)*B
        Gu((i - 1) * nx + 1:i * nx, (j - 1) * nu + 1:j * nu) = A^(i - j) * B;
        % Gw: i-th row block and j-th column block is A^(i-j)*D
        Gw((i - 1) * nx + 1:i * nx, (j - 1) * nw + 1:j * nw) = A^(i - j) * D;
    end
end
```

Model Predictive Control

We would like our controller to minimize a certain cost function during the predicted steps, e.g.:

$$J = \sum_{k=1}^{N_\mathrm{p}} (x(k)^\mathrm{T} Q x(k) + u(k-1)^\mathrm{T} R u(k-1))$$

- Q penalizes stored units quadratically
- R penalizes placing orders quadratically



The minimization of this cost function leads to a convex optimization problem (can be solved efficiently)

Note that for simplicity we penalize deviations with respect to the state x=0. A change of basis is enough to regulate the state to a different setpoint.

Model Predictive Control

Exercise 4:

What is the cost of the system for u(0)=1, u(1)=2, u(2)=3, w(0)=4, w(1)=1, w(2)=2 and x(0)=5? Q and R are equal to 1.

$$J = \sum_{k=1}^{N_\mathrm{p}} (x(k)^\mathrm{T} Q x(k) + u(k-1)^\mathrm{T} R u(k-1))$$

Model Predictive Control

Exercise 4 (cont.):

Calculate J using the aggregated vectors X, U and W.

$$J = \sum_{k=1}^{N_\mathrm{p}} (x(k)^\mathrm{T} Q x(k) + u(k-1)^\mathrm{T} R u(k-1))$$

$$J = X^T \begin{bmatrix} Q & 0 & 0 & 0 \\ 0 & \ddots & 0 & 0 \\ 0 & 0 & Q & 0 \\ 0 & 0 & 0 & Q \end{bmatrix} X + U^T \begin{bmatrix} R & 0 & 0 & 0 \\ 0 & \ddots & 0 & 0 \\ 0 & 0 & R & 0 \\ 0 & 0 & 0 & R \end{bmatrix} U$$

$$J = X^T \hat{Q} X + U_T \hat{R} U \text{ with } \hat{Q} \in \mathbb{R}^{N \cdot n_x \times N \cdot n_x}, \hat{R} \in \mathbb{R}^{N \cdot n_u \times N \cdot n_u}$$

If we take into account now that: $X = G_x x(0) + G_u U + G_w W$

$$J = X^T \hat{Q} X + U^T \hat{R} U =$$

$$\left(G_x x(0) + G_u U + G_w W \right)^T \hat{Q} \left(G_x x(0) + G_u U + G_w W \right) + U^T \hat{R} U$$

$$J = \underbrace{\cancel{\left(G_x x(0)\right)^T \hat{Q}\left(G_x x(0)\right)}}_{const. \rightarrow remove} + \left(G_x x(0)\right)^T \hat{Q}\left(G_u U\right) + \underbrace{\cancel{\left(G_x x(0)\right)^T \hat{Q}\left(G_w W\right)}}_{const. \rightarrow remove}$$
$$+ \left(G_u U\right)^T \hat{Q}\left(G_x x(0)\right) + \left(G_u U\right)^T \hat{Q}\left(G_u U\right) + \left(G_u U\right)^T \hat{Q}\left(G_w W\right)$$
$$+ \underbrace{\cancel{\left(G_w W\right)^T \hat{Q}\left(G_x x(0)\right)}}_{const. \rightarrow remove} + \left(G_w W\right)^T \hat{Q}\left(G_u U\right) + \underbrace{\cancel{\left(G_w W\right)^T \hat{Q}\left(G_w W\right)}}_{const. \rightarrow remove} + U^T \hat{R} U$$

Trick #1: We want to optimize J w.r.t. U. Hence, constant terms can be dropped because they do not affect the result.

$$J = \left(G_x x(0)\right)^T \hat{Q}\left(G_u U\right)$$
$$+ \left(G_u U\right)^T \hat{Q}\left(G_x x(0)\right) + \left(G_u U\right)^T \hat{Q}\left(G_u U\right) + \left(G_u U\right)^T \hat{Q}\left(G_w W\right)$$
$$+ \left(G_w W\right)^T \hat{Q}\left(G_u U\right) + U^T \hat{R} U$$

Trick #2: What is J? Is it a vector? Is it a scalar?

$$J = 2\underbrace{\left(\left(G_x x(0)\right)^T \hat{Q} G_u + \left(G_w W\right)^T \hat{Q} G_u\right)}_{F} U + U^T \underbrace{\left(G_u^T \hat{Q} G_u + \hat{R}\right)}_{H} U$$

This is equivalent to optimize this function:

$$\frac{1}{2} U^T H U + F \cdot U$$

quadprog - Quadratic programming

This MATLAB function returns a vector x that minimizes 1/2*x'*H*x + f'*x.

x = quadprog(H,f)

Matlab

```
% Define cost function and expected disturbances
Q=1;
R=1;
W=[1 1 1 1 1]';  % expected demand (this is just an example)

% Build R_hat
R_hat = kron(eye(N),R);

% Build Q_hat
Q_hat=kron(eye(N),Q);

% Build cost function
H=Gu'*Q_hat*Gu+R_hat;
F=x'*Gx'*Q_hat*Gu+W'*Gw'*Q_hat'*Gu;
```

You can **play with quadrog** now

Model Predictive Control

Besides minimizing:

$$J = \sum_{k=1}^{N_p} (x(k)^T Q x(k) + u(k-1)^T R u(k-1))$$



We may want to satisfy constraints, e.g.:

$$u_{min} \leq u(k) \leq u_{max}$$

U = quadprog(H,F,[],[],[],[],LB,UB)

UB: Upper Bound Vector
LB: Lower Bound Vector

You can **play with quadrog** now