

Contents

1	Problem Formulation	1
1.1	The Brachistochrone Paradigm	1
1.1.1	Development of a Problem Formulation	4
1.1.2	Introduction to Structured Optimization	13
1.1.3	Avoiding Common Errors # 1	15
1.1.4	Scaling: How to Design Your Own Units!	20
1.1.5	Alternative Problem Formulations	23
1.2	Changing the Paradigm	31
1.3	The Target Set	37
1.4	Guideposts for Problem Formulations	42
1.4.1	The Standard Problem B	42
1.4.2	A Practical Problem Formulation	46
1.4.3	Avoiding Common Errors # 2	48
1.4.4	Transformational Tips and Tricks	55
1.5	*Real-Time Optimal Control (RTOC)	63
1.5.1	Feedback Control via RTOC	64
1.5.2	**Towards Artificial Intelligence via RTOC	69
1.6	**Tychastic Optimal Control	75
1.6.1	Specification of Tychastic Target Sets	77
1.6.2	Chance-Constrained Optimal Control	79
1.6.3	Connections to Quantum Control and Other Applications	80
1.7	Endnotes	82
2	Pontryagin's Principle	85
2.1	The Standard Problem	85
2.2	Introduction to Covectors	89

2.3	The Covectors in Optimal Control	93
2.4	Introducing Pontryagin's Principle	99
2.4.1	The Basics	100
2.4.2	In Search of the Missing Boundary Conditions	108
2.4.3	Formulating the Boundary Value Problem B^λ	111
2.4.4	Solving the BVP: Work, Plug, Pray	113
2.5	Minimizing the Hamiltonian	114
2.5.1	How Things Go Wrong	115
2.5.2	KKT Conditions for Problem HMC	119
2.5.3	Time-Varying Control Space	121
2.5.4	Solving Problem HMC	121
2.5.5	A Tale of Two, Maybe Three Hamiltonians	125
2.6	A Cheat Sheet for Pontryagin's Principle	127
2.7	*Pontryagin's Principle for Problem P	131
2.8	Avoiding Common Errors # 3	144
2.9	Kalman and the Curse of Sensitivity	150
2.9.1	An Introduction to the Curse of Sensitivity	150
2.9.2	**Escaping the Curse of Sensitivity	157
2.10	Pontryagin and the Calculus of Variations	165
2.11	Endnotes	168
3	Example Problems	171
3.1	Solving the Brachistochrone Problem	171
3.2	The Brachistochrone Problem via DIDO	177
3.2.1	Basic Verification and Validation (V&V)	180
3.2.2	Problem-Specific V&V	184
3.2.3	*Some Finer Points	185
3.2.4	**Pseudospectral Theory vs Exact Solutions	187
3.3	A Double-Integrator Quadratic Problem	191
3.3.1	Closed-Loop Versus Closed-Form	200
3.3.2	An Optimal-Control Perspective of PD Controllers	201
3.3.3	Applications to Optimal Missile Guidance	203
3.3.4	Everything is Optimal	206
3.4	Kalman's Linear-Quadratic Control Problem	208
3.5	The Time-Optimal Double-Integrator	218

CONTENTS

3.6	The Powered Explicit Guidance Problem	231
3.7	**A Myth-Busting Example Problem	240
4	Exercise Problems	247
4.1	One-Dimensional Problems	247
4.1.1	Linear-Quadratic Exercise Problems	248
4.1.2	A Simple Exercise in BVPs	249
4.1.3	An Example Problem of Problems	252
4.1.4	Illustrating Think Twice, Code Once	253
4.1.5	A Medley of Thought-Provoking Problems	254
4.2	A Suite of Test Problems and Solutions	258
4.3	RTOC for Nonbelievers	260
4.3.1	RTOC for Problem DQC	260
4.3.2	Strap-On Optimal Guidance	263
4.4	Double-Integrator-Type Problems	266
4.4.1	A Modified Quadratic Control Problem	266
4.4.2	An L^1 -Optimal Control Problem	268
4.4.3	*Breakwell's Problem	270
4.4.4	*Gong's Motion Planning Test Problem	272
4.4.5	*A Minimum Energy Problem	276
4.4.6	*Fuller's Problem	277
4.5	Optimal Control For Nonbelievers	277
4.5.1	A Mollified Minimum-Time Problem	278
4.5.2	*A Jerk-Limited Minimum-Time Problem	278
4.5.3	*A Rate-Limited Minimum-Time Problem	280
4.5.4	*A Constrained Minimum-Time Problem	280
4.6	Queen Dido and the Badlands	282
4.7	Zermelo Problems	285
4.7.1	A Classic Nonlinear Zermelo Problem	286
4.7.2	**A Tychastic Zermelo Problem	287
4.8	Astrodynamic Optimization	290
4.8.1	A Moon-Landing Problem	290
4.8.2	Velocity Steering	292
4.8.3	Max-Energy Orbit Transfer: Cartesian Formulation	293
4.8.4	Max-Energy Orbit Transfer: Polar Formulation	301

4.8.5	Minimum-Propellant Orbit Transfer	303
4.9	A Rigid-Body Steering Problem	306
4.10	A Coupled Motion-Planning Problem	311
4.11	Management Science Problems	316
4.11.1	A Wheat-Trading Problem	316
4.11.2	Checking-Schedule for Detecting Failures	319
4.12	Endnotes	322
	References	323
	Index	333

Chapter 2

Pontryagin's Principle

*You want proof?
You can't handle the proof!*

In this chapter, we investigate and illustrate the optimality conditions for the standard optimal control problem developed in Chapter 1. The main result is called Pontryagin's Principle. It is also referred to as the Minimum Principle, the Maximum Principle,* Pontryagin's Minimum Principle or Pontryagin's Maximum Principle.

2.1 The Standard Problem

See Fig. 2.1. As discussed in §1.4, page 42, a standard optimal control problem can be defined in terms of finding a dynamically feasible state-control function pair, $\{\mathbf{x}(\cdot), \mathbf{u}(\cdot)\}$, that transfers the state of system, $\mathbf{x} \in \mathbb{R}^{N_x}$, from a given initial condition, $\mathbf{x}(t_0) = \mathbf{x}^0$, to a target condition, $\mathbf{e}(\mathbf{x}_f, t_f) = \mathbf{0}$, while minimizing a given cost functional, J . Following Chapter 1, this standard problem can be organized in a structured format[†] as

*See §2.5.5 for further discussions on minimum versus maximum.

[†]This format is nearly identical to the actual structure of a DIDO code.

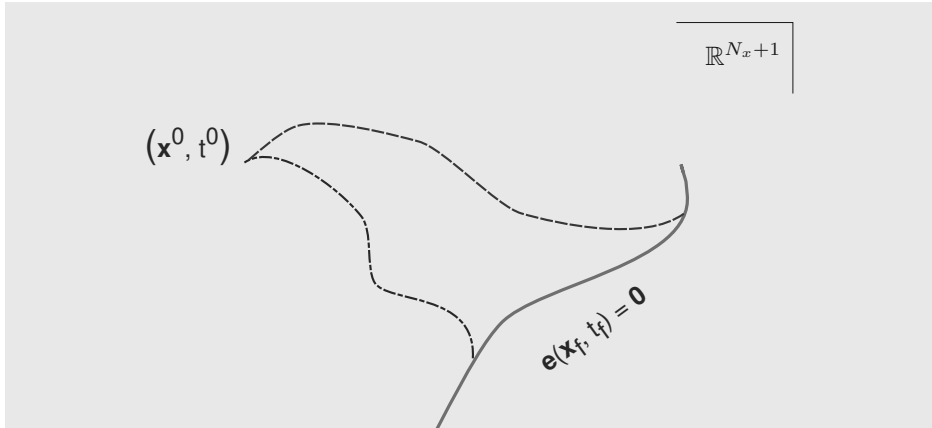


Figure 2.1: Schematic for the standard problem; same as Fig. 1.27, repeated here for quick reference.

$$\begin{array}{l}
 \mathbf{x} \in \mathbb{X} = \mathbb{R}^{N_x} \quad \mathbf{u} \in \mathbb{U} \subseteq \mathbb{R}^{N_u} \quad \left. \vphantom{\mathbf{x} \in \mathbb{X}} \right\} \text{(preamble)} \\
 \text{problem } \left\{ \begin{array}{l}
 \text{Minimize} \quad J[\mathbf{x}(\cdot), \mathbf{u}(\cdot), t_f] = E(\mathbf{x}(t_f), t_f) \\
 \qquad \qquad \qquad + \int_{t_0}^{t_f} F(\mathbf{x}(t), \mathbf{u}(t), t) dt \quad \left. \vphantom{J[\mathbf{x}(\cdot), \mathbf{u}(\cdot), t_f]} \right\} \text{(cost)} \\
 \text{Subject to} \quad \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t) \quad \left. \vphantom{\dot{\mathbf{x}}(t)} \right\} \text{(dynamics)} \\
 \qquad \qquad \qquad \mathbf{x}(t_0) = \mathbf{x}^0 \\
 \qquad \qquad \qquad t_0 = t^0 \\
 \qquad \qquad \qquad e(\mathbf{x}_f, t_f) = \mathbf{0} \quad \left. \vphantom{e(\mathbf{x}_f, t_f)} \right\} \text{(endpoints)}
 \end{array} \right.
 \end{array}$$

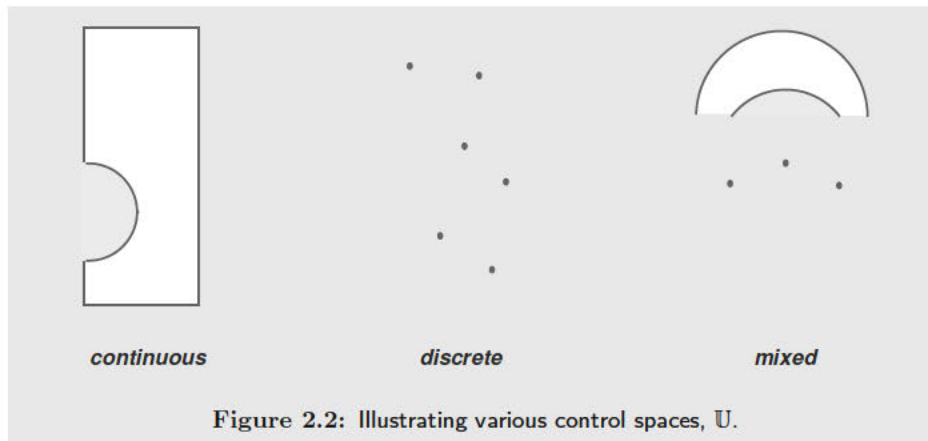
In addition to \mathbb{U} , \mathbf{x}^0 and t^0 , the four functions,

- $E : (\mathbf{x}_f, t_f) \mapsto \mathbb{R}$ (endpoint cost)
- $e : (\mathbf{x}_f, t_f) \mapsto \mathbb{R}^{N_e}$ (endpoint constraint)
- $F : (\mathbf{x}, \mathbf{u}, t) \mapsto \mathbb{R}$ (running cost)
- $\mathbf{f} : (\mathbf{x}, \mathbf{u}, t) \mapsto \mathbb{R}^{N_x}$ (dynamics)

are called the **problem data**. The data functions are assumed to be continuously differentiable with respect to the state variable.

Notation Alert: By $x \in \mathbb{R}^{N_x}$, we mean the N_x -dimensional state variable. This is distinguished from $x(\cdot)$, which is a function (for example, $t \mapsto x$), the graph of which, with respect to time, is a curve. The function $x(\cdot)$ is called the *state trajectory* or *arc* (the latter term is somewhat antiquated). Similarly, $u \in \mathbb{R}^{N_u}$ is the N_u -dimensional control variable, while $u(\cdot)$ is the control function (for example, $t \mapsto u$) called the *control trajectory*; see Fig. 1.26. The pair $\{x(\cdot), u(\cdot)\}$ is called the *system trajectory*. In addition, x_0 is shorthand for $x(t_0)$, while x^0 means generic numerical data. Similar notation is used for the the final-time conditions, as well. Thus, for example, the set defined by the equation $e(x_f, t_f) = \mathbf{0}$ means that x_f and t_f are variables in $\mathbb{R}^{N_x} \times \mathbb{R}$, whereas x^f means a specific value of $x(t_f) \equiv x_f$; see Fig. 2.1. Finally, we use upper cases for cost (or cost-like) functions and lower cases for the corresponding constraints. Thus the pairs (E, e) and (F, f) go together. *See page [xix](#) for additional nomenclature.*

In this standard problem formulation, the control space, \mathbb{U} , is quite an arbitrary set: it may be *continuous* (i.e. a subset of \mathbb{R}^{N_u} with the power of the continuum), *discrete* (i.e. have finite cardinality), disjoint, non-convex ... and a host of other possibilities that are crucial for practical applications; see Fig. 2.2. The folklore that practical sets are “nice” is simply not true. A simple example of such a practical set is shown in Fig. 2.3.



Problem B was conceived by Pontryagin in 1955[33]. Today, on hindsight, it is a clear and natural way to formulate many problems in engineering, economics, physics, management, environment, social sciences, epidemiology and

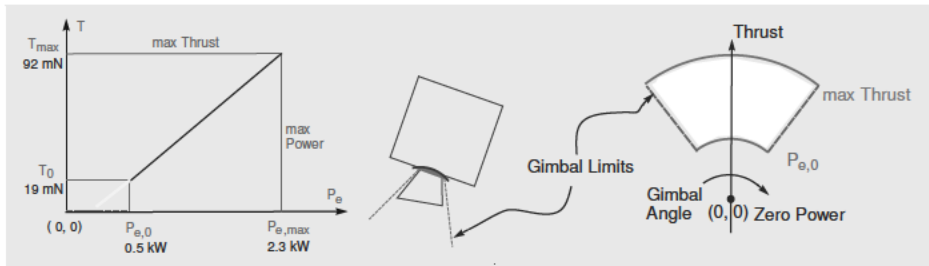


Figure 2.3: The control space for a practical electrically powered space thruster is nonconvex and disjoint: Zero power yields zero thrust; however, a minimum amount of nonzero power, $P_{e,0}$, is necessary to generate a nonzero minimum thrust, T_0 .

other disciplines. In the 1950s, there was an absence of this clarity because the concepts in control and optimization were still evolving. The symbol u was adopted for control because, in Russian, *upravlenie* means control. The *calculus of optimal control* developed by Pontryagin and his students requires the concept of covectors.

Illustrating the Concept: Problem Formulation

As developed in Chapter 1, the Brachistochrone problem fits quite readily under the constructs of Problem B. Recall that the Brac:1 formulation (see page 12) is given by

$$\begin{array}{l}
 \left. \begin{array}{l}
 \mathbf{X} = \mathbb{R}^3 \\
 \mathbf{x} = (x, y, v)
 \end{array} \right\} \text{(state)} \\
 \left. \begin{array}{l}
 \mathbf{U} = \mathbb{R} \\
 u = \theta
 \end{array} \right\} \text{(control)} \\
 \left. \begin{array}{l}
 \text{Minimize } J[\mathbf{x}(\cdot), \mathbf{u}(\cdot), t_f] = t_f \\
 \text{Subject to } \begin{array}{l}
 \dot{x} = v \sin \theta \\
 \dot{y} = v \cos \theta \\
 \dot{v} = g \cos \theta
 \end{array} \\
 \begin{array}{l}
 (t_0, x_0, y_0, v_0) = (0, 0, 0, 0) \\
 (x_f - x^f, y_f - y^f) = (0, 0)
 \end{array}
 \end{array} \right\} \text{(problem)} \\
 \text{(Brac : 1)}
 \end{array}
 \left. \begin{array}{l}
 \text{(cost)} \\
 \text{(dynamics)} \\
 \text{(endpoints)}
 \end{array} \right\}$$

In mapping the various symbols and notation of Problem B to Brac:1, we have $N_x = 3$, $N_u = 1$, $E(x_f, t_f) = t_f$, $F = \Theta$, $f_1(x, u) = v \sin \theta$, $f_2(x, u) = v \cos \theta$, $f_3(x, u) = g \cos \theta$, $e_1(x_f, t_f) = x_f - x^f$, and $e_2(x_f, t_f) = y_f - y^f$.

2.2 Introduction to Covectors

By definition, the state vector \mathbf{x} is just a medley of N_x variables stacked on top of one another:

$$\mathbf{x} := \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{N_x} \end{bmatrix} \begin{array}{l} x_1\text{-units} \\ x_2\text{-units} \\ \vdots \\ x_{N_x}\text{-units} \end{array} \quad (2.1)$$

We have purposefully written Eq. (2.1) with its units alongside to emphasize the point that \mathbf{x} typically comprises a collection of “heterogeneous” scalars. Thus, our intuitive notion of vector as a quantity with magnitude and direction must be seriously revised.

Illustrating the Concept in \mathbb{R}^3

In (Brac:1), the state vector is defined by

$$\mathbf{x} := \begin{bmatrix} x \\ y \\ v \end{bmatrix} \begin{array}{l} \text{position-units} \\ \text{position-units} \\ \text{velocity-units} \end{array} \quad (2.2)$$

If the “magnitude” of \mathbf{x} is computed by the standard Euclidean norm

$$\|\mathbf{x}\|_2 = \sqrt{x^2 + y^2 + v^2}$$

the result would be physically meaningless. In addition, we cannot legitimately add the square of position units to the square of velocity units.

If a vector is defined naïvely as a quantity with magnitude and direction, what, then, is the magnitude of this vector? What is its direction? What happens to the magnitude and direction if we change units?

These simple observations indicate that we need a potentially new approach to define a meaningful and natural way to measure vectors.

To motivate a new set of ideas, consider the following everyday example: Suppose we describe a sandwich that comprises the variables $x_1, x_2, x_3 \dots$ where x_1 is bread, x_2 is mayonnaise, x_3 is lettuce, x_4 is meat, and so on. Then it is clear that we can describe the state of the sandwich in terms of a vector \mathbf{x} , of N_x ingredients, in much the same way as Eq. (2.1). Each component of

this vector can be measured in terms of appropriate units. For instance, x_1 can be measured in terms of the number of slices of bread, x_2 in terms of ounces of mayonnaise, x_3 in terms of the number of lettuce leaves and so on. See Fig. 2.4. With these preliminaries, consider a two-dimensional model of a sandwich composed of just bread and meat

$$x := \begin{bmatrix} b \\ m \end{bmatrix} \quad \begin{array}{l} \text{slices} \\ \text{ounces} \end{array}$$

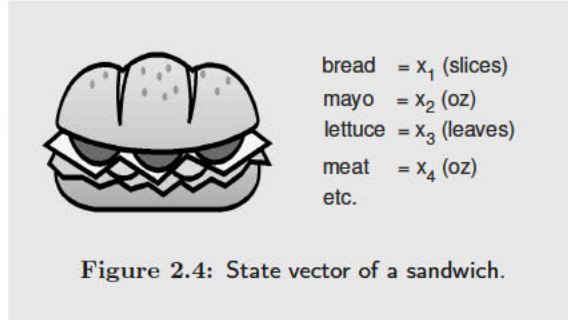


Figure 2.4: State vector of a sandwich.

where b is bread measured in number of slices and m is meat measured in ounces. Suppose the numerical value of the state vector is given by

$$x = \begin{bmatrix} 2 \\ 5 \end{bmatrix} \quad \begin{array}{l} \text{slices} \\ \text{ounces} \end{array}$$

Obviously, we would not compute the size of the sandwich using Euclidean ideas

$$\|x\| = \sqrt{4 \text{ slices}^2 + 25 \text{ ounces}^2}$$

One common approach to measure the “size of the sandwich” is in terms of its calorie count. Suppose that each slice of bread is 80 *calories* and each ounce of (lean) meat is 55 *calories*; then, the calorie count of the sandwich is given by

$$\underbrace{80 \times 2}_{\text{bread calories}} + \underbrace{55 \times 5}_{\text{meat calories}} = \underbrace{435}_{\text{total calories}}$$

We formalize this mathematics as follows: We agree to measure the “size” of the vector x (representing the sandwich) in terms of a *common unit* (CU) (which is calories for the sandwich). Then we let $\lambda_b = 80$ calories per slice and $\lambda_m = 55$ calories per slice and perform a *linear measure* of x using the (linear)

operation

$$\begin{aligned}\omega(\mathbf{x}) &:= \lambda_b \times b + \lambda_m \times m & (2.3) \\ &= 80 \times 2 + 55 \times 5 \\ &= 435 \text{ calories}\end{aligned}$$

The linear measure of a vector is not unique in the following sense: Instead of the calorie count, suppose we decided to measure the size of the sandwich in terms of its weight, a process adopted in many cafeterias. Then suppose we have $\tilde{\lambda}_b = 1$ ounce per slice and $\tilde{\lambda}_m = 1$ ounce per ounce; this generates a new linear measure of \mathbf{x} that uses the same linear operation as before, but with new coefficients:

$$\begin{aligned}\omega(\mathbf{x}) &= \tilde{\lambda}_b \times b + \tilde{\lambda}_m \times m \\ &= 1 \times 2 + 1 \times 5 \\ &= 7 \text{ ounces}\end{aligned}$$

Yet another way to measure the size of the sandwich is its dollar cost. Suppose $\hat{\lambda}_b = 20$ cents per slice and $\hat{\lambda}_m = 30$ cents per ounce; then, the new linear measure of \mathbf{x} is given by

$$\begin{aligned}\omega(\mathbf{x}) &= \hat{\lambda}_b \times b + \hat{\lambda}_m \times m \\ &= 20 \times 2 + 30 \times 5 \\ &= 190 \text{ cents } (= \$1.90)\end{aligned}$$

Thus, there are many ways to construct a linear measure of \mathbf{x} .

Study Problem 2.1

1. Construct at least two additional linear measures for a sandwich. What are the numerical values of λ_b and λ_m for these new linear measures?
2. Pick any two items from your grocery bag that have an FDA label pasted on them. Identify at least two linear measures adopted by the FDA to inform a consumer.

We formalize our sandwich example and declare that we will measure vectors in \mathbb{R}^{N_x} by a linear scalar function, ω , defined by

$$\omega(\mathbf{x}) := \lambda_1 x_1 + \lambda_2 x_2 + \cdots + \lambda_{N_x} x_{N_x} \tag{2.4}$$

The collection of N_x quantities, $\lambda_1, \lambda_2, \dots, \lambda_{N_x}$, that we choose to measure a vector can be stacked up in a manner similar to \mathbf{x}

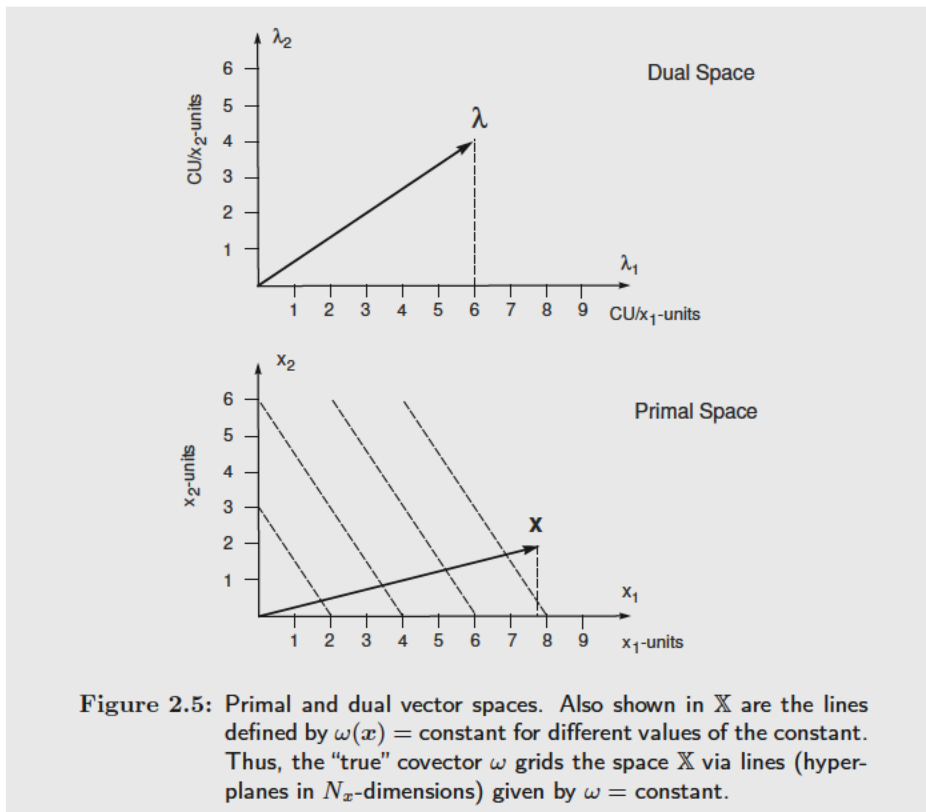
$$\boldsymbol{\lambda} := \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_{N_x} \end{bmatrix} \begin{array}{l} CU/x_1\text{-units} \\ CU/x_2\text{-units} \\ \vdots \\ CU/x_{N_x}\text{-units} \end{array} \tag{2.5}$$

to produce a new vector, $\boldsymbol{\lambda}$. This vector is called a *covector*. Obviously, $\boldsymbol{\lambda} \in \mathbb{R}^{N_x}$; however, as evident from its units, *its home-space is not* \mathbb{X} ; hence:

$\boldsymbol{\lambda} \in \mathbb{R}^{N_x} \quad \text{but} \quad \boldsymbol{\lambda} \notin \mathbb{X}$

That is, even though $\boldsymbol{\lambda}$ has the same dimension as \mathbf{x} , it does not occupy the same space as \mathbf{x} . Hence, $\mathbf{x} + \boldsymbol{\lambda}$ is not a legal operation. Obviously, $\mathbf{x}_1 + \mathbf{x}_2$ is a legal operation and so is $\boldsymbol{\lambda}_1 + \boldsymbol{\lambda}_2$. Thus, we have created a new vector space, out of thin air (!), that is different from \mathbb{X} . This space is called a *dual space*; it is the home space of $\boldsymbol{\lambda}$. The dual space is said to be dual to the original or *primal space*, \mathbb{X} . See Fig. 2.5.

Tech Talk: Strictly speaking, it is the *linear scalar function* ω , given by Eq. (2.4), that is called a covector. In fact, a dual vector space, or *covector space*, is the space of all linear scalar functions, $\omega : \mathbb{X} \rightarrow \mathbb{R}$ where \mathbb{X} can be any vector space. In our case, because \mathbb{X} is finite-dimensional, the difference between ω and $\boldsymbol{\lambda}$ seems nuanced. The vector $\boldsymbol{\lambda}$ is called a *representation* of the linear function ω . Following the age-old practice of convenient abuse of terminology, we call $\boldsymbol{\lambda}$ a covector. It is fair warning to say that we have taken much more liberty than terminology in introducing the concept of a covector. A mathematician will undoubtedly cringe at this presentation in much the same way as science fictionados react to the phrase “Vulcan mind trick.”



2.3 The Covectors in Optimal Control

The *common unit* for measurement in an optimal control problem is the *cost unit*; that is, the unit for measuring the value of J . On the basis of this *common unit*, we define a covector λ in exactly the manner as Eq. (2.5), where CU now stands for the *cost unit*! This particular covector is called the *costate*.

Illustrating the Concept: Constructing the Costate

In Brac:1, $x = (x, y, v) \in \mathbb{R}^3$. This implies that λ is in \mathbb{R}^3 , as well. To aid its bookkeeping, we deliberately choose the subscripts of the components of $\lambda \in \mathbb{R}^3$ to be λ_x, λ_y and λ_v so that it is properly tagged to the relevant

components of the state variables. Thus, the costate for Brac:1 is given by

$$\boldsymbol{\lambda} := \begin{bmatrix} \lambda_x \\ \lambda_y \\ \lambda_v \end{bmatrix} \begin{array}{l} TU/x\text{-units} \\ TU/y\text{-units} \\ TU/v\text{-units} \end{array} \quad (2.6)$$

where we have used the fact that the cost unit in Brac:1 is the same as the time unit, TU .

A quick examination of the data in Problem B reveals that

$$\mathbf{f}(\mathbf{x}, \mathbf{u}, t) \in \mathbb{R}^{N_x}$$

which can be elaborated to

$$\mathbf{f}(\mathbf{x}, \mathbf{u}, t) := \begin{bmatrix} f_1(\mathbf{x}, \mathbf{u}, t) \\ f_2(\mathbf{x}, \mathbf{u}, t) \\ \vdots \\ f_{N_x}(\mathbf{x}, \mathbf{u}, t) \end{bmatrix} \begin{array}{l} x_1\text{-units}/t\text{-units} \\ x_2\text{-units}/t\text{-units} \\ \vdots \\ x_{N_x}\text{-units}/t\text{-units} \end{array} \in \mathbb{R}^{N_x}$$

Illustrating the Concept: Constructing the Vector, $\mathbf{f}(\mathbf{x}, \mathbf{u}, t)$

In Brac:1, if we choose

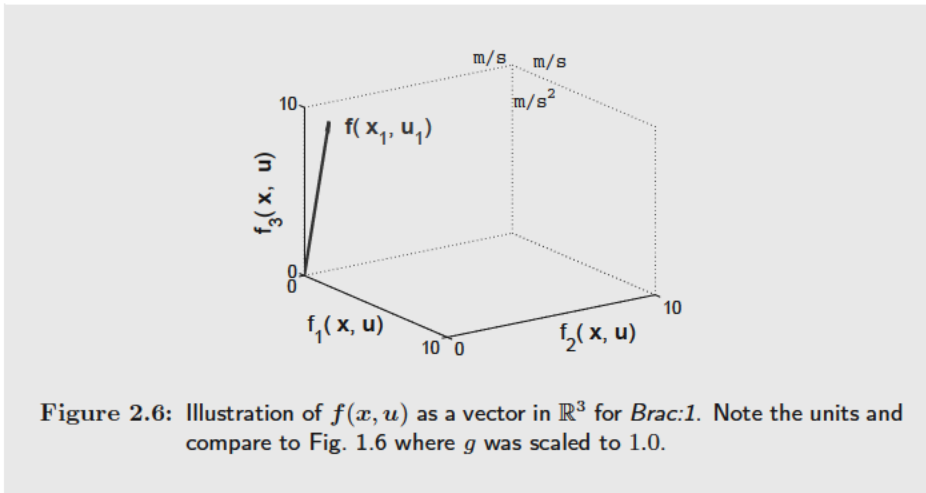
$$\mathbf{x}_1 = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} \begin{array}{l} m \\ m \\ m/s \end{array} \quad \text{and} \quad \mathbf{u}_1 = \pi/8$$

we get (by taking $g = 9.8 \text{ m/s}^2$)

$$\mathbf{f}(\mathbf{x}_1, \mathbf{u}_1) = \begin{bmatrix} 0.3827 \\ 0.9239 \\ 9.0540 \end{bmatrix} \begin{array}{l} m/s \\ m/s \\ m/s^2 \end{array} \in \mathbb{R}^3$$

This vector is plotted in Fig. 2.6.

Clearly, the dimension of $\mathbf{f}(\mathbf{x}, \mathbf{u}, t)$ is always exactly equal to the dimension



of \mathbb{X} ; however, as apparent by its units, note that

$$f(x, u, t) \in \mathbb{R}^{N_x} \quad \text{but} \quad f(x, u, t) \notin \mathbb{X}$$

By direct substitution, it can be easily verified that the “dot product”

$$\begin{aligned} \lambda \cdot f(x, u, t) \\ &:= \sum_{i=1}^{N_x} \lambda_i f_i(x, u, t) \\ &\quad (\text{in } CU/TU \text{ units}) \end{aligned}$$

is a legal operation and generates a scalar in CU/TU -units where TU is the unit of time or the independent variable. That is, we can use λ to measure $f(x, u, t)$.

Recall from the discussions on page 89 that the dot product, $x \cdot x$, is not a legal operation and neither is $f(x, u, t) \cdot f(x, u, t)$; but $\lambda \cdot f(x, u, t)$ is indeed legal. This point further reinforces the fact that λ is a *representation of a linear scalar function* (called a *linear form*) and that the space \mathbb{X} is not necessarily equipped with a legal dot product.

Computational Tip: In a computational environment, a majority of software (including DIDO) compute quantities like $(\mathbf{x} \cdot \mathbf{x})$, $(\mathbf{u} \cdot \mathbf{u})$, etc. with ruthless and total disregard for units. This computational policy is used near universally in many software products as a unified means to measure distances, lengths and so on. Hence, it is imperative that the designer of a *specific* computational problem choose judicious units that do not mask the relative importance of one component (or variable) against another. In Fig. 2.6, the numerical value of $f_3(\mathbf{x}_1, \mathbf{u}_1)$ dominates over the values of $f_1(\mathbf{x}_1, \mathbf{u}_1)$ and $f_2(\mathbf{x}_1, \mathbf{u}_1)$ but this is not the case in Fig. 1.6. Refer back to §1.1.4, page 20, for further discussions on scaling. DIDO also uses a myriad of covectors for computing various other quantities as discussed in much of this chapter; hence, the scaling procedures discussed in §1.1.4 must be balanced with this perspective. Additional concepts and details on *scaling and balancing* are discussed in [81] and [84].

The unit of measurement for the running cost, F , is CU/TU; hence, we can legally add $F(\mathbf{x}, \mathbf{u}, t)$ to the scalar, $\boldsymbol{\lambda} \cdot \mathbf{f}(\mathbf{x}, \mathbf{u}, t)$. The sum of these two quantities,

$$H(\boldsymbol{\lambda}, \mathbf{x}, \mathbf{u}, t) := F(\mathbf{x}, \mathbf{u}, t) + \boldsymbol{\lambda}^T \mathbf{f}(\mathbf{x}, \mathbf{u}, t) \quad (2.7)$$

in CU/TU units is called the *Hamiltonian for Problem B*.

Illustrating the Concept: Constructing the Hamiltonian

There is no running cost in *Brac:1*; hence the Hamiltonian is given by “dotting” $\boldsymbol{\lambda}$ to $\mathbf{f}(\mathbf{x}, \mathbf{u}, t)$ where $\boldsymbol{\lambda}$ is given by Eq. (2.6). Performing this simple operation, we have

$$H(\boldsymbol{\lambda}, \mathbf{x}, \mathbf{u}) := \lambda_x v \sin \theta + \lambda_y v \cos \theta + \lambda_v g \cos \theta \quad (2.8)$$

The Hamiltonian here is not an explicit function of time; hence, it is written as $H(\boldsymbol{\lambda}, \mathbf{x}, \mathbf{u})$ and not as $H(\boldsymbol{\lambda}, \mathbf{x}, \mathbf{u}, t)$. Note that the unit of this particular Hamiltonian is given by TU/TU; hence, H is dimensionless.

Tech Talk: The Hamiltonian is also known by other names in the literature: the Pontryagin-Hamiltonian, the Pontryagin H -function, the “unminimized Hamiltonian,” the pseudo-Hamiltonian (older literature) or the control Hamiltonian. Later, when it is necessary to use other Hamiltonians, we will use other adjectives to clarify the appropriate reference. In this book, the adjective-free Hamiltonian will always be the *control Hamiltonian* given by Eq. (2.7).

The costate λ is in \mathbb{R}^{N_x} ; hence, in principle, we can use it to measure any appropriate vector in \mathbb{R}^{N_x} . In the construction of the Hamiltonian, we used it to measure the vector $\mathbf{f}(\mathbf{x}, \mathbf{u}, t)$. We can use it to measure \mathbf{x} , as well, but it doesn't generate anything useful. This is because, it turns out, that what matters in optimal control is not the measurement of “input” vectors like \mathbf{x} and \mathbf{u} , but their “outputs” or combined effects given in the form of the problem data.

By inspection, Problem B has four data functions, E, F, \mathbf{f} and \mathbf{e} :

$$\begin{aligned}
 E : (\mathbf{x}_f, t_f) &\mapsto \mathbb{R} && \text{(CU)} \\
 \mathbf{e} : (\mathbf{x}_f, t_f) &\mapsto \mathbb{R}^{N_e} && \text{(e-units)} \\
 F : (\mathbf{x}, \mathbf{u}, t) &\mapsto \mathbb{R} && \text{(CU/TU)} \\
 \mathbf{f} : (\mathbf{x}, \mathbf{u}, t) &\mapsto \mathbb{R}^{N_x} && \text{(f-units = x-units/TU)}
 \end{aligned}$$

The functions E and F are scalar-valued. Only two of the data functions, \mathbf{f} and \mathbf{e} , are vector functions. We applied the covector λ to measure the vector $\mathbf{f}(\mathbf{x}, \mathbf{u}, t)$. We cannot apply λ to measure \mathbf{e} because it does not generate a legal operation either in terms of its units or dimensions. To see this, examine the vector, $\mathbf{e}(\mathbf{x}_f, t_f) \in \mathbb{R}^{N_e}$, more closely:

$$\mathbf{e}(\mathbf{x}_f, t_f) := \begin{bmatrix} e_1(\mathbf{x}_f, t_f) \\ e_2(\mathbf{x}_f, t_f) \\ \vdots \\ e_{N_e}(\mathbf{x}_f, t_f) \end{bmatrix} \begin{array}{l} e_1\text{-units} \\ e_2\text{-units} \\ \vdots \\ e_{N_e}\text{-units} \end{array} \in \mathbb{R}^{N_e}$$

In general $N_e \neq N_x$; hence, the operation $\lambda \cdot \mathbf{e}(\mathbf{x}_f, t_f)$ cannot be performed. Even in the special case of $N_e = N_x$, the dot product $\lambda \cdot \mathbf{e}(\mathbf{x}_f, t_f)$ does not generate a legal operation in terms of units. This suggests that we need to

define an *endpoint covector*, in a manner similar to λ , and given by

$$\boldsymbol{\nu} := \begin{bmatrix} \nu_1 \\ \nu_2 \\ \vdots \\ \nu_{N_e} \end{bmatrix} \begin{array}{l} CU/e_1\text{-units} \\ CU/e_2\text{-units} \\ \vdots \\ CU/e_{N_e}\text{-units} \end{array} \quad (2.9)$$

This covector legalizes the “dot product”

$$\boldsymbol{\nu} \cdot \mathbf{e}(\mathbf{x}_f, t_f) := \sum_{i=1}^{N_e} \nu_i e_i(\mathbf{x}_f, t_f)$$

and produces a measurement mechanism for the data given by the \mathbf{e} -function. It can be easily verified that the unit of the scalar resulting from this operation is CU . This is exactly the same as the unit of E ; hence, we can legally add E to the scalar product $\boldsymbol{\nu} \cdot \mathbf{e}(\mathbf{x}_f, t_f)$, and generate the quantity,

$$\boxed{\bar{E}(\boldsymbol{\nu}, \mathbf{x}_f, t_f) := E(\mathbf{x}_f, t_f) + \boldsymbol{\nu}^T \mathbf{e}(\mathbf{x}_f, t_f)} \quad (2.10)$$

The function \bar{E} is called the *Endpoint Lagrangian*.

Illustrating the Concept: Constructing the Endpoint Lagrangian

For Problem *Brac:1*, $N_e = 2$; hence $\boldsymbol{\nu} \in \mathbb{R}^2$. This implies that we can define

$$\boldsymbol{\nu} := \begin{bmatrix} \nu_1 \\ \nu_2 \end{bmatrix} \begin{array}{l} TU/x\text{-units} \\ TU/y\text{-units} \end{array} \quad (2.11)$$

Then, the *Endpoint Lagrangian* is given by

$$\bar{E}(\boldsymbol{\nu}, \mathbf{x}_f, t_f) := t_f + \nu_1(x_f - x^f) + \nu_2(y_f - y^f), \quad \text{in } TUs \quad (2.12)$$

Thus, through a construction of two covectors, λ and $\boldsymbol{\nu}$, we have essentially packed all four data functions (E, F, \mathbf{f} and \mathbf{e}) of Problem *B* in just two bags of scalar functions, H and \bar{E} . These scalar functions form the bases for generating a collection of conditions known as Pontryagin's Principle.

2.4 Introducing Pontryagin's Principle

By definition, the cost functional J is the prescribed means to measure performance. That is, we say $u^1(\cdot)$ is a better control trajectory than $u^2(\cdot)$ if the cost for executing the former is lower than the latter. Hence, we can say that J is the chosen means to measure the “length” of $u(\cdot)$, and that the problem is to find the “smallest” $u(\cdot)$ measured in terms of the values of J . Recall that we are not interested in $u(\cdot)$ only, and that the decision or independent variables are the triple[‡]

$$\{x(\cdot), u(\cdot), t_f\}$$

That is, J measures the entire triple by some formula

$$J : \{x(\cdot), u(\cdot), t_f\} \mapsto \mathbb{R}$$

chosen by the problem designer. See Fig. 2.7. In this context, it is the problem

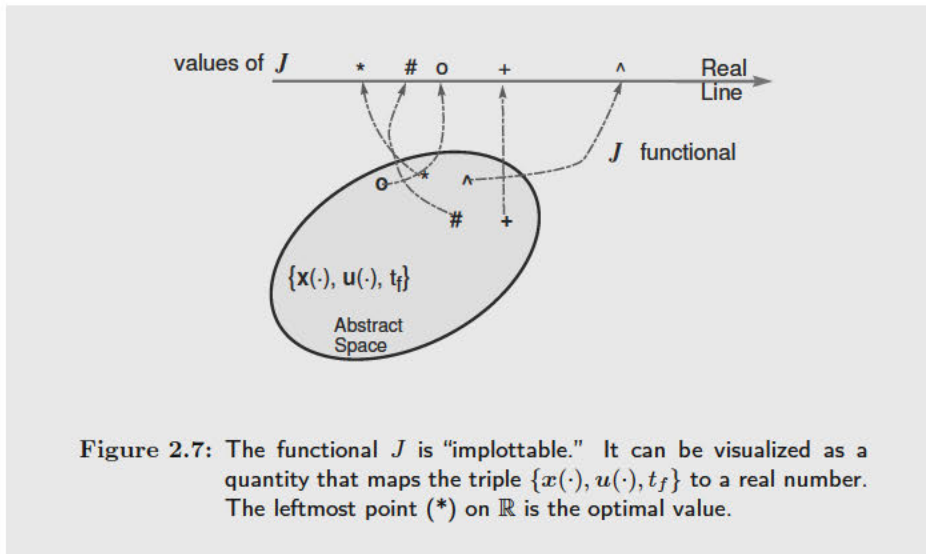


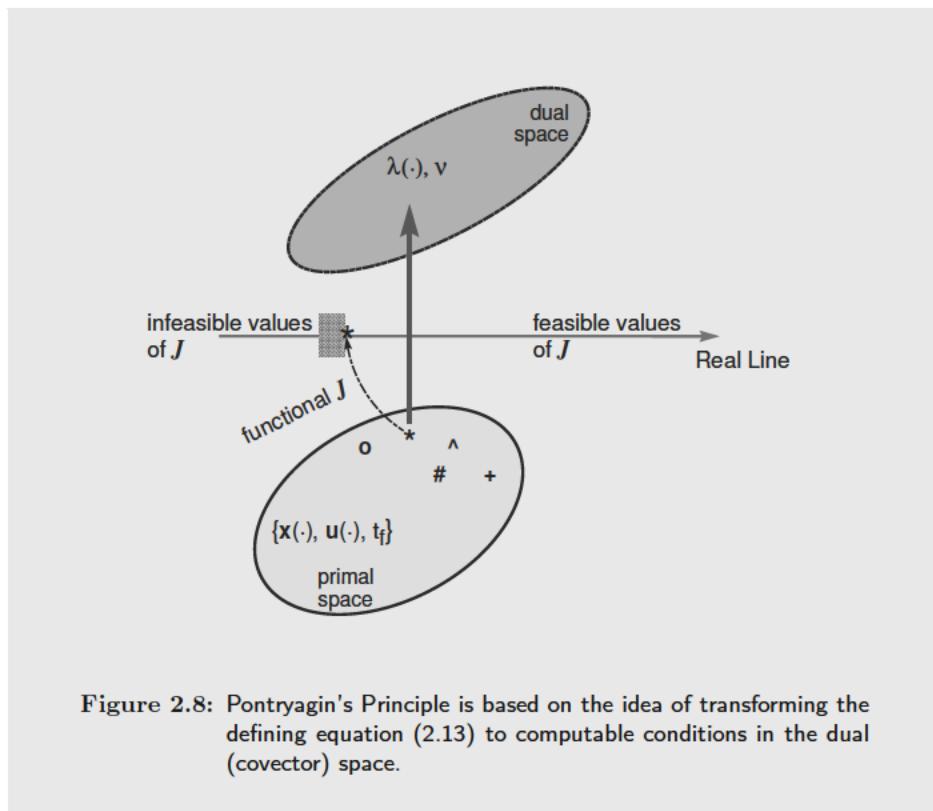
Figure 2.7: The functional J is “implottable.” It can be visualized as a quantity that maps the triple $\{x(\cdot), u(\cdot), t_f\}$ to a real number. The leftmost point ($*$) on \mathbb{R} is the optimal value.

designer who stipulates that J is the mechanism to measure the system trajectory; hence, we declare the decision variables, $x^*(\cdot)$, $u^*(\cdot)$ and t_f^* , to be optimal if

$$J[x^*(\cdot), u^*(\cdot), t_f^*] \leq J[x(\cdot), u(\cdot), t_f] \tag{2.13}$$

[‡]Refer back to Section 1.4.3 (page 48) on avoiding common errors # 2.

for all $x(\cdot), u(\cdot)$ and t_f that are feasible. Pontryagin's Principle is based on the idea that this *defining inequality* can be transferred to the *instantaneous minimization of the Hamiltonian function* provided that the covectors are chosen appropriately. In other words, we have, so far, defined λ and ν only in terms of their units and dimensions, and nothing else. As these covectors are the measurement devices, Pontryagin's Principle is based on translating the definition given by Eq. (2.13) to additional requirements on λ and ν . These concepts can be visualized as shown in Fig. 2.8.



2.4.1 The Basics

The Hamiltonian packs an enormous amount of information on the optimal control problem. Recall that

$$H(\lambda, x, u, t) := F(x, u, t) + \lambda^T f(x, u, t) \quad (\text{in } CU/TU \text{ units})$$

From the elementary observation

$$\frac{\partial H}{\partial \lambda} = f(\mathbf{x}, \mathbf{u}, t)$$

we can immediately write:

$$\dot{\mathbf{x}} = \frac{\partial H}{\partial \lambda} \tag{2.14}$$

That is, we can recover the state dynamical equations from the Hamiltonian.

Study Problem 2.2

Show that the units of $\partial H/\partial \lambda$ are the same as that of $\dot{\mathbf{x}}$.

Next, observe that

$$\text{units of } \frac{\partial H}{\partial \mathbf{x}} = \frac{CU/TU}{\mathbf{x}\text{-units}} = \frac{CU/\mathbf{x}\text{-units}}{TU} = \frac{\lambda\text{-units}}{TU} = \text{units of } \dot{\lambda}$$

This is not a coincidence! It turns out that the costate satisfies the adjoint differential equation or, simply, the **adjoint equation**

$$-\dot{\lambda} = \frac{\partial H}{\partial \mathbf{x}} \tag{2.15}$$

where the minus sign is part of the mathematical technicality that makes Eq. (2.15) into an “adjoint”: The adjoint of the differential operator, d/dt is given by $-d/dt$. This is why the costate is also known as the **adjoint covector**.

Illustrating the Concept: Constructing the Adjoint Equations

From Eq. (2.15), the adjoint equations for Brac:1 are given by

$$\begin{aligned} -\dot{\lambda}_x &:= \partial_x H(\lambda, \mathbf{x}, \mathbf{u}) &= 0 \\ -\dot{\lambda}_y &:= \partial_y H(\lambda, \mathbf{x}, \mathbf{u}) &= 0 \\ -\dot{\lambda}_v &:= \partial_v H(\lambda, \mathbf{x}, \mathbf{u}) &= \lambda_x \sin \theta + \lambda_y \cos \theta \end{aligned} \tag{2.16}$$

The H -function is called a Hamiltonian because the state-costate pair satisfies the same differential equation as the one encountered in **Hamiltonian**

mechanics

$$\begin{aligned} \dot{q} &= \frac{\partial \mathcal{H}}{\partial p} \\ -\dot{p} &= \frac{\partial \mathcal{H}}{\partial q} \end{aligned} \tag{2.17}$$

where q is a generalized coordinate and p is the momentum.

The adjoint covector trajectory can be visualized in several different ways. In one approach that parallels the concept illustrated in Fig. 2.5 on page 93, it can be viewed as a “shadow trajectory” that shadows the state trajectory in dual space; see Fig. 2.9.

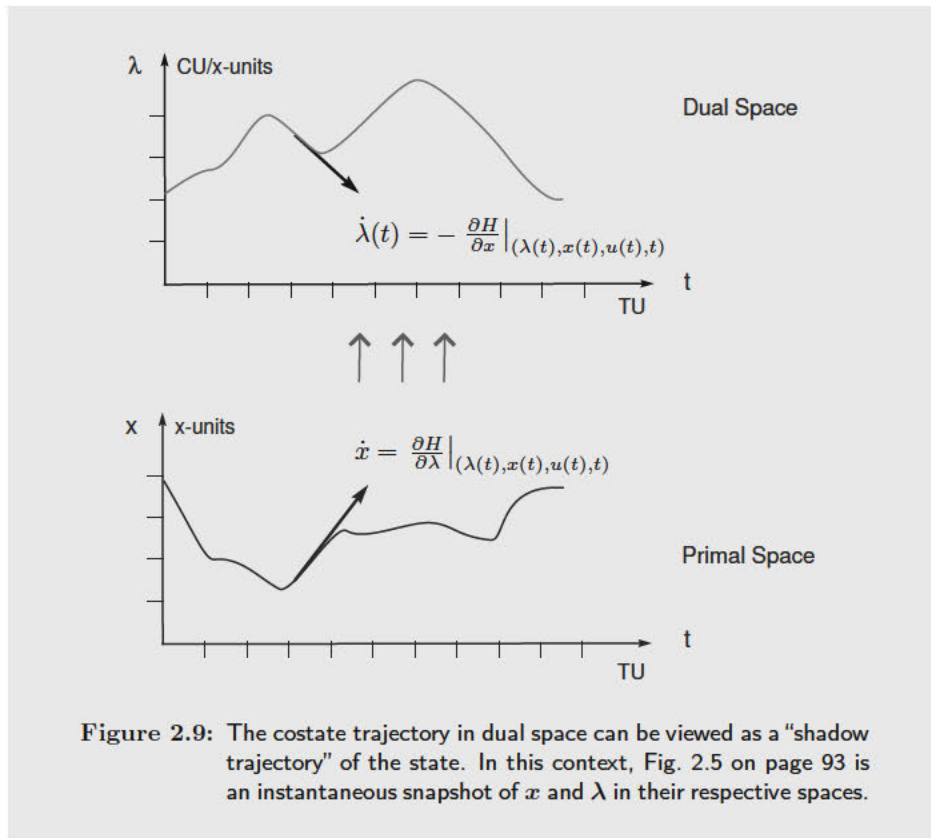


Figure 2.9: The costate trajectory in dual space can be viewed as a “shadow trajectory” of the state. In this context, Fig. 2.5 on page 93 is an instantaneous snapshot of x and λ in their respective spaces.

For the control function $u(\cdot)$ to be optimal, Pontryagin's Principle requires that u globally minimize the Hamiltonian for every $t \in [t_0, t_f]$. In other words, minimizing the Hamiltonian with respect to u (while holding λ and x

constant) yields a candidate solution for the optimal control. Hence, solving the *pointwise static* (finite dimensional) optimization problem in the parameter \mathbf{u}

$$\text{(HMC)} \quad \left\{ \begin{array}{l} \text{Minimize}_{\mathbf{u}} \quad H(\boldsymbol{\lambda}, \mathbf{x}, \mathbf{u}, t) \\ \text{Subject to} \quad \mathbf{u} \in \mathbb{U} \end{array} \right.$$

at each instant of time t , yields a candidate function for the optimal control. This simple statement is called the *Hamiltonian Minimization Condition* (HMC) and is the heart of Pontryagin's Principle.

Illustrating the Concept: Constructing Problem HMC

Problem HMC for Brac:1 is quite simply formulated as the unconstrained minimization problem

$$\left\{ \begin{array}{l} \text{Minimize}_{\theta} \quad H(\boldsymbol{\lambda}, \mathbf{x}, \mathbf{u}) := \lambda_x v \sin \theta + \lambda_y v \cos \theta + \lambda_v g \cos \theta \\ \text{Subject to} \quad \theta \in \mathbb{R} \end{array} \right.$$

where we have taken the usual liberty of calling θ a real number rather than its more appropriate representation, $\theta \in S^1$.

Tech Alert: The minimization in Problem HMC is to be performed only with respect to \mathbf{u} , and hence the notation, $\text{Minimize}_{\mathbf{u}}$ (i.e., H is to be regarded as a function of \mathbf{u} *only*, in Problem HMC).

Let us further clarify the meaning of Problem HMC by denoting its objective function as $R(\mathbf{u}, t)$:

$$(\mathbf{u}, t) \xrightarrow{R} H(\boldsymbol{\lambda}, \mathbf{x}, \mathbf{u}, t)$$

That is, let $R(\mathbf{u}, t)$ be the same as $H(\boldsymbol{\lambda}, \mathbf{x}, \mathbf{u}, t)$ with $\boldsymbol{\lambda}$ and \mathbf{x} held constant. Then Problem HMC can be written without the clutter of $\boldsymbol{\lambda}$ and \mathbf{x} as:

$$\left\{ \begin{array}{l} \text{Minimize}_{\mathbf{u}} \quad R(\mathbf{u}, t) \\ \text{Subject to} \quad \mathbf{u} \in \mathbb{U} \end{array} \right. \quad (\text{for each } t)$$

This concept is illustrated in Fig. 2.10 where a snapshot of the Hamiltonian as a function of u only (i.e., R) is taken at discrete times.

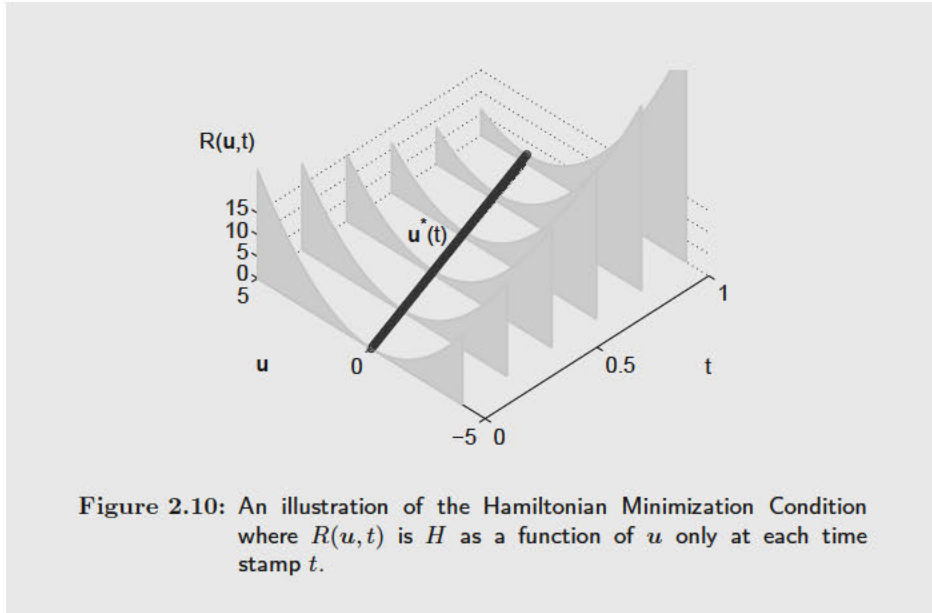


Figure 2.10: An illustration of the Hamiltonian Minimization Condition where $R(u, t)$ is H as a function of u only at each time stamp t .

Problem HMC is a finite-dimensional optimization problem; hence, all tools from finite dimensional optimization theory can be used to solve this problem.

Illustrating the Concept: Solving Problem HMC - 1/2

Problem HMC for Brac:1 can be solved quite readily. Since the problem is unconstrained, we can set $\partial_\theta H(\lambda, x, u) = 0$ at each instant of time t . This immediately generates the equation:

$$\lambda_x v \cos \theta - \lambda_y v \sin \theta - \lambda_v g \sin \theta = 0 \tag{2.18}$$

According to Pontryagin's Principle, this equation must hold at each instant of time; hence, we can write:

$$\lambda_x(t)v(t) \cos \theta(t) - \lambda_y(t)v(t) \sin \theta(t) - \lambda_v(t)g \sin \theta(t) = 0$$

As a sidebar, note that $\partial_\theta^2 H(\lambda, x, u) = -H(\lambda, x, u)$. Thus, the convexity condition $\partial_\theta^2 H(\lambda, x, u^) \geq 0$ requires that $H(\lambda, x, u^*) \leq 0$ over the entire trajectory.*

Deferring a discussion on the details of solving it to Section 2.5, assume, for the moment, that Problem HMC can be solved. Then, its solution \mathbf{u}^* is given by some function \mathbf{g} :

$$(\boldsymbol{\lambda}, \mathbf{x}, t) \xrightarrow{\mathbf{g}} \mathbf{u}^* \tag{2.19}$$

This control function $\mathbf{u}^* = \mathbf{g}(\boldsymbol{\lambda}, \mathbf{x}, t)$ is known as the Pontryagin-extremal control or simply the *extremal control*.

Illustrating the Concept: Solving Problem HMC - 2/2

The extremal control for Problem Brac:1 is obtained from Eq. (2.18). This equation can be solved quite readily as:

$$\underbrace{\theta^*}_{\mathbf{u}^*} = \tan^{-1} \left(\underbrace{\frac{\lambda_x v}{\lambda_y v + \lambda_v g}}_{\mathbf{g}(\boldsymbol{\lambda}, \mathbf{x})} \right) \tag{2.20}$$

As simple as this equation seems, note that inverses of trigonometric functions are multi-valued.

A solution to Problem HMC generates an extremal control as some function \mathbf{g} of the costate, state and time:

$$\mathbf{u}^* = \mathbf{g}(\boldsymbol{\lambda}, \mathbf{x}, t) \tag{2.21}$$

Substituting this function in the dynamics, $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, t)$, eliminates \mathbf{u} and generates a new differential equation for the states:

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{f}(\mathbf{x}, \mathbf{u}, t) \\ &= \mathbf{f}(\mathbf{x}, \mathbf{g}(\boldsymbol{\lambda}, \mathbf{x}, t), t) \\ &:= \tilde{\mathbf{f}}_1(\mathbf{x}, \boldsymbol{\lambda}, t) \end{aligned} \tag{2.22}$$

The state trajectory $\mathbf{x}^*(\cdot)$ resulting from Eq. (2.22) is called an an *extremal state trajectory* or an extremal *arc* (in some classical texts), and the state-control function pair, $\{\mathbf{x}^*(\cdot), \mathbf{u}^*(\cdot)\}$, is called an *extremal* solution.

Illustrating the Concept: Generating ODEs for the Extremal States

The extremal states for Brac:1 satisfy the differential equations:

$$\dot{\mathbf{x}} := \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} v \sin \left(\tan^{-1} \left(\frac{\lambda_x v}{\lambda_y v + \lambda_v g} \right) \right) \\ v \cos \left(\tan^{-1} \left(\frac{\lambda_x v}{\lambda_y v + \lambda_v g} \right) \right) \\ g \cos \left(\tan^{-1} \left(\frac{\lambda_x v}{\lambda_y v + \lambda_v g} \right) \right) \end{bmatrix}$$

$\underbrace{\hspace{10em}}_{\tilde{\mathbf{f}}_1(x, \lambda)}$

It is apparent that even though we may have found a candidate optimal control \mathbf{u}^* by solving Problem HMC, a production of the extremal states is not complete because we need a quantitative knowledge of the costates. This knowledge can be obtained by substituting \mathbf{u}^* in the adjoint equation in much the same way as was done in producing Eq. (2.22). This procedure produces a new differential equation for the costates that does not depend on \mathbf{u} . That is, substituting Eq. (2.21) in the adjoint equation, $\dot{\boldsymbol{\lambda}} = -\partial_{\mathbf{x}}H(\boldsymbol{\lambda}, \mathbf{x}, \mathbf{u}, t)$, generates a new differential equation for the costates:

$$\begin{aligned} -\dot{\boldsymbol{\lambda}} &= \partial_{\mathbf{x}}H(\boldsymbol{\lambda}, \mathbf{x}, \mathbf{u}, t) \\ &= \partial_{\mathbf{x}}H(\boldsymbol{\lambda}, \mathbf{x}, \mathbf{u}, t) \Big|_{\mathbf{u}=\mathbf{g}(\boldsymbol{\lambda}, \mathbf{x}, t)} \\ &:= \tilde{\mathbf{f}}_2(\mathbf{x}, \boldsymbol{\lambda}, t) \end{aligned} \tag{2.23}$$

Illustrating the Concept: Generating ODEs for the Extremal Costates

The extremal costates for Brac:1 must satisfy the differential equations:

$$\begin{aligned} -\dot{\lambda}_x &= 0 \\ -\dot{\lambda}_y &= 0 \\ -\dot{\lambda}_v &= \lambda_x \sin \left[\tan^{-1} \left(\frac{\lambda_x v}{\lambda_y v + \lambda_v g} \right) \right] \\ &\quad + \lambda_y \cos \left[\tan^{-1} \left(\frac{\lambda_x v}{\lambda_y v + \lambda_v g} \right) \right] \end{aligned} \tag{2.24}$$

$\underbrace{\hspace{10em}}_{\tilde{\mathbf{f}}_2(x, \lambda)}$

From the general equation of (2.23) and the example of Eq. (2.24) it is evident

that we need a quantitative knowledge of the states to produce a quantitative knowledge of the costates and vice versa. That is, solving Problem HMC leads to a pair of of coupled differential equations:

$$\begin{bmatrix} \dot{\mathbf{x}} \\ -\dot{\boldsymbol{\lambda}} \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{f}}_1(\mathbf{x}, \boldsymbol{\lambda}, t) \\ \tilde{\mathbf{f}}_2(\mathbf{x}, \boldsymbol{\lambda}, t) \end{bmatrix}$$

In other words, we need to solve a system of $2N_x$ state-costate differential equations *simultaneously*. To achieve this task, we need $2N_x$ point conditions: preferably N_x initial conditions on the states and N_x initial conditions on the costates. We also need the value of the initial (clock) time t_0 to propagate the initial condition as well the value of the final time t_f to stop the propagation. In all, we need $(2N_x + 2)$ numbers or point conditions. Collecting all of the available point data for Problem B , we have:

$$\begin{aligned} t_0 &= t^0 && (1 \text{ equation}) \\ \mathbf{x}(t_0) &= \mathbf{x}^0 && (N_x \text{ equations}) \\ \mathbf{e}(\mathbf{x}(t_f), t_f) &= \mathbf{0} && (N_e \text{ equations}) \end{aligned}$$

Illustrating the Concept: Available Point Conditions

The totality of point conditions for Brac:1 are:

$$\begin{aligned} t_0 &= 0 && (1 \text{ equation}) \\ (x_0, y_0, v_0) &= (0, 0, 0) && (3 \text{ equations}) \\ (x_f - x^f, y_f - y^f) &= (0, 0) && (2 \text{ equations}) \end{aligned}$$

(Recall that $N_x = 3$ and $N_e = 2$.)

Clearly, not only do we not have any initial condition on $\boldsymbol{\lambda}$, we also do not have the requisite $(2N_x + 2)$ point conditions. The deficit of point conditions is given by

$$\underbrace{(2N_x + 2)}_{\text{required}} - \underbrace{(1 + N_x + N_e)}_{\text{available}} = \underbrace{N_x + 1 - N_e}_{\text{deficit}} \tag{2.25}$$

For the Brac:1 formulation of the Brachistochrone problem, this deficit of point conditions is $8 - 6 = 2$.

Study Problem 2.3

1. Under what conditions is there a zero deficit of point conditions?
2. Is it possible to have $N_e > N_x + 1$?

2.4.2 In Search of the Missing Boundary Conditions

A deficit of point conditions in the Hamiltonian system of $2N_x$ differential equations implies that there are additional optimality conditions that await discovery. This point should also be apparent from the fact that, so far, we have not used the endpoint cost function E in generating any of the optimality conditions discussed in the previous section.

In seeking the missing point conditions, we cannot just add additional conditions on \mathbf{x}_0 or \mathbf{x}_f without changing the problem. This implies that the missing point conditions must be on $\boldsymbol{\lambda}(t_0)$ and/or on $\boldsymbol{\lambda}(t_f)$.

So far, we have only used the H function and hence the pair (F, \mathbf{f}) . We have not used the pair (E, \mathbf{e}) and hence the function \bar{E} . As part of this analysis, consider the gradient of \bar{E} with respect to \mathbf{x}_f :

$$\frac{\partial \bar{E}(\boldsymbol{\nu}, \mathbf{x}_f, t_f)}{\partial \mathbf{x}_f} := \begin{bmatrix} \partial_{x_{1f}} \bar{E}(\boldsymbol{\nu}, \mathbf{x}_f, t_f) \\ \partial_{x_{2f}} \bar{E}(\boldsymbol{\nu}, \mathbf{x}_f, t_f) \\ \vdots \\ \partial_{x_{N_x f}} \bar{E}(\boldsymbol{\nu}, \mathbf{x}_f, t_f) \end{bmatrix} \begin{array}{l} CU/x_{1f}\text{-units} \\ CU/x_{2f}\text{-units} \\ \vdots \\ CU/x_{N_x f}\text{-units} \end{array} \quad (2.26)$$

Obviously, the units of $\partial_{\mathbf{x}_f} \bar{E}$ are the same as the units of $\boldsymbol{\lambda}$. Given that \bar{E} is a function of the final time variables, it is not too hard to guess that

$$\boldsymbol{\lambda}(t_f) = \frac{\partial \bar{E}}{\partial \mathbf{x}_f} \quad (2.27)$$

is a “missing” boundary condition. This equation is called the *terminal Transversality Condition*.

Illustrating the Concept: Terminal Transversality Condition

From Eq. (2.12) on page 98, the endpoint Lagrangian for Brac:1 is given by

$$\bar{E}(\boldsymbol{\nu}, \mathbf{x}_f, t_f) := t_f + \nu_1(x_f - x^f) + \nu_2(y_f - y^f)$$

Applying Eq. (2.27), we get:

$$\begin{aligned}\lambda_x(t_f) &= \frac{\partial \bar{E}}{\partial x_f} = \nu_1 \\ \lambda_y(t_f) &= \frac{\partial \bar{E}}{\partial y_f} = \nu_2 \\ \lambda_v(t_f) &= \frac{\partial \bar{E}}{\partial v_f} = 0\end{aligned}\tag{2.28}$$

It is clear that the transversality condition for $\lambda_x(t_f)$ and $\lambda_y(t_f)$ provide no new information in the sense that it says that these two unknown quantities are equal to two other unknown quantities. The story on the transversality condition $\lambda_v(t_f) = 0$ is different; it provides a non-trivial boundary condition. Recall that we need 2 point conditions for Brac:1; hence, we still need one more point condition to complete the circle.

Study Problem 2.4

Show that the units of $\lambda_x, \lambda_y, \nu_1$ and ν_2 in Eq. (2.28) are all consistent with their definitions given by Eqs. (2.5) and (2.9).

Equation (2.27) provides us N_x point conditions for $\boldsymbol{\lambda}(t)$. From Eq. (2.25) we needed $(N_x + 1 - N_e)$ point conditions; hence, the new number of missing boundary conditions are

$$(N_x + 1 - N_e) - N_x = 1 - N_e$$

This number seems a little strange as we now seem to have an excess of point conditions when $N_e > 1$; however, as evident from Eq. (2.28), we do not actually have N_x point conditions from Eq. (2.27) but something less. This is because the N_x equations from the terminal transversality condition contain an additional

set of N_e unknowns in terms of $\boldsymbol{\nu} \in \mathbb{R}^{N_e}$. In other words, the effective number of equations from Eq. (2.27) is not N_x but $(N_x - N_e)$. This why in Brac:1, we were able to extract only $3 - 2 = 1$ useful condition by applying the transversality condition. Thus, the correct number of missing boundary conditions is,

$$(N_x + 1 - N_e) - (N_x - N_e) = 1$$

This “last” missing point condition comes from the **Hamiltonian Value Condition**[§]

$$H[@t_f] = -\frac{\partial \bar{E}}{\partial t_f} \quad (2.29)$$

where we have used a shorthand notation, $H[@t_f]$, for the value of the Hamiltonian at $t = t_f$; that is,

$$H[@t_f] \equiv H(\boldsymbol{\lambda}(t_f), \mathbf{x}(t_f), \mathbf{u}(t_f), t_f)$$

Illustrating the Concept: Hamiltonian Value Condition

For Brac:1, we have

$$\frac{\partial \bar{E}}{\partial t_f} = 1 \quad (2.30)$$

Hence, the Hamiltonian value condition is given by

$$\lambda_x(t_f)v(t_f)\sin\theta(t_f) + \lambda_y(t_f)v(t_f)\cos\theta(t_f) + \lambda_v(t_f)g\cos\theta(t_f) = -1 \quad (2.31)$$

Study Problem 2.5

Write down the missing units in Eq. (2.29). What are the units in Eq. (2.30)?

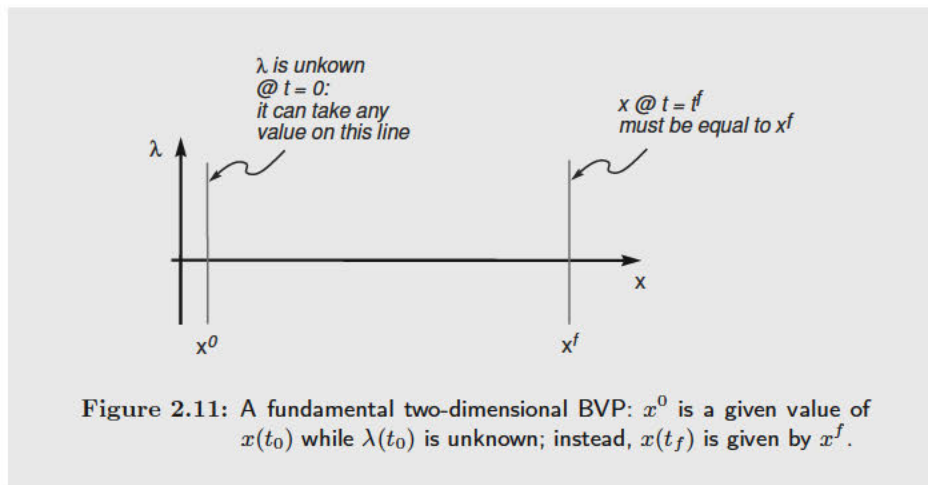
Equations (2.27) and (2.29) complete the set of missing boundary conditions necessary to obtain a *candidate solution* to the optimal control problem. The collection of all these equations constitutes a special **boundary value prob-**

[§]The Hamiltonian value condition applies to the *lower* Hamiltonian discussed later in Section 2.5.5. No serious error results in mixing up these two Hamiltonians at the the present time.

lem (BVP). We denote the BVP generated from Problem B as Problem B^λ .

2.4.3 Formulating the Boundary Value Problem B^λ

A generic BVP is essentially two or more differential equations where some of the point conditions are specified at the initial time and the remainder at the final time; see Fig. 2.11. If all conditions are specified at the initial time, it is called



an *initial value problem*, which is generally considered to be a solved problem as a consequence of good numerical integration techniques like Runge-Kutta methods.

The BVP developed in the preceding two subsections is not generic; it is *structured in a special manner* as follows:

- A pair of $2N_x$ (state-costate) differential equations

$$\begin{aligned}\dot{x} &= \tilde{f}_1(x, \lambda, t) && (N_x \text{ equations}) \\ -\dot{\lambda} &= \tilde{f}_2(x, \lambda, t) && (N_x \text{ equations})\end{aligned}$$

- $2N_x$ effective boundary conditions given in the algebraic form

$$\begin{aligned} x_0 &= x^0 && (N_x \text{ equations}) \\ e(x_f, t_f) &= 0 && (N_e \text{ equations}) \\ \lambda(t_f) &= \frac{\partial \bar{E}}{\partial x_f} && ((N_x - N_e) \text{ effective equations}) \end{aligned}$$

- Two clock conditions (to start and stop)

$$\begin{aligned} t_0 &= t^0 && (1 \text{ equation}) \\ H[\textcircled{t}_f] &= -\frac{\partial \bar{E}}{\partial t_f} && (1 \text{ equation}) \end{aligned}$$

Thus, the application of Pontryagin's Principle has not solved Problem B ; instead, it has converted or "mapped" it to a BVP denoted by Problem B^λ ; see Fig. 2.12. *Note that this BVP is always given by an even number of differential equations (i.e., $2N_x$).*

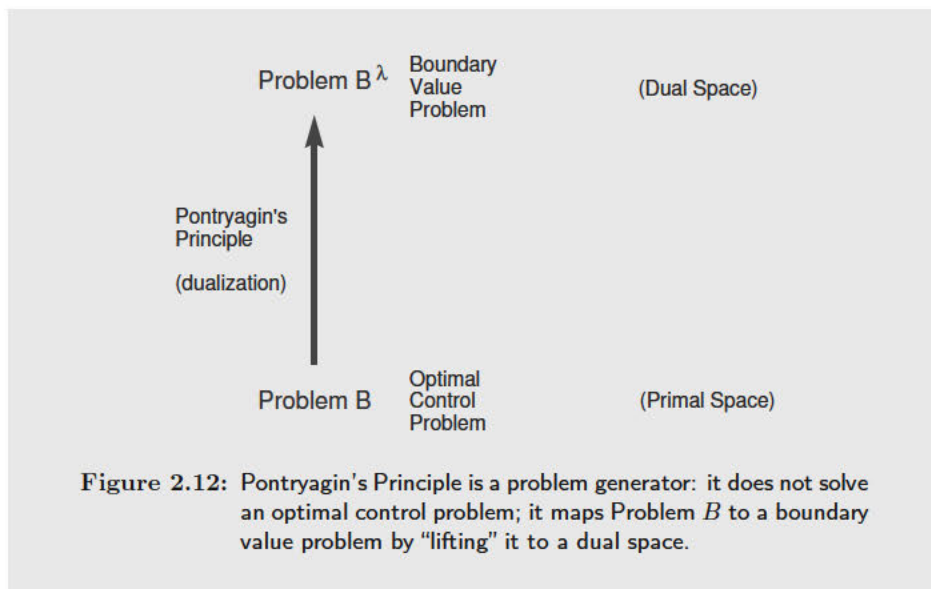


Figure 2.12: Pontryagin's Principle is a problem generator: it does not solve an optimal control problem; it maps Problem B to a boundary value problem by "lifting" it to a dual space.

Study Problem 2.6

Collect the complete set of differential and boundary conditions derived in the preceding pages for Brac:1. Obtain, or discuss how to obtain, a solution to this BVP. Explore a numerical solution for this BVP in MATLAB or any other tool set. Discuss the pitfalls.

2.4.4 Solving the BVP: Work, Plug, Pray

It is possible to “solve” the Brachistochrone problem by a simpler process than the procedure identified in Study Problem 2.6. One such process is semi-analytic; it is discussed later in Section 3.1, page 171. Semi-analytic procedures, often employed in academic-strength problems, usually use clever coordinate transformations and other mathematical “tricks” to avoid or beat down the BVP to something “manageable.” While such ad hoc techniques are indeed useful and important for the analysis of specific problems, they are not portable to the broader Problem B . Furthermore, ad hoc techniques have the appearance of making “easy” problems look hard. On the other hand, the systematic process summarized in Fig. 2.12 is generic and extremely powerful. The generation of Problem B^λ might seem arduous at first, but note that the process is routine; hence, it can be algorithmized. The routineness of this process is demonstrated in Chapter 3 while an algorithm to solve Problem B^λ is encoded in DIDO.

**Easy Problems Made Hard**

Recall that the production of the differential equations

$$\begin{aligned}\dot{\mathbf{x}} &= \tilde{\mathbf{f}}_1(\mathbf{x}, \boldsymbol{\lambda}, t) \\ -\dot{\boldsymbol{\lambda}} &= \tilde{\mathbf{f}}_2(\mathbf{x}, \boldsymbol{\lambda}, t)\end{aligned}$$

assumed that we can solve Problem HMC explicitly in an equation form:

$$\mathbf{u} = \mathbf{g}(\boldsymbol{\lambda}, \mathbf{x}, t)$$

Unfortunately, this is often not possible and is the subject of Section 2.5 on page 114. In other words, it is not easy or possible to produce an “unconstrained”

BVP; more often, it can be formulated only as a *differential-algebraic inequality*. Standard methods for solving differential-algebraic inequalities are based on “optimal control techniques,” such as *collocation methods*, which are briefly introduced later in Section 2.9.2, page 157. This, coming to a full circle, implies that Problem B^λ is not necessarily simpler than Problem B ! While this realization may seem “obvious” today, note, however, that it is result of a culmination of research that took place from the 1960s to the year 2000.

Interestingly, if a problem is posed as a differential-algebraic inequality, it may be possible to “map it” down to an optimal control problem, that is, a down arrow in Fig. 2.12. In this case, the differential equations may be viewed as a manifestation of something more fundamental: an optimal control problem!



Symplectic BVPs

Problem B^λ is not a generic BVP: the $2N_x$ differential equations have a *symplectic* structure

$$\dot{\mathbf{y}} = \mathfrak{J} \nabla_{\mathbf{y}} H(\mathbf{y}, \mathbf{u}, t), \quad \mathfrak{J} := \begin{bmatrix} \mathbf{0} & \mathbf{I}_d \\ -\mathbf{I}_d & \mathbf{0} \end{bmatrix} \quad (2.32)$$

where $\mathbf{y} := (\mathbf{x}, \boldsymbol{\lambda}) \in \mathbb{R}^{N_x} \times \mathbb{R}^{N_x}$ and \mathbf{I}_d is an $N_x \times N_x$ identity matrix. Hence, in principle, the entire power of symplectic vector space analysis can be brought to bear on analyzing Problem B^λ . Note, however, that the analysis of Problem B^λ is a little more complicated than those encountered in mathematical physics due to the presence of \mathbf{u} . Conversely, the entire power of optimal control theory can be brought to bear on the analysis of problems in mathematical physics. In this spirit, as physicists discover that more and more of Nature's laws are symplectic, it is apparent that Nature must be solving an optimal control problem (à la Hamilton's principle of “least” action). Thus, a grand physics Problem B awaits discovery.

2.5 Minimizing the Hamiltonian

It is evident from the previous section that the problem of minimizing the Hamiltonian with respect to the control \mathbf{u} (i.e., Problem HMC) is a critical step in applying Pontryagin's Principle for any given problem. Problem HMC is a *static*

optimization problem that must be performed at each instant of time. In many instances, \mathbb{U} is a continuous set; then, Problem HMC reduces to a *nonlinear programming (NLP)* problem.

2.5.1 How Things Go Wrong

Too often, Problem HMC is erroneously oversimplified to the condition:[¶]

$$\left. \frac{\partial H}{\partial u} \right|_{u=u^*} = 0 \quad (2.33)$$

As sketched in Fig. 2.13, Eq. (2.33) would isolate points b and c as candidate solutions, neither of which are correct. Any claim that the *convexity condition*,

$$\left. \frac{\partial^2 H}{\partial u^2} \right|_{u=u^*} \geq 0 \quad (2.34)$$

would isolate point c from point b , and hence would generate a *local* minimum to Problem B is also not true! *This is simply because there is no guarantee that*

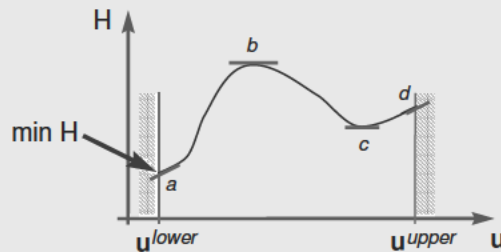


Figure 2.13: A snapshot of the graph of a possible Hamiltonian sketched as a function of u only; i.e., the graph of the function, $[u^{lower}, u^{upper}] \ni u \mapsto H(\lambda, x, u, t)$.

a local minimum to Problem HMC generates a local minimum for Problem B. Conversely, a global minimum to Problem HMC does not guarantee a global minimum for Problem B . The global minimum of H is only a part of the totality of necessary conditions for the local minimum of Problem B . In Fig. 2.13, this

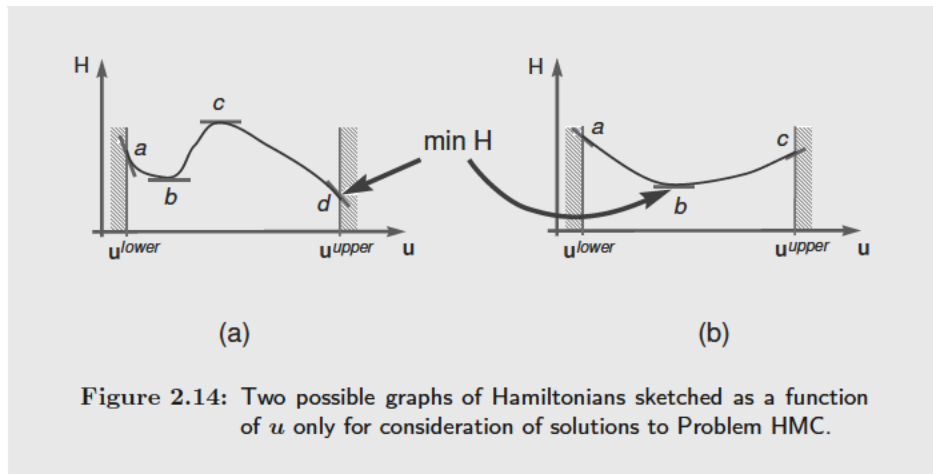
[¶]We plead guilty to this charge in our illustration in Fig. 2.10; our defense is pedagogy.

occurs at point a which is quite far from point c ; hence, *any claims of "sub optimality" for the point c would also be false.*

In the situation illustrated in Fig. 2.13, the solution to Problem HMC is the simple statement

$$u^* = u^{lower}$$

where u^{lower} is the *lower bound* of u . Obviously, the solution to Problem HMC is not always going to be at a lower bound of u ; it might occur at its *upper bound* or at some point in the interior as illustrated in Fig. 2.14. Thus,



the conditions given by Eqs. (2.33) and (2.34) are not always incorrect; they are just not inclusive of all possibilities.

One simple way to articulate the findings of the preceding discussion is to determine all points that satisfy Eq. (2.33) and compare the values of the objective function (H) at these points to those at the corners (lower and upper bounds). The point with the lowest value of H is the optimal one. This tedious process of isolation and enumeration can be substantially reduced by observing the following three properties:

1. If $u^* = u^{lower}$, then $\partial_u H > 0$. This is the situation at point a in Fig. 2.13.
2. If $u^* = u^{upper}$, then $\partial_u H < 0$. This is the situation at point d in Fig. 2.14 (a).
3. If $u^{lower} < u^* < u^{upper}$, then $\partial_u H = 0$. This is the situation at point b

in Fig. 2.14 (b).

Study Problem 2.7

Sketch figures to show the possibility of the following:

1. $\partial_{\mathbf{u}}H = \mathbf{0}$ when $\mathbf{u}^* = \mathbf{u}^{lower}$.
2. $\partial_{\mathbf{u}}H = \mathbf{0}$ when $\mathbf{u}^* = \mathbf{u}^{upper}$.

Based on these possibilities, revise the three conditions discussed in the paragraph preceding this problem.

The revised three conditions (resulting from the solution to Study Problem 2.7) can be unified through the following additional observations: When $\mathbf{u}^* = \mathbf{u}^{lower}$, we can rewrite the condition, $\partial_{\mathbf{u}}H \geq \mathbf{0}$, as

$$\partial_{\mathbf{u}}H + \boldsymbol{\mu} = \mathbf{0}, \quad \boldsymbol{\mu} \leq \mathbf{0} \quad (2.35)$$

where $\boldsymbol{\mu}$ is the “equalizer”; that is, it cancels out $\partial_{\mathbf{u}}H$ exactly. This seemingly silly trick is wonderfully powerful. To see this, observe that

$$\text{units of } \boldsymbol{\mu} = \text{units of } \frac{\partial H}{\partial \mathbf{u}} = \frac{H\text{-units}}{\mathbf{u}\text{-units}} \quad \left(= \frac{CU/TU}{\mathbf{u}\text{-units}} \right)$$

Hence, $\boldsymbol{\mu}$ is a covector that can be used to measure \mathbf{u} in terms of H -units ($= CU/TU$). Consequently, the dot product

$$\boldsymbol{\mu} \cdot \mathbf{u} \quad (H\text{-units})$$

is a legal operation that produces a scalar in exactly the same units as H . This implies we can add this quantity to H to produce a new function

$$\overline{H}(\boldsymbol{\mu}, \boldsymbol{\lambda}, \mathbf{x}, \mathbf{u}, t) := H(\boldsymbol{\lambda}, \mathbf{x}, \mathbf{u}, t) + \boldsymbol{\mu}^T \mathbf{u}$$

called the *Lagrangian of the Hamiltonian*. Now observe that

$$\partial_{\mathbf{u}}\overline{H} = \partial_{\mathbf{u}}H + \boldsymbol{\mu}$$

Comparing this equation to Eq. (2.35), we can write

$$\partial_{\mathbf{u}}\bar{H} = \mathbf{0}, \quad \mu \leq 0 \tag{2.36}$$

when $\mathbf{u}^* = \mathbf{u}^{lower}$.

Study Problem 2.8

Following the same process that led to Eq. (2.36), show that

1. $\partial_{\mathbf{u}}\bar{H} = \mathbf{0}, \quad \mu \geq 0$ if $\mathbf{u}^* = \mathbf{u}^{upper}$.
2. $\partial_{\mathbf{u}}\bar{H} = \mathbf{0}, \quad \mu = 0$ if $\mathbf{u}^{lower} \leq \mathbf{u}^* \leq \mathbf{u}^{upper}$.

Collecting the results of Study Problem 2.8 and Eq. (2.36), the three conditions generated by Study Problem 2.7 on page 117 can be formalized as

- the stationarity condition

$$\frac{\partial \bar{H}}{\partial \mathbf{u}} = \mathbf{0} \tag{2.37}$$

- and the complementarity condition

$$\mu_i \begin{cases} \leq 0 & \text{if } u_i = u_i^{lower} \\ = 0 & \text{if } u_i^{lower} < u_i < u_i^{upper} \\ \geq 0 & \text{if } u_i = u_i^{upper} \end{cases} \tag{2.38}$$

This set of two conditions is a special case of more general conditions known as the **Karush-Kuhn-Tucker (KKT) conditions**. Equations (2.37) and (2.38) are the necessary conditions for the special “box-constrained” HMC problem:

$$\text{(box-constrained HMC)} \quad \begin{cases} \text{Minimize}_{\mathbf{u}} & H(\lambda, \mathbf{x}, \mathbf{u}, t) \\ \text{Subject to} & \mathbf{u}^{lower} \leq \mathbf{u} \leq \mathbf{u}^{upper} \end{cases}$$

2.5.2 KKT Conditions for Problem HMC

Consider the case when $\mathbf{u} \mapsto H(\boldsymbol{\lambda}, \mathbf{x}, \mathbf{u}, t)$ is differentiable and the control space \mathbb{U} is given by functional inequalities

$$\mathbf{h}^L \leq \mathbf{h}(\mathbf{u}) \leq \mathbf{h}^U \quad (2.39)$$

where \mathbf{h}^L and \mathbf{h}^U are the lower and upper bounds on $\mathbf{h}(\mathbf{u})$, respectively. That is,

$$\mathbb{U} = \left\{ \mathbf{u} \in \mathbb{R}^{N_u} : \mathbf{h}^L \leq \mathbf{h}(\mathbf{u}) \leq \mathbf{h}^U \right\} \quad (2.40)$$

Then, Problem HMC is a *nonlinear programming (NLP) problem*:

$$(HMC = NLP) \quad \begin{cases} \underset{\mathbf{u}}{\text{Minimize}} & H(\boldsymbol{\lambda}, \mathbf{x}, \mathbf{u}, t) \\ \text{Subject to} & \mathbf{h}^L \leq \mathbf{h}(\mathbf{u}) \leq \mathbf{h}^U \end{cases} \quad (2.41)$$

Notation Alert: Recall that (see page xxiii) we use uppercase letters for cost functions and the corresponding lowercases for constraints. In remaining true to this style, we use the letter \mathbf{h} for naming functions and its corresponding bounds for constraints that belong to minimizing H . This is also exactly the same reason why we chose the symbol \mathbf{h} for the path function in Problem P in Section 1.4.2 on page 46. This notational convention also implies that \bar{H} goes with the pair (H, \mathbf{h}) .

According to Pontryagin's Principle, the Hamiltonian must be minimized *at each instant of time t* . To do this, we apply the KKT conditions to Problem $HMC = NLP$; that is, we define the *Lagrangian of the Hamiltonian*

$$\bar{H}(\boldsymbol{\mu}, \boldsymbol{\lambda}, \mathbf{x}, \mathbf{u}, t) := H(\boldsymbol{\lambda}, \mathbf{x}, \mathbf{u}, t) + \boldsymbol{\mu}^T \mathbf{h}(\mathbf{u}) \quad (2.42)$$

where $t \mapsto \boldsymbol{\mu} \in \mathbb{R}^{N_h}$ is a time-dependent KKT multiplier function associated with the functional constraint given by Eq. (2.39); that is $\boldsymbol{\mu}$ is a *path covector*

defined by

$$\boldsymbol{\mu} := \begin{bmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_{N_h} \end{bmatrix} \begin{array}{l} \frac{CU/TU}{h_1\text{-units}} \\ \frac{CU/TU}{h_2\text{-units}} \\ \vdots \\ \frac{CU/TU}{h_{N_h}\text{-units}} \end{array} \quad (2.43)$$

From the KKT conditions, the Lagrangian of the Hamiltonian \overline{H} must be stationary with respect to the control \mathbf{u} :

$$\frac{\partial \overline{H}}{\partial \mathbf{u}} = \frac{\partial H}{\partial \mathbf{u}} + \left(\frac{\partial \mathbf{h}}{\partial \mathbf{u}} \right)^T \boldsymbol{\mu} = \mathbf{0} \quad (2.44)$$

In addition, at each instant of time t the multiplier-constraint pair $(\boldsymbol{\mu}, \mathbf{h}(\mathbf{u}))$ must satisfy the **complementarity condition**:

$$\mu_i \begin{cases} \leq 0 & \text{if } h_i(\mathbf{u}) = h_i^L \\ = 0 & \text{if } h_i^L < h_i(\mathbf{u}) < h_i^U \\ \geq 0 & \text{if } h_i(\mathbf{u}) = h_i^U \\ \text{unrestricted} & \text{if } h_i^L = h_i^U \end{cases} \quad (2.45)$$

These complementarity conditions determine the **switching structure** of the optimal control as they can also be written as

$h_i(\mathbf{u}(t)) = h_i^L$	<i>if</i>	$\mu_i(t) \leq 0$	(2.46)
$h_i^L < h_i(\mathbf{u}(t)) < h_i^U$	<i>if</i>	$\mu_i(t) = 0$	
$h_i(\mathbf{u}(t)) = h_i^U$	<i>if</i>	$\mu_i(t) \geq 0$	
$h_i^L = h_i^U$	<i>if</i>	$\mu_i(t)$ <i>unrestricted</i>	

where the time-dependence of the relevant variables is explicitly noted.

2.5.3 Time-Varying Control Space^{||}

In many practical problems, the control space \mathbb{U} is not “static” but varies in time as well as with variations in the state \mathbf{x} . When \mathbb{U} is time-varying, we denote it as $\mathbb{U}(t)$. Pontryagin’s Principle continues to hold in all these cases with embarrassing simplicity. To illustrate this point, let

$$\mathbb{U}(t) := \left\{ \mathbf{u} \in \mathbb{R}^{N_u} : \mathbf{h}^L(t) \leq \mathbf{h}(\mathbf{u}, t) \leq \mathbf{h}^U(t) \right\} \quad (2.47)$$

That is, the inequalities that define the control space depend explicitly on time: Compare Eq. (2.47) with (2.40). Since Problem HMC requires that the Hamiltonian be minimized at each instant of time, the KKT conditions are the same as before except that the conditions must hold explicitly at each instant of time t ; thus, we have:

$$\mu_i \begin{cases} \leq 0 & \text{if } h_i(\mathbf{u}, t) = h_i^L(t) \\ = 0 & \text{if } h_i^L(t) < h_i(\mathbf{u}, t) < h_i^U(t) \\ \geq 0 & \text{if } h_i(\mathbf{u}, t) = h_i^U(t) \\ \text{unrestricted} & \text{if } h_i^L(t) = h_i^U(t) \end{cases} \quad (2.48)$$

2.5.4 Solving Problem HMC

When \mathbb{U} is a continuous set (see Fig. 2.2 on page 87), Problem HMC is an NLP. If \mathbb{U} is discrete, Problem HMC is a discrete/integer programming problem. Hence, in general, Problem HMC is a nonlinear mixed-variable (i.e., continuous-discrete) programming problem. Barring an extremely limited number of special cases, there are no closed-form solutions to NLPs or integer programming problems. *Not surprisingly, in many practical applications, Problem HMC cannot be solved “by hand” or “analytically.”* Fortunately, this is not a major deterrent for generating usable solutions to practical optimal control problems. To understand this critical point, it is necessary to appreciate the agonies, nuances, caveats and joys of producing usable solutions.

^{||}This section may be skipped without loss in continuity.

The Agonies of Analytical Solutions

An analytical or “closed form” solution is generally defined as a quantity that can be written in terms of elementary operations (e.g., $+$, $-$, \dots) over elementary functions: sines, cosines, exponentials, etc. A solution is generally considered to be “approximate” or, worse, “numerical” (apparently, of lower quality) if it is expressed in terms of a large number (such as a hundred or more) of elementary operations over elementary functions; for example, a truncated finite sum of an infinite series. The problem with these (mis)perceptions is that the computations of elementary functions are eventually approximate and maybe even done “surreptitiously” by a truncated series expansion (for example, consider how a computer computes $\sin(12.3)$, $e^{4.56}$, π etc.). In other words, when the claims of an analytical solution are dissected, it turns out it is an unfortunate myth that must be rejected with extreme prejudice. There is no grand universal mathematical theorem which guarantees that all solutions to all mathematical problems are expressible in terms of a handful of elementary functions. Within the context of mathematically accurate expectations, Problem HMC is indeed solvable in a vast number of situations.

A solution to Problem HMC can be stated compactly using the argmin notation

$$\mathbf{u}^* = \arg \min_{\mathbf{u} \in \mathbb{U}} H(\boldsymbol{\lambda}, \mathbf{x}, \mathbf{u}, t) \quad (2.49)$$

which is shorthand for the statement that the argument (i.e., \mathbf{u}) of the minimum of H with respect to \mathbf{u} is equal to \mathbf{u}^* . This seemingly high-brow notation is useful and clever at the same time. It is useful in the sense that it avoids the use of the generic symbol \mathbf{g} , used previously (see Eq. (2.21) on page 105) and replaces it by a more evocative notation given by Eq. (2.49). See also the *Tech Talk* box later on page 196. Equation (2.49) is quite clever because, through the argmin notation, every optimization problem looks solved in an exact analytical form!

The Phantom of Global Optimization

A *necessary* condition for solving Problem B is a globally optimal solution to Problem HMC. Consider the situation illustrated in Fig. 2.15. The globally optimal solution is point d . The necessary conditions for this globally optimal solution are Eqs. (2.33) and (2.34). They are also the necessary conditions for

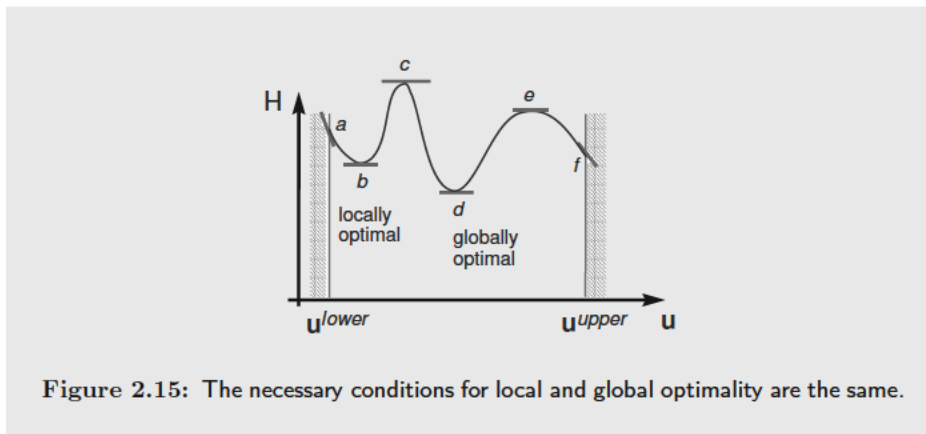


Figure 2.15: The necessary conditions for local and global optimality are the same.

local optimality; hence they are satisfied at point b , as well. In other words, short of enumeration, there is no way to detect a globally optimal solution.

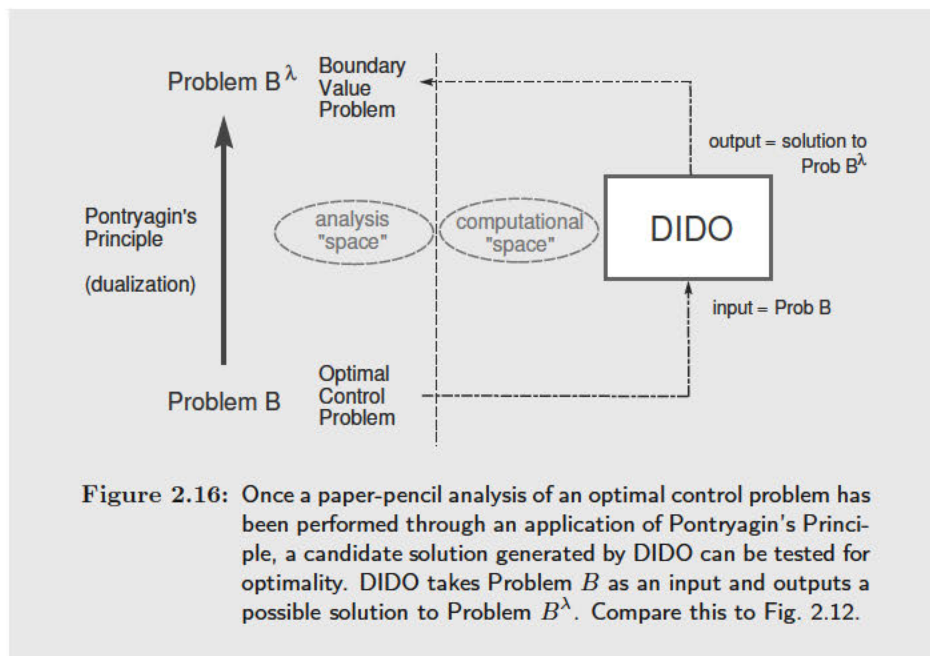
Even if a globally optimal solution to Problem HMC is found, it is only part of the necessary conditions for Problem B . That is, a globally optimal solution to Problem HMC does not even guarantee a locally optimal solution to Problem B .

When U is parameterized by functional inequalities, the necessary conditions for Problem HMC are given by the KKT conditions. The KKT conditions do not solve Problem HMC. Consequently, replacing Problem HMC with its KKT conditions in the collection of necessary conditions for Problem B produces a gap — and possibly a large one — between the true necessary conditions and its replacement. *The analysis of this gap must be part of the analysis of a candidate optimal solution.*

The Notion of Reigning Optimal Solutions

The preceding discussions appear to suggest that it is extremely hard to produce (globally) optimal solutions to Problem B . It indeed is if the requirement is a mathematical proof of global optimality; however, there is a perceptible divide between what is provably optimal and what can be easily generated through a software package like DIDO. Given a Problem B , a software like DIDO generates a system trajectory ($t \mapsto (x, u)$) and the associated covector functions ($t \mapsto \lambda$ and ν). That is, DIDO takes Problem B as an input, in nearly the same format as defined in Section 2.1, and outputs a *candidate*

solution to Problem B^λ ; see Fig. 2.16. This candidate solution can then be tested for optimality by applying Pontryagin's Principle "by hand." This is the "analysis space" in Fig. 2.16. *Recognizing the nuances discussed in the preceding paragraphs is critical to carrying out these tests.* These tests help a more careful analysis of many practical problems which, in turn, support the generation of better inputs for DIDO, and possibly alternative problem formulations as discussed in Chapter 1. This is part of the *problem of problems* introduced in Chapter 1.



In the end, a practical test for optimality is not the claim of a globally optimal solution; rather, it is whether a new solution to an old problem is better than the old solution (and by how much) or whether the problem posed is itself new so that any (feasible) solution is desirable to the alternative (of no solution). Short of a mathematical proof to the contrary, a reigning optimal solution can stake the claim of global optimality to a given problem until a better solution is found!

2.5.5 A Tale of Two, Maybe Three Hamiltonians

A solution to Problem HMC generates the function:

$$(\boldsymbol{\lambda}, \mathbf{x}, t) \xrightarrow{g} \mathbf{u}^* = \arg \min_{\mathbf{u} \in \mathbb{U}} H(\boldsymbol{\lambda}, \mathbf{x}, \mathbf{u}, t) \quad (2.50)$$

When \mathbf{u}^* is substituted for \mathbf{u} in $H(\boldsymbol{\lambda}, \mathbf{x}, \mathbf{u}, t)$, it yields the *minimized Hamiltonian*:

$$\mathcal{H}(\boldsymbol{\lambda}, \mathbf{x}, t) := \min_{\mathbf{u} \in \mathbb{U}} H(\boldsymbol{\lambda}, \mathbf{x}, \mathbf{u}, t) \equiv H(\boldsymbol{\lambda}, \mathbf{x}, \mathbf{u}, t)|_{\mathbf{u}=g(\boldsymbol{\lambda}, \mathbf{x}, t)} \quad (2.51)$$

Illustrating the Concept: The Minimized Hamiltonian

The minimized Hamiltonian for Brac:1 is obtained by substituting Eq. (2.20) in (2.8):

$$\begin{aligned} \mathcal{H}(\boldsymbol{\lambda}, \mathbf{x}) &:= \lambda_x v \sin \left(\tan^{-1} \left(\frac{\lambda_x v}{\lambda_y v + \lambda_v g} \right) \right) \\ &\quad + \lambda_y v \cos \left(\tan^{-1} \left(\frac{\lambda_x v}{\lambda_y v + \lambda_v g} \right) \right) \\ &\quad + \lambda_v g \cos \left(\tan^{-1} \left(\frac{\lambda_x v}{\lambda_y v + \lambda_v g} \right) \right) \end{aligned}$$

Clearly, \mathcal{H} is **nonlinear in $\boldsymbol{\lambda}$** (while H is always linear).

At this point, we do not know if the left-hand side of Eq. (2.51) is still a Hamiltonian, that is, if it satisfies Eq. (2.17). It turns out that it indeed does if we assume the function g in Eq. (2.50) is differentiable and \mathbf{u}^* is interior to \mathbb{U} .

**Study Problem 2.9**

Show that \mathcal{H} is a Hamiltonian; that is, under appropriate assumptions, it satisfies the equations

$$\begin{aligned} \dot{\mathbf{x}} &= \frac{\partial \mathcal{H}}{\partial \boldsymbol{\lambda}} \\ -\dot{\boldsymbol{\lambda}} &= \frac{\partial \mathcal{H}}{\partial \mathbf{x}} \end{aligned} \quad (2.52)$$

Catalog all the assumptions needed for this proof.

The function, \mathcal{H} , is called the *lower Hamiltonian*. The *upper Hamiltonian* is obtained if H were to be maximized instead of minimized. In their original work, Pontryagin et al maximized the Hamiltonian to be consistent with the archetypal or *Hamilton's Hamiltonian*. This original Hamiltonian is obtained in the classical calculus of variations through a process known as the Legendre transform. Thanks to the simplifications and insights offered by Pontryagin's Principle, the only difference between the two Hamiltonians is a sign change; see also Section 1.4.4 on page 55. In this book, we use the "minimum version" of Pontryagin's Principle.

In general, \mathcal{H} is nonlinear in λ and non-differentiable; however, it evolves according to a very simple equation known as the *Hamiltonian Evolution Equation*:

$$\boxed{\frac{d\mathcal{H}}{dt} = \frac{\partial H}{\partial t}} \tag{2.53}$$

Illustrating the Concept: The Hamiltonian Evolution Equation

From Eq. (2.8) it is clear that the (control) Hamiltonian for Problem Brac:1 does not depend explicitly with time; hence, we have:

$$\frac{\partial H}{\partial t} = 0$$

From the Hamiltonian evolution equation, this implies:

$$\mathcal{H}(\lambda(t), \mathbf{x}(t)) = \text{constant} \quad (\text{w.r.t. time})$$

More explicitly, we can write:

$$\lambda_x(t)v(t) \sin \theta^*[\@t] + \lambda_y(t)v(t) \cos \theta^*[\@t] + \lambda_v(t)g \cos \theta^*[\@t] = \text{constant} \tag{2.54}$$

where $\theta^*[\@t]$ is the shorthand notation for

$$\theta^*[\@t] \equiv \theta^*(\mathbf{x}(t), \lambda(t)) = \tan^{-1} \left(\frac{\lambda_x(t)v(t)}{\lambda_y(t)v(t) + \lambda_v(t)g} \right)$$


The Hamiltonian evolution equation is an *integral of motion*: It has significant practical value in solving both academic and industrial optimal control problems.

Study Problem 2.10

In Problem B, introduce an $(N_x + 1)$ th state variable

$$\dot{x}_{N_x+1} = 1 \quad (2.55)$$

with initial condition, $x_{N_x+1}(t_0) = t_0$. Then, the data in the modified problem become time-invariant (i.e., not an explicit function of time) with $\mathbf{x} \in \mathbb{R}^{N_x+1}$.

1. Show that $\lambda_{N_x+1}(t) = -\mathcal{H}(\boldsymbol{\lambda}(t), \mathbf{x}(t), t)$ where \mathcal{H} is the lower Hamiltonian to the original Problem B.
2.  In the original Problem B, it is sufficient for \mathbf{f} and F to be merely measurable with respect to t . In view of this, criticize the suggestion that time-invariant and time-varying problem data are equivalent under the introduction of Eq. (2.55).

2.6 A Cheat Sheet for Pontryagin's Principle

The totality of necessary conditions for Problem B can be summarized in the form of a major result:

Theorem 2.1 (Pontryagin's Principle) *Given an optimal solution to Problem B, there exists an absolutely continuous covector function $\boldsymbol{\lambda}(\cdot)$ and a covector $\boldsymbol{\nu}$ that satisfy*

- the three Hamiltonian conditions:
 1. Hamiltonian minimization condition,
 2. Hamiltonian value condition,
 3. Hamiltonian evolution equation
- the adjoint equations, and
- the transversality condition.

Remark 2.1.1 *Pontryagin's Principle has not solved Problem B: it simply states the necessary conditions that a candidate optimal solution must satisfy.*

Moreover, one of these necessary conditions, that is, the Hamiltonian Minimization Condition

$$(HMC) \quad \begin{cases} \text{Minimize}_{\mathbf{u}} & H(\boldsymbol{\lambda}, \mathbf{x}, \mathbf{u}, t) \\ \text{Subject to} & \mathbf{u} \in \mathbb{U} \end{cases} \quad (2.56)$$

is posed as a problem in itself, whose solution can be symbolically stated as:

$$(\boldsymbol{\lambda}, \mathbf{x}, t) \mapsto \mathbf{u}^* = \arg \min_{\mathbf{u} \in \mathbb{U}} H(\boldsymbol{\lambda}, \mathbf{x}, \mathbf{u}, t) \quad (2.57)$$

Once the function $\mathbf{u}^*(\cdot)$ is obtained in the form of Eq.(2.57) (i.e., by solving Problem HMC), then an extremal may be obtained by solving the Hamiltonian system of $2N_x$ ordinary differential equations

$$\dot{\mathbf{x}} = \frac{\partial H}{\partial \boldsymbol{\lambda}} \quad (2.58a)$$

$$-\dot{\boldsymbol{\lambda}} = \frac{\partial H}{\partial \mathbf{x}} \quad (2.58b)$$

with boundary conditions

$$\mathbf{x}(t_0) = \mathbf{x}^0 \quad (2.59a)$$

$$\mathbf{e}(\mathbf{x}(t_f), t_f) = \mathbf{0} \quad (2.59b)$$

$$\boldsymbol{\lambda}(t_f) = \frac{\partial \bar{E}}{\partial \mathbf{x}_f} \quad (2.59c)$$

that must be satisfied at the start of the clock at the given time

$$t_0 = t^0 \quad (2.60)$$

and at the "optimal" stopping time

$$\mathcal{H}[@t_f] = -\frac{\partial \bar{E}}{\partial t_f} \quad (2.61)$$

where $\mathcal{H}[@t_f]$ is shorthand for for the value of \mathcal{H} at t_f :

$$\mathcal{H}[@t_f] \equiv \mathcal{H}(\boldsymbol{\lambda}(t_f), \mathbf{x}(t_f), t_f)$$

Remark 2.1.2 *The Hamiltonian evolution equation*

$$\frac{d\mathcal{H}}{dt} = \frac{\partial H}{\partial t} \tag{2.62}$$

is an integral of motion. It serves an extremely important role in the verification and validation of the computed solution, either by means of analytical equations for simple problems or in providing an equation to check the optimality of a numerical solution.

At first, all these equations might seem a little overwhelming, particularly to a beginner. After some practice (i.e., after studying Chapters 3 and 4), they will seem relatively straightforward and even logical. In the meantime, a beginning student might find it useful to have some quick ways of remembering all the necessary conditions. To this end, we offer the following mnemonics:

- **3HAT**: This stands for the three Hamiltonian conditions, the adjoint equation and the transversality condition summarized in Theorem 2.1.
- **HAMVET**: Pontryagin's HAMVET (with apologies to Shakespeare) comes from the following systematic steps that must be carried out for any given problem to develop its necessary conditions for optimality:

(a) Construct the H amiltonian.	H
(b) Develop the A djoint equations.	A
(c) M inimize the Hamiltonian.	M
(d) Evaluate the Hamiltonian V alue condition.	V
(e) Integrate the Hamiltonian E volution equation.	E
(f) Formulate the T ransversality conditions.	T
- **Engineers Value MATH**: This is simply anagramming *HAMVET* to *EV MATH*. Interestingly, it turns out that *MATH* forms the core of Pontryagin's Principle.

Remark 2.1.3  The control Hamiltonian given in Eq. (2.7) is in its “normal form” and not quite complete; the “complete” Hamiltonian for Problem *B* is actually given by

$$H(\boldsymbol{\lambda}, \mathbf{x}, \mathbf{u}, t) = \nu_0 F(\mathbf{x}, \mathbf{u}, t) + \boldsymbol{\lambda}^T \mathbf{f}(\mathbf{x}, \mathbf{u}, t) \tag{2.63}$$

where ν_0 is called the *cost multiplier*. Additional necessary conditions include the *constancy and nonnegativity* of the cost-multiplier function: $t \mapsto \nu_0 = \text{constant} \geq 0$. This condition implies two cases: either $\nu_0 = 0$ or $\nu_0 \neq 0$.

1. $\nu_0 = 0$

If $\nu_0 = 0$, then the Hamiltonian and all the equations derived from it are independent of the cost! This situation is possible but is deemed “abnormal.”

2. $\nu_0 \neq 0$

If $\nu_0 \neq 0$, then it simply scales the Hamiltonian by a positive constant; hence, we can divide H by ν_0 and analyze the “scaled” Hamiltonian. This scaling is called the *Normality Condition* and is equivalent to arbitrarily setting $\nu_0 = 1$ in Eq. (2.63).

Clearly, we must have nontrivial multipliers, $(\nu_0, \boldsymbol{\lambda}) \neq (0, \mathbf{0})$; that is, we must not have all covectors equal to zero. This *Nontriviality Condition* can also be stated as

$$\nu_0 + \|\boldsymbol{\lambda}(\cdot)\|_{L^\infty} > 0 \quad (2.64)$$

Pontryagin's Principle as stated in Theorem 2.1 is for the normal case. It can be restated for all cases by writing the Hamiltonian in the form given by Eq.(2.63) and including the additional nuances of constancy, nonnegativity and nontriviality as part of the necessary conditions.



Geek Speak: A lot of technical details are missing in our statement of Pontryagin's Principle as codified in Theorem 2.1. For instance, the standard assumptions for the function space for the state and control trajectories are $\mathbf{x}(\cdot) \in W^{1,1}$ and $\mathbf{u}(\cdot) \in L^\infty$. Because we assumed \mathbf{f} to be Lipschitz in \mathbf{x} (actually, C^1 in \mathbf{x}), we can write $(\mathbf{x}(\cdot), \mathbf{u}(\cdot)) \in W^{1,\infty} \times L^\infty$.

Study Problem 2.11

Construct the complete set of differential and boundary conditions obtained for any one of the other alternative formulations of the Brachistochrone problem; e.g., Problem Brac : 2 developed in Chapter 1. Does this alternative formulation generate any new information? Discuss.

(Hint: Yes! This is why different mathematical transcriptions of the same word problem are extremely useful in better understanding the structure of optimal solutions.)

Study Problem 2.12

Using the extremal controls obtained in Study Problem 2.11, write down the expressions for \mathcal{H} , the lower Hamiltonian, for each of these example problems. Show that \mathcal{H} is not necessarily linear in λ or even a differentiable function. Is $\dot{\mathbf{x}} = \partial_{\lambda}\mathcal{H}$? Is $\dot{\lambda} = -\partial_x\mathcal{H}$? Is it possible for \mathcal{H} to be multivalued?

2.7  Pontryagin's Principle for Problem P

Problem P is defined in Section 1.4.2 on page 47, and is repeated here for quick reference:

$$\begin{array}{l}
 \left. \begin{array}{l}
 \mathbb{X}_{search} = \mathbb{R}^{N_x} \\
 \mathbf{x} = (x_1, \dots, x_{N_x})
 \end{array} \right\} \\
 \left. \begin{array}{l}
 \mathbb{U}_{search} = \mathbb{R}^{N_u} \\
 \mathbf{u} = (u_1, \dots, u_{N_u})
 \end{array} \right\} \text{(preamble)} \\
 \\
 \text{problem } \left\{ \begin{array}{l}
 \text{Minimize } J[\mathbf{x}(\cdot), \mathbf{u}(\cdot), t_0, t_f] := \\
 \qquad E(\mathbf{x}_0, \mathbf{x}_f, t_0, t_f) + \int_{t_0}^{t_f} F(\mathbf{x}(t), \mathbf{u}(t), t) \\
 \text{Subject to } \qquad \qquad \qquad \dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t) \\
 \qquad \qquad \qquad \qquad \qquad \qquad \mathbf{e}^L \leq \mathbf{e}(\mathbf{x}_0, \mathbf{x}_f, t_0, t_f) \leq \mathbf{e}^U \\
 \qquad \qquad \qquad \qquad \qquad \qquad \mathbf{h}^L \leq \mathbf{h}(\mathbf{x}(t), \mathbf{u}(t), t) \leq \mathbf{h}^U
 \end{array} \right\} \begin{array}{l}
 \text{(cost)} \\
 \text{(dynamics)} \\
 \text{(events)} \\
 \text{(path)}
 \end{array}
 \end{array}$$

The pair (F, \mathbf{f}) is the same in both problems P and B; hence, the definition

of the control or Pontryagin's Hamiltonian remains unchanged:

$$H(\boldsymbol{\lambda}, \mathbf{x}, \mathbf{u}, t) := F(\mathbf{x}, \mathbf{u}, t) + \boldsymbol{\lambda}^T \mathbf{f}(\mathbf{x}, \mathbf{u}, t)$$

The endpoint Lagrangian depends only on the pair (E, \mathbf{e}) ; hence, \bar{E} is modified only in terms of its functional dependencies:

$$\bar{E}(\boldsymbol{\nu}, \mathbf{x}_0, \mathbf{x}_f, t_0, t_f) := E(\mathbf{x}_0, \mathbf{x}_f, t_0, t_f) + \boldsymbol{\nu}^T \mathbf{e}(\mathbf{x}_0, \mathbf{x}_f, t_0, t_f)$$

Because a complementarity condition goes hand-in-hand with inequality-type constraints, we define $\boldsymbol{\nu} \dagger \mathbf{e}$ as shorthand for the conditions

$$\nu_i \left\{ \begin{array}{ll} \leq 0 & \text{if } e_i(\mathbf{x}_0, \mathbf{x}_f, t_0, t_f) = e_i^L \\ = 0 & \text{if } e_i^L < e_i(\mathbf{x}_0, \mathbf{x}_f, t_0, t_f) < e_i^U \\ \geq 0 & \text{if } e_i(\mathbf{x}_0, \mathbf{x}_f, t_0, t_f) = e_i^U \\ \text{unrestricted} & \text{if } e_i^L = e_i^U \end{array} \right. \quad (2.65)$$

with $\boldsymbol{\mu} \dagger \mathbf{h}$ defined similarly.

The control space in Problem P is state-dependent and parameterized by functional inequalities:

$$\mathbb{U}(\mathbf{x}, t) := \left\{ \mathbf{u} \in \mathbb{R}^{N_u} : \mathbf{h}^L \leq \mathbf{h}(\mathbf{x}, \mathbf{u}, t) \leq \mathbf{h}^U \right\}$$

The symbol $\mathbb{U}(\mathbf{x}, t)$ means that \mathbb{U} is not a constant (as in Problem B ; see Fig. 1.26 on page 43) but that it depends jointly on \mathbf{x} and t . Thus $\mathbb{U}(\mathbf{x}, t)$ is also a map from (\mathbf{x}, t) to a set or a *set-valued map*.^{*} Despite this infusion of practically inspired mathematical complication, Problem HMC has the same functional form as before:

$$(HMC) \quad \left\{ \begin{array}{l} \underset{\mathbf{u}}{\text{Minimize}} \quad H(\boldsymbol{\lambda}, \mathbf{x}, \mathbf{u}, t) \\ \text{Subject to} \quad \mathbf{h}^L \leq \mathbf{h}(\mathbf{x}, \mathbf{u}, t) \leq \mathbf{h}^U \end{array} \right. \quad (2.66)$$

^{*}This simple observation illustrates why set-valued analysis has direct practical applications.

Thus, the *Lagrangian of the Hamiltonian* is given by

$$\overline{H}(\boldsymbol{\mu}, \boldsymbol{\lambda}, \mathbf{x}, \mathbf{u}, t) := H(\boldsymbol{\lambda}, \mathbf{x}, \mathbf{u}, t) + \boldsymbol{\mu}^T \mathbf{h}(\mathbf{x}, \mathbf{u}, t)$$

and the KKT conditions for Problem HMC can be written compactly as

$$\partial_{\mathbf{u}} \overline{H} = \mathbf{0} \quad \text{and} \quad \boldsymbol{\mu} \dagger \mathbf{h}$$

where $\boldsymbol{\mu} \dagger \mathbf{h}$, as noted previously, is defined similar to Eq. (2.65).

The lower Hamiltonian is also functionally the same as before

$$\mathcal{H}(\boldsymbol{\lambda}, \mathbf{x}, t) := \min_{\mathbf{u} \in \mathbb{U}(\mathbf{x}, t)} H(\boldsymbol{\lambda}, \mathbf{x}, \mathbf{u}, t)$$

with the caveat that \mathbb{U} depends on \mathbf{x} and t .

One of the major modifications to Pontryagin's original principle is that the adjoint equations, while similar to that of Problem B, are now based on \overline{H} and not H :

$$-\dot{\boldsymbol{\lambda}} = \frac{\partial \overline{H}}{\partial \mathbf{x}}$$

From the complementarity condition $\boldsymbol{\mu} \dagger \mathbf{h}$, it follows that the adjoint covector has the same co-dynamics as Problem B when the state trajectory is "inside" the path constraint (if necessary, refer to Fig. 1.28 on page 46). Obviously, when the state trajectory rides the path constraint, the adjoint equation has additional terms. See Study Problem 2.14 at the end of this section on page 136.

The terminal transversality condition remains the same as before (functionally)

$$\boldsymbol{\lambda}(t_f) = \frac{\partial \overline{E}}{\partial \mathbf{x}_f}$$

while we "gain" an *initial transversality condition*:

$$\boldsymbol{\lambda}(t_0) = -\frac{\partial \overline{E}}{\partial \mathbf{x}_0}$$

Likewise, we have *two Hamiltonian value conditions*

$$\mathcal{H}[@t_0] = \frac{\partial \overline{E}}{\partial t_0} \quad \text{and} \quad \mathcal{H}[@t_f] = -\frac{\partial \overline{E}}{\partial t_f}$$

that correspond to optimal start and stop times. Thus, the *HAMVET* mnemonic

and other ones discussed on page 129 still apply, albeit with a few different details.

The collection of all unknowns resulting from Pontryagin's Principle for Problem P are as follows:

1. The system trajectory, $t \mapsto (\mathbf{x}, \mathbf{u}) \in \mathbb{R}^{N_x} \times \mathbb{R}^{N_u}$;
2. The adjoint covector function, $t \mapsto \boldsymbol{\lambda} \in \mathbb{R}^{N_x}$;
3. The path covector function, $t \mapsto \boldsymbol{\mu} \in \mathbb{R}^{N_h}$;
4. The endpoint covector, $\boldsymbol{\nu} \in \mathbb{R}^{N_e}$; and
5. The initial and final times, $t_0 \in \mathbb{R}$ and $t_f \in \mathbb{R}$.

These unknowns must satisfy a collection of differential and algebraic constraints, with the latter structured in terms of conditional inequalities. Collecting all these equations and inequalities we can constitute Problem P^λ as follows:

$$(P^\lambda) \left\{ \begin{array}{ll}
 \dot{\mathbf{x}}(t) - \partial_{\lambda} \overline{H}(\boldsymbol{\mu}(t), \boldsymbol{\lambda}(t), \mathbf{x}(t), \mathbf{u}(t), t) = \mathbf{0} & \text{(state eqns)} \\
 \dot{\boldsymbol{\lambda}}(t) + \partial_x \overline{H}(\boldsymbol{\mu}(t), \boldsymbol{\lambda}(t), \mathbf{x}(t), \mathbf{u}(t), t) = \mathbf{0} & \text{(costate eqns)} \\
 \mathbf{h}^L \leq \mathbf{h}(\mathbf{x}(t), \mathbf{u}(t), t) \leq \mathbf{h}^U & \text{(path condition)} \\
 \partial_u \overline{H}(\boldsymbol{\mu}(t), \boldsymbol{\lambda}(t), \mathbf{x}(t), \mathbf{u}(t), t) = \mathbf{0} & \text{(Hamiltonian} \\
 & \boldsymbol{\mu} \dagger \mathbf{h} \quad \text{Minimization)} \\
 \mathbf{e}^L \leq \mathbf{e}(\mathbf{x}_0, \mathbf{x}_f, t_0, t_f) \leq \mathbf{e}^U & \text{(endpoint eqns)} \\
 \boldsymbol{\lambda}(t_0) + \partial_{x_0} \overline{E}(\boldsymbol{\nu}, \mathbf{x}_0, \mathbf{x}_f, t_0, t_f) = \mathbf{0} & \text{(initial and} \\
 \boldsymbol{\lambda}(t_f) - \partial_{x_f} \overline{E}(\boldsymbol{\nu}, \mathbf{x}_0, \mathbf{x}_f, t_0, t_f) = \mathbf{0} & \text{final transversality} \\
 & \boldsymbol{\nu} \dagger \mathbf{e} \quad \text{conditions)} \\
 \mathcal{H}[\@t_0] - \partial_{t_0} \overline{E}(\boldsymbol{\nu}, \mathbf{x}_0, \mathbf{x}_f, t_0, t_f) = 0 & \text{(Hamiltonian} \\
 \mathcal{H}[\@t_f] + \partial_{t_f} \overline{E}(\boldsymbol{\nu}, \mathbf{x}_0, \mathbf{x}_f, t_0, t_f) = 0 & \text{value conditons)}
 \end{array} \right.$$

Problem P^λ is not a “simple” BVP (compare with Problem B^λ) because the inequalities and complementarity conditions generate switches, jumps, phases and so on in one or more variables at one or more (unknown) interior points. It will be apparent later (see Section 2.9 on page 150) that solving even a simple BVP is cursed with extreme sensitivity because of the symplectic structure (see page 114) of the Hamiltonian system (state-costate pair). This is why the simpler path to finding a solution to Problem P^λ is through the covector mapping principle discussed later in Section 2.9.2 on page 157.

Study Problem 2.13

Show that the “additional” necessary conditions for Problem B are given by

1. $\lambda(t_0) = -\nu_{x_0}$
2. $\mathcal{H}[@t_0] = \nu_{t_0}$

where (ν_{x_0}, ν_{t_0}) is the covector pair associated with the initial conditions $\mathbf{x}_0 = \mathbf{x}^0$ and $t_0 = t^0$, respectively.

**A Caveat or Two About Problem P^λ**

In Pontryagin et al[75], the development and formulation of Problem P^λ is different from our presentation: It does not contain the mixed state-control constraints, and the formulation of the necessary conditions is based on differentiating \mathbf{h} . The addition of a path constraint (to Problem B), first as a pure state constraint (i.e., \mathbf{h} is a function of \mathbf{x} only) and later as a mixed state-control constraint (as in Problem P) has generated a substantial amount of literature since Pontryagin's ground-breaking work. See [24] and the references contained therein. New technical difficulties arise in extending the necessary conditions for Problem P . As a result, there are a number of different “versions” of “Pontryagin's” Principle. So, the question becomes which one should we choose?

We have chosen the version presented on page 134 because it aligns with the covector mapping principle implemented in DIDO. Refer to the discussions associated with Fig. 2.28 on page 164. According to Dmitruk[26, 27], this version of Pontryagin's Principle was developed by Milyutin and Dubovitskii and published in Russian over the period 1963–1981. Starting with his doctoral thesis[20], Clarke[21, 22, 24] and others (see [99] and the references contained

therein) have shown that *nonsmooth analysis* is the “simpler” and unified framework for a general theory of optimal control. From this starting point, Problem P^λ is also derived in [22, 24] with additional abstractions.

We have ignored a large number of technical assumptions that go along with Problem P^λ , the most dominant of these being in the form of constraint qualifications. In other words, problem formulation, once again, takes center stage! Because Problem P is widely used for practical applications, it is important to acquire a working knowledge of Problem P^λ . These elements are described next; however, to better understand these details, we strongly recommend an attempt to solve Study Problem 2.14.

Study Problem 2.14

Recall that the control space \mathbb{U} in Problem Brac:3 defined on page 28 was state dependent. Pontryagin's Principle for Problem B is inapplicable to this problem formulation. Using the results from this section, develop the necessary conditions for Problem Brac:3. Do these conditions generate any new insights?

Computational Tip: The obstacle avoidance problem and other RTOC implementations require moving state constraints in the problem formulation. See, for example, page 71 and [17, 40, 59]. As a result, it is every easy to inadvertently formulate problems that violate many of the assumptions needed in the statement of Problem P^λ . In such situations, great care must be exercised in how the problem is formulated, particularly in a DIDO environment, because the coded problem may either produce incorrect results, or may generate correct results but only after a long computational time[84]. Reformulating a problem is critical to producing correct solutions. See, for example, Section 4.1.4 on page 253.

A Deeper Dive on The Impact of Path Constraints

The costate and path covector trajectories $t \mapsto (\boldsymbol{\lambda}, \boldsymbol{\mu})$ have characteristically different shapes in the presence of active state and mixed constraints. This can

be quickly observed by expanding the costate equations for Problem P as

$$\begin{aligned} -\dot{\lambda} &= \partial_x \bar{H} \\ &= \partial_x H + \left(\frac{\partial \mathbf{h}}{\partial \mathbf{x}} \right)^T \boldsymbol{\mu} \quad (\text{with } \boldsymbol{\mu} \dagger \mathbf{h}) \end{aligned} \quad (2.67)$$

If $\mu_i = 0$, then the corresponding costate trajectory has “unconstrained” co-dynamics, but over regions where $\mu_i \neq 0$, the related adjoint equation acquires additional terms that reshape the (costate) trajectory.

Illustrating the Concept: A Constrained Brachistochrone Problem

Consider Brac:1 with an added path constraint:

$$y \leq ax + b \quad (2.68)$$

With $h(\mathbf{x}) := y - ax$, $h^L = -\infty$, $h^U = b$, we have

$$-\dot{\lambda}_x(t) = -a\mu(t) \quad (2.69a)$$

$$-\dot{\lambda}_y(t) = \mu(t) \quad (2.69b)$$

$$-\dot{\lambda}_v(t) = \lambda_x \sin \theta + \lambda_y \cos \theta \quad (2.69c)$$

where $\mu(t)$ is complementary to $h(\mathbf{x}(t))$ according to

$$\mu(t) \begin{cases} = 0 & \text{if } y(t) - ax(t) < b \\ > 0 & \text{if } y(t) - ax(t) = b \end{cases} \quad (2.70)$$

Compare this with Eq. (2.16) on page 101. Equations (2.69) and (2.70) are illustrated in Fig. 2.17.

Study Problem 2.15

Consider Brac:1 with the addition of Eq. (2.68).

1. Prove that λ_x must be non-decreasing while λ_y must be non-increasing as apparent from Fig. 2.17.
2. Prove that $t \mapsto \lambda_v$ is affine over the constraint boundary $y = ax + b$.
3. Is the trajectory $t \mapsto \mu$ continuous?

4. Are the costate trajectories $t \mapsto (\lambda_x, \lambda_y, \lambda_v)$ continuous? Continuously differentiable?

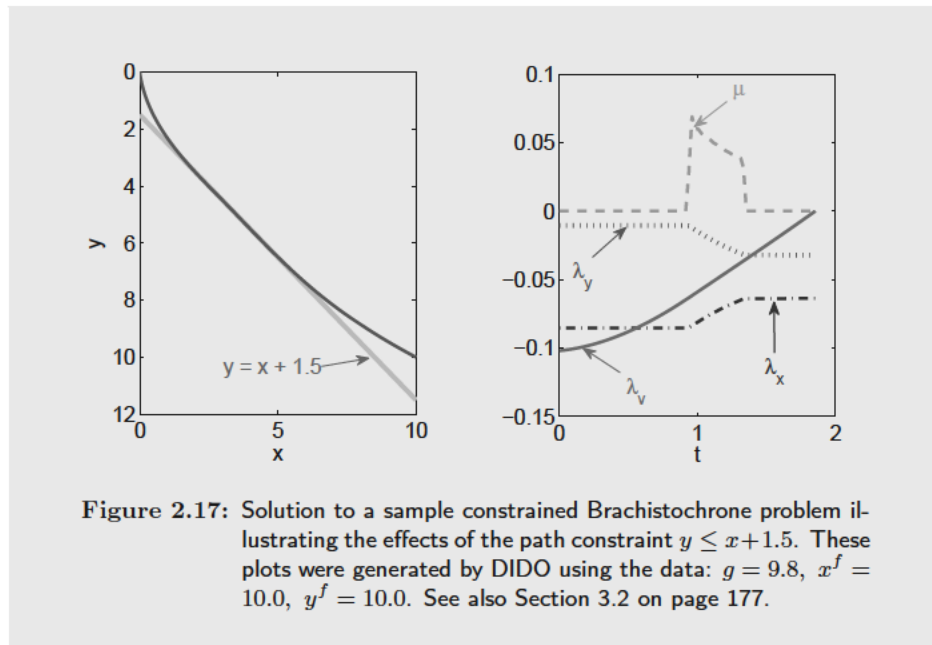


Figure 2.17: Solution to a sample constrained Brachistochrone problem illustrating the effects of the path constraint $y \leq x + 1.5$. These plots were generated by DIDO using the data: $g = 9.8$, $x^f = 10.0$, $y^f = 10.0$. See also Section 3.2 on page 177.

A quick examination of Problem P^λ reveals that there are no dynamics associated with the covector μ ; that is, there is no $\dot{\mu}$ anywhere; hence, μ is inertia-less. This implies that the trajectory $t \mapsto \mu$ may have discontinuities — a point quite apparent from Fig. 2.17. Because we assumed h was differentiable with respect to x , a discontinuous path-covector trajectory $t \mapsto \mu$ has the same effect on the costate trajectory as a discontinuous control. See Eq. (2.67) and observe that μ is like a co-control on the co-dynamics.

Now, recall that Pontryagin's Principle for Problem B states the existence of an *absolutely continuous* costate trajectory $t \mapsto \lambda$. Absolute continuity is simply a mathematician's way of saying “differentiable . . . but not the way you think.” See $\lambda_x(t)$ and $\lambda_y(t)$ in Fig. 2.17, particularly over the two points where $\mu(t)$ jumps. The discussions would end here if we were assured that $t \mapsto \mu$ behaves “no worse” than a control trajectory; that is, piecewise continuous (technically,

L^∞). Unfortunately, this is not the case:

$t \mapsto \mu$ may have impulses.

An impulsive μ implies (see Eq. (2.67)) that the costate trajectory may have jumps; hence, $t \mapsto \lambda$ is no longer assured to be absolutely continuous. This phenomenon is not some mathematical minutia that can be pushed under the rug; it is quite real and quite practical, as illustrated by the following physical problem due to Breakwell[16].

Illustrating the Concept: Impulsive Path Covector Trajectory

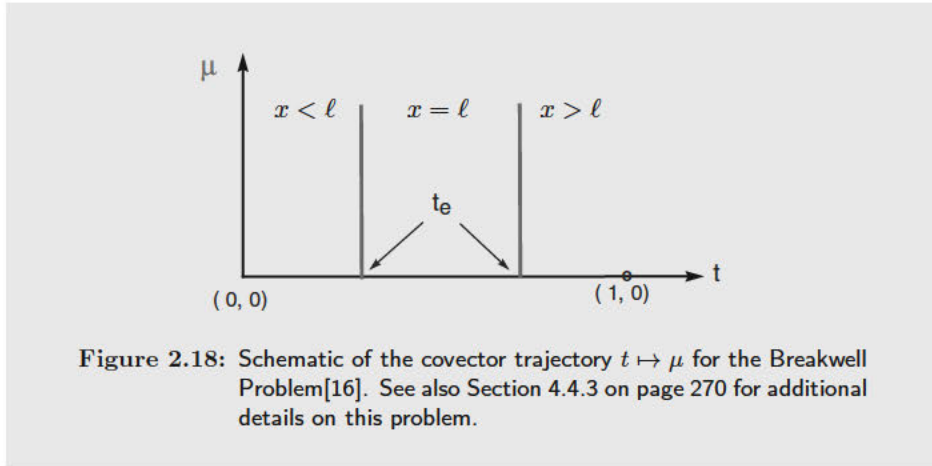
The Breakwell problem is a simple double integrator $\ddot{x} = u$ with a path constraint $x \leq \ell$ and quadratic cost $\int_0^1 u^2(t)dt$; see Section 4.4.3 on page 270 for details. As outlined in §4.4.3, we have

$$-\dot{\lambda}_x(t) = \mu(t) \begin{cases} = 0 & \text{if } x(t) < \ell \\ = 0 & \text{if } x(t) = \ell \end{cases} \quad (2.71)$$

If we hastily conclude that $\lambda_x(t)$ must be a constant, then we get no solution for the boundary condition $x(0) = x(1) = 0$, $\dot{x}(0) = -\dot{x}(1) = 1$, an erroneous result that is quite apparent from elementary analysis. This "paradox" is easily solvable if we "allow" $\mu(t)$ to have precise and **finite impulses** such that

$$\int_{t_e - \epsilon}^{t_e + \epsilon} \mu(t) dt = \eta \quad (> 0 \text{ as } 0 < \epsilon \rightarrow 0) \quad (2.72)$$

where t_e is an entry or exit time; that is, t_e is the time when $x(t)$ enters or leaves the boundary $x(t) = \ell$; see Fig. 2.18. With this expansion of our mathematical horizons, Eq. (2.71) now tells us that $t \mapsto \lambda_x$ is **piecewise constant** with allowable jumps at t_e . As a result of the sign of $\mu > 0$ (or $\eta > 0$) these jumps must be downward for $\lambda_x(t)$.



A quick approach for detecting what may happen with finite impulsive elements in $\mu(t)$ is to integrate Eq. (2.67) over an epsilon neighborhood of a candidate jump point

$$\int_{t_e-\epsilon}^{t_e+\epsilon} -\dot{\lambda} dt = \int_{t_e-\epsilon}^{t_e+\epsilon} \left[\partial_x H + \left(\frac{\partial h}{\partial x} \right)^T \mu \right] dt$$

and note that we can write (with some minor abuse of ϵ):

$$\lambda(t_e^-) - \lambda(t_e^+) = \left(\frac{\partial h}{\partial x} \right)^T_{x=x_e} \underbrace{\int_{t_e-\epsilon}^{t_e+\epsilon} \mu(t) dt}_{\eta} \tag{2.73a}$$

$$= \left(\frac{\partial h}{\partial x} \right)^T_{x=x_e} \eta \quad (\text{with } \eta \dagger h[@t_e]) \tag{2.73b}$$

Equation (2.73b) can be used as follows: If h does not depend on a particular state variable, then its corresponding costate is continuous. See for example Eq. (2.69c) on page 137. This is a sufficient condition; it is not necessary. That is, simply because $\partial_x h \neq 0$ does not mean there will be jumps. See, for example, Eqs. (2.69a) and (2.69b) and Fig. 2.17. This is because $t \mapsto \mu$ may not have finite impulses.

Equation (2.73b) is discussed in some textbooks and papers as a *jump condition* but without the insight of the connection between η and μ . The comple-

mentarity of ν with \mathbf{h} at t_e (denoted by $\boldsymbol{\eta} \dagger \mathbf{h}[\@t_e]$) is inherited from $\boldsymbol{\mu}$ because the integration is over a positive measure.



Tech Talk: Because of the need to incorporate atomic measures, Eq. (2.73a) can be written more “cleanly” as a Lebesgue-Stieltjes integral:

$$\int \partial_x \mathbf{h}[\@t] \boldsymbol{\mu}(t) dt = \int \partial_x \mathbf{h}[\@t] d\tilde{\boldsymbol{\mu}}(t)$$

Such an approach is commonly found in the mathematical literature with extensive use of Radon measures. Another approach is to write $\boldsymbol{\mu}(t)$ as the sum of two functions

$$\boldsymbol{\mu} = \boldsymbol{\mu}_{L^\infty} + \boldsymbol{\mu}_\delta$$

where $\boldsymbol{\mu}_\delta$ is of atomic measure. We have used this approach implicitly in characterizing $\boldsymbol{\mu}_\delta$ as a collection of finite impulses.

The Hamiltonian Evolution Equation for Problem P^λ

Recall that the Hamiltonian evolution equation was an integral of motion for Problem B. For Problem P, this evolution equation is given analogously by

$$\frac{d\mathcal{H}}{dt} = \frac{\partial \bar{H}}{\partial t} \tag{2.74}$$

Similar to Problem B, Eq. (2.74) serves an extremely important role in the *verification and validation* of a computed solution. Similar to Eq. (2.67), we can write Eq. (2.74) as

$$\frac{d\mathcal{H}}{dt} = \partial_t H + \left(\frac{\partial \mathbf{h}}{\partial t} \right)^T \boldsymbol{\mu} \quad (\text{with } \boldsymbol{\mu} \dagger \mathbf{h})$$

Because $t \mapsto \boldsymbol{\mu}$ may have finite impulses, \mathcal{H} may jump similarly to $\boldsymbol{\lambda}(t)$. This jump condition may be derived similarly to the procedure used for the costate trajectory and written as

$$\mathcal{H}[\@t_e^+] - \mathcal{H}[\@t_e^-] = \left(\frac{\partial \mathbf{h}}{\partial t} \right)_{t=t_e}^T \boldsymbol{\eta} \quad (\text{with } \boldsymbol{\eta} \dagger \mathbf{h}[\@t_e]) \tag{2.75}$$

Illustrating the Concept: A Constrained Brachistochrone Problem

Because $t \mapsto \mu$ had no impulses in the constrained Brachistochrone problem (see Fig. 2.17 on page 138) we have $\eta = 0$; hence, the lower Hamiltonian is continuous. Furthermore, because $\partial_t H = 0$, we must have

$$\mathcal{H}(\lambda(t), x(t)) = -1 \quad (\forall t)$$

That $\mathcal{H}[\cdot]$ must not jump and be a constant of value -1 even in the presence of the state constraint is a powerful statement for a $V \& V$ of a solution. This aspect of gaining confidence in a computed solution is illustrated in Fig. 2.19.

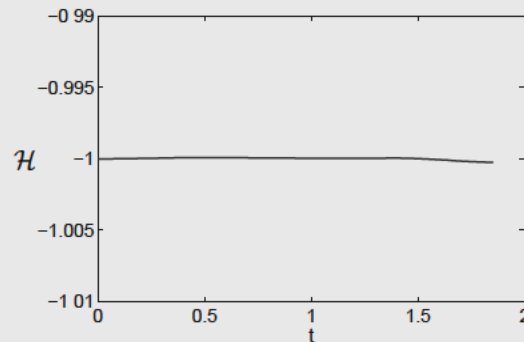


Figure 2.19: The lower Hamiltonian $t \mapsto \mathcal{H}(\lambda(t), x(t))$ generated by DIDO for the sample constrained Brachistochrone problem discussed on page 138.

How to Use Problem P^λ : Some Dos and Don'ts

The caveats and warnings that go along with Problem P^λ are a further indication of the maxim that it is best to use the necessary conditions as necessary conditions, that is, as checks on the optimality of a candidate solution instead of a sanction for a technique to solve problems. Doing so would generate a “hard” problem *by choice*.

At the other end of the spectrum is the use of a software package like DIDO.

Because DIDO makes problem-solving “easy,” it is essential to exercise great restraint in solving problems without analysis; that is, generating solutions without analyzing (not solving!) Problem P^λ . An analysis of Problem P (and hence P^λ) includes a cognizance of the structure of $t \mapsto \boldsymbol{\mu}$. Recall that the Breakwell problem warned us that $t \mapsto \boldsymbol{\mu}$ may have finite impulses, while the constrained Brachistochrone problem only gave us discontinuities in $\boldsymbol{\mu}(t)$. It is critical to understand Problem P^λ in advance of coding (see Section 4.1.4 on page 253) or modifying a working code before declaring a problem to be “hard.”

In a practical and computational setting, a useful rule of thumb for the detection of jumps is the notion of the *order of a state constraint*. If \mathbf{h} depends on \mathbf{x} only, and possibly t , then its time derivative is given by

$$\begin{aligned}\frac{d\mathbf{h}(\mathbf{x}, t)}{dt} &= \partial_{\mathbf{x}}\mathbf{h}(\mathbf{x}, t)^T \dot{\mathbf{x}} + \partial_t\mathbf{h}(\mathbf{x}, t) \\ &= \partial_{\mathbf{x}}\mathbf{h}(\mathbf{x}, t)^T \mathbf{f}(\mathbf{x}, \mathbf{u}, t) + \partial_t\mathbf{h}(\mathbf{x}, t)\end{aligned}\quad (2.76)$$

If Eq. (2.76) depends explicitly on \mathbf{u} , then \mathbf{h} is said to be of order one. For example, Eq. (2.68) is of order one because

$$\begin{aligned}\dot{h}(\mathbf{x}) &= \dot{y} - a\dot{x} \\ &= v \cos \theta - av \sin \theta \quad (\theta \text{ is control})\end{aligned}$$

When a constraint is of order one, it is typical for $t \mapsto \boldsymbol{\mu}$ to not have impulses; see Fig. 2.17.

It is quite possible that Eq. (2.76) may contain no control terms. For example, in the Breakwell problem we have

$$\begin{aligned}\dot{h}(\mathbf{x}) &= \dot{x} \\ &= v\end{aligned}$$

In such a situation, we can differentiate \mathbf{h} again. If \mathbf{u} appears explicitly for the second derivative, then \mathbf{h} is called a path constraint of order two. Performing this exercise for the Breakwell problem we have

$$\begin{aligned}\ddot{h}(\mathbf{x}) &= \dot{v} \\ &= u\end{aligned}$$

Hence, the constraint $h(\mathbf{x}) = x \leq \ell$ is of second order. It is typical for second and higher-order constraints to have impulsive path covector trajectories; see Fig. 2.18.

Note from these discussions that as the problem becomes more practical, Pontryagin's Principle provides many more conditions for analysis.

Computational Tip: In one of its “dual” operating modes, DIDO uses a Galerkin approximation for the covector trajectories. This mode “smooths out” impulses and discontinuities in dual spaces to achieve computational efficiency. For higher *dual accuracy*, other operating modes may be used but at the price of reduced computational speed. Because accuracy of the control trajectory is the last word in optimal control, an “efficient” computational strategy in the presence of path constraints is to use the Galerkin option for the dual operating mode. Under this option, the controls converge more rapidly than the dual variables; hence, the covector outputs of DIDO must be used as approximations while performing V & V. This mathematical insight on the mechanics of pseudospectral optimal control theory is due to Q. Gong.

2.8 Avoiding Common Errors # 3

Pontryagin's Principle is a statement of the *necessary* conditions for Problem B . It asserts the *existence* of certain covector functions. The origin of many common errors can be traced to overlooking these fundamentals.

1. Necessary, Not Sufficient

A solution to an optimal control problem (B) is also a solution to the boundary value problem (B^λ). The converse is not necessarily true:

$$\begin{array}{ccc} \text{Solution to Problem } B & \Rightarrow & \text{Solution to Problem } B^\lambda \\ & \Leftarrow & \end{array}$$

See Section 4.1.3 on page 252 for a demonstration of the insufficiency of Pontryagin's Principle based on a simple problem. It is very tempting to use the backward implication and claim optimality, for instance, by solving the BVP. Fortunately, the forward implication is much easier to use; see Section 2.9.

2. Bane of Non-unique Existence

Pontryagin's Principle simply asserts the existence of covector functions; it does not imply their uniqueness. The non-uniqueness of covectors is not part of some fine print. It occurs frequently and often goes unrecognized as shown by the following problem (inspired by the unpublished results of M. Karpenko and Q. Gong). See also Section 4.9 on page 306.

Study Problem 2.16

The kinematics of a rigid body can be expressed in terms of Euler parameters or "quaternions" as

$$\begin{aligned}\dot{\mathbf{q}} &= \frac{1}{2} (q_4 \boldsymbol{\omega} + \mathbf{q} \times \boldsymbol{\omega}) \\ \dot{q}_4 &= -\frac{1}{2} \boldsymbol{\omega} \cdot \mathbf{q}\end{aligned}$$

where $\mathbf{q} \in \mathbb{R}^3$ is the Euler vector and q_4 is the scalar (component of the quaternion). Assuming no running cost, show that the adjoint covector functions obey the same dynamics as the quaternions; i.e.,

$$\begin{aligned}\dot{\boldsymbol{\lambda}}_q &= \frac{1}{2} (\lambda_4 \boldsymbol{\omega} + \boldsymbol{\lambda}_q \times \boldsymbol{\omega}) \\ \dot{\lambda}_4 &= -\frac{1}{2} \boldsymbol{\omega} \cdot \boldsymbol{\lambda}_q\end{aligned}$$

Consequently, $\boldsymbol{\lambda}_q \cdot \boldsymbol{\lambda}_q + \lambda_4^2 = \text{constant}$. A useful calculus for this proof is the rule

$$\frac{\partial}{\partial \vec{x}} \left(\vec{a} \cdot (\vec{b} \times \vec{x}) \right) = \vec{a} \times \vec{b}$$

Discuss the conditions that generate an infinite number of costate trajectories. (Hint: Consider the transversality conditions). See also Section 4.9 on page 306.

3. Analyze, Not Solve

An application of Pontryagin's Principle to Problem B generates a different, presumably simpler problem (B^λ); see Fig. 2.12 on page 112. Embedded inside

Problem B^λ is another problem (HMC). At first, it seems reasonable to assume that the inner problem (HMC) needs to be solved to analyze the outer problem (B^λ). This is not necessarily true because Problem HMC can be analyzed via its KKT conditions.

The KKT conditions are necessary conditions; they do not solve Problem HMC. Fortunately, there is no need to “solve” Problem HMC. Its inclusion in Theorem 2.1 facilitates an analysis of the the inner and outer problems as a single unit. Such an analysis is facilitated through an application of DIDO as illustrated in Fig. 2.16 on page 124.

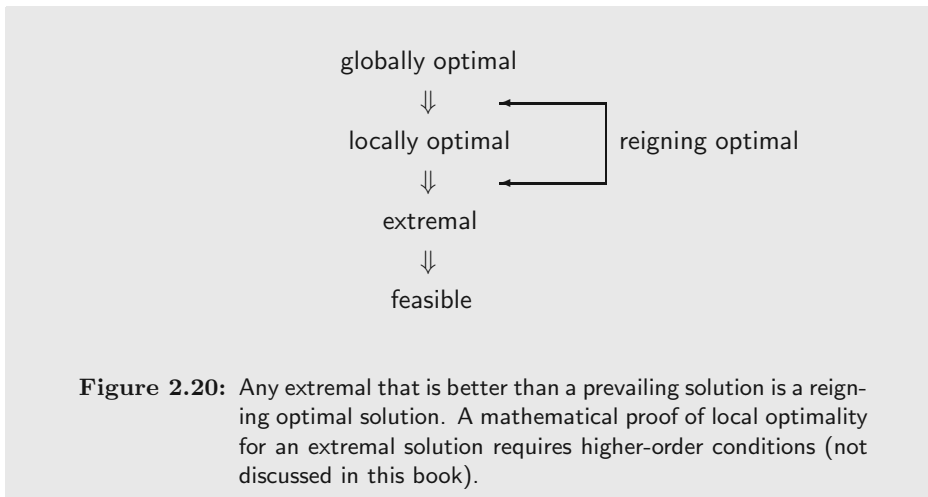
4. Sufficient, But Unnecessary

Based on the misconception that it is necessary to produce an analytic solution to Problem HMC, many beginners and some practitioners fall into the trap of formulating the wrong problem (e.g., “augmenting” unnecessary cost functions or performing ruthless linearizations). Recall (see page 122) that problem solving does not imply analytic solutions. As we gradually move away from the abacus to a computer, it becomes increasingly clear that is better to solve the correct problem “approximately” than solve the wrong problem “exactly.” This is why it is more important to formulate the correct problem than perform unnecessary simplifications based on the presumed supremacy of analytic solutions. Analytic solutions may indeed enter in the formulation of the *problem of problems* as noted in Chapter 1, but in their proper context, for instance, as part of the development of the right problem, or as a means to verify and validate (V&V) a computer code.

5. Chasing Phantoms

Pontryagin's Principle is a formulation of the necessary conditions for optimality; consequently, it is an analysis tool, not a problem-solving tool. It can be correctly used as a necessary condition, that is, to test the optimality of a candidate solution regardless of how it is obtained. In Chapter 3, we take the liberty of using it as a problem-solving tool because it can indeed be used as such on a very small class of academic-strength problems. In industrial applications, Pontryagin's Principle is used at its fundamental level: as a necessary condition. That is, if a candidate solution fails Pontryagin's test, it is not an optimal solution, but an optimal solution must satisfy the necessary conditions.

Solutions that satisfy Pontryagin's Principle are called *extremals*. By definition, the extremals are feasible but they are not necessarily optimal. Not even locally optimal. These facts are not a deterrent to generating extremely good and usable solutions to many problems provided sufficient care and caveats are used. In many instances, an extremal solution might be fully satisfactory if its cost is sufficiently low. For example, if the cost of an extremal solution is lower than the best prevailing solution, it can be crowned the reigning optimal solution until a better solution is found. See remarks on page 123 in Section 2.5.4.



Sometimes, the claim or demand of global optimality is uninformed; however, there are many application problems where it is possible to estimate a lower bound for $J[\mathbf{x}(\cdot), \mathbf{u}(\cdot), t_f]$ and analyze the optimality of an extremal solution. That is, suppose we can find an a priori number, \mathcal{J}^{lower} , such that

$$J[\mathbf{x}(\cdot), \mathbf{u}(\cdot), t_f] \geq \mathcal{J}^{lower}$$

for all feasible decision variables, $(\mathbf{x}(\cdot), \mathbf{u}(\cdot), t_f)$. Now let $(\mathbf{x}^\#(\cdot), \mathbf{u}^\#(\cdot), t_f^\#)$ be an extremal solution (e.g., one that is successfully computed by DIDO). Then, based on the difference,

$$J[\mathbf{x}^\#(\cdot), \mathbf{u}^\#(\cdot), t_f^\#] - \mathcal{J}^{lower} \quad (\geq 0)$$

any number of practical conclusions can be made:

1. Is the difference small enough[†] that it is not economical to find the globally optimal solution?
2. Is the difference so large that there might be an error in the estimation of \mathcal{J}^{lower} ?
3. Is the difference so large that it is worth exploring finding another extremal solution closer to \mathcal{J}^{lower} ?
4. Is there a mistake in the problem formulation itself? See Chapter 1, particularly Fig. 1.50 on page 84.

As an illustrative example of these considerations, let us examine a problem from astronautics. In space maneuvering, nearly all problems are fundamentally driven by propellant usage because of the extremely high dollar-cost of launch. Thus, when the cost functional is propellant, it is easy to write

$$\mathcal{J}^{lower} = 0$$

In Bedrossian's celebrated zero propellant maneuvers (ZPMs) implemented on-board the International Space Station[5], the extremal solutions satisfy

$$J[\mathbf{x}^\#(\cdot), \mathbf{u}^\#(\cdot), t_f^\#] = 0$$

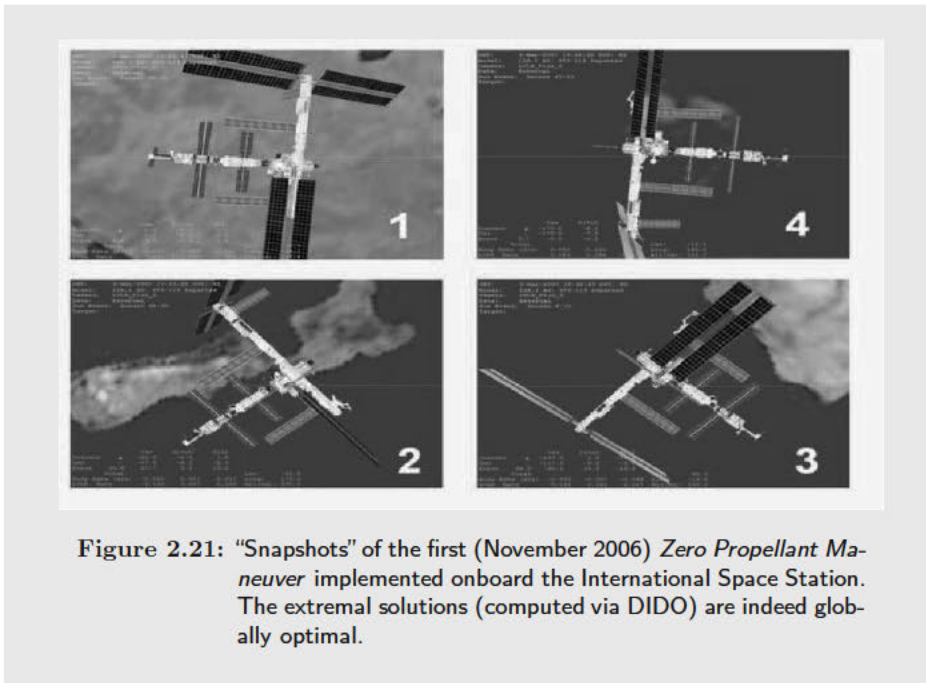
Hence, the triple $(\mathbf{x}^\#(\cdot), \mathbf{u}^\#(\cdot), t_f^\#)$ is indeed globally optimal. See Fig. 2.21.

In many situations, (provable) global optimality is not necessarily the most important criterion. A user may be fully satisfied with a feasible solution if its cost is lower than some targeted value \mathcal{J}^{target} . Then, any non-zero difference

$$\mathcal{J}^{target} - J[\mathbf{x}^\#(\cdot), \mathbf{u}^\#(\cdot), t_f^\#] \quad (\geq 0)$$

is considered a bonus, and any extremal solution that bests the other is the reigning optimal solution. For instance, in 2012 (August), the "optimal" propellant maneuver[7] replaced the ZPM on the Space Station even though it expended propellant because the time to complete the maneuver was less than one orbital period, a "soft" requirement that was highly valued by NASA.

[†]In an industrial environment, it is common to use phrases like "within single digits" for small or "double-digit improvements in performance" for large.



Thus, the practice of optimal control is not necessarily the quest for globally or even locally optimal solutions, rather, it is a quest for usable solutions that are candidates for the reigning optimal. Pontryagin's Principle helps eliminate spurious solutions through mathematical tests.

5. Feasible = Optimal

There are times when some users/customers imply that they are more interested in finding feasible solutions than optimal solutions. In this case, all feasible solutions are globally optimal! In DIDO, this can be done by simply turning off the cost function.

2.9 Kalman and the Curse of Sensitivity

In 1964, R. Kalman presented a paper in Yorktown Heights, N.Y., at an IBM symposium on control[46].[‡] Based on the challenges he and others faced in computing optimal controls, Kalman argued that the problem was “intrinsically difficult”; hence, he suggested, there must be some underlying fundamental mathematical principles that could form the basis for a “theory of difficulty.” Roughly speaking, Kalman contended that any computational method based on numerically integrating (simulating) differential equations was doomed, and that the only escape from this eventual disaster was some “newfangled algebra” that could “algebraize” the dynamics. Based on the somewhat contentious discussions that ensued at the end of his presentation (as recorded in [46]) and the subsequent lack of citations to his paper, it is apparent that Kalman’s ideas were not quite well received. Inspired by Bellman’s *curse of dimensionality*, we refer to Kalman’s concept of intrinsic difficulty as the *curse of sensitivity*. The escape clause on the curse of sensitivity (i.e., the newfangled algebra that Kalman predicted) is the basis for *pseudospectral (PS) optimal control theory*[87]. The curse and its escape are illustrated and introduced in this section. The details are dispersed over several papers cited in [87].

2.9.1 An Introduction to the Curse of Sensitivity

Consider the following problem:

$$\left\{ \begin{array}{l} x \in \mathbb{R} \quad u \in \mathbb{R} \\ \text{Minimize} \quad J[x(\cdot), u(\cdot)] = \frac{1}{2} \int_{t_0}^{t_f} u^2(t) dt \\ \text{Subject to} \quad \dot{x} = ax + bu \\ \quad \quad \quad x_0 = x^0 \\ \quad \quad \quad t_0 = 0 \\ \quad \quad \quad x_f = x^f \\ \quad \quad \quad t_f = t^f \end{array} \right. \quad (2.77)$$

[‡]This symposium was attended by a veritable Who’s Who in applied mathematics and control: One of the presenters at this conference was none other than Pontryagin himself!

where a and b are given real numbers. An application of Pontryagin's Principle generates (show this!) the following pair of state-costate differential equations:

$$\begin{aligned} \dot{x} &= ax - b^2\lambda && \text{(primal)} \\ \dot{\lambda} &= -a\lambda && \text{(dual)} \end{aligned} \quad (2.78)$$

It can be shown (do it!) that the exact solution to the state trajectory is given by

$$x(t) = x^0 e^{at} + \frac{b^2 \lambda_0}{2a} (e^{-at} - e^{at}) \quad (2.79)$$

where $\lambda_0 \equiv \lambda(0)$ is the unknown quantity. Evaluating Eq. (2.79) at $t = t_f$, and rearranging the terms, we get:

$$\lambda_0 = \frac{2a}{b^2} \left(\frac{x_f - x^0 e^{at_f}}{e^{-at_f} - e^{at_f}} \right) \quad (2.80)$$

Substituting the boundary conditions, $t_f = t^f$ and $x(t_f) = x^f$ in Eq. (2.80), we can compute the exact value of $\lambda(0)$ as

$$\lambda^0 := \frac{2a}{b^2} \left(\frac{x^f - x^0 e^{at^f}}{e^{-at^f} - e^{at^f}} \right) \quad (2.81)$$

If λ^0 is substituted for λ_0 in Eq. (2.79), we will have obtained the exact solution for the state trajectory, $t \mapsto x$; see Fig. 2.22.

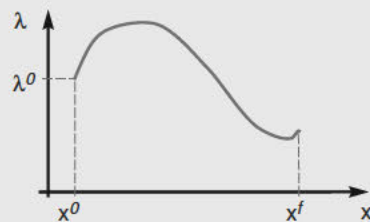


Figure 2.22: Using the data (x^f, t^f, x^0) , the exact value of the missing initial condition (λ^0) is computable from Eq. (2.81). When this value is used as the initial condition for the primal-dual system, it achieves the final time condition $x(t^f) = x^f$ (precisely).

Study Problem 2.17

Show that:

1. The Pontryagin extremal control to the problem defined by Eq. (2.77) is given by

$$u = -\lambda b$$

2. The lower Hamiltonian to Problem 2.77 is given by

$$\mathcal{H}(\lambda, x) := ax\lambda - \frac{b^2\lambda^2}{2}$$

3. The partials $\partial_\lambda \mathcal{H}$ and $\partial_x \mathcal{H}$ produce the right-hand sides of the primal-dual system given by Eq. (2.78). Will such a relationship hold for all problems? (Hint: Examine this for Brac:1.)
4. A particular solution to the primal system (defined in Eq. (2.78)) is given by

$$x_{\text{particular}}(t) = \frac{b^2\lambda_0}{2a}e^{-at}$$

In a general problem, the key formula, $(x^f, t^f, x^0) \mapsto \lambda^0$, as given by Eq. (2.81) for Problem 2.77, is not known. This is because of the absence of closed-form solutions to generic nonlinear differential equations; see also page 122 for a discussion of analytical solutions. Despite this “setback,” it is easy to generate the map $\lambda_0 \mapsto x_f$ through numerical integration (e.g., a standard Runge-Kutta procedure; see Fig. 2.23). As a means to understand the consequences of solving BVPs through a sequence of initial value problems (IVPs), let us assume that we had to solve Eq. (2.78) iteratively. This BVP is given by

$$\begin{aligned} \dot{x} &= ax - b^2\lambda & t_0 &= 0, & x(t_0) &= x^0 \\ \dot{\lambda} &= -a\lambda & t_f &= t^f, & x(t_f) &= x^f \end{aligned} \quad (2.82)$$

The intuitive, and prima facie logical-sounding **shooting algorithm**, would proceed as follows: Guess a value for λ_0 , say λ^{Guess} . Solve the *initial value*

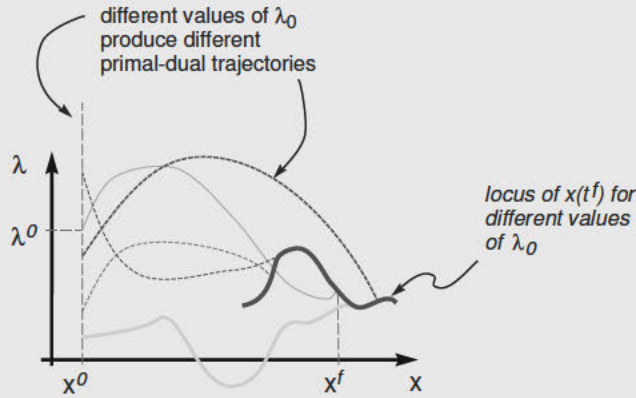


Figure 2.23: When λ_0 is not equal to its exact value (λ^0), it generates $x(t^f)$, whose locus can be determined through numerical propagation.

problem

$$\begin{aligned} \dot{x} &= ax - b^2\lambda & t_0 &= 0, \quad x(t_0) = x^0, \quad \lambda(t_0) = \lambda^{Guess} \\ \dot{\lambda} &= -a\lambda & t_f &= t^f \end{aligned} \quad (2.83)$$

all the way up to the given value of $t_f = t^f$. This process produces a number $x(t^f)$ that we can use to compare to its specified value x^f ; see Fig. 2.23. Assuming we did not correctly guess the exact value of λ_0 (i.e., the value given by Eq. (2.81)) we will have

$$x(t^f) \neq x^f$$

The shooting algorithm is based on the idea that we can use some measure of the difference, $(x(t^f) - x^f)$, to generate a new guess for λ_0 and repeat the process until $x(t^f) = x^f$. Because the process is iterative,[§] it is unlikely that we will obtain $x(t^f) = x^f$ exactly; hence, we simply require that

$$x(t_f) \approx x^f \quad (2.84)$$

[§]This iterative process is a nonlinear programming problem; e.g., the objective may be to minimize error norms subject to tolerance constraints. Consequently, it is erroneous to imply that a shooting method is not based on nonlinear programming techniques as is sometimes indicated in the literature.

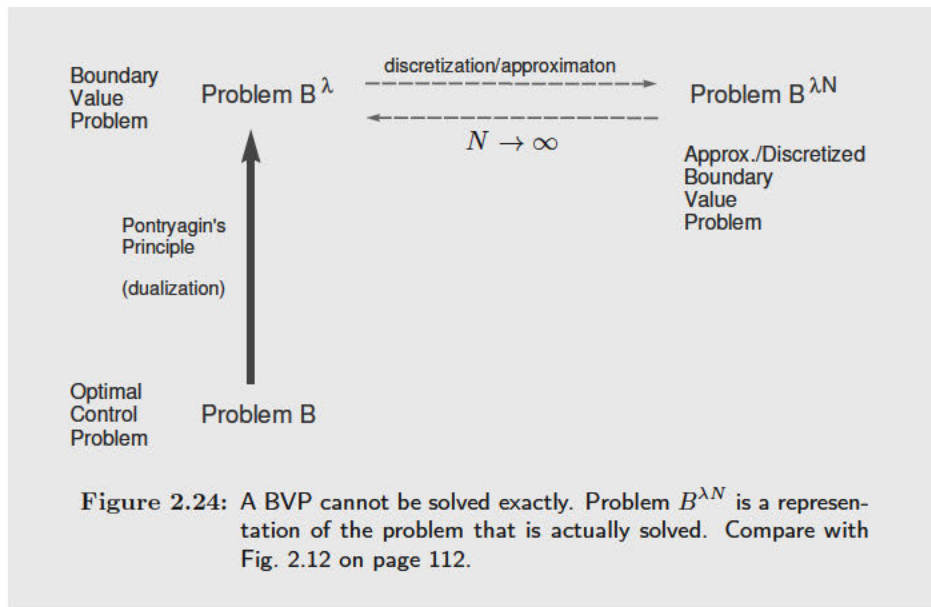
within some pre-specified tolerance. Furthermore, because $x(t^f)$ is itself computed through numerical integration, its value will also be approximate; hence, we can write Eq. (2.84) a little more precisely as

$$x^N(t^f) \approx x^f \tag{2.85}$$

where $x^N(t^f)$ is itself an approximation to $x(t^f)$ such that

$$\lim_{N \rightarrow \infty} x^N(t^f) \rightarrow x(t^f)$$

The number N is an abstract representation of an approximation process: It may be construed as denoting the number of integration steps (i.e., the inverse of the step size) or similar other notion of discretization. This concept is codified in Fig. 2.24 where $B^{\lambda N}$ represents the computational problem that is actually solved. Because we expect convergence as $N \rightarrow \infty$, an argument can be made that the better the accuracy of the numerical integration, the higher the accuracy of the solution.



This procedure is so intuitive and simple that it smells like it ought to work seamlessly. *It doesn't!* Here's why: *Assume we have perfect integration;* then,

subtracting Eq. (2.81) from (2.80) and rearranging the terms, we get

$$x_f - x^f = \frac{b^2}{2a} (e^{-at_f} - e^{at_f}) (\lambda_0 - \lambda^0) \quad (2.86)$$

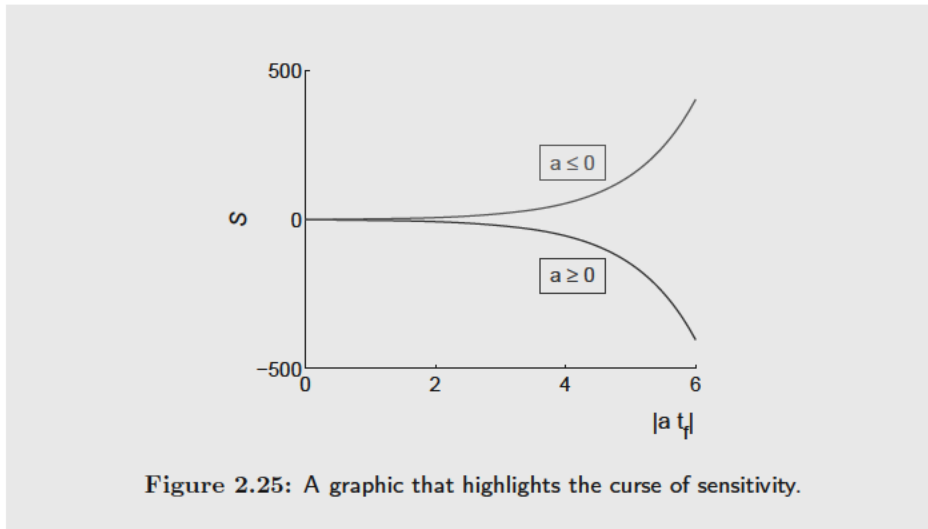
This equation shows that for different values of $\lambda_0 \neq \lambda^0$, we get different values of $x_f \neq x^f$; hence, for different values of $\lambda^{Guess} (= \lambda_0)$, the computed value of $x(t_f) = x_f$ is not equal to the specified number x^f . The difference between x_f and x^f varies as

$$S := (e^{-at_f} - e^{at_f})$$

The absolute value of S is exponentially large and is independent of the sign of a . The greater the quantity

$$|a|t_f \quad (2.87)$$

the larger the number $|x_f - x^f|$, even when $\lambda_0 \equiv \lambda^{Guess}$ is near λ^0 . In other words, the value of $x(t_f)$ is quite sensitive to the value of λ_0 . See Fig. 2.25. This sensitivity relationship can also be inferred from Eq. (2.79) by evaluating



$x(t)$ at $t = t_f$ and taking the partial derivative

$$\frac{\partial x_f}{\partial \lambda_0} = \frac{b^2}{2a} (e^{-at_f} - e^{at_f})$$

This leads us to the following observations:

1. Small changes in λ_0 produce exponentially large changes in x_f .
2. The sensitivity of the mapping $\lambda_0 \mapsto x_f$ is independent of the sign of a ; hence, it is a myth that “stable” systems produce stable BVPs.

The last statement about stability comes from the observation that the quantity a is an eigenvalue of the dynamical system. The eigenvalues of the primal-dual system given by Eq. (2.78) are a and $-a$; see Fig. 2.26. This fact follows quite readily from the eigenvalues of the system matrix,



$$\begin{bmatrix} a & -b^2 \\ 0 & -a \end{bmatrix}$$

Hence, from Eq. (2.87) we conclude that the product of the absolute value of the eigenvalue of the system and the horizon play a central role in quantifying the sensitivity of the system: the greater this product the higher the sensitivity.



Figure 2.26: Eigenvalues of the primal-dual system for the problem given by Eq. (2.77) on page 150.

Study Problem 2.18



1. If the dynamical system in an optimal control problem has linear time-varying coefficients, discuss its impact on the curse of sensitivity. Is it possible to generate a formula similar to Eq. (2.87) for such a system?
2.  If the dynamical system in an optimal control problem is nonlinear (e.g., given by a generic equation, $\dot{x} = f(x, u, t)$), what is the meaning of the eigenvalue of the system? Can a connection be made to the Lipschitz constant? See [92].
3.  Does an infinite horizon problem have infinite sensitivity?

 **Study Problem 2.19**

Explain how the non-uniqueness of the costates (refer back to Section 2.8) impacts a procedure for solving the BVP (Problem B^λ).

  **Study Problem 2.20**

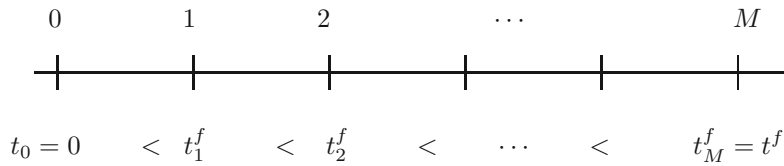
Critique the argument that geometric or symplectic integration can provide an escape from the curse of sensitivity.

2.9.2   **Escaping the Curse of Sensitivity**

The shortcomings of the shooting method have been observed since the early 1960s. In 1962, Morrison et al[72] wrote,

[As] happens altogether too often, the differential equations are so unstable that they “blow up” before the initial value problem can be completely integrated. This can occur even in the face of extremely accurate guesses for the initial values.

It is now apparent that this observation is a direct consequence of Eq. (2.87). The eigenvalues of a dynamical system are intrinsic; hence, we cannot change the quantity a in formula Eq. (2.87). The only variable is t_f . Based on this observation, Morrison et al[72] proposed a *multiple shooting method*. The concept involves dividing the time interval $[t_0, t_f] := [0, t^f]$ into M smaller intervals so that the $|a|t^f$ is locally small:

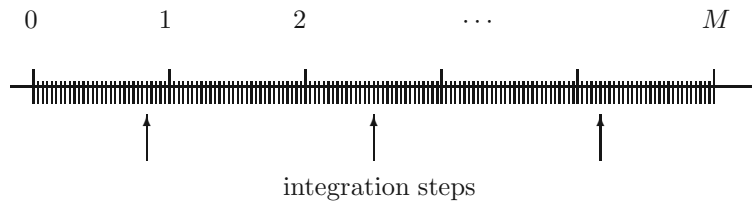


Because the local initial and final-time conditions are unknown, the concept requires an enforcement of continuity conditions:

$$\begin{array}{ccccccc}
 x(t_0) = 0 & \cdots & x(t_2^{f-}) = x(t_2^{f+}) & \cdots & \cdots & x(t_M^f) = x^f \\
 \lambda(t_0) = \lambda_0^{guess} & \cdots & \lambda(t_2^{f-}) = \lambda(t_2^{f+}) & \cdots & \cdots & \lambda(t_M^f) = N/A
 \end{array}$$

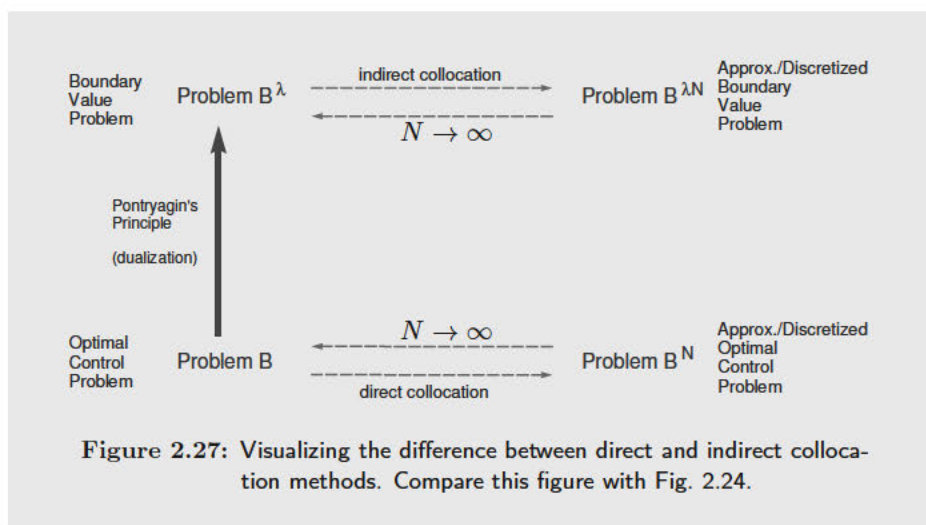
These conditions form the basis for concatenating local shooting elements where sensitivity is reduced as a consequence of shorter horizons.

Observe that each shooting element in a multiple shooting method involves discretization (via integration):






Thus, the multiple shooting element may be re-framed as a procedure that involves a “micro-discretization” at the element level coupled with a “macro-discretization” of the elements themselves. Therefore, it stands to reason that if shooting were done at the micro-discretization level, the result would be the greatest reduction in sensitivity coupled with simplifications resulting from the elimination of the macro-discretizations. This concept is the basis of *collocation methods*.

In 1987, Hargraves and Paris[36] suggested that instead of discretizing and solving Problem B^λ , it might far easier and computationally more efficient to discretize and solve Problem B directly. They based their argument of computational efficiency from the simple observation that while Problem B^λ involves discretizing $2N_x$ equations (in x and λ variables), Problem B involves only half as many: A discretization in N_x variables only (in just x variables). Their concept came to be known as *direct collocation methods* in contrast to the prior art of *indirect collocation methods*; see Fig. 2.27.



Study Problem 2.21

1. By comparing Figures 2.24 and 2.27 explain the meaning of a direct shooting method.
2.  Does direct shooting alleviate the curse of sensitivity?
3.   Pick any three optimal control software packages from the industry and categorize them (i.e., direct or indirect) based on the method used.

Collocation methods formed the basis of software packages developed in the 1990s. The main reason why these concepts were not developed and exploited in

the 1960s is because of the absence of the requisite computer memory available at that time. To appreciate this limitation, observe that a problem with N_x variables and N points requires the manipulation of $N_x N$ variables that may involve matrices of $N_x N \times N_x N$ dimensions. Ignoring sparsity, a simple problem with $N_x = 6$ and $N = 1000$ requires about $6^2 \times (1000)^2 \times 8$ Bytes or about 275 MB of memory — a number that seemed inconceivable in the 1960s. If we limit N to 100 or less, then, the memory requirements reduce to less than 3 MB, a more realistic number for a high-end computer of that era. Taking step sizes in Runge-Kutta methods with $N \leq 100$ generally produces low accuracy; hence, these methods got branded (incorrectly) as low-accuracy methods.

Study Problem 2.22

Estimate the computer memory required to solve an optimal control problem by a multiple shooting method. Based on this estimate, discuss the scale (i.e., numerical value of N_x) of problems that can be solved in a present-day computer under the absence of the curse of sensitivity.

Using the number $N_x^2 \times N^2$ as the basis of computational memory requirements it is apparent that a curse-free method is limited by the scale of the problem (N_x) for a prescribed accuracy (determined inversely by N). A low value of N implies low accuracy. Although computational capacities in the 1990s were substantially larger than those of the 1960s, collocation methods were viewed with suspicion because the practice of these methods necessitated a choice of N that was frequently low for reasonable accuracy. That is, the problems that were actually being solved were too far to the right in Fig. 2.27. As a result, the reigning idea in the 1990s was to use the low-accuracy solution of collocation methods as a seed for a shooting method on the presumption that a good guess would alleviate the curse of sensitivity (i.e., almost ignoring Eq. (2.87)). Direct collocation methods were viewed with even greater suspicion because there was no connection between the computed solution and Pontryagin's Principle. The wide gap between theory and practice opened the door for an increasingly large collection of ad hoc methods that further fueled the folklore that optimal control problems were just hard. A turning point occurred around the year 2000 through the introduction of *pseudospectral (PS) optimal control theory*[87].

The theory of PS optimal control is based on functional analysis. It is a “third” concept that is separate from either Pontryagin’s or Bellman’s. The approach is theoretically founded on the classical *Stone–Weierstrass theorem* and practically implemented by a “sufficiently high” order polynomial representation of a function in much the same way as other elementary functions (like sines, cosines, exponentials, etc.) are implemented through polynomial representations. By cutting the “middle man” out (i.e., polynomials masquerading as sines, cosines, etc.) PS optimal control theory directly seeks *designer functions* that solve Problem B . As a result, it fundamentally intertwines theory with computation. This is why it is sometimes confused as a pure computational method, albeit it did start out as such.

PS optimal control is founded on two fundamental notions:

- The idea of using a solution-centric framework that uses the differential and other constraints to shape the solution instead of “solving” the equations; and,
- The exploitation of the connection between the covectors of Problems B and B^N — a concept known as the *Covector Mapping Principle (CMP)*.

In a spectral method, the state trajectory $\mathbf{x}(\cdot)$ is expressed as an infinite series expansion

$$\mathbf{x}(t) = \sum_{m=0}^{\infty} \mathbf{a}_m P_m(t) \quad (2.88)$$

where $P_m(t)$ is a polynomial in t of degree m . The justification for expressing $\mathbf{x}(\cdot)$ in terms of an infinite degree polynomial stems from the fact that the state trajectory is absolutely continuous over the finite horizon; hence the Stone–Weierstrass theorem guarantees the existence of Eq. (2.88). Note also that elementary functions (sines, cosines, logs, exponentials, etc.) are also often expressed or computed through polynomial expansions. Hence, we may view elementary functions as shorthand notations for a special set of polynomials. From this perspective, Eq. (2.88) seeks to express the solution to an optimal control problem through a “*designer set*” of *polynomials* by cutting out the “middle man” (of elementary functions).

Note that Eq. (2.88) has the “look and feel” of a Fourier series expansion. This is not an accident! Equation (2.88) is indeed a *generalized Fourier expansion* with “amplitude” \mathbf{a}_m and “frequency” $P_m(t)$. Typical choices for the frequency basis functions ($P_m(t)$) are the “big two” orthogonal polynomials: *Legendre* and *Chebyshev*.

A key principle in a PS approach is that the coefficients \mathbf{a}_m , called the spectral coefficients, are computed indirectly by transforming Eq. (2.88) to an equivalent form

$$\mathbf{x}(t) = \sum_{j=0}^{\infty} \mathbf{b}_j \phi_j(t) \quad (2.89)$$

where t_j , $j = 0, 1, 2, \dots$ are discrete points in time called *nodes* and $\phi_j(t)$ is a Lagrange interpolating polynomial that satisfies the Kronecker relationship:

$$\phi_j(t_k) = \delta_{jk} \quad (2.90)$$

Satisfaction of the Kronecker relationship implies that

$$\mathbf{x}(t_k) = \sum_{j=0}^{\infty} \mathbf{b}_j \phi_j(t_k) = \mathbf{b}_k \quad (2.91)$$

That is, the coefficient \mathbf{b}_j in Eq. (2.89) is the value of $\mathbf{x}(t)$ at $t = t_j$. It is this *sampling property*, which is absent in Eq. (2.88), that makes a PS approach distinct from the direct use of Eq. (2.88). As a result, we can now rewrite Eq. (2.89) using a more evocative notation:

$$\boxed{\mathbf{x}(t) = \sum_{j=0}^{\infty} \mathbf{x}_j \phi_j(t)} \quad (2.92)$$

Equation (2.92) is called a *nodal representation* of $\mathbf{x}(\cdot)$ to distinguish it from its equivalent *modal representation* given by Eq. (2.88).

Using the same arguments, we write

$$\boldsymbol{\lambda}(t) = \sum_{j=0}^{\infty} \boldsymbol{\lambda}_j \phi_j(t) \quad (2.93)$$

Lastly, although it may seem logical to write the control function in the same way, that is, $\mathbf{u}(t) = \sum_{j=0}^{\infty} \mathbf{u}_j \phi_j(t)$, it turns out that this is not necessary and is potentially very limiting in applicability. The optimal control is expressed as

$$\mathbf{u}(t) = \sum_{j=0}^{\infty} \mathbf{u}_j \psi_j(t) \quad (2.94)$$

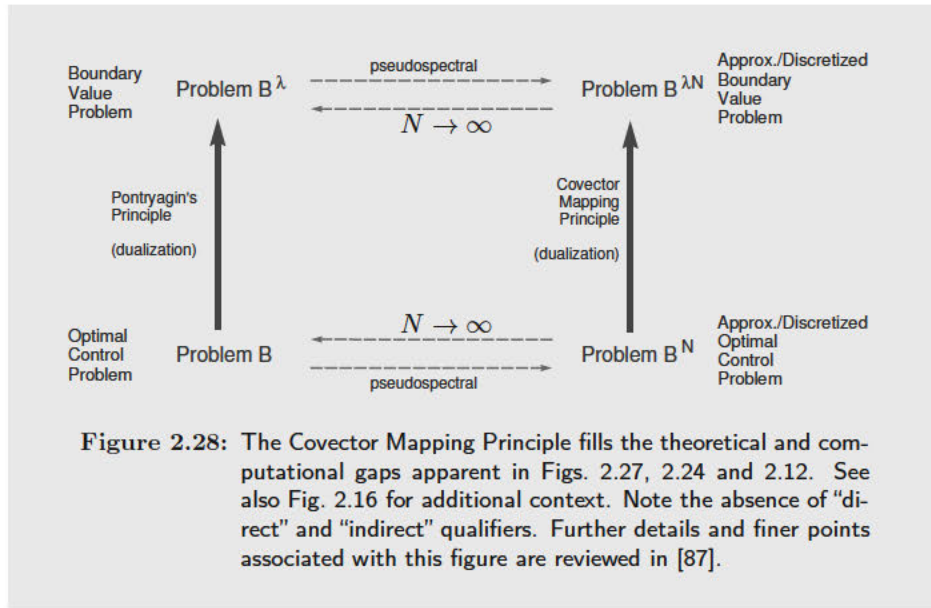
where $\psi_j(t)$ is a special interpolating function (not necessarily Lagrange) that makes the pair $t \mapsto (\mathbf{x}, \mathbf{u})$ dynamically feasible. This aspect of ψ_j has inspired an alternative PS method known as the *Bellman pseudospectral method* [84, 86]. The Bellman PS method facilitates real-time optimal control (RTOC) through the use of pseudospectral theory.

Using these elementary ideas, in addition to some newfangled algebra as Kalman predicted, it can be shown that PS optimal controls can be generated to very high accuracy (e.g., 10^{-6}) with fairly low N (e.g., $N \leq 100$). For instance, the accuracy achievable with a Runge-Kutta method for $N_{RK} \sim 1000$ can typically be obtained with an order of magnitude fewer points ($N_{PS} \sim 100$) using a spectral method. (See Section 3.2.4 on page 187 for a quantitative discussion with regard to the Brachistochrone problem.) Thus, for instance, a six-dimensional problem (i.e., $N_x = 6$) can be solved to high accuracy with less than 3 MB of computer memory — a feat that could have been achieved in the 1960s (see page 160).

One of the most powerful features of PS optimal control theory is that coefficients of the finite expansion adjoint covector function

$$\boldsymbol{\lambda}^N(t) = \sum_{j=0}^N \boldsymbol{\lambda}_j^N \phi_j(t) \quad (2.95)$$

and other covectors that arise in optimal control transform linearly to the covectors of the finite Problem B^N . This implies that Pontryagin's Principle and computation are fundamentally connected in much the same way as Bellman's Principle is connected to the adjoint covectors through the gradient of the value function. This connectivity is called the *Covector Mapping Principle (CMP)*; see Fig. 2.28. The CMP removes the suspicions (of the 1990s) of computed solutions by intimately connecting the results to Pontryagin's Principle.



In effect, the CMP completes the quest for filling the gaps that are now apparent in Figs. 2.24 and 2.27.

Note also that the CMP renders the terms “direct” and “indirect” obsolete. The plot in the story of the CMP also reveals that the obvious and “shorter path” of dualizing first and computing afterward is strewn with difficulties while a “longer path” of reversing the operations eliminates the curse of sensitivity. Thus, the CMP facilitates replacing the historical *plug-and-pray* approach with a *plug-and-play* technique by simply *commuting the key operations* indicated in Fig. 2.28.

The story of optimal control theory is replete with similar discoveries of commuting operations to produce better ways to both formulate and solve problems[84]. This notion of commuting operations to produce new and useful results can also be contextualized as a central theme in distinguishing Pontryagin’s calculus of optimal control from the classical calculus of variations. In the former, we differentiate first and minimize afterward while in the latter we minimize first and differentiate afterward. Recall that in Pontryagin’s calculus we need to determine $\partial_x H$, which is very easy to compute. By re-mapping this idea to the classical calculus of variations, we need to find the lower Hamiltonian

first

$$\mathcal{H}(\boldsymbol{\lambda}, \boldsymbol{x}, t) := \min_{\boldsymbol{u} \in \mathcal{U}} H(\boldsymbol{\lambda}, \boldsymbol{x}, \boldsymbol{u}, t)$$

and differentiate afterward

$$-\dot{\boldsymbol{\lambda}} = \frac{\partial \mathcal{H}}{\partial \boldsymbol{x}}$$

As evident from the problems and examples discussed in the preceding pages, this procedure ($\partial_x \mathcal{H} = \partial_x (\min_{\boldsymbol{u}} H)$) is considerably painful or impossible to perform as \mathcal{H} is, quite often, not differentiable even when H is very smooth.

Study Problem 2.23

Pick any recent journal or conference paper on optimal control. Critique it using the concepts developed in this chapter, and particularly, this section.

2.10 Pontryagin and the Calculus of Variations[¶]

Today, Problem *B* may seem quite a natural way to formulate many problems in disparate fields. Prior to 1955, the language of optimal control did not exist: It was invented by Pontryagin and his students[33, 66]. Obviously, without a language, optimal control problems went unrecognized in many fields. Such problems — the domain of an extremely small group of specialists[66] — had to be formulated using the old (i.e., non-control) language of the “calculus of variations.” The calculus of variations — a term coined by Leonhard Euler (1707–1783) — was invented by Joseph-Louis Lagrange (1736–1813) when he was only 19 years old! Prior to Lagrange’s invention, Euler (Bernoulli’s student) had formulated the first general problem of a “new calculus,” which can be formally stated (in modern notation) as

$$\left\{ \begin{array}{ll} \text{Minimize} & J[\boldsymbol{x}(\cdot), t_f] = \int_{t_0}^{t_f} L(\boldsymbol{x}(t), \dot{\boldsymbol{x}}(t), t) dt \\ \text{Subject to} & \boldsymbol{x}(t_0) = \boldsymbol{x}^0 \\ & \boldsymbol{e}(\boldsymbol{x}_f, t_f) = \mathbf{0} \end{array} \right. \quad (2.96)$$

[¶]This section may be skipped without loss of continuity, but why skip a good story?

Note that there is no concept of a control function in Problem 2.96, although $\dot{\mathbf{x}}(t)$ is treated somewhat independently of $\mathbf{x}(t)$. Ignoring the possible confusions arising from the meaning of such problems, an optimal control problem, prior to Pontryagin, had to be “fitted” inside of Problem 2.96. (Try this as an exercise!)

Euler derived the first necessary condition for Problem 2.96 as

$$\frac{d}{dt} \frac{\partial L(\mathbf{x}(t), \dot{\mathbf{x}}(t), t)}{\partial \dot{\mathbf{x}}} - \frac{\partial L(\mathbf{x}(t), \dot{\mathbf{x}}(t), t)}{\partial \mathbf{x}} = \mathbf{0} \quad (2.97)$$

He derived this condition through a process of discretization and limit taking[69, 95] — see bottom part of Fig. 2.28 on page 164 — obviously well before the advent of computers (which also reminds us that discretization and limit taking are independent of computation and computers). Lagrange derived the same condition using his invention of “variations.” Euler was so impressed by this idea that he abandoned his own and branded the new calculus as the calculus of variations[95]; this is why Eq. (2.97) is called the *Euler-Lagrange equation*. In the decades and centuries that followed, Problem 2.96 drew the attention of some of the greatest mathematicians. Legendre, Weierstrass, Jacobi and many others derived additional necessary conditions that bear their names. In the early 20th century (1930s), the hub of this activity was at the University of Chicago[66, 74, 95].

In the 1950s, aerospace engineers were facing a number of problems in optimal control (when the term did not even exist) and the closest available tool at that time was the calculus of variations[33]. Even if they were successful in force-fitting their problems inside the language of the calculus of variations, they had to subsequently wade through a fog of arcane necessary conditions and a large number of theorems to pull out something useful. As McShane notes, *mastery of this subject gave answers to questions no one was asking*[66]. In sharp contrast, Pontryagin — a well-known topologist at that time — completely abandoned his field to answer engineering questions that arose in the 1950s[33].

In the Spring of 1955 two (Russian) Air Force Colonels visited Pontryagin at the Steklov Mathematical Institute and posed a problem on time-optimal aircraft maneuvers[33]. Pontryagin recognized that this problem, and many other emerging ones at that time, required a new calculus. This new calculus of optimal control, developed by Boltyanski, Gamkrelidze and Pontryagin[27, 74, 75], not only generalized the old one but unified, simplified and extended

it in a way that engineers could quickly apply and use the results to develop insights into the problems they were facing in the 1950s. Among many, one of the fundamental shifts was the introduction of the *state-space model*, $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, t)$, as a constraint in a *dynamic optimization problem*. This “simple” perspective changed everything! For instance, Problem (2.96) can be formulated in this “new language” as

$$\left\{ \begin{array}{ll} \text{Minimize} & J[\mathbf{x}(\cdot), \mathbf{u}(\cdot), t_f] = \int_{t_0}^{t_f} L(\mathbf{x}(t), \mathbf{u}(t), t) dt \\ \text{Subject to} & \mathbf{x}(t_0) = \mathbf{x}^0 \\ & \dot{\mathbf{x}}(t) = \mathbf{u}(t) \\ & \mathbf{e}(\mathbf{x}_f, t_f) = \mathbf{0} \end{array} \right. \quad (2.98)$$

What is more remarkable about this new perspective is that through the introduction of Pontryagin’s Hamiltonian, the awkward Euler-Lagrange equation given by Eq. (2.97) transforms to the very elegant *dual form* given by

$$\frac{\partial H}{\partial \mathbf{u}} = \mathbf{0} \quad (2.99)$$

The price for this elegance is, of course, the introduction of covectors as a fundamental analysis tool. Equation (2.99) is *equivalent* to, but is not the same as, the Euler-Lagrange equation. Equation (2.99) follows from Problem HMC, but under three severe conditions:

1. The control space \mathbb{U} must be continuous (see Fig. 2.2 on page 87),
2. The H -function must be differentiable with respect to \mathbf{u} , and
3. The optimal control must be interior to \mathbb{U} .

In Chapter V of [75], Pontryagin et al derived the Euler-Lagrange equation and all the other necessary conditions of the calculus of variations using their new *simpler* tools while simultaneously showing the limitations of the former. Despite this, many in the mathematical community were reluctant to accept their results as something completely new[1]. On the other hand, engineers wholeheartedly accepted and endorsed the new ideas[1, 66] because they were practical and systematic: There was no longer a need to understand or apply the chaos of limitless theorems of the calculus of variations. Not to be beaten,

the traditionalists reorganized the calculus of variations using Pontryagin's new ideas and "showed" that the Hamiltonian minimization condition was "merely" a generalization of Weierstrass' condition. Using similar rewrites, it can be shown that

$$\frac{\partial^2 H}{\partial \mathbf{u}^2} \geq 0 \quad (2.100)$$

is a restatement of the *Legendre-Clebsch condition*. Hence, the argument went, it was "obvious" from equations (2.99) and (2.100) that the Hamiltonian must be minimized. This gave credibility to those educators who chose to inflict the classical calculus of variations upon unsuspecting students. A genuine new calculus blending the old with the new had to wait until Clarke's breakthrough of nonsmooth calculus[20, 23].

In 1989, McShane warned[66] that mathematicians had not learned from the history of the 1950s. New "baroque theorems," he criticized, were being generated of "increasing intricacy [that is] of interest to a steadily shrinking collection of experts in the subject." Even worse, theory and computation got completely divorced in the 1990s, leading to such notions as "direct" methods (apparently not based on theory) and "indirect" methods (apparently based on Pontryagin's Principle). See Section 2.9.2 on page 157 and [77] on the origins of this "divorce," and Section 3.7 on page 240 for a counter example. A turning point occurred around the year 2000 when the Covector Mapping Principle was introduced[87]. In one fell swoop, Pontryagin's theory, computation and problem-solving became fundamentally intertwined; see Fig. 1.50 on page 84. In 2006, *SIAM News* heralded the dawn of a new era: Pseudospectral optimal control theory had debuted flight.

2.11 Endnotes

Pontryagin's Principle is named in honor of Lev S. Pontryagin (1908–1988), who, along with his associates at the Steklov Institute of Mathematics, Moscow, was the first (1956) to articulate all major elements of this fundamental concept[1]. Barring a footnote on page 45 in [75] where they refer to *covariant* vectors, Pontryagin et al never used the word *covector* in their entire book; however, they were absolutely clear everywhere — through their tensor notation — that their "auxiliary variables" (i.e., covectors in this book) were indeed covariant. The word covector also means a *covariant vector*.

After Pontryagin, many applied textbooks introduced these covectors as Lagrange multipliers — a name change that does little to unlock their mystery.^{||} In the mathematics literature, Pontryagin's auxiliary variables were indeed identified as covectors but in their abstract form as linear functionals. The mystery and suspicion of Lagrange multipliers, their apparent lack of physical meaning funneled through the curse of sensitivity fueled the folklore that optimal control problems were hard. They indeed were! But everything changed around the year 2000 when the escape clause on the curse of sensitivity was thoroughly exploited. See [84] for historical and technical details.

The concept of covectors, as presented in Sections 2.2 and 2.3, grew as an outgrowth of the scaling and balancing procedure used in DIDO when the software was first created in 2001. Subsequent experience with DIDO helped codify a re-interpretation of covectors as presented in this chapter. Interestingly, the physical interpretation of covectors presented in this chapter is more closely aligned with Pontryagin's original geometric view of the calculus of optimal control. The presentation in Section 2.2, particularly the bottom of Fig. 2.5, was inspired by the combination of Pontryagin's geometric insight on duality and the classic interpretation of covectors in physics as presented by Misner, Thorne and Wheeler[68].

Our treatment of covectors is in sharp contrast to their tricky introduction as Lagrange multipliers presented in many textbooks on applied optimal control where there is a frequent lamentation that the multipliers have no physical meaning. Having dispensed with this myth, the measuring and scaling procedure offered by the covectors can be used very effectively to solve many optimal control problems in DIDO. This implies that a practical understanding of Pontryagin's Principle is more important than ever before. Simply running a code does little in understanding Pontryagin's Principle or the problem being solved. That is, to solve a "hard" optimal control problem "easily," it is critical to apply Pontryagin's Principle in a manner that exploits the problem-specific covectors using the "new rules" and insights presented in this chapter. Simple examples illustrating part of this methodical procedure are presented in Chapter 3 with additional problems and insights in Chapter 4. For application problems, such as some of those presented in Chapter 4, Pontryagin's Principle must be applied

^{||}Lagrange multipliers were actually invented by Euler as part of his quest to solve Queen Dido's problem[23, 96]: See Section 4.6 on page 282 for a discussion on this problem and [96] for a well-researched historical account.

as indicated in Fig. 2.16, page 124. This is why a pragmatic understanding of Pontryagin's Principle is essential to producing good DIDO application codes. In other words, *theory is not divorced from computation!* More importantly, we now have an intimate connection between theory, computation and problem formulation itself!! Solutions from an initial problem formulation feed new problem formulations. This is part of the *problem of problems* concept (described in Chapter 1) where the DIDO solutions from a given problem are analyzed and used to reformulate the problem repeatedly until the correct problem is formulated; see the work flow outlined in Fig. 1.50 on page 84. Welcome to a journey of discovery!