Discrete Mechanical Interpolation of Keyframes

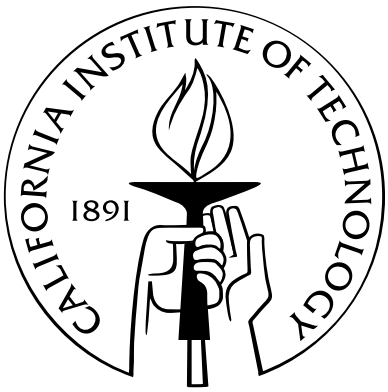
Thesis by

Weiwei Yang

In Partial Fulfillment of the Requirements

for the Degree of

Master of Science



California Institute of Technology

Pasadena, California

2007

(Submitted Dec 31, 2007)

Acknowledgments

To no absolutes; To Absolut.

This page intentionally left blank.

Abstract

Advances in computer hardware and software have made it possible to use physical simulation methods to add unprecedented degrees of realism to computer animations. Unfortunately, the reliance on such methods makes it difficult or even impossible to control the results in order to achieve artistic goals. We present a method for allowing artistic control of physical realism by interpolating between given keyframes. The method is based on the discrete Lagrange-d'Alembert principle; we introduce additional *ghost forces* that are calculated to bring the system into the configuration requested by the artist. We derive a cost function that when minimized ensures the corresponding motion to be smooth and physical looking. We describe the implementation and show an example of animating a multi-particle mass-spring system in 3D using our method.

This page intentionally left blank.

Contents

This page intentionally left blank.

List of Figures

This page intentionally left blank.

Chapter 1

Introduction

The strive for realism in computer animation has made the task of animating objects such as hair, smoke and water increasingly difficult and impractical. On the other hand, simulating such objects using physics leaves little room for artistry: once initial conditions are set, the resulting simulations are at the complete mercy of the underlying physics and numerical integrators. Furthermore, these systems are usually highly nonlinear, and therefore make tweaking initial conditions to achieve desired results a black art. For example, a simulated skirt may flow realistically around a character, but it may also contain visually distracting creases that are impossible to iron out without changing the material properties of the skirt. What it is needed is a scheme that allows artistic control of physical realism.

In this thesis, we present an interpolation scheme based on the discrete Lagrange-d'Alembert principle. Our goal is to provide a range of compromises between artistic control and the laws of physics. We expect to see stylized, yet "physical looking" animation.

In chapter 2, we give an overview of Lagrangian dynamics, which is the foundation for our method. In chapter 3, we describe the Hamilton-Pontryagin-based simulation method, which uses Lagrangian dynamics to simulate physical materials such as a rubbery, elastic bunny. In chapter 4 we present our method for interpolating keyframes based on discrete mechanics. Our implementation details are covered in chapter 5. Chapter 6 contains results of animating coupled spring-mass systems using our method.

Chapter 2

Overview of Lagrangian Dynamics

What follows is a brief overview of Lagrangian dynamics; for a more detailed description and derivation, please refer to Matthew West's Ph.D. thesis [26] on variational integrators.

2.1 Continuous Lagrangian dynamics

Let $q = (q^1, q^2, q^3, \ldots, q^n)$ be the instantaneous configuration, or state, of a dynamical system in $\mathbb{R}^n$, and let $\dot{q}$ be the time derivative of q, $\dot{q}(t) = dq/dt$. The Lagrangian L of the system is given as a function of q and $\dot{q}$ and is defined as kinetic energy, K, minus potential energy, W:

$$L(q, \dot{q}) = K(\dot{q}(t)) - W(q(t)). \tag{2.1}$$

The principle of stationary action states that, when the system is conservative, the motion of the system from time t_1 to t_2 is such that the line integral

$$I(q) = \int_{t_1}^{t_2} L(q(t), \dot{q}(t))\, dt, \tag{2.2}$$

which is called *action*, is stationary with respect to the variation of q for the correct path of the motion between the fixed end points $q(t_1) = q_1$ and $q(t_2) = q_2$. In other words, the motion is such that the variation of the line integral I for fixed t_1 and t_2 is zero, as shown in figure (2.1):

$$\delta I(q) = \delta \int_{t_1}^{t_2} L(q(t), \dot{q}(t))\, dt = 0 \tag{2.3}$$

Computing $\delta I(q)$ using integration by parts, we obtain the following:

$$\begin{aligned}
\delta I(q) &= \delta \int_{t_1}^{t_2} L(q(t), \dot{q}(t))\, dt \\
&= \int_{t_1}^{t_2} \left[\frac{\partial L}{\partial q} \cdot \delta q + \frac{\partial L}{\partial \dot{q}} \cdot \delta \dot{q} \right] dt \\
&= \int_{t_1}^{t_2} \left[\frac{\partial L}{\partial q} - \frac{d}{dt} \left(\frac{\partial L}{\partial \dot{q}} \right) \right] \delta q \; dt + \left. \frac{\partial L}{\partial \dot{q}} \cdot \delta q \right|_{t_1}^{t_2}
\end{aligned}$$

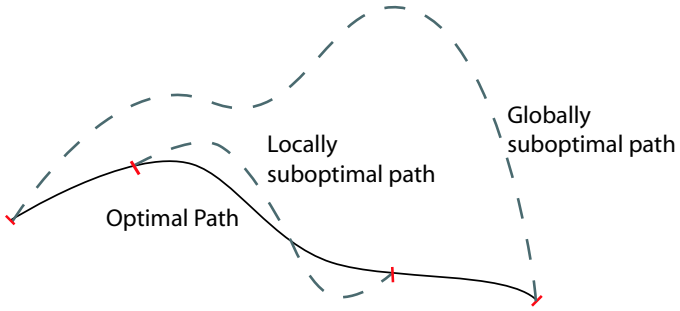


Figure 2.1: Principle of stationary action.

Since the endpoints of $q(t)$ are held fixed with respect to all variations $\delta q(t)$ (i.e., $\delta q(t_1) = \delta q(t_2) = 0$), the last term of the above equation becomes 0, thus yielding the *Euler-Lagrange equation*:

$$\frac{\partial L(q, \dot{q})}{\partial q} - \frac{d}{dt} \left(\frac{\partial L(q, \dot{q})}{\partial \dot{q}} \right) = 0 \tag{2.4}$$

Example: Let $K = {}^1\!/_2\, \dot{q}^T M \dot{q}$, where M is a mass matrix. Then (2.4) becomes $M\ddot{q} = -\nabla W(q)$, which is simply Newton's second law: the sum of all forces equals mass times acceleration. As we can see from this example, the Euler-Lagrange formulation is equivalent to the formulation used in Newtonian mechanics.

2.2 Discrete Lagrangian

A motion $q(t)$ of a system for $t \in [0, T]$ can be discretized as a sequence of states $Q \equiv \{q_k\}_{k=0}^{N}$. Taking two adjacent states q_0 and q_1 that are Δt apart, where $q_0 = q(t_0)$ and $q_1 = q(t_0 + \Delta t)$, we can approximate $\dot{q}$ as $(q_1 - q_0)/\Delta t$. Let $W_d(q_0, q_1)$ denotes the potential energy function, W, evaluated according to a quadrature based on q_0 and q_1. Then the *discrete Lagrangian*[1], $L_d(q_0, q_1, \Delta t)$, is an

[1]This term could also be called an action, as it is a time integral of a Lagrangian; however, just as the term "discrete curvature" in computer graphics refers to a small local integral of a continuous curvature, we use this naming convention here.

approximation to the action integral $I(q)$, equation (2.2), along the segment of path between q_0 and q_1. One such approximation is done using the rectangle rule — the length of the interval, Δt, multiplied by the value of the integrand,

$$L_d(q_0, q_1) = \Delta t \ (K(\dot{q}_d) - W_d(q_0, q_1)). \tag{2.5}$$

The *discrete action* S_d, can be calculated by summing up the discrete Lagrangians of each pair within the sequence of discrete states $\{q_k\}_{k=0}^N$,

$$S_d(\{q_k\}) = \sum_{k=0}^{N-1} L_d(q_k, q_{k+1}). \tag{2.6}$$

And similarly to the continuous case, its variation becomes as follows:

$$\begin{aligned}
\delta S_d(\{q_k\}) &= \delta \sum_{k=0}^{N-1} L_d(q_k, q_{k+1}) \\
&= \sum_{k=0}^{N-1} [D_1 L_d(q_k, q_{k+1}) \cdot \delta q_k + D_2 L_d(q_k, q_{k+1}) \cdot \delta q_{k+1}] \\
&= \sum_{k=1}^{N-1} [D_2 L_d(q_{k-1}, q_k) + D_1 L_d(q_k, q_{k+1})] \cdot \delta q_k \\
&\quad + D_1 L_d(q_0, q_1) \cdot \delta q_0 + D_2 L_d(q_{N-1}, q_N) \cdot \delta q_N
\end{aligned}$$

Here $D_i L_d$ is the slot derivative with respect to the i^{th} argument of L_d. Again, the variation of the action is zero for any δq_k, with $\delta q_0 = \delta q_N = 0$; the *discrete Euler-Lagrange (DEL) equation*,

$$D_2 L_d(q_{k-1}, q_k) + D_1 L_d(q_k, q_{k+1}) = 0, \tag{2.7}$$

follows, and must hold for each $k \in [1, N-1]$.

Example: Given a discrete Lagrangian

$$L_d(q_{n-1}, q_n) = \Delta t \left[\frac{1}{2} \left(\frac{q_n - q_{n-1}}{\Delta t} \right)^T M \left(\frac{q_n - q_{n-1}}{\Delta t} \right) - W(q_n) \right]$$

it follows that

$$\begin{aligned}
D_2 L_d(q_{k-1}, q_k) &= M \left(\frac{q_k - q_{k-1}}{\Delta t} \right) \\
D_1 L_d(q_k, q_{k+1}) &= -\left[M \left(\frac{q_{k+1} - q_k}{\Delta t} \right) + \Delta t \cdot \nabla W(q_k) \right]
\end{aligned}$$

and (2.7) becomes the discretization of Newton's second law:

$$M\left(\frac{q_{k+1}-2q_k+q_{k-1}}{(\Delta t)^2}\right) = -\nabla W(q_k).$$

2.3 Lagrange-d'Alembert principle

Non-conservative systems, which contain forcing or dissipating terms, can be described using the *Lagrange-d'Alembert principle* in the continuous case, where $F(q(t), \dot{q}(t))$ is the non-conservative force:

$$\delta \int L(q(t), \dot{q}(t))\, dt + \int F(q(t), \dot{q}(t)) \cdot \delta q\, dt = 0 \tag{2.8}$$

The *discrete Lagrange-d'Alembert principle* states that

$$\delta \sum_{k=0}^{N-1} L_d(q_k, q_{k+1}) + \sum_{k=0}^{N-1} [F_d^-(q_k, q_{k+1}) \cdot \delta q_k + F_d^+(q_k, q_{k+1}) \cdot \delta q_{k+1}] = 0. \tag{2.9}$$

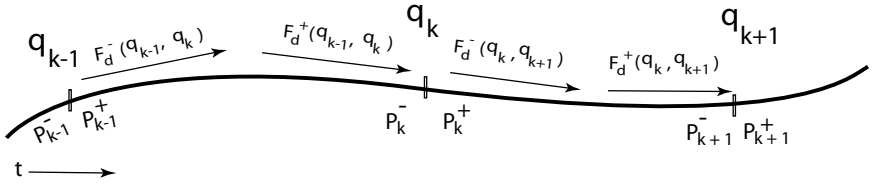


Figure 2.2: Early and late discrete force, F_d, and momenta P.

Here L_d is the discrete Lagrangian, F_d^- and F_d^+ are the discrete force, F_d, on the early and late side of q_k, shown in figure (2.2), where

$$F_d^-(q_k, q_{k+1}) \cdot \delta q_k + F_d^+(q_k, q_{k+1}) \cdot \delta q_{k+1} \approx \int_{t_k}^{t_{k+1}} F(q(t), \dot{q}(t)) \cdot \delta q\, dt. \tag{2.10}$$

Letting $F_k \equiv F_d^+(q_{k-1}, q_k) + F_d^-(q_k, q_{k+1})$, the *forced discrete Euler-Lagrange equation* can be derived:

$$D_1 L_d(q_k, q_{k+1}) + D_2 L_d(q_{k-1}, q_k) + F_k = 0 \tag{2.11}$$

Let's define the early and late momenta at time step k as follows:

$$p_k^+ = -D_1 L_d(q_k, q_{k+1}) - F_d^-(q_k, q_{k+1}) \tag{2.12}$$

$$p_k^- = D_2 L_d(q_{k-1}, q_k) + F_d^+(q_{k-1}, q_k) \tag{2.13}$$

$$\tag{2.14}$$

Now the DEL equation, (2.11), can be rewritten simply as

$$p_k^+ - p_k^- = 0. \tag{2.15}$$

Chapter 3

Hamilton-Pontryagin-Based Simulation

Let us now sketch the method for physically based simulation presented by Kharevych *et al.* [13] based on the *Hamilton-Pontryagin Principle*, which states that

$$\delta \int_0^T [p(\dot{q} - v) + L(q, v)]\, dt = 0 \tag{3.1}$$

For more details on this method, and our novel update through minimization please refer to the paper.

In the *Hamilton-Pontryagin Principle*, the configuration variable q, the velocity v, and the momentum p are treated as *independent* variables, that is $q(t)$, $v(t)$, and $p(t)$ vary independently with end-point conditions on $q(t)$. Taking the variation of the three independent variables, $\delta p(t), \delta q(t)$ and $\delta v(t)$ yields that

$$v = \dot{q},\ \frac{dp}{dt} = \frac{\partial L(q, v)}{\partial q}, \text{ and } p = \frac{\partial L(q, v)}{\partial v}. \tag{3.2}$$

Time Discretization Similar to the discretization of $q(t)$ as the sequence $\{q_k\}_{k=0}^N$, $v(t)$ and $p(t)$ are discretized by the sequences $\{v_k\}_{k=1}^N$ and $\{p_k\}_{k=1}^N$. Velocities v_{k+1} and momenta p_{k+1} are viewed as approximations *within* the intervals $[t_k, t_{k+1}]$, which are staggered with respect to the positions q_k. We will call h_k the *time step* between time t_k and t_{k+1}. Even though the time step can be adjusted throughout the computation based on standard time step control ideas, it is observed in practice that the numerics of a simulation are more stable when h_k is fixed in time.

Discrete Hamilton-Pontryagin Principle Once a discrete Lagrangian in the form of (2.5) is given, then its corresponding *discrete Hamilton-Pontryagin principle* can be expressed as follows:

$$\delta \sum_{k=0}^{N} \left[p_{k+1} \left(\frac{q_{k+1} - q_k}{h_k} - v_{k+1} \right) h_k + L^d(q_k, v_{k+1}) \right] = 0 \tag{3.3}$$

Discrete Variational Equations The discrete Hamilton-Pontryagin principle yields, upon taking discrete variations with respect to each state variable with fixed end q's:

$$\delta p : \qquad q_{k+1} - q_k = h_k v_{k+1} \tag{3.4}$$

$$\delta q : \qquad p_{k+1} - p_k = D_1 L^d(q_k, v_{k+1}) \tag{3.5}$$

$$\delta v : \qquad h_k \; p_{k+1} = D_2 L^d(q_k, v_{k+1}) \tag{3.6}$$

Here D_1 and D_2 denote the differentiation with respect to the first (q_k) and second (v_{k+1}) arguments of L^d.

Update Procedure Given a point in the discrete Pontryagin state-space (q_k, v_k, p_k), the above equations are to be solved for $(q_{k+1}, v_{k+1}, p_{k+1})$ in the following way:

- Replacing p_{k+1} by a function of p_k and $D_1 L^d(q_k, v_{k+1})$ based on (3.5) and (3.6).
- The resulting equation,

$$D_2 L^d(q_k, v_{k+1}) - h_k p_k - h_k D_1 L^d(q_k, v_{k+1}) = 0, \tag{3.7}$$

  can now be solved for v_{k+1} with any non-linear solver or used to compute v_{k+1} directly in the explicit case.

- q_{k+1} and p_{k+1} are found by applying (3.4) and (3.6) respectively.

Thus, given a system's discrete Lagrangian and initial configurations (q_0, v_0, p_0), one can simulate its motion by forward time stepping. Conversely, given the final configurations (q_f, v_f, p_f), one can simulate the system backward in time by doing backward time stepping.

Chapter 4

Discrete Mechanical Interpolation

It is useful that we can simulate a physical system based on its initial configurations according to the Hamilton-Pontryagin principle. The resulting simulation looks realistic. However, what happens if we are not completely satisfied with realistic results? There might be an artistic need to exaggerate a movement in some places to convey an emotional subtlety, or to slow down a movement to build a dramatic effect. The scheme described above would require the user to quantify such artistic elements, and represent them in the Lagrangian of the system, which may be highly inconvenient, or even physically impossible for many non-linear systems.

What we really need is not only a scheme capable of generating realistic motion based on the laws of physics, but one that is also capable of producing realistic-looking motion while *violating* the laws of physics. We will borrow the term *key frames* from computer animation, and define it as those states of motion we want our system to reach regardless of underlying physics.

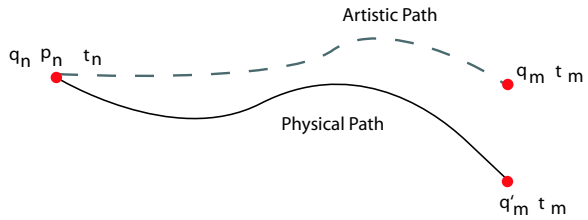


Figure 4.1: Simulation results differ from the desired key frame.

4.1 Ghost force

Let's look at two key frames q_n and q_m at times t_n and t_m from a system S, and try to construct a smooth, physical looking animation between them. Assume we know the Lagrangian, L_s, of the system. If S is simulated using the Hamilton-Pontryagin method described in the previous section,

starting with q_n and p_n as initial conditions, it may reach state q'_m at the time t_m instead of the desired q_m, as seen in figure (4.1). In order for the motion to deviate to q_m, some mysterious force, hereon known as the *ghost force, f*, must be acting on it. Similarly to q and p, f can be discretized as the sequence $\{f_k\}_{k=0}^{N}$. Even though little is known a priori about the nature of the ghost force, we know that the Lagrange-d'Alembert principle (2.11) must hold at every t_i.

If frame q_k lies midway in time between q_n and q_m, then $t_k - t_n = t_m - t_k = h$. According to (2.11), we can write down the following:

$$D_1 L_d(q_n, q_k) + p_n^- + F_n^+ + f_n^+ = 0 \tag{4.1}$$

$$D_1 L_d(q_k, q_m) + D_2 L_d(q_n, q_k) + F_k + f_k = 0 \tag{4.2}$$

$$-p_m^+ + D_2 L_d(q_k, q_m) + F_m^- + f_m^- = 0 \tag{4.3}$$

Here F denotes physical non-conservative forces from equation (2.11), which are discretized as the sequence $\{F_k\}_{k=0}^{N}$. f_i and p_i represent ghost force and momenta, whose values are all based on the time interval centered at frame q_i; see figure (4.2). $p_i = 1/2\,(p_i^+ + p_i^-)$, where p_i^+ and p_i^- can be calculated by following either (2.12) or from $\dot{q}$, if it is given, as $p = \partial L(q, \dot{q})/\partial \dot{q}$.

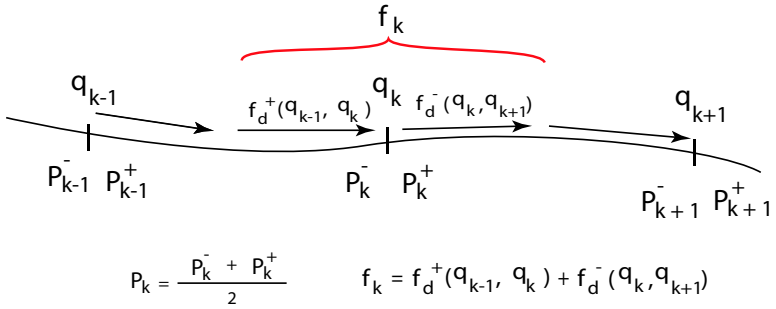


Figure 4.2: Discrete force and momentum distribution along a motion path.

4.2 DMOC method

The approach of using the Lagrange-d'Alembert principle to find the path of a system satisfying given conditions is not new. It is very similar to the discrete mechanics and optimal control (DMOC)[11] method used in solving optimal control problems for mechanical systems. In DMOC, a cost function,

$$J(q, f) = \int_{t_0}^{t_1} C(q(t), \dot{q}(t), f(t))\, dt, \tag{4.4}$$

is minimized under the constraint (2.8). This method is very efficient in finding a motion when the mechanical systems involved only have a few degrees of freedom, (below a few hundreds). However, it does not scale well and becomes inadequate for the types of physical systems we simulate in computer animation, where we could have up to a few thousand degrees of freedom.

4.3 Criteria for ghost forces

There are many possible ghost forces that could get the system to q_m from q_n. However, an f with large magnitude will most likely cause dramatic motion, which would not look physical; on the other hand, a non-smooth f will likely cause abrupt changes in the motion, which will not look smooth, figure 4.3.

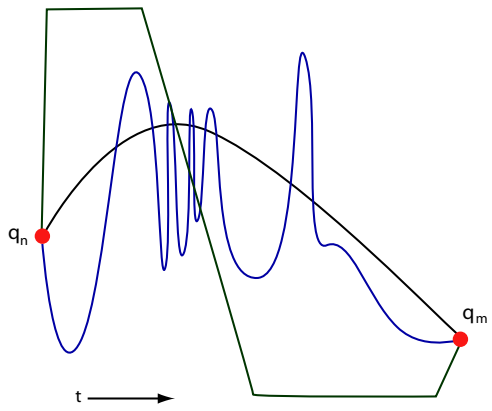


Figure 4.3: Depending on the ghost forces, there are many possible motions containing the two defined key frames, q_n and q_m; here are three examples. Most ghost forces will result in motions that are neither smooth nor physical looking. Smaller and smoother ghost forces tend to correspond to smoother and more physical looking paths.

Since it is impractical to solve the entire motion at once *à la* DMOC, we will take a more local approach by finding one state at a time that satisfies the DEL equation with the smallest discrete ghost force, f. In this way, we are finding the sequence $\{f_k\}_{k=0}^{N}$, where each discrete f_k is the smallest possible ghost within its respective interval. Furthermore, to ensure coherence of the motion, we want the f to be as smooth as possible.

4.4 Our approach

Let's start by establishing some relation between f_k and q_k. To do this, we go back to the DEL equations (4.1)–(4.3) and solve for each discrete f_k:

$$\begin{aligned} f_n^+ &= D_1L_d(q_n, q_k) + p_n^- + F_n^+ \\ f_k &= D_1L_d(q_k, q_m) + D_2L_d(q_n, q_k) + F_k \\ f_m^- &= -p_m^+ + D_2L_d(q_k, q_m) + F_m^- \end{aligned}$$

Each of these is of the form

$$f_i = p_i^+ - p_i^-. \tag{4.5}$$

This means that the smaller the difference between a pair of early and late discrete momenta, the smaller the ghost force within that interval. This should not come as a surprise, since when there is no ghost force present, $p_i^+ - p_i^- = 0$, (2.15). One way to minimize f_i is to minimize its norm; here we use the L^2 norm. Incidentally, one way to ensure smoothness of f is by imposing the condition that the difference between every adjacent f be as small as possible; in other words, the sum of all L^2 norm of the two differences, $f_{i-1} - f_i$ and $f_{i+1} - f_i$, is minimized.

We define:

$$E(p^-, p^+, \Delta t) = \left(\frac{p^+ - p^-}{\Delta t}\right)^2 \Delta t, \tag{4.6}$$

which is the L^2 norms of the difference $p^+ - p^-$, where p^+ and p^- are both associated with the same frame, normalized with respect to its corresponding time step. And

$$E_s(f^1, f^2, \Delta t) = \left(\frac{f^2 - f^1}{\Delta t}\right)^2 \Delta t, \tag{4.7}$$

the L^2 norm of the differences between f_k, and its two adjacent ghost forces, f_n^+ and f_m^-, also normalized with respect to its corresponding time step.

Let λ_s denote a constant that weighs the contribution of the smoothness versus the magnitude of the ghost forces in the objective function. It can be adjusted according to desired results.

Now we are ready to define a cost function J_{q_k}, that when minimized with respect to q_k will give us a sequence of small and smooth $\{f_k\}_{k=0}^N$.

$$\begin{aligned} J_{q_k} = \quad & E(p_n + F_n^+, -D_1L_d(q_n, q_k), \frac{h_k}{2}) + E(D_2L_d(q_n, q_k) + F_k, -D_1L_d(q_k, q_m), h_k) + \\ + \quad & E(D_2L_d(q_k, q_m) + F_m^-, p_m, \frac{h_k}{2}) + \lambda_s \left(E_s(f_n^+, \frac{f^k}{2}, h_k) + E_s(\frac{f^k}{2}, f_m^-, h_k) \right) \end{aligned} \tag{4.8}$$

There are three E terms in J_{q_k}, since a change in the location of q_k will affect not only p_k^+, p_k^-,

and their associated f_k, but also p_n^+, p_m^+, and f_n^+, f_m^-.

For systems with more than two key frames, we repeat the minimization process on every two adjacent key frames. To generate enough intermediate frames for a smooth animation, repeat the minimizing process in an iterative manner as illustrated in figure (4.4), treating frames solved from previous iterations as key frames. Please refer to the pseudocode at the end of section 5. Note, the computation of p_i involves the two frames adjacent to it, so even though only the two immediately neighboring frames appear in the expression (4.8), a total of four frames, two preceding and two following q_k, are actually taken into account in computing q_k, thus expanding the smoothness of the motion.

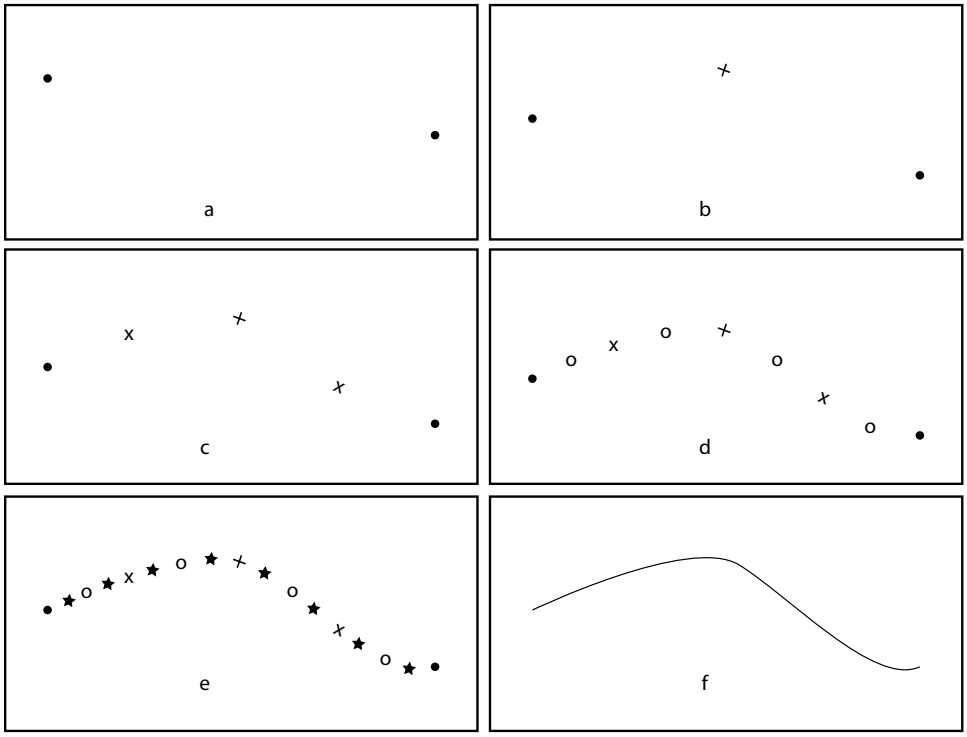


Figure 4.4: Iterative process of finding middle frames. a. The given two key-frames; b. First iteration, find the frame, half way in time between the two key frames; c. Second iteration, find the frames half way in time between the know frames; d. Third iteration, find the frames half way in time between the known frames; e. Fourth iteration; f. Constructed smooth motion path of the system.

Chapter 5

Implementation Details

5.1 Solver

Any solver capable of handling non-linear system could be used. In our implementation, we used the package Ipopt (Interior Point OPTimizer)[25]. Because the Hessian of the objective function (4.8) contains second order derivatives, its analytical expression can become quite lengthy and prone to human errors, especially for complicated potential functions $W(q_k)$. Therefore we opted to use the built-in numerical Hessian approximation function in Ipopt.

5.2 Regularization

Since the objective functions, (4.8), are usually non-linear, it is not always easy to minimize them. Another difficulty, especially in the initial steps where the time steps between frames, h_k, are large, is that the $-h_k \nabla W$ term dominates, making the gradient non-positive-definite. In the first few iterations, we thus need to help the solver converge by adding a regularization term to (4.8). The regularization term should vanish as h_k between frames becomes small, so that the regularization does not affect the motion of the system at finer levels. To ensure this, we write the regularization term with a scalar factor $c(h_k)$, as $c(h_k)E_{reg}$, where

$$\lim_{h_k \to 0} c(h_k) = 0.$$

There are many ways to regularize, depending on the system and desired results; any one method, or a combination of all described below can be used.

Spatial Laplacian: It is observed in most physical systems that neighboring parts tend to move in unison. When neighboring parts do not move in unison, the system exhibits shaking/jitteriness,

Calculate p_0^- and p_n^+ based on given initial conditions.
repeat
  Set all frames as valid
  for each valid frame i **do**
    Compute p_i based on its two adjacent valid frames.
    // Except the 1st key frame, $p_0 := p_0^-$, and last key frame, $p_n := p_n^+$
  Subdivide in time to create a new frame between every two adjacent valid frames.
  Set the newly created frames as invalid
  repeat
    for each invalid frame j **do**
      Compute q_j by minimizing (4.8)
      Set frame j as valid
    Invalidate the previously valid frames, except for key frames
    for each valid frame i **do**
      compute p_i
  until (relaxation criteria satisfied)
until (enough middle frames is found)

Figure 5.1: Pseudocode of our physical interpolation scheme.

which tends to be visually distracting. If our system is spatially discretized as N particles, with each particle i connecting to j other particles, then the motion of i should be similar to the average motion of all those particles connected to it, or its acceleration, Δ^i, relative to them should be small:

$$\begin{aligned} \Delta^i &= \sum_{m=0}^{j} \left(\frac{d^2 q^i}{dt^2} - \frac{d^2 q^m}{dt^2} \right) \\ \Delta(q_k) &= \sum_{i=0}^{N-1} \Delta^i \\ E_{reg}(q_k) &= (\Delta(q_k))^2 \end{aligned}$$

Rayleigh Damping: Since we want as smooth as possible a motion between frames, it is fair to assume that neighboring frames should be quite similar. When neighboring frames exhibit dramatic differences, the resulting motion becomes incoherent. To ensure some coherence between q_k and its adjacent neighboring frames, q_{k-1} and q_{k+1}, we impose two damping forces, $F^d_{damp}(q_k, q_{k+1})$ and $F^d_{damp}(q_{k-1}, q_k)$, on q_k. The damping used here is based on Rayleigh damping, and computed using the method described in [13], where k_d is the discretized damping constant regulating the rate of decay. The regularization term becomes

$$E_{reg} = (F^d_{damp}(q_k, q_{k+1}))^2 + (F^d_{damp}(q_{k-1}, q_k))^2$$

with

$$F^d_{damp}(q_n, q_{n+1}) = -k_d \nabla W(q_n, q_{n+1}).$$

Temporal spline interpolation: Another way to ensure a smooth motion path between specified key

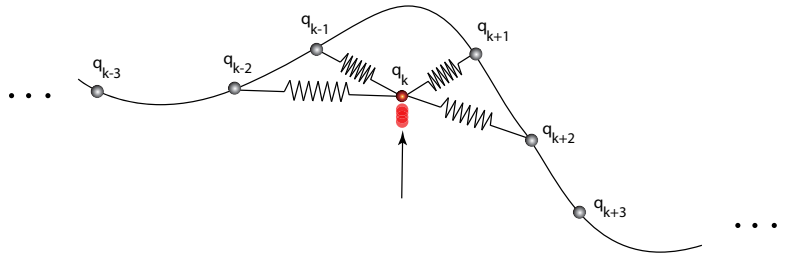


Figure 5.2: Regularization by attaching fictitious springs to neighboring frames.

frames is to do a spline interpolation in the system's phase-space. The given key frames would be used as knot vectors of the spline. This interpolation is physically modeled by attaching fictitious springs with a natural length of 0 between frame q_k and its neighboring frames $q_{k-2}, q_{k-1}, q_{k+1}, q_{k+2} \cdots$, figure (5.2), to help guide q_k to a state that satisfies the desired smoothness. The desired smoothness determines the number of frames and choice of spring constants. For example, for a C^1 continuous spline interpolation we use the frames $q_{k-2}, q_{k-1}, q_{k+1}, q_{k+2}$ as basis for computing q_k with its respective spring constant -1/16, 9/16, 9/16, -1/16. Note that even though the overall motion of the system in this case is not physical, since $c(h_k)E_{reg}$ vanishes with smaller h_k, the desired Lagangian becomes increasingly dominating, garnishing the finer motion with desired physical traits. For a system consisting of $2n$ fictitious springs, E_{reg} is calculated as follows, where q_i is the i^{th} neighboring frame of q_k, and k^i is the spring constant of the spring connecting q_k and q_i:

$$\begin{aligned} E_{reg}(q_k) &= \sum_{i=-n}^{n} (F^i_{sp}(q_k))^2 \\ F^i_{sp} &= -k^i(q_k - q_{k+i}) \end{aligned}$$

5.3 Relaxation

Adding auxiliary forces to make the solver converge obviously will change the state of the system, so solutions achieved at any finite discretization with such regularization does not necessarily correspond well to the motion we desire. Even for simple and well behaved systems which do not require regularization terms, like the 1-D spring-mass system described in the example section, it is not always possible to find solutions without iterating. To help the numerics converge to a desirable result, and to recover from potentially deleterious effects of regularization, we insert relaxation steps between iterations. Once new frames are computed in an iteration, we treat the previously computed frames as invalid, and recompute them based on the newly computed frames, as shown

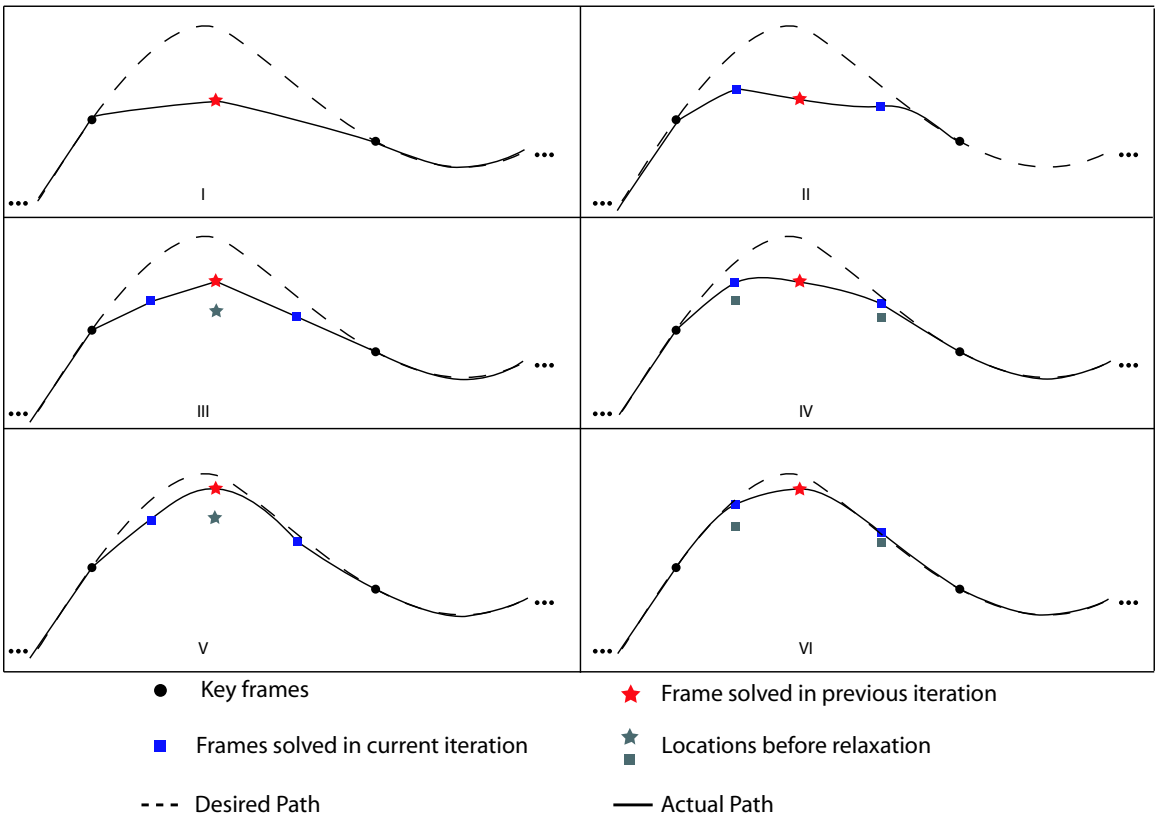


Figure 5.3: Relaxation process. The dotted path is the desired motion path we want our system to converge to. The solid path is the actual motion path. Through relaxation, the actual path is slowly converinge to the desired path. I. Given two key frames in black, and a previously solved frame in red; II. Blue frames are calculated based on previously known frames (red and black); III. First relaxation step, recalculate red frame based on blue frames; IV. Second relaxation step, recalculate blue frames based on the new red frame location; V. Third relaxation step, recalculate red frame based on new blue frame location; VI. Fourth relaxation step, recalculate blue frames based on new red frame location.

in figure (5.3). This process can be repeated many times. The optimal number of relaxation steps needed depends on the number of states in that system. The method used here is slow, as during one relaxation step, a change to a frame will only be propagated to its two adjacent frames. To help speed up the relaxation process, we employ an numerical trick along the principle of successive over-relaxation [22]. If q_o is the location of frame q before relaxation, and q_n is the new location after, then $dq = q_n - q_o$. Assuming dq is along the right direction for q to converge to, then we can speed up the convergence process by introducing an over-relax factor c_r, so instead of $q = q_o + dq$, we let $q = q_o + c_r\, dq$. It is best to keep $c_r \in [1, 2)$, as over shooting could happen which will result in q unnecessarily oscillating around the value it is trying to converge to. An example of the effects of relaxation on a motion path is given in the 1-D mass-spring example below.

Chapter 6

Examples

6.1 A simple 1-D mass spring system

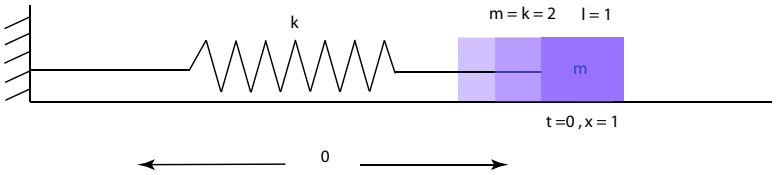


Figure 6.1: 1-D mass-spring system.

A mass M is attached to a spring, and it is constrained to move only in the X direction. Assume the spring is massless, there are no frictional forces acting on the system, and the springs obey Hook's law. Let q_k be the location of M, and $\dot{q}_k$ its velocity at time t_k. For simplicity, let $M = k = 2$, at time $t_1 = 0, q_1 = 1, \dot{q}_1 = 0$; $t_2 = 2\pi, q_2 = 1, \dot{q}_2 = 0$. The Lagrangian of the system is $L(t) = \dot{q}^2(t) - q^2(t)$; therefore a discrete Lagrangian can be defined as:

$$L_d(q_n, q_{n+1}, h_n) = h_n \left(\left(\frac{q_{n+1} - q_n}{h_n} \right)^2 - \left(\frac{1}{2} q_k + \frac{1}{2} q_{k+1} \right)^2 \right) \tag{6.1}$$

A simple physical example: The motion of this 1-D mass spring system is $q(t) = cos(t)$. As a test of the merit of our scheme, we will construct a smooth animation between t_1 to t_2 by minimizing the objective functions (4.8). Results are shown in figure 6.2. As we progress through iterations and fill in the middle states more and more, the motion path of the system converges to the physical path, $cos(t)$, as we expect.

A simple non-physical example: What if the key frames specified do not lie on the physical path of the system modeled by the Lagrangian above? For instance, let $q_2 = 8$ at $t_2 = 6\pi$, with $\dot{q}_2 = 0$. Figure 6.3 is the result from our method for this case. As we can see, we reached this result by simply

moving the second key frame to a new location, and left to our method to find the most physical looking path without having to explicitly model the additional boosting force in the Lagrangian.

About relaxation: It is important to note that the relaxation step plays an integral part in finding the right looking path, especially for an oscillating system where the given key frames span multiple periods. To illustrate this, consider our spring-mass system now spanning two periods of oscillation, with $q_2 = 1$ at $t_2 = 4\pi$. Figure 6.4 is the resulting motion paths from two relaxation schemes. Path a, (in red), with 5 relaxation steps within each iteration and path b, (in green) with a progressive relaxation process where we increase the number of relaxation as the iteration progresses. In this particular case, 90 overall relaxation steps were performed, where there was no relaxation in the first two iterations, and 40 in the final, fifth, iteration. In path a, even though the motion starts and finishes quite physical looking, however, the middle portion defies the laws of physics and staggers at some deformed configuration, which makes the resulting animation look unrealistic. Path b, on the other hand, behaves physically. This example demonstrates that by heavy use of relaxation, we are able to construct the physical results. It is impractical to perform 90 relaxation steps on complicated systems, but keep in mind that the example here is an extreme case. Specifying key frames in closer intervals, say, two per period, cuts down the number of relaxation steps.

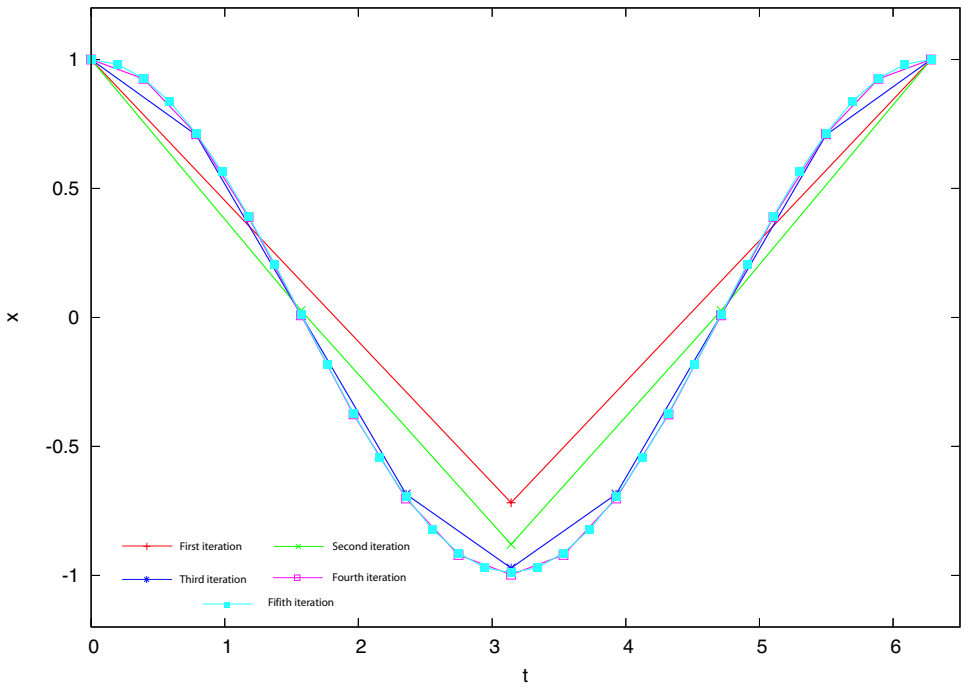


Figure 6.2: The iterative results of finding the motion path between $q_1 = 1, \dot{q}_1 = 0, t_1 = 0$ and $q_2 = 1, \dot{q}_2 = 0, t_2 = 2\pi$

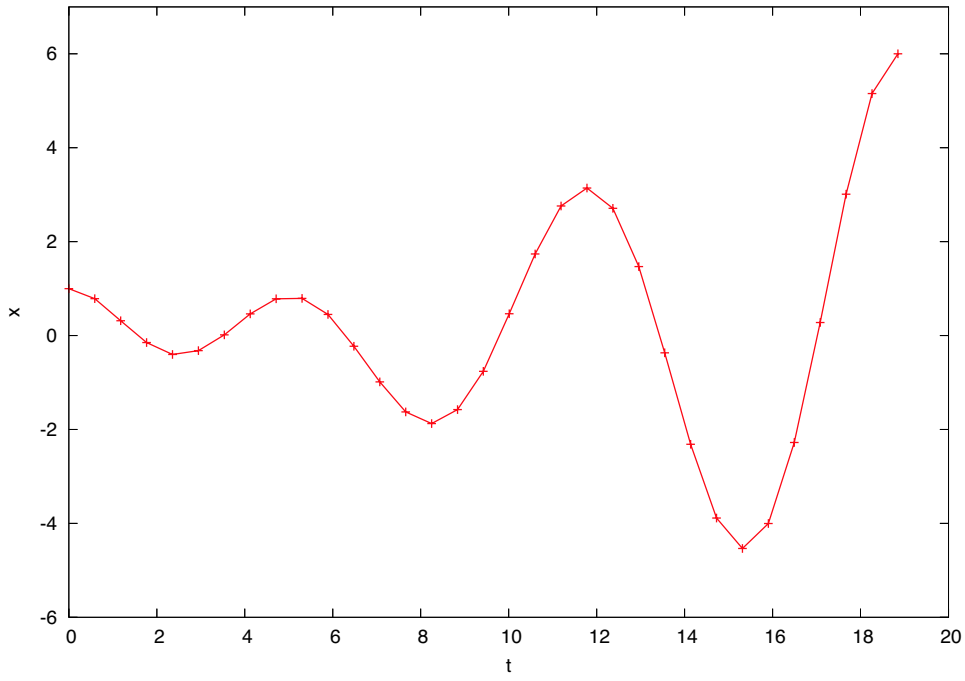


Figure 6.3: The resulting motion path from non-physical key frames. Here $q_2 = 8, \dot{q}_2 = 0, t_2 = 6\pi$.

6.2 Coupled mass-spring system in 3-space

Now we consider a system of coupled masses free to move in 3-space. The masses have identical weight and are attached to springs with identical rest-length and spring constant; see figure 6.5. There are no external forces acting on the system. For this example, we will consider the case of seven masses connected by six springs. As seen in figure 6.6, the system starts in configuration 1, and goes through configurations 2, 3, and 4 to spell out the letters "D", "M", "O", "C" .
The Lagrangian of the system is as follows:

$$L_d(q_n, q_{n+1}, h_n) = h_n \left(\frac{1}{2} \left(\frac{q_{n+1} - q_n}{h_n} \right)^T M \left(\frac{q_{n+1} - q_n}{h_n} \right) - \frac{1}{2} W(q_n) - \frac{1}{2} W(q_{n+1}) \right) \tag{6.2}$$

Here M is a 7x7 mass matrix, and the spring potential W is evaluated using the trapezoid rule as the average of W_n and W_{n+1}, where W is defined as the sum of the potentials of all 6 springs:

$$W = \sum_{i=1}^{6} W_{e^i} = \sum_{i=1}^{6} (|\bar{e}| - |e^i|)^2 \frac{1}{|\bar{e}|} \tag{6.3}$$

Here W_{e^i} is the potential of the i_{th} spring, $|\bar{e}|$ is the rest length of the springs, e^i the current configuration of the i^{th} spring, and $|e^i|$ the current length of the i^{th} spring. Consequently, the

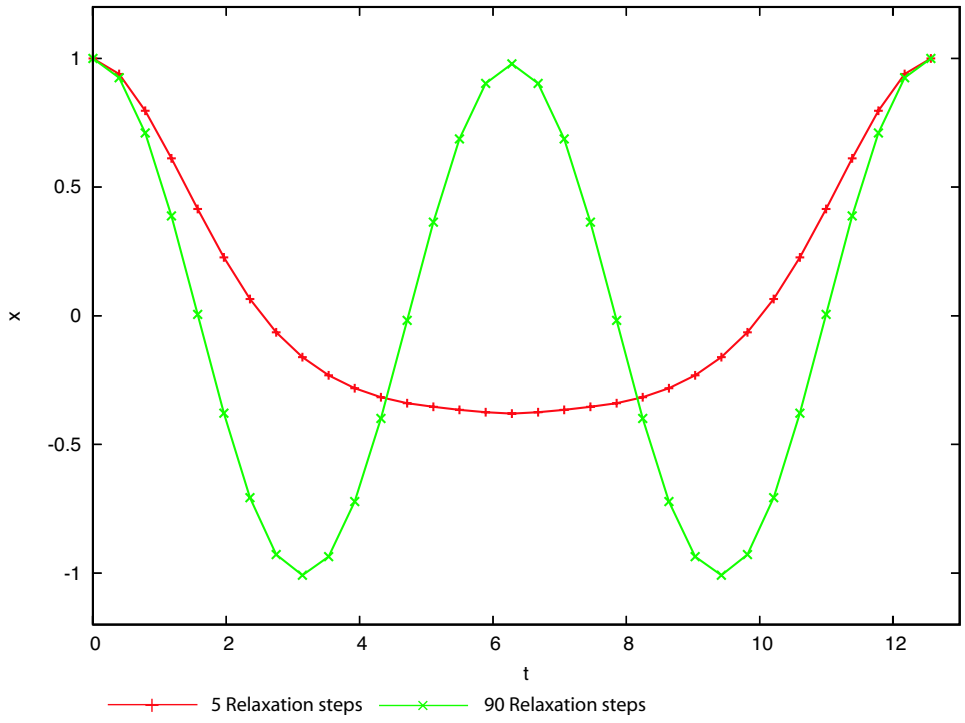


Figure 6.4: Comparison of motion paths from 5 relaxation steps per iteration and a progressive scheme with 90 total relaxation steps. Key frames, $q_1 = 1, \dot{q}_1 = 0, t_1 = 0$ and $q_2 = 1, \dot{q}_2 = 0, t_2 = 4\pi$, span two periods.

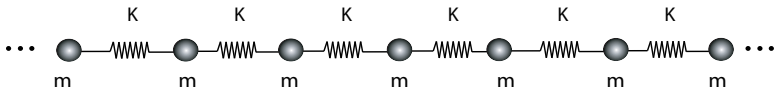


Figure 6.5: Coupled mass-spring system.

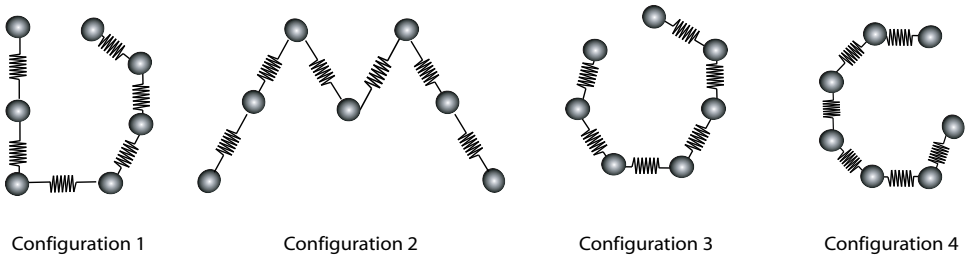


Figure 6.6: Key frames for coupled mass spring animation.

partial derivatives of the Lagrangian are

$$D_1 L_d(q_n, q_{n+1}, h_n) = -h_n \left(M \left(\frac{q_{n+1} - q_n}{h_n} \right) + \frac{1}{2} \nabla W(q_n) \right)$$

and

$$D_2 L_d(q_n, q_{n+1}, h_n) = h_n \left(M \left(\frac{q_{n+1} - q_n}{h_n} \right) - \frac{1}{2} \nabla W(q_{n+1}) \right),$$

where

$$\frac{\partial W_{e^i}}{\partial q_i} = -2 \left(\frac{e^i}{|\bar{e}|} - \frac{e^i}{|e^i|} \right).$$

Results: The key frames are spaced 1 unit apart in time. All the springs have rest length of 1 unit in space, and have been compressed at all four key frames, which is why the springs extend in the animation. Below are frames of the resulting simulation from six iterations of our mechanical interpolation scheme with 50 relaxation steps between each iteration.

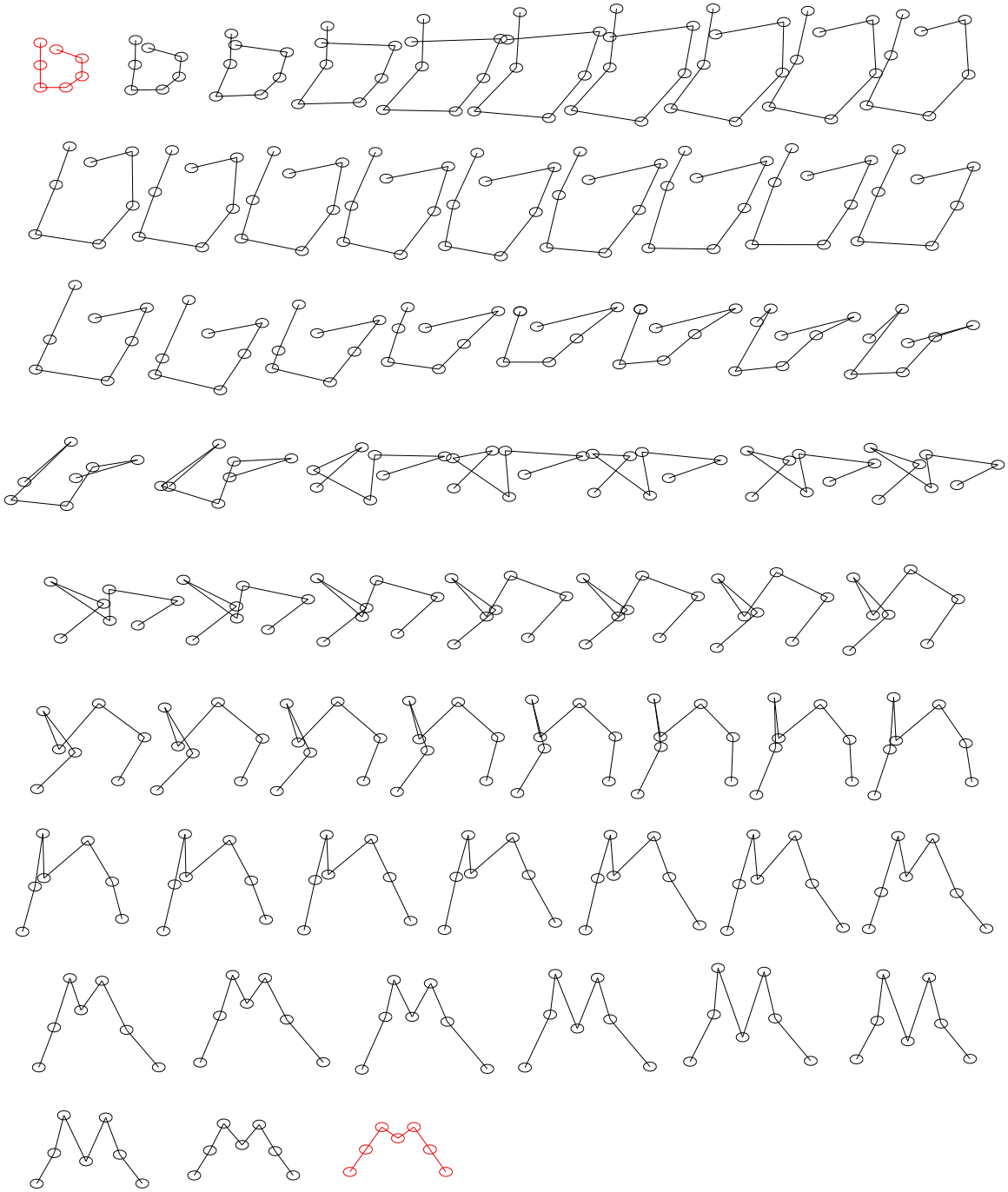


Figure 6.7: The frames changing letter “D” to letter “M”

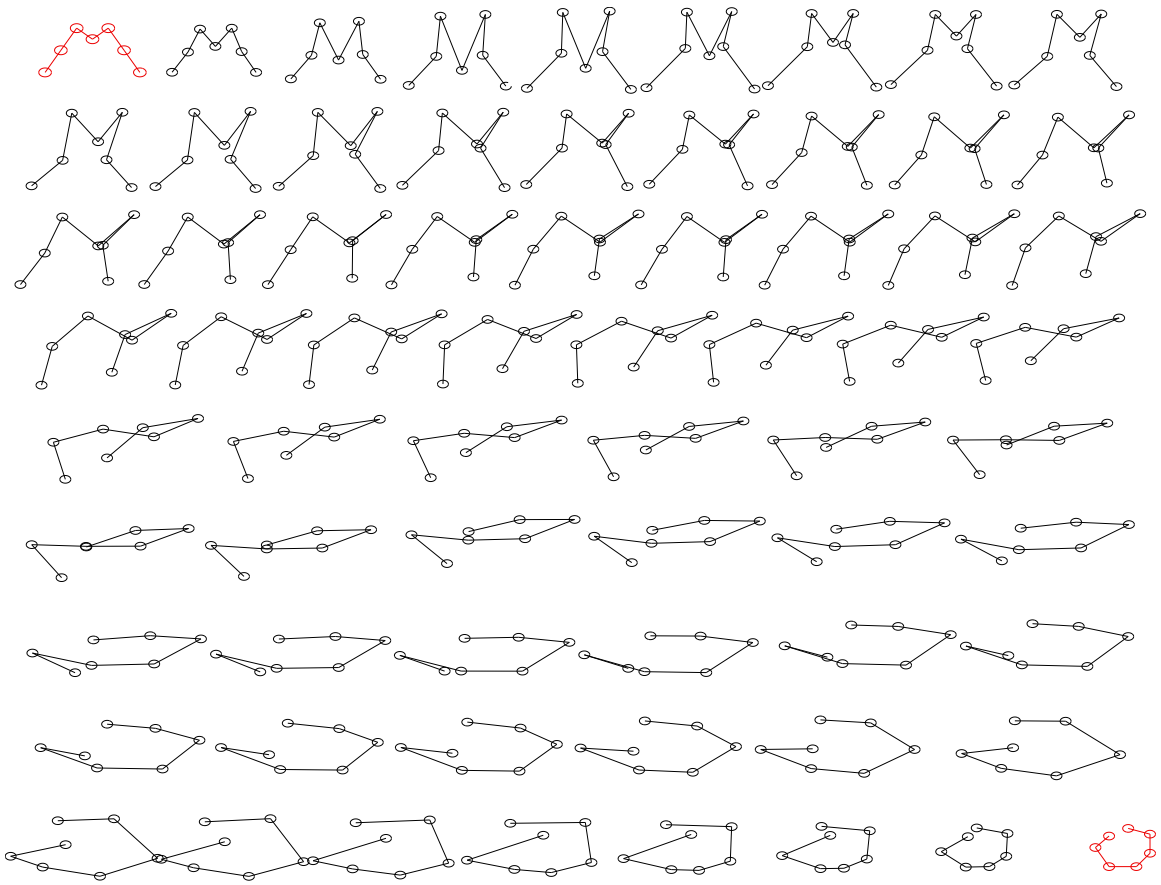


Figure 6.8: The frames changing letter “M” to letter “O”

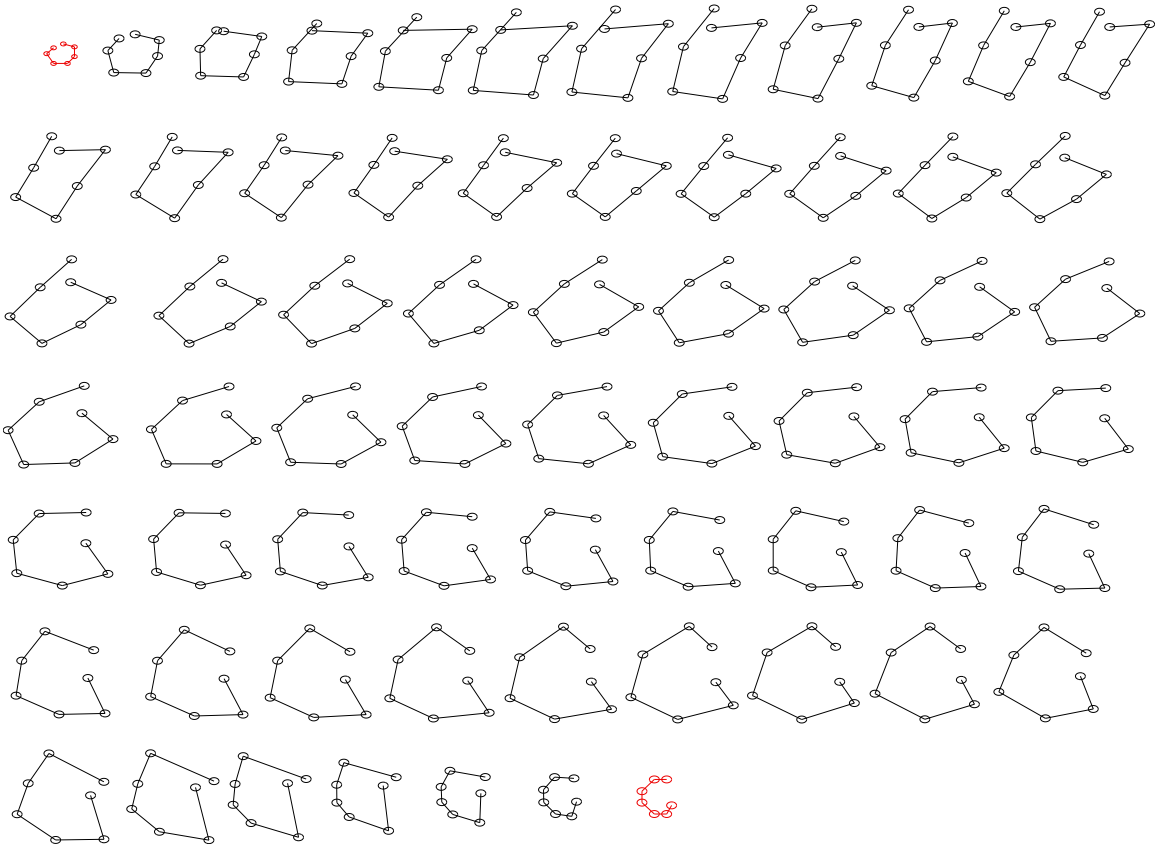


Figure 6.9: The frames changing letter “O” to letter “C”

Chapter 7

Discussion about TRACKS

Bergou *et al.* recently propose a process called TRACKS [4] with similar goals to ours; in their case, they add physical details, thin shell behavior, to an animation. TRACKS is a three-step process that first builds a correspondence between guide (coarse input) and tracked (fine output) shapes, then creates weak-form constraints using Petrov-Galerkin test functions, and finally solves constrained Lagrangian mechanics equations to add physical details while enforcing the matched constraints. It was shown to be robust and effective in adding physical details to an existing animation. By simulating or animating the coarse guide-shapes instead of the fine ones, TRACKS greatly reduces the computation required, but does not eliminate the hassles of having to manually animate an entire sequence and having to wrestle with the initial conditions of a simulation.

Our proposed discrete mechanical-interpolation scheme, on the other hand, is less trouble to set up, requiring only a few input keyframes. However, owing to the number of relaxation steps required, our method might not be practical for very detailed shapes, which have many degrees of freedom. If we combine TRACKS and our process, we might be able to reap the benefits of both. Our discrete mechanical interpolation scheme can be used to generate input animation for coarse-input guide-shapes; then one can use TRACKS to flesh out the fine physical detailed animation.

Chapter 8

Conclusion and Future Work

In this thesis we presented a novel interpolation scheme based on the Lagrange-d'Alembert principle capable of generating plausible looking motion with physically impossible constraints, along with its practical implementation and examples using our scheme. Our scheme is similar to DMOC, (discrete mechanics and optimal control)[11], method used in optimal control, but instead of solving for the entire motion in one shot, we take a local approach, and solve for a motion one discrete state at a time using an iterative process. This method can handle systems with much larger degrees of freedom. It is also excellent in integrating stylistic and artistic choice with the laws of physics to generate physical looking motion while satisfying conflicting constraints.

For future work, we plan to explore ways to cut down computational time, perhaps by taking a more DMOC approach by solving for and relaxing multiple states *simultaneously*, instead of one at a time. We would also like to adapt our methods to handling interpolation of colliding objects and systems with discontinuous Lagrangians.

Bibliography

[1] David Baraff and Andrew Witkin. Large steps in cloth simulation. In *ACM SIGGRAPH*, pages 43–54, 1998.

[2] Jernej Barbič and Doug James. Real-Time Subspace Integration for St. Venant-Kirchhoff Deformable Models. *ACM Trans. on Graphics*, 24(3):982–990, August 2005.

[3] Steven J. Benson, Lois Curfman McInnes, Jorge Moré, and Jason Sarich. TAO user manual (revision 1.7). Technical Report ANL/MCS-TM-242, Argonne National Lab, 2004.

[4] Miklós Bergou, Saurabh Mathur, Max Wardetzky, and Eitan Grinspun. TRACKS: Toward Directable Thin Shells. *SIGGRAPH (ACM Transactions on Graphics)*, Aug 2007.

[5] J. Bonet and A. Burton. A Simple Average Nodal Pressure Tetrahedral Element for Incompressible and Nearly Incompressible Dynamic Explicit Applications. *Comm. in Num. Meth. in Eng.*, 14(5):437–449, 1998.

[6] R. C. Fetecau, J. E. Marsden, M. Ortiz, and M. West. Nonsmooth Lagrangian Mechanics and Variational Collision Integrators. *SIAM J. Applied Dynamical Systems*, 2(3):381–416, 2003.

[7] Razvan C. Fetecau, Jerrold E. Marsden, Michael Ortiz, and Matt West. Nonsmooth Lagrangian Mechanics and Variational Collision Integrators. *SIAM J. Applied Dynamical Systems*, 2:381–416, 2003.

[8] Ernst Hairer, Christian Lubich, and Gerhard Wanner. *Geometric Numerical Integration: Structure-Preserving Algorithms for ODEs*. Springer, 2002.

[9] Michael Hauth, Olaf Etzmuß, and Wolfgang Straßer. Analysis of Numerical Methods for the Simulation of Deformable Models. *The Visual Computer*, 19(7-8):581–600, 2003.

[10] H.Yoshimura and J.E.Marsden. Dirac Structures and Lagrangian Mechanics. *J. Geom. and Physics*, 2006.

[11] O. Junge, J.E. Marsden, and S. Ober-Blöbaum. Discrete Mechanics and Optimal Control. In *Proc. of IFAC World Congress*, pages We–M14–TO/3, 2005.

[12] Couro Kane, Jerrold E. Marsden, Michael Ortiz, and Matt West. Variational Integrators and the Newmark Algorithm for Conservative and Dissipative Mechanical Systems. *I.J.N.M.E.*, 49:1295–1325, 2000.

[13] L. Kharevych, Weiwei, Y. Tong, E. Kanso, J. E. Marsden, P. Schrder, and M. Desbrun. Geometric, Variational Integrators for Computer Animation. *ACM/EG Symposium on Computer Animation*, pages 43–51, 2006.

[14] P. Krysl, S. Lall, and J.E. Marsden. Dimensional Model Reduction in Non-linear Finite Element Dynamics of Solids and Structures. *I.J.N.M.E.*, 51:479–504, 2000.

[15] S.K. Lahiri, J. Bonet, J. Peraire, and L. Casals. A Variationally Consistent Fractional Time Step Integration Method for Lagrangian Dynamics. *Int. J. Numer. Meth. Engng*, 3:1–23, 2003.

[16] S. Lall and M. West. Discrete variational Hamiltonian mechanics. *J. Phys. A: Math. Gen.*, 39:5509–5519, 2006.

[17] Adrian Lew. *Variational Time Integrators in Computational Solid Mechanics*. PhD thesis, Caltech, May 2003.

[18] Adrian Lew, Jerrold E. Marsden, Michael Ortiz, and Matt West. Asynchronous Variational Integrators. *Arch. Rational Mech. Anal.*, 167:85–146, 2003.

[19] J.E. Marsden and M. West. Discrete Mechanics and Variational Integrators. *Acta Numerica*, pages 357–515, 2001.

[20] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Series in Operations Research. Springer, 1999.

[21] Richard Parent. *Computer Animation: Algorithms and Techniques*. Morgan Kaufmann, 2001.

[22] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 2nd edition, 1992.

[23] Raul Radovitzky and Michael Ortiz. Error Estimation and Adaptive Meshing in Strongly Nonlinear Dynamic Problems. *Comput. Method. Appl. M*, 172(1-4):203–240, 1999.

[24] Ari Stein and Mathieu Desbrun. Discrete geometric mechanics for variational time integrators. In *Discrete Differential Geometry*. ACM SIGGRAPH Course Notes, 2006.

[25] A. Wächter and L. T. Biegler. On the Implementation of a Primal-Dual Interior Point Filter Line Search Algorithm for Large-Scale Nonlinear Programming. *Mathematical Programming 106(1)*, pages 25–57, 2006.

[26] Matthew West. *Variational Integrators*. PhD thesis, Caltech, June 2003.