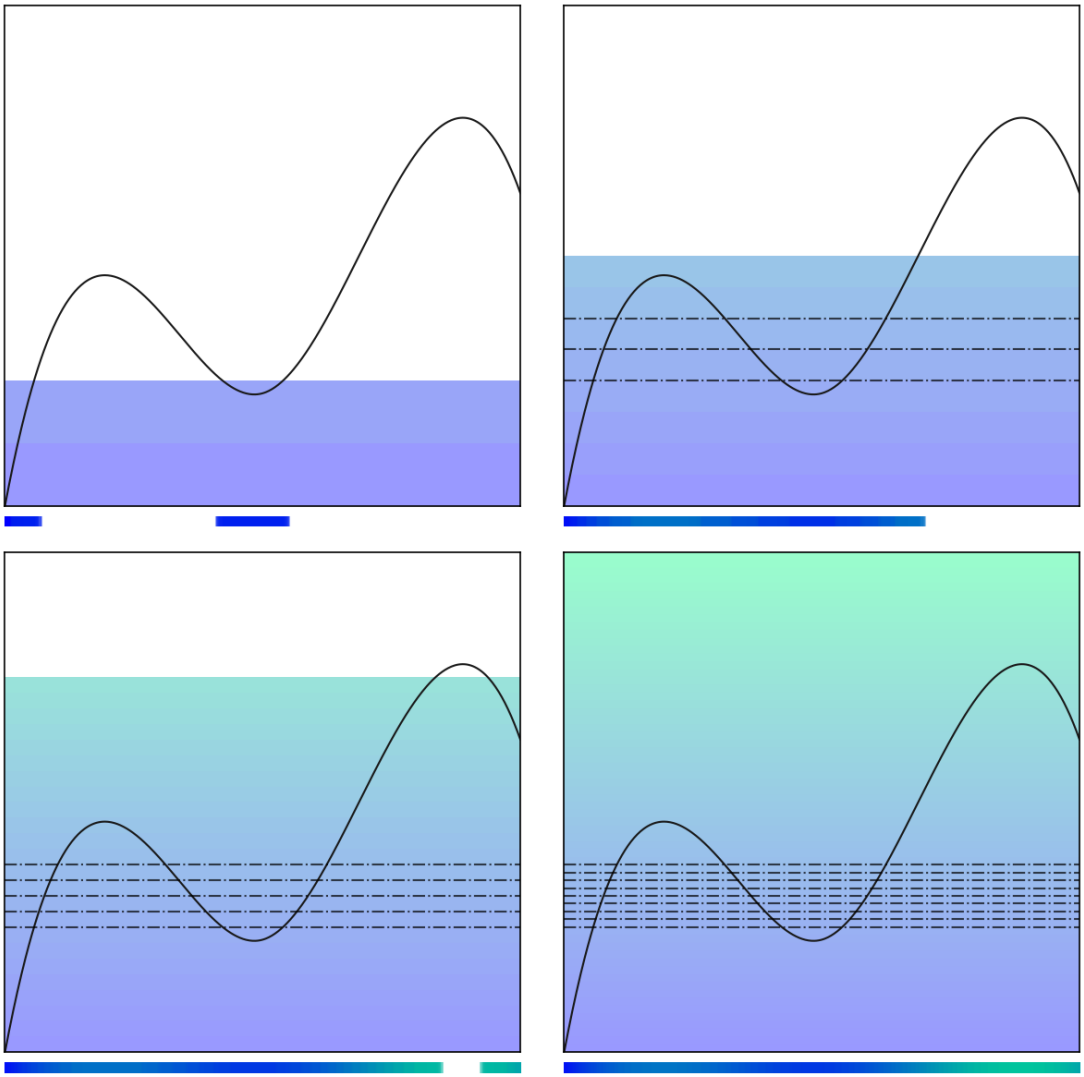
The Lebesgue Integral

From σ-algebras to simple functions;
building up to defining the Lebesgue Integral

Author: Daniel Ajibola Adelodun
Supervisor: Professor Gromov
Third Year Project

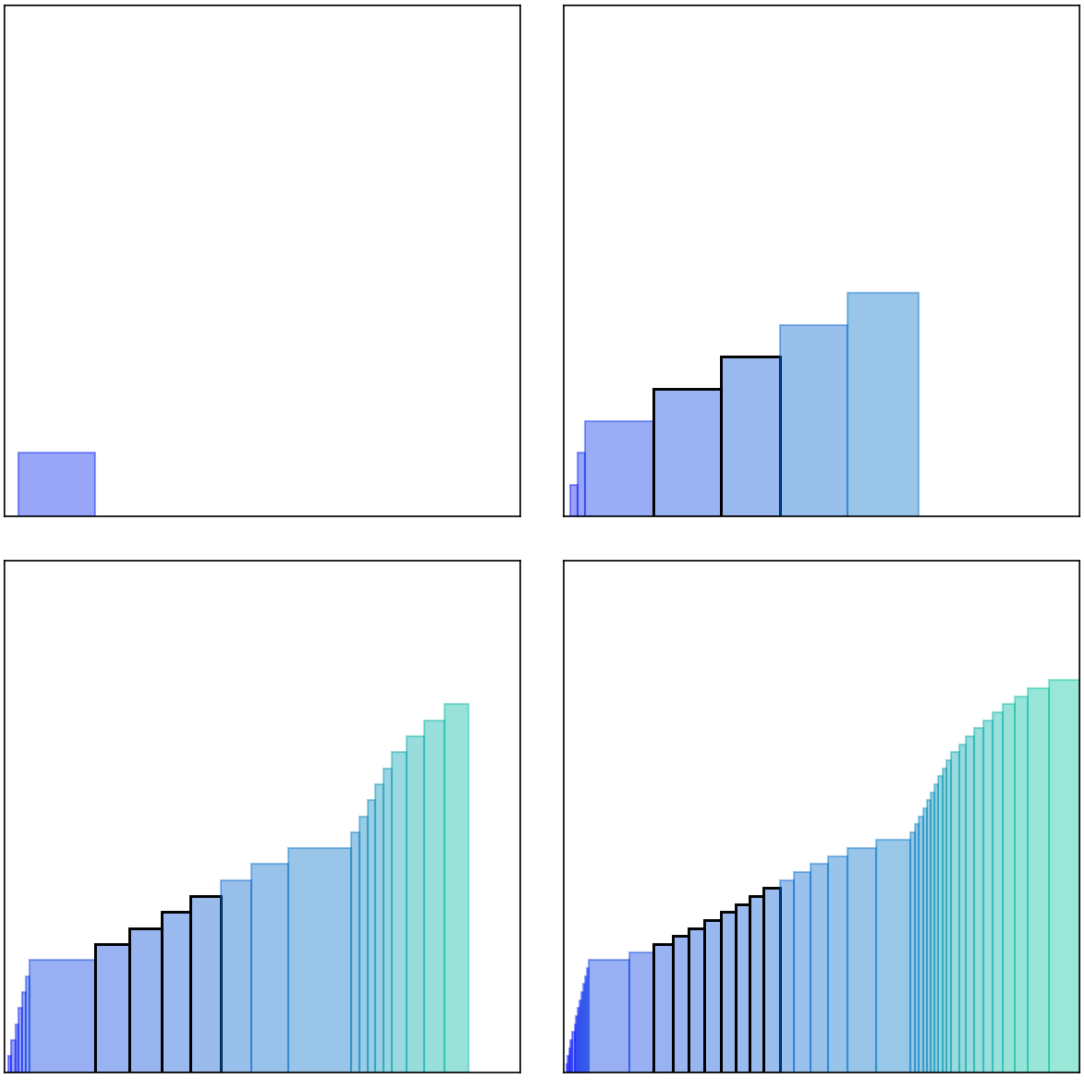
22nd April 2019



Abstract

In this paper we work our way up to defining the *Lebesgue Integral* by introducing relevant definitions from Set Theory and Measure Theory. We make comparisons between the *Lebesgue Integral* and the *Riemann Integral*, noting the relative strengths and weaknesses of each.

I will also touch on other uses of Measures Theory, including it's use in Axiomatic Probability Theory in the definition of a *Random Variable*. I do my best to provide both intuitive explanations, as well as precise/concise definitions... At least that was the plan, had I started sooner.



Contents

1 Introduction

1.1 The Definition

The aim of this paper is to eventually define the Lebesgue Integral - so lets try to do that...

Definition 1.1 ***(Lebesgue Integration).*** Given

- A *Measure Space*, $(X, \mathcal{A}, \mu)$,
- An $(\mathcal{A}/\bar{\mathcal{B}})$*-Measurable Function*, $u\colon X \to \bar{\mathbb{R}}$,
- $u^+, u^- : u = u^+ - u^-$ and $u^{\pm}(x) \geq 0 \ \ \forall x \in X$,
- $S^+, S^- \subset \mathcal{E}^+_\mu : s \in S^{\pm} \Leftrightarrow s \leq u^{\pm}$, and $s \in \mathcal{E}^+_\mu$.

We define the μ*-Integral* of u — $\int u\, d(\mu)$ — in terms of the μ*-Integrals* of the positive functions u^+ and u^-.

The μ*-Integral* of a *positive* function (so including both u^- and u^+ but not necessarily u itself) is defined as

$$\int u^+ \, d(\mu) := \sup\{ I_\mu(s) : \quad s \in \mathcal{E}^+ \quad \text{and} \quad s \leq u^+ \}, \tag{1.1'}$$

and going back to our more general u,

$$\int u \, d(\mu) := \int u^+ \, d(\mu) - \int u^- \, d(\mu). \tag{1.1}$$

If the domain of our function is $\mathbb{R}^n$, with the 'standard' n-dimensional *Borel* σ*-Algebra* and *Lebesgue Measure* on it — i.e. $(X, \mathcal{A}, \mu) = (\mathbb{R}^n, \mathcal{B}, \lambda^n)$ — then this becomes the λ^n*-Integral,* (read as *Lebesgue Integral* instead of lambda integral — or we just call it the *Integral*, confusingly) and we write

$$\int u \, d(\lambda^n) = \int u(x) \, d(\lambda^n) =: \int u(x) \, dx. \tag{1.1''}$$

* * *

1.2 Why Not Just Stop Here?

That's quite a long definition, one which raises more questions than it answers. The biggest issue here is that, even assuming you've taken a course in Analysis, there are many undefined terms;

- What are *Measures* and σ*-Algebras* And what is a *Measure Space*?
- What is the *Borel* σ*-algebra*
- What is a *Measurable Function*? What is a *Simple Function*?
- What is I_μ and why can it seemingly only be used on positive *Simple Functions*?

That last bullet point touches on the second key issue with this definition; why. Why any of this. What about this definition of integration makes it any better than the *Riemann Integral*? Do they even describe similar things? This is not *at all* immediately obvious, since the Riemann Integral is defined on functions $f\colon Y \to \mathbb{R}$ where Y is some compact[1] subset of $\mathbb{R}$, and this Integral is defined on...well, *Measurable Functions* on any set at all, apparently.

Even if we can decipher this definition, and then somehow convince ourselves it's compatible with Riemann's definition, it still seems a little convoluted - like, what about *Simple Functions* is so special that the Integrals of all other functions are defined in terms of them. Why is the Integral of a function defined in terms of the Integrals of positive functions, while the Integral of a positive function is defined in terms of positive simple ones?

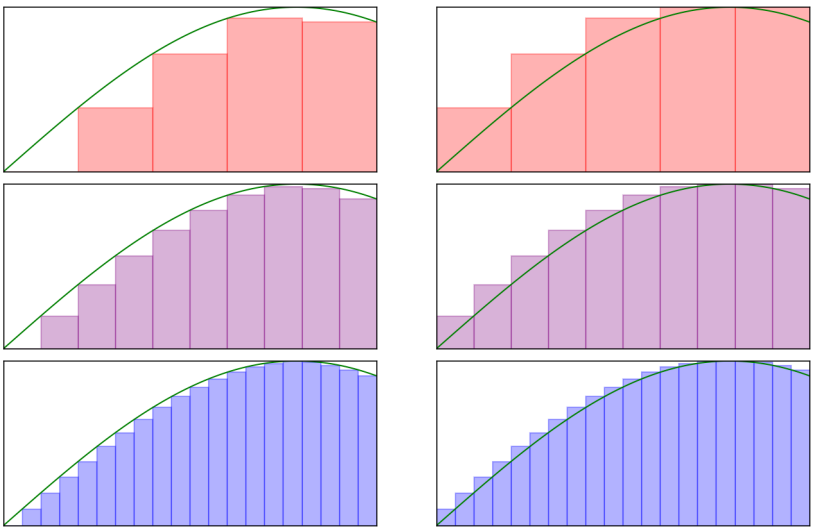
Interestingly (to me, at least) the *Riemann Integral* is defined in a similarish way - we find the upper Riemann Integral of a function by finding all the *Step Functions* which bound the function from above, define/find the *Integrals* of these using their 'steppyness', then take the infinium - we do the same thing from below, and then if these limits coincided then we call that the Riemann Integral. In order to understand the motivations behind Lebesgue Integration better, let's take a slightly closer look at the similarities and differences between these two.

[1] any closed and bounded interval - so an interval in the form $[a, b]$

2 Riemann vs. Lebesgue

2.1 The Motivation

Our goal when we integrate a function is to find the amount of 'space' between the graph and x-axis (or the Abscissa if you're fancy). It's pretty difficult to say how much space an arbitrary curve takes up, but we can work out the area of rectangles very easily — by calculating the *base* × *height*. Riemann integration takes advantage of this, and defines the integral of a function $f : \mathbb{R} \to \mathbb{R}$ by first bounding it from above with rectangular functions and finding the smallest of these areas (which would be an upper bound for the area of the function), then we bound the function from below and find an lower bound, then we hope that these two bounds match up.



(a) Bounding from Below. (b) Bounding from Above.

Figure 2.1: *Approximating Area with Step Functions.*

If the domain of our function (the 'x-axis'), is $\mathbb{R}^2$ instead of just $\mathbb{R}$, we'd actually calculating a volume by finding the 'space' between the graph and the axis. Curves are bad enough, but it's even harder to find the volume under a surface. Basic Riemann Integration doesn't apply to these functions — but if we wanted, we could make a similar observation and simplification. The analog to a *rectangle* in $\mathbb{R}^3$ is a cuboid, and we can think of its size as being calculated in much the same way as in $\mathbb{R}^2$; the size of the base (which is now an area, instead of a length) × the height. We could continue like this for all the Euclidean spaces, $\mathbb{R}^n$. It'd be nice if there was a way to generalize this, and take advantage of it, so that we could integrate any function where we can make sense of *'the size of the base.'* With this is mind, lets clear up some of the terms we are going to use:

Lebesgue integration is the method of integration defined above. Just like Riemann Integration, it is used to find the space between the graph of a function and the domain. However, unlike Riemann, the domain can be any set at all. Thinking back to rectangles and cuboids and *base* × *height*, if we wanted make sense of what it means for there to be space between the graph and the x-axis, we should be able to make sense of the 'size' of parts of the domain, so that we can figure out how big our base is and multiply that by the height...

The Lebesgue Measure seems then like it'd be a way to assign sizes to, or 'measure' sets in any domain; but it is not! It's a term used *specifically* when we give sizes to the subsets of $\mathbb{R}^n$, and those sizes 'measures' coincided with our usual idea of the size of a set in $\mathbb{R}^n$ — i.e. it is used when we are considering functions $\mathbb{R}^n \supset Y \to \mathbb{R}$, and in $\mathbb{R}$ the measure (length) of the interval $[1, 0]$ is 1, in $\mathbb{R}^2$ the measure (area) of the rectangle $[1, 0] \times [1, 0]$ is 1, in $\mathbb{R}^3$ the measure (volume) of the cube $[1, 0] \times [1, 0] \times [1, 0]$ is 1... etc.

The Lebesgue Integral is what we get when we do Lebesgue Integration on a set with the Lebesgue Measure on it. i.e. it's Integral of normal functions $\mathbb{R}^n \supset Y \to \mathbb{R}$. This is the (basically) the same domain as the Riemann integral and so we'd want to check that the Lebesgue Integral and the Riemann Integral match up — and then see if the Lebesgue integral is better in some way.

Already, we can see an improvement over Riemann Integration; Lebesgue Integration is defined on functions with any domain (kind of), and even the more restricted Lebesgue Integral can be calculated directly for any function with a domain in $\mathbb{R}^n$ — so less double/triple integration nonsense. To see another benefit of the Lebesgue Integral over Riemann, let's look at the canonical example of a function for which the Riemann Integral fails; $f : \mathbb{R} \to \mathbb{R}, \quad f(x) = \mathbb{1}_{\mathbb{Q}}(x)$ (Figure 3.1). This is also an example of a *Simple Function* since it is just the Indicator Function of a *Measurable* set, but lets not get ahead of ourselves... First, we should define the *Riemann Integral*.

2.2 The Definition of the Riemann Integral

The definition of the Riemann Integral involves a few steps, and so it's going to take a little patience. The first step is defining a *Step Function* - no pun intended.

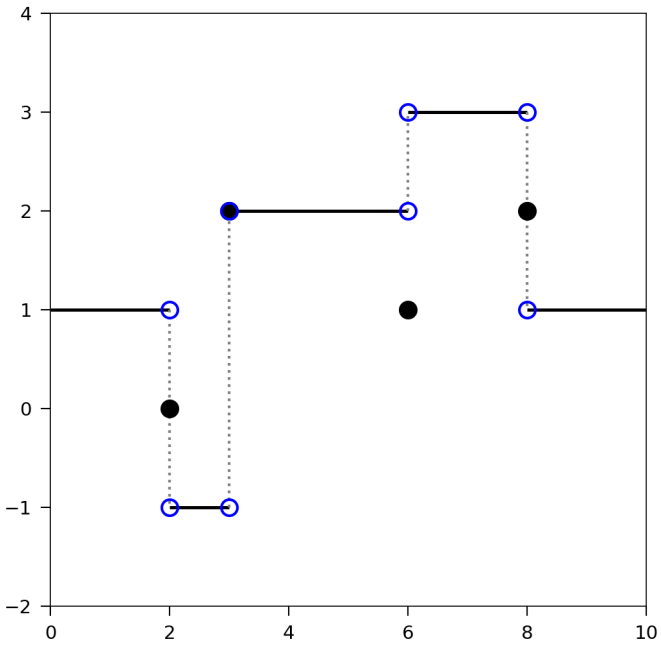


Figure 2.2: *A Step Function*

A function is called a *Step Function* if and only if there exists some finite sequence of points such that the function is constant between any two adjacent points. So, for the function with the graph shown in Figure 2.2, the points $\{0, 2, 3, 6, 8, 10\}$ are a finite sequence which make this a step function. We will often index the elements of sequence using n, and label each element x_n. We can call sequence this a *'representation'* of the function, and we can call the sequence of constant heights taken just *before* the n^{th} point in our representation the corresponding *'height sequence'*. The *height sequence* along with the representation fully describe the function.

Note two important points;

- We could have picked $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$ to be our representation, since the function also happens to constant between any two adjacent integers.
- It doesn't really matter what the function is *at* the points x_n — we only care whether the function is constant *between* our given points. In our graph, 2, 3, 6, and 8 are discontinues, but that doesn't matter since they're also in our sequence of points.

The formal definition is as follows;

Definition 2.1 ***(Step Function).*** Given ϕ: $[a, b] \to \mathbb{R}$, ϕ is a *Step Function*

$:\Leftrightarrow \ \phi \in \mathcal{S}([a, b])$

$:\Leftrightarrow \ \exists N \in \mathbb{N} \ : \ \exists X = (x_n)_{n=0}^{N} : a = x_0 < x_1 < \ldots < x_{n-1} < x_n = b,$ and $\forall m \in \mathbb{N}_{<N}$ we have $\exists! c_m \in \mathbb{R} : \forall z \in (x_{m-1}, x_m)$ we have $\phi(z) = c_m$.

* * *

This also defines *the set of all step functions on* $[a, b]$ as $\mathcal{S}[a, b]$. This kind of thing is done a lot in analysis — we often want to look at the set of all things which satisfy some property. We normally don't have to *find* them all — we just need a way to represent the full set. The fact that it'd be near impossible to try to imagine the set $\mathcal{S}[a, b]$ is irrelevant — it's just important we know it exists. The sequence X is a representation.

Next, we want to define what it means to *Integrate* a step function. Remember, the point of working with step functions is because it's easy to find the area of rectangles, and integration is all about finding areas. So where are the rectangles?

As is hinted at by Figure 2.2, we can make the bases of the rectangles the widths of the intervals (x_{n-1}, x_n), and the heights the constant c_ns which values in these interval take. This is made even clearer by Figure 2.3, where we can also see that the sometimes our heights will have negative values, and that's okay — these areas are just taken away from the total area instead of added.

What's not okay is that any one step function may have different sequences which we can use to prove that it's a step function — or in other words, the same step function may have different but equally valid representations. We want to define the integral of a step function in terms it's representation, but how do we know that different representations of the same function produce the same integral? Figure 2.3 demonstrates that, given the representations $\{0, 2, 3, 6, 8, 10\}$ and $\{0, 1, 2, \ldots, 10\}$ of 2.2, we're fine, and the areas are equal. It turns out that this is the case in general — the integral of a step function is independent of the chosen representation.

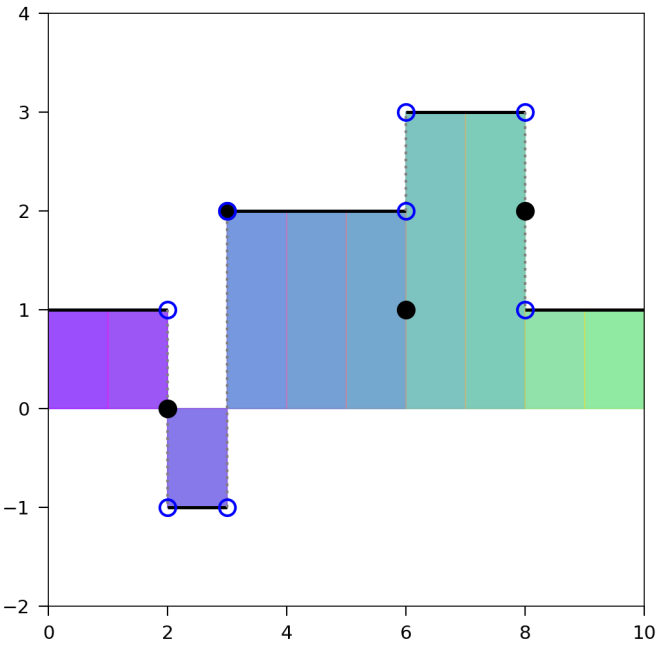


Figure 2.3: *Integral of 2.2*

Definition 2.2 ***(Integral of a Step Function).*** Given

- $\phi \in \mathcal{S}([a, b])$,
- $N \in \mathbb{N}$ and the representation $(x_n)_{n=0}^N$ of ϕ,
- the corresponding *height sequence* $(c_n)_{n=1}^N : \phi(x) = c_n \quad \forall x \in (x_{n-1}, x_n)$,

then the *Integral* of ϕ as a function of x on $[a, b]$ is

$$\begin{aligned} &\int_a^b \phi(x)\,dx \\ &:= \sum_{i=1}^{N} c_i(x_i - x_{i-1}). \end{aligned} \tag{2.1}$$

* * *

To show that the Integral is independent of the choice of representation, we'd need to show that if we had two arbitrary representations $(x_n)_{n=0}^N$ and $(x'_m)_{m=0}^M$ with *height sequences* $(c_n)_{n=1}^N$ and $(c'_m)_{m=1}^M$ resp. then,

$$\sum_{i=1}^{N} c_i(x_i - x_{i-1}) = \sum_{j=1}^{M} c'_j(x'_j - x'_{j-1}).$$

The idea behind why this is true is the following;

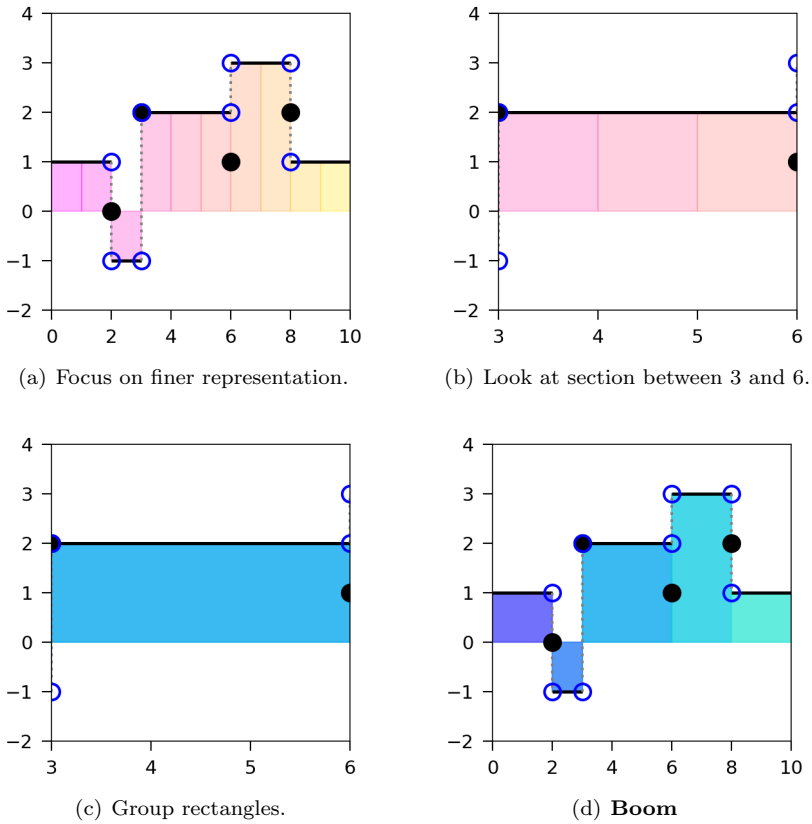
× Assuming first that one sequence is a subsequence of the other, we will focus on the representation that is more 'fine' — i.e. the one that has smaller width rectangles; ones which are divisions of the rectangles in the 'coarser' representation.

× We look at it in sections; with the sections being the steps of the coarser representation. So, in 2.2, we'd look at $\{0, 1, 2 \ldots, 10\}$ between 0 and 2, then 2 and 3, then 3 and 6 ... etc.

× The two representations are for a single the step function; and we use this group adjacent rectangles with the same height. $\phi(x)$ must be constant in the section we are looking at, and so we know that all the rectangles here have the same height

× Then we can use the distributive property of multiplication to view the smaller width rectangles of the same height as one big rectangle with the width of the whole section. And boom, we're done, since those big rectangles are exactly the ones which you would use when integrating using the coarser representation.

⊗ If neither representation is contained within the other, then we can just create a third representation which is the union the first ones. Both original sequences will be coarser than this new one, and so we can use the same reasoning as before, plus transitivity and symmetry, to prove the integrals are the same.



(a) Focus on finer representation.

(b) Look at section between 3 and 6.

(c) Group rectangles.

(d) **Boom**

Figure 2.4: *Different representations of the same function.*

Proposition 2.1. *Given,*

- $\phi \in \mathcal{S}([a, b])$,

- $N, M[= N'] \in \mathbb{N}$ *and the representations* $(x_n)_{n=0}^{N}$ *and* $(x'_{m[=n']})_{m=0}^{M}$ *of* ϕ,
- *The respective height sequences* $(c_n)_{n=1}^{N}$ *and* $(c'_m)_{m=1}^{M}$ *of these representations,*

then

$$\sum_{i=1}^{N} c_i(x_i - x_{i-1}) = \sum_{j=1}^{M} c'_j(x'_j - x'_{j-1}). \tag{2.2}$$

Proof.
Case 1: $\{x_n; n \in \mathbb{N}_{n<N}\} \subset \{x'_m; m \in \mathbb{N}_{m<M}\}$:

$$\begin{aligned} & \{x_n; n \in \mathbb{N}_{<N}\} \subset \{x'_m; m \in \mathbb{N}_{<M}\} \\ \Leftrightarrow \quad & \exists J = (j_n)_{n=0}^{N} \\ & : 0 = j_0 < j_1 < \ldots < j_{n-1} < j_n = m \text{ and } k \in \mathbb{N}_{<N} \text{ we have } x_k = x'_{j_k} \end{aligned}$$

Hence,

$$\begin{aligned} \sum_{j=1}^{M} c'_j(x'_j - x'_{j-1}) &= \sum_{k=0}^{N} \sum_{j=j_{k-1}}^{j_k} c'_j(x'_j - x'_{j-1}) \\ &= \sum_{k=0}^{N} c'_{j_k} \sum_{j=j_{k-1}}^{j_k} (x'_j - x'_{j-1}) \\ &= \sum_{k=1}^{N} c_k(x_k - x_{k-1}) \end{aligned}$$

Case 2: Arbitrary representations $X = \{x_n; n \in \mathbb{N}_{<N}\}$ and $X' = \{x'_{n'}; n' \in \mathbb{N}_{<N'}\}$:

Let $X'' = \{x''_{n''}; x"_{n''} \in X \cup X'\}$.

$$\begin{aligned} & \{x_n; n \in \mathbb{N}_{n<N}\} \subset \{x''_{n''}; n'' \in \mathbb{N}_{<N''}\} \supset \{x'_{n'}; n' \in \mathbb{N}_{m<M}\} \\ \Rightarrow \quad & \sum_{X} = \sum_{X''} = \sum_{X'} \end{aligned}$$

□

Now we are ready to define the *Riemann Integral* of any function $f\colon [a,b] \to \mathbb{R}$ ... almost. We first need to define 2 separate upper and lower integrals, and then if they are the same, what we call that the *Riemann Integral* of the function. This is actually why $\mathbb{1}_{\mathbb{Q}}(x)$ is not *Riemann Integrable*; it's upper and lower integrals do not coincide.

To find the *Upper Riemann Integral* of f, first find *all* the step functions which bound it from above, then calculate their integrals using Definition 2.2 and then, finally, define the smallest of these integrals to be the Upper Riemann Integral. To find the *Lower Riemann Integral*, you do the same thing, but from below. The set of all Riemann Integrable functions on $[a,b]$ is $\mathcal{R}([a,b])$.

Figure 2.1 tries to illustrate this, but it's important to remember that we aren't looking at a sequence of step functions, we are looking at *all of them.* How to calculate/approximate integrals using sequences of [step] functions is a whole thing in itself — I wish I had started writing this sooner so that I could get into it because it's super interesting, but alas... Anyway, now we can define Riemann Integration for real:

Definition 2.3 ***(Riemann Integration).*** Given

- $f\colon [a,b] \to \mathbb{R}$,
- $S_l \subset \mathcal{S}([a,b]) : \psi \in S_l \Leftrightarrow \ \psi(x) \leq f(x) \quad \forall x \in [a,b]$,
- $S_u \subset \mathcal{S}([a,b]) : \phi \in S_u \Leftrightarrow \ \phi(x) \geq f(x') \quad \forall x' \in [a,b]$,

then the *Lower Riemann Integral* of f is

$$\underline{\int} f(x)\,dx := \sup\left\{\int_a^b \psi(x)\,dx; \ \ \psi \in S_l\right\} \tag{2.3}$$

and the *Upper Reimann Integral* is

$$\overline{\int} f(x)\,dx := \inf\left\{\int_a^b \phi(x)\,dx;\quad \phi \in S_u\right\}. \tag{2.4}$$

If (2.3) = (2.4), then f is *Riemann Integrable* the *Riemann Integral* of f as a function of x is

$$\int_a^b f(x)\,dx := \underline{\int} f(x)\,dx = \overline{\int} f(x)\,dx \tag{2.5}$$

* * *

3 Conclusion

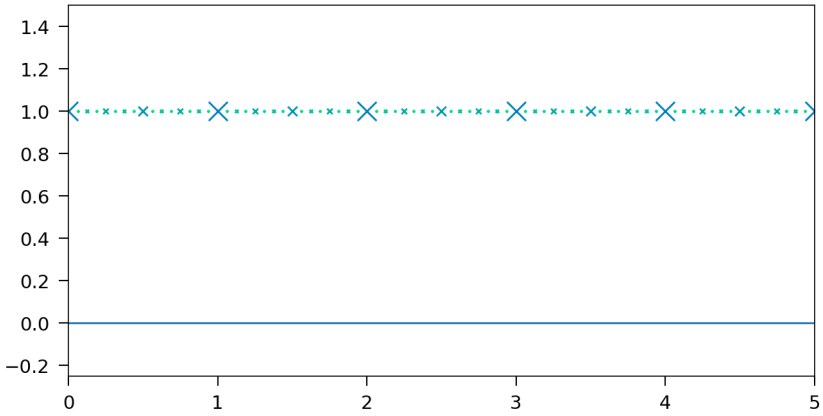


Figure 3.1: *An illustration of the indicator function $\mathbb{1}_{\mathbb{Q}}(x)$, showing a few rationals that are multiples of negative powers of 2.*

I have run out of time :(. So, my conclusion will be me starting why the Riemann Integral fails in a certain case, how the Lebesgue Integral improves on those shortcomings, and what I would have talked about had I started earlier.

When we try to integrate a the indicator function of the rationals — $\mathbb{1}_{\mathbb{Q}}(x)$ — we run into a problem. If we wanted to bound it from above with step functions, every possible interval we could pick will contain at least one rational. What this means is that, at a minimum, the function will be a constant one. More obviously, if we wanted to bound it from below, we'd end up with a constant zero.

Intuition says that if we *could* integrate this function, then it's integral would be very close to zero. Why? Because the set of rational numbers is so much smaller that $\mathbb{R}$, and so you'd expect the space underneath it to be much smaller space underneath (or really, not underneath, in this case) the reals. In fact, compared to the $\mathbb{R}$, the size rationals basically zero...

Lebesgue Integration works because it makes use of the branch of mathematics that formalizes the whole idea of what it means to find the 'size' of set. Had I started sooner, I would mentioned other uses of this branch of mathematics, since being to measure things is, obviously, very useful. Namely, I would have talked about it's use in the definition of a *Random Variable*.

Nevertheless, I have included a full list of definitions as an appendix to this paper, so we did in the end (literally) do what we set out to do; work our way up to defining the Lebesgue Integral.

Appendices

A List of Definitions

Here I list some definitions which I did not have time to work into the main body of the text. The notation here mostly follows the style documented at the Wikipedia page for mathematical notation. Sets are normally denoted with capitals, $A, B, C \ldots$, and sets of [sub]sets in calligraphic font $\mathcal{A}, \mathcal{B}, \mathcal{C} \ldots$ Notably, the power set $\mathcal{P}(X)$ of a set X follows this pattern.

Definition A.1 *(The Extended Real Line, $\bar{\mathbb{R}}$).*

$$\bar{\mathbb{R}} := \mathbb{R} \cup \{-\infty, +\infty\}$$

$$* \ * \ *$$

Definition A.2 *(σ-Algebra).* Given

- A set, X,
- A family of subsets of X, $\mathcal{A} \subset \mathcal{P}(X)$.

Then $\mathcal{A}$ is a *σ-Algebra* of $X :\Leftrightarrow$

$$\text{(i)} \ \ \emptyset \in \mathcal{A} \ ,$$

$$\text{(ii)} \ \ (A_j)_{j \in \mathbb{N}} \subset \mathcal{A} \Rightarrow \bigcup_{\mathbb{N}} A_j \in \mathcal{A},$$

$$\text{(iii)} \ \ A \in \mathcal{A} \Rightarrow X \backslash A[\equiv A^c] \in \mathcal{A}.$$

$$* \ * \ *$$

Definition A.3 *(Measurable Space).* Given

- A set, X,
- A *σ-Algebra* of X, $\mathcal{A}$.

Then

- *The ordered pair $(X, \mathcal{A})$ is a Measurable Space.*
- A set $\mathcal{A} \in \mathcal{A}$ *is Measurable.*

$$* \ * \ *$$

Definition A.4 *(The σ-Algebra Generated by $\mathcal{G}, \sigma(\mathcal{G})$).* Given

- A set, X,
- $\mathcal{G} \subset \mathcal{P}(X)$.

Then $\sigma(\mathcal{G})$ is the *σ-Algebra Generated by $\mathcal{G}$* $:\Leftrightarrow$

$$\text{(i)} \ \ \mathcal{G} \subset \sigma(\mathcal{G}),$$

$$\text{(ii)} \ \ \sigma(\mathcal{G}) \text{ is a } \sigma\text{-Algebra of } X,$$

$$\text{(iii)} \ \ \mathcal{A} \text{ is a } \sigma\text{-Algebra of } X \text{ and } \mathcal{G} \subset \mathcal{A} \Rightarrow \sigma(\mathcal{G}) \subset \mathcal{A}.$$

$$* \ * \ *$$

Definition A.5 *(The family of half open rectangles in $\mathbb{R}^n$, $\mathcal{J}^{o,n}$).*

$$\mathcal{P}(\mathbb{R}^n) \supset \mathcal{J}^{o,n} := \{(a_1, b_1] \times (a_2, b_2] \times \cdots \times (a_n, b_n]; \ \forall a_k, b_k \in \mathbb{R}\}$$

$$* \ * \ *$$

Definition A.6 *(The Borel σ-Algebra on $\mathbb{R}^n, \mathcal{B}(\mathbb{R}^n)$).* Fix $n \in \mathbb{N}$. Given the family of half open rectangles $\mathcal{J}^{o,n}$

$$\textit{The } \sigma\textit{-Algebra generated by } \mathcal{J}^{o,n} \textit{ is the Borel } \sigma\textit{-Algebra on } \mathbb{R}^n$$

* * *

Definition A.7 *(The Extended Borel σ-Algebra on $\bar{\mathbb{R}}^n, \bar{\mathcal{B}}(\bar{\mathbb{R}}^n)$).* Fix $n \in \mathbb{N}$. $S^* \in \bar{\mathcal{B}} :\Leftrightarrow$

$$\exists B \in \mathcal{B}(\mathbb{R}) \textit{ and } \exists S \in \{\emptyset,\ +\infty,\ -\infty, \{+\infty, -\infty\}\} : \quad S^* = B \cup S$$

* * *

Definition A.8 *(Measures).* Given

- A *Measurable Space*, $(X, \mathcal{A})$,
- A map, $\mu \colon \mathcal{A} \to \mathbb{R}$.

Then μ is a *Measure* on the *Measurable Space* $(X, \mathcal{A}) :\Leftrightarrow$

$$\text{(i)}\ \ \mu(\emptyset) = 0$$

$$\text{(ii)}\ \ A, B \in \mathcal{A};\ A \cap B = \emptyset \quad \Rightarrow \quad \mu(A \cup B) = \mu(A) + \mu(B)$$

* * *

Definition A.9 *(Measure Space).* Given

- A *Measurable Space*, X,
- A *Measure*, μ, on $(X, \mathcal{A})$.

Then

$$\textit{The ordered triple } (X, \mathcal{A}, \mu) \textit{ is a Measure Space.}$$

* * *

Definition A.10 *([Discrete] Probability Space).* Given

- $\otimes$ $(\omega_n)_{n\in\mathbb{N}} = \Omega$ [*for a Discrete Probability Space*],
- A *Measure Space*, $(\Omega, \mathcal{A}, p)$.

Then $(\Omega, \mathcal{A}, p)$ is a *[Discrete] Probability Space* and p a *Probability Measure* $:\Leftrightarrow$

$$p(\Omega) = 1.$$

$$\text{(ii*)}\quad \bigcup_{\mathbb{N}} \mu(\{\omega_n\}) = 1 \textit{ [Discrete].}$$

* * *

Definition A.11 *(The Lebesgue Measure λ^n).* Fix $n \in \mathbb{N}$. Given

- The *Measurable Space*, $\big(\mathbb{R}^n, \mathcal{B}(\mathbb{R}^n)\big)$,
- A *Measure*, λ^n, on $\big(\mathbb{R}^n, \mathcal{B}(\mathbb{R}^n)\big)$.

Then λ^n is the *Lebesgue Measure* on $\big(\mathbb{R}^n, \mathcal{B}(\mathbb{R}^n)\big) :\Leftrightarrow$

$$\forall X \in \mathcal{J}^{o,n}, \quad \lambda^n(X) = \prod_{k=0}^{n} (b_k - a_k).$$

* * *

Definition A.12 *(Measurable Mapping).* Given

- *Measurable Spaces*, $(X, \mathcal{A})$ and $(X', \mathcal{A}')$,

- A map, $T\colon X \to X'$.

Then T is an *$(\mathcal{A}/\mathcal{A}')$-Measurable Map* $:\Leftrightarrow$

$$A' \in \mathcal{A}' \Rightarrow T^{-1}(A') \in \mathcal{A}.$$

* * *

Definition A.13 ***(Image Measure).*** Given

- *Measurable Spaces*, (X, A), $(X', \mathcal{A}')$,
- A *Measurable Mapping*, $T\colon X \to X'$,
- An *Measure*, μ, on (X, A).

Then,

$$\mu'\colon \mathcal{A}' \to \mathbb{R},\ \ A \mapsto \mu\big(T^{-1}(A')\big) \text{ is the Image Measure of } \mu \text{ under } T.$$

* * *

Definition A.14 ***(Measurable [Numerical] Function).*** Fix $n \in \mathbb{N}$. Given

- *Measure Space*, $(X, \mathcal{A})$, $(X', \mathcal{A}')$,
- An *$(\mathcal{A}/\mathcal{A}')$-Measurable Mapping*, $u\colon X \to X'$.

Then, if $(X', \mathcal{A}') = \big(\mathbb{R}, \mathcal{B}(\mathbb{R})\big)$ $[\,\big(\bar{\mathbb{R}}, \bar{\mathcal{B}}(\bar{\mathbb{R}})\big)\,]$,

$$u \text{ is an } (\mathcal{A}/\mathcal{B}[\bar{\mathcal{B}}])\text{-Measurable [Numerical] Function, } u\colon X \to \mathbb{R}\ [\bar{\mathbb{R}}]$$

* * *

Definition A.15 ***([Real-Valued] Random Variable).*** Given

- A *Probability Space*, $(\Omega, \mathcal{A}, p)$, and a *Measurable Space*, $(X[=\mathbb{R}], \mathcal{A}')$,
- An *$(\mathcal{A}/\mathcal{A}')$-Measurable Mapping*, $Y\colon \Omega \to X[=\mathbb{R}]$.

Then Y is a *[Real-Valued] Random Variable*. The *Image Measure* of p under Y gives the *Probabilities* of the *Events*, $E \in \mathcal{A}'$.

* * *

Definition A.16 ***(Simple Function).*** Given

- A *Measurable Space*, $(X, \mathcal{A})$ along with $\big(\mathbb{R}, \mathcal{B}(\mathbb{R})\big)$,
- An *$(\mathcal{A}, \mathcal{B})$-Measurable Function*, $u : X \to \mathbb{R}$.

Then u is a *Simple Function* $:\Leftrightarrow$

$$\exists N \in \mathbb{N} :\ \exists (A_j)_{j=0}^N \text{ with } A_j \in \mathcal{A} \text{ and } \exists (c_j)_{j=0}^N \text{ with } c_j \in \mathbb{R}\ : u(x) = \sum_{j=0}^{N} c_j \cdot \mathbb{1}_{A_j}(x)\ \ \forall x \in X$$

. Given a measure, μ, on the space $(X, \mathcal{A})$; *the set of all Simple Functions* on $(X, \mathcal{A}, \mu)$ is $\mathcal{E}_\mu$. *the set of all non-negative Simple Functions* is $\mathcal{E}_\mu^+$.

* * *

Definition A.17 ***(Integral of a Simple Function).*** Given

- A *Measure Space*, $(X, \mathcal{A}, \mu)$,
- A non-negative *Simple Function*, $s\colon X \to \mathbb{R}_{\geq 0}$,
- $N \in \mathbb{N}$,
- Sets, $(A_j)_{j=0}^N$ and $(c_j)_{j=0}^N : s(x) = \sum_{j=0}^N c_j \cdot \mathbb{1}_{A_j}(x)\ \ \forall x \in X$.

Then $I_\mu(s)$ is the *Integral of the Simple Function*, s, where

$$I_\mu(s) = \sum_{j=0}^{N} c_j \cdot \mu(A_j)$$

* * *

B Approximating area — Riemann Style

This is the code I wrote to draw the sequence of approximations shown in Figure 2.1. It can be used to visualise the approximation of the Integral of any function in a *'Riemann style'* — given that you already have a way to write it as a Python function with the form:

```
# f(x) -----> number.
```

It can be modified easily to produce quite a few of approximations - here we just set n to 6.

```
import matplotlib.pyplot as plt
import numpy as np

n = 6
```

First, we want to set the limits of integration, $[a, b]$, and use this to calculate a total width.

```
xmax = 6
xmin = 0
totalwidth = xmax - xmin
```

Then we use this, and our n, to split up the axis into $5 \times 2^{n-1}$ equal sized sections. We will divide the lengths of each bar by 2 every step of the approximation, so this ensures that our coarsest approximation has 5 bars.

```
# 160 points equally spaced points (plus zero) in the domain
num = (5 * 2**(n-1)) + 1
x = np.linspace(xmin, xmax, num=num, endpoint=True)
```

Define the function of interest, f. Calculate y values to plot a graph.

```
# Make a list of 'n' colours
colour1 = '#95d0fc' #light blue
colour2 = '#75bbfd' #sky blue
colour3 = '#00ffff' #cyan
colour4 = '#13eac9' #aqua
colour5 = '#929591' #grey
colour6 = '#06c2ac' #turquoise
colour = [colour1, colour2, colour3, colour4, colour6, colour5]

f = np.sin
y = f(x)

# To draw graph
xs = np.linspace(xmin, xmax, 100)
ys = f(xs)

fig, axesr = plt.subplots(nrows=int(n/2), ncols=2, dpi=300, figsize=(5, n*3/4))
fig.subplots_adjust(hspace = 0.075)
axes = [ax for axes in axesr for ax in axes]

fig.subplots_adjust(wspace=0.05, hspace=0.075)

for ax in axes:
    ax.plot(xs, ys, 'g-', linewidth=0.5)
    ax.tick_params(top=False,
                   bottom=False,
                   left=False,
                   right=False,
                   labelleft=False,
                   labelbottom=False,
                   labelright=False,
                   labeltop=False)
```

```
    for axis in ['top','bottom','left','right']:
      ax.spines[axis].set_linewidth(0.4)

    ax.set_xlim(xmin, xmax)
```

A quick explanation of this next step is that it's setting up the *'representations'* of the *Step Functions* used to approximate the function of interest. Those sequences are defined by

$$x_{n,m} = x_{n,m-1} + \frac{totalwidth}{5 \times 2^{n-1}}$$

In this example, I am bounding the function from below, but by changing 'min' to 'max', you can bound it from above.

```
reps = [x[:-1:2**n] for n in range(n)]
bars = [(x, []) for x in reps]
for interval_steps, bar in zip(reps, bars):
    for i in range(len(interval_steps)):
        xright = xmax if i == len(interval_steps) - 1 else interval_steps[i+1]
        xleft = interval_steps[i]
        interval = np.linspace(xleft, xright, 1000)
        interval_mapped = [f(x) for x in interval]
        interval_min = min(interval_mapped) ## Can change 'min' to 'max'
        bar[1].append(interval_min)         ## to bound from above
```

Now we can plot the bars at the right heights, using the colours we picked earlier. Also, what I mean by *'bounding from below'* is that we are finding the *height sequence* where every height is the greatest lower bound (infinium) of the values the function takes in the corresponding step. But that's a bit of a mouthful.

```
i = n
for ax in reversed(axes):
    i -= 1
    div = 5 * 2**i
    ax.bar(*bars[n-i-1],
        align='edge', width=(totalwidth/div),
        fc=colour[i], ec=colour[i], alpha=0.95)

fig.savefig('RiemannExample.png')
```

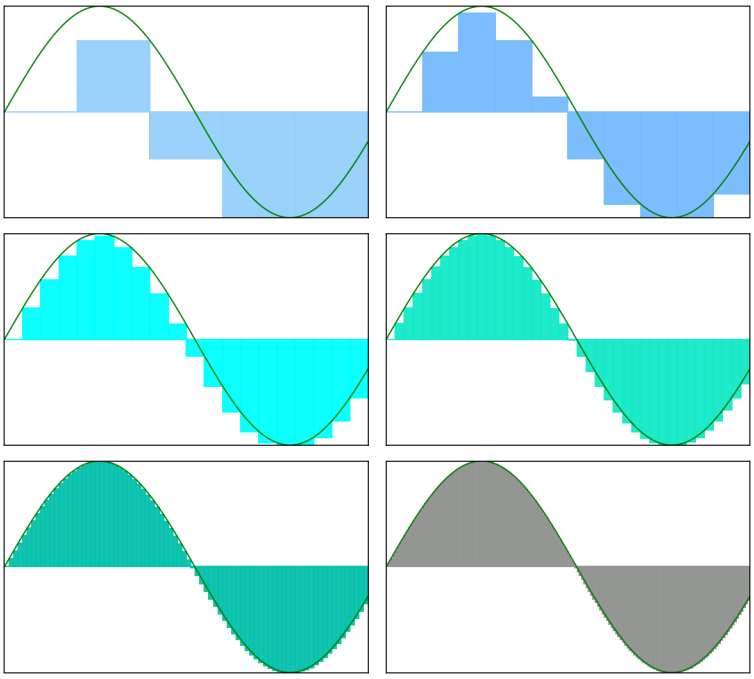


Figure B.1: *RiemannExample.png*

C Approximating Area — Lebesgue Style

This is the code I used to draw the sequence of approximations shown on the Title Page and in the Abstract. It can be used to visualise approximating the integral of any *positive Measurable Function* in a *'Lebesgue Style'*, given that you have a way to write it as a simple Python function with the form:"

```
# f(x) -----> non-negative number.
```

It can be modified easily to produce quite a few approximations. Here we do 5.

```
import matplotlib.pyplot as plt
import matplotlib.patches as patches
import numpy as np
import matplotlib.cm as cm
from scipy.interpolate import interp1d
from mpl_toolkits.axes_grid1.axes_divider import make_axes_locatable

# How many approximations?
STEPS = 5
```

In a sense, we are actually doing 2 stages of approximation. The first is that, instead of working with $(\mathbb{R}^n, \mathcal{B}(\mathbb{R}^n))$ and finding the actual *Lebesgue Integral*, I discretise the domain — i.e. split it into equal sized sections, and focus on some point in the middle — and give each point a measure that is proportional the size of each section.

If you're smarter than me, and have more time than I do, you could do everything I do below for a rectangle in R^n by dividing the domain into squares, or cubes, etc., and numbering each one. In this case, I split the domain into 500 sections. If the total length of the domain is l, each point x_n has a measure of $\frac{l}{500}$.

```
GRAN = 500
X_MIN = 0
X_MAX = 10
POINT_SIZE = (X_MAX-X_MIN)/GRAN
```

The second stage of approximation is where things get interesting. Here is where we approximate the given function as a sequence of *Simple Functions.*. Technically, because of the first approximation, our new function is already a Simple Function — any finite number of points in the domain means that we could always just define the function as

$$\sum_{\text{all } x} \mathbb{1}_{\{x\}} \cdot g(x)$$

But the point is that this method could be used on *any Measurable Function*; however I personally don't have a way to write a python function that says 'Find the all the elements in the potentially uncountable set of inverses of y, $f^{-1}(y)$, and then measure that set. The idea behind this method of approximation is as follows:

Lets imagine we have a totally random *Measure Space*, (X, A, μ), and a *non-negative Measurable Function*, u, on that space.

The goal is to approximate the area in increasing steps, with every step getting closer to the true value.

At each step in the approximation, n (starting at 0), we will focus on the function below a certain (strictly-increasing) height. Specifically, n itself.

Above this height, we will just give the Simple Function the value zero - but the higher we go, the better the approximation. Moreover, if f is bounded, then we are guaranteed to eventually reach a height where all the points are accounted for.

At every step we divide the range below n into $n \cdot 2^n$ equal sized levels, so that between 0 and 1, for instance, there are 2^n levels. Notice that with each step, the level sets are getting finer.

Fix an n. Now, let

$$y_{n,j} = y_j = j \cdot 2^{-n} \quad \forall j \in \mathbb{N}_{\leq n2^n} \cup \{0\},$$

so that the $y_{n,j}$s are the points where split the domain into levels.

For all y_j except the last one, find 'level sets'

$$X_j = \{x : f(x) \in [y_j, y_{j+1})\}$$

. i.e. find

$$f^{-1}([y_j, y_{j+1}))$$

The Simple Function, s_n, which we use to approximate the area is then

$$\sum_{j=0}^{j=n\cdot 2^{n-1}} y_j \cdot \mathbb{1}_{X_j}(x)$$

Note that since f is a Measurable Function, and since all (a, b) in the space $(\mathbb{R}, \mathcal{B})$ are Measurable, we must have

$$f^{-1}((y_j, y_{j+1})) \in \mathcal{A}.$$

That's why this works, and why f (or u or whatever I'm calling it) needs to be a *positive Measurable Function*.

We then find $I_\mu(s_n)$ using the *Measure* μ on X.

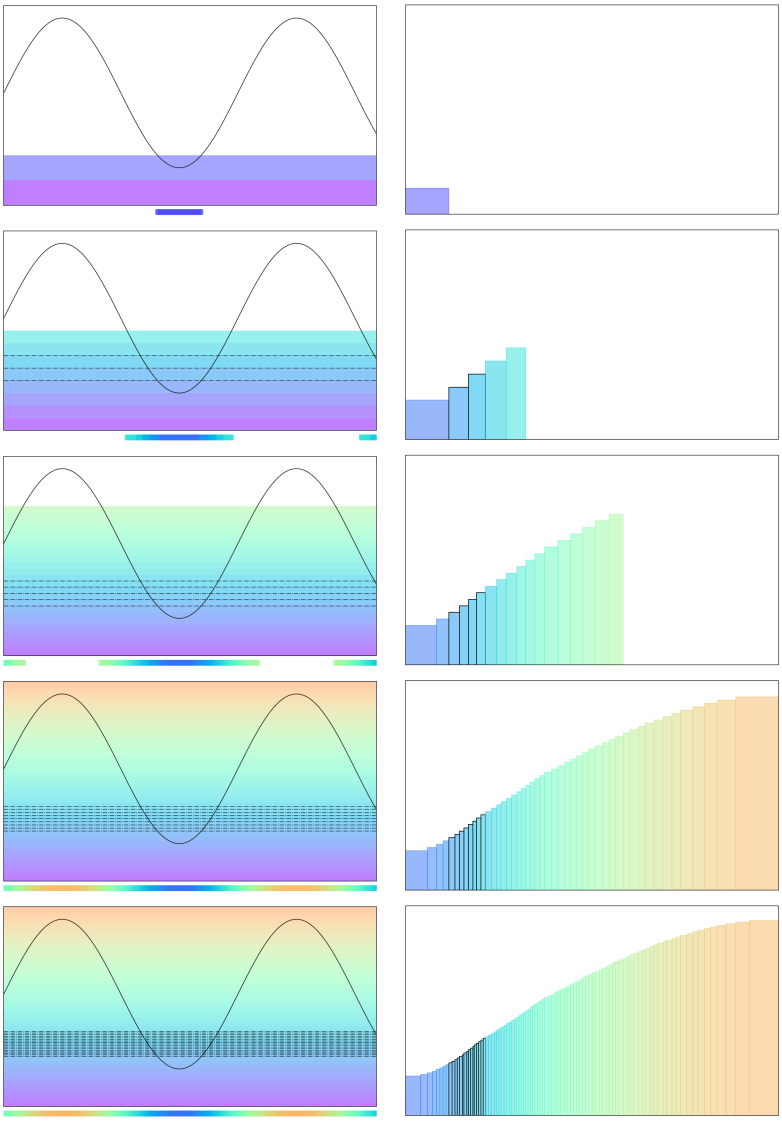
In our case, instead of using the λ-*Measure* on $\mathbb{R}$, we have to use a simplified and discretised Measure Space - but one that gets 'closer' to λ the more points you pick.

With all that out of the way, lets do my favorite part in all this, and pick the colours.

```
def colour(x=1.0):
    r = list(cm.rainbow(x/STEPS))
    r[3] = 0.5  # I think it looks better with a lower 'alpha'
    return tuple(r)

# We will be calculating 2^(-n) a lot to figure out the step size
# So this is a function to do that
def ss(x):
    return 2**(-x)

# Set up the graph and function.
# Our graph in this case is drawn by interpolation,
# but you can change the definition of f to any function
# which takes a float and returns another float.
xi = np.linspace(X_MIN, X_MAX, num=5, endpoint=True)
yi = [0, 1.75, 0.9, 2.5, 2.5]
f = interp1d(xi, yi, kind='cubic')
# The points we use to plot our graph
# and find the inverses of the levels; the level sets
def f2(x):
    return (np.sin(x) + 1.5) * 3/2
f = f2
xs = np.linspace(X_MIN, X_MAX, num=GRAN+1, endpoint=True)
ys = f2(xs)

# Set up the axes.
# For the first half
fa1 = plt.subplots(nrows=STEPS, dpi=300, figsize=(5, STEPS*3))
fig1, axes1 = fa1
# For the second half
fa2 = plt.subplots(nrows=STEPS, dpi=300, figsize=(5, STEPS*3))
fig2, axes2 = fa2

for fig in [fig1, fig2]:
    fig.subplots_adjust(wspace=0.075, hspace=0.075)
```

```
for fig, axes in [fa1, fa2]:
    for ax in axes:
        ax.set_xlim(X_MIN, X_MAX)
        ax.set_ylim(0, 4) # For this example specifically
        ax.tick_params(
           axis='both',
           which='both',
           bottom=False,
           top=False,
           left=False,
           labelleft=False,
           labelbottom=False)

###

# Calculate the level_sets.
# Code makes this easy, the difficulty is in how to organise the data
# The way I do it here is for every step in the the approximation, n,
# I have a dictionary with key vaule pairs
#                           y_j : X_j = f^(-1)([y_j, y_j+step_size))
# And those dictionaries are kept in another dictionary called
# all 'level_sets'. So all_level_sets[n][y_base] gives the level set
# for [y_base, y_base + step_size) for the step n
all_level_sets = {}
for j in range(1, 1+STEPS):
    step_size = ss(j)
    y_bases = np.arange(0, j, step_size)
    level_sets = {}
    for y_base in y_bases:
        X = [x for x in xs if \
             f(x) >= y_base and f(x) < y_base + step_size]
        level_sets[y_base] = X
    all_level_sets[j] = level_sets

# For the first figure, we want to visualise what's going on by colouring
# the different levels, and then underneath the graph the we show the level
# sets in matching colours. Unfortunately, since some many points overlap,
# this might not be obvious...
for step, ax in enumerate(axes1, 1):
    ax.plot(xs, ys, color='0.1', lw=0.5, linestyle='-')
    step_size = ss(step)
    y_bases = np.arange(X_MIN, step, step_size)
    for y_base in y_bases:
        ax.axhspan(y_base, y_base + step_size, # Colour the different levels
            facecolor=colour(y_base),
            edgecolor='none')
        if y_base >= 1 and y_base <= 1.5: # Highlight this section
            ax.axhline(y_base,
                       lw=0.5,
                       color='k',
                       alpha=0.7,
                       linestyle='-.'
                      )
#            ax.axhspan(y_base, y_base + step_size,
#                       facecolor=colour(y_base),
#                       edgecolor='none'
#                      )

    # Add a colour bar at the bottom, showing the different level
    # sets in different colours.
    # I wish I had a better way to do this...
```

```
        ax_divider = make_axes_locatable(ax)
        cax = ax_divider.append_axes("bottom", size="3%", pad="2%")
        cax.set_xlim(0,10)
        cax.set_ylim(0,0.1)
        cax.tick_params(
        axis='both',
        which='both',
        bottom=False,
        top=False,
        left=False,
        labelleft=False,
        labelbottom=False)
        cax.axis('off')
        for y_base in np.arange(X_MIN, step, step_size):
            xmatch = all_level_sets[step][y_base]
            cax.scatter(xmatch, [0.05]*len(xmatch),
                        s=15, c=[colour(y_base)], marker='s')

    # For the second figure, we'd like to rearrange points in the domain, so
    # that points in the same level set are next to each other.
    # This can be done by just counting the number of points in a level set,
    # then making a rectangle which has a width to this size.
    for step, ax in enumerate(axes2, 1):
        step_size = ss(step)
        y_bases = np.arange(X_MIN, step, step_size)
        left_edge = 0
        for y_base in y_bases:
            no_of_xs = len(all_level_sets[step][y_base])
            measure = POINT_SIZE * no_of_xs
            # Draw boxes around the rectangles correspoding to the highlighted
            # level sets.
            right_edge = measure
            rect = patches.Rectangle((left_edge, 0),
                                     measure,
                                     height=y_base,
                                     color=colour(y_base))
            if y_base >= 1 and y_base < 1.5:
                rect.set_edgecolor('k')
            ax.add_patch(rect)
            left_edge += measure
    fig1.savefig('LebesgueExample1.png')
    fig2.savefig('LebesgueExample2.png')
```



(a) LebesgueExample1.png (b) LebesgueExample2.png

Figure C.1: *Approximating Area with Simple Functions.*

References

[1] René L. Schilling, *Measures, integrals and martingales*, Cambridge University Press, New York, 2005. MR2200059

[2] Elias M. Stein and Rami Shakarchi, *Real analysis*, Princeton Lectures in Analysis, vol. 3, Princeton University Press, Princeton, NJ, 2005. Measure theory, integration, and Hilbert spaces. MR2129625

[3] Garside B, *Probability Theory* (2014), https://www.youtube.com/playlist?list=PLAvgI3H-gclbyWnE9WrSq68_HRW9rOZUw. A video series on Probability.

[4] William Weiss, *An Introduction To Set Theory*, CreateSpace Independent Publishing Platform, 2014.

[5] N. L. Carothers, *Real analysis*, Cambridge University Press, Cambridge, 2000. MR1772332