# Dangers of Two-point Holonomic Constraints for Variational Integrators

Elliot R. Johnson and Todd D. Murphey
Mechanical Engineering
Northwestern University
Evanston, IL 60208
elliot.r.johnson@u.northwestern.edu and t-murphey@northwestern.edu

***Abstract*— Variational integrators are powerful tools for simulating constrained mechanical systems as well as computing optimal control strategies. While there are well-established methods to incorporate holonomic constraints in variational integrators, there have also been alternative suggestions that have appealing benefits with respect to computational efficiency. We compare the standard holonomic constraints method with what are called two-point constraint methods. In this paper both methods are tested on two different, relatively simple systems. The results reveal multiple problems with the two-point constraints. They are found to be numerically unstable and to under some circumstances provide unreliable constraint force values. This has implications for implementation of both variational integrators for simulation purposes and control purposes.**

## I. INTRODUCTION

There has recently been a great deal of research in the use of discrete mechanics for numerically simulating mechanical systems [5] and optimal control [4]. A result of this research is a class of integrators called *variational integrators* that propagate states directly from a variational principle rather than numerically integrating a continuous ordinary differential equation. Variational integrators conserve (or nearly conserve, depending on the integrator) important quantities like the symplectic form, momentum, and energy [6]. They are also well suited for problems involving impacts and non-smooth phenomenon [1].

Mechanical simulations often require constraints on the configuration manifold to properly represent a system. These *holonomic constraints* can represent closed kinematic chains, pin joints, sliding joints, etc. Holonomic constraints are different from a *non-holonomic constraints* that act on a system's tangent bundle. Non-holonomic constraints (also called velocity constraints) simulate non-slip phenomenon like a tire that can roll but not slide. Non-holonomic constraints can work in the variational integrator framework but are not discussed in this paper.

Holonomic constraints are added to variational integrators using a straightforward derivation that is analogous to continuous Lagrangian mechanics with constraints. This technique, which we refer to as *one-point* constraints, is well established and has been thoroughly analyzed and tested[8]. [7] suggests a modification to one-point constraints that is appealing from a philosophical and implementational perspective. This technique, which we call a *two-point* constraint, has not been rigorously derived, analyzed, or tested.

This paper presents findings from a thorough comparison of the two methods. Both seem equally reasonable and two-point constraints can be faster for some implementations, but experience has shown that two-point constraints have unaddressed problems. We do not resolve these issues, however, and leave them as open questions.

Section I-A briefly introduces variational integrators. Section II discusses the two techniques to include holonomic constraints. Section III uses simulation of a point constrained to a circle in the plane to demonstrate what can go wrong using the two-point constraint method if one is using a quadrature rule that is not convex in the local coordinate choice. Section IV shows that this is not specific to the prior example by using the example of a point constrained to a line in the plane. We end with conclusions in Section V.

### A. Variational Integrators

In discrete mechanics, we find a sequence $\{(t_0, q_0), (t_1, q_1), \ldots, (t_n, q_n)\}$ that approximates the actual trajectory of a mechanical system ($q_k \approx q(t_k)$). In this paper, use a constant time-step ($t_{k+1} - t_k = \Delta t \ \forall \ k$), but in general the time-step can be varied to use adaptive time-stepping algorithms.

A variational integrator is derived by defining a discrete Lagrangian, $L_d$, that approximates the action integral of a continuous mechanical system over a short time interval.

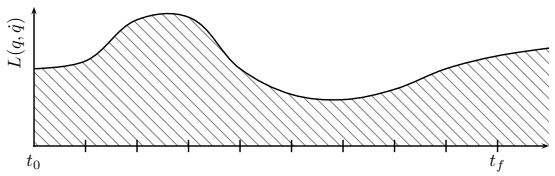
$$L_d\left(q_k, q_{k+1}\right) \approx \int_{t_k}^{t_{k+1}} L(q(\tau), \dot{q}(\tau))\mathrm{d}\tau \tag{1}$$

The discrete Lagrangian allows us to replace the system's action integral with an approximating action sum.

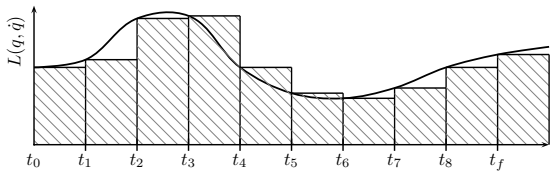
$$S(q([t_0, t_f]) = \int_{t_0}^{t_f} L(q(\tau), \dot{q}(\tau))\mathrm{d}\tau \approx \sum_{k=0}^{n-1} L_d\left(q_k, q_{k+1}\right) \tag{2}$$

where $t_f = t_n$. This approximation is illustrated in Fig. 1. The shaded region in Fig. 1a is the continuous action integral. The shaded boxes in Fig. 1b represent values of the discrete Lagrangian, which are summed to calculate the discrete action.

In continuous mechanics, a variational principle is applied to extremize the action integral and derive the well-known Euler-Lagrange equation. The same approach is used to extremize (2) to get the discrete Euler-Lagrange (DEL)




(a) Continuous Action Integral




(b) Discrete Action Sum

1: The discrete action sum approximates the continuous action integral.

equation[1].

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

This is an implicit difference equation that depends on the previous, current, and future states. Given $q_{k-1}$ and $q_k$, (3) is treated as a root-finding problem to find $q_{k+1}$. After advancing $k$, this process is repeated to simulate the system for as long as desired.

We will use a simple variational integrator derived by applying the generalized midpoint quadrature rule to define the discrete Lagrangian:

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

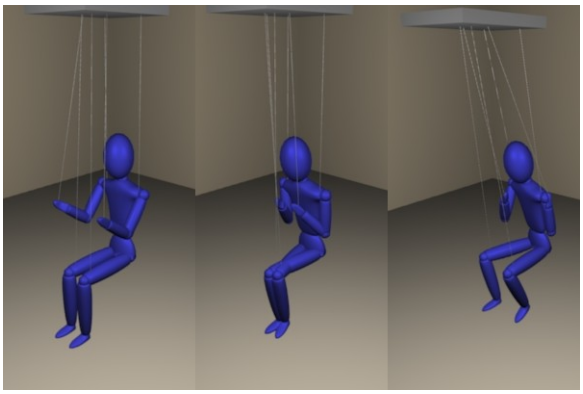
where $\alpha \in [0,1]$ is an algorithm parameter. Choosing $\alpha = 0$ leads to a first order, explicit integrator while $\alpha = 1$ is a first order, implicit integrator. $\alpha = \frac{1}{2}$ leads to a second order integrator [9] and will be our choice throughout this paper.

## II. Holonomic Constraints

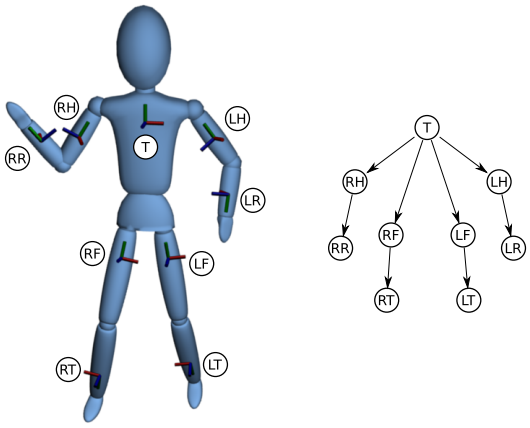
We are interested in simulation and control of high dimensional nonlinear systems. The example that we have been using recently is that of a marionette–see Fig. 2a. The marionette has roughly fifty degrees of freedom and consists of five coupled, closed kinematic chains. Hence, its dynamics are very complex, and it turns out that normal ordinary differential equation solvers tend to be unstable. Variational integrators, however, are stable even with time steps of 0.1 seconds, making them ideally suited for simulation of this type of structure. Efficient evaluation of a variational integrator can take advantage of the graph structure associated with its mechanical topology, as seen in Fig.2b. Details of how to do this may be found in [3] and [2].

We now move on to considering constraints for variational integrators. Holonomic constraints restrict a system to a sub-manifold of its configuration. They are typically defined

[1] $D_n f(\dots)$ is the derivative of $f(\dots)$ with respect to its $n$-th argument. This is sometimes called the *slot derivative[9]*



(a) Simulation of a fifty-degree-of-freedom string marionette




(b) Its associated graph representation

2: Simulation using recursive tree forms. See *http://puppeteer.colorado.edu* for this simulation.

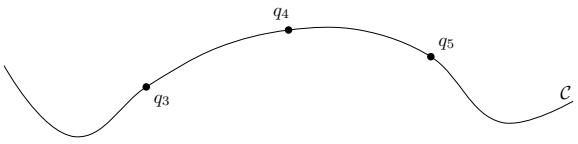
by a constraint function $g(q)$ such that for all admissible configurations $g(q) = 0$. Formally, we define the constrained sub-manifold as $\mathcal{C} = \{q \in \mathcal{Q} \mid g(q) = 0\}$ where $\mathcal{Q}$ is the configuration manifold of the mechanical system. Looking at (3)) and (4), there are two natural choices of where to enforce a holonomic constraint; at the discrete configurations $q_0, q_1, \dots, q_N$ or at the quadrature point where $L$ is being evaluated. The first is the *one-point constraint* method while the second is the *two-point constraint* method.

### A. One-Point Constraints

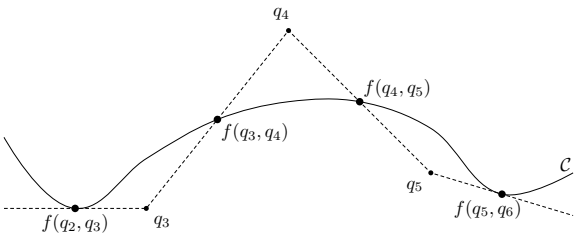
The variational integrator framework is extended to include holonomic constraints by requiring that each $q_k$ of the discrete trajectory satisfy the constraint (i.e. $g(q_k) = 0 \;\forall\; k$). The constraints are introduced to the discrete action with Lagrange multipliers, $\lambda_k$, at each time-step.

$$S = \sum_{k=0}^{N-1} L_d(q_k, q_{k+1}) - \langle \lambda_{k+1}, g(q_{k+1}) \rangle \tag{5}$$

The discrete $\lambda_k$ (and $\lambda_{k+1}$) is analogous to the Lagrangian multipliers $\lambda$ in the continuous case and are related to the magnitude of the constraint forces at each time step. Applying the variational principle to (5) with $q_k$ and $\lambda_k$ as the dynamic variables leads to the constrained discrete Euler-Lagrange equations:




(a) One Point Constraint




(b) Two Point Constraint

3: The two methods for including constraints in a variational integrator. One-point constraints force each $q_k$ to satisfy the constraint. Two-point constraints force some intermediate point between $q_k$ and $q_{k+1}$ to satisfy the constraint.

$$D_2L_d(q_{k-1}, q_k) + D_1L_d(q_k, q_{k+1}) - \langle \lambda_k, Dg(q_k) \rangle = 0 \tag{6a}$$

$$g(q_{k+1}) = 0 \tag{6b}$$

We integrate the system using the same procedure described in section I-A with a slight modification. The root-finder now solves for both $q_{k+1}$ and $\lambda_k$ at each time step by evaluating (6a) and (6b).

For an in-depth discussion of one-point constraints, including proofs, see [8] and [9].

### B. Two-Point Holonomic Constraints

[7] suggests that the holonomic constraints could be enforced at intermediate points (i.e. $f(q_k, q_{k+1})$) between discrete configurations instead of at the configurations themselves. We refer to this idea as *two-point* constraints. Figure 3 illustrates the difference between one- and two-point constraints.

This method is desirable, for example, with the discrete Lagrangian in (4) where the dynamics are evaluated at $q = (1 - \alpha)q_k + \alpha q_{k+1}$. With the two point constraint, the constraint can be evaluated at this same point. This guarantees that the dynamics will be evaluated at points in the constraint set.

There can also be performance benefits to two-point constraints. In [3], an algorithm is presented to implement numeric variational integrators for arbitrary mechanical systems, without using symbolic algebra software. The key performance optimization for this method is extensively reusing earlier calculated values to avoid duplicated efforts (i.e. caching). The dynamics require that values are evaluated using the midpoint state, so the results from those calculations are already available. A two-point constraint implementation will use them without additional computational effort. A one-point constraint, on the other hand, has to re-evaluate the same equations using a normal state (i.e. non-midpoint).

To include two-point constraints, we define a new constraint equation:

$$\bar{g}(q_k, q_{k+1}) = g(f(q_k, q_{k+1})) \tag{7}$$

where $f(q_k, q_{k+1})$ is an interpolating function that takes two configurations and returns a new configuration. For the generalized midpoint rule $f(q_k, q_{k+1}) = (1-\alpha)q_k + \alpha q_{k+1}$.

The action sum is updated with the new constraint function:

$$S = \sum_{k=0}^{N-1} L_d(q_k, q_{k+1}) - \langle \lambda_{k+1}, \bar{g}(q_k, q_{k+1}) \rangle \tag{8}$$

The action is again extremized with the variational principle to obtain a different set of constrained discrete Euler-Lagrange equations:

$$D_1L_d(q_k, q_{k+1}) - \langle \lambda_{k+1}, D_1\bar{g}(q_k, q_{k+1}) \rangle + D_2L_d(q_{k-1}, q_k) - \langle \lambda_k, D_2\bar{g}(q_{k-1}, q_k) \rangle = 0 \tag{9}$$

$$g(q_{k+1}) = 0 \tag{10}$$

This integrator is advanced using the same algorithm as the one-point constraint integrator. However, it requires an additional initial condition for $\lambda_1$.

Unfortunately, the derivation doesn't provide an obvious choice. [7] suggests a procedure that is essentially equivalent to $\lambda_1 = 0$. For now we use that.

There is still another ambiguity for the initial conditions. The initial conditions can satisfy $g(q_0) = g(q_1) = 0$ or $\bar{g}(q_0, q_1) = 0$, but not always both. This depends on the coordinate choice and shape of the constraint manifold. If the constraint is convex[2] in the chosen coordinates, both can be satisfied. Otherwise, a decision must be made. Neither [7] nor the derivation suggest one over the other. Initially we will choose initial conditions so that the two configurations satisfy constraints.

## III. Example: Point on a Circle

We first compare the two methods by considering a particle in the plane that is constrained to the unit circle[3].

### A. Euler Coordinates

Let the configuration $q = (x, y)$ where $x$ and $y$ are the coordinates of the point in the plane. Ignoring gravity and setting $m = 1$, the Lagrangian for this system is

$$L(q, \dot{q}) = \tfrac{1}{2}(\dot{x}^2 + \dot{y}^2) \tag{11}$$

[2]In this context, *convex* means for all $q_0, q_1 \in \mathcal{C}$, $f(q_0, q_1) \in \mathcal{C}$. Convexity, therefore, depends on the constraint, the coordinate representation of $\mathcal{Q}$, and the interpolation method $f(\cdot, \cdot)$.

[3]Note that for both examples we could eliminate the constraint by choosing a more appropriate configuration.

The discrete Lagrangian for the variational integrator is defined by the generalized midpoint rule (4) with $\alpha = \frac{1}{2}$:

$$L_d(q_k, q_{k+1}) = L\left(\frac{q_k + q_{k+1}}{2}, \frac{q_{k+1} - q_k}{\Delta t}\right)\Delta t \qquad (12)$$

The particle is constrained to the unit circle.

$$g(q) = x^2 + y^2 - 1 \qquad (13)$$

The two-point constraint is defined using the same generalized midpoint rule used in the discrete Lagrangian.

$$\bar{g}(q_k, q_{k+1}) = g\left(\frac{q_k + q_{k+1}}{2}\right) \qquad (14)$$

The system was simulated in Mathematica using one-point and two-point constraints. Both used the same initial conditions (with $g(q_0) = g(q_1) = 0$) and were simulated with a constant time-step of $0.01$ seconds over a time of $0.65$ seconds.

The resulting trajectories are shown in Fig. 4. The one-point integrator behaves as expected. The particle travels at a constant velocity and remains on the circle.

The two-point integrator is unstable. The particle bounces in and out of the circle while still satisfying the the constraints between steps. The points are also further apart after each time step, suggesting that the particle is gaining energy.

The instability could be caused by the wrong initial condition $\lambda_1$. This is unlikely. Suppose we had the correct $\lambda_1$. We could augment the integrator to include forcing and apply a force that cancels out the correct constraint force so that the combined terms equal the current initial constraint force. An applied force should alter the trajectory, but not cause instability.

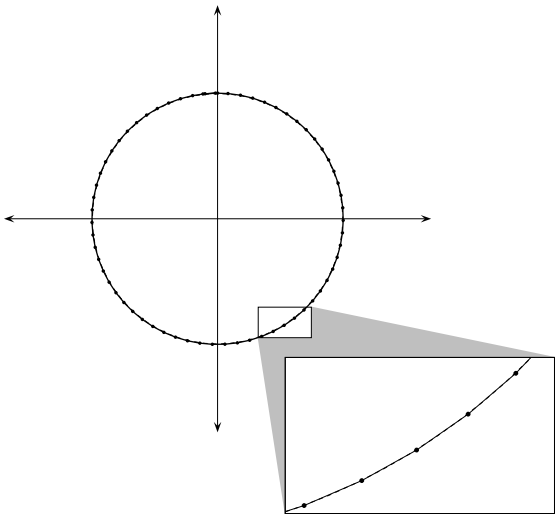
Another possible cause is the choice of initial configurations $q_0$ and $q_1$. For this coordinate representation and interpolation, the circular constraint is not convex; the initial configurations and their midpoint cannot simultaneously satisfy the constraint. For Fig. 4b, we chose $q_0$ and $q_1$ to satisfy the constraint as it is straightforward and more natural to specify configurations that independently satisfy the constraint.

Fig. 5 shows a two-point simulation in $XY$ coordinates that has been improved by considering these two issues. The initial configurations are now chosen so that $f(q_0, q_1) \in \mathcal{C}$ while $q_0, q_1 \notin \mathcal{C}$. Initial condition $\lambda_1$ was varied experimentally and finally set at $\lambda_1 = -\frac{1}{2}$. This is the same value of every $\lambda_k$ in the one-point constraint simulation. The trajectory is improved but is still unstable. This strongly suggests that even with the correct $\lambda_1$, the two-point constraint method would be *numerically* unstable.

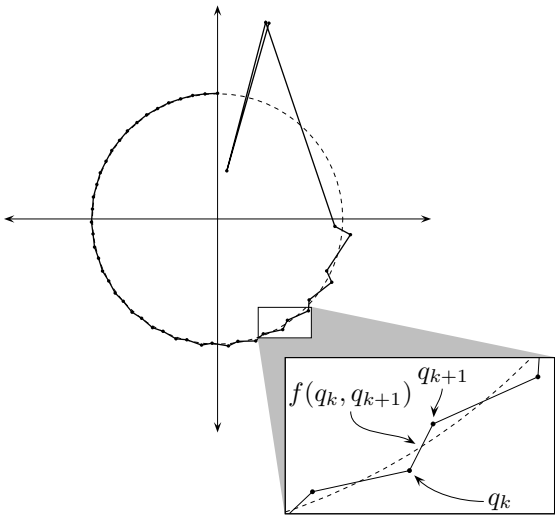
### *B. Polar Coordinates*

The comparison continues by choosing a different coordinate representation. Polar coordinates are an appealing choice because they naturally express the circular constraint.

The new configuration is $q = (\theta, r)$ where $\theta$ is the angle between the horizontal axis and a line from the origin to the particle, and $r$ is the distance from the origin to the particle.

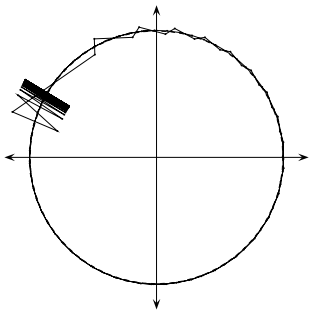


(a) One-Point Constraint

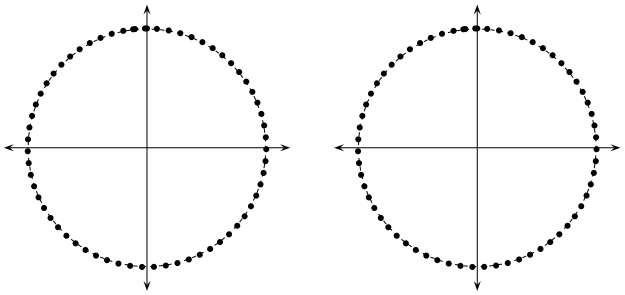



(b) Two-Point Constraint

4: Simulation results for one-point and two-point constraints in XY coordinates. The particle is constrained to the unit circle.

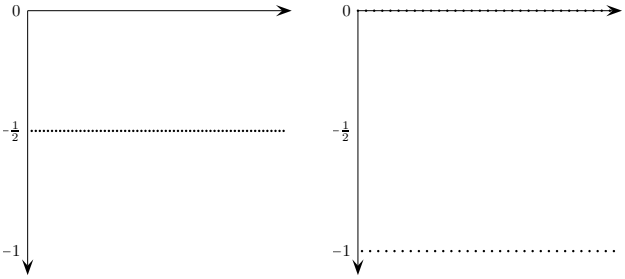


5: Improved simulation results for the two-point circle constraint in $XY$ coordinates.



(a) One-Point Constraint (b) Two-Point Constraint

6: Simulation results using one and two point constraints in polar coordinates. The particle is constrained to the unit circle.




(a) One-Point Constraint (b) Two-Point Constraint

7: $\lambda_k$ for the simulations using one and two point constraints in polar coordinates.

Ignoring gravity and choosing $m = 1$ leads to the continuous Lagrangian

$$L(q, \dot{q}) = \tfrac{1}{2}\left(\dot{r}^2 + (r\dot{\theta})^2\right) \tag{15}$$

The constraint equation is

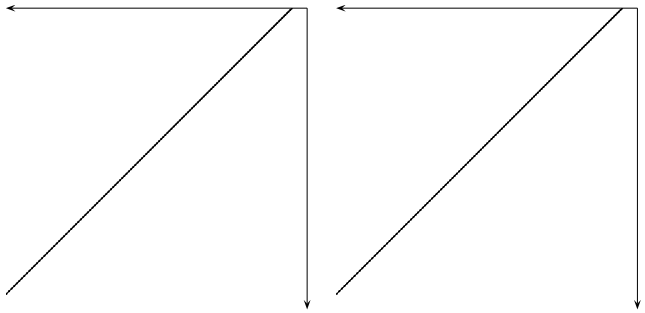
$$g(q) = r^2 - 1 \tag{16}$$

The discrete Lagrangian and the two-point constraint are the same definitions from the previous derivation (Equ. (12) and (14)).

The system was simulated using the same implementation in Mathematica with a time-step of 0.01 seconds for a duration of 0.65 seconds. The resulting trajectories are shown in Fig. 6. In this case, both trajectories are stable.

The results do differ for the constraint forces. Figure 7 shows the $\lambda_k$ sequences for both simulations. The one-point constraint is a constant value as expected for a particle moving in a circle at a constant velocity.

The two-point constraint, on the other hand, oscillates between 0 and -1. This is related to the initial condition $\lambda_1$. If $\lambda_1$ is small, $\lambda_2$ takes on a large value to compensate. In turn, $\lambda_3$ takes on a small value to compensate for $\lambda_2$. This establishes the oscillating pattern seen in the results. By choosing $\lambda_1$ to be the average of these two values, we can run the simulation again and get a constant value for $\lambda_k$ as expected. This strongly suggests that the starting procedure from [7] is incorrect.



(a) One-Point Constraint (b) Two-Point Constraint

8: Simulation results using one and two point constraints in XY coordinates. The particle is constrained to the line $y = x + 1$. Gravity is included to make the linear trajectory nontrivial.

Surprisingly, we find the value of $\lambda_1$ affects the $\lambda_k$ sequence but does not affect the state trajectory in the polar coordinate simulation.

A potential reason for the difference in stability is that in polar coordinates, we do simultaneously satisfy $g(q_0) = g(q_1) = 0$ and $\bar{g}(q_0, q_1) = 0$.

## IV. Example: Point on a Line

Motivated by the previous results, we consider a particle confined to the line $y = x+1$. In this case, the constraint set is convex in $XY$ coordinates, but not in polar coordinates. We now include gravity to make the linear trajectories non-trivial (ie, so the constraint forces must be non-zero).

Using $q = (x, y)$ and $m = 1$, the continuous Lagrangian is:

$$L = \tfrac{1}{2}(\dot{x}^2 + \dot{y}^2) - gy \tag{17}$$

The constraint to stay on the line $y = x + 1$ is:

$$g(q) = (x + 1) - y \tag{18}$$

The discrete Lagrangian and two-point constraint are defined as in (12) and (14).

The two integrators were simulated in Mathematica with a time-step of 0.01 seconds for a duration of 2 seconds. The results are shown in Fig. 8. Both the one- and two-point constraints have stable solutions and we see the same oscillating behavior in $\lambda_k$ for the two-point constraint.

### A. Polar Coordinates

The continuous Lagrangian in polar coordinates with gravity is:

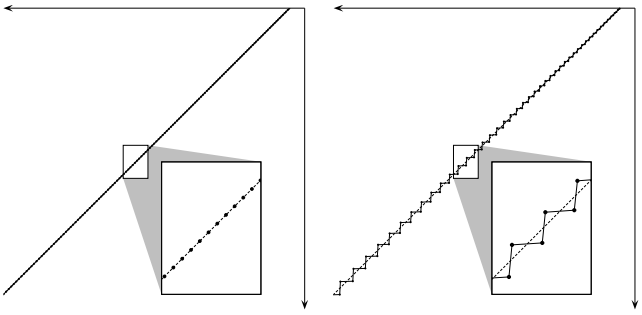
$$L(q, \dot{q}) = \tfrac{1}{2}(\dot{r}^2 + (r\dot{\theta})^2) - gr\sin\theta \tag{19}$$

The constraint for the line $y = x + 1$ in polar coordinates is:

$$g(q) = (r\cos\theta + 1) - r\sin\theta \tag{20}$$

The discrete Lagrangian and two-point constraints are defined as in (12) and (14).

The two integrators were simulated in Mathematica with a time-step of 0.01 seconds and a duration of 2 seconds.



(a) 1 Point, Polar Coordinates (b) 2 Point, Polar Coordinates

9: Simulation results using one and two point constraints in polar coordinates. The particle is constrained to the line $y = x + 1$. Gravity is included to make the linear trajectory nontrivial.

The resulting trajectories are plotted in Fig. 9. The one-point constraint is stable while the two-point is unstable. Note that the line constraint is non-convex in polar coordinates.

## V. CONCLUSIONS AND FUTURE WORKS

The results of this comparison demonstrate that the two-point constraint method is, in its current form, a poor choice. Stability strongly depends on the relationship between the coordinate choice and shape of the constraint manifold. Our results suggest stability related to the quadrature choice being convex in local coordinates. We have demonstrated that solutions may be be stable in one coordinate choice but not another. Coordinate invariance is a major theme in Lagrangian mechanics, so this is particularly undesirable.

Additionally, there is currently no clear, justifiable starting procedure for either the initial configurations or $\lambda_1$. It is natural to specify initial configurations that satisfy the constraints, but this can be inconsistent with how the constraints are enforced by the simulation. On the other hand, specifying initial configurations to satisfy the two-point constraint could be a difficult problem for sufficiently complex mechanical systems. Our results suggest that a better starting procedure will improve results, but may still be subject to numerical instability.

The conclusions drawn from this work are based entirely on experimental results. Open questions on this topic are an analysis on the convexity-stability relationship and an improved initialization procedure for $\lambda_1$.

## VI. ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under CAREER award CMS-0546430. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

## REFERENCES

[1] R.C. Fetecau, J.E. Marsden, M. Ortiz, and M. West. Nonsmooth lagrangian mechanics and variational collision integrators. *SIAM Journal on Applied Dynamical Systems*, 2003.

[2] E. Johnson and T. D. Murphey. Scalable variational integrators for constrained mechanical systems in generalized coordinates. *IEEE Transactions on Robotics*, Submitted.

[3] E.R. Johnson and T.D. Murphey. Discrete and continuous mechanics for tree representations of mechanical systems. In *International Conference on Robotics and Automation*, 2008.

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

[5] A. Lew, J.E. Marsden, M. Ortiz, and M. West. An overview of variational integrators. *Finite Element Methods: 1970's and Beyond*, 2003.

[6] A. Lew, J.E. Marsden, M. Ortiz, and M. West. Variational time integrators. *International Journal for Numerical Methods in Engineering*, pages 153–212, 2004.

[7] Adrián Lew. Variational time integrators in computational solid mechanics. *California Institute of Technology Thesis*, 2003.

[8] J. E. Marsen and M. West. Discrete mechanics and variational integrators. *Acta Numerica*, pages 357–514, 2001.

[9] Matthew West. Variational integrators. *California Institute of Technology Thesis*, 2004.