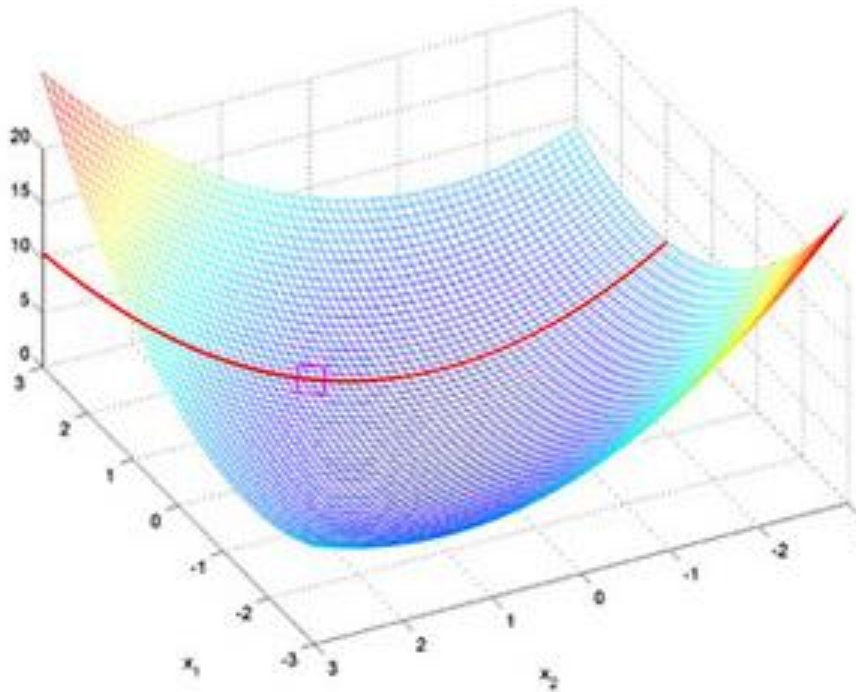


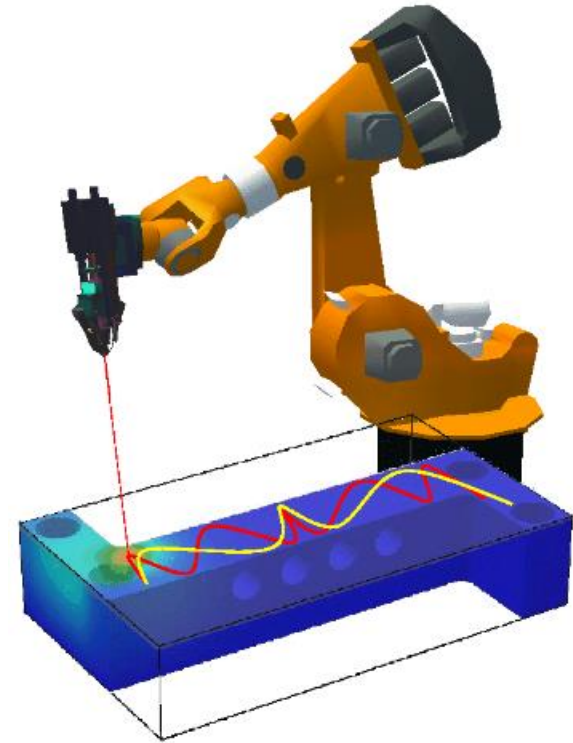
# MEGR 7090/8090: Advanced Optimal Control



$$V_n(\mathbf{x}_n) = \min_{\{\mathbf{u}_n, \mathbf{u}_{n+1}, \dots, \mathbf{u}_{N-1}\}} \left[ \frac{1}{2} \sum_{k=n}^{N-1} (\mathbf{x}_k^T \mathbf{Q}_k \mathbf{x}_k + \mathbf{u}_k^T \mathbf{R} \mathbf{u}_k) + \frac{1}{2} \mathbf{x}_N^T \mathbf{Q}_N \mathbf{x}_N \right]$$

$$\begin{aligned} V_n(\mathbf{x}_n) &= \min_{\{\mathbf{u}_n, \mathbf{u}_{n+1}, \dots, \mathbf{u}_{N-1}\}} \left[ \frac{1}{2} \sum_{k=n}^{N-1} (\mathbf{x}_k^T \mathbf{Q}_k \mathbf{x}_k + \mathbf{u}_k^T \mathbf{R} \mathbf{u}_k) + \frac{1}{2} \mathbf{x}_N^T \mathbf{Q}_N \mathbf{x}_N \right] \\ &= \min_{\mathbf{u}_n} \left[ \frac{1}{2} (\mathbf{x}_n^T \mathbf{Q}_n \mathbf{x}_n + \mathbf{u}_n^T \mathbf{R} \mathbf{u}_n) + \underbrace{\min_{\{\mathbf{u}_{n+1}, \dots, \mathbf{u}_{N-1}\}} \left[ \frac{1}{2} \sum_{k=n+1}^{N-1} (\mathbf{x}_k^T \mathbf{Q}_k \mathbf{x}_k + \mathbf{u}_k^T \mathbf{R} \mathbf{u}_k) + \frac{1}{2} \mathbf{x}_N^T \mathbf{Q}_N \mathbf{x}_N \right]}_{V_{n+1}(\mathbf{x}_{n+1})} \right] \\ &= \min_{\mathbf{u}_n} \left[ \frac{1}{2} (\mathbf{x}_n^T \mathbf{Q}_n \mathbf{x}_n + \mathbf{u}_n^T \mathbf{R} \mathbf{u}_n) + V_{n+1}(\mathbf{x}_{n+1}) \right] \end{aligned}$$

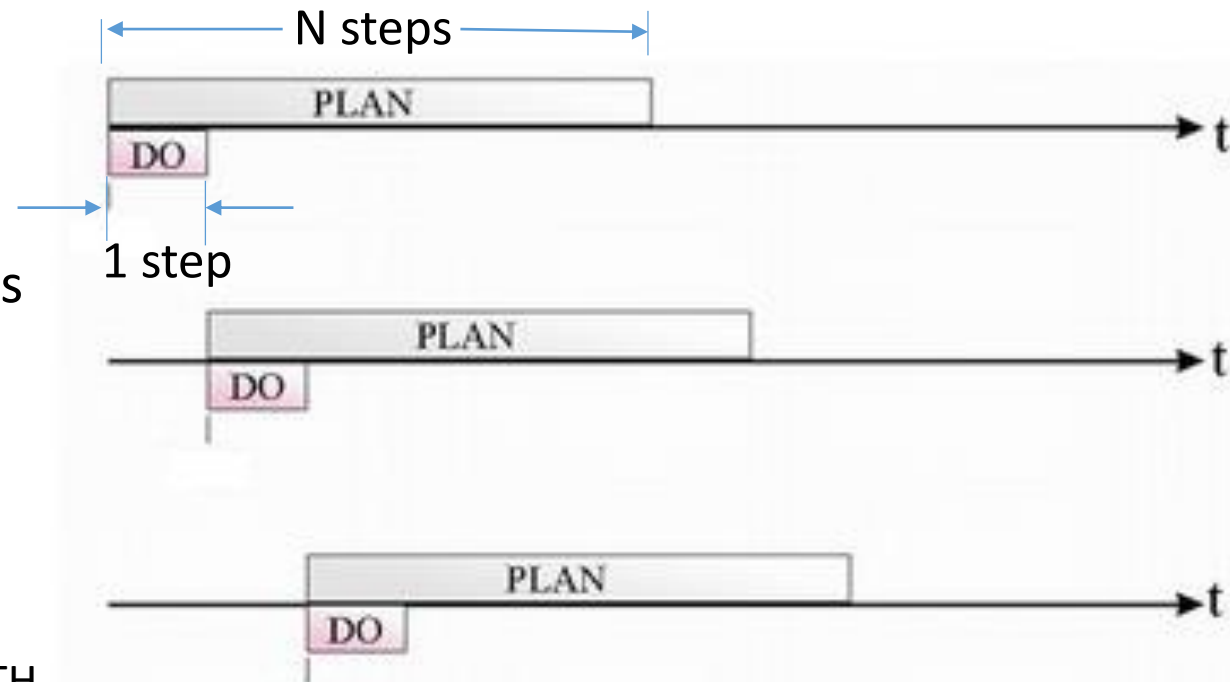
$$V_n(\mathbf{x}_n) = \min_{\mathbf{u}_n} \left[ \frac{1}{2} (\mathbf{x}_n^T \mathbf{Q}_n \mathbf{x}_n + \mathbf{u}_n^T \mathbf{R} \mathbf{u}_n) + V_{n+1}(\mathbf{x}_{n+1}) \right]$$



Lecture 18  
October 24, 2017

# Main Idea of MPC – Reminder

- At time 0, compute the control trajectory that minimizes  $J(\mathbf{u}, \mathbf{x}(0)) = \sum_{i=0}^{N-1} g(\mathbf{x}(i), u(i)) + h(\mathbf{x}(N))$ , subject to constraints. Implement  $u^*(0)$ .
- At the next time step (1), compute the control signal that minimizes  $J(\mathbf{u}, \mathbf{x}(1)) = \sum_{i=1}^N g(\mathbf{x}(i), u(i)) + h(\mathbf{x}(N+1))$ , subject to constraints
- Repeat every time step
- This process is known as ***model predictive control (MPC)***, also known as ***receding horizon control***



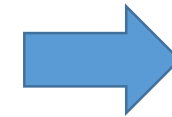
# MPC Notation – Reminder

**Note:** At every time step, MPC optimizes a **sequence** of control signals. Those sequences overlap. (Example: If  $N = 10$ ), then a value for  $u(9)$  is computed at steps 0, 1,..., 9. To differentiate between these different values, we use what Dr. V calls “slash notation”:

- $\mathbf{u}(k) = [u(k|k) \quad \dots \quad u(k + N - 1|k)]^T$  = candidate control sequence at step  $k$
- $\mathbf{x}_{seq}(k) = [\mathbf{x}(k|k) \quad \dots \quad \mathbf{x}(k + N|k)]^T$  = predicted state sequence at step  $k$
- In general, the notation is interpreted as (time step at which the variable is evaluated | time step at which the optimization is performed)

## General mathematical formulation for MPC:

$$\mathbf{u}^*(k) = \arg \min \left[ \sum_{i=k}^{N-1} g(\mathbf{x}(i|k), u(i|k)) + h(\mathbf{x}(k + N|k)) \right]$$



$$u(k) = u^*(k|k)$$

(Implement the first step of the optimized sequence)

subject to:  $\mathbf{x}(i|k) \in X, i = k \dots k + N$   
 $u(i|k) \in U, i = k \dots k + N - 1$  and:  $\mathbf{x}(i + 1|k) = f(\mathbf{x}(i|k), u(i|k))$

# Linear MPC – Reminder

**Constrained discrete-time LQR problem – also known as *linear MPC*:**

$$\mathbf{u}^*(k) = \arg \min \left[ \sum_{i=k}^{N-1} (\mathbf{x}^T(i|k)Q\mathbf{x}(i|k) + Ru^2(i|k)) + \mathbf{x}^T(k+N|k)S\mathbf{x}(k+N|k) \right] \Rightarrow u(k) = u^*(k|k)$$

Subject to:

$$\begin{aligned} M_1\mathbf{x}(i|k) - \mathbf{b}_1 &\leq 0, i = k \dots k+N \\ M_2u(i|k) - \mathbf{b}_2 &\leq 0, i = k \dots k+N-1 \end{aligned} \quad \text{and} \quad \mathbf{x}(i+1|k) = A\mathbf{x}(i|k) + Bu(i|k)$$

**Solution – *Quadratic programming (QP)*:**

- Refer to lecture 9 notes for proof that the above problem is convex and therefore QP can be used reliably

**Simulink's built-in MATLAB toolbox can be used to perform linear MPC (constrained discrete-time LQR)**

# Benefit and Limitations to Linear MPC



**Major benefit:** Convex constraints on the state and control signal can be enforced

## Limitations:

- System dynamics *must be linear* (other than constraints) or linearized
- Stage cost *must be quadratic* in the states and control signal

Today, we will discuss nonlinear MPC, which removes the above limitations at the expense of computational expensiveness.

# General Nonlinear MPC Setup

**General optimization problem (note the free-form objective function, constraints, and model):**

$$\mathbf{u}^*(k) = \arg \min \left[ \sum_{i=k}^{N-1} g(\mathbf{x}(i|k), u(i|k)) + h(\mathbf{x}(k+N|k)) \right] \quad \longrightarrow \quad u(k) = u^*(k|k)$$

Subject to:  $\mathbf{x}(i|k) \in X, i = k \dots k+N$  and  $\mathbf{x}(i+1|k) = f(\mathbf{x}(i|k), u(i|k))$   
 $u(i|k) \in U, i = k \dots k+N-1$

## Solution techniques:

- Sequential quadratic programming (SQP) – refer to lecture 12-13 notes for a refresher
- Dynamic programming (DP) – usually, the actual DP calculations must be done online, and a lookup table (generated by DP) can then be used to determine the required control signal

# Dynamic Programming for MPC (Building a Lookup Table)



**Recall:** With **backward recursion**, DP allows you to *specify a constraint on the terminal state* ( $\mathbf{x}(N)$ ) and *determine the optimal control sequence for all possible initial states* ( $\mathbf{x}(0)$ )

## **Process for using DP to build a lookup table for MPC:**

- Perform DP using backward recursion (gives you the result indicated above) – refer back to lectures 14-16 for a reminder on how to do this)
- For each gridded state value, store  $u^*(0)$ , noting that with MPC, we will take  $u(k) = u^*(0)$
- Use a 1D lookup table (available in Simulink) to determine  $u(k)$  for interpolated points

# Sequential Quadratic Programming (SQP) for MPC



## Recall:

- At each iteration, SQP first derives a quadratic approximation of the Lagrangian (setting up the “QP subproblem”)
- SQP then moves in the direction specified by the QP subproblem solution
- `fmincon` can be used to perform SQP

**Challenge:** `fmincon` is not supported by Simulink!!

## Workarounds:

- Option 1 – Choose another SQP solver
- Option 2 – Write SQP yourself
- Option 3 – Generate a ***level 2 s-function*** – We will focus on this one



# Level 2 s-functions – Main Ideas and References



## Main ideas:

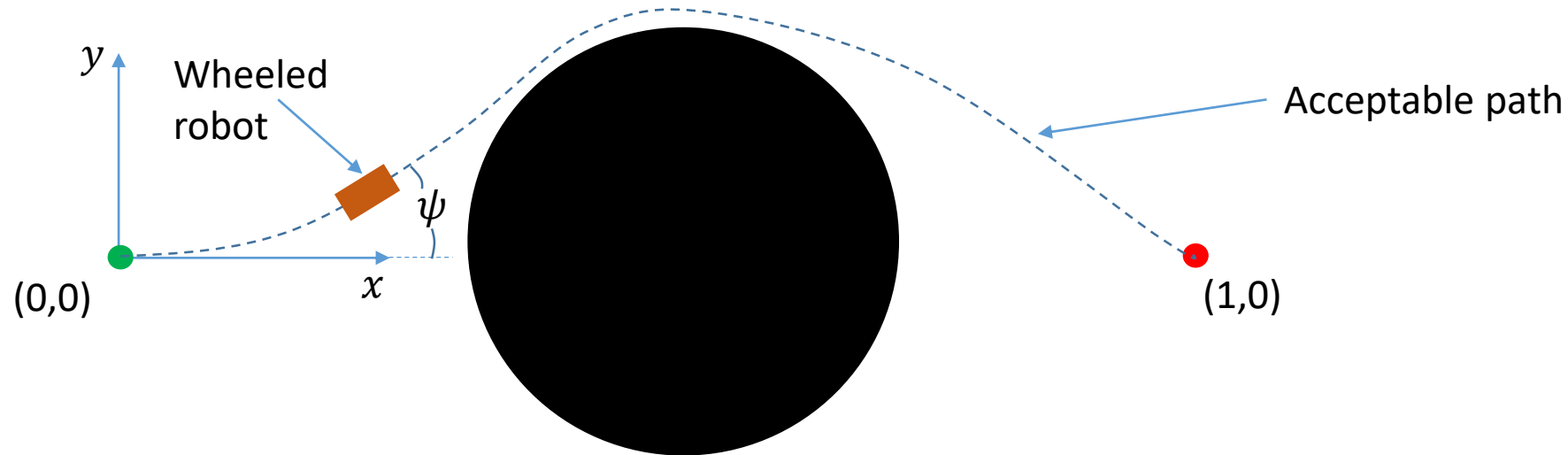
- Embedded MATLAB functions only support a ***subset*** of MATLAB functionality (for code generation reasons)
- s-functions provide a way for you to take code written in a non-native language or non-supported MATLAB code and run it in Simulink
- Level 2 s-functions are for code that was written in MATLAB but not supported by Simulink blocks (fmincon is an example of this)
- We will work through a level 2 s-function example...a good video tutorial to level 2 s-functions can be found at:

<https://www.youtube.com/watch?v=X-qVign6BLg>

<https://www.youtube.com/watch?v=ZnvRoGA23uU>

# Nonlinear MPC Example

**Scenario:** A wheeled mobile robot must travel from point A (0,0) to point B (1,0) while avoiding a circular obstacle of radius  $R = 0.25$



**Dynamic model:**

$$\dot{x} = v \cos \psi$$

$$\dot{y} = v \sin \psi$$

$$\dot{\psi} = u$$

where:

$$v = v_0((x - 1)^2 + (y - 1)^2)$$

$$-5 \leq u(t) \leq 5, \forall t$$

The robot slows down automatically as it nears the finish line.

# Nonlinear MPC Example – Continued



## Tasks:

- Given the scenario described on the previous slide, determine an acceptable stage cost ( $g(\mathbf{x}(i|k), \mathbf{u}(i|k))$ ), terminal cost ( $h(\mathbf{x}(k + N|k), \mathbf{u}(k + N|k))$ ), and constraint sets ( $U$  and  $X$ )
- Implement a Simulink model of the robot and MPC-based controller, using `fmincon` and a level 2 s-function

Sample code available on Canvas

# Nonlinear MPC Setup with Disturbances

General optimization problem (note the free-form objective function, constraints, and model):

$$\mathbf{u}^*(k) = \arg \min \left[ \sum_{i=k}^{N-1} g(\mathbf{x}(i|k), u(i|k)) + h(\mathbf{x}(k+N|k)) \right] \quad \longrightarrow \quad u(k) = u^*(k|k)$$

Subject to:  $\mathbf{x}(i|k) \in X, i = k \dots k+N$  and  $\mathbf{x}(i+1|k) = f(\mathbf{x}(i|k), u(i|k), d(i|k))$   
 $u(i|k) \in U, i = k \dots k+N-1$

## Assessment of candidate solution techniques:

- Offline dynamic programming with online lookup table: The DP solution will be ***different for every disturbance profile*** ( $d(i|k), i = k \dots k+N-1$ )...this makes the DP-based lookup table ***cumbersome to generate***
- SQP, on the other hand, can take into account the new value of the disturbance at each time step

# Preview of Upcoming Lectures



**October 26 – Stability and robustness of MPC**

**After October 26 – Optimal control for continuous-time systems**