# Real-Time Trajectory Generation for a Planar Crane using Discrete Mechanics

Jarvis Schultz and Todd D. Murphey

***Abstract*— We present a receding horizon control scheme that utilizes a discrete time version of a nonlinear, iterative, projection-based optimization routine. The routine utilizes a *projection operator* to ensure feasible system trajectories at each iteration of the optimization resulting in valid control signals even when constraints on computation time prevent full convergence. An additional feature of this work is that the underlying discrete time system representation is provided by a discrete mechanics derived variational integrator as opposed to, e.g., a Runge-Kutta based integrator. Recent work has described a midpoint variational integrator that provides a causal, one-step map that admits both first and second order linearizations facilitating use in standard discrete time control and estimation techniques such as linear quadratic regulators and extended Kalman filters. In this work, this variational integrator is combined with the aforementioned receding horizon optimization routine to generate feasible system trajectories for real-time control of an experimental planar crane system.**

## I. Introduction

In a general 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. For practical implementation, this traditionally has restricted RHC to systems with slow dynamics, such as industrial processes, or systems with special structure simplifying the optimization problem [3]. Recently, much work has focused on ways to increase the efficiency of this optimization [4].

In this work, we present a discrete time extension to a nonlinear, projection based optimization routine, originally described in [5], that is easily adapted to a receding horizon control framework. This optimization technique is particularly well-suited to a RHC framework due to the projection operator used in the optimization. In this iterative optimization procedure, the system state $x$ and the inputs to the system $u$ are both varied when determining a descent direction. This descent direction added to the current optimization iterate produces dynamically infeasible trajectories. The projection operator then maps these infeasible trajectories to the set of feasible trajectories. So for a given RHC optimization even if the optimization doesn't have sufficient computation time to converge to a prescribed tolerance, as long as there is time to take a single step, a valid set of controls is produced. In other constrained optimization techniques, such as penalty function methods, this is not the case [6], [7]. In these techniques if the optimization fails to converge in the allotted time there is only an infeasible set of controls to send to the system.

This optimization technique's performance is further improved by employing recent advances from the discrete mechanics community. In discrete mechanics, one utilizes approximations of a physical principle, e.g. Hamilton's principle, to directly derive a discrete time map, called a variational integrators (VI), that provides a numerical approximation to a system's state evolution. Traditionally, this map is arrived at by first considering a continuous time model of the system and then building the discrete approximation by employing a numerical integration techniques. Much of the work in the discrete mechanics community has focused on studying the numerical benefits of these variational integrators. Variational integrators 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 [8]–[10].

Recent work by the authors has begun to investigate how the desirable numerical properties of variational integrators affect the performance of standard control and estimation routines in real-world embedded systems [11], [12]. Additionally in [13], the authors presented a particular VI amenable to use in standard control and estimation routines as well as the first and second order linearizations of this VI. Any discrete time system representation must admit these linearizations for the optimization routine presented in this work. The primary message of [11], [12] is that the numerical properties of the VI leads to more reliable controllers and estimators at low frequencies when compared to the low-order Runge-Kutta methods typically used in estimation and control. Thus, in a situation where one needs to run an RHC at low frequency to provide sufficient computation time, the integrator itself can help compensate for some of the disadvantages inherent to the low frequency. If one can lower the frequency and still have well-behaved estimators and controllers, the lowered frequency can also provide a

J. Schultz jschultz@northwestern.edu
T.D. Murphey t-murphey@northwestern
Department of Mechanical Engineering, Northwestern University, Evanston, IL

secondary advantage to the RHC. In a discrete time optimization the computation time required to solve the optimization is fundamentally tied to the number of timesteps involved. In many situations, the performance of the RHC is tied to the time-length of the optimization window [14]. So at lower frequencies, not only does one have greater computation time, but for a fixed dimension of optimization problem, the total time horizon that can be optimized is also greater which can lead to a further increase in performance. Results illustrating this will be shown in Section IV.

In Section II, an overview of discrete mechanics and the particular VI presented in [13] is provided. Section III follows with a description of the optimization routine employed in this work and its receding horizon formulation. Finally Section IV describes an experimental planar crane embedded system and presents results utilizing the present receding horizon control scheme to stabilize this system to reference trajectories generated on-the-fly.

## II. Discrete Mechanics

In the discrete mechanics framework, one attempts to find a sequence $\{(t_0, q_0), (t_1, q_1), \ldots, (t_n, q_n)\}$ that approximates a continuous time trajectory of a system — i.e., $q_k \approx q(t_k)$ where $t$ is time, and $q \in Q$, the configuration space of the system. In traditional variational mechanics Hamilton's principle is used to derive governing differential equations for a system, and in practice, numerical integration techniques must be used to approximate their solutions. In discrete mechanics, approximations are instead applied to the physical principle, and variational methods produce governing difference equations. To derive a VI, we begin by approximating a system's Lagrangian using an arbitrary quadrature rule over a timestep $\Delta t = t_{k+1} - t_k$

$$L_d(q_k, q_{k+1}) \approx \int_{t_k}^{t_{k+1}} L(q(\tau), \dot{q}(\tau)) d\tau. \tag{1}$$

This quantity is referred to as the discrete Lagrangian. Next the action integral is approximated with an action sum as

$$S(q[t_0, t_f]) = \int_{t_0}^{t_f} L(q(\tau), \dot{q}(\tau)) d\tau \approx \sum_{k=0}^{n-1} L_d(q_k, q_{k+1}). \tag{2}$$

Hamilton's principle states that the evolution of a mechanical system is a stationary point in the action. Taking the first variation of Eq. (2), and invoking the fundamental lemma of the calculus of variations [15], one can derive the unforced, unconstrained Discrete Euler-Lagrange (DEL) equations[1]

$$D_1 L_d(q_k, q_{k+1}) + D_2 L_d(q_{k-1}, q_k) = 0. \tag{3}$$

In this form the DEL equations provide a discrete map $(q_{k-1}, q_k) \rightarrow (q_k, q_{k+1})$ that is implicitly solved using a numerical root-finding algorithm [9]. The precise quadrature rule chosen to define the discrete Lagrangian in Eq. (1) will determine the order of the integrator derived and whether it is an explicit or an implicit method. Any quadrature rule used to define a discrete Lagrangian produces a corresponding set of DEL equations, and in-turn a corresponding VI. Following [13], we use the midpoint rule to define a VI yielding the following discrete Lagrangian

$$L_d(q_k, q_{k+1}) = L\left(\frac{q_k + q_{k+1}}{2}, \frac{q_{k+1} - q_k}{\Delta t}\right) \Delta t. \tag{4}$$

Differentiating Eq. (4) with respect to its arguments allows computation of all terms in Eq. (3) in terms of the original continuous Lagrangian. Using the midpoint rule to define the discrete Lagrangian produces a second-order VI [16].

In the given form, the DEL equations provide a two-step map from $Q \times Q \rightarrow Q \times Q$. To express Eq. (3) as a one-step map as required by many standard control and estimation routines, we invoke the discrete Legendre transform to define the discrete generalized momentum as

$$p_k = -D_1 L_d(q_k, q_{k+1}) = D_2 L_d(q_{k-1}, q_k). \tag{5}$$

Eq. (3) can now be viewed as the following set of equations

$$p_k = -D_1 L_d(q_k, q_{k+1}) \tag{6a}$$

$$p_{k+1} = D_2 L_d(q_k, q_{k+1}) \tag{6b}$$

To integrate this form of the DEL equations, we start with the given pair $(q_k, p_k)$, and then implicitly solve Eq. (6a) to obtain $q_{k+1}$. Equation (6b) then explicitly yields $p_{k+1}$. The DEL equations are now the one-step map $T^*Q \rightarrow T^*Q$ [9].

VIs are easily extended to incorporate holonomic constraints and non-conservative forcing [9]. See [13] for details on how these characteristics are incorporated into midpoint VI just discussed.

## III. 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 [5]. First is building a *projection operator*, second is finding a descent direction, and third is performing a line search [6] 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

$$\xi^* = \underset{\xi \in \mathcal{T}}{\operatorname{argmin}} \, J(\xi)$$

$$\text{where } J(\xi) = \sum_{k=0}^{N-1} \ell(x(k), u(k), k) + m(x(N)).$$

[1] Here we have used the *slot derivative* notation where $D_i f(\cdot)$ represents a derivative of $f$ with respect to its $i$th argument.

This is a constrained optimization problems because the optimizer $\xi^*$ 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 $\mathcal{P} : \bar{\xi} \mapsto \xi$. I.e., the projection operator takes 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

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

In practice, the projection operator is found by solving an LQR problem for the system linearized about the current iterate to obtain a stabilizing feedback law [15], [17].

In an iterative trajectory optimization routine, given a dynamically feasible iterate, $\xi$, one must find a descent direction. 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

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

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.

### A. 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

$$\begin{aligned} \xi_k^* &= \underset{\xi_k \in \mathcal{T}}{\operatorname{argmin}} J(\xi_k, k) \\ \text{where } J(\xi_k) &= \sum_{i=k}^{k+N-1} l(x(i), u(i), i) \\ &\quad + m(x(k+N)). \end{aligned} \tag{7}$$

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

$$\begin{aligned} l\left(x(k), u(k), k\right) = \frac{1}{2}(x(k) - x_{ref}(k))^{\mathrm{T}} Q (x(k) - x_{ref}(k)) + \\ \frac{1}{2}(u(k) - u_{ref}(k))^{\mathrm{T}} R (u(k) - u_{ref}(k)) \end{aligned}$$

and

$$m(x(N)) = \frac{1}{2}(x(N) - x_{ref}(N))^{\mathrm{T}} P_1 (x(N) - x_{ref}(N))$$

where $Q$, $R$, and $P_1$ are positive semidefinite, symmetric weighting matrices. One important thing to note about this framework is that for a given control frequency, $f_{control}$, we allow the optimization to run for up to $\Delta t = 1/f_{control}$ seconds. Since this computation takes a finite amount of time, the horizon 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. This detail is expressed in Algorithm 1. The trajectory optimization performed in step 10 of Algorithm 1 is expressed algorithmically in Algorithm 2

**Algorithm 1** RHC Algorithm

1: Initialize: $u_{key} = u_{prev} = 0$, $k = 0$, $x(k) = x_0$
2: **loop**
3: Increment $k$
4: Send command for current time, $u_{key}$, to system
5: $z(k) \leftarrow$ measurement
6: $x(k) \leftarrow$ estimate of current state
7: $x_{pred} \leftarrow$ predict state at $k+1$ using dynamics
8: $\bar{\xi}_{ref} \leftarrow$ get reference trajectory for $[k+1, k+1+N]$
9: $\xi_0 \leftarrow$ calculate initial iterate for optimization
10: $\xi^* \leftarrow$ trajectory optimization
11: Store results: $u_{prev} = u_{key}$, $u_{key} = u^*(0)$
12: **end loop**

**Algorithm 2** Trajectory Optimization

1: Given: initial iterate $\xi_0$, reference trajectory $\bar{\xi}_{ref,\,k}$, tolerance ($\epsilon$)
2: $\xi_i \leftarrow$ initial iterate $\xi_0$
3: $i \leftarrow 0$
4: **loop**
5: $\mathcal{P} \leftarrow$ build projection around current iterate $\xi_i$
6: $\delta\xi_i \leftarrow$ calculate descent direction
7: **if** $|\delta\xi_i| < \epsilon$ **or** ELAPSED TIME $> \Delta t$ **then**
8: **return** $\xi_i$
9: **end if**
10: $\gamma_i \leftarrow$ line search to satisfy sufficient decrease
11: $\xi_i \leftarrow$ step and project $\mathcal{P}(\xi_i + \gamma_i \delta\xi_i)$
12: **end loop**

## IV. Results

### A. Planar Crane System

The system we are interested in controlling is a planar crane as shown in Fig. 1. We treat the horizontal and vertical position of the mass $(x, y)$ as dynamic configuration variables and the horizontal position of the robot and the length of the string $(x_r, r)$ as kinematic configuration variables. These kinematic configurations require the assumption that the actuators are powerful relative to the inertias they

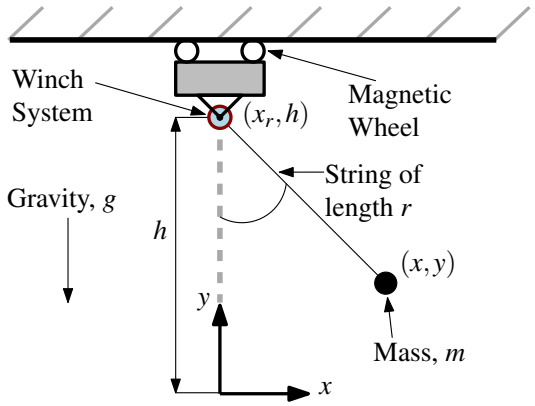
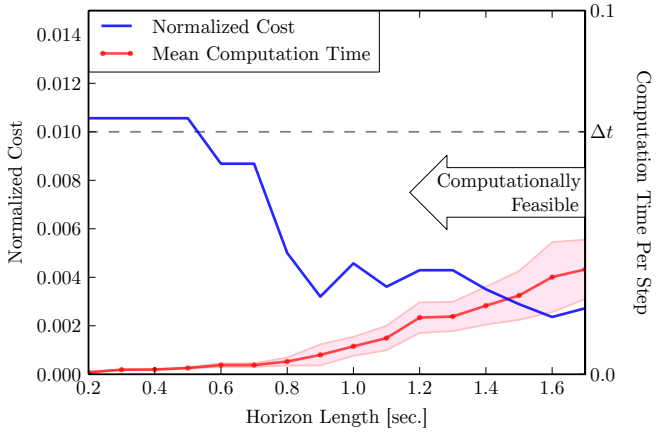
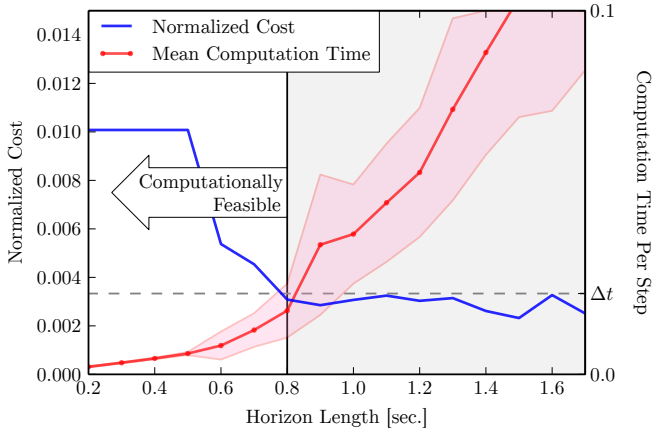



Fig. 1: Schematic of planar crane system including relevant geometric parameters.




(a) 10 Hz




(b) 30 Hz

Fig. 2: RHC performance and computation for an arbitrary reference trajectory as a function of receding horizon length.

are controlling, and that their dynamics can effectively be neglected. This results in a dynamic model that uses configurations of the kinematic variables as the inputs instead of generalized forces. The robotic system used for this work was designed with this modeling paradigm in mind, and it is capable of accurately tracking the kinematic trajectories we are interested in. More about the modeling of this system can be read in [11], [12], [18], [19].

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 [20], [21]. All code for controlling this system is available at `https://github.com/jarvisschultz/receding_planar_sys`.

### B. Feasible Computation Time

In Section I it was mentioned that the performance of an RHC was often governed by the total amount of time that could be considered in the optimization and still converge given one timestep of computation time. In other words, for a fixed-complexity optimization containing $N$ timesteps, at lower frequencies the amount of actual time contained in the window is greater, and the RHC performance may actually increase. This idea is illustrated in Fig. 2 which 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. Each point on the plots in Fig. 2 is generated by first simulating the system using the RHC algorithm with the appropriate $N$ for the specified horizon length for a 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, and then when the trajectory is complete, the mean and standard deviation of the computation times is calculated to produce the computation time per step and the shaded error bounds in the figures. Once the trajectory is complete, the normalized cost is computed using

$$J_{norm}(\xi) = \frac{1}{M} \sum_{i=0}^{M-1} l(x(i), u(i), i) + m(x(M)) \qquad (8)$$

where $l(\cdot)$ and $m(\cdot)$ are unchanged from their definitions in Section III-A and $Q$, $R$, and $P_1$ are the same values used in each step of the RHC. All trajectories and their corresponding normalized costs shown in Fig. 2 were calculated using the same weights.

There are several important things to note about Fig. 2. For both the 10 Hz and 30 Hz trajectories, the normalized cost drops rapidly as the horizon length is increased past a threshold of ~0.6 seconds. Then beyond ~1.0 seconds, further horizon increases are met with diminishing performance gains at the expense of much higher computational costs. This is unsurprising as it has been shown in [22] that for discrete systems, there exists some finite horizon length for which a RHC considering a longer horizon will be stabilizing. This suggests that for a given set of control weights, and a given reference trajectory there is an optimal horizon length that balances performance and computational

Fig. 3: An image of a user controlling the interactive, receding horizon controlled suspended mass system.

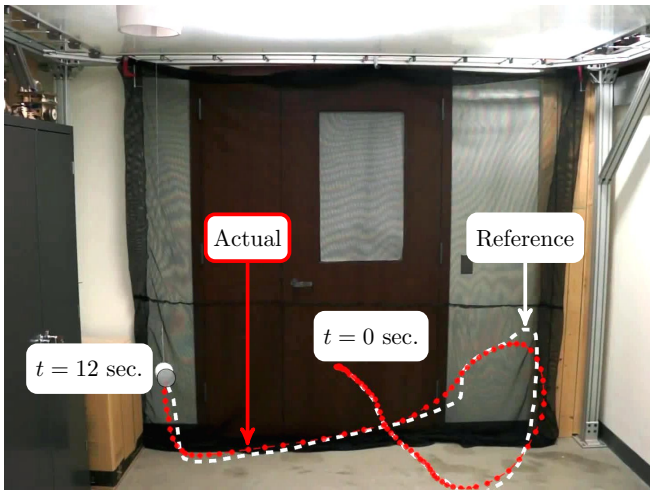



Fig. 4: Still image of the interactive receding horizon experimental system in motion with suspended mass reference (white dashed) and measured (red dotted) trajectory overlays.

cost. On both plots in Fig. 2, the timestep is indicated with a gray horizontal dashed line. For a given horizon length if the computation time per step is greater than the timestep, then there is not sufficient time to complete the optimizations, and that horizon length is not feasible to implement on the planar crane embedded system. This is why only the left half of the 30 Hz plot is marked as *Computationally Feasible*. In the other region, which is filled lightly in gray, the mean plus one standard deviation of the per-step computation time is greater than the timestep, and thus, horizon lengths in this region are infeasible to implement. Note that the 10 Hz controller achieves very similar performance to the 30 Hz controller with a much higher safety factor on the computational time constraints. In the real embedded system, total computational burden is increased through the processing of measurement data and the evaluation of estimators. Moreover, in the real system sensor noise is introduced, unreliable wireless communication links are relied on, and actuator limits exist. These factors generally produce optimizations that require more steps to converge, and each step to take slightly longer. As a result the safety factor is important. Figure 2 indicates that at 30 Hz it should be possible to achieve reliable convergence as long as the horizon length is shorter than ~0.7 seconds ($N \approx 21$). However experiments with the embedded system reveal that the limit on horizon length is closer to 0.5 seconds. Additionally, the value of this threshold is sensitive to the particular choice of cost function weights.

It is evident in Fig. 2 that with a horizon length of 0.5 seconds the performance at both 30 Hz and 10 Hz is lower than with a window length of 1.0 seconds, but the longer window length is only computationally feasible at 10 Hz. This suggests one reason why one might expect higher performance from the 10 Hz RHC. The 10 Hz RHC not only has a higher safety factor on computation limits, but the standard deviation of the computation time is lower, this likely results in increased robustness to chosen parameters.

### C. Real-Time Reference Generation

To demonstrate the real-time nature of this control scheme, an interface was developed that allows a user to specify the desired trajectory of the suspended mass using a computer mouse. As this reference is generated, the RHC algorithm is used to generate controls for the robot. As the RHC requires the reference trajectory to be defined for a future window length $w = (N)(\Delta t) = (\text{steps into the future})(\text{controller period})$, the embedded system operates with a $w$-second time delay from the reference being generated by the user. An image of a user controlling the system can be seen in Fig. 3.

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 Figs. 4 and 5. Note that the overlay shown in Fig. 4 is the same data as Fig. 5, but it is cut off at 12 seconds to keep the image from being too cluttered.

## V. Conclusions

The optimization routine presented herein is a discrete time extension to the continuous time optimization originally presented in [5]. This algorithm is also amenable to use in a receding horizon control scheme. For receding horizon control, one of the biggest features of the algorithm is the use of a projection operator that provides feasible system trajectories even if the optimization cannot fully converge. Inspired by our recent work involving estimation and control using variational integrators [11]–[13], our implementation of this algorithm employed a variational integrator as its underlying discrete time system representation. The low-frequency performance of estimators and controllers employing this variational integrator representation combined lead us to hypothesize that the receding control paradigm may actually achieve better performance at lower frequencies. In

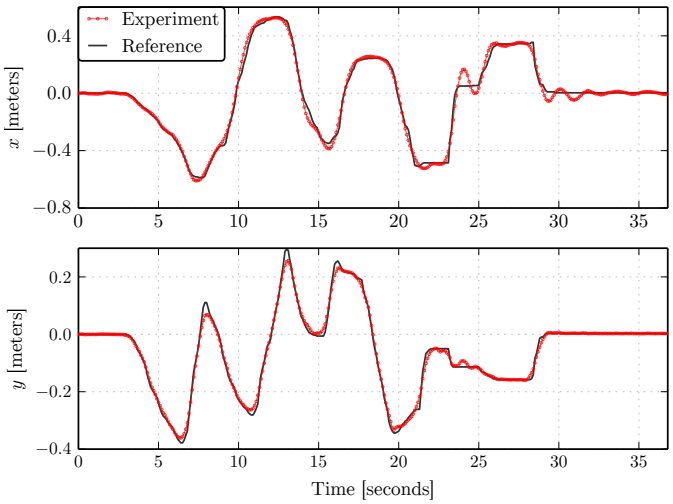



Fig. 5: Example experimental time evolution of the $x$-position (top) and $y$-position (bottom) of the suspended mass controlled by the interactive receding horizon controller.

Section IV-B we presented simulations that validated this claim.

Using this receding horizon control scheme with the variational integrator, we successfully controlled a planar crane embedded system in experiment. The experiment has run successfully hundreds of times using both interactively-generated and pre-defined reference trajectories, and has demonstrated remarkably reliable performance. Presently, very little work has been done on investigating any sort of stability guarantees, but the experimental reliability provides some evidence that these investigations would be worthwhile. Our previous work leads us to believe that the numerical properties of the variational integrator is having a significant impact on the performance of this control scheme, especially as the control frequency is lowered. However, at this point, it is not well understood exactly *which* numerical properties are the most valuable to this control scheme. Investigating the effects of particular integrators on the optimization routine itself would be another valuable area for future investigations.

## REFERENCES


[1] J. Mattingley, Y. Wang, and S. Boyd, "Receding horizon control," *IEEE Control Systems*, vol. 31, no. 3, pp. 52–65, June 2011.

[2] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, June 2000.

[3] J. Mattingley, Y. Wang, and S. Boyd, "Code generation for receding horizon control," in *2010 IEEE Int. Symposium on Computer-Aided Control System Design (CACSD)*, Sept. 2010, pp. 985–992.

[4] T. Ohtsuka, "A continuation/GMRES method for fast computation of nonlinear receding horizon control," *Automatica*, vol. 40, no. 4, pp. 563–574, Apr. 2004.

[5] J. Hauser, "A projection operator approach to the optimization of trajectory functionals," in *IFAC World Congress*, Barcelona, Spain, July 2002.

[6] C. T. Kelley, *Iterative Methods for Optimization*. Philadelphia, PA: Society for Industrial and Applied Mathematics (SIAM), 1999.

[7] J. Nocedal and S. Wright, *Numerical Optimization*, 2nd ed. New York: Springer, Aug. 2006.

[8] N. Bou-Rabee and H. Owhadi, "Stochastic variational integrators," *IMA J. Numerical Anal.*, vol. 48, no. 2, pp. 421–443, Apr. 2009.

[9] J. E. Marsen and M. West, "Discrete mechanics and variational integrators," *Acta Numerica*, vol. 10, pp. 357–514, 2001.

[10] O. Junge, J. E. Marsden, and S. Ober-Blöbaum, "Discrete mechanics and optimal control," in *In Proc. of the 16th IFAC World Congress*, July 2005.

[11] J. Schultz and T. D. Murphey, "Embedded control synthesis using one-step methods in discrete mechanics," in *American Controls Conf. (ACC)*, Washington, D.C., 2013, pp. 5393–5298.

[12] ——, "Extending filter performance through structured integration," in *American Controls Conf. (ACC)*, Portland, OR, June 2014.

[13] E. Johnson, J. Schultz, and T. Murphey, "Structured linearization of discrete mechanical systems for analysis and optimal control," *IEEE Trans. on Automation Sci. and Eng.*, to be published.

[14] P. Serkies and K. Szabat, "Application of the MPC to the position control of the two-mass drive system," *IEEE Trans. on Industrial Electronics*, vol. 60, no. 9, pp. 3679–3688, Sept. 2013.

[15] D. E. Kirk, *Optimal Control Theory: An Introduction*. Dover Publications, Apr. 2004.

[16] E. Johnson, "Trajectory optimization and regulation for constrained discrete mechanical systems," Ph.D. dissertation, Northwestern University, 2012.

[17] B. D. O. Anderson and J. B. Moore, *Optimal Control: Linear Quadratic Methods*. Dover Publications, Feb. 2007.

[18] E. R. Johnson and T. D. Murphey, "Dynamic modeling and motion planning for marionettes: Rigid bodies articulated by massless strings," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, Roma, Italy, Apr. 2007, pp. 330–335.

[19] 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.

[20] (2011) Robot Operating System. Willow Garage. [Online]. Available: http://www.ros.org

[21] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "ROS: an open-source Robot Operating System," in *IEEE Int. Conf. on Robotics and Automation (ICRA) Workshop on Open Source Robotics*, Kobe, Japan, May 2009.

[22] J. A. Primbs and V. Nevistić, "Feasibility and stability of constrained finite receding horizon control," *Automatica*, vol. 36, no. 7, pp. 965–971, July 2000.