Richard J. Arietta
CIS599 – Independent Master's Study
Advisor: Dr. Stephen Lane
CGGT Fall 2014

Dynamic Space-Time Optimization of Articulated Character Motion
Final Report

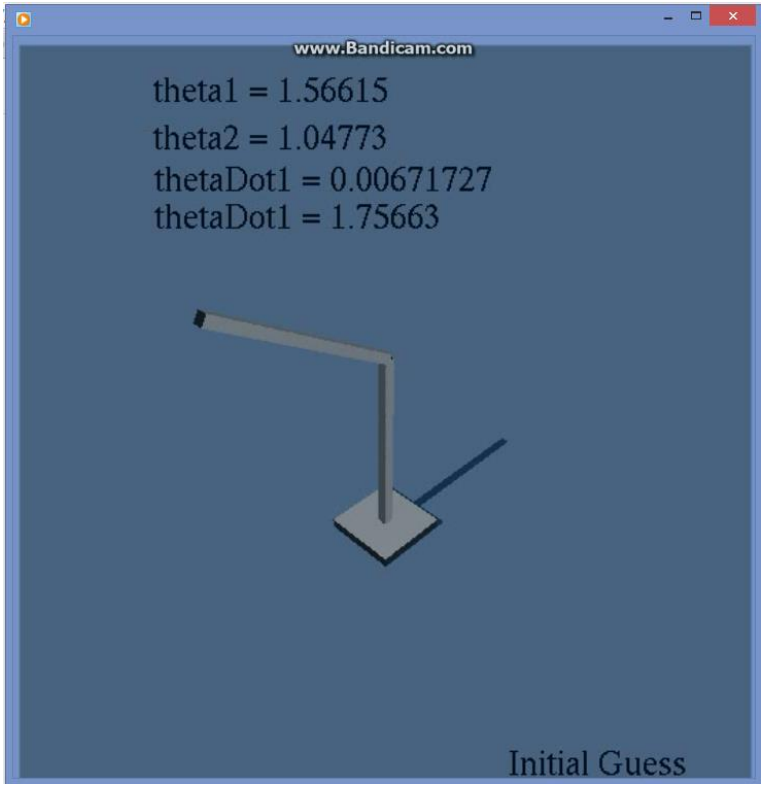

Figure 1:
The Spacetime Optimization program running on the simple two-joint robotic arm system utilized for the analytic approach to this project

Acknowledgements

Before entering the discussion of my research project, I would simply like to thank Dr. Lane, my advisor, for his continued help and guidance during this semester. I may not have gotten this project to the point either of us were initially hoping, but his patience and tutelage helped me to learn a great deal about dynamic systems, optimal control, and software engineering regardless. I could not have gotten this project even to the point that it currently exists without his help. I am thankful for his generous understanding of the challenges to my success and his willingness to renegotiate the terms of the project as new such challenges became apparent.

Abstract

The main goal of this research project is to design an algorithm for the spacetime optimization of articulated character motion. By this, I mean this project intends to find a sequence of input torque vectors u for a system describing a multi-joint character that traces a trajectory from a predefined initial state, as specified by the angular orientation and angular velocity of each joint in the system, to a predefined final state. This input torque sequence, in order to be optimal, must minimize the energy function $L = \sum_t u^T u$ put into the system over the entire time period of the trajectory.

The dynamic system is implemented within the existing and widely available physics simulation engine PhysX. Furthermore, while initially having loftier goals of incorporating automatic numeric differentiation libraries into the dynamics calculations and generalizing to higher joint counts and degrees of freedom, this implementation suffered from some much-larger-than-anticipated setbacks and a lack of successful results and instead incorporates a purely analytic definition of a two-joint pendulum system.

Contents of Final Report

I. Project Overview

For my independent research this semester, I originally sought out to design an algorithm for the spacetime optimization of generalizable articulated character motion. By this, I mean I intended to find an optimal sequence of input torque vectors u for a system describing a multi-joint character that minimizes the energy function $L = \sum_t u^T u$ put into the system while satisfying one or more constraints on the system's orientation and velocity. These constraints, such as the initial and final state of the system, are presented in the form of state vectors which specify the orientation angle and angular velocity for each joint in the articulated system (as is discussed in Section II below).

At the suggestion of my advisor Dr. Lane, I intended to utilize the capabilities of an existing and publically available physics simulation engine (PhysX, in this case) and the automatic numeric differentiation library ADOL-C (Automatic Differentiation by OverLoading C++). The drive to build the project around the PhysX engine was its easy integration into existing gaming and animation technologies like the Unity Engine. The benefit of the ADOL-C library would be the avoidance of complex analytic computations and dynamic relationships for multi-joint systems, since such analytic matrices can easily grow out of control for any system with more than two simple joints and inhibit the possibility of optimizing the inverse dynamics formulas.

Very unfortunately, this project did not turn out as I had hoped. I am somewhat disappointed in the ultimate results, but I feel that I worked extremely hard to get where I am and learned a great deal regardless of the setbacks.

There was a much steeper learning curve than I anticipated in my attempts to incorporate the PhysX API and establish a stable scene. The interface to this physical simulation engine was complicated and poorly documented, and it took me a long time to install the libraries and navigate the correct way to set up my system. Getting the physical values out of the scene that were required by the various approaches to the optimization project was not intuitive, and rounding errors by the system sometimes complicated my attempts to validate the computations. This unfortunately put me a bit behind my anticipated schedule for the remainder of the project, which made it difficult to accomplish what I originally set out to do, but I'm at least happy to say I have tackled the engine successfully and learned a great deal about interfacing with such technologies.

I had some further issues linking the ADOL-C project to my own project later on in the course of the semester. After some contact with the project team and multiple attempts to build the project on my own in order to avoid version conflicts, I was finally able to link these libraries successfully as well and learn about their use.

However, I regrettably ended up not following this approach at all. In an attempt to push forward during these linking struggles, I came to the unfortunate realization that my implementation of the iterative solver was not functioning properly. I ended up not being able to implement the automatic numeric differentiation approach to the problem before the ultimate deadline. Instead, I reached the decision with my advisor to put the more-complex, automatically-differentiable system on the back burner and to focus on successfully

implementing the dynamic optimization solver for a simple two-joint system under an analytic approach.

In what started as a preemptive attempt to provide a means of validation for my numeric derivatives for simple cases that never came to fruition, I took an analytic approach to the optimization for a simple two-joint robotic arm. The implementation of this hard-coded approach to the system proved to me that the dynamic optimization approach I had in place was not as stable as anticipated, and was not ready for a more generalized or complex system with heavy numeric computations, and this analytic, two-joint approach became the ultimate basis of my project. I used a series of inverse dynamics formulas adapted from Asada's "Introduction to Robotics" to define the analytic relationships of the system, and also analytically computed the derivatives of the system by hand rather than numerically doing so via a third party library.

The rest of this final report will give a breakdown of the optimization approaches I implemented with unfortunate little success for my project, along with an intensive breakdown of the analytic system of equations used in each approach and a full overview of the code base with respect to the various files and function definitions.

II. Background of Spacetime Optimization

This section includes an explanation of seminal works in the field of space time optimization in animated character motion, as well as a general survey of the current state of the art. Many of the papers and methods discussed below form the foundation of my research and inspire my implementation of the spacetime problem in a new dynamic optimization problem space. In entering this project, my hope – though unrealized – was to build on these existing ideas to bolster the efficiency and utility of the approach.

The idea of spacetime constraints was introduced in 1988 by Andrew Witkin and Michael Kass **[WK88]**. At this early point in the field of computer animation, there were two opposing extremes for defining the motion of animated characters or objects. The first of these was fully driven by an animating artist, who defined the position, orientation, and motion of each joint of each character manually through the use of key frames. While this method is extremely useful in defining style and direction, it is often difficult for an animator to establish realistic and physically motivated motion paths. These paths require complex adherence to physical laws and creating an optimal course of motion demonstrating believable physical weight still poses a large artistic challenge to this day. On the other hand was (and still is) the alternate extreme: physical simulation. Simulation takes all control and direction away from the animator or user, requiring only the definition of a start state and relying entirely on physical systems to perpetuate the motion and state vector of the animated object in question. The motion cannot be controlled or constrained in any way to reach desired positions or accomplish desired goals in any sense beyond altering this start state.

Witkin and Kass saw an opportunity to bridge this gap between artist-driven animation and physical simulation. Their seminal paper on spacetime constraints sought to establish a means

for artists to define a set of controls on an animation and rely on physical principles to easily optimize the motion between these controls in a physically plausible way. Based on user-input constraints governing the "what" and the "how" of the animated motion – i.e. "jump from here to there, clearing a hurdle in between," and "don't waste energy," or "come down hard enough to splatter whatever you land on" – their method simulates the locally optimal path of motion. The method works by iteratively solving a predefined objective cost function subject to a series of constraint functions using Sequential Quadratic Programming (SQP) as introduced in **[GMW81]**, thus retaining artistic control without sacrificing physical accuracy.

Unfortunately for the field of spacetime constraints, the methodology is very costly in terms of runtime and incredibly difficult for users to initiate (it is rather complicated to set up reasonable constraints, objective functions, and character graphs for this kind of system). Since Witkin and Kass' initial proposal of the spacetime constraint paradigm, various authors and researchers have sought to improve its efficiency or realistic usability. Five years later, J. Thomas Ngo and Joe Marks presented "Spacetime Constraints Revisited" at SIGGRAPH '93 **[NM93]**. This paper took a much different approach at solving the spacetime constraint problem, actually structuring it in an entirely different way much more akin to modern A-Life research. They used an iterative genetic algorithm on a computed set of reflexes to optimize the character behavior. This iterative approach mirrors the approaches taken in this project to a degree, but this research does much more work on reflexes and character systems that the state-based iterative optimization presented in this project.

One much-noted contribution to the field is that of Michael F. Cohen of the University of Utah **[C92]**. His 1992 Paper entitled "Interactive Spacetime Control for Animation," proposes a solution of some of the issues in Witkin and Kass' method. While the practical solution to the spacetime constraint problem involves the use of B-Spline polynomial curves to express the continuous forces and positions over the time period, the computational complexity associated with this process still poses a number of practical problems. Cohen's work allowed for user's to interact with the optimization and guide it to a stable, realistic state after less iteration. He also proposed performing complex computations on smaller regions of the motion paths, known in his work as windows. This also helped to reduce the complexity of the problem.

In 1996, Liu completed his thesis work at Princeton University on the subject matter **[L96]**. Building off of these earlier works, Liu's contributions included a new symbolic interface with a recursive evaluation scheme for reducing the time needed by the numerical optimization, as well as an added key frame specification interface for artists that build on earlier work by Cohen. This key frame input method served to both increase artist control and reduce the computational complexity of the solver. Finally, Liu's work also introduced the idea of wavelets to solve the nonlinear variational problems associated with the problem formulation.

One contribution of interest is the 2004 paper by Pär Winzell, based on work done for 1998 Master's thesis while Winzell was a researcher at the Linköping Institute of Technology in Sweden **[W04]**. The title of the paper is "An Implementation of the Spacetime Constraints Approach to the Synthesis of Realistic Motion", and it made a number of contributions to the field of spacetime constraints by proposing a realistic implementation. Winzell's work gives a thorough description of the process, accounting for all forces and constraints and establishing a set of Hermite basis functions for expressing the variable force equations over the time intervals, which he divides into stages based on local regions of continuity. He also proposes and

details the use of the ADOL-C library for C++ in order to efficiently compute the necessary derivative functions for the calculation (i.e. Jacobian and Hessian matrices). This ADOL-C library is discussed in depth in a later publication by Walther and Griewank **[WG12]**. Winzell's work further establishes the definitions for the objective cost function to be optimized, which he defines as a function of the work done by each muscle on each joint over the course of the defined time period. He also gives a solid explanation of the character setup and the recursive methods necessary for computing these well-derived torques and Cartesian forces. As a useful addition, Winzell also gives a relatively detailed description for his C++ implementation and a list of necessary classes.

III. Implementation Details

As described above, the overall goals of this project went through many twists and turns as the end of the semester drew nearer. Similarly, the approach to the optimization solver itself went through many changes as my advisor and I learned more about the process and the existing research.

We initially intended to follow an SQP (Sequential Quadratic Programming) approach using a B-Spline curve to account for various constraints on the system's state. After delving into the existing literature and looking at various spacetime optimization approaches, however, Dr. Lane and I decided to reappropriate the Dynamic Optimization defined in Poulsen's 2012 course notes **[P12]** to control the multi-joint articulated character as the first substantial approach to the optimal input torque sequence and resulting optimal trajectory. This approach will be discussed completely and specifically in subsection **(b)** of this section.

Although the mathematical system for this optimization is seemingly implemented according to the approach defined in the existing research, this approach was met with unfortunate little success. Thus, with little time left in the semester, we quickly tried to find an alternative approach to the optimization that could be quickly adapted to my code base and existing system definition. Once again at the suggestion of Dr. Lane, the algorithm chose was an implementation discussed in Tassa et al.'s 2012 paper "Synthesis and Stabilization of Complex Behaviors through Online Trajectory Optimization" **[TET12]**. This approach will be discussed completely and specifically in subsection **(c)** of this section.

In the end, however, this implementation also faced some challenges and satisfying results were not ultimately achieved. Regardless of the shortcomings of the methods chosen and the lack of satisfying results, the incredible amount of work done on this project was still a thorough and substantial exploration into dynamic control systems and spacetime optimization techniques. The algorithms are worth exploring here in depth, for they have shown success in other applications and demonstrate a complex and thorough understanding of inverse dynamics and differential analytic linear systems on the part of the researcher.

a. Definition of Dynamic System

Before diving into the various algorithms attempted in this spacetime optimization, it is necessary to outline the crucial details of how the system is represented. The system is a

two-joint robotic arm or pendulum. The state X of the system at any point in time is defined as a vector of the angular joint orientations θ (defined in local space with respect to the parent body) and the angular joint velocities $\dot{\theta}$. In the case of our simple two-joint system, we only define one degree of freedom about the x-axis, such that the pendulum system operates entirely within the YZ-plane of space. Thus, this state vector is a 4x1 state vector and is defined as follows for any point in time t:

$$X_t = \begin{bmatrix} \theta_{1_t} \\ \theta_{2_t} \\ \dot{\theta}_{1_t} \\ \dot{\theta}_{2_t} \end{bmatrix}$$

The derivative of this state vector defines how the system moves through space over time, just as in any dynamic system. To define this derivative $\dot{X}$ at any point in time t, we rely on the inverse dynamics formula as defined in the Asada reading **[A05]**:

$$\dot{X}_t = \begin{bmatrix} \dot{\theta}_t \\ \ddot{\theta}_t \end{bmatrix} = \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ M^{-1}[u - C - G] \end{bmatrix}$$

In the above inverse dynamics formula defining the angular acceleration of the system, we see a set of crucial matrix variables (G, C, and M) that persist throughout the optimization calculations. These matrices represent a set of state-specific values:

- M represents the mass matrix of the system
- u represents the input torque to the system (this is the sequence we are trying to optimize over the timeframe in order to minimize the energy put into the system while still achieving the desired constraint conditions)
- C represents the contributions on the system from centripetal and Coriolis effects
- and G represents the angular force due to gravity

All of these state-specific system variables as they apply to the two-joint analytic system in both approaches are defined fully in the respective appendices.

b. Computation of Initial Estimate with PD Controller

Despite the multiple different and unsuccessful approaches to the optimization, however, there was one stage of the program that stayed constant and displayed very successful results. This stage was the initial estimation of the input torque sequence to be iteratively improved by each approach, which is entirely necessary for any optimization.

The PD controller estimate relies on two coefficients K_p and K_v which control the position (angular orientation) and velocity respectively. The input torque vector is defined at each time step as

$$u_t = C_t + G_t + {M_t}^{-1}\left[K_p(\theta_d - \theta_t) - K_v\dot{\theta}\right]$$

This estimate provides the system with a reasonable initial torque sequence on which the iterative optimization can be performed. The results of the PD controller's initial estimate can be seen in the included video entitled "PD_Controller.avi" which takes the system from state $[0 \quad 0 \quad 0 \quad 0]^T$ to state $\left[\frac{3\pi}{4} \quad \frac{-\pi}{2} \quad 0 \quad 0\right]^T$.

c. Overview of Approach 1: Dynamic Spacetime Optimization Algorithm

For my implementation of this Dynamic Optimization algorithm, the constraints on the system are simple. I define a start state and an end state with the goal of moving the system from one to the other with an optimally minimal total input torque.

The algorithm works as follows:

1. First, we make an initial estimate of the input torque vector sequence over the entire time frame $t_0 - t_f$ using a simple PD controller.

2. Following this initial estimation, we enter the optimization solver. As described in the Poulsen notes, this solver is based on a forwards and backwards iterative approach. The forwards integration involves the state vector X that was discussed previously, while the backwards integration relies on a dimensionally consistent co-state vector λ. The process is as follows:

    a. First, using the current definition of u, we integrate the state forwards using the inverse dynamics formula $\dot{X} = f(X, u)$ and it follows that $X_{t+1} = X_t + \dot{X}\Delta t$. We integrate the physical system as such for all of the time steps over the entire range and save the state vectors, as well as the calculations of the C, G, M, and M^{-1} matrices for each time step.

    b. Next, we seek to integrate the co-state vector λ backwards through time. To do this, we utilize the pre-computed state matrices at each time step and their derivatives, the computations of which are described in the next section. Starting at the final time step t_f, where $\lambda_{t_f} = -\left[{f_x}^T\right]^{-1} * u_{t_f}$ utilizing the right pseudo-inverse of the 4x2 matrix f_x, we integrate backwards via the relationship ${\dot{\lambda}_t}^T = -\frac{\partial L}{\partial X} + {\lambda_t}^T\frac{\partial f}{\partial X}$

    c. Finally, we solve for the next iteration of the input torque vector sequence via the formula ${u_t}^T = -{\lambda_t}^T\frac{\partial f}{\partial u}$

The algorithm repeats this three-phase iteration on the system until the torque sequence converges to an optimal solution, which we identify as occurring when the difference between two successive input torque sequences (as defined by sum of square differences over the time period in question) passes below a predefined threshold. At this point, the algorithm is deemed to have found the energy-minimizing optimal input torque solution for the constraints on the system.

As discussed previously, the results of this approach were unsatisfactory. Despite the math of the optimization being consistent with the approach defined in both Poulsen's notes and other descriptions of the general dynamic optimization approach **[LW]**, the optimized input torque sequence quickly drives the system to an unstable trajectory and successive iterations do not converge to a correct solution but only display greater instability.

d. Overview of Approach 2: Discrete Online Trajectory Optimization Algorithm

Rather than re-computing the entire input torque vector sequence in each iteration as the previous approach suggests, this discrete online trajectory optimization algorithm works by iteratively modifying the existing sequence of input torque and states by a computed differential.

It focuses on minimizing the cumulative value function over the sequence, where the value at each time step is defined as the "cost-to-go" until the end state plus the error on the end state (the overall difference between the end state and the desired end state).

Much like in the above implementation, this is done in two sweeps: a backwards sweep that uses the derivatives of the system and of the value function at time t to determine a pair of coefficients $k(t)$ and $K(t)$, and a forwards sweep that uses these precomputed coefficients to compute a modification of the existing input sequence.

Following the initial estimate as described earlier, the main logic of the optimization iteration is as follows:

1. For the final state, compute $V(t_f) = 0.5 * \left(X_d - X_{t_f}\right)^T \left(X_d - X_{t_f}\right)$

2. Perform a backward pass of the iteration, which entails:

    a. Compute the value $V(t)$ and its derivatives for each time step t to be used in the expansion coefficient calculations

    b. Compute all expansion coefficients Q_u, Q_x, Q_{xx}, Q_{xu}, Q_{ux}, and Q_{uu} using the modified equations listed in the relevant appendices

c. Compute the values of $k(t) = -Q_{uu}^{-1}Q_u$ and $K(t) = -Q_{uu}^{-1}Q_{ux}$

3. Perform a forward pass, which entails:

   a. Initialize $\hat{x}(0) = X_0$

   b. Starting at $t = 0$, compute $\hat{u}(t) = u(t) + k(t)(\hat{x}(t) - x(t))$, where $x(t)$ is the current state value and where $\hat{x}(t)$ is the updated state value computed in the previous time step

   c. Compute the updated state value $\hat{x}(t+1) = f(\hat{x}(t), \hat{u}(t))$ for the next time step given $\hat{x}(t)$ and $\hat{u}(t)$

4. Repeat 1 – 3 until convergence

Unfortunately, the dimensionality of all necessary derivatives and value functions for my implementation did not seem to align with the formulas put forth in the paper. This forced me to spend some time modifying the algorithm of the paper for compatibility which, due to the limited time available to me between beginning this implementation approach and the final submission deadline, did not leave me with enough time to debug the algorithm and produce viable results. The entire algorithm is present in the code as it currently stands

The full breakdown of all necessary expansion coefficients and derivative functions for this approach is presented in Appendix C.

IV. Overview of Code Base

Here is a rundown of the program structure (which admittedly has some repeated functionality between files because of the multiple different approaches taken):

- The main header class is **Spacetime.h**, which contains all relevant function definitions and variables for the system.

- The main program loop is found in **SpacetimeApplication.cpp.** The program takes three arguments, which are the initial global joint angles for the two joints (from the positive X-axis) and the number of time steps for which to run the simulation. The program can be called from the terminal with the SpacetmeOptimization.exe executable in Bin/win64

- The optimization wrapper code currently lives in **Render.cpp**. The render loop calls the makeInitialGuess() command for each of numTimeSteps time steps to estimate the next successive input torque vector using the PD controller we defined previously. Once the initial input torque sequence is estimated over the entire time range, the render loop calls the IterateOptimization() function to iteratively modify the input torque sequence based on the dynamic optimization approach involving the state and costate sequences.

Both the makeInitialGuess() function and IterateOptimization() function – which for the purposes of this submission is still implementing Approach 1 from the paper above – live in **SpacetimeOptimization.cpp.** One very important thing to note about the program structure of Render.cpp is the "mode" variable found in the renderCallback() function. This mode, which can be set to 'ANALYTIC', 'DISCRETE', or 'NUMERIC' controls which approach is taken in the simulation. As noted in this report, the results from the 'ANALYTIC' mode are incorrect, while the results from the 'DISCRETE' mode currently encounter some as-of-yet-unidentified runtime errors. Hopefully these will be ironed out in the near future and the implementation will run as expected. The 'NUMERIC' mode would take advantage of the ADOL-C approach were it completed.

- The computations for the derivatives used in the Dynamic Spacetime Optimization approach are found in two mirroring files, **SpacetimeDerivatives_Analytic.cpp** and **SpacetimeDerivatives_Numeric.cpp**. Since the simple analytic case proved largely unsuccessful, the numeric derivatives file is mostly a shell right now waiting for ADOL-C tapes to be fed into it for reading of automatically computed numeric derivatives. However, all the analytic derivatives, partial and full, are computed and implemented in the analytic case.

- The computations for the derivatives used in the Discrete Online Trajectory Optimization approach are alternatively found in **SpacetimeDerivatives_Discrete.cpp**, which additionally includes the computations of second derivative state matrices.

- The dynamics functions for the Dynamic Spacetime Optimization approach, used to compute the crucial G, C, and M terms utilized in all the computations and to step the physics simulation forward by one time step, are found in **SpacetimeDynamics_Analytic.cpp** and **SpacetimeDynamics_Numeric.cpp**. Just like with the derivative functions, the numeric versions of these functions are not being called since the analytic implementation never got off the ground. However, the main difference here between the two versions is that the numeric dynamics functions utilize the values retrieved from the PhysX engine to compute the dynamic state values directly, whereas the analytic version uses the state variables to compute these according to analytic formulas.

- The dynamics functions for the Discrete Online Trajectory Optimization approach are alternatively found in **SpacetimeDynamics_Discrete.cpp**.

- The file **SpacetimeState.cpp** contains functions for retrieving and setting the state of the system. It is essentially a set of translation functions from the form of the state vector X that is required for our optimization approach and the data returned and required by the PhysX API respectively. There are also two functions to save and restore the state of the system over a short period of time, which additionally account for the linear velocities of the system as defined by PhysX.

- The system itself is initialized and set up in **SpacetimeInit.cpp** and **Spacetime.cpp**. The first of these files is used to initialize the physics engine and define all rigid dynamic and static actors in the scene, as well as create the relevant joints between the actors. The second file contains the constructors for the Spacetime class.

- Any additional utility files are defined in the remaining files, such as some simple mathematical functions, the SvzMatrix/Algebra classes, and the Render/CameraUtil functions that were adapted from some PhysX example projects.

- A simple file for cleaning up the physics engine upon exiting the program is found in **SpacetimeCleanup.cpp**.

V. Discussion of Results

I regret to admit that, in the end, the results of this project are not the most stable, nor the most impressive. The video included with this final submission shows the iterative solver in action for the Approach 1. Approach 2 was not able to produce any results in the shortened time period allotted to its implementation.

The first pass of the system shows the PD controller guiding the system along a non-optimal path through space and time to approximate the end goal as best as possible. This part of the implementation works very well and demonstrates the success of the analytic dynamics equations utilized to advance the physics of the scene. The successive runs of the system over the time frame represent the iterations of the dynamic optimization process. Each successive run should, in theory, demonstrate a series of input torque vectors that more closely resembles the energy-minimizing optimum solution, until the system actually converges to this solution at which point the simulation terminates and such an optimal sequence is returned. As you may notice, despite an incredible amount of time and effort put into this project and the painstaking mathematical checks on the analytic solutions, the dynamic optimization of the forward/backward sweep approach did not deliver the optimal results I was hoping to achieve. The system is unstable and, rather than converging to a successful solution, seems to diverge to instability and explode. Neither I nor Dr. Lane understands why this situation is occurring as the dynamic optimization mathematics appears, with every check, to be correct and stable. It is a disappointing find at the end of the day, but perhaps demonstrative of this method's inability to successfully optimize a constrained system such as this one.

VI. Discussion of Future Work

Especially since this project did not reach its ultimate and potentially lofty initial goals, I do intend to continue working on this implementation to see if I can't get some stable results out of the analytic system. If such a point comes along that this endeavor is possible and successful, and then my project should lend itself nicely to the use of the ADOL-C library in the more generalized numerical computation of such values as listed above (though I admit this may certainly require some potentially major adjustments to the dynamics processing and a very sizeable time commitment). Eventually, as was thrown around at the start of the semester, the next step beyond that would be to port this project to the GPU to exploit the parallelism of the hardware in any way possible. A real time iterative solution such as this could have novel and exciting applications in games that learn as players progress through the various challenges.

VII. Works Cited

[A05] Asada, H. Harry. Introduction to Robotics. Massachusetts Institute of Technology. 2005.

[C92] Cohen, M. F. Interactive Spacetime Control for Animation, Computer Graphics 26(2) (July 1992), pp. 293-302

[G07] Grupen, Roderic. Newton/Euler Equations. 2007.

[GMW81] Phillip Gill, Walter Murray, and Margret Wright. Practical Optimization. Academic Press, New York, NY, 1981

[H11] Høifødt, Herman. Dynamic Modeling and Simulation of Robot Manipulators: The Newton-Euler Formulation. Norwegian University of Science and Technology, Department of Engineering Cybernetics. June 2011.

[L96] Liu, Zicheng. Efficient Animation Techniques Balancing Both User Control and Physical Realism. Princeton University. November 1996.

[LW] Lenhart, S. and Workman, J. T. Rutgers University.

[NM93] Ngo, J. T. and Marks, J. Spacetime Constraints Revisited, SIGGRAPH Proceedings (1993), pp. 343-350

[P12] Poulsen, Niels Kjolstadt. Dynamic Optimization, Informatics and Mathematical Modelling, The Technical University of Denmark. 2012.

[S01] Shaw, Taylor. Spacetime Constraints Attempted. 14 September 2001.

[TET12] Tassa, Y., Erez, T. and Todorv, E. Synthesis and Stabilization of Complex Behaviors through Online Trajectory Optimization, International Conference on Intelligent Robots and Systems. October 7-12, 2012. Vilamoura, Algarve, Portugal.

[W04] Winzell, Pär. An Implementation of the Spacetime Constraints Approach to the Synthesis of Realistic Motion. 26 November 2004.

[WG12] A. Walther and A. Griewank: Getting started with ADOL-C. In U. Naumann und O. Schenk, Combinatorial Scientific Computing, Chapman-Hall CRC Computational Science, pp. 181-202 (2012).

[WK88] Witkin, A. and Kass, M., Spacetime Constraints, Computer Graphics 22(4) (August 1988), pp. 159-168

VIII. Appendix A: State-Specific Matrix Definitions and Derivatives

This section outlines the analytical computations and relationships between the state variables and the dynamics formulas as applied in both the Dynamic Spacetime Optimization algorithmic approach (Approach 1) and the Discrete Online Trajectory Optimization algorithmic approach (Approach 2).

The first three matrix definitions are adopted from the aforementioned Asada reading to describe the gravitational and centripetal/Coriolis effects on the systems, as well as the mass matrix. The following sections outline my computations of these matrices partial derivatives and the combinations thereof to define the optimization.

a. Analytic matrix definitions for 2-joint, 1-DOF articulated system

$$G = \begin{bmatrix} G_1 \\ G_2 \end{bmatrix} = \begin{bmatrix} m_1 g l_{c1} \cos\theta_1 + m_2 g (l_{c2} \cos(\theta_1 + \theta_2) + l_1 \cos\theta_1) \\ m_2 g l_{c2} \cos(\theta_1 + \theta_2) \end{bmatrix}$$

$$C = \begin{bmatrix} C_1 \\ C_2 \end{bmatrix} = \begin{bmatrix} -m_2 l_1 l_{c2} \sin\theta_2 \, \dot{\theta}_2{}^2 - 2 m_2 l_1 l_{c2} \sin\theta_2 \, \dot{\theta}_1 \dot{\theta}_2 \\ m_2 l_1 l_{c2} \sin\theta_2 \, \dot{\theta}_1{}^2 \end{bmatrix}$$

$$M = \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} = \begin{bmatrix} m_1 l_{c1}{}^2 + I_1 + m_2 (l_1{}^2 + l_{c2}{}^2 + 2 l_1 l_{c2} \cos\theta_2) + I_2 & m_2 \left(l_{c2}{}^2 + l_1 l_{c2} \cos\theta_2 \right) + I_2 \\ m_2 \left(l_{c2}{}^2 + l_1 l_{c2} \cos\theta_2 \right) + I_2 & m_2 l_{c2}{}^2 + I_2 \end{bmatrix}$$

b. Analytic matrix first order partial derivatives for 2-joint, 1-DOF articulated system

$$\frac{\partial G}{\partial \theta_1} = \begin{bmatrix} \frac{\partial G_1}{\partial \theta_1} \\ \frac{\partial G_2}{\partial \theta_1} \end{bmatrix} = \begin{bmatrix} -m_1 l_{c1} g \sin\theta_1 - m_2 g l_{c2} (\sin\theta_1 \cos\theta_2 + \cos\theta_1 \sin\theta_2) - m_2 g l_1 \sin\theta_1 \\ -m_2 g l_{c2} (\sin\theta_1 \cos\theta_2 + \cos\theta_1 \sin\theta_2) \end{bmatrix}$$

$$\frac{\partial G}{\partial \theta_2} = \begin{bmatrix} \frac{\partial G_1}{\partial \theta_2} \\ \frac{\partial G_2}{\partial \theta_2} \end{bmatrix} = \begin{bmatrix} -m_2 g l_{c2}(\cos\theta_1 \sin\theta_2 + \sin\theta_1 \cos\theta_2) \\ -m_2 g l_{c2}(\cos\theta_1 \sin\theta_2 + \sin\theta_1 \cos\theta_2) \end{bmatrix}$$

$$\frac{\partial G}{\partial \dot{\theta}_2} = \begin{bmatrix} \frac{\partial G_1}{\partial \dot{\theta}_1} \\ \frac{\partial G_2}{\partial \dot{\theta}_1} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\frac{\partial G}{\partial \dot{\theta}_2} = \begin{bmatrix} \frac{\partial G_1}{\partial \dot{\theta}_2} \\ \frac{\partial G_2}{\partial \dot{\theta}_2} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\frac{\partial C}{\partial \theta_1} = \begin{bmatrix} \frac{\partial C_1}{\partial \theta_1} \\ \frac{\partial C_2}{\partial \theta_1} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\frac{\partial C}{\partial \theta_2} = \begin{bmatrix} \frac{\partial C_1}{\partial \theta_2} \\ \frac{\partial C_2}{\partial \theta_2} \end{bmatrix} = \begin{bmatrix} -m_2 l_1 l_{c2} \cos\theta_2 \left(\dot{\theta_2}^2 + 2\dot{\theta}_1 \dot{\theta}_2\right) \\ m_2 l_1 l_{c2} \cos\theta_2 \, \dot{\theta_1}^2 \end{bmatrix}$$

$$\frac{\partial C}{\partial \dot{\theta}_2} = \begin{bmatrix} \frac{\partial C_1}{\partial \dot{\theta}_1} \\ \frac{\partial C_2}{\partial \dot{\theta}_1} \end{bmatrix} = \begin{bmatrix} -2m_2 l_1 l_{c2} \sin\theta_2 \, \dot{\theta}_2 \\ 2m_2 l_1 l_{c2} \sin\theta_2 \, \dot{\theta}_1 \end{bmatrix}$$

$$\frac{\partial C}{\partial \dot{\theta}_2} = \begin{bmatrix} \frac{\partial C_1}{\partial \dot{\theta}_2} \\ \frac{\partial C_2}{\partial \dot{\theta}_2} \end{bmatrix} = \begin{bmatrix} -2m_2 l_1 l_{c2} \sin\theta_2 \left(\dot{\theta}_2 + \dot{\theta}_1\right) \\ 0 \end{bmatrix}$$

$$\frac{\partial M}{\partial \theta_1} = \begin{bmatrix} \frac{\partial M_{11}}{\partial \theta_1} & \frac{\partial M_{12}}{\partial \theta_1} \\ \frac{\partial M_{21}}{\partial \theta_1} & \frac{\partial M_{22}}{\partial \theta_1} \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

$$\frac{\partial M}{\partial \theta_2} = \begin{bmatrix} \frac{\partial M_{11}}{\partial \theta_2} & \frac{\partial M_{12}}{\partial \theta_2} \\ \frac{\partial M_{21}}{\partial \theta_2} & \frac{\partial M_{22}}{\partial \theta_2} \end{bmatrix} = \begin{bmatrix} -2m_2 l_1 l_{c2} \sin\theta_2 & -m_2 l_1 l_{c2} \sin\theta_2 \\ -m_2 l_1 l_{c2} \sin\theta_2 & 0 \end{bmatrix}$$

$$\frac{\partial M}{\partial \dot{\theta}_1} = \begin{bmatrix} \frac{\partial M_{11}}{\partial \dot{\theta}_1} & \frac{\partial M_{12}}{\partial \dot{\theta}_1} \\ \frac{\partial M_{21}}{\partial \dot{\theta}_1} & \frac{\partial M_{22}}{\partial \dot{\theta}_1} \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

$$\frac{\partial M}{\partial \dot{\theta}_2} = \begin{bmatrix} \frac{\partial M_{11}}{\partial \dot{\theta}_2} & \frac{\partial M_{12}}{\partial \dot{\theta}_2} \\ \frac{\partial M_{21}}{\partial \dot{\theta}_2} & \frac{\partial M_{22}}{\partial \dot{\theta}_2} \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

$$\frac{\partial M^{-1}}{\partial \theta_1} = -M^{-1} \frac{\partial M}{\partial \theta_1} M^{-1}$$

$$\frac{\partial M^{-1}}{\partial \theta_2} = -M^{-1} \frac{\partial M}{\partial \theta_2} M^{-1}$$

$$\frac{\partial M^{-1}}{\partial \dot{\theta}_1} = -M^{-1} \frac{\partial M}{\partial \dot{\theta}_1} M^{-1}$$

$$\frac{\partial M^{-1}}{\partial \dot{\theta}_2} = -M^{-1} \frac{\partial M}{\partial \dot{\theta}_2} M^{-1}$$

c. Analytic matrix second order partial derivatives for 2-joint, 1-DOF articulated system

$$\frac{\partial^2 M^{-1}}{\partial X^2} = \left[\frac{\partial^2 M^{-1}}{\partial X \partial \theta_1}, \frac{\partial^2 M^{-1}}{\partial X \partial \theta_2}, \frac{\partial^2 M^{-1}}{\partial X \partial \dot{\theta}_1}, \frac{\partial^2 M^{-1}}{\partial X \partial \dot{\theta}_2}\right]$$

$$\frac{\partial^2 M^{-1}}{\partial X \partial \theta_1} = \left[\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}\right]$$

$$\frac{\partial^2 M^{-1}}{\partial X \partial \theta_2} = \left[\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} -2m_2 l_1 l_{c2} \cos\theta_2 & -m_2 l_1 l_{c2} \cos\theta_2 \\ -m_2 l_1 l_{c2} \cos\theta_2 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}\right]$$

$$\frac{\partial^2 M^{-1}}{\partial X \partial \dot{\theta}_1} = \left[\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}\right]$$

$$\frac{\partial^2 M^{-1}}{\partial X \partial \dot{\theta}_2} = \left[\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}\right]$$

$$\frac{\partial^2 C}{\partial X^2} = \left[\frac{\partial^2 C}{\partial X \partial \theta_1}, \frac{\partial^2 C}{\partial X \partial \theta_2}, \frac{\partial^2 C}{\partial X \partial \dot{\theta}_1}, \frac{\partial^2 C}{\partial X \partial \dot{\theta}_2}\right]$$

$$\frac{\partial^2 C}{\partial X \partial \theta_1} = \left[\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \end{bmatrix}\right]$$

$$\frac{\partial^2 C}{\partial X \partial \theta_2} = \left[\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} m_2 l_1 l_{c2} \sin\theta_2 \left(\dot{\theta}_2^2 + 2\dot{\theta}_1 \dot{\theta}_2\right) \\ -m_2 l_1 l_{c2} \sin\theta_2 \dot{\theta}_1^2 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \end{bmatrix}\right]$$

$$\frac{\partial^2 C}{\partial X \partial \dot{\theta}_1} = \left[\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}-2m_2 l_1 l_{c2} \cos\theta_2 \, \dot{\theta}_2\\ 2m_2 l_1 l_{c2} \cos\theta_2 \, \dot{\theta}_1\end{bmatrix}, \begin{bmatrix}0\\ 2m_2 l_1 l_{c2} \sin\theta_2\end{bmatrix}, \begin{bmatrix}-2m_2 l_1 l_{c2} \sin\theta_2\\ 0\end{bmatrix}\right]$$

$$\frac{\partial^2 C}{\partial X \partial \dot{\theta}_2} = \left[\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}-2m_2 l_1 l_{c2} \cos\theta_2 \, (\dot{\theta}_1 + \dot{\theta}_2)\\ 0\end{bmatrix}, \begin{bmatrix}-2m_2 l_1 l_{c2} \sin\theta_2\\ 0\end{bmatrix}, \begin{bmatrix}-2m_2 l_1 l_{c2} \sin\theta_2\\ 0\end{bmatrix}\right]$$

$$\frac{\partial^2 G}{\partial X^2} = \left[\frac{\partial^2 G}{\partial X \partial \theta_1}, \frac{\partial^2 G}{\partial X \partial \theta_2}, \frac{\partial^2 G}{\partial X \partial \dot{\theta}_1}, \frac{\partial^2 G}{\partial X \partial \dot{\theta}_2}\right]$$

$$\frac{\partial^2 G}{\partial X \partial \theta_1} = \left[\begin{bmatrix}-m_1 g l_{c1} \cos\theta_1 - m_2 g (l_{c2} \cos(\theta_1 + \theta_2) - l_1 \cos\theta_1)\\ -m_2 g l_{c2} \cos(\theta_1 + \theta_2)\end{bmatrix}, \begin{bmatrix}-m_2 g l_{c2} \cos(\theta_1 + \theta_2)\\ -m_2 g l_{c2} \cos(\theta_1 + \theta_2)\end{bmatrix}, \begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}0\\0\end{bmatrix}\right]$$

$$\frac{\partial^2 G}{\partial X \partial \theta_2} = \left[\begin{bmatrix}-m_2 g l_{c2} \cos(\theta_1 + \theta_2)\\ -m_2 g l_{c2} \cos(\theta_1 + \theta_2)\end{bmatrix}, \begin{bmatrix}-m_2 g l_{c2} \cos(\theta_1 + \theta_2)\\ -m_2 g l_{c2} \cos(\theta_1 + \theta_2)\end{bmatrix}, \begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}0\\0\end{bmatrix}\right]$$

$$\frac{\partial^2 G}{\partial X \partial \dot{\theta}_1} = \left[\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}0\\0\end{bmatrix}\right]$$

$$\frac{\partial^2 G}{\partial X \partial \dot{\theta}_2} = \left[\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}0\\0\end{bmatrix}\right]$$

IX. Appendix B: Analytic Definitions for Use in Dynamic Spacetime Optimization Implementation

Primary Functions, where f is $\dot{X}$ and L_t is the cost at time step t

$$f(X,u) = \begin{bmatrix} f_1(X,u) \\ f_2(X,u) \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} X_3 \\ X_4 \end{bmatrix} \\ M^{-1}[\tau - C - G] \end{bmatrix}$$

$$L_t(X_t, u_t) = {u_t}^T u_t$$

Analytic matrix full derivatives for 2-joint, 1-DOF articulated system

$$f_x = \frac{\partial f}{\partial X} = \begin{bmatrix} \frac{\partial f_1}{\partial X} \\ \frac{\partial f_2}{\partial X} \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial \theta_1} & \frac{\partial f_1}{\partial \theta_2} & \frac{\partial f_1}{\partial \dot{\theta}_1} & \frac{\partial f_1}{\partial \dot{\theta}_2} \\ \frac{\partial}{\partial X}(M^{-1}[u - C - G]) \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ \left(\frac{\partial M^{-1}}{\partial X} u\right) - \left(\frac{\partial M^{-1}}{\partial X} C + M^{-1} \frac{\partial C}{\partial X}\right) - \left(\frac{\partial M^{-1}}{\partial X} G + M^{-1} \frac{\partial G}{\partial X}\right) \end{bmatrix}$$

$$f_u = \frac{\partial f}{\partial u} = \begin{bmatrix} \frac{\partial f_1}{\partial u} \\ \frac{\partial f_2}{\partial u} \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial u_1} & \frac{\partial f_1}{\partial u_2} \\ \frac{\partial}{\partial u}(M^{-1}[u - C - G]) \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ M^{-1} \end{bmatrix}$$

$$L_x = \frac{\partial L}{\partial X} = \begin{bmatrix} \frac{\partial L}{\partial \theta_1} & \frac{\partial L}{\partial \theta_2} & \frac{\partial L}{\partial \dot{\theta}_1} & \frac{\partial L}{\partial \dot{\theta}_2} \end{bmatrix} = [0 \quad 0 \quad 0 \quad 0]$$

$$L_u = \frac{\partial L}{\partial u} = \begin{bmatrix} \frac{\partial L}{\partial u_1} & \frac{\partial L}{\partial u_2} \end{bmatrix} = [u_1 \quad u_2]$$

Initial Input Sequence Guess (PD Controller)

$$u_t = C_t + G_t + M\left(K_P(\theta_d - \theta_t) - K_v \dot{\theta}_t\right)$$

Optimization Equations (Repeated from Above)

$$\dot{X}_t = f(X_t, u_t)$$

$$\dot{\lambda_t}^T = -\frac{\partial L}{\partial X} - {\lambda_t}^T \frac{\partial f}{\partial X}$$

$${u_t}^T = -{\lambda_t}^T \frac{\partial f}{\partial u}$$

X. Appendix C: Analytic Definitions for Use in Discrete Online Trajectory Optimization

Discrete time definition of dynamics simulation function

$$x_{t+1} = f(x_t, u_t) = x_t + f'(x_t, u_t) = \begin{bmatrix} (\theta_1)_t \\ (\theta_2)_t \\ (\dot{\theta}_1)_t \\ (\dot{\theta}_2)_t \end{bmatrix} + \Delta t \begin{bmatrix} (\dot{\theta}_1)_t \\ (\dot{\theta}_2)_t \\ {M_t}^{-1}[u_t - C_t - G_t] \end{bmatrix}$$

Cost function

$$L_t(X_t, u_t) = {u_t}^T u_t$$

First Derivative Computations for Synthesis and Simulation Implementation

$$f_x = \frac{\partial f}{\partial X} = I_{4\times4} + \Delta t \begin{bmatrix} \frac{\partial f'_1}{\partial X} \\ \frac{\partial f'_2}{\partial X} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} + \Delta t \begin{bmatrix} \frac{\partial f'_1}{\partial \theta_1} & \frac{\partial f'_1}{\partial \theta_2} & \frac{\partial f'_1}{\partial \dot{\theta}_1} & \frac{\partial f'_1}{\partial \dot{\theta}_2} \\ \frac{\partial}{\partial X}(M^{-1}[u - C - G]) \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} + \Delta t \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ \left(\frac{\partial M^{-1}}{\partial X} u\right) - \left(\frac{\partial M^{-1}}{\partial X} C + M^{-1} \frac{\partial C}{\partial X}\right) - \left(\frac{\partial M^{-1}}{\partial X} G + M^{-1} \frac{\partial G}{\partial X}\right) \end{bmatrix}$$

$$f_u = \frac{\partial f}{\partial u} = \begin{bmatrix} \frac{\partial f_1}{\partial u} \\ \frac{\partial f_2}{\partial u} \end{bmatrix} = \Delta t \begin{bmatrix} \frac{\partial f_1}{\partial u_1} & \frac{\partial f_1}{\partial u_2} \\ \frac{\partial}{\partial u}(M^{-1}[u - C - G]) \end{bmatrix} = \Delta t \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ H^{-1} \end{bmatrix}$$

$$L_x = \frac{\partial L}{\partial X} = \begin{bmatrix} \frac{\partial L}{\partial \theta_1} & \frac{\partial L}{\partial \theta_2} & \frac{\partial L}{\partial \dot{\theta}_1} & \frac{\partial L}{\partial \dot{\theta}_2} \end{bmatrix} = [0 \quad 0 \quad 0 \quad 0]$$

$$L_u = \frac{\partial L}{\partial u} = \begin{bmatrix} \frac{\partial L}{\partial u_1} & \frac{\partial L}{\partial u_2} \end{bmatrix} = [u_1 \quad u_2]$$

$$L_{uu} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$L_{ux} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$L_{xu} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

$$L_{xx} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$f_{uu} = \begin{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \end{bmatrix}$$

$$f_{ux} = \begin{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \frac{\partial M^{-1}}{\partial \theta_2 X} \end{bmatrix}, \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \end{bmatrix}$$

$$f_{xx} = \Delta t \begin{bmatrix} 0 \\ \frac{\partial f'_x}{\partial X} \end{bmatrix} = \Delta t \begin{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{\partial f'_x}{\partial \theta_1} \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{\partial f'_x}{\partial \theta_2} \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{\partial f'_x}{\partial \dot{\theta}_1} \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{\partial f'_x}{\partial \dot{\theta}_2} \end{bmatrix} \end{bmatrix}$$

where for any variable $y \in \{\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2\}$

$$\frac{\partial f'_x}{\partial y} = \frac{\partial^2 M^{-1}}{\partial X \partial y}(u - C - G) - 2 * \frac{\partial M^{-1}}{\partial y}\left(\frac{\partial C}{\partial X} + \frac{\partial G}{\partial X}\right) - M^{-1}\left(\frac{\partial^2 C}{\partial X \partial y} + \frac{\partial^2 G}{\partial X \partial y}\right)$$

$$f_{xu} = \Delta t \begin{bmatrix} 0 \\ \frac{\partial f'_x}{\partial u} \end{bmatrix} = \Delta t \begin{bmatrix} \begin{bmatrix} 0 \\ \frac{\partial f'_x}{\partial u_1} \end{bmatrix} & \begin{bmatrix} 0 \\ \frac{\partial f'_x}{\partial u_2} \end{bmatrix} \end{bmatrix}$$

$$= \Delta t \begin{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \left(\frac{\partial M^{-1}}{\partial X \partial \theta_1}\right)_{11} & \left(\frac{\partial M^{-1}}{\partial X \partial \theta_2}\right)_{11} & \left(\frac{\partial M^{-1}}{\partial X \partial \dot{\theta}_1}\right)_{11} & \left(\frac{\partial M^{-1}}{\partial X \partial \dot{\theta}_2}\right)_{11} \\ \left(\frac{\partial M^{-1}}{\partial X \partial \theta_1}\right)_{21} & \left(\frac{\partial M^{-1}}{\partial X \partial \theta_2}\right)_{21} & \left(\frac{\partial M^{-1}}{\partial X \partial \dot{\theta}_1}\right)_{21} & \left(\frac{\partial M^{-1}}{\partial X \partial \dot{\theta}_2}\right)_{21} \end{bmatrix} & \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \left(\frac{\partial M^{-1}}{\partial X \partial \theta_1}\right)_{12} & \left(\frac{\partial M^{-1}}{\partial X \partial \theta_2}\right)_{12} & \left(\frac{\partial M^{-1}}{\partial X \partial \dot{\theta}_1}\right)_{12} & \left(\frac{\partial M^{-1}}{\partial X \partial \dot{\theta}_2}\right)_{12} \\ \left(\frac{\partial M^{-1}}{\partial X \partial \theta_1}\right)_{22} & \left(\frac{\partial M^{-1}}{\partial X \partial \theta_2}\right)_{22} & \left(\frac{\partial M^{-1}}{\partial X \partial \dot{\theta}_1}\right)_{22} & \left(\frac{\partial M^{-1}}{\partial X \partial \dot{\theta}_2}\right)_{22} \end{bmatrix} \end{bmatrix}$$

Backwards Sweep

$$V(t_f) = L\left(X_{t_f}, u_{t_f}\right) = 0.5 * \left(X_d - X_{t_f}\right)^T \left(X_d - X_{t_f}\right)$$

$$Q_x = L_x + {f_x}^T V_x'$$

$$Q_u = L_u + {f_u}^T V_x'$$

$$Q_{xx} = L_{xx} + {f_x}^T V_{xx}' f_x + (f_{xx})^T (V_x')^T$$

$$Q_{uu} = L_{uu} + {f_u}^T V_{xx}' f_u + (f_{uu})^T (V_x')^T$$

$$Q_{ux} = L_{ux} + {f_u}^T V_{xx}' f_x + (f_{ux})^T (V_x')^T$$

$$Q_{xu} = L_{xu} + {f_x}^T V_{xx}' f_u + (f_{xu})^T (V_x')^T$$

$$k(t) = -Q_{uu}^{-1} {Q_u}^T$$

$$K(t) = -Q_{uu}^{-1} Q_{ux}$$

$$\Delta V(t) = -\frac{1}{2} Q_u Q_{uu}^{-1} Q_u$$

$$V_x(t) = Q_x - Q_u Q_{uu}^{-1} Q_{ux}$$

$$V_{xx}(t) = Q_{xx} - Q_{xu} Q_{uu}^{-1} Q_{ux}$$

Forwards Sweep

$$\hat{X}(t_0) = X_0$$

$$\hat{u}(t) = u(t) + k(t)(\hat{x}(t) - x(t))$$

$$\hat{x}(t+1) = f(\hat{x}(t), \hat{u}(t))$$