Maze-Solving Robot using PID, LQR, MPC, and Sliding Mode Controllers

By

Gautham Ramkumar, Eshwar Srinivasan, Shriman Raghav Srinivasan, Tapan Patil

1. Problem Statement and Scope

Autonomous robots frequently must navigate complicated environments with reliability and efficiency in robotics navigation tasks, especially when faced with uncertainties, non-linearities, and dynamic disturbances. Path-following, or navigating a robot along a predetermined, collision-free path from an initial location to a desired destination while maintaining high precision and resilience, is a key challenge in autonomous navigation.

The objective of this project is to apply, assess, compare, and evaluate several control techniques on a unicycle-model mobile robot navigating intricate 2D maze environments. These techniques include Proportional-Integral-Derivative (PID), Linear Quadratic Regulator (LQR), Model Predictive Control (MPC), and Sliding Mode Control (SMC). Through its sampling-based methodology, the Probabilistic Roadmap (PRM) planning algorithm—which has shown notable benefits in managing complex configuration spaces—was used to construct the paths for these environments. By randomly sampling the configuration space and joining adjacent collision-free configurations, the PRM algorithm effectively creates collision-free paths. This makes it especially appropriate for complicated labyrinth environments where conventional deterministic methods would not work well.

Specifically, our project focuses on the following key tasks:

- Generating collision-free reference paths using PRM in maze environments of varying sizes and complexities.
- Implementing four different path-following control strategies: PID, LQR, MPC, and SMC.
- Comparing these controllers based on quantitative performance metrics, including path-tracking accuracy, control effort, robustness to uncertainty, and computational efficiency.

This project's scope involves MATLAB simulations, control strategy development, robot kinematics modelling (unicycle model), and performance evaluation. With these initiatives, we aim to determine the comparative advantages and disadvantages of every control strategy and offer precise solutions for robotic navigation tasks.

2. Background

Autonomous robotic navigation in unpredictable or challenging situations is a fundamental subject matter with an array of applications, from unmanned aerial vehicles and autonomous cars and industrial automation to planetary exploration robots. The

objective of robot navigation is to achieve accurate path-following behavior that is robust despite environmental disturbances, sensor noise, and modeling errors. This challenge has become increasingly crucial as robots are employed in more complex and dynamic environments where traditional control techniques might not be sufficient.
Path planning and control are two related issues that have traditionally been separated out of the scope of autonomous navigation. The robot's accuracy and efficiency in following paths are determined by control strategies, whereas path planning concentrates on identifying viable routes through the environment. Commonly employed for differential-drive robots, the unicycle model adds complexity by limiting the robot's instantaneous movement in arbitrary directions due to its nonholonomic limitations. Researchers have created a variety of methods throughout the years to deal with the control issues in robotic route following. PID and other traditional control techniques are straightforward, but they frequently have trouble with complex dynamics. To overcome some drawbacks of traditional methods, sophisticated strategies like predictive control (MPC), resilient control (SMC), and optimum control (LQR) have been developed. Each approach offers different trade-offs between tracking precision, resistance to disturbances, and computational complexity. Specifically, labyrinth settings, with their constrained passageways, abrupt curves, and possible dead ends, offer a great platform for testing these control schemes in demanding circumstances that mimic actual navigation situations.

3. Motivation

Autonomous navigation in constrained environments such as mazes presents a multifaceted challenge for robotic systems. Effective navigation not only demands precise control over the robot's trajectory but also requires robustness to environmental disturbances, adaptability to sharp path changes, and computational efficiency to ensure real-time performance. The selection of a suitable control strategy plays a pivotal role in determining how reliably and efficiently a robot can follow a planned path through such environments.

This project was driven by the goal of evaluating and comparing four well-established control strategies—Proportional-Integral-Derivative (PID), Linear Quadratic Regulator (LQR), Model Predictive Control (MPC), and Sliding Mode Control (SMC) each bringing distinct benefits and limitations to the problem of maze navigation:

- **PID** control is one of the most widely used techniques in robotics due to its simplicity, intuitive tuning, and low computational demands. It relies on error feedback to adjust control inputs and performs well in structured environments. Despite its limitations in handling nonlinearity and dynamic changes, PID serves as an essential baseline to benchmark the effectiveness of more advanced controllers.

- **LQR** provides an optimal control solution by minimizing a predefined quadratic cost function that penalizes state deviations and control effort. It is particularly useful in systems that can be linearized around a reference trajectory. LQR ensures smooth and energy-efficient motion, and its gains can be computed systematically, making it advantageous for systems with known dynamics.
- **SMC** is a nonlinear control method designed to handle uncertainties, disturbances, and model inaccuracies effectively. It drives the system state onto a sliding surface and maintains it despite environmental variations. Its robustness makes it well-suited for unpredictable or noisy environments, although it may introduce high-frequency switching (chattering) if not carefully tuned.
- **MPC** adds predictive capabilities by solving a finite-horizon optimization problem at each control step, considering future states, constraints, and trajectory deviations. This allows the robot to anticipate and adapt to upcoming challenges in the maze. With its ability to explicitly enforce constraints and plan, MPC excels in complex environments, though it requires higher computational resources than other methods.

By developing a modular simulation framework integrating these controllers, this project aims to understand their behavior across varying maze complexities, explore trade-offs between tracking accuracy, robustness, and execution time, and provide insights into their suitability for real-world robotic navigation tasks.

4. Methodology

The implementation framework for this project was developed entirely in MATLAB, utilizing its Robotics System Toolbox for environment modeling, visualization, and simulation. The robot is modeled as a unicycle-type mobile robot, governed by non-holonomic kinematic equations. It receives linear and angular velocity commands and operates within predefined velocity and acceleration constraints.

A collection of maze environments ranging in complexity from 15×15 to 50×50 grids were used to evaluate performance. These environments were represented using binary occupancy maps, where '0' denotes free space and '1' denotes obstacles. Start and goal positions for each map were predefined.

4.1 Path planning

To generate a feasible trajectory for the robot, the **Probabilistic Roadmap (PRM)** algorithm was employed. PRM constructs a graph of randomly sampled collision-free nodes, connects them based on distance thresholds and line-of-sight checks, and finds the shortest path using graph search. Obstacle inflation was applied to ensure a safe margin during planning.

4.2 Robot Modeling

The robot is modeled using the standard unicycle model:$\dot{x} = v\cos(\theta)\,, \quad \dot{y} = v\sin(\theta)\,, \quad \dot{\theta} = \omega$

This captures essential constraints of wheeled mobile robots, such as the inability to move sideways.

4.3 Controller Implementation

Each controller was integrated into the same simulation framework to allow for consistent evaluation:

- **PID**: Used the heading error and distance to the next waypoint to compute angular and linear velocities. Anti-windup logic and clamping were introduced to improve stability and responsiveness.
- **LQR**: Linearized the robot dynamics around the reference pose and applied optimal gain matrices computed via the LQR algorithm. Q and R matrices were selected to balance tracking performance and control effort.
- **SMC**: Implemented a nonlinear control law using a sliding surface based on pose error. The controller-maintained robustness to model uncertainties and disturbances. A smooth `tanh` function was used to mitigate chattering effects.
- **MPC**: At each control step, solved a finite horizon optimization problem to minimize predicted tracking error while respecting velocity constraints. This allowed anticipatory control and better handling of tight turns and dynamic path changes.

4.4 Simulation and Evaluation

The robot was initialized at the start point of each maze and commanded to follow the PRM-generated path using the selected controller. During simulation, the following metrics were recorded:

- Mean tracking error between the robot's actual trajectory and the reference path.
- Total control steps taken to reach the goal.
- Time taken to reach the goal.
- Control input profiles (linear and angular velocities).
- Controller efficiency measured as error per unit time.

The results were visualized and compared using trajectory plots, bar charts, and controller performance summaries to draw insights about accuracy, robustness, and computational feasibility of each method.

5.Results and Analysis

5.1 Execution Time Performance

- LQR demonstrated superior execution efficiency, completing navigation tasks in approximately 200 seconds for the 15×15 maze and 600 seconds for the 45×45 maze. This represents a linear scaling relationship with maze complexity.
- MPC required substantially longer execution times, ranging from 350 seconds for the simplest maze to approximately 1500 seconds for the most complex maze. The execution time increased disproportionately with maze complexity.
- SMC maintained moderate execution times between LQR and MPC, showing reasonable scaling with increased maze complexity.
- PID exhibited inconsistent scaling behavior, performing efficiently in simple mazes but requiring significantly more time in complex environments, particularly in the 45×45 maze.

5.2 Control Efficiency

- LQR consistently required the fewest control steps across all maze sizes, demonstrating superior control efficiency with approximately 9,000 steps for the most complex maze.
- MPC showed moderate control step requirements, using approximately 11,000 steps for the 45×45 maze.
- SMC and PID both required substantially more control steps, approaching 27,000 iterations for the most complex maze, suggesting they make frequent small adjustments during navigation.

5.3 Tracking Accuracy

- All controllers maintained comparable error metrics between 6-10 units across most maze sizes, with no controller demonstrating consistently superior tracking accuracy. MPC achieved the lowest mean error (approximately 6.4 units) in the 30×30 maze, showing effective optimization in moderately complex environments.
- A notable performance anomaly occurred in the 35×35 maze, where all controllers experienced increased tracking errors, suggesting this maze contained challenging features.

- For the most complex 45×45 maze, all controllers demonstrated improved tracking accuracy compared to some simpler mazes, with errors converging to approximately 6 units.

6. Discussion

6.1 Major Design Decisions

- **Unicycle Model Selection**: We chose a unicycle kinematic model for the robot to capture nonholonomic constraints while maintaining mathematical tractability. This model accurately represents the motion capabilities of wheeled robots without excessive complexity.
- **Controller Parameter Tuning Strategy**: For each controller, we employed different parameter tuning approaches:
  - LQR: We balanced position tracking with control effort minimization
  - MPC: We increased the prediction horizon to 25 steps and path error weight to 45.0 for better anticipatory behavior
  - SMC: We used higher gains with smoothing to reduce chattering
  - PID: We implemented conservative gains with anti-windup protection
- **Path Planning Approach**: We integrated MATLAB's PRM implementation with an inflation radius of 4 units to create safety margins around obstacles, balancing path optimality with collision avoidance.
- **Modular Architecture**: We developed a modular code structure that separated path planning, controller implementation, and performance evaluation, enabling fair comparison between controllers.

6.2 Implementation Challenges

Throughout the project, we encountered several significant challenges:

1. **MPC Optimization Stability**: The nonlinear optimization in MPC frequently faced convergence issues in complex mazes. We addressed this by implementing improved optimizer settings and fallback mechanisms for optimization failures.
2. **Waypoint Switching Logic**: Determining the optimal time to switch to the next waypoint proved challenging for all controllers. Simple distance thresholds caused issues in sharp turns, requiring more sophisticated switching logic incorporating path curvature and heading alignment.
3. **Nonholonomic Constraints**: The unicycle model's inability to move sideways created difficulties in tracking paths with sharp turns. This required careful velocity modulation and anticipatory steering, particularly challenging for PID and SMC implementations.

4. **Computational Resource Management**: Balancing real-time performance with control accuracy, especially for MPC, required significant optimization of the control loop timing and integration with visualization components.
5. **Controller Fairness Comparison**: Establishing fair comparison metrics across fundamentally different control approaches required careful design of the evaluation framework and standardized performance metrics.

6.3 Key Learnings and Takeaways

This project provided several valuable insights into control system design and implementation:

1. **Performance-Complexity Tradeoff**: We observed a clear relationship between controller complexity and performance. MPC offered superior tracking in specific scenarios but at substantially higher computational cost, while LQR provided excellent efficiency with good performance. This demonstrates that more complex controllers are not always better for all applications.
2. **Environment-Specific Optimization**: Controller performance varied significantly across different maze complexities. This reinforces the importance of environment-specific optimization and suggests that adaptive control approaches might be beneficial.
3. **Anticipatory Control Value**: Controllers with predictive capabilities (MPC) & enhanced lookahead showed advantages in complex environments, highlighting the value of anticipatory control in trajectory tracking.
4. **Implementation Practicality**: The theoretical advantages of certain controllers did not always translate to practical performance benefits. For instance, MPC's theoretical optimality came with significant implementation challenges and computational overhead that sometimes outweighed its performance advantages.
5. **Cross-Controller Design Transfer**: Techniques that improved one controller often benefited others. For example, path lookahead strategies initially developed for MPC were successfully adapted for PID and SMC, improving their performance in complex environments.

7. Next Steps

7.1. Implement Alternative Path Planning Algorithms

7.2. Explore and implement additional **Path Planning Algorithms** to compare against the PRM method. Some possible algorithms to consider include:

7.2.1. **A*** a graph-based algorithm that finds the shortest path between start and goal.

7.2.2. **RRT (Rapidly exploring Random Tree)** which is an incremental method used for motion planning in high-dimensional spaces.
7.2.3. **Dijkstra's Algorithm** a classic algorithm for finding the shortest path in weighted graphs, which can be useful for maze-solving tasks.

7.3. Extend Simulation to ROS2 Gazebo & Isaac Sim using MATLAB plugin for ROS.

7.4. Hardware Implementation and Validation, after testing the controllers and algorithms in the ROS2 and Isaac Sim environments, implement the controllers on actual hardware (e.g. Turtlebot3 or Another Mobile Platform)

7.5. Validate the performance differences between simulation and real-world deployment by comparing key metrics such as time to goal, path efficiency, and control effort.

7.6. Real-World Testing and Fine-Tuning:

7.6.1. Fine-tune the controllers and path planning algorithms based on real-world testing results. Address any issues that arise from the difference between simulation and actual robot performance, such as sensor noise, actuator limitations, or environmental disturbances.

8. Conclusion

In this project, we implemented and compared four control algorithms (PID, LQR, MPC, and SMC) for maze-solving robots. The performance of each controller was evaluated across seven different maze sizes. The results from the simulations were analyzed based on **control steps**, **mean tracking error**, **path length**, and **total time** to solve the maze.

8.1. **PID** demonstrated the fastest solution times and lowest computational overhead but struggled with accuracy in complex environments.

8.2. **LQR** provided smooth control with lower error rates but required accurate modeling and showed some limitations in handling nonlinearity.

8.3. **MPC** excelled in accuracy, especially in complex maze configurations, but was computationally expensive and slow.

8.4. **SMC** provided robustness to disturbances and uncertainties but suffered from chattering, making it less suitable for smooth trajectory tracking.

Annexure-Findings

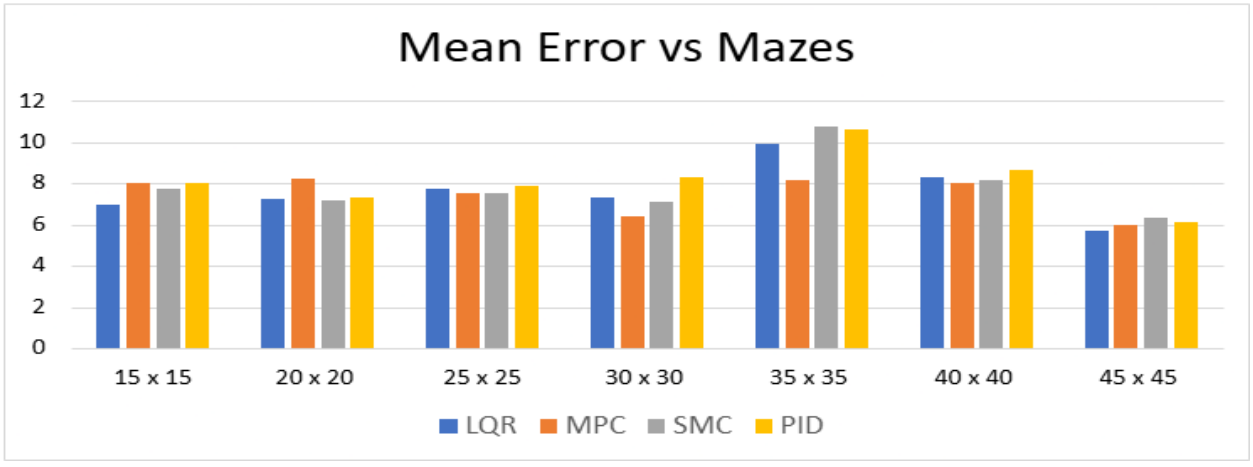
Metric	LQR	MPC	SMC	PID
Execution Time (45x45 maze)	600 s	1500 s	550s	800 s
Scalability With Complexity	Linear Increase	Exponential Increase	Moderate Increase	Variable Increase
Control Steps (45x45 maze)	~9000	~11,000	~27,000	~27,000
Mean Tracking Error	6-10 m	6-10 m	6-10 m	6-10 m
Best Error Performance	5.8 m	6.4 m	6.3 m	6.1 m
Worst Error Performance	9.9 m	8.2 m	10.8 m	10.5 m
Path length variation	Minimal	Minimal	Slightly longer in medium mazes	Shortest in simple mazes
Computational Efficiency	Highest	Lowest	Moderate	High in simple mazes
Performance-to-Complexity Ratio	Excellent	Good	Moderate	Variable
Best Application Scenario	Resource-constrained systems	Precision critical applications	Variable environments	Simple environments

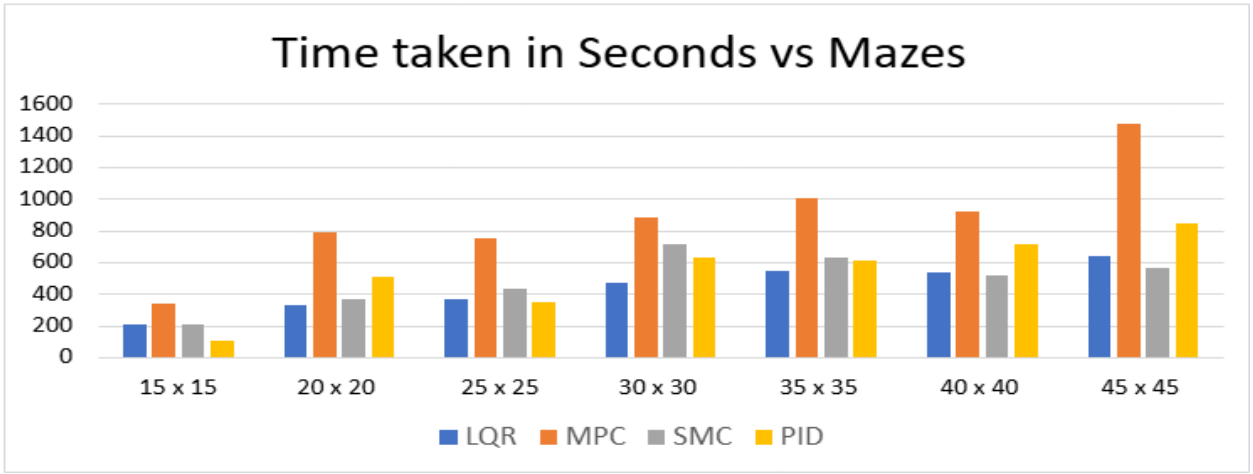
Key Findings:

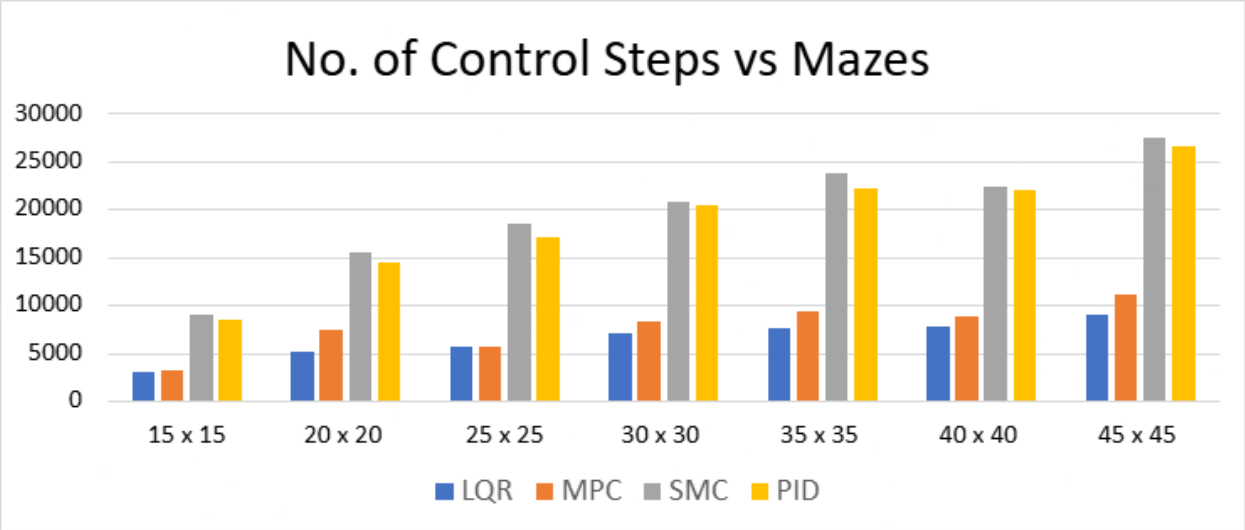
- LQR offers the best balance of speed and accuracy across all maze sizes
- MPC provides good tracking but with significantly higher computational cost
- SMC and PID require approximately 3× more control steps than LQR

- All controllers struggle with certain maze complexities

Annexure Graphs






No. of Control Steps vs Mazes
30000
25000
20000
15000
10000
5000
0
15 x 15
20 x 20
25 x 25
30 x 30
35 x 35
40 x 40
45 x 45
LQR
MPC
SMC
PID