The Journal of Open Source Software

LinearMPC.jl: A Julia Package for Embedded Linear Model Predictive Control

Daniel Arnström [1¶]

1 Independent Researcher, Sweden ¶ Corresponding author

DOI: 10.21105/joss.10384

Software

- Review
- Repository
- Archive

Editor: Johan Larsson

Reviewers:

- @jbcaillau
- @goerz
- @Yuricst

Submitted: 28 March 2026
Published: 17 June 2026

Summary

`LinearMPC.jl` is a Julia package for model predictive control (MPC) of linear systems. It provides a user-friendly development environment for designing, simulating, and deploying MPC controllers while targeting high-performance embedded implementations. The package bridges the gap between rapid prototyping in a high-level language and deployment on resource-constrained hardware by generating allocation-free, library-free C code that can run on any microcontroller. `LinearMPC.jl` supports both online optimization via the dual active-set quadratic programming (QP) solver DAQP (Arnström et al., 2022) and explicit MPC solutions computed by the multi-parametric QP solver ParametricDAQP (Arnström & Axehill, 2024). Additional features include robust MPC with constraint tightening, state estimation via Kalman filters, hybrid MPC with binary controls, game-theoretic MPC for Nash equilibria, and complexity certification for real-time guarantees.

Statement of need

Model predictive control is the dominant advanced control strategy in industry, used in applications ranging from process control to autonomous vehicles and robotics. At each sampling instant, MPC solves an optimization problem that accounts for a dynamical model, constraints, and a performance objective. For linear systems, this optimization problem is a QP. Deploying MPC on embedded systems, such as microcontrollers in automotive, aerospace, and robotic applications, poses several challenges: the controller must execute within strict timing deadlines, the code must be lightweight and allocation-free, and the implementation must be verifiable.

State of the field

Existing tools address parts of the above-mentioned workflow. `ModelPredictiveControl.jl` (Gagnon et al., 2026) provides a high-level Julia interface for MPC design but does not target embedded deployment. Similarly, `do-mpc` (Fiedler et al., 2023) provides a high-level Python interface for MPC design, but without any support for code generation to deploy on embedded hardware. The `acados` framework (Verschueren et al., 2022) provides fast embedded solvers for nonlinear optimal control, but is not tailored for *linear* MPC applications. MATLAB's Model Predictive Control Toolbox offers a mature commercial solution. However, there is a gap for a tool that combines (i) an expressive, high-level Julia interface for rapid prototyping, (ii) generation of lightweight, allocation-free C code for any embedded target, (iii) state-of-the-art explicit MPC computation, and (iv) complexity certification for hard real-time guarantees. `LinearMPC.jl` fills this gap by providing a unified workflow from design to deployment for linear MPC.

Software design

The architecture of `LinearMPC.jl` follows a layered design (Figure 1):

1. **Model layer.** Users define system dynamics as discrete-time or continuous-time state-space models, transfer functions (via `ControlSystems.jl` integration), or nonlinear models that are automatically linearized using `ForwardDiff.jl`. Measured disturbances are also supported.
2. **MPC formulation layer.** An expressive API lets users configure the controller: setting objectives with tracking, rate-of-change, and linear cost terms; defining input, output, and general polyhedral constraints (hard or soft); specifying prediction and control horizons with move blocking; and adding prestabilizing feedback for numerical conditioning.
3. **Optimization layer.** The MPC problem is condensed into a dense QP and solved using the DAQP solver (Arnström et al., 2022), a dual active-set method specialized for embedded applications. For hybrid systems with binary controls, DAQP's branch-and-bound capabilities are used. Game-theoretic MPC problems are solved by computing Nash equilibria of coupled QPs.
4. **Explicit MPC layer.** The parametric solution of the QP, a piecewise affine function over polyhedral regions (Bemporad et al., 2002), can be computed using ParametricDAQP (Arnström & Axehill, 2024), which is approximately 100 times faster than competing solvers. A binary search tree (Tøndel et al., 2003) is constructed for efficient online point location.
5. **Deployment layer.** The `codegen` function generates standalone, allocation-free C code for both online and explicit MPC controllers, including state observers when configured. The generated code depends on no external libraries and can be compiled for any target with a C compiler.
6. **Verification layer.** Complexity certification via the `ASCertain` package (Arnström & Axehill, 2022) provides worst-case iteration bounds for the DAQP solver, enabling hard real-time guarantees before deployment.

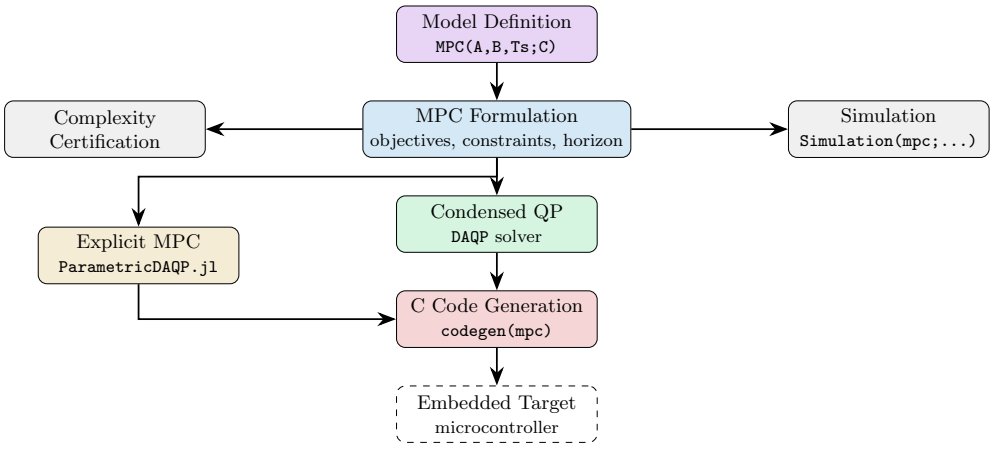

Figure 1: Simplified workflow of `LinearMPC.jl`: from model definition and MPC design in Julia to embedded C code deployment.

The central optimization problem solved at each sampling instant is

$$\begin{aligned} \underset{u_0,\dots,u_{N-1}}{\text{minimize}} \quad & \frac{1}{2}\sum_{k=0}^{N-1}\left((Cx_k - r)^T Q(Cx_k - r) + u_k^T R u_k + \Delta u_k^T R_r \Delta u_k + l_k^T u_k\right) \\ \text{subject to} \quad & x_{k+1} = Fx_k + Gu_k, \quad k = 0, \dots, N-1 \\ & x_0 = \hat{x} \\ & \underline{b} \leq A_x x_k + A_u u_k \leq \overline{b}, \quad k = 0, \dots, N-1 \end{aligned} \tag{1}$$

where $\hat{x}$ is the current state estimate and r is the desired reference. The objective weights Q, R, R_r and the constraint matrices A_x, A_u are configured by the user. Optional terminal costs, cross terms, reference preview, and time-varying linear costs extend this basic formulation.

Key features

- **Rapid prototyping.** Controllers are designed and tested in Julia with a concise, expressive API. Closed-loop simulations and plotting are built in.
- **Embedded code generation.** Allocation-free, library-free C code is generated for both online and explicit MPC, including Kalman filter observers.
- **Explicit MPC.** State-of-the-art computation of piecewise affine control laws, with visualization of critical regions and feedback surfaces.
- **Robust MPC.** A priori constraint tightening handles bounded process noise and state estimation uncertainty, with guarantees that constraints are satisfied under worst-case disturbances.
- **Hybrid MPC.** Binary control variables are supported for systems with on/off actuators, solved via branch-and-bound within DAQP.
- **Game-theoretic MPC.** Generalized Nash equilibria can be computed for multi-agent linear MPC problems where each agent has its own objective.
- **Complexity certification.** Worst-case solver iteration counts can be certified offline, enabling hard real-time guarantees.
- **Numerical conditioning.** Prestabilizing feedback mitigates ill-conditioning for unstable systems with long prediction horizons.

Research impact statement

`LinearMPC.jl` has been used to generate high-performing and real-time-certified MPC controllers for nano-quadcopters (Wikner et al., 2026), and to handle prioritized constraints in MPC applications (Arnström & Garofalo, 2025). Moreover, the sister package `lmpc`, which is a Python wrapper of `LinearMPC.jl`, has been used to investigate novel methods for learning-based control (Schmidtobreick et al., 2026).

Zooming out, the underlying solver `DAQP` has been used in several real-world applications, including automotive, robotics, and aerospace applications. The parametric solver `PDAQP` has recently been integrated into `cvxpygen` to support generating closed-form solutions to parametric optimization problems (Schaller et al., 2025).

Example usage

The following example demonstrates the core workflow: defining a system, designing an MPC controller, simulating, and generating embedded C code:

```
using LinearMPC

# Continuous-time inverted pendulum on a cart
```

```
A = [0 1 0 0; 0 -10 9.81 0; 0 0 0 1; 0 -20 39.24 0]
B = 100*[0; 1.0; 0; 2.0;;]
C = [1.0 0 0 0; 0 0 1.0 0]

# Create MPC with sample time 0.01, horizon 50/5
mpc = LinearMPC.MPC(A, B, 0.01; C, Np=50, Nc=5)
set_objective!(mpc; Q=[1.2^2, 1], R=[0.0], Rr=[1.0])
set_bounds!(mpc; umin=[-2.0], umax=[2.0])

# Compute a control action
u = compute_control(mpc, [0,0,0,0], r=[1, 0])

# Simulate closed-loop and generate C code
sim = Simulation(mpc; x0=zeros(4), r=[1,0], N=100)
LinearMPC.codegen(mpc; dir="mpc_codegen")
```

AI usage disclosure

Generative AI (GitHub Copilot, powered by Claude) was used to assist in drafting parts of this manuscript. The core software design decisions, algorithmic implementations, and all technical content were made by the human author. All AI-assisted output was reviewed and edited by the author.

References

Arnström, D., & Axehill, D. (2024). A high-performant multi-parametric quadratic programming solver. *2024 IEEE 63rd Conference on Decision and Control (CDC)*, 303–308. https://doi.org/10.1109/CDC56724.2024.10886132

Arnström, D., & Axehill, D. (2022). A unifying complexity certification framework for active-set methods for convex quadratic programming. *IEEE Transactions on Automatic Control*, *67*(6), 2758–2770. https://doi.org/10.1109/TAC.2021.3090749

Arnström, D., Bemporad, A., & Axehill, D. (2022). A dual active-set solver for embedded quadratic programming using recursive LDL^T updates. *IEEE Transactions on Automatic Control*, *67*(8), 4362–4369. https://doi.org/10.1109/TAC.2022.3176430

Arnström, D., & Garofalo, G. (2025, December 20). *Prioritized constraints in optimization-based control*. https://doi.org/10.48550/arXiv.2512.18458

Bemporad, A., Morari, M., Dua, V., & Pistikopoulos, E. N. (2002). The explicit linear quadratic regulator for constrained systems. *Automatica*, *38*(1), 3–20. https://doi.org/10.1016/S0005-1098(01)00174-1

Fiedler, F., Karg, B., Lüken, L., Brandner, D., Heinlein, M., Brabender, F., & Lucia, S. (2023). do-mpc: Towards FAIR nonlinear and robust model predictive control. *Control Engineering Practice*, *140*, 105676. https://doi.org/10.1016/j.conengprac.2023.105676

Gagnon, F., Thivierge, A., Desbiens, A., & Bagge Carlson, F. (2026, May 5). *ModelPredictiveControl.jl: Advanced process control made easy in Julia*. https://doi.org/10.48550/arXiv.2411.09764

Schaller, M., Arnström, D., Bemporad, A., & Boyd, S. (2025, June 21). *Automatic generation of explicit quadratic programming solvers*. https://doi.org/10.48550/arXiv.2506.11513

Schmidtobreick, E. J., Arnström, D., Häusner, P., & Sjölund, J. (2026, May 19). *Warm-starting active-set solvers using graph neural networks*. https://doi.org/10.48550/arXiv.2511.13174

Tøndel, P., Johansen, T. A., & Bemporad, A. (2003). Evaluation of piecewise affine control via binary search tree. *Automatica*, *39*(5), 945–950. https://doi.org/10.1016/S0005-1098(02)00308-4

Verschueren, R., Frison, G., Kouzoupis, D., Frey, J., Duijkeren, N. van, Zanelli, A., Novoselnik, B., Albin, T., Quirynen, R., & Diehl, M. (2022). acados—a modular open-source framework for fast embedded optimal control. *Mathematical Programming Computation*, *14*(1), 147–183. https://doi.org/10.1007/s12532-021-00208-8

Wikner, J., Arnström, D., & Axehill, D. (2026, March 10). *On real-time feasibility of high-rate MPC using an active-set method on nano-quadcopters*. https://doi.org/10.48550/arXiv.2603.09342