Low-Infrastructure Real-Time Embedded Control via Variational Integrators

Jarvis Schultz[1]* and **Todd Murphey**[1]

[1] Northwestern University, Dept. of Mechanical Engineering, Evanston, IL, USA

This paper presents a discrete time receding horizon control scheme that leverages the numerical properties of a variational integrator to facilitate real-time control generation on an embedded system. The variational integrator employed is well-suited to classical estimation and control algorithms, e.g. LQR, extended Kalman filters, and particle filters. The structure-preserving properties of this variational integrator lead to increased performance of estimation and control routines, especially in low-bandwidth applications. Several experimental examples are presented that illustrate the features of this receding horizon control scheme when leveraging the desirable numerical properties of the variational integrator.

1 Introduction

In a receding horizon control (RHC) framework, finite horizon optimal control problems are repeatedly solved to generate open-loop system trajectories [1]. The open-loop controls from an optimal trajectory are then used to command the system for a small fraction of their total time horizon. Then a new finite horizon problem is solved to generate the controls for the next fraction of time. There are many advantages to this control approach including its ability to handle state or actuator constraints, and its ability to respond to unplanned disturbances [2]. One disadvantage to this approach is the requirement that an optimization problem must be solved at each timestep which, due to computational limitations, restricts applicability.

In this work, we utilize a discrete time extension to a nonlinear, projection based optimization routine, originally described in [3]. This optimization technique is well-suited to RHC due to the projection operator used for projecting dynamically infeasible system trajectories back to the feasible set. Even if the optimization lacks sufficient computation time to achieve a desired convergence tolerance, as long as there is time to take a single step a valid set of controls is produced, and these controls are guaranteed to lower the cost.

The projection-based optimization technique's performance is further improved by employing recent advances from the variational integration community. Variational integrators (VIs) provide stable long-term energy behavior, they perfectly conserve symmetries of the system, they guarantee satisfaction of holonomic constraints, and they can accurately predict statistics of stochastic systems [4, 5]. Recent work by the authors has investigated how these desirable numerical properties affect the performance of standard control and estimation routines in real-world embedded systems [6]. Additionally in [7], the authors presented a VI amenable to use in standard control and estimation routines as well as the first and second order linearizations of this VI. This VI is employed for all discrete system representations in this work.

2 Discrete Projection-Based Optimal Control

The optimization algorithm used in this work is an iterative procedure where each iteration can be broken down into three steps [3]. First is building a *projection operator*, second is finding a descent direction, and third is performing a line search to satisfy a sufficient decrease condition. We begin by defining $\mathcal{T}$ as the set of admissible trajectories for a dynamic system. The trajectory space is embedded in an inner product space V so that $\mathcal{T} \subseteq V$. In discrete time, $\mathcal{T} = \{\xi = (x,u) \in V : x(k+1) = f(x(k), u(k), k) \;\forall\; k = 0, \ldots, N-1\}$ where N is the number of timesteps in the time horizon of interest. We use $\xi = (x, u)$ to denote elements of $\mathcal{T}$, and $\delta\xi = (\delta x, \delta u)$ to denote elements of the tangent space $T\mathcal{T}$, i.e., elements that obey linearized dynamics. Elements of V use over-bars to indicate they do not obey the system dynamics e.g., $\bar{\xi} = (\bar{x}, \bar{u}) \in V$ while elements in the tangent space TV use the notation $\delta\bar{\xi} = (\delta\bar{x}, \delta\bar{u})$.

Given an initial trajectory $\xi_0 = (x_0, u_0)$ we are trying to find

$$\xi^* = \operatorname*{argmin}_{\xi \in \mathcal{T}} J(\xi) \quad \text{where} \quad J(\xi) = \sum_{k=0}^{N-1} \ell(x(k), u(k), k) + m_c(x(N)).$$

This is a constrained optimization problems because the optimizer ξ^* must satisfy the system dynamics, i.e., $\xi^* \in \mathcal{T}$. This is the motivation for introducing the projection operator $\mathcal{P}$ as the mapping $\mathcal{P} : \bar{\xi} \mapsto \xi$. The projection operator takes

* Corresponding author: e-mail jschultz@northwestern.edu

infeasible trajectories in V and maps them to nearby feasible trajectories in $\mathcal{T}$. Presuming that this operator exists, it allows the transformation from a constrained optimization to an equivalent unconstrained optimization

$$\xi^* = \underset{\xi \in \mathcal{T}}{\operatorname{argmin}}\, J(\xi) \quad \Longleftrightarrow \quad \xi^* = \underset{\bar{\xi} \in V}{\operatorname{argmin}}\, J\left(\mathcal{P}\left(\bar{\xi}\right)\right).$$

In practice, the projection operator is found by solving a linear-quadratic regulator (LQR) problem for the system linearized about the current iterate, ξ_i to obtain a stabilizing feedback law [8].

In an iterative trajectory optimization routine, given a dynamically feasible iterate, ξ, one must find a descent direction $\delta\xi$. In standard optimization methods, this descent direction is typically found by minimizing a local quadratic approximation of the cost, and the present technique is no different. The optimization for determining a descent direction is given by

$$\delta\xi^* = \underset{\delta\xi \in T_{\xi_i}\mathcal{T}}{\operatorname{argmin}}\; 2DJ(\xi_i) \circ \delta\xi + q(\xi) \circ (\delta\xi, \delta\xi)$$

where, as in standard optimizations, the bilinear operator $q(\xi) \circ (\delta\xi, \delta\xi)$ can be chosen to provide different descent algorithms. For example, choosing $q(\xi) \circ (\delta\xi, \delta\xi) = \langle \delta\xi, \delta\xi \rangle$ yields the steepest descent method, choosing $q(\xi) \circ (\delta\xi, \delta\xi) = D^2 J(\xi_i) \circ (\delta\xi, \delta\xi)$ yields a quasi-Newton method, and $q(\xi) \circ (\delta\xi, \delta\xi) = D^2 J(\mathcal{P}(\xi_i)) \circ (\delta\xi, \delta\xi)$ yields Newton's method where $\langle \cdot, \cdot \rangle$ represents an inner product.

2.1 Receding Horizon Formulation

To utilize the optimization routine described in the previous section in a receding horizon control framework, we simply have to define the cost function over the discrete time horizon that will be optimized at each timestep. We begin by defining N to be the number of timesteps in the window. Thus, at timestep k the reference trajectory for the receding optimization is given by $\bar{\xi}_{ref,\,k} = (x_{ref,\,k}, u_{ref,\,k}) = \left(\{x_{ref}(i)\}_{i=k}^{k+N}, \{u_{ref}(i)\}_{i=k}^{k+N-1}\right) \in V$ over the horizon $t_{ref,\,k} = \{t_{ref}(i) = i\Delta t \mid i = k, \ldots, k+N\}$. The optimization problem statement at time k is then

$$\xi_k^* = \underset{\xi_k \in \mathcal{T}}{\operatorname{argmin}}\, J(\xi_k, k) \quad \text{where} \quad J(\xi_k) = \sum_{i=k}^{k+N-1} l(x(i), u(i), i) + m_c(x(k+N)). \tag{1}$$

The running cost Lagrangian $l(\cdot)$ and the terminal cost $m_c(\cdot)$ are given by

$$l\left(x(k), u(k), k\right) = \frac{1}{2}\left\|x(k) - x_{ref}(k)\right\|_{Q_J}^2 + \frac{1}{2}\left\|u(k) - u_{ref}(k)\right\|_{R_J}^2 \quad \text{and} \quad m_c(x(k)) = \frac{1}{2}\left\|x(k) - x_{ref}(k)\right\|_{P_J}^2$$

where Q_J, R_J, and P_J are positive semidefinite, symmetric weighting matrices. One important thing to note about this framework is that for a given control frequency, $\nu_{control}$, we allow the optimization to run for up to $\Delta t = 1/\nu_{control}$ seconds. Since this computation takes a finite amount of time, the horizons for the optimizations are actually one timestep ahead of real-world time. This way the optimization will have completed by the time its result is needed.

3 Case Study

In this section we present results illustrating the benefits of using the VI from [7] along with the RHC algorithm from the previous section. The system we utilize in this RHC case study is a planar crane system. While this system is low-dimensional, it is nonlinear and underactuated — simple linear controllers should not be expected to work well. Further, this system has received extensive study in literature as it has many real-world applications. In our model, we treat the horizontal and vertical position of the payload (x, y) as dynamic configuration variables and the horizontal position of the robot and the length of the string (x_r, r) as kinematic configuration variables. More about the modeling of this system can be read in [6, 9, 10]. The robotic system consists of a single magnetically-suspended vehicle equipped with a winch system for controlling the string length. A Microsoft Kinect® is used for measuring the dynamic and kinematic configuration variables, and the Robot Operating System (ROS) is used for measurement processing, control calculations, and data recording. [1]

3.1 Feasible Computation Time

The performance of an RHC is often governed by the amount of time that may be considered in a single optimization while still allowing convergence with the constrained computation time. In this section we consider computational feasibility for the planar crane system using the RHC controller of the previous section and the VI from [7] as the discrete system representation. At lower frequencies more computation time is available for solving the RHC optimization, and if the structure-preserving

[1] Code for controlling this system is available at `github.com/jarvisschultz/receding_planar_sys`.

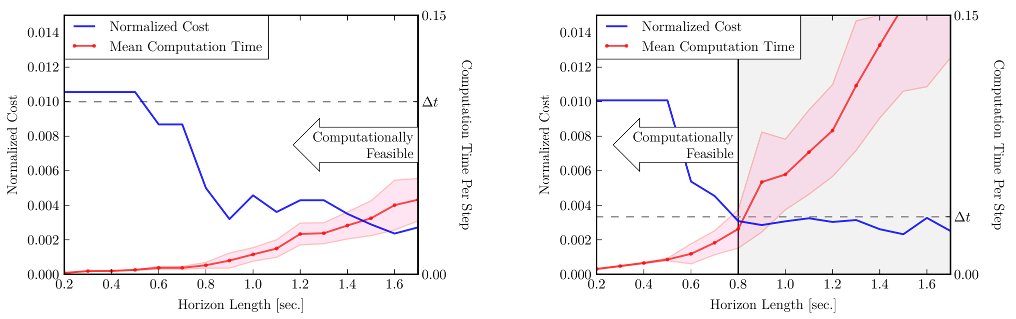

Fig. 1: RHC performance and computation time vs. receding horizon length at 10 Hz (left) and 30 Hz (right).

Table 1: Experimental normed cost for 5 controller variants at 3 frequencies.

Controller	6 Hz	10 Hz	30 Hz
Inverse Kinematics	0.2973	0.2786	0.2870
LQR	0.1909	0.1705	0.1512
Full Horizon no Feedback	0.0177	0.0102	0.0091
Full Horizon w/ Feedback	0.0174	0.0064	0.0068
RHC ($N = 15$)	0.0073	0.0097	0.2150

nature of the VI allows control and estimation routines to perform well as the frequency is decreased one could expect increased performance at lower frequency.

Figure 1 shows simulated results of the RHC algorithm at both 10 Hz and 30 Hz. The blue curves show normalized cost as a function of horizon length[2]. Each point on the plots of Fig. 1 is generated by simulating the system using the RHC algorithm with the number of timesteps in each window, N, corresponding to a specified horizon length. The RHC algorithm is run every Δt seconds throughout the total time horizon of t_{per}. This produces a complete system trajectory $\xi = (x, u) = \left(\{x(i)\}_{i=0}^{M}, \{u(i)\}_{i=0}^{M-1}\right)$ where $M = \frac{t_{per}}{\Delta t}$. Each step of the RHC is allowed to fully converge to a prescribed tolerance regardless of computation time, but the computation time to achieve convergence for each optimization is stored. When the trajectory is complete, the mean and standard deviation of the computation times are calculated to produce the computation time per step and the plus-minus one standard deviation shaded error bounds shown in Fig. 1. All trajectories and their corresponding normalized costs shown in Fig. 1 were calculated using the same values for Q_J, R_J, and P_J.

For both frequencies the normalized cost behavior is similar, and the performance increases dramatically when the horizon length is greater than ~0.8 seconds. On both plots in Fig. 1, the timestep is indicated with a gray horizontal dashed line. For a given horizon length, if computation time per step is greater than Δt, then that horizon length is infeasible to implement in practice. For this system we achieve similar performance at 10 Hz, while also providing a significantly higher margin of safety for computational feasibility. This is at least partially attributed to the performance of the VI at low frequency.

3.2 Comparison with Standard Controllers

In this section, several different control schemes are used to stabilize the planar crane embedded system for a parametrically-defined reference trajectory. The time evolution of the payload's configuration is set to have the payload move back and forth in the horizontal direction, the robot's reference horizontal position is directly above the payload, and the reference string length is the vertical distance from the payload to the robot. The reference trajectory for all momentum terms is zero, producing a dynamically infeasible reference.

In order to quantify the performance of the RHC algorithm on the embedded system, five different controllers were run at three different frequencies for a total of 15 experimental trials. For each trial, the normed cost was computed as a single metric on controller performance. The results of these experiments are detailed in Table 1 and Fig. 2. The "Inverse Kinematics" controller calculates values for the robot's kinematic inputs (horizontal position and string length) by solving the inverse kinematics from the reference trajectory. For the "LQR" controller, the system is linearized about its stable equilibrium, then an infinite horizon LQR problem [8] is solved to generate an optimal, stabilizing controller. The "Full Horizon no Feedback" controller uses the discrete projection-based optimal control routine, described in Section 2, to pre-compute an optimal trajectory for the whole 10 second time horizon; the inputs from this trajectory are then sent to the system open-loop. The "Full Horizon w/ Feedback" controller additionally adds a stabilizing feedback law to the full horizon optimal trajectory. The "RHC ($N = 15$)" controller is implemented using the algorithm from Section 2.1 with $N = 15$. At both 6 Hz and 10 Hz the RHC controller performs nearly as well as both full horizon controllers while avoiding pre-computation. The "LQR" and

[2] The total trajectory cost is normalized to account for the fact that there are a non-constant number of points in a complete trajectory, depending on the value of Δt.

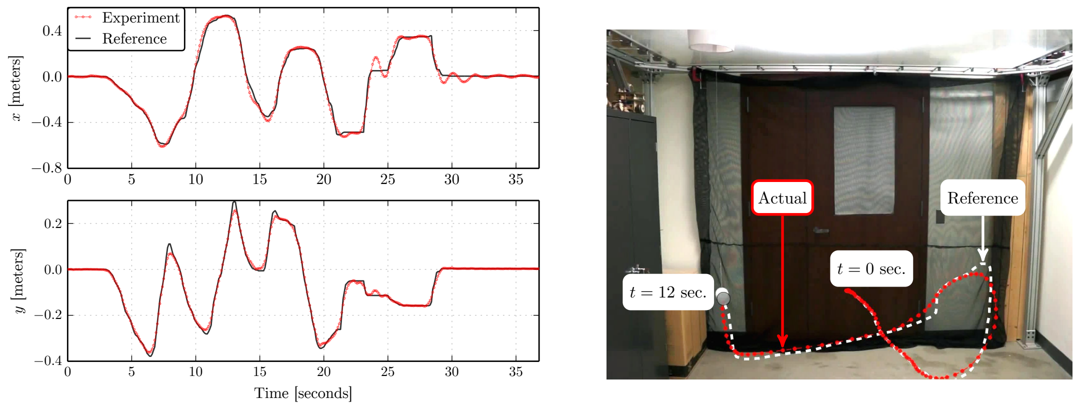

Fig. 3: Example experimental trial using the real-time reference generation capabilities. On left, the time evolution of the x-position (top) and y-position (bottom) of the payload is shown, and on the right the first 12 seconds of experimental data is overlaid on an image of the experiment running.

"Inverse Kinematics" controllers have significantly worse performance than the other variants illustrating the difficulty of this control problem[3].

3.3 Real-Time Reference Generation

To demonstrate the real-time nature of this control scheme, we created an interface that allows a user to specify the desired trajectory of the payload using a computer mouse as the experiment is executed. As this reference is generated, the RHC algorithm is used to generate controls for the system on-the-fly. Using this interface and control strategy, high reliability has been demonstrated in both simulation and experiment. A single experimental trial illustrating the experimental performance can be seen in Fig. 3 [4].

4 Conclusions

The optimization routine presented herein is a receding horizon discrete time optimization algorithm that is based on the continuous time optimization originally presented in [3]. Inspired by our recent work illustrating reliable estimation and control at low frequency when using VIs [6,7], our implementation of this algorithm employs a variational integrator as its underlying discrete time system representation. Future work will be focused on investigating which numerical properties of the VI are the most valuable to this control scheme.

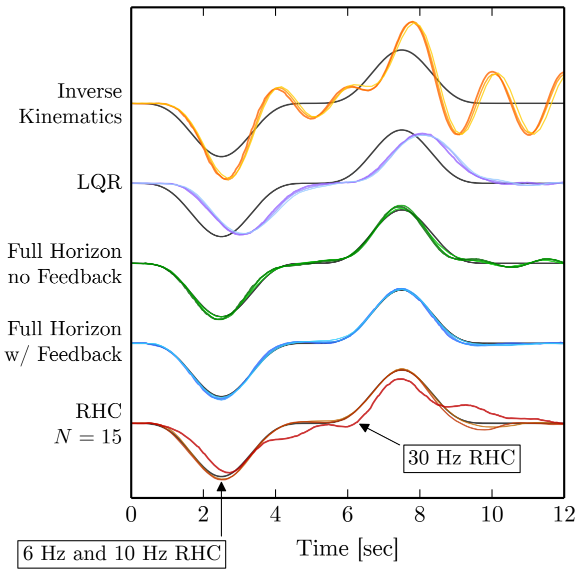

Fig. 2: Time evolution of the payload x-position for all 15 controllers in Table 1.

References

[1] J. Mattingley, Y. Wang, and S. Boyd, IEEE Control Systems **31**(3), 52–65 (2011).
[2] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert, Automatica **36**(6), 789–814 (2000).
[3] J. Hauser, A projection operator approach to the optimization of trajectory functionals, in: IFAC World Congress, (July 2002).
[4] N. Bou-Rabee and H. Owhadi, IMA J. Numerical Anal. **48**(2), 421–443 (2009).
[5] J. E. Marsen and M. West, Acta Numerica **10**, 357–514 (2001).
[6] J. Schultz and T. D. Murphey, Extending filter performance through structured integration, in: American Controls Conf. (ACC), (June 2014), pp. 430–436.
[7] E. Johnson, J. Schultz, and T. Murphey, IEEE Trans. on Automation Sci. and Eng. **12**(1), 140–152 (2014).
[8] B. D. O. Anderson and J. B. Moore, Optimal Control: Linear Quadratic Methods (Dover Publications, February 2007).
[9] E. R. Johnson and T. D. Murphey, IEEE Trans. on Robotics **25**(October), 1249–1261 (2009).
[10] J. Schultz and T. Murphey, Trajectory generation for underactuated control of a suspended mass, in: IEEE Int. Conf. on Robotics and Automation (ICRA), (May 2012), pp. 123–129.

[3] The poor performance of the 30 Hz RHC is due to lack of computation time; there were many timesteps where the RHC optimization had to be abandoned before convergence. This behavior is in agreement with the predictions of Fig. 1

[4] This setup has been run hundreds of times, and has never experienced an instability.