Receding Horizon Control: Automatic Generation of High-Speed Solvers

J. Mattingley, Y. Wang, and S. Boyd

IEEE Control Systems Magazine, 31(3):52–65, June 2011.
Shorter version appeared with title *Code Generation for Receding Horizon Control* in *Proceedings IEEE Multi-Conference on Systems and Control*, pages 985–992, Yokohama, Japan, September 2010.

- CSM paper
- Final manuscript
- Proceedings MSC paper

Receding horizon control (RHC), also known as model predictive control (MPC), is a general purpose control scheme that involves repeatedly solving a constrained optimization problem, using predictions of future costs, disturbances, and constraints over a moving time horizon to choose the control action. RHC handles constraints, such as limits on control variables, in a direct and natural way, and generates sophisticated feed-forward actions. The main disadvantage of RHC is that an optimization problem has to be solved at each step, which leads many control engineers to think that it can only be used for systems with slow sampling (say, less than one Hz). Several techniques have recently been developed to get around this problem. In one approach, called explicit MPC, the optimization problem is solved analytically and explicitly, so evaluating the control policy requires only a lookup table search. Another approach, which is our focus here, is to exploit the structure in the optimization problem to solve it efficiently. This approach has previously been applied in several specific cases, using custom, hand written code. However, this requires significant development time, and specialist knowledge of optimization and numerical algorithms. Recent developments in convex optimization code generation have made the task much easier and quicker. With code generation, the RHC policy is specified in a high-level language, then automatically transformed into source code for a custom solver. The custom solver is typically orders of magnitude faster than a generic solver, solving in milliseconds or microseconds on standard processors, making it possible to use RHC policies at kilohertz rates. In this paper we demonstrate code generation with four simple control examples. They show a range of problems that may be handled by RHC. In every case, we show a speedup of several hundred times from generic parser-solvers.