Contents

1 Prologue

1.1 Programming and Understanding

One way to become aware of the precision required to unambiguously communicate a mathematical idea is to program it for a computer. Rather than using canned programs purely as an aid to visualization or numerical computation, we use computer programming in a functional style to encourage clear thinking. Programming forces us to be precise and unambiguous, without forcing us to be excessively rigorous. The computer does not tolerate vague descriptions or incomplete constructions. Thus the act of programming makes us keenly aware of our errors of reasoning or unsupported conclusions.[1]

Although this book is about differential geometry, we can show how thinking about programming can help in understanding in a more elementary context. The traditional use of Leibniz's notation and Newton's notation is convenient in simple situations, but in more complicated situations it can be a serious handicap to clear reasoning.

A mechanical system is described by a Lagrangian function of the system state (time, coordinates, and velocities). A motion of the system is described by a path that gives the coordinates for each moment of time. A path is allowed if and only if it satisfies the Lagrange equations. Traditionally, the Lagrange equations are written

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}} - \frac{\partial L}{\partial q} = 0.$$

What could this expression possibly mean?

Let's try to write a program that implements Lagrange equations. What are Lagrange equations for? Our program must take a proposed path and give a result that allows us to decide if the path is allowed. This is already

[1] The idea of using computer programming to develop skills of clear thinking was originally advocated by Seymour Papert. An extensive discussion of this idea, applied to the education of young children, can be found in Papert [13].

a problem; the equation shown above does not have a slot for a path to be tested.

So we have to figure out how to insert the path to be tested. The partial derivatives do not depend on the path; they are derivatives of the Lagrangian function and thus they are functions with the same arguments as the Lagrangian. But the time derivative d/dt makes sense only for a function of time. Thus we must be intending to substitute the path (a function of time) and its derivative (also a function of time) into the coordinate and velocity arguments of the partial derivative functions.

So probably we meant something like the following (assume that ω is a path through the coordinate configuration space, and so $w(t)$ specifies the configuration coordinates at time t):

$$\frac{d}{dt}\left(\left.\frac{\partial L(t,q,\dot{q})}{\partial \dot{q}}\right|_{\substack{q=w(t)\\ \dot{q}=\frac{dw(t)}{dt}}}\right) - \left.\frac{\partial L(t,q,\dot{q})}{\partial q}\right|_{\substack{q=w(t)\\ \dot{q}=\frac{dw(t)}{dt}}} = 0.$$

In this equation we see that the partial derivatives of the Lagrangian function are taken, then the path and its derivative are substituted for the position and velocity arguments of the Lagrangian, resulting in an expression in terms of the time.

This equation is complete. It has meaning independent of the context and there is nothing left to the imagination. The earlier equations require the reader to fill in lots of detail that is implicit in the context. They do not have a clear meaning independent of the context.

By thinking computationally we have reformulated the Lagrange equations into a form that is explicit enough to specify a computation. We could convert it into a program for any symbolic manipulation program because it tells us *how* to manipulate expressions to compute the residuals of Lagrange's equations for a purported solution path.[2]

1.2 Functional Abstraction

But this corrected use of Leibniz notation is ugly. We had to introduce extraneous symbols (q and $\dot{q}$) in order to indicate the argument position

[2]The *residuals* of equations are the expressions whose value must be zero if the equations are satisfied. For example, if we know that for an unknown x, $x^3 - x = 0$ then the residual is $x^3 - x$. We can try $x = -1$ and find a residual of 0, indicating that our purported solution satisfies the equation. A residual may provide information. For example, if we have the differential equation $df(x)/dx - af(x) = 0$ and we plug in a test solution $f(x) = Ae^{bx}$ we obtain the residual $(b-a)Ae^{bx}$, which can be zero only if $b = a$.

specifying the partial derivative. Nothing would change here if we replaced q and $\dot{q}$ by a and b.[3] We can simplify the notation by admitting that the partial derivatives of the Lagrangian are themselves new functions, and by specifying the particular partial derivative by the position of the argument that is varied

$$\frac{d}{dt}\left((\partial_2 L)\left(t, w(t), \frac{d}{dt}w(t)\right)\right) - (\partial_1 L)\left(t, w(t), \frac{d}{dt}w(t)\right) = 0,$$

where $\partial_i L$ is the function which is the partial derivative of the function L with respect to the ith argument.[4]

Two different notions of derivative appear in this expression. The functions $\partial_2 L$ $\partial_1 L$, constructed from the Lagrangian L, have the same arguments as L.

The derivative d/dt is an expression derivative. It applies to an expression that involves the variable t and it gives the rate of change of the value of the expression as the value of the variable t is varied.

These are both useful interpretations of the idea of a derivative. But functions give us more power. There are many equivalent ways to write expressions that compute the same value. For example $1/(1/r_1 + 1/r_2) = (r_1 r_2)/(r_1 + r_2)$. These expressions compute the same function of the two variables r_1 and r_2. The first expression fails if $r_1 = 0$ but the second one gives the right value of the function. If we abstract the function, say as $\Pi(r_1, r_2)$, we can ignore the details of how it is computed. The ideas become clearer because they do not depend on the detailed shape of the expressions.

So let's get rid of the expression derivative d/dt and replace it with an appropriate functional derivative. If f is a function then we will write Df as the new function that is the derivative of f:[5]

$$(Df)(t) = \frac{d}{dx}f(x)\bigg|_{x=t}.$$

To do this for the Lagrange equation we need to construct a function to take the derivative of.

[3] That the symbols q and $\dot{q}$ can be replaced by other arbitrarily chosen nonconflicting symbols without changing the meaning of the expression tells us that the partial derivative symbol is a logical quantifier, like forall and exists ($\forall$ and $\exists$).

[4] The argument positions of the Lagrangian are indicated by indices starting with zero for the time argument.

[5] An explanation of functional derivatives is in Appendix B, page 202.

Given a configuration-space path w, there is a standard way to make the state-space path. We can abstract this method as a mathematical function Γ:

$$\Gamma[w](t) = \left(t, w(t), \frac{d}{dt}w(t)\right).$$

Using Γ we can write:

$$\frac{d}{dt}\left((\partial_2 L)\left(\Gamma[w](t)\right)\right) - \left(\partial_1 L\right)\left(\Gamma[w](t)\right) = 0.$$

If we now define composition of functions $(f \circ g)(x) = f(g(x))$, we can express the Lagrange equations entirely in terms of functions:

$$D\left((\partial_2 L) \circ (\Gamma[w])\right) - (\partial_1 L) \circ (\Gamma[w]) = 0.$$

The functions $\partial_1 L$ and $\partial_2 L$ are partial derivatives of the function L. Composition with $\Gamma[w]$ evaluates these partials with coordinates and velocites appropriate for the path w, making functions of time. Applying D takes the time derivative. The Lagrange equation states that the difference of the resulting functions of time must be zero. This statement of the Lagrange equation is complete, unambiguous, and functional. It is not encumbered with the particular choices made in expressing the Lagrangian. For example, it doesn't matter if the time is named t or τ, and it has an explicit place for the path to be tested.

This expression is equivalent to a computer program:[6]

```
(define ((Lagrange-equations Lagrangian) w)
  (- (D (compose ((partial 2) Lagrangian) (Gamma w)))
     (compose ((partial 1) Lagrangian) (Gamma w))))
```

In the Lagrange equations procedure the parameter `Lagrangian` is a procedure that implements the Lagrangian. The derivatives of the Lagrangian, for example `((partial 2) Lagrangian)`, are also procedures. The state-space path procedure `(Gamma w)` is constructed from the configuration-space path procedure `w` by the procedure `Gamma`:

[6]The programs in this book are written in Scheme, a dialect of Lisp. The details of the language are not germane to the points being made. What is important is that it is mechanically interpretable, and thus unambiguous. In this book we require that the mathematical expressions be explicit enough that they can be expressed as computer programs. Scheme is chosen because it is easy to write programs that manipulate representations of mathematical functions. An informal description of Scheme can be found in Appendix A. The use of Scheme to represent mathematical objects can be found in Appendix B. A formal description of Scheme can be obtained in [10]. You can get the software from [21].

```
(define ((Gamma w) t)
  (up t (w t) ((D w) t)))
```

where `up` is a constructor for a data structure that represents a state of the dynamical system (time, coordinates, velocities).

The result of applying the `Lagrange-equations` procedure to a procedure `Lagrangian` that implements a Lagrangian function is a procedure that takes a configuration-space path procedure `w` and returns a procedure that gives the residual of the Lagrange equations for that path at a time.

For example, consider the harmonic oscillator, with Lagrangian

$$L(t, q, v) = \frac{1}{2}mv^2 - \frac{1}{2}kq^2,$$

for mass m and spring constant k. this lagrangian is implemented by

```
(define ((L-harmonic m k) local)
  (let ((q (coordinate local))
        (v (velocity local)))
    (- (* 1/2 m (square v))
       (* 1/2 k (square q)))))
```

We know that the motion of a harmonic oscillator is a sinusoid with a given amplitude a, frequency ω, and phase φ:

$$x(t) = a\cos(\omega t + \varphi).$$

Suppose we have forgotten how the constants in the solution relate to the physical parameters of the oscillator. Let's plug in the proposed solution and look at the residual:

```
(define (proposed-solution t)
  (* 'a (cos (+ (* 'omega t) 'phi))))

(show-expression
 (((Lagrange-equations (L-harmonic 'm 'k))
   proposed-solution)
  't))

;; should produce \cos(\omega t + \varphi) a (k-m\omega^2)
```

The residual here shows that for nonzero amplitude, the only solutions allowed are ones where $(k - m\omega^2) = 0$ or $\omega = \sqrt{k/m}$.

But, suppose we had no idea what the solution looks like. We could propose a literal function for the path:

```
(show-expression
 (((Lagrange-equations (L-harmonic 'm 'k))
   (literal-function 'x))
  't))
;; should produce $$kx(t)+mD^2 x(t)$$
```

If this residual is zero we have the Lagrange equation for the harmonic oscillator.

Note that we can flexibly manipulate representations of mathematical functions. (See Appendices A and B.)

We started out thinking that the original statement of Lagrange's equations accurately captured the idea. But we really don't know until we try to teach it to a naive student. If the student is sufficiently ignorant, but is willing to ask questions, we are led to clarify the equations in the way that we did. There is no dumber but more insistent student than a computer. A computer will absolutely refuse to accept a partial statement, with missing parameters or a type error. In fact, the original statement of Lagrange's equations contained an obvious type error: the Lagrangian is a function of multiple variables, but the d/dt is applicable only to functions of one variable.