Mini-Project 3: Advanced Nonlinear MPC Methods

Model Predictive Control Course

Thursday 1st January, 2026

1 Project Overview

1.1 Objectives

This mini-project aims to deepen your understanding of advanced nonlinear Model Predictive Control (MPC) techniques through hands-on implementation and comparative analysis. You will:

I. Implement four distinct nonlinear MPC strategies for the same plant model used in Mini-Projects 1 and 2.

II. Develop a sequential linearization approach with Quadratic Programming (QP).

III. Implement the MPC with Nonlinear Prediction and Linearization along the Trajectory (MPC-NLPT).

IV. Design a Nonlinear MPC using Interior-Point (IP) method.

V. Design a Nonlinear MPC using Sequential Quadratic Programming (SQP).

VI. Compare performance against a baseline Linear Quadratic (LQ) controller.

VII. Analyze computational complexity, tracking accuracy, and robustness.

1.2 Plant Model

Use the same nonlinear system from Mini-Project 1.

$$x_{k+1} = f(x_k, u_k, d_k) \tag{1}$$

$$y_k = g(x_k) \tag{2}$$

where $x \in \mathbb{R}^{n_x}$, $u \in \mathbb{R}^{n_u}$, and $d \in \mathbb{R}^{n_d}$ represent states, control inputs, and disturbances respectively.

1.3 Trajectory Tracking Task

Track the reference trajectory $r(t)$ defined in Mini-Project 2. Also, the performance metrics to evaluate are:

- Integral Absolute Error (IAE): $\int_0^T |r(t) - y(t)|dt$
- Maximum deviation: $\max_t |r(t) - y(t)|$
- Control effort: $\int_0^T u^T(t)Ru(t)dt$

2 Controller Design Specifications

2.1 Method 1: Sequential Linearization MPC with QP

At each time step k, linearize the nonlinear model around the current operating point as given in the class:

$$\delta x_{k+1} = A_k \delta x_k + B_k \delta u_k + c_k \tag{3}$$

where $\delta x = x - x_k$, $\delta u = u - u_{k-1}$.
Derive the prediction model in compact form:

$$\mathbf{x} := \begin{bmatrix} x_{k+1|k} \\ x_{k+2|k} \\ \vdots \\ x_{k+N_p|k} \end{bmatrix} = S_x x_k + S_u \boldsymbol{\Delta}\mathbf{u} + S_c \tag{4}$$

where $\boldsymbol{\Delta}\mathbf{u} = \begin{bmatrix} \Delta u_k & \Delta u_{k+1} & \cdots & \Delta u_{k+N_c-1} \end{bmatrix}^T$.
Formulate the QP problem:

$$\min_{\boldsymbol{\Delta}\mathbf{u}} \quad \frac{1}{2}\boldsymbol{\Delta}\mathbf{u}^T H \boldsymbol{\Delta}\mathbf{u} + g^T \boldsymbol{\Delta}\mathbf{u} + \text{const} \tag{5}$$

$$\text{s.t.} \quad \underline{u} \le u \le \overline{u} \tag{6}$$

$$\underline{\Delta u} \le \boldsymbol{\Delta}\mathbf{u} \le \overline{\Delta u} \tag{7}$$

$$\underline{x} \le S_x x_k + S_u \boldsymbol{\Delta}\mathbf{u} + S_c \le \overline{x} \tag{8}$$

Implementation Requirements:

- Re-linearize at each sampling time
- Apply the obtained control law to the *original nonlinear* model
- Tune prediction horizon N_p and control horizon N_c

2.2 Method 2: MPC-NLPT

This method iteratively linearizes along a predicted trajectory:

Algorithm 1 MPC-NLPT Algorithm

1: At time k, initialize $\boldsymbol{\Delta}\mathbf{u}^{(0)} = \mathbf{0}$
2: Predict trajectory $\mathbf{x}^{(0)}$ using constant inputs ($\boldsymbol{\Delta}u = 0$) applied to Nonlinear model
3: Linearize nonlinear model along $\mathbf{x}^{(0)}$ to get $(A^{(0)}, B^{(0)})$
4: Formulate QP using (4)
5: Solve QP to obtain $\boldsymbol{\Delta}\mathbf{u}$
6: Apply $u_k = u_{k-1} + \Delta u$

It is notable that, the entire process can be repeated further until the desired linearization accuracy is reached. This can be performed by approximating the future trajectory of the system using the newly determined system inputs (This is Not necessary in the project).

2.3 Method 3: Nonlinear MPC using Interior-Point Method

Formulate the complete nonlinear optimization:

$$\min_{\mathbf{x},\mathbf{u}} \quad \sum_{i=0}^{N_p-1} \ell(x_i, u_i) + \ell_f(x_{N_p}) \tag{9}$$

$$\text{s.t.} \quad x_{i+1} = f(x_i, u_i), \quad i = 0, \ldots, N_p - 1 \tag{10}$$

$$x_0 = x(t_k) \tag{11}$$

$$\underline{x} \leq x_i \leq \overline{x}, \quad i = 1, \ldots, N_p \tag{12}$$

$$\underline{u} \leq u_i \leq \overline{u}, \quad i = 0, \ldots, N_c - 1 \tag{13}$$

$$u_i = u_{N_c-1}, \quad i = N_c, \ldots, N_p - 1 \tag{14}$$

Implement an Interior-point method. Only for this part, you MAY apply an Interior-Point solver (e.g., IPOPT via CasADi).

Implementation Requirements:

- Compare computation time with QP-based methods
- Analyze sensitivity to initial guess

2.4 Method 4: Nonlinear MPC using SQP

Implement SQP with (These are some recommendations NOT requirements):

- Quasi-Newton Hessian approximation (BFGS)
- Line search or trust region globalization
- Warm-starting strategies

Form QP sub-problem at each SQP iteration:

$$\min_{\mathbf{d}} \quad \frac{1}{2}\mathbf{d}^T \nabla^2 \mathcal{L} \mathbf{d} + \nabla \mathcal{J}^T \mathbf{d} \tag{15}$$

where $\mathbf{d}$ is the search direction and $\mathcal{L}$ is the Lagrangian.

Implementation Requirements:

- Limit SQP iterations to meet real-time constraints
- Compare convergence with IP method

2.5 Method 5: Baseline LQ Controller

Implement the infinite-horizon LQ controller from Mini-Project 2 for comparison.

3 Implementation Guidelines

Your submission must include a main file `run_all.m` or `run_all.py` that:

a) Defines the nonlinear plant model

b) Sets up simulation parameters and reference trajectory

c) Calls all four MPC controller functions

d) Simulates the closed-loop system

e) Generates all plots and performance metrics

4 Deliverables

4.1 1. Video Presentation (Max 10 minutes)

Upload to the course platform as a private/unlisted video. Content must include:

- **Introduction**: Brief overview of the project objectives
- **Theory**: Key mathematical formulations for each method
- **Code Walkthrough**: Show the structure and key implementation details
- **Simulation Demo**: Run `run_all` and show real-time results
- **Discussion**: Key findings and challenges

Requirements:

- Screen recording
- Show code execution and plotting
- Include captions for accessibility

4.2 2. Technical Report (Max 10 pages)

It should include the following sections:

1. **Abstract** (100 words max): Summary of work and key findings
2. **Introduction**: Problem statement and objectives
3. **Theoretical Background**: Mathematical derivations for each method
4. **Implementation Details**: Software architecture, parameters, and tuning
5. **Results**:
   - Trajectory tracking plots for all controllers
   - Performance metrics comparison table
   - Computation time histograms
   - Control input and state constraint satisfaction plots
6. **Discussion**:
   - Comparison of linear vs. nonlinear methods
   - Trade-offs between accuracy and computation
   - Robustness analysis
   - Limitations and challenges
7. **Conclusion**: Key takeaways

4.3 3. Code Submission

Upload as a single ZIP file containing:

- `run_all.m/.py` - Main execution script
- `models/` - Plant model
- `controllers/` - All controller implementations
- `results/` - Saved figures and data (generated by code)