IDIAP RESEARCH REPORT

Online statistical estimation for vehicle control

Christos Dimitrakakis [a]

IDIAP–RR 06-13

March 2006

[a] IDIAP, CP952, 1920 Martigny, Switzerland, dimitrak@idiap.ch

IDIAP Research Institute **www.idiap.ch**
Rue du Simplon 4 P.O. Box 592 1920 Martigny – Switzerland
Tel: +41 27 721 77 11 Fax: +41 27 721 77 12 Email: info@idiap.ch

IDIAP Research Report 06-13

Online statistical estimation for vehicle control

Christos Dimitrakakis

March 2006

Abstract. This tutorial examines simple physical models of vehicle dynamics and overviews methods for parameter estimation and control. Firstly, techniques for the estimation of parameters that deal with constraints are detailed. Secondly, methods for controlling the system are explained.

Contents

1 Introduction

This tutorial deals with a number of topics related to vehicle control. The primary problem related to control is physical modelling, or more specifically the estimation of parameters in parametric physical models.

Statistical estimation can be used to estimate unknown quantities given a model for the world, when there some noise or uncertainty is admitted into the measurements or process. One of the first examples of the use statistical methods had been in the estimation of planetary movements in the solar system. This estimation posed the following problem: while the trajectory of a planet can be found exactly by solving a system of equations given only a small number of observations, the observations were noisy and different sets of observations would give different trajectories. The solution was to find a single set of parameters that approximately satisfy all the equations that result from the application of the planetary laws of motion to all our observation. In general, this principle can be stated as follows: given a parametric model and some observations, find the parameters that best explain the observations. The field is ultimately a mere application of statistics and essentially identical techniques are used everywhere from physical modelling to artificial intelligence.

The purpose of this paper is to give a tutorial on model estimation and its application to vehicle control. A number of motivating examples will be used throughout the text, with particular emphasis on the relatively simple problems of estimating the physical properties of a braking model and the necessary controls for executing braking manoeuvres.

We first give an overview of a parametrised braking/acceleration model and we outline how it can be employed in a lapped race framework. Such a framework is particularly interesting because it allows for taking repeated measurements around the track. Furthermore, it admits a simple parametrised model for properties of the track surface. We specify the given quantities and parameters of the model and explain how they can be used to build an accurate prediction of the acceleration that results from the application of car controls. Subsequent sections are more general in scope, however the initial model parametrisation is used throughout.

Section 3 outlines how to perform parameter estimation and how to use the estimated parameters to solve the control problem when an inverse solution available in closed form. This section begins by viewing the problem of parameter estimation as an optimisation problem and providing gradient-based optimisation techniques for solving it. Subsequently it touches upon more advanced estimation topics, such as the estimation of parameters with constraints or with a given prior probability distribution and it explores the relations between the two.

Section 3 deals with the case where given a model, it is possible to solve for the optimal control and the task is to determine what the model is. A complementary situation occurs when the model is known but there is no closed form solution for the optimal control. In that case, if a cost can be defined analytically, the problem can be formulated as that of optimising the cost function with respect to the free parameters. Section 4 focuses on this view of control as a deterministic optimisation problem and gives two examples: firstly, that of determining the optimal braking input such that we stop after a given number of meters – secondly, that of determining the optimal trajectory of a vehicle around a track.

The hardest case occurs when the model is unknown, and neither the optimal control nor the cost function can be formulated analytically. Section 5 considers this extreme case and compares the different methodologies.

Finally, I would like to note that this paper will be expanded and corrected as necessary. Please feel free to email me with corrections, suggestions and questions, especially about sections which seem particularly unclear.

2 Modelling the car

We will first try to estimate the acceleration that results from the application of the brake or gas pedal at different speeds. We have some given quantities, but the model is too complex for us to be able to model everything: different parts of the race track might have characteristics that are easy to throw off any model that we might care to make. Furthermore, a very detailed model has the disadvantage that it then becomes difficult to solve. In the end we will wish to calculate appropriate braking values for various situations using our model, so this an important fact to keep in mind.

We try to estimate the following model, which ignores all but the simplest vehicle dynamics. In particular, the variability due to terrain conditions, road curvature and inclination and other factors, are not taken into account; building an explicit model for this purpose can be error-prone. The following equation is our model of the car's acceleration given the current speed, the mass, the horse and braking power, some known physical constants and some free parameters:

$$du/dt = \mathrm{dtan}\left(w_b x_b + \frac{w_a x_a}{\max(u, 10)}\right)(\mu + w_\mu)(G + \frac{C_D u^2}{m}) - \frac{C_R |u| u}{m} \tag{1}$$

where x_a, x_b are the acceleration and braking input respectively, μ is an *a priori* given friction coefficient and G is the assumed gravitational acceleration. The vehicle speed is denoted by u, and C_D, C_R represent the aerodynamic coefficients for downforce and resistance.

A lot of the system's state is not taken into account in this equation. We attempt to model this unknown variability by introducing additional terms: scalar functions which depend on the state. What the "state" is a design issue. In some cases, a single scalar value, rather than a function, will be sufficient. In our application it was decided to use linear functions that depend on a representation of the track position. This implementation is described below.

2.1 The parameters

Our purpose is to estimate w_a, w_b, w_μ. These are not scalar quantities, but parametrised functions of the form $w = \theta^T p$, where θ is a parameter vector and p summarises information about the position.

The predominant application we have in mind is racing multiple laps on a single track. To apply the method in this case, the track is split into n logical segments and we define p to have the form $(1, 0, \ldots, 0, 1, 0, \ldots, 0)$ and length $n + 1$, where n is number of logical segments in which the track is split. Thus, the first component of each parameter vector will be globally adjusted for the whole track, and the others will describe local variations. It is possible to use smoother functions, but we are not concerned with those.

2.2 The model

Before we derive updates, we explain equation (1) in a bit more detail. The first factor describes the accelerating effect of gas or break pedal application. The terms $x_a, x_b \in [0, 1]$ measure the amount of gas/break pedal application. The term w_a models the car's available power. Since $F = P/u$, this offers a natural way to model the possible force. The maximum value of this force is clamped to be that available at a speed of 10 ms^{-1}. The term $w_a b$ is the maximum braking force possible. Both w_a, w_b are implicitly scaled by the car's mass.

Since the maximum possible fricative force is given by the second and third multiplicative terms, we use the dtan function to clamp the magnitude possible acceleration to this value. It is defined as:

$$\mathrm{dtan}(x) = \begin{cases} -1 & x < -1 \\ 1 & x > 1 \\ x & \text{otherwise} \end{cases}$$

with derivative

$$\partial \operatorname{dtan}(x)/\partial x = \begin{cases} 1 & x \in [-1, 1] \\ 0 & \text{otherwise} \end{cases}$$

The clamping of the acceleration force is essential. Correctness notwithstanding, it will be possible to fit many equivalent models to the same observations [1] if this constraint is not introduced.

Why do we need to model these quantities? Firstly, these might not be easy to measure *a priori*. But even if they are, the simplicity of our model means it can be thrown off by variability in track layout. By using position-varying functions we can model effects such as the reduction in acceleration due to climbing an incline without explicitly modelling the incline itself. This results in firstly a significant reduction in computation and secondly, and most importantly, the ability to approximately model unknown, or difficult to compute, physical effects.

The following section outlines methods of estimating those parameters and derives an inverse control solution for our model.

3 Parameter estimation and inverse control

We wish to estimate the parameters of our model from observations. Since our model can make predictions, a good measure of how good our parameters are is how close our predictions are to the observations.

For the above model, we consider the case of measuring the speed $u(t)$ at different times t. From these measurements we can calculate $du(t)/dt = a = (u(t+\Delta_t) - u(t_1))/\Delta_t$, the acceleration. Formally, these accelerations will be our *observations*. Our *prediction* will be our estimate of acceleration, which we will call $\hat{a}(w)$. Our purpose is to have $\hat{a}$ as close to a as possible. The *prediction error* can be is written as $a - \hat{a}(w)$. We can measure the magnitude of the predicition error over all the observed data through the *mean squared error*

$$f(w) = E[(a - \hat{a}(w))^2]$$

the expected value of the squared prediction error. Now we can formulate our *parameter estimation* problem into a *minimisation* problem where we try and minimise the average square error. In order to estimate the 'true' parameters for our model, we will try to find the parameters that minimise this function[2]. To do that, we employ one of the many methods for the minimisation of functions: stochastic steepest gradient descent, possibly the simplest online estimation method.

3.1 Minimisation with gradient descent.

Gradient descent methods are among the most commonly used methods for optimisation tasks. This short tutorial will offer an exposition to the simplest available gradient methods for statistical estimation. For a further look into optimisation techniques Bertsekas (1999) offers a good overview, with a healthy amount of mathematical rigour, while Boyd and Vandenberghe (2004) additionally refers to the use of optimisation techniques for statistical estimation.

First, we give some definitions:

3.1 Definition (Gradient) *The gradient of a scalar function f with respect to $x = (x_1, \ldots, x_n)$ is defined as the vector of partial derivatives:*

$$\nabla_x f \equiv \frac{\partial f}{\partial x} \equiv \begin{bmatrix} \partial f/\partial x_1 \\ \vdots \\ \partial f/\partial x_n \end{bmatrix}$$

[1] For a model of the type $du/dt = x_b w_b \mu$, it is possible to scale down μ N times and scale up w_b by the same amount to obtain an equivalent solution. In that case μ will not have the physical meaning that we would expect.

[2] We assume that the true parameters will be close to the parameters that minimise f.

For a function $f \in \mathbb{R}^m$ the result is a matrix.

$$\nabla_x f \equiv \frac{\partial f}{\partial x} \equiv \begin{bmatrix} \partial f_1/\partial x_1 & \cdots & \partial f_m/\partial x_1 \\ \vdots & \ddots & \vdots \\ \partial f_1/\partial x_n & \cdots & \partial f_m/\partial x_n \end{bmatrix}$$

Gradient descent methods attempt to minimise a function by moving in the direction of the gradient, i.e. “downwards”. For smooth functions they are guaranteed to find at least a local minimum. For convex functions, such as quadratic functions, if there exists a minimum it is unique.

3.2 Definition (Steepest gradient descent) *Given a vector $x_t \in \mathbb{R}^n$, and $f : \mathbb{R}^n \to \mathbb{R}$, update parameters in the steepest descent direction:*

$$x_{t+1} = x_t - \alpha_t \nabla f,$$

where α_t is a step size *or* learning rate *parameter.*

It might appear difficult to apply such a method in our case, where our function to minimise is a statistical expectation (the expected squared prediction error). However, this is far from being true. On the contrary, it is quite trivial to do so. Remember that the expectation of a random variable Z is defined as:

3.3 Definition (Expectation) *For a random variable Z with realisation $Z(t)$ at time t, the following holds:*

$$E[Z] = \lim_{N\to\infty} \frac{1}{N} \sum_{t=1}^{N} Z(t).$$

The expectation is also frequently called the mean. *For a limited number of samples N, the quantity $\frac{1}{N}\sum_{t=1}^{N} Z(t)$ is called the* sample mean.

Hence, the solution is to only take a limited number of samples of our variable and attempt to minimise the sample mean instead of the true expectation. Let's say that $f = E[Z]$, where Z is some stochastic function that depends on parameters x and that we try to find x such as to minimise f. We can instead minimise

$$\hat{f} = \hat{E}[Z] \propto \sum_{t=1}^{N} Z(t).$$

The derivative of this is simply

$$\nabla \hat{f} \propto \sum_{t=1}^{N} \nabla Z(t).$$

So we only need to find the gradient with respect to each sample of Z. However, we need to collect N samples before we can apply our method. How can we get around that? The simplest way is to use what is called *stochastic gradient descent*. Instead of waiting to collect a lot of data and performing the parameter updates later, we simply perform a parameter update everytime we have a new Z. If we use a sufficiently small step size, this will not be a problem.[3]

3.1 Example (Mean square estimation) *In our particular case, where the random variable Z is the prediction error, we have $Z = \|a - \hat{a}\|^2$ and thus we try to minimise the cost function*

$$\hat{f} = \hat{E}[Z] = \sum_i (a_i - \hat{a}_i)^2, \tag{2}$$

where, since we are minimising over multiple instances, the index i denotes index of the predictions of and observations.

[3]Stochastic gradient descent can be viewed as “gradient descent with errors”. The errors normally have a mean of zero and do not affect convergence in a bad way. In fact, as far as first-order gradient methods go, stochastic steepest gradient descent seems to perform the best in a wide range of applications.

3.2 Application of gradient descent.

Let's consider again our model: we have a function $\hat{a}$, a model of the acceleration and observe values a. We have parameters w_b, w_a, w_μ, which we can collectively call w for convenience. We want to minimise the sum of prediction errors, which can be done by moving in the descent direction for each individual prediction error. The only thing we need to find is the descent direction of function f for an observed value a and a prediction $\hat{a}$ for each of the parameters in w.

To determine the derivative we need the derivative chain rule:

$$\frac{\partial f}{\partial g} = \frac{\partial f}{\partial h}\frac{\partial h}{\partial g}.$$

Using this, we can calculate the gradient of our cost function for each pair of observations and predictions.

$$\begin{aligned}
\nabla_w(\hat{a}-a)^2 &= \nabla_{(\hat{a}-a)}(\hat{a}-a)^2\nabla_w(\hat{a}-a)\\
&= \nabla_{(\hat{a}-a)}(\hat{a}-a)^2(\nabla_w\hat{a}-\nabla_w a)\\
&= 2(\hat{a}-a)(\nabla_w\hat{a}) \quad (a \text{ does not depend on } w).
\end{aligned}$$

Now we have to continue applying the chain rule until we have terms that do not contain any gradient operations and that directly relate to our parameters. So we need to compute $\nabla_w\hat{a}$.

$$\begin{aligned}
\nabla_w\hat{a} &= \nabla_w \operatorname{dtan}\left(w_b x_b + \frac{w_a x_a}{\max(u,10)}\right)(\mu+w_\mu)(G+\frac{C_D u^2}{m}) - \nabla_w\frac{C_R|u|u}{m}\\
&= \nabla_w \operatorname{dtan}\left(w_b x_b + \frac{w_a x_a}{\max(u,10)}\right)(\mu+w_\mu)(G+\frac{C_D u^2}{m}).
\end{aligned}$$

We substitute some terms to simplify the look of this:

$$\nabla_w\hat{a} = \nabla_w \operatorname{dtan}(f)(g)(h)$$

Now, recall that $w = (w_b, w_a, w_\mu)$. We have:

$$\nabla_w\hat{a} = \begin{bmatrix}\partial\hat{a}/\partial w_a\\ \partial\hat{a}/\partial w_b\\ \partial\hat{a}/\partial w_\mu\end{bmatrix} = \begin{bmatrix}\frac{\partial \operatorname{dtan}(f)}{\partial f}\frac{\partial f}{\partial w_a}gh\\ \frac{\partial \operatorname{dtan}(f)}{\partial f}\frac{\partial f}{\partial w_b}gh\\ \operatorname{dtan}(f)\frac{\partial g}{\partial w_\mu}h\end{bmatrix}$$

$$= \begin{bmatrix}\frac{\partial \operatorname{dtan}(f)}{\partial f}x_a gh\\ \frac{\partial \operatorname{dtan}(f)}{\partial f}x_b gh\\ \operatorname{dtan}(f)h\end{bmatrix} = \begin{cases}\begin{bmatrix}x_a gh\\ x_b gh\\ \operatorname{dtan}(f)h\end{bmatrix} & f\in[-1,1]\\ \begin{bmatrix}0\\ 0\\ \operatorname{dtan}(f)h\end{bmatrix} & f\notin[-1,1]\end{cases}$$

Remember that the full gradient is $2(\hat{a}-a)(\nabla_w\hat{a})$. The update rule will be

$$w(t+1) = w(t) + 2(a-\hat{a})(\nabla_w\hat{a}),$$

with each parameter in the vector w updated separately. When one looks upon the meaning of this update, it is easy to see that the system works nicely: when our estimated values indicate that the braking/accelerating force is beyond the limits of the current friction estimate, only the friction coefficient is updated. In experiments, it was possible to calculate values for the friction coefficient, motor power and braking force with error around 5% (see figure 1). In experiments with

TORCS (Espié et al., 2006), it was found that the error in modelling is smallest when we completely ignore the addition of the downforce to the reaction force. This suggest that we need to add additive and/or multiplicative parameters to G and/or C_R.

Further to the above, it is also possible to have each one of the components of w itself be a function, with parameters θ. In that case we have to calculate $2(a-\hat{a})(\nabla_w \hat{a} \nabla_\theta w)$. For our problem, we do what was initially outlined in section 2.1. We have $w = \theta^T p$, where the elements of p are in $\{0, 1\}$. This causes p to effectively *select* elements from θ. These are summed together in the model prediction. Furthermore, the gradient is 0 for all those elements of w which correspond to elements in p which are 0. This makes the implementation very simple.

A step further from this method is to use allow the elements p to take values in $\mathbb{R}$, and even further than that, to allow w to be a nonlinear function. We will not attempt to derive updates for other cases, however the reader should not be afraid to try. The method remains the same. All that is necessary is to repeatedly calculate the derivatives until a simple parameter update can be computed.

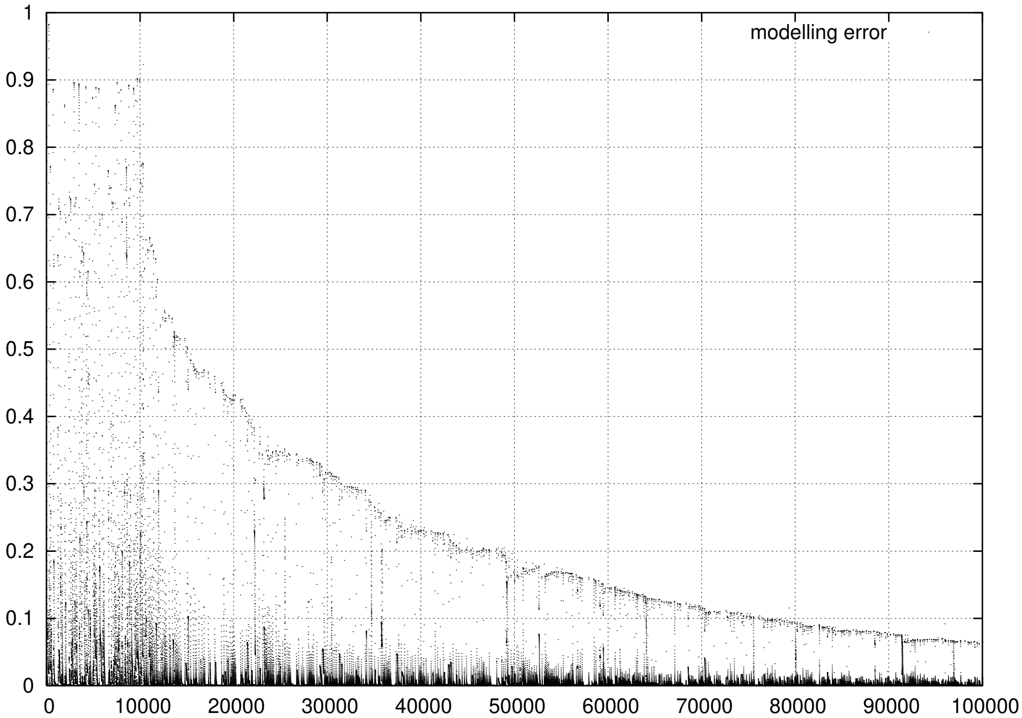

Figure 1: Ratio of modelling error to actual value of acceleration. The system was initialised with very poor parameters.

3.3 Control using estimates

The difficulties begin when we consider controlling the system. Herein we will make use of the simplest control method: taking the differential equation that we use for a model, and solving it to determine the quantity of interest. In a lot of cases there is an analytic solution.

We consider the following simple model:

$$du/dt = -(ax + b),$$

where we have a current speed $u(0)$ need to find x such that we stop after s meters, $x \in [0, 1]$ (if

$x \in \mathbb{R}$, the problem becomes ill-posed). We have:

$$u(T) = u(0) - \int_0^T (ax+b)dt = u(0) - (ax+b)T.$$

And from this,

$$\begin{aligned} s(T) = \int_0^T u(t)dt = \int_0^T \big(u(0) - (ax+b)t\big) \\ = \int_0^T u(0)\ dt - \int_0^T (ax+b)t\ dt = u(0)T - \frac{1}{2}(ax+b)T^2 \end{aligned} \tag{3}$$

.

It must hold that $T = u(0)/(ax+b)$ if we stop at time T. We substitute this and obtain

$$s(t) = \frac{u(0)^2}{ax+b} - \frac{u(0)^2}{2(ax+b)} = \frac{u(0)^2}{2(ax+b)} \tag{4}$$

$$2(ax+b) = \frac{u(0)^2}{s(t)} \tag{5}$$

$$x = \frac{1}{a}\left(\frac{u(0)^2}{2s(t)} - b\right) \tag{6}$$

This is quite a useful equation. Given a desired stopping distance, we can calculate x, the brake control. Or, we can calculate the minimum stopping distance by setting $x = 1$. However this calculation relies on the assumption that our model is correct. This is never the case. There are two types of possible errors for our model: errors in parameter estimation and errors in model selection. Both types can cause problems. We first consider parameter estimation errors. More specifically we analyse our solution in the presence of additive errors ϵ_a, ϵ_b on a, b.

$$\begin{aligned} x' - x = \frac{1}{a+\epsilon_a}\left(\frac{u(0)^2}{2s(t)} - (b+\epsilon_b)\right) - x \\ = \frac{1}{a+\epsilon_a}\left(\frac{u(0)^2}{2s(t)} - (b+\epsilon_b)\right) - \frac{1}{a}\left(\frac{u(0)^2}{2s(t)} - b\right) \\ = -\frac{\epsilon_a\left(\frac{u(0)^2}{2s(t)} - b\right) + \epsilon_b(a+\epsilon_a)}{(a+\epsilon_a)a} \\ = -\frac{\epsilon_a}{a+\epsilon_a}\frac{\frac{u(0)^2}{2s(t)} - b + \epsilon_b}{a} \end{aligned} \tag{7}$$

In figure 2 we can see that the squared error of estimation $\|x' - x\|^2$ depends linearly on the parameter error for positive errors, but geometricaly for negative errors. This can be a serious cause of instability for control and in fact arises in a lot of cases where we must perform inversion of a function whose parameters we are estimating. Its occurence in such a simple model is disturbing.

The problem arises mainly because our model makes an implicit assumption which has been ignored: namely that a is positive definite. To get rid of this problem, we must place constraints on our parameters.

3.4 Estimation of constrained parameters

There are a number of ways to deal with assumptions that force constraints on parameters. There are three main methods that deal with constraints: *projection, penalty terms* and *boundary methods*.

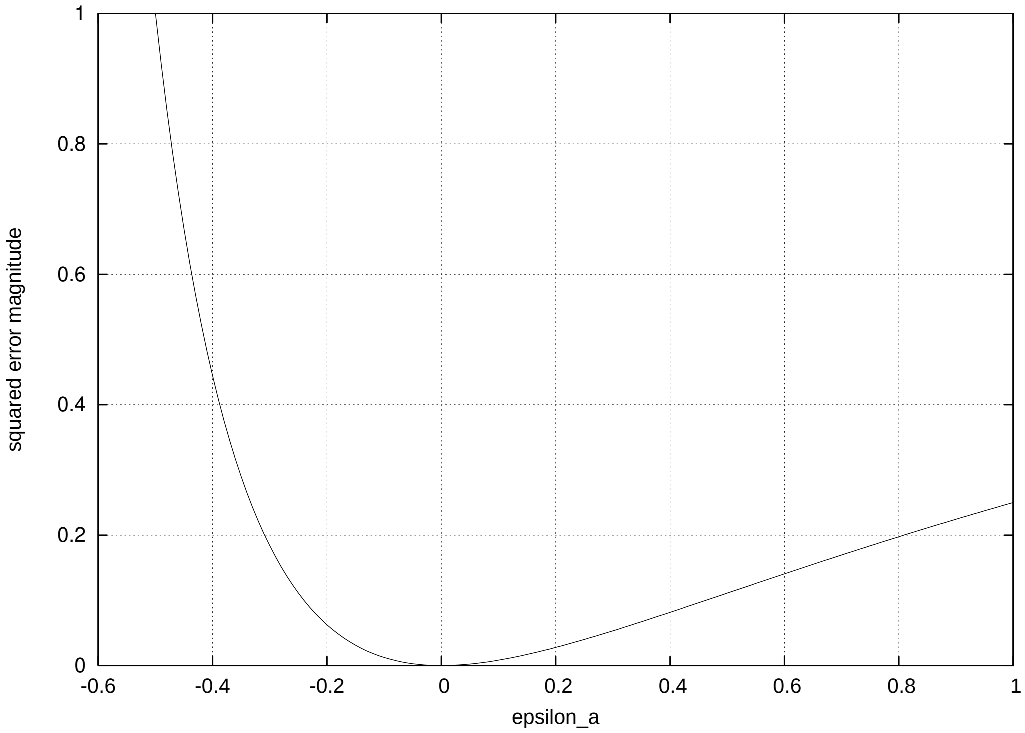

Figure 2: Squared estimation error dependence on ϵ_a.

In some contexts we may assume a *non-uniform* distribution of a parameter over a given set (or over the whole of the real line). Making such an assumption is usually called *placing a prior distribution over the parameters* and is the subject of Bayesian methods. As we will see later on, the use of all the three methods mentioned above places an implicit prior over the parameters.[4]

3.4.1 Projections

Suppose that we want to estimate a quantity a in some set $\mathcal{S}$, where $\mathcal{S}$ represents the possible values that a can take. Imagine we have an estimation algorithm with parameter $x \in \mathcal{U} \supseteq \mathcal{S}$. Instead of directly setting $x = E[a]$, we can define a function $g : \mathcal{U} \to \mathcal{S}$ which projects any parameter value of x into the constrained set $\mathcal{S}$. This means we will try to estimate $g(x) = E[a]$.

3.2 Example (Mean estimation of an unknown parameter) *Possibly the simplest case is to calculate the expected value $E[A] = a$ of some random variable A with realisations $(A_1, A_2, \ldots, A_N)$. One can say that these realisations represent noisy measurements of a. Then the sample mean gives us*

$$\hat{E}[A] \equiv E[A|A_1, \ldots, A_N] = \sum_{i=1}^{N} A_i/N.$$

This is elementary, but imagine we have the constraint that $a > 0$ and that, additionalyl, the measurements can take any value, i.e. $A \in \mathbb{R}$. One way to solve this problem is to estimate $x \in \mathbb{R}$ and

[4]If there is no prior (meaning that the prior distribution is uniform) all parameter values are equally likely and we are doing a "maximum likelihood" estimate. On the other hand, an estimate that uses both a prior and the data is called a maximum *a posteriori* (MAP) estimate.

Method	Mean squared estimation error	Squared error variance
Linear	0.0989	0.0184
Square root	0.0879	0.0110
Logarithmic	0.0768	0.0092

Table 1: Estimation errors for $a = 0.5$

use a projection g to put obtain a value in the constrained set. Two different examples of suitable projections follow.

3.3 Example (Square root estimation of a strictly positive parameter) *We have $a > 0$, with realisations $A_i \in \mathbb{R}$. We need to estimate $x \in \mathbb{R} : g(x) = \hat{E}[A]$ where $g : \mathbb{R} \to \mathbb{R}^+$ defines the following projection:*

$$g(x) = x^2.$$

This is called a square root method because the parameter x corresponds to the square root of the value that we need to estimate.

3.4 Example (Logarithmic estimation of a strictly positive parameter) *We have $a > 0$, with realisations $A_i \in \mathbb{R}$. We need to estimate $x \in \mathbb{R} : g(x) = \hat{E}[A]$ where $g : \mathbb{R} \to \mathbb{R}^+$ defines the following projection:*

$$g(x) = e^x$$

This is called a logarithmic because the parameter x corresponds to the logarithm of the value that we need to estimate.

In order to apply either method we will find $x^* : E[\|A - g(x^*)\|^2] \leq E[\|A - g(x)\|^2 \; \forall x \in \mathbb{R}$. To do that we simply minimise the sample square error between the realisations A_i and g. Thus, our problem becomes the problem of minimising $f(x) = \sum_i \|A_i - g(x)\|^2$. For each sample A_i, we have:

$$\frac{\partial f}{\partial x} = 2(A_i - g)(-\frac{\partial g}{\partial x}).$$

In the first case, $\frac{\partial g}{\partial x} = 2x$ and in the second case $\frac{\partial g}{\partial x} = xe(x)$ so we have:

$$\frac{\partial f}{\partial x} = -2(A - g) \quad \text{for } g(x) = x, \tag{8}$$

$$\frac{\partial f}{\partial x} = -4(A_i - g)x \quad \text{for the square root method,} \tag{9}$$

$$\frac{\partial f}{\partial x} = -2(A_i - g)e^x \quad \text{for the logarithmic one.} \tag{10}$$

We now perform a small experiment to test the performance of those methods. We need to estimate $a > 0$ from 10 random measurements A_i which are drawn from a normal distribution with mean a and variance 1. We run 1000 experiments, for which we estimate a with the three above methods. The linear method sometimes estimated a negative value for a. The tables summarise the results.

As can be seen, the projection methods not only are free from the possibility of violating the imposed constraints, but can also exhibit better average performance. Whether or not they do depends upon whether the distribution of a agrees with that implicitly defined by the method.

Method	Mean squared estimation error	Squared error variance
Linear	0.0986	0.0192
Square root	0.0812	0.0139
Logarithmic	0.0718	0.0113

Table 2: Estimation errors for a uniforly distributed in $[0.01, 1.1]$

Method	Mean squared estimation error	Squared error variance
Linear	0.0986	0.0192
Square root	0.0977	0.024996
Logarithmic	0.11544	0.10461

g

Table 3: Estimation errors for $a > 0$ exponentially distributed with unit variance

3.4.2 Penalty and barrier methods

Penalty and barrier methods are conceptually similar. In some sense, they both penalise solutions that are beyond, or merely close to, a boundary that defines the constraints. Although related in this way, they optimisation problem is formulated differently in each case. Herein we consider only the penalty formulation with gradient descent.

Our problem again is to estimate $g(x) = E[A]$, with constraints $f(x) > 0$. We may formulate the problem as the minimisation of the cost function:

$$C(x) = \|g(x) - E[A]\|^2 + \lambda h\big(f(x)\big),$$

where $h(\cdot)$ is monotonic increasing, bounded from below, and $\lambda > 0$. Contrary to the other methods, there is nothing to prevent x from assuming invalid values. However, as $\lambda \to \infty$, $C(x)$ also approaches infinity for any x that violates the constraints. Finding appropriate values for λ is not trivial. There are three approaches:

1. Use a fixed λ.
2. Start with a small λ and slowly increase until x does not violate the constraints.
3. Use a Lagrange multiplier method to simultaneously solve for x, λ.

3.4.3 Alternative measures

It is of course possible to use other measures than the euclidean norm. These can be better understood in the probabilistic framework, described below.

3.5 Bayesian methods

Here we make use of a probabilistic framework. We assume a *prior distribution* $p(x)$. We then create a model of the form $p(A|x)$ and attempt to find the most probable parameters by *maximising*

$$p(x|A) \propto p(A|x)p(x). \tag{11}$$

Let us first consider the case where $p(x)$ is uniform, i.e. all possible values of x have equal probability. That means we have to find x that maximises $p(A|x) = p(A_1, A_2, \dots |x)$. Firstly, we assume that the measurements are independent and we have:

$$p(A|x) = \prod_i p(A_i|x).$$

So we need to calculate the density for each individual measurement A_i, and find parameters that maximise the product. For the particular case where $p(A_i|x) \propto \exp(-\|A_i - x\|^2)$, we have:

$$p(A|x) = \frac{1}{Z}\prod_i \exp(-\|A_i - x\|^2) = \frac{1}{Z}\exp(-\sum \|A_i - x\|^2),$$

where Z is a normalisation constant. Now, let's turn this into a minimisation problem. First of all, an x that minimises $f(x)$, will also minimise $g(f(x))$ for any monotic increasing function g. In this case, we can use the log function to obtain:

$$\log p(A|x) \propto -\sum \|A_i - x\|^2.$$

Now, we reverse signs to turn the maximisation problem into minimisation one, and.. it turns out we are minimising $\sum_i \|A_i - x\|^2$, which is exactly the same as before. So, how is that useful?

Firstly, nothing limits us to using a Gaussian for the densities $p(A_i|x)$. We may use different densities in order to specify our beliefs about how the observations are related to the unknown parameters x.

3.5.1 The prior as a penalty term

Secondly, we can make use of the density $p(x)$. Observe the following:

$$p(x|A) \propto p(A|x)p(x) \propto \frac{1}{Z}\prod_i \exp(-\|A_i - x\|^2) = \frac{1}{Z}\exp(-\sum \|A_i - x\|^2)p(x).$$

If we now let $p(x) = \exp(-\lambda h(x))$, we have:

$$p(x|A) \propto \exp(-\sum \|A_i - x\|^2)\exp(-\lambda h(x))$$

and finally

$$-\log p(x|A) \propto \sum \|A_i - x\|^2 + \lambda h(x).$$

Here h is a penalty term, i.e. a function that is monotically increasing and bounded from below by 0, and $\lambda > 0$. If we choose h to represent some sort of constraints, then $e^{-\lambda h(x)}$ will represent the prior probability of the parameters being x. For values that violate the constraints, $h(x) > 0$, meaning that this probability drops for such values. The probability drops faster the larger λ is and the more we violate the constraints.

Thus, adding a penalty term view to constraints is equivalent to specifying a prior distribution for the parameters, i.e. the optimisation and probabilistic viewpoints are mathematically equivalent.

3.5.2 The prior as a projection

We now consider the case where x is a function g of some parameters θ. We write

$$p(x|\theta) = f_{\mathfrak{P}}(x - g(\theta)),$$

where $\mathfrak{P}$ is the density of a zero-mean distribution. We consider first the case where it is equal to the delta function, i.e.

$$p(x|\theta) = \delta(x - g(\theta)),$$

with $p(x)$ having singular density at $x = g(\theta)$. This means that the relationship between θ and x is determined solely by the deterministic function g, the projection. Now, let us go back to determining the distribution of x given our data. We have

$$p(x|A) \propto p(A|x)p(x) = p(A|x)\int p(x|\theta)p(\theta)d\theta.$$

For our choice of distribution, we have:

$$p(x) = \int \delta(x - g(\theta))p(\theta)d\theta.$$

We set $\theta = g^{-1}(t) = h(t)$ to obtain

$$p(x) = \int \delta(x - t)p(h(t))\frac{dh(t)}{dt}dt$$

Let's say we have a uniform $p(\theta)$:

$$p(x) \propto \int \delta(x - t)\frac{dh(t)}{dt}dt = g(\theta_0)\frac{dh(t)}{dt}\bigg|_{t=g(\theta_0)}$$

Also, equivalently,

$$p(x) \propto \int \delta(x - g)\frac{1}{\frac{dg}{d\theta}}dg = g(\theta_0)\left(\frac{dg}{d\theta}\right)^{-1}\bigg|_{\theta=\theta_0}$$

where $g(\theta_0) = x$.

Further on, we are going to delve deeper into equivalences. As it turns out, estimation, control and minimisation are all intrinsically related, in the sense that they are different views of the same problem. The probabilist framework is a good, general formal way to represent our beliefs about reality and to represent knowledge. Although the differences between the different views are only conceptual, in some cases some problems are easier to tackle than others in different frameworks.

4 Control

In this section we shall describe how to control a *known* system $\partial y/\partial t = f(v, y)$, where y is the system's state and v is some controlling input. In the discrete case the system can be defined as: $dy = f(v, y)dt$, i.e.

$$y_{t+dt} \approx y_t + f(v_t, y_t)dt$$

The classic control problem consists of choosing the control v such that we are close to some desired state. However this view has a few problems, such as defining closeness, and the need to impose additional constraints, such as constraints on time taken and energy consumed. Sometimes even defining the desired state might be difficult. More generally, however, we can view control problems as the set of problems whereby we select controls v such that some particular cost is minimised. This means that we can use optimisation techniques to solve control problems. For the case when it is possible to define a desired state, the cost will be simply some measure of the distance from the current to the desired state[5]. We will now take the braking control cases again as an example.

4.1 Braking example

Previously, we wanted to select a value of braking input such that we stop after s meters. In our previous exaple we could calculate this directly. Alternatively, however, we can formulate it as an optimisation problem by saying that we wish to find a control v such that $E[(s - \hat{s})]$ is minimised, where $\hat{s}$ is the actual stopping distance when we apply the control v.

According to (4) our model predicts a stopping distance of $s' = \frac{u^2}{2(ax+b)}$. We will minimise the sample error mean:

$$C = \sum_i C_i = \sum_i \left(s_i - \frac{u_i^2}{2(ax_i + b)}\right)^2.$$

[5]Which is why a lot of control problems can be formulated as shortest-path problems.

The question is, what should x_i be? It could be many things: A fixed value, a different, independent value for every different desired s_i, or some kind of smooth function depending on s_i. In all of those cases, it can be said that x_i will be a function of s_i and possibly other variables. In the following we shall be writing ∇x_i to indicate the gradient of x_i with some parameters. (If there are no other parameters, then $\nabla x_i = 1$.)

By taking the derivative

$$\begin{aligned}\nabla C_i &= -u_i^2\left(s_i - \tfrac{u_i^2}{2(ax_i+b)}\right)\nabla_{x_i}\tfrac{1}{ax_i+b}\nabla x_i \\ &= u_i^2\left(s_i - \tfrac{u_i^2}{2(ax_i+b)}\right)\tfrac{a}{(ax_i+b)^2}\nabla x_i.\end{aligned}$$

Now it is possible to apply gradient methods to optimise for some x that will minimise the average squared error between the desired and actual stopping distance. Probably the most convenient representation is one close to the inverse solution (6), such as for example, the following 2-parameter model:

$$x = \frac{\theta_1}{a}\left(\frac{u^2}{2s} + \theta_2 - b\right).$$

4.2 Trajectory optimisation

Trajectory optimisation is the problem of finding the trajectory of a particular car through the racing course so that the total time is minimised. It is a constrained optimisation problem and intuitively has a solution which should depend on the traction of different parts of the track, the track's inclination, the acceleration and braking power of the car and inherent delay between changes in control input due to driver reaction times and inertia of components. In this case however we will be concentrating on minimising a surrogate cost, the squared lateral acceleration of a car driving with constant speed on the trajectory. This is much simpler to optimise, since it does not take into account neither the car's nor the track's characteristics. In practice, it gives a good trajectory for cars with a very high horsepower.

4.2.1 Trajectory representation

At each point i in the track, the trajectory passes from the points $p(i) = l(i)w(i) + r(i)(1 - w(i))$. The values $l(i), r(i) \in \mathbb{R}^2$ represent the coordinates of the left and right points of the current track segment, while the parameter $w(i) \in \mathbb{R}$ represents how close to the left edge of the track the trajectory should be. We also say that the movement direction vector is

$$u(i) = \frac{p(i) - p(i-1)}{\|p(i) - p(i-1)\|},$$

which is a unit vector that describes the direction of movent, while the lateral acceleration is

$$a(i) = \frac{u(i) - u(i-1)}{\|p(i) - p(i-1)\|},$$

Thus the lateral acceleration vector describes the amount by which

4.2.2 The cost function

Because we have chosen to look at direction vectors, we can try and minimise is the average square of the lateral acceleration. This is a quite natural quantity to minimise and has the advantage that it does not rely on knowledge of the car.

$$C = \sum_i^n \|a(i)\|^2.$$

4.2.3 The gradient

If we use gradient descent to minimise this cost, then we obtain

$$\nabla_{w(i)} C = \sum_{j}^{n} \nabla_{w(i)} p(j) \nabla_{p(j)} u(j) \nabla_{u(j)} a(j) \nabla_{a(j)} C$$

This is not as complex as it seems. The term $w(i)$ appears only in the expressions containing $p(i)$, so it appears in $u(i)$ and $u(i+1)$, and thus also in $a(i), (a(i+1), a(i+2)$. This means that the full gradient is

$$\nabla_{w(i)} C = \sum_{j=i}^{i+2} \nabla_{w(i)} a(j) \nabla_{a(j)} C$$

We will take the elements one by one.

$$\nabla_{w(i)} p(i) = l(i) - r(i);$$

Let $d = p(i) - p(i-1)$. Then,

$$\begin{aligned}
\nabla_{p(i)} u(i) &= \nabla_{p(i)} d \; \nabla_d \frac{d}{\|d\|} \\
&= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \|d\| + d_x/\|d\| & d_x d_y/\|d\| \\ d_x d_y/\|d\| & \|d\| + d_y/\|d\| \end{bmatrix} \frac{1}{\|d\|^2}
\end{aligned}$$

Then

$$\nabla_{u(i)} a(i) = I,$$

the identity matrix. Finally,

$$\nabla_{a(i)} C = 2a(i).$$

So, the cost derivative (dropping the is for simplification)

$$\begin{aligned}
\frac{\partial a(i)}{\partial w(i)} \frac{\partial C}{\partial a(i)} &= (l - r) \frac{1}{\|d\|^2} \begin{bmatrix} \|d\| + d_x/\|d\| & d_x d_y/\|d\| \\ d_x d_y/\|d\| & \|d\| + d_y/\|d\| \end{bmatrix} 2a \\
&= \frac{2}{\|d\|^2} \begin{bmatrix} l_x - r_x & l_y - r_y \end{bmatrix} \begin{bmatrix} \|d\| + d_x/\|d\| & d_x d_y/\|d\| \\ d_x d_y/\|d\| & \|d\| + d_y/\|d\| \end{bmatrix} \begin{bmatrix} a_x \\ a_y \end{bmatrix} \\
&= \frac{2}{\|d\|^2} \begin{bmatrix} (l_x - r_x)(\|d\| + d_x/\|d\| + d_x d_y/\|d\|) \\ (l_y - r_y)(d_x d_y/\|d\| + \|d\| + d_y/\|d\|) \end{bmatrix}' \begin{bmatrix} a_x \\ a_y \end{bmatrix} \\
&= a_x (l_x - r_x)(\|d\| + d_x/\|d\| + d_x d_y/\|d\|) \\
&+ a_y (l_y - r_y)(\|d\| + d_y/\|d\| + d_x d_y/\|d\|).
\end{aligned}$$

The other terms are similarly derived. Note that this cost function suffers from the problem that it does not take into account the constraint that $w(i) \in [0, 1]$. There are, as we previously saw, three ways to fix that. Through a projection, a penalty, or a barrier. In this case we're going to be using a combination of a barrier term and a projection so that we always have a viable solution. The penalty term is

$$C_p = \max(0, c(w(i) - (b))^2, c(w(i) - (1 - b))^2),$$

where $c > 0$ and $b \in (0, 0.5)$ determines at which point from the boundary the penalty term is applied.

4.2.4 Results

A gradient descent method was used, with an incremental Gauss-Newton method (using a λ of 0.9). The step size was initialised at $a^0 = 0.01$ and the following step-size reduction rule was used

$$a^{k+1} = \begin{cases} a^k & \text{if} \nabla C(w^{k+1}) d^k \leq 0, \\ \beta a^k & \text{otherwise.} \end{cases}$$

with $\beta = 0.9$. While not strictly necessary, the above modifications are essentials for fast convergence. This can be provided by to some extent by applying either method on its own. The method was applied on a track consisting of 132 discrete segments, and thus an equal number of parameters. The resulting trajectory is shown below:

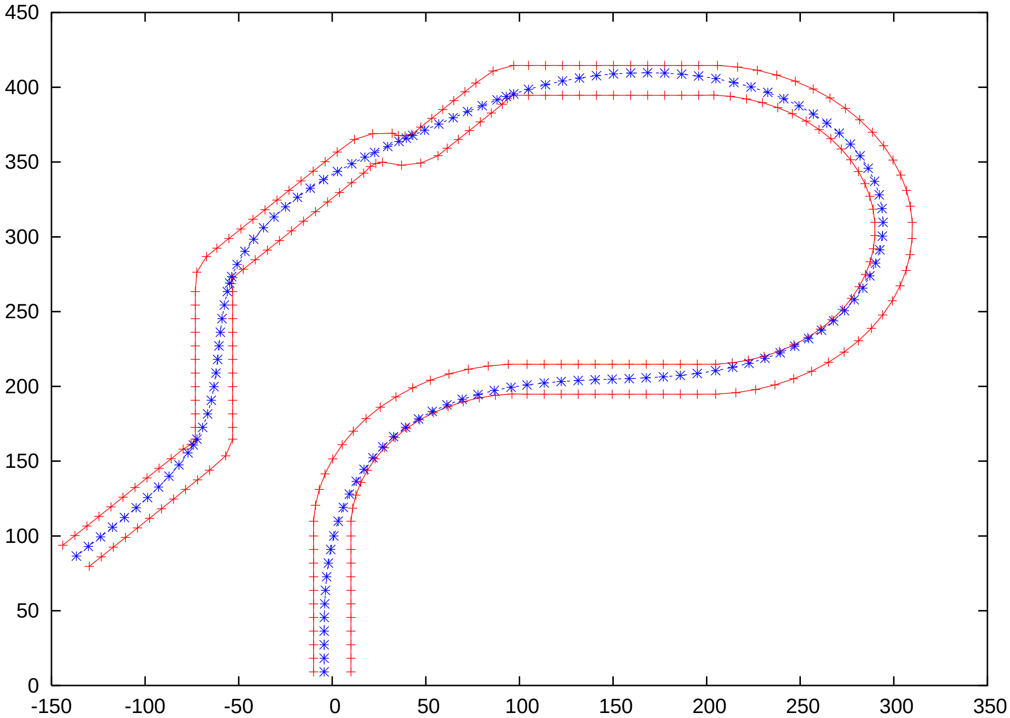

Figure 3: Optimal trajectory calculated by minimising the mean square lateral acceleration that must be applied on an object moving with constant speed on this trajectory.

For larger tracks, or tracks that are split into more segments, it will be advantageous to use a method whereby the positioning for each segment is determined by many parameters and where each parameter affects the positioning for many segments, with the aim being to have a small total number of parameters. A possible model would be

$$w(i) = \sum_{j}^{m} K_{ij} x(j),$$

where there is a total of m parameters and K is some fixed matrix. For the case reviewed before, $m = n$ and K is the identity matrix. Possible options for the K matrix are a random sparse matrix, or a kernel with some spatial signficance.[6]

[6]A simple such kernel arises naturally by associating each parameter $x(j)$ with a position in the track and then have

5 Stochastic control

Similarly to the previous section, we are solving an optimisation problem. However, this time we are hampered by a lack of knowledge. We need to both optimise the control parameters according to our estimates, and to improve our estimates. In the stochastic control case we are interested in both model-based and model-free approaches. In model-free approaches, one is merely interested in inferring the best control input from the system's observable state. In model-based approaches, we infer the best control input from the system's observable state and from our estimated model.

The major problem with stochastic control is that initially the model is inaccurate. This leads to a need to balance the behaviour of the controller between two modes: (a) an exploration mode, where the system is controlled in order to improve our model and (b) an exploitation mode, where the system is controlled in order to minimise our cost function assuming our current model is correct. . The exploration is usually achieved by inducing some sort of stochasticity in the system, through the control inputs, although in some cases the inherent stochasticity of the environment is sufficient.

A good overview of the classical theory of optimal control is given in Stengel (1994), while Bertsekas (2001) focuses more on modern approximations of the classic Dynamic programming approach (Bellman, 1957). Talking about all the aspects of stochastic control is currently beyond the scope of this tutorial, however a gradient-based technique for solving this kind of problem is given by Baxter and Bartlett (2000) in the context of policy-gradient methods in reinforcement learning (Sutton and Barto, 1998).

A Notation

$E[\cdot]$ denotes the expectation operator. $E[X|Y]$ denotes the expectation of random variable X given variable Y. We will use this notation for our calculated expectation of X given some measurements Y. Frequently this is written in shorthand as $\hat{E}[X]$. When we are sampling X under a distribution π we will alternatively use $E[X|\pi]$ or $E_\pi[X]$, depending on context.

References

Jonathan Baxter and Peter L. Bartlett. Reinforcement learning in POMDP's via direct gradient ascent. In *Proc. 17th International Conf. on Machine Learning*, pages 41–48. Morgan Kaufmann, San Francisco, CA, 2000. URL `citeseer.nj.nec.com/baxter00reinforcement.html`.

Richard Ernest Bellman. *Dynamic Programming.* Princeton University Press, 1957. Republished by Dover in 2004.

Dimitri P. Bertsekas. *Nonlinear Programming.* Athena Scientific, 1999.

Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control.* Athena Scientific, 2001.

Stephen Boyd and Lieven Vandenberghe. *Convex Optimization.* Cambridge University Press, March 2004. ISBN 0521833787. URL `http://www.stanford.edu/ boyd/cvxbook/`.

Eric Espié, Christophe Guionneau, Bernhard Wymann, Christos Dimitrakakis, Charalampos Alexopoulos, Patrice Espié, Eliam, Olaf Sassnick, Thierry Thomas, Eugen Treise, Andrew Sumner, Christophe Macours, Rémi Coulom, Andrea Alfieri, Asdas Asda, Patrick Wisselo, Bernhard Kaindl, Gernot Galli, Henrik Enqvist, Per Oyvind Karslen, Matthias Saou, Neil Winton, Butch K/Cendra, Vincent Moyet, Jens Thiele, Paul Bain, and Jean-Christophe Durieu. TORCS, the open racing car simulator, 2006. URL `http://www.torcs.org`.

the kernel related to the track-based distance between the jth parameter and the ith track segment. For example, if $l_x(j)$ is the distance from the startline of parameter $x(j)$ and $l_w(i)$ is the distance from the startline of segment i, then a suitable kernel would be $Kij = \exp(-\beta|l_w(i) - l_x(j)|)$.

Robert F. Stengel. *Optimal Control and Estimation.* Dover, second edition, 1994.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction.* MIT Press, 1998.