Chapter 5: Basic Calculus Concepts

This chapter provides a quick review of basic calculus concepts, including limits, continuity, differentiation, optimization, and integration. We illustrate some ways to use WL to explore these concepts.

5.1 Sequences

Sequences are usually conceptualized as functions with a domain that is the natural numbers or a subset thereof. Sequences can be finite or infinite. We often use the term 'sequence' to denote an infinite sequence and the term 'tuple' to denote a finite sequence. WL represents tuples as lists. In this chapter, we focus on real-valued sequences.

Explicitly Generating Sequence Prefixes

We cannot explicitly generate an infinite sequence, but we can often generate enough of it to be useful. The initial *N* elements of an infinite sequence is the *N*-prefix of the sequence. We often try to understand the behavior of a sequence by examining an *N*-prefix.

Many interesting sequences can be explicitly characterized by functions on the natural numbers. To map a function over an integer interval, an obvious choice is the `Array` command. (As we know from our discussion of lists, we could alternatively use `Table` or `Map` with `Range`.) For example, we can produce $1/n$ for the first 10 natural numbers as follows:

```
Array[n ↦ 1 / n, 10]
```

$$\left\{1, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \frac{1}{5}, \frac{1}{6}, \frac{1}{7}, \frac{1}{8}, \frac{1}{9}, \frac{1}{10}\right\}$$

A second common way to generate sequences is by means of recurrence relations. The simplest recurrence relation is function iteration, for which we can use the `NestList` command. The inputs are the function to iterate, a starting value, and a number of iterations. (Note that the initial value is included in the result.)

```
NestList[n ↦ n / 2, 1, 10]
```

$$\left\{1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}, \frac{1}{64}, \frac{1}{128}, \frac{1}{256}, \frac{1}{512}, \frac{1}{1024}\right\}$$

We can also use the `RecurrenceTable` command, which generates sequences for recurrence relations.

```
ClearAll[a]
RecurrenceTable[{a[t] == 0.9 a[t - 1], a[0] == 100}, a, {t, 0, 10}]
```

```
{100., 90., 81., 72.9, 65.61, 59.049, 53.1441, 47.8297, 43.0467, 38.742, 34.8678}
```

When the recurrence relation is linear, as in the case just illustrated, the `LinearRecurrence` command is an alternative to the use of `RecurrenceTable`.

```
LinearRecurrence[{0.9}, {100}, 11]
```

```
{100., 90., 81., 72.9, 65.61, 59.049, 53.1441, 47.8297, 43.0467, 38.742, 34.8678}
```

Look-and-Say Sequence

The look-and-say sequences have more entertainment value than practical utility. However, exploring them provides a chance to introduce some useful WL commands.

A look and say sequence is based on iterating verbal descriptions of base 10 number literals, counting up the length of runs of identical numerals. Each run is then described in terms of its length and the numeral involved. For example, the number 1 may be described as “one 1”, which is then transcribed as 11. This in turn is described as two consecutive instances of the number 1, or two ones, transcribed as 21. This in turn becomes one 2 followed by one 1, or 1211. And so forth. First we break the number up into a list of digits, which we then split into runs of numerals. We can use the `IntegerDigits` and `Split` commands for this. Recall that the `Length` command returns the length of a list and the `First` command returns the first element of a list.

Expression	Result	Comment
`digits=IntegerDigits[2233222]`	`{2, 2, 3, 3, 2, 2, 2}`	list of digits
`runs=Split[digits]`	`{{2, 2}, {3, 3}, {2, 2, 2}}`	split into runs
`{Length[#],First[#]}&/@runs`	`{{2, 2}, {2, 3}, {3, 2}}`	pair with counts

With that background, let us implement a `lookAndSay` function, which works as follows. After each run is represented by its length and a specification of the numeral. We string these representations together into a list of digits (using `Catenate`), which we then use to form the next number in the sequence (using `FromDigits`).

```
lookAndSay[i_Integer?Positive] :=
 Module[{digits = IntegerDigits[i], cts},
  cts = {Length[#], First[#]} & /@ Split[digits];
  FromDigits@Catenate[cts]
 ]
```

As an example, let us start with a seed of 1 and produce the next 5 members of the sequence. We can use `NestList` to do function interaction while accumulating the intermediate results.

```
Column[NestList[lookAndSay, 1, 5]]
```

```
1
11
21
1211
111221
312211
```

Note how the last numeral is always our seed. If we start with a seed of 3, the resulting sequence is sometimes called the Conway sequence.

```
Column[NestList[lookAndSay, 3, 5]] (* five elements of Conway Sequence *)
```

```
3
13
1113
3113
132 113
1 113 122 113
```

Exploring the look-and-say sequence is just an entertainment. However, the commands introduced in this section prove useful in many settings. These included `IntegerDigits`, `FromDigits`, and `Split`.

5.2 Limits

Sequence Limits

Recall that an infinite sequence has a limit L iff almost all of its points are arbitrarily close to L. Sometimes it is easy to apply this definition with WL, and sometimes it takes a little work.

Consider a sequence that can readily be represented as a function of the natural numbers, In the simplest case, the function can simply be extended to the real numbers without affecting the limit, and we can just use the `Limit` command. The basic version of this command takes two arguments: a expression, and a rule characterizing the limit we are interested in.

```
Limit[1 / t, t → ∞]
```

```
0
```

One possible complication is that extension of a function to the real numbers can change its limiting behavior. For example,

```
Limit[Sin[t π], t -> ∞]
```

```
Interval[{-1, 1}]
```

In such cases, we may be able to simply set a domain restriction using the `Assumptions` option of the `Limit` command.

```
Limit[Sin[t π], t -> ∞, Assumptions → Element[t, Integers]]
```

```
0
```

Should domain restriction fail, we may be able to apply the `Sum` command to the first differences of the sequence. Note that $x_T = x_0 + \sum_{t=1}^{T} (x_t - x_{t-1})$.

```
Sin[0 π] + Sum[Simplify[Sin[t π] - Sin[(t - 1) π]], {t, 1, ∞}]
```

```
0
```

If you try to evaluate a sum that does not converge, you will receive a warning.

```
Sum[1 / k, {k, 1, ∞}]
```

Sum: Sum does not converge.

$$\sum_{k=1}^{\infty} \frac{1}{k}$$

Function Limits

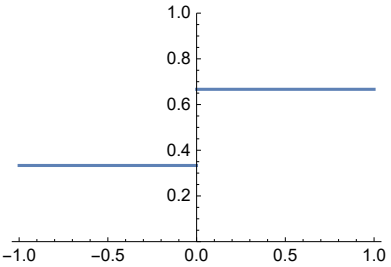
Recall that a function f has a limit L at point x_0 within its domain if $f[x]$ approaches L as x approaches x_0. A function that is discontinuous at a point can nevertheless have a one-sided limit (from the left or from the right) at that point.

Use `Limit` to determine the limit of a function at a point. Here is a simple example of a piecewise continuous function with a single discontinuity. (Here we use the `If` command in our definition, but we could alternatively have defined this function using the `Piecewise` or the `UnitStep` commands.)

```
f = x ↦ If[x < 0, 1 / 3, 2 / 3];
```

In order to produce a nice jump in the plot of this function, we can use the `Exclusions` option to warn the `Plot` command to exclude the point of discontinuity.

```
Plot[f[x], {x, -1, 1}, Exclusions → {0},
 PlotRange → {0, 1}, ImageSize → 200]
```



The `Limit` command produces one-sided limits. The default is to takes the limits from the right (except at ∞). For example,

```
Limit[f[x], x → 0]
```

$$\frac{2}{3}$$

However you can ask for a limit from the left by setting the `Direction` option to 1. For example, the the limit at 0 from the left is found as follows:

```
Limit[f[x], x → 0, Direction → 1]
```

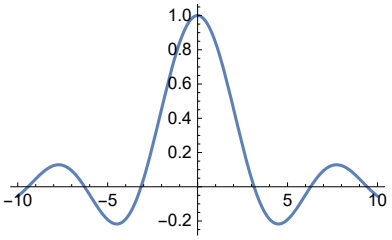
$$\frac{1}{3}$$

A function has a limit at a point x if and only if it has the same from the left and the right. So in the example above, we see that $f[x]$ does not have a limit at $x = 0$, even though both one-sided limits exist.

A function need not be defined at a point in order to have a well-defined limit there. Consider the classic example of $\sin(x)/x$. The plot suggests that although the function is undefined at $x = 0$ it still has a well-

defined limit there.

```
Plot[Sin[x]/x, {x, -10, 10}, Exclusions → {0}, ImageSize → 200]
```



We can confirm this with the `Limit` command. We see that at 0 the limit from the left matches the limit from the right for sin(*x*)/*x*, even though the function is not defined at *x* = 0.

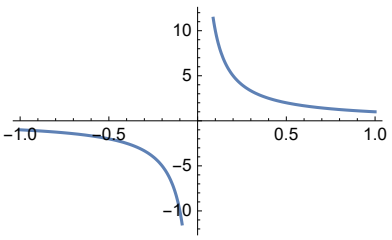
Expression	Result	Comment
`Limit[Sin[x]/x, x→0, Direction→1]`	1	limit from left
`Limit[Sin[x]/x, x→0]`	1	limit from right

When *f* has a limit at *x* that is the same as *f*[*x*], we say that *f* is continuous at *x*. So the following function is continuous:

```
x ↦ If[x == 0, 1, Sin[x] / x];
```

In contrast, consider 1/*x*. It too is undefined at *x* = 0, but it does not have a limit there. See why by examining the plot. (We use the `Exclusions` option to inform Mathematica that the function is not defined at *x* = 0.)

```
Plot[1/x, {x, -1, 1}, Exclusions → {0}, ImageSize → 200]
```



The `Limit` command still gives the right guidance in such cases.

Expression	Result
`Limit[1/x, x→0, Direction→1]`	-∞
`Limit[1/x, x→0]`	∞

Parameter Dependent Limits

Often we can take limits with respect to one variable of expressions in multiple variables. These other variables then parameterize the expression of the limit. For example, suppose there is an interest rate *i* compounded at a rate of *f* times per period. Then we find that as we increase the frequency of compounding, the effective gross interest rate over the period has a limiting value of

```
Clear[r, f, n]
Limit[(1 + r / f) ^ f, f → ∞]
```

e^r

When our expression includes a parameter, its limiting value may naturally depend on the permitted values of that parameter. [torrence.torrence-2009-cup]_ (p.198) offer the following example.

```
Clear[x, n]
Limit[(1 - x^n) / n, x → 0]
```

$$\text{Limit}\left[\frac{1 - x^n}{n}, x \to 0\right]$$

Without more information, we cannot evaluate the limit. However, we can pin down the limit if we know whether *n* is positive or negative. We can use the `Assumptions` option to introduce this information.

Expression	Result
$\text{Limit}\left[\frac{1-x^n}{n}, x\to 0, \text{Assumptions}\to\{n<0\}\right]$	∞
$\text{Limit}\left[\frac{1-x^n}{n}, x\to 0, \text{Assumptions}\to\{n>0\}\right]$	$\frac{1}{n}$

Continuity

Recall that a function is continuous at a point x_0 within its domain if it has a limit at x_0 that equals its value at x_0. Polynomials are examples of continuous functions.

```
ClearAll[f]
f = x ↦ x * x;
Limit[f[x], x → x0, Direction → 1] == Limit[f[x], x → x0, Direction → -1] == f[x0]
```

```
True
```

In contrast, rational functions can have points of discontinuity. Sometimes we need only observe that the limit is different from the right and left.

```
ClearAll[f]
f = x ↦ 1 / x;
Limit[f[x], x → 0, Direction → 1] == Limit[f[x], x → 0, Direction → -1]
```

```
False
```

Other times, however, we need to ensure that we are actually dealing with a point in the domain.

```
ClearAll[f]
f = x ↦ 1 / Abs[x];
Limit[f[x], x → 0, Direction → 1] == Limit[f[x], x → 0, Direction → -1]
```

```
True
```

Finally, we may have the right-side and left-side limits equal but fail the requirement that these equal the function value.

```
ClearAll[f]
f = x ↦ If[x ≠ 0, 1/3, 2/3];
Limit[f[x], x → 0, Direction → 1] == Limit[f[x], x → 0, Direction → -1] == f[0]
```

```
False
```

Box-Cox Transformation

Macro economists usually focus on production relationships that can be written as

$$\tau(Y/A) = \alpha\,\tau(K) + \beta\,\tau(N) \tag{1}$$

where we introduce symbols for output (Y), capital stock (K), labor (N), and technology (A). Since we should not use capital letters as WL variables, here we double the lower case versions.

```
ClearAll[τ, yy, kk, nn, aa] (* do not use caps for WL symbols! *)
eq1 = τ[yy / aa] == α * τ[kk] + β * τ[nn]
```

$$\tau\left[\frac{yy}{aa}\right] == \alpha\,\tau[kk] + \beta\,\tau[nn]$$

Usually $\alpha \in (0, 1)$ and $\beta = 1 - \alpha$. A particularly popular choice for τ is the Box-Cox transformation.

```
Clear[b]
τ = x ↦ (x^b - 1)/b;
```

The Box-Cox transformation attracted interest because it subsumes some common function forms. In particular, the result is linear in x if $b = 1$ and becomes logarithmic in x as $b \to 0$.

```
Limit[τ[x], b → 0]
```

```
Log[x]
```

(Here's one way to see that this is the case. Let $Y = e^x$ and use l'Hopital's rule to show $\tau(Y) \to x$. We discuss l'Hopital's rule in the next section.)

Implied Production Technology

With the Box-Cox definition of τ, our production technology becomes

```
eq1
```

$$\frac{-1 + \left(\frac{yy}{aa}\right)^b}{b} == \frac{\left(-1 + kk^b\right)\alpha}{b} + \frac{\left(-1 + nn^b\right)\beta}{b}$$

Let us look at the implied production technology. We can determine the implied relationship between output and inputs by using `Solve`. (We use the `Simplify` command to force a simple representation, and we use the `Quiet` command to suppress a warning that the solutions may not be exhaustive.)

```
bc`yy = First@Solve[eq1, yy] // Simplify // Quiet
```

$$\left\{yy \to aa\left(1 + \left(-1 + kk^b\right)\alpha + \left(-1 + nn^b\right)\beta\right)^{\frac{1}{b}}\right\}$$

We can rewrite this in more familiar notation as

$$Y = A\left((1-\alpha-\beta) + \alpha K^b + \beta N^b\right)^{1/b} \tag{2}$$

Economists are most familiar with the case of constant returns to scale, when $\beta = 1 - \alpha$.

Constant Returns to Scale

```
ces`yy = bc`yy /. {β → 1 - α} // Simplify
```

$$\left\{yy \to aa\left(-nn^b\left(-1+\alpha\right) + kk^b\,\alpha\right)^{\frac{1}{b}}\right\}$$

$$Y = A\left[\alpha K^b + (1-\alpha) N^b\right]^{1/b} \tag{3}$$

This is known as the constant elasticity of substitution (CES) production function. It embodies both linear production and Cobb-Douglas production.

Linear Production Technology

When $b = 1$, the CES becomes the linear production function

```
b1`yy = bc`yy /. {b → 1} // Simplify
```

$$\left\{yy \to aa\left(1 + \left(-1+kk\right)\alpha + \left(-1+nn\right)\beta\right)\right\}$$

$$Y = A\left((1-\alpha-\beta) + \alpha K + \beta N\right) \tag{4}$$

When $\beta = 1 - \alpha$, the linear technology becomes

$$Y = A(\alpha K + (1-\alpha) N) \tag{5}$$

Cobb-Douglas Production Technology

When $b \to 0$, we get a log-linear technology.

```
b0`yy = Limit[yy /. bc`yy, b → 0]
```

$$aa\ kk^{\alpha}\ nn^{\beta}$$

So our production relationship is linear in logs.

$$\ln Y = \ln A + \alpha \ln K + \beta \ln N \tag{6}$$

$$Y = A K^{\alpha} N^{\beta} \tag{7}$$

When $\beta = 1 - \alpha$ this gives us the famous Cobb and Douglas (1927) production function (previously used by Wicksell).

```
b0`yy /. {β → 1 - α}
```

$$aa\ kk^{\alpha}\ nn^{1-\alpha}$$

$$Y = A K^{\alpha} N^{1-\alpha} \tag{8}$$

5.3 Univariate Differential Calculus

In this section, we introduce the use of univariate differential calculus to analyze real-valued functions of one real variable. We begin by developing an understanding of the “slope” concept that is applicable to

nonlinear functions. We can continue in this, producing ever higher-order derivatives of the primitive function.

Difference Quotient

A discrete shift of a function *f* is a new function *g* such that *g*[*x*] == *f*[*x* + *h*]. A common notation for the shift in the argument for a real function is *h*, which is often called the "step". Other notations are common. For example, when the argument is represented by the variable *x*, it is also common to represent the step as Δx or dx. We can use the `DiscreteShift` command to produce a discrete shift of a function. The first argument is an expression in a variable, and the second argument specifies the shift with respect to that variable. Here we specify that *f*[*x*] is our expression, that the shift is with respect to the variable *x*, that we shift once, and that the shift is of size *h*:

```
ClearAll[f, x, h]
DiscreteShift[f[x], {x, 1, h}]
```

```
f[h + x]
```

More concretely, let *f* be the squaring function and let *h* = 0.1.

```
DiscreteShift[x^2, {x, 1, 0.1}]
```

$$(0.1 + x)^2$$

The difference between a discrete shift of the function and the original function is a new function, often called the difference-delta of the function. We may represent this as $\triangle_h f[x] = f[x + h] - f[x]$. A difference-delta of a function evaluates to the change in the value of the function given a step change in its argument. We can use the `DifferenceDelta` command to compute these values as the first-order discrete difference of the function.

```
DifferenceDelta[f[x], {x, 1, h}]
△{x,1,h} f[x] (* equivalent expression *)
```

```
-f[x] + f[h + x]
```

```
-f[x] + f[h + x]
```

Note that the difference-delta of a polynomial is another polynomial, with decremented degree. For example, we illustrate this outcome here by using the `Exponent` command, which returns the highest power in the expression that is its first argument of the variable that is its second argument.

Expression	Result
Exponent[x^3,x]	3
ddx3=DifferenceDelta[x^3,{x,1,h}]	$h^3 + 3\,h^2\,x + 3\,h\,x^2$
Exponent[ddx3,x]	2

A difference quotient of a function expresses the slope between two points of the function. This is just the ratio of a difference-delta of a function to the step, but it is simplest to use the `DifferenceQuotient` command. (Note: some call signatures for this command are inconsistent with those of `DifferenceDelta`; read the documentation carefully.)

```
DifferenceQuotient[f[x], {x, 1, h}]
```

$$\frac{-f[x]+f[h+x]}{h}$$

A line segment between two points of a function is called a secant segment. A difference quotient gives the slope of a secant segment. This slope may be affect both by the initial point and by the step size.

```
f = x ↦ x^2;
x0 = 0.3;
DifferenceQuotient[f[x], {x, 1, h}]
% /. {x → x0, h → 0.5}
```

$$h+2x$$

```
1.1
```

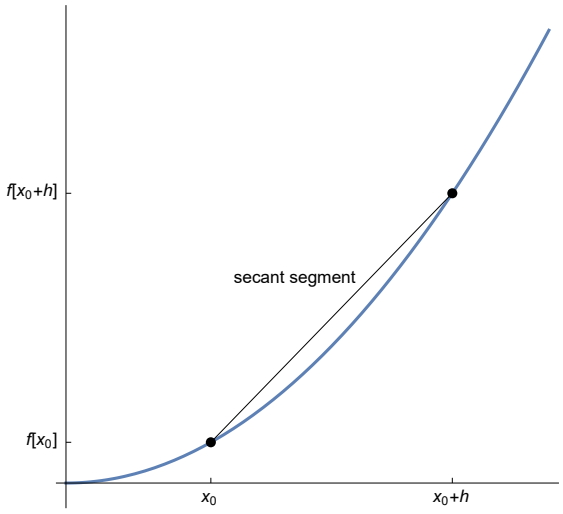


Figure 8: A Secant Segment

We take a slightly different approach, defining the first-order difference quotient as a function of three arguments. Produce the slope between (x, $f[x]$) and ($x+h$, $f[x+h]$) as follows:

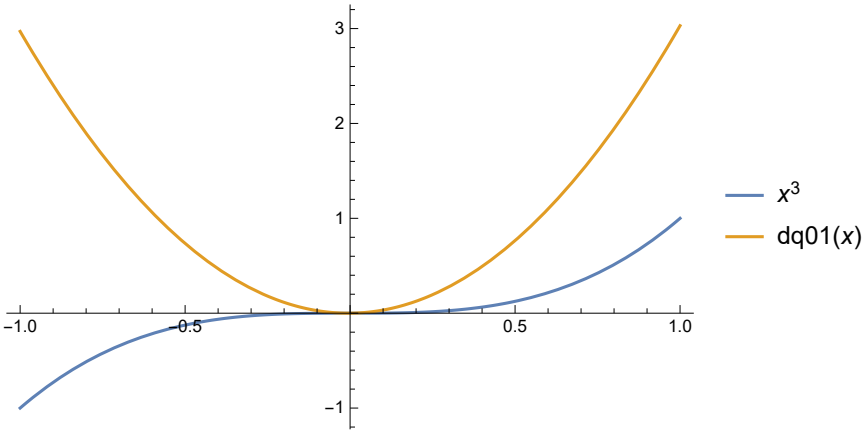
```
qfxh = {f, x, h} ↦ (f[x + h] - f[x])/h;
```

As an example, let us compute a difference quotient for the function $f[x] = x^3$ with a step of 0.01. Since we have not specified a value for x, we need to produce a function applicable at any value of x. We can use partial function application for this.

```
dq01 = qfxh[x ↦ x^3, #, 0.01] &;
```

This is an univariate real function, so we are now in a position to compute the value of this difference quotient for any argument.

```
Plot[{x^3, dq01[x]}, {x, -1, 1},
 PlotLegends → "Expressions"]
```



Advanced: Note that the `DiscreteShift` command produces an expression in the variable *x* (and in *h*, if it is not specified). This introduces a subtlety, if we want to actually produce function whose value is this expression. We can do so as follows.

```
ClearAll[f, x, h]
g = x ↦ Evaluate@DiscreteShift[f[x], {x, 1, h}];
```

Note the use of `Evaluate`. If we neglect this, then `DiscreteShift` will not be evaluated until we call *g*, and this call will evaluate the expression on the right after replacing its occurrences of *x* with the value of the argument provided to *g*. But `DiscreteShift` manipulates the variable (i.e., symbol) in this expression, and it will therefore choke on a numerical value. (Try it.) Our call to `Evaluate` immediately produces the expression in *x* that we actually want on the right, before creating the delayed assignment. This is just what we want.

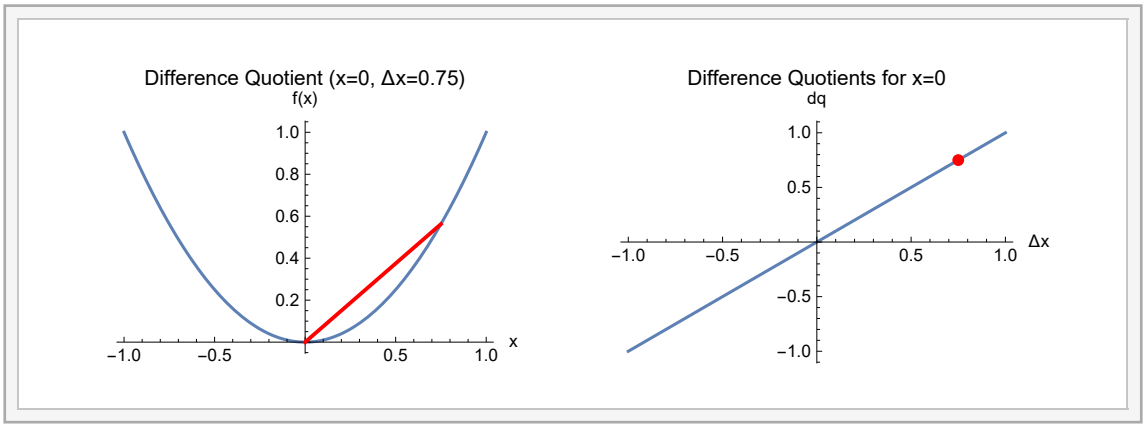
```
g[2]
```

```
f[2 + h]
```

The same considerations apply to the `DifferenceDelta` and `DifferenceQuotient` commands.

Average Rate of Change

As usual, the slope between two points of a continuous function is an average rate of change.



First-Order Derivative

Let us return to considering the difference quotient function that we defined above.

```
ClearAll[f, x, h]
qfxh[f, x, h]
```

$$\frac{-f[x]+f[h+x]}{h}$$

For any given value of x, we can think of this as a univariate function (of h). For the most part, the continuity of this function depends on the continuity of f. However, even if f is continuous, the difference quotient (as a function of h) is not a continuous function at $h=0$; in fact, it is not defined at $h=0$. We are going to try to plug this hole: we are going to propose a value for qfxh(f, x, 0) so that the difference quotient defined this way is continuous in h. Of course, to do that, we need the difference quotient to have a limit at $h=0$. If it does, we will use this limit to plug the hole, we will call it the derivative of f at x, and we will say the underlying function f is differentiable at x.

To put this another way, a function is differentiable at x if there is a limiting value of the difference quotient at x as $h \to 0$. When it exists, we define the derivative at x to be this limiting value of the difference quotient. It is traditional to write the derivative of f at x as $f'[x]$.

$$f'[x] = \lim_{h\to 0} \frac{f[x+h]-f[x]}{h}$$

If a function is differentiable at every point in its domain, we call it a differentiable function. As we have seen, continuity is a necessary condition for differentiability: we need there to be no “jumps” in the function values. But more than that is required: but we also need no sudden changes in the slope. We need there to be no kinks in the function. For example, even though it is continuous, the absolute value function $f : x \to |x|$ is differentiable everywhere except at $x=0$.

```
Limit[qfxh[Abs, 0, h], h → 0, Direction → 1] == Limit[qfxh[Abs, 0, h], h → 0, Direction → -1]
```

```
False
```

As a simple example of a differentiable function, consider the limit of the difference quotient of the function $f : x \to x^2$. We can find the derivative using our difference quotient function.

Expression	Result
f=x⟼x^2	Function[x, x^2]
dq=qfxh[f,x,h]	$\frac{-x^2+(h+x)^2}{h}$
Simplify[dq]	h + 2 x
Limit[dq,h→0]	2 x

In fact, WL provides commands to do find derivatives, along with a variety of common notations to represent them. To begin with, we can simply ask Mathematica to evaluate `f'[x]`. This is a very natural notation, but there is one additional notation that we will often use. Given an expression representing differentiable function in some variable, we often use the `D` command to produce its derivative with respect to that variable. (The mnemonic for `D` is "derivative".) Overlapping in functionality with the `D` command is the `Grad` command. (The mnemonic for `Grad` is "gradient".)

Expression	Result
f'[x]	2 x
D[f[x],x]	2 x
Grad[f[x],{x}]	{2 x}

Note that the `Grad` command produces a list; we will discuss it further when we get to multivariate calculus. There is yet another notation available: ` $\partial_x f[x]$ ` is a shorthand for `D[f[x],x]`, but we rarely use this shorthand in this book

It is useful to conceptualize the derivative $f'[x]$ as the slope of the tangent line at the point $(x, f(x))$. Recall that we can write the function for the the tangent line at $(x_0, f[x_0])$ as

$$f[x_0] + (x - x_0)\, f'[x_0] \tag{12}$$

For example, here is a graphical representation of the function $f : x \to x^2$ along with a tangent line to the function graph. The slope of the tangent line is the derivative of the function at the first coordinate of the point of tangency

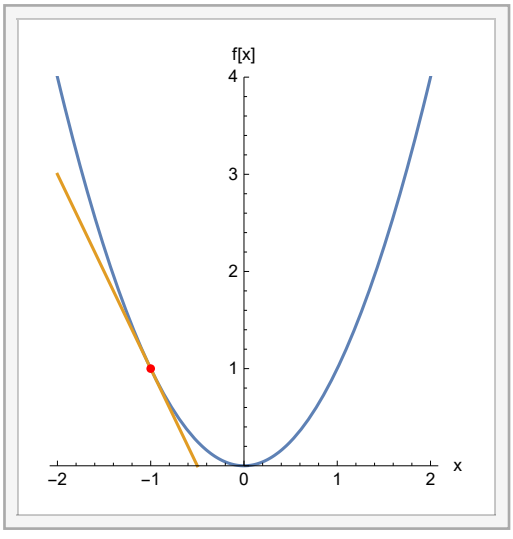


Figure 9: Function with Tangent Line

Advanced

The `Derivative` command returns a differentiation operator. For example, `Derivative[1]` returns an operator that maps a univariate function to its first-order derivative function. The prime notation (e.g., f')

is a shorthand for the application of this operator. Differentiation of a pure function produces a new, derivative function.

Expression	Result
FullForm[f']	Derivative[1][f]
f=x⟼x^2	Function[x, x^2]
f'	Function[x, 2 x]

Rules of Differentiation

Introductory treatments of differential calculus typically present (or derive) a collection of "rules" for differentiation. Here we illustrate a few of them.

Expression	Result	Comment
D[f[x]+g[x],x]	$f'[x] + g'[x]$	sum rule
D[f[x]*g[x],x]	$g[x]\, f'[x] + f[x]\, g'[x]$	product rule
D[1/f[x],x]	$-\frac{f'[x]}{f[x]^2}$	quotient rule
D[f[g[x]],x]	$f'[g[x]]\, g'[x]$	chain rule
D[InverseFunction[f][x],x]	$\frac{1}{f'\left[f^{(-1)}[x]\right]}$	inverse function rule

Elasticity vs Slope

If we take the derivative of the logarithm of a function with respect to the function argument (using the chain rule), we get the percentage change in the function resulting from a unit change in the argument. This is known as the semi-elasticity of the function.

```
D[Log[f[x]], x]
```

$$\frac{f'[x]}{f[x]}$$

Exponential functions have a constant semi-elasticity.

```
D[Log[a ℯ^(b x)], x]
```

b

If we scale the semi-elasticity by the function argument, we get the elasticity of the function. The elasticity of a function is the percentage change in its value given a percentage change in its argument.

```
elasticity = {f, x} ↦ f'[x] * x / f[x];
```

Economists often prefer to talk about elasticities instead of slopes, because elasticities are unit free. For example, we report the same price elasticity of demand no matter whether the good is measured in kilograms or tons. Power functions have a constant elasticity.

```
elasticity[x ↦ a x^b, x]
```

b

Application: Marginal Revenue

In economics, the term ‘marginal’ translates roughly as “slope”. Thus marginal cost is the slope of the cost curve, marginal revenue is the slope of the revenue curve, and marginal utility is the slope of the utility function with respect to some good. Since a derivative is a slope, we are going to apply our understanding of the derivative to characterizing the marginal revenue of production for a monopolist. We begin with a characterization of the demand curve facing the monopolist.

```
ClearAll[qd, q, p]
qd = p ↦ 150 - p / 2; (* qty demanded as a fn of price *)
```

Given the demand curve, we can easily compute the total revenue at each price: total revenue is just the product of the monopolist’s price and the quantity demanded at that price. However, for reasons that will soon become clear, economists usually represent total revenue as a function of quantity instead of price. In order to do so, we will work with the inverse of the demand function.

```
pd = InverseFunction[qd]; (* demand price as fn of qty *)
```

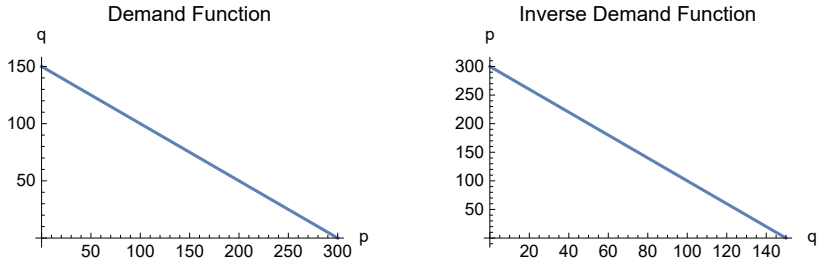


Figure 10: Demand and Inverse Demand

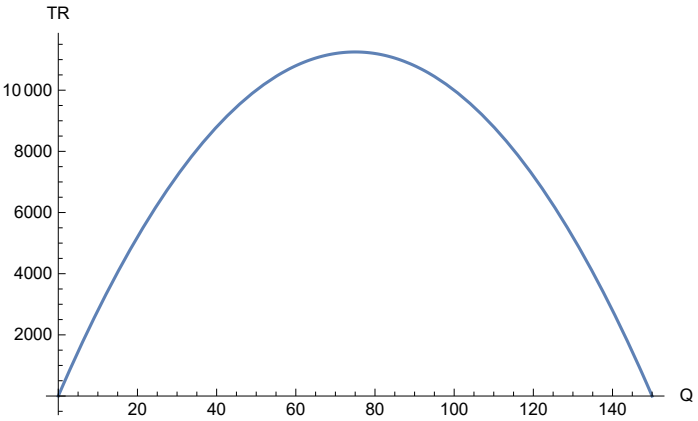
Given the inverse demand function, we can write total revenue as a function of quantity. (Here we use the `Expand` command to force the multiplication to be carried out before the expression is displayed.)

```
tr = q ↦ pd[q] * q;
tr[q] // Expand
```

$300\,q - 2\,q^2$

If we plot total revenue, we see that it reaches a maximum around 75.

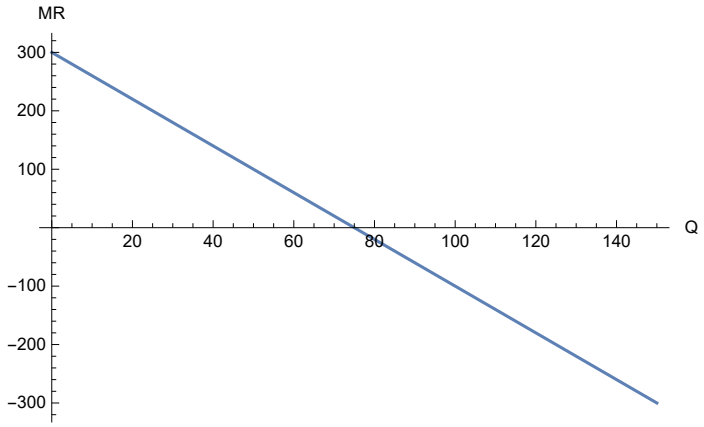
```
Plot[tr[q], {q, 0, 150}, AxesLabel → {"Q", "TR"}]
```



Marginal revenue is the slope of the total revenue curve. It follows that if we plot marginal revenue, it will be positive for values where total revenue is rising, negative for values where total revenue is falling, and zero right at the maximum. (We will return to this observation.)

```
D[tr[q], q] // Expand
Plot[Evaluate@D[tr[q], q], {q, 0, 150}, AxesLabel → {"Q", "MR"}]
```

```
300 - 4 q
```



Application: The Logistic Distribution

When a continuous probability distribution has a differentiable cumulative distribution function, the derivative is the probability distribution function.

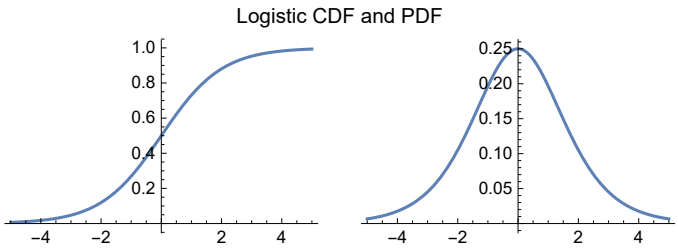
The logistic distribution has a sigmoid-shaped cumulative distribution function. This has found many applications in applied statistics. The simplest version of the logistic cumulative distribution function is $1/(1+e^{-x})$. From this we can derive the probability distribution function, which is symmetric around the mean of the distribution.

```
logisticCDF = x ↦ 1/(1 + ℯ^-x);
logisticPDF = x ↦ logisticCDF'[x]
logisticPDF[x]
```

```
Function[x, logisticCDF′[x]]
```

$$\frac{e^{-x}}{\left(1+e^{-x}\right)^2}$$

```
GraphicsRow[{
  Plot[logisticCDF[x], {x, -5, 5}],
  Plot[logisticPDF[x], {x, -5, 5}]
 }, PlotLabel → "Logistic CDF and PDF"]
```



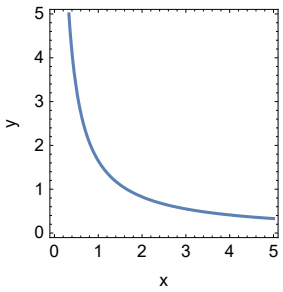
Application: Marginal Rate of Substitution

Consider a simple and popular representation of the utility (or satisfaction) of a consumer of two goods.

```
ClearAll[u, x, y, k]
u = {x, y} ↦ Log[x y];
```

Combinations of the two goods that yield a common level of utility are a level set of the utility function, known to economists as an indifference curve. For example:

```
ContourPlot[0.5 == u[x, y], {x, 0, 5}, {y, 0, 5},
 FrameLabel → {"x", "y"}, ImageSize → 144]
```



We are interested in determining the slope of the indifference curve. However, we have the utility function, not the functional relationship between *y* and *x* implied by the indifference curve. This relationship is implicit.

One possible approach is to make it explicit by solving the indifference curve equation for *y* in terms of *x*.

```
ySolns = Solve[k == u[x, y], y, Reals]
```

$$\left\{\left\{y \to \frac{e^k}{x}\right\}\right\}$$

Since there only one solution is offered, we do no need to choose among solutions.

```
ySoln = y /. First[ySolns]
```

$$\frac{e^k}{x}$$

Once we explicitly express the functional relationship, we can find a slope as usual.

```
mrsExplicit = D[ySoln, x]
```

$$-\frac{e^k}{x^2}$$

Economics call the absolute value of this slope the marginal rate of substitution of *x* for *y*. It is the amount of *y* that one unit of *x* can replace without any reduction in consumer utility.

Another approach to the same problem is implicit differentiation. Recall that along an indifference curve, *y* is implicitly a function of *x*. If we had our hands on that function, *y*[*x*], then the following would be an identity:

```
indiff = k == Log[x * y[x]];
```

Differentiating the implicit function, we can solve for the desired slope.

```
mrsSolns = Solve[D[indiff, x], y'[x]]
```

$$\left\{\left\{y'[x] \to -\frac{y[x]}{x}\right\}\right\}$$

Of course, since we relied on the implicit function, our solution remains somewhat implicit.

```
mrsImplicit = y'[x] /. First[mrsSolns]
```

$$-\frac{y[x]}{x}$$

However, if we substitute our explicit solution for *y*, we of course recover the same explicit expression for the slope of the indifference curve.

```
mrsImplicit /. {y[x] -> ySoln}
```

$$-\frac{e^k}{x^2}$$

We will illustrate one last way to compute this slope, using the `Dt` command to produce a total differential of the indifference equation.

```
Dt[0.5 == u[x, y]]
```

$$0 == \frac{y\,\text{Dt}[x] + x\,\text{Dt}[y]}{x\,y}$$

If we set the change in *x* to unity and solve for the change in *y*, we produce the sought after slope.

```
Solve[% /. {Dt[x] → 1}, Dt[y]]
```

$$\left\{\left\{\text{Dt}[y] \to -\frac{y}{x}\right\}\right\}$$

In this particular case, the very simple form of the result provides some new understanding of the marginal rate of substitution for this utility function.

Application: Marginal Rate of Transformation

Robin Crusoe lives alone on an island, where she allocates 8 hours per day to food production. Her labor can be allocated either to catching fish or gather coconuts. (Both are measured in pounds.) We

are given the following description of production possibilities. (An additional constraint, which we do not state algebraically, is that the labor allocated to each activity must be nonnegative; we will return to this.)

```
ff = lf ↦ 4 Sqrt[lf]; (* fish production *)
fc = lc ↦ 5 Sqrt[lc]; (* coconut production *)
constraint = 8 == lc + lf; (* constraint on total labor *)
```

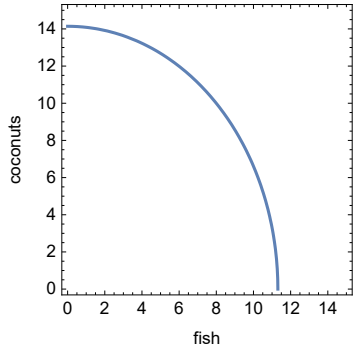
We want to describe the production possibilities available to Crusoe. In order to do so, we will substitute inverse functions (which give the labor requirements for the production of fish and coconuts) into the resource constraint (on total available labor).

```
ppf = constraint /. {lf → InverseFunction[ff][fish], lc → InverseFunction[fc][coconuts]}
```

$$8 == \frac{\text{coconuts}^2}{25} + \frac{\text{fish}^2}{16}$$

We can use the `ContourPlot` command to illustrate the production possibilities that we have discovered. The result is called the production-possibilities frontier (PPF).

```
ContourPlot[Evaluate@ppf, {fish, 0, 15}, {coconuts, 0, 15},
 FrameLabel → {"fish", "coconuts"}, ImageSize → Small]
```



Next we ask the question, what must Crusoe give up in coconuts to get another fish? The answer to this depends on the current allocation of labor; it is given by the slope of the production-possibilities frontier. We will take a couple different approaches to this problem.

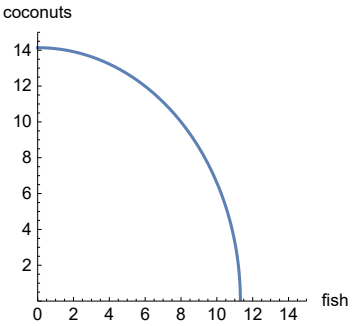
Our first approach will be to explicitly solve for coconuts in terms of fish using `Solve`. We find that we get two solutions, but only the second solution describes outcomes in the first quadrant (where both fish and coconuts are nonnegative).

```
solns = Solve[ppf, coconuts]
```

$$\left\{\left\{\text{coconuts} \to -\frac{5}{4}\sqrt{128 - \text{fish}^2}\right\}, \left\{\text{coconuts} \to \frac{5\sqrt{128 - \text{fish}^2}}{4}\right\}\right\}$$

If we plot this solution, we of course get the same representation of production possibilities.

```
soln02 = coconuts /. Last[solns];
Plot[soln02, {fish, 0, 15}, PlotRange → {{0, 15}, {0, 15}},
 AxesLabel → {"fish", "coconuts"}, AspectRatio → 1, ImageSize → Small]
```



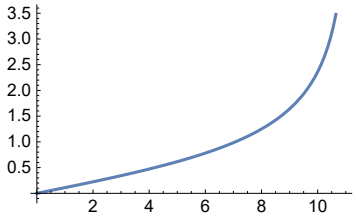
We are now ready to address the question, at what rate must we give up coconuts to get more fish?

```
mrtExplicit = D[soln02, fish]
```

$$-\frac{5\,\text{fish}}{4\sqrt{128-\text{fish}^2}}$$

Economists have a couple of different names for the absolute value of this slope. It is called the marginal rate of transformation of fish into coconuts. Roughly, it tells us how many more pounds of coconuts we can have if we give up one pound of fish. It is also referred to as the marginal opportunity cost of fish. This is because it tells us roughly how many pounds of coconuts we give up to get another fish. The concave shape of the PPF means that the marginal opportunity cost is increasing.

```
Plot[Abs[mrtExplicit], {fish, 0, 11}, ImageSize → Small]
```



Our second approach to the same question will not involve producing an explicit solution coconuts as a function of fish along the production-possibilities frontier. Instead, we will make use of implicit differentiation of the production-possibilities frontier. This time, instead of explicitly solving for the functional dependence of coconuts on fish along the PPF,

```
Solve[D[ppf /. {coconuts → coconuts[fish]}, fish], coconuts'[fish]]
mrtImplicit = coconuts'[fish] /. First[%];
```

$$\left\{\left\{\text{coconuts}'[\text{fish}] \to -\frac{25\,\text{fish}}{16\,\text{coconuts}[\text{fish}]}\right\}\right\}$$

While this is not as explicit, it has its own virtues: particularly the simplicity of the expression (in this particular case). And of course it does tell us that as we have more fish and correspondingly fewer coconuts, the marginal opportunity cost of fish is increasing. Finally, if we actually substitute our explicit solution for coconuts into this implicit solution, it naturally gives us the same description of marginal opportunity cost as before.

```
mrtImplicit /. {coconuts[fish] → soln02}
```

$$-\frac{5\,\text{fish}}{4\sqrt{128-\text{fish}^2}}$$

Here is one last approach to the same problem. Use the `Dt` command to totally differentiate the PPF, thereby establishing a relationship between the change in coconuts and the change in fish.

```
dppf = Dt[ppf]
```

$$0 == \frac{2}{25}\,\text{coconuts}\,\text{Dt}[\text{coconuts}] + \frac{1}{8}\,\text{fish}\,\text{Dt}[\text{fish}]$$

Setting the change in fish to unity, we can solve for the change in coconuts. This again yields the desired slope.

```
Solve[dppf /. {Dt[fish] → 1}, Dt[coconuts]]
```

$$\left\{\left\{\text{Dt}[\text{coconuts}] \to -\frac{25\,\text{fish}}{16\,\text{coconuts}}\right\}\right\}$$

The Rule of l'Hôpital

Derivatives can help us evaluate other limits. In this section we introduce l'Hôpital's rule, which helps us determine apparently indeterminate limits. (Following Stigler's law of eponymy, l'Hôpital's rule was apparently introduced to l'Hôpital by Johann Bernoulli.) Suppose $f[c] = g[c] = 0$ for differentiable functions f and g, and we are trying to evaluate $\lim_{x\to c} f[x]/g[x]$. This appears to be indeterminate, but if $\lim_{x\to c} f'[x]/g'[x]$ exists, it is also the limit for our original expression. Here is a first step towards a proof, assuming f and g are continuously differentiable.

$$\lim_{x\to c}\frac{f[x]}{g[x]} = \lim_{x\to c}\frac{f[x]-f[c]}{g[x]-g[c]} = \lim_{x\to c}\frac{(f[x]-f[c])/(x-c)}{(g[x]-g[c])/(x-c)} = \frac{f'[c]}{g'[c]} = \lim_{x\to c}\frac{f'[x]}{g'[x]} \tag{13}$$

A corresponding result applies if the function limits are infinite instead of 0.

As an application, we consider the Box-Cox transformation of a variable x. This is a popular variable transformation in econometrics.

```
ClearAll[τ, b]
τ = x ↦ (x^b - 1)/b;
```

As $b \to 0$, both the numerator and denominator approach 0. Let us apply l'Hôpital's rule. Here we use the `Numerator` and `Denominator` commands to extract the numerator and denominator of the Box-Cox transformation.

Expression	Result
num=Limit[D[Numerator[τ[x]],b],b→0]	Log[x]
den=Limit[D[Denominator[τ[x]],b],b→0]	1
num/den	Log[x]

We find that the Box-Cox transformation becomes the logarithmic transformation as $b \to 0$.

Higher-Order Derivatives

In our discussion of the first-order derivative, we showed that from a differentiable primitive function *f* we can derive a new function, called the derivative function. The derivative function may itself be differentiable. If it is, we refer to the derivative of the derivative function as the second-order derivative of the primitive function *f*, or simply as the second derivative. WL allows us to slightly modify the second argument of the `D` command to represent higher-order derivatives.

Expression	Result	Comment
D[f[x],x]	f′[x]	first-order derivative
D[f[x],{x,1}]	f′[x]	first-order derivative
D[f[x],{x,2}]	f″[x]	second-order derivative
D[f[x],{x,3}]	f⁽³⁾[x]	third-order derivative

Recall that the derivative *f*'[*x*] can be considered the slope of *f* at *x*. Similarly, the second-order derivative *f*''[*x*] can be consider the slope of *f*' at *x*, which is the curvature of *f* at *x*. That is, a function has a positive second-order derivative when its slope is increasing and a negative second-order derivative when its slope is decreasing. A function whose second-order derivative is always positive is therefore convex. Similarly, a function whose second-order derivative is always negative is concave. Here is a simple example or a function that switches from concave to convex (at $x = 0$).

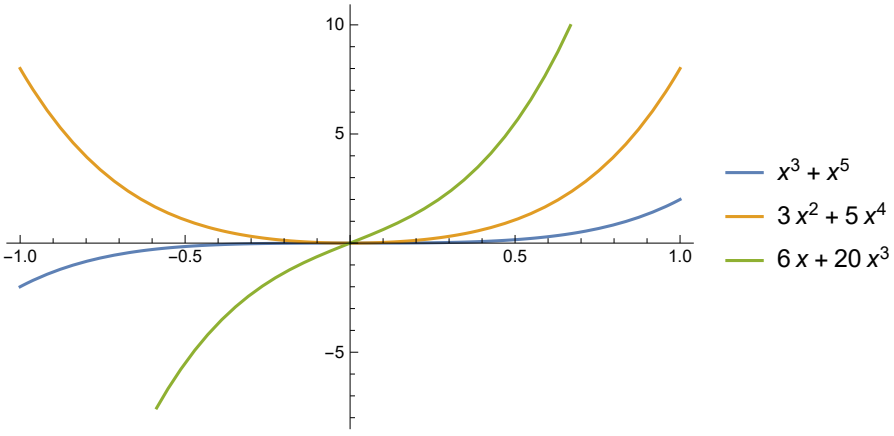
```
derivatives = D[x^3 + x^5, {x, #}] & /@ Range[0, 2];
Column[derivatives] (* successive derivatives *)
```

$x^3 + x^5$

$3\,x^2 + 5\,x^4$

$6\,x + 20\,x^3$

```
Plot[Evaluate@derivatives, {x, -1, 1},
 PlotLegends → "Expressions"]
```



Two functions we often encounter have rather interesting derivatives. The slope of the exponential function is the same as the value of the function, so this statement is true of each of its derivative functions as well.

```
D[ℯ^x, {x, #}] & /@ Range[5]
```

$$\{e^x, e^x, e^x, e^x, e^x\}$$

The slope of the logarithmic function is a power function, so this statement is true of each of its derivative functions as well.

```
D[Log[x], {x, #}] & /@ Range[5]
```

$$\left\{\frac{1}{x}, -\frac{1}{x^2}, \frac{2}{x^3}, -\frac{6}{x^4}, \frac{24}{x^5}\right\}$$

Note that the numerators in these results are factorials.

```
Factorial[#] & /@ Range[0, 4]
```

```
{1, 1, 2, 6, 24}
```

Recall the Box-Cox transformation, defined on the nonnegative reals give a positive value of the parameter *b*.

```
ClearAll[τ, b]
τ = x ↦ (x^b - 1)/b;
```

We can readily determine the first and second derivatives.

Expression	Result
D[τ[x],x]	x^{-1+b}
D[τ[x],{x,2}]	$(-1+b)\,x^{-2+b}$

We see that the function is concave for $b \in (0, 1)$. For these restricted values of *b*, the Box-Cox transformation is often used to represent consumer preferences over wealth outcomes. This popularity is largely due to a property known as constant relative risk aversion. The coefficient of relative risk aversion is just the elasticity of the derivative function.

```
D[τ[x], {x, 2}] * x / D[τ[x], x]
```

```
-1 + b
```

Taylor Series

WL can express a Taylor-series expansion of an arbitrary function *f* around a point *p*. For example, here is a third-order Taylor-series expansion.

```
Series[f[x], {x, p, 3}]
```

$$f[p] + f'[p]\,(x-p) + \frac{1}{2}f''[p]\,(x-p)^2 + \frac{1}{6}f^{(3)}[p]\,(x-p)^3 + O[x-p]^4$$

You can also do this for specified functions.

```
expSeries = Series[Exp[x], {x, 0, 3}]
```

$$1 + x + \frac{x^2}{2} + \frac{x^3}{6} + O[x]^4$$

```
CoefficientList[expSeries, x]
```

$$\left\{1, 1, \frac{1}{2}, \frac{1}{6}\right\}$$

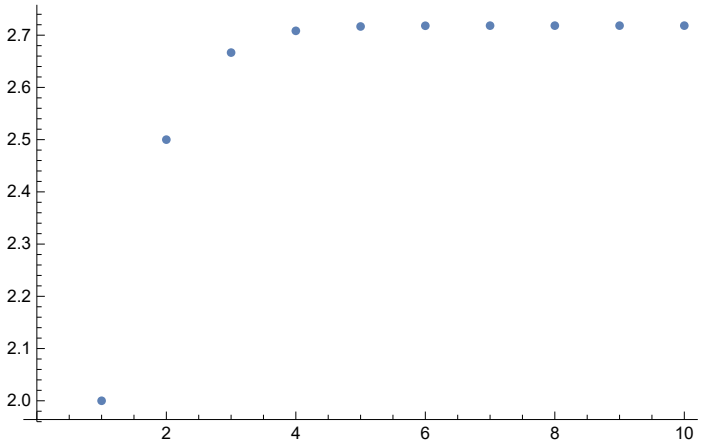
The output of the Series command is a SeriesData object. (Use the the InputForm or FullForm command to examine it.) If you need a normal expression, use the Normal command.

```
Normal[expSeries]
```

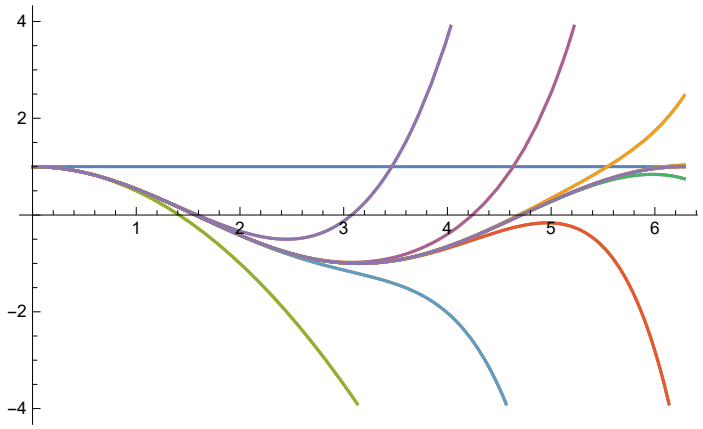
$$1 + x + \frac{x^2}{2} + \frac{x^3}{6}$$

```
Map[
   Function[{k}, Normal[Series[Exp[x], {x, 0, k}]]],
   Range[10]
  ] /. {x → 1} // N
ListPlot[%]
```

```
{2., 2.5, 2.66667, 2.70833, 2.71667, 2.71806, 2.71825, 2.71828, 2.71828, 2.71828}
```



```
(* Interesting example from Mathematica documentation *)
(* Note use of Normal; cannot directly plot a Series *)
Plot[Evaluate[Table[Normal[Series[Cos[x], {x, 0, n}]], {n, 20}]], {x, 0, 2 Pi}]
```



5.4 Logistic Map

In this section, we combine our interest in sequences and slopes by explore the behavior of the logistic function. Recall that the basic logistic function on the unit interval has the form $a\,x\,(1-x)$, given the amplitude parameter $a \in (0\,..4)$.

```
ClearAll[f1]
f1 = {a, x} ↦ a x (1 - x); (* parameterized logistic function *)
```

Changes in the amplitude parameter simply scale the function values. Here are a few examples. As we expect from the product rule of differentiation, we see that the slope of the function is correspondingly scaled by the amplitude parameter.

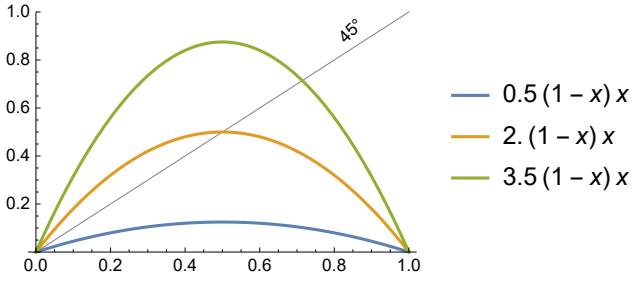


Figure 11: Logistic Function (Various Amplitudes)

We can visualize fixed points of a univariate function by looking at where it crosses the 45 ° line. We can also attempt to solve for them algebraically.

```
fixedPoints01 = Solve[x == f1[a, x], x]
```

$$\left\{\{x \to 0\}, \left\{x \to \frac{-1+a}{a}\right\}\right\}$$

For a logistic function, 0 is always a fixed point, regardless of the amplitude parameter. Given $a \in (0\,..1]$,

it is only fixed point in the unit interval. But for larger values of the amplitude parameter, we find a second fixed point at $(a - 1)/a$.

Given any initial value in the unit interval, iteration of a logistic function produces new values in the unit interval. We can view the resulting sequence as the outcome of a dynamical system, with the logistic function as our evolution rule. The fixed points of the logistic function are then the steady states of this dynamical system.

Use the `D` command to produce a symbolic derivative, allowing us to characterize the logistic function's slope for any given amplitude parameter.

```
Clear[a, x]
D[f1[a, x], x] // Simplify
```

```
a - 2 a x
```

The slope determines how a logistic sequence evolves when we are near but not at a steady state. Stability of a steady state requires that the evolution rule have a slope there that is less than unity in absolute value. Let us evaluate the logistic function's slope at the two steady states we have discovered (i.e., 0 and $1 - 1/a$).

```
D[f1[a, x], x] /. fixedPoints01 // Simplify
```

```
{a, 2 - a}
```

So for $a \in (0, 1)$, the fixed point at 0 is stable. But for $a \in (1 ..3)$, it is unstable, and there is a stable fixed point at $(a - 1)/a$. At higher values of the amplitude parameter, there are no stable fixed points

Any fixed point of a function f is of course also a fixed point of $f \circ f$. However, something interesting happens when the amplitude parameter exceeds 3, so that the logistic function no longer has a stable steady state. By examining the composition of the logistic function with itself, we can find two more fixed points of $f \circ f$.

```
ClearAll[f2]
f2 = {a, x} ↦ f1[a, f1[a, x]];
fixedPoints02 = Solve[x == f2[a, x], x] // Simplify
```

$$\left\{\{x \to 0\}, \left\{x \to \frac{-1+a}{a}\right\}, \left\{x \to \frac{1+a-\sqrt{-3-2a+a^2}}{2a}\right\}, \left\{x \to \frac{1+a+\sqrt{-3-2a+a^2}}{2a}\right\}\right\}$$

However, since we already knew about 0 and $1 - 1/a$, solving directly for all the fixed points was really doing too much work. Let us instead form the polynomial whose roots we're seeking and divide it by the polynomial whose roots we already found. First, let us characterize the quadratic polynomial whose roots are the fixed points of the logistic function.

```
Clear[x]
poly4p1 = Times @@ (x - (x /. fixedPoints01))
```

$$x\left(-\frac{-1+a}{a}+x\right)$$

We can make this easier to read if we use the `Collect` command to collect together coefficients on the different powers of x.

```
poly4fp1 = Collect[poly4fp1, x]
```

```
poly4fp1
```

The next step is where we take advantage of our understanding that fixed points of the f1 are also fixed points of f2: we use the `PolynomialQuotient` command to divide the polynomial whose roots are the fixed points of f2 by the polynomial whose roots are the fixed points of f1.

```
poly4fp2 = Collect[PolynomialQuotient[x - f2[a, x], x - f1[a, x], x], x]
```

$$1 + a + \left(-a - a^2\right) x + a^2 x^2$$

By construction, this must divide evenly, since the fixed points of f1 must also be fixed points of f2.

```
PolynomialRemainder[x - f2[a, x], x - f1[a, x], x]
```

```
0
```

Making use of this information left us with a much simpler problem: we just need to solve another quadratic in order to find the additional fixed points of f2.

```
Solve[poly4fp2 == 0, x]
```

$$\left\{\left\{x \to \frac{a + a^2 - a\sqrt{-3 - 2a + a^2}}{2a^2}\right\}, \left\{x \to \frac{a + a^2 + a\sqrt{-3 - 2a + a^2}}{2a^2}\right\}\right\}$$

Note that the discriminant is positive for $a > 3$, which is the case we are considering. We can see this particularly easily by factoring the discriminant as a quadratic in the amplitude parameter.

```
Factor[-3 - 2 a + a^2]
```

$$\left(-3 + a\right)\left(1 + a\right)$$

Note that we do not get new fixed points of $f \circ f$ until we reach $a = 3.0$. Then suddenly we have twice as many.

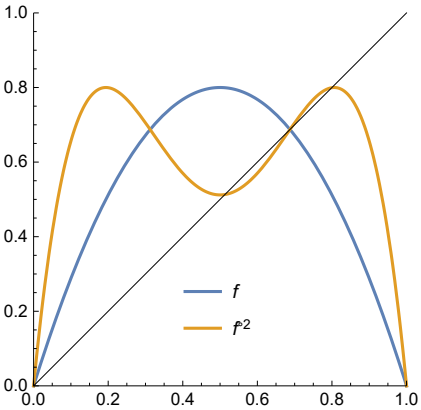


Figure 12: Logistic Function and Its First Iterate

What's more, although our old stable fixed point is now unstable, the two new points are initially stable. In an interesting bit of symmetry, the slope of $f \circ f$ is the same at each of the new fixed points.

```
D[f2[x], x] /. fixedPoints02 // Simplify
Plot[{4 + 2 a - a^2, 1, -1}, {a, 3, 4},
 PlotStyle → {Blue, {Gray, Dashed}, {Gray, Dashed}}]
Solve[4 + 2 a - a^2 == -1, a]
% // N
```

... Function: Too many parameters in {a, x} to be filled from Function[{a, x}, f1[a, f1[a, x]]][x].

... Function: Too many parameters in {a, x} to be filled from Function[{a, x}, 0][x].

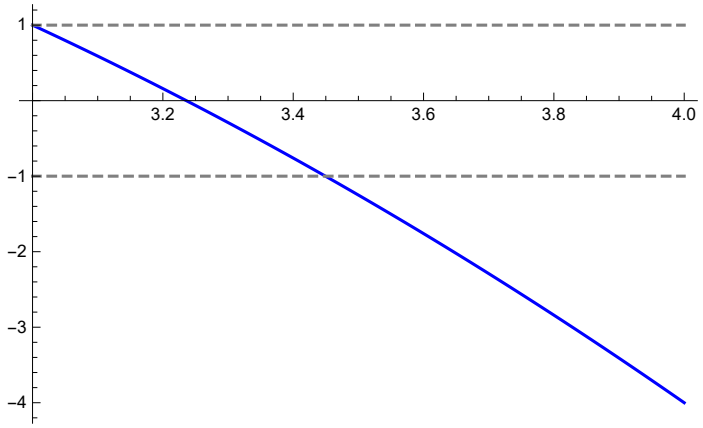
... Function: Parameter specification {a, 0} in Function[{a, 0}, 0] should be a symbol or a list of symbols.

... Function: Parameter specification {a, 0} in Function[{a, 0}, 0] should be a symbol or a list of symbols.

... Function: Parameter specification {a, 0} in Function[{a, 0}, f1[a, f1[a, 0]]] should be a symbol or a list of symbols.

... General: Further output of Function::flpar will be suppressed during this calculation.

$$\{\text{Function}[\{a, 0\}, 0][0] + \text{Function}[\{a, 0\}, f1[a, f1[a, 0]]]'[0],$$
$$\text{Function}\left[\left\{a, \frac{-1+a}{a}\right\}, 0\right]\left[\frac{-1+a}{a}\right] + \text{Function}\left[\left\{a, \frac{-1+a}{a}\right\}, f1\left[a, f1\left[a, \frac{-1+a}{a}\right]\right]\right]'\left[\frac{-1+a}{a}\right],$$
$$\text{Function}\left[\left\{a, \frac{1+a-\sqrt{-3-2a+a^2}}{2a}\right\}, 0\right]\left[\frac{1+a-\sqrt{-3-2a+a^2}}{2a}\right] +$$
$$\text{Function}\left[\left\{a, \frac{1+a-\sqrt{-3-2a+a^2}}{2a}\right\}, f1\left[a, f1\left[a, \frac{1+a-\sqrt{-3-2a+a^2}}{2a}\right]\right]\right]'\left[\frac{1+a-\sqrt{-3-2a+a^2}}{2a}\right], \text{Function}\left[\left\{a, \frac{1+a+\sqrt{-3-2a+a^2}}{2a}\right\}, 0\right]\left[\frac{1+a+\sqrt{-3-2a+a^2}}{2a}\right] +$$
$$\text{Function}\left[\left\{a, \frac{1+a+\sqrt{-3-2a+a^2}}{2a}\right\}, f1\left[a, f1\left[a, \frac{1+a+\sqrt{-3-2a+a^2}}{2a}\right]\right]\right]'\left[\frac{1+a+\sqrt{-3-2a+a^2}}{2a}\right]\}$$



$$\{\{a \to 1 - \sqrt{6}\}, \{a \to 1 + \sqrt{6}\}\}$$

$$\{\{a \to -1.44949\}, \{a \to 3.44949\}\}$$

We can again proceed in the same fashion. But it is $f^{\circ 4}$ not $f^{\circ 3}$ that provides the next periodic attractors. Naturally, any fixed point of $f^{\circ 2}$ is also a fixed point of $f^{\circ 4}$, but we may add new ones.

```
f3 = {a, x} ↦ Nest[f1[a, #] &, x, 3];
f4 = {a, x} ↦ Nest[f1[a, #] &, x, 4];
```

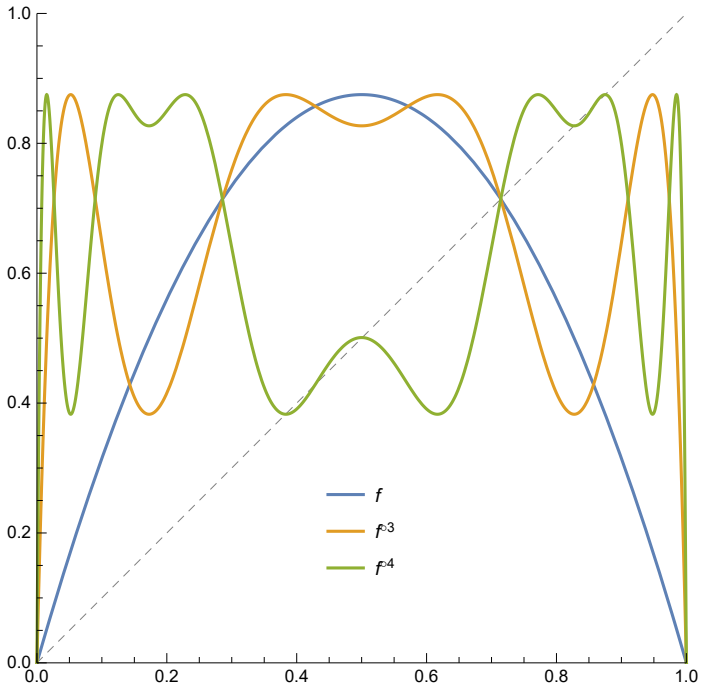


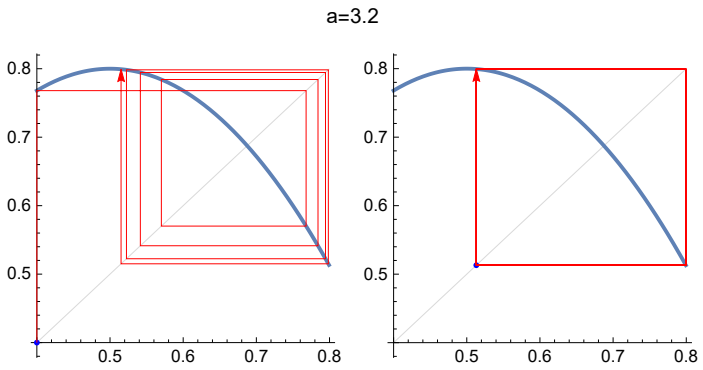
Figure 13: Logistic Function (a=3.5) and Some Iterates

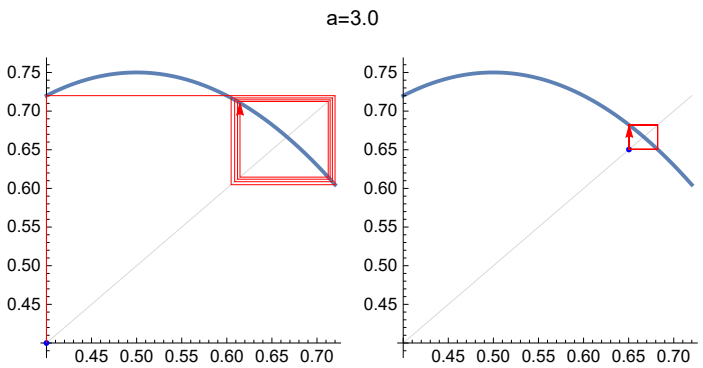
Cobweb Plots

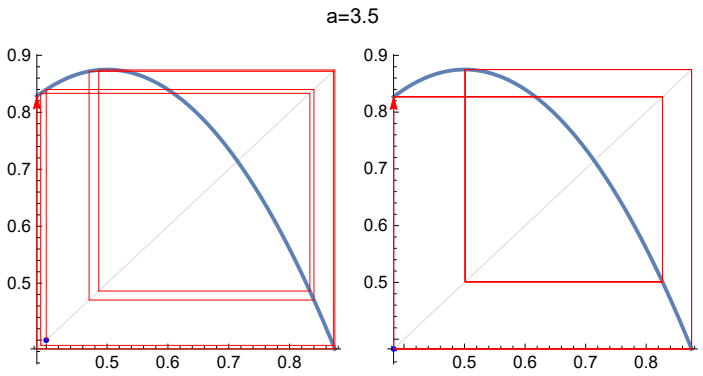
```
ClearAll[pts, min, max, f, x, x0, niter, cobwebPoints, cobwebPlot]
cobwebPoints[f_, x0_, niter_] := Module[{x = x0},
  Table[{{x, x}, {x, x = f[x]}}, {niter}] ~ Flatten ~ 1
 ]
cobwebPlot[f_, x0_, niter_, burn_: 0] := Module[{pts, min, max},
  pts = cobwebPoints[f, x0, niter + burn];
  min = Min@Flatten@pts;
  max = Max@Flatten@pts;
  Plot[f[x], {x, min, max},
   PlotStyle → {Thick},
   PlotRange → All,
   AxesOrigin → {min, min},
   AspectRatio → 1,
   Prolog → {Thin, LightGray, Line[{{min, min}, {max, max}}]},
   Epilog → {{Blue, Point[pts[[2 * burn + 1]]]}, Thin, Red, Arrow[pts[[2 * burn + 1 ;;]]]}
  ]]
```

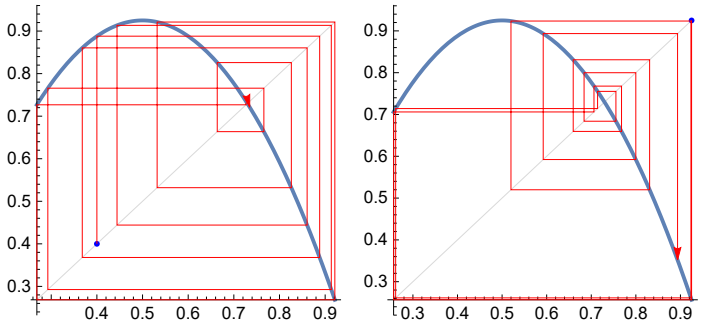
```
GraphicsRow[{cobwebPlot[3.2 # (1 - #) &, 0.4, 9], cobwebPlot[3.2 # (1 - #) &, 0.4, 9, 20]},
 PlotLabel → "a=3.2"]
GraphicsRow[{cobwebPlot[3.0 # (1 - #) &, 0.4, 9], cobwebPlot[3.0 # (1 - #) &, 0.4, 9, 200]},
 PlotLabel → "a=3.0"]

GraphicsRow[{cobwebPlot[3.5 # (1 - #) &, 0.4, 9], cobwebPlot[3.5 # (1 - #) &, 0.4, 9, 100]},
 PlotLabel → "a=3.5"]
GraphicsRow[{cobwebPlot[3.7 # (1 - #) &, 0.4, 15], cobwebPlot[3.7 # (1 - #) &, 0.4, 15, 100]}]
```









5.5 Univariate Optimization

WL provides a variety of optimization algorithms, which overlap in functionality. (See the Wolfram Language Guide entitled Optimization.) Our focus is on finding extrema: smallest and largest values of the function. Correspondingly, we are interested in finding the points in the domain where the function reaches these extrema.

Optimization by Incantation

The simplest approach to global optimization is to use `Minimize` or `Maximize`, which (in simple cases) only require as arguments an expression and a specification of the choice variable in that expression. The return value is a function value and a list of rules, specifying a point in the (real) domain where the function takes on the extreme value.

Expression	Result
`Minimize[x*x,x]`	$\{0, \{x \to 0\}\}$
`Maximize[-x*x,x]`	$\{0, \{x \to 0\}\}$
`Maximize[Sin[x],x]`	$\left\{1, \left\{x \to \frac{\pi}{2}\right\}\right\}$

In simple cases, it is even possible to do symbolic optimization. However, even in simple cases, the unspecified parameters can crucially affect the solution. For example:

```
{fmax, xmax} = Maximize[-a * (x - 2)^2, x]
```

$$\left\{\begin{cases} \infty & a < 0 \\ 0 & \text{True} \end{cases}, \left\{x \to \begin{cases} 2 & a > 0 \\ \text{Indeterminate} & a < 0 \\ 0 & \text{True} \end{cases}\right\}\right\}$$

In this case, we see that `fmax` is ∞ if $a < 0$, but otherwise is 0. Similarly, we have to consider cases when characterizing the optimizer. If $a = 0$, then any value of x is maximizing, but only the value $x = 0$ is returned. If $a < 0$, then both very positive and very negative values of x push the value of the function towards infinity, so that value of x is indeterminate. But if $a > 0$, then there is a unique global maximizer, $x = 2$.

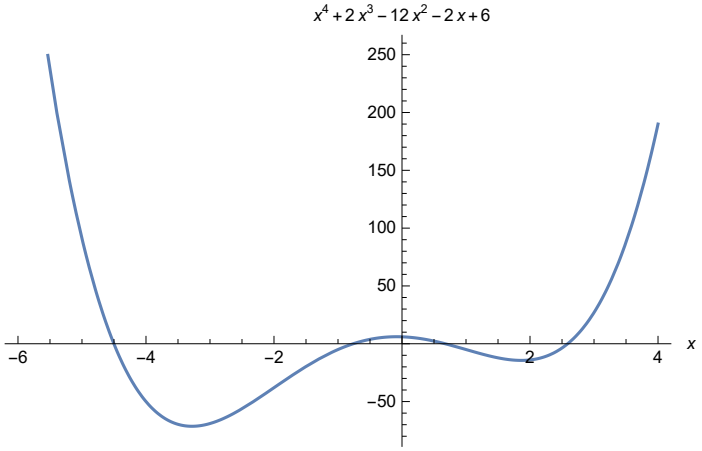
Most often we are interested in cases where we know enough about the parameters to ensure the existence of a maximum. We can force our symbolic solution to consider only the case of interest by using `Simplify` with a constraining assumption as the second argument.

```
Simplify[xmax, a > 0]
```

```
{x → 2}
```

The `NMaximize` and `NMinimize` functions work similarly but find approximate rather than exact solutions. In fact, `Maximize` and `Minimize` will invoke `NMaximize` and `NMinimize` if we pass an objective function containing approximate numbers. (An implication is that our symbolic evaluation above worked because we used the exact number 2 and not the approximate number 2.0 in our maximand.

A local extremum is reached at a point where the function value is extreme relative to any point nearby. The simplest approach to local optimization is to use `FindMaximum` or `FindMinimum`, which require a starting value for search. To see the implications of this, consider the function $f[x] = x^4 + 2\,x^3 - 12\,x^2 - 2\,x + 6$, illustrated below.



Recall that `Minimize` finds one global minimizer. We can apply `FindMinimum` to find either of the local minima, depending on the starting value we provide. Here we contrast the minimizer found by `FindMinimum` near −3 with that found near 3. In contrast, `Minimize` returns a global minimizer. However, as an example of the overlap in functionality of the optimization algorithms, we also illustrate the use of `Minimize` to find the other local minimum by adding a positivity constraint on the values of *x*.

Expression	Result
`FindMinimum[f[x],{x,-3}]`	`{-71.3728, {x → -3.28183}}`
`FindMinimum[f[x],{x,3}]`	`{-14.3969, {x → 1.86358}}`
`N@Minimize[f[x],x]`	`{-71.3728, {x → -3.28183}}`
`N@Minimize[{f[x],x>0},x]`	`{-14.3969, {x → 1.86358}}`

Calculus-Based Optimization

WL provides very good optimization algorithms. Unfortunately, using them does not generate much understanding in the user. This book therefore often uses other approaches to optimization, particularly standard calculus-based approaches to characterizing stationary points and function curvature. A first-order necessary condition for an extremum of a differentiable function *f* is that *f*'[*x*] == 0: we must be at a

stationary point. Once we know we are at a stationary point, then we can examine the curvature. If the function is locally concave at a stationary point, we have found a maximum. If the function is locally convex at a stationary point, we have found a local minimum. (If it is neither, we have found an inflection point.)

To illustrate, let us define an arbitrary polynomial to explore.

```
f = x ↦ x^3 - x;
```

Our first step will be to take the first derivative and set it to zero, thereby characterization the first-order necessary conditions for an extremum.

```
foc = f'[x] == 0
```

$$-1 + 3\,x^2 == 0$$

Since our function was a cubic, we will have to solve a quadratic to produce the solutions to the first-order condition. In WL, we can just use `Solve`, which returns a general representation of the solutions.

```
solns = Solve[foc, x, Reals]
```

$$\left\{\left\{x \to -\frac{1}{\sqrt{3}}\right\}, \left\{x \to \frac{1}{\sqrt{3}}\right\}\right\}$$

We see that two points in the domain are stationary points. But are they maxima, or minima, or inflexions? To determine this, we can examine the curvature of the function at each stationary point.

```
f''[x] /. solns
```

$$\left\{-2\sqrt{3}, 2\sqrt{3}\right\}$$

The second derivatives have determinate signs at the stationary points. So we have found a local maximum and a local maximum. We can verify these numerical results by plotting the function.

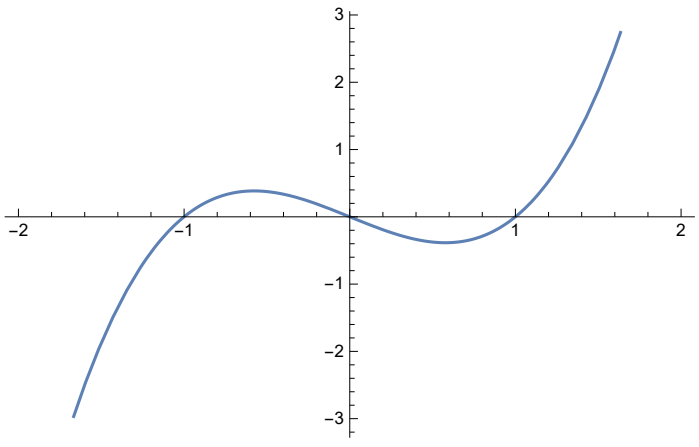


Figure 14: Plot of Cubic

Let us define another arbitrary polynomial to explore, this time one that is a bit more challenging.

```
f = x ↦ x^4 + 2 x^3 - 12 x^2 - 2 x + 6;
```

Our first step will be to take the first derivative and set it to zero, thereby characterization the first-order

necessary conditions for an extremum.

```
foc = f'[x] == 0
```

$-2 - 24\,x + 6\,x^2 + 4\,x^3 == 0$

Since our function was a quartic, we will have to solve a cubic to produce the solutions to the first-order condition. In WL, we can just use `Solve`, which returns a general representation of the solutions. We use the `N` command to convert these general representations to numerical representations.

```
solns = N@Solve[foc, x, Reals]
```

```
{{x → -3.28183}, {x → -0.0817535}, {x → 1.86358}}
```

We see that three points in the domain are stationary points. But are they maxima, or minima, or inflexions? To determine this, we can examine the curvature of the function at each stationary point.

```
f''[x] /. solns
```

```
{65.8627, -24.9008, 40.0381}
```

The second derivatives have determinate signs at the stationary points. So we have found two local minima and a local maximum. We can support these numerical results by plotting the function.

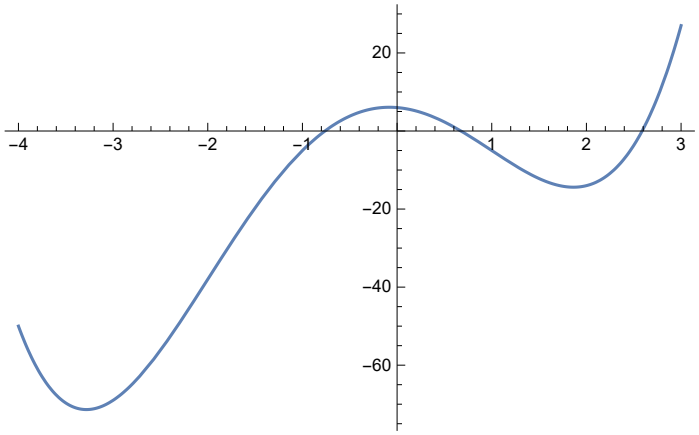


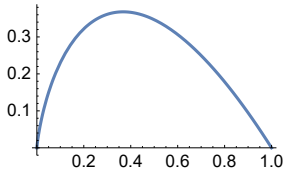
Figure 15: Plot of Quartic

5.6 Simple Applications

Maximum Entropy

For a discrete probability distribution characterized by $P(X = x_k) = p_k$, we define the "entropy" of the distribution to be $H(p) = -\sum_k p_k \text{Log}[p_k]$. The p_k naturally lie in the unit interval (and must sum to unity). We need to address the meaning of $p_k = 0$ in the expression for entropy, since Log[0] is undefined. We will adopt the limit from the right as the meaning of this expression. A simple plot reveals this to be 0.

```
Plot[-pk * Log[pk], {pk, 0, 1}, ImageSize → 144]
```



The `Limit` command confirms this.

```
Limit[-pk Log[pk], pk → 0, Direction → -1]
```

```
0
```

To demonstrate that this is the limit, we will use l'Hôpital's rule, which tells us that

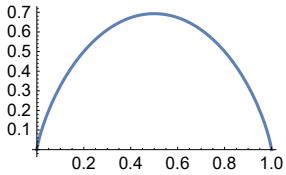
$$\lim_{x\to 0} -\frac{\text{Log}[x]}{1/x} = \lim_{x\to 0} -\frac{1/x}{-1/x^2} = \lim_{x\to 0} x = 0 \tag{14}$$

Note that we have a concave function that reaches a maximum at $p_k = 1/e$.

Expression	Result
D[-p Log[p],p]	$-1 - \text{Log}[p]$
D[-p Log[p],{p,2}]	$-\frac{1}{p}$

Now consider a categorical distribution with only two categories, whose probabilities are p_1 and p_2. Since $p_1 + p_2 = 1$, we adopt the notation $p_1 = p$ and $p_2 = 1 - p$. The entropy of this categorical distribution is therefore $-p \text{Log}[p] - (1 - p) \text{Log}[1 - p]$. Let us take a look at the plot of the entropy of this distribution as a function of p.

```
ent = p ↦ -p Log[p] - (1 - p) Log[1 - p];
Plot[ent[p], {p, 0, 1}, ImageSize → 144]
```



This looks nice concave and symmetrical; we guess that there is a maximum at $p = 0.5$. Maximizing this entropy confirms our guess.

```
Maximize[ent[p], p]
```

$$\left\{\text{Log}[2], \left\{p \to \frac{1}{2}\right\}\right\}$$

Rather than simply maximizing by incantation, we can of course invoke our understanding of differential calculus. First we find a stationary point at $p = 0.5$.

```
Solve[ent'[p] == 0, p]
```

$$\left\{\left\{p \to \frac{1}{2}\right\}\right\}$$

Then we confirm that the function is concave at the stationary point. In fact, it is globally concave on its

domain.

```
ent''[p]
```

$$-\frac{1}{1-p}-\frac{1}{p}$$

Laffer Curve

The so-called Laffer curve postulates that tax revenue is zero if the tax rate is 0% or 100%, but is positive in (0, 1). Typically the relationship between tax revenues and tax rates is assumed to be concave. Here is a simple algebraic representation of a Laffer curve, with a corresponding plot. (By positivity, we have $\theta > 0$.)

```
taxes = θ (t - t^2);
```

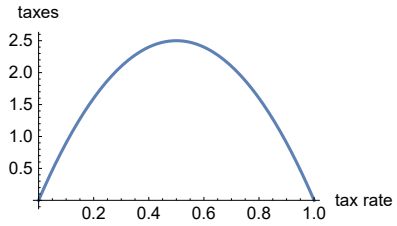


Figure 16: Laffer Curve (Θ=10)

Visually, it looks like the revenue-maximizing tax rate will be around 50%. Lets put our optimization tools to work and prove this.

Expression	Result	Comment
foc=D[taxes,t]==0	$(1-2\,t)\,\Theta == 0$	first-order condition
Solve[foc,t]	$\{\{t \to \frac{1}{2}\}\}$	potential extremum
D[taxes,{t,2}]	$-2\,\Theta$	confirm maximum

Some empirical work suggests the Laffer curve is not symmetric [trabandt.uhlig-2011-jme]_. We might represent a skewed Laffer curve as

```
taxes = θ t Sqrt[1 - t];
```

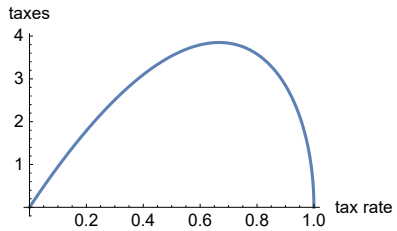


Figure 17: Skewed Laffer Curve (Θ=10)

The first-order condition is now slightly more complicated.

```
foc = D[taxes, t] == 0
```

$$\sqrt{1-t}\ \theta - \frac{t\,\theta}{2\sqrt{1-t}} == 0$$

We could just apply `Solve` to this equation, but in order to be more explicit about the production of the solution, we will manipulate this equation. In order to do so, it helps to understand that `foc` is an expression with a head of `Equal`.

```
Head[foc]
```

```
Equal
```

So this expression has a head of `Equal` and two elements (the left-hand side and right-hand side of the equation). We can `Map` over the elements of any expression, just like we map over the elements of a list. So we can multiply both sides of the equation by the denominator in the second term like this:

```
foc02 = 2 Sqrt[1 - t] * # & /@ foc
```

$$2\sqrt{1-t}\left(\sqrt{1-t}\ \theta - \frac{t\,\theta}{2\sqrt{1-t}}\right) == 0$$

It is a bit easier to see what we have produced if we use `Expand` to carry out the multiplications on the left of our new equation.

```
Expand[foc02]
```

$$2\theta - 3\,t\,\theta == 0$$

Clearly we have identified a stationary point at $t = 2/3$. Just to illustrate, we produce the solution explicitly.

```
solns = Solve[foc02, t]
```

$$\left\{\left\{t \to \frac{2}{3}\right\}\right\}$$

If we check the second-order condition, and we find that the curvature at the stationary point is negative (for any $\theta > 0$), so we have found a maximum.

```
D[taxes, {t, 2}] /. First@solns
```

$$-\frac{3\sqrt{3}\ \theta}{2}$$

Monopolist Maximization

We will put our understanding of maximization to work on a simple profit maximization example. We begin by formulating a simple demand equation, where the quantity demanded is a linear function of price. We will solve the problem of a profit maximizing monopolist facing this demand curve.

We begin by characterizing the demand curve facing this monopolist.

```
Clear[q, p]
qd = 150 - p / 2; (* demand as a function of price *)
```

Given the demand curve, we can easily compute the total revenue at each price: total revenue is just the product of the monopolist's price and the quantity demanded at that price. However, for reasons that will soon become clear, economists usually represent total revenue as a function of quantity instead of price. In order to do so, we can work with the inverse of the demand curve.

```
inverseDemand = First@Solve[q == qd, p]
```

$$\{p \to -2\ (-150 + q)\}$$

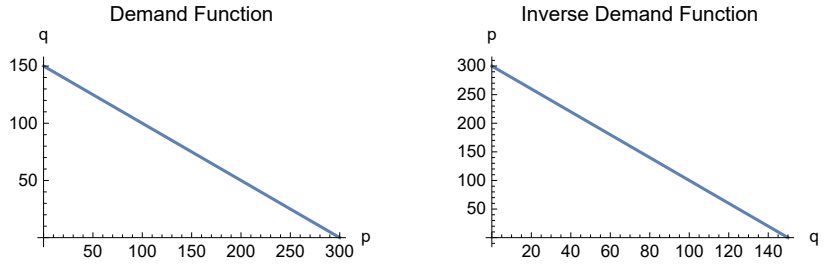


Figure 18: Demand and Inverse Demand

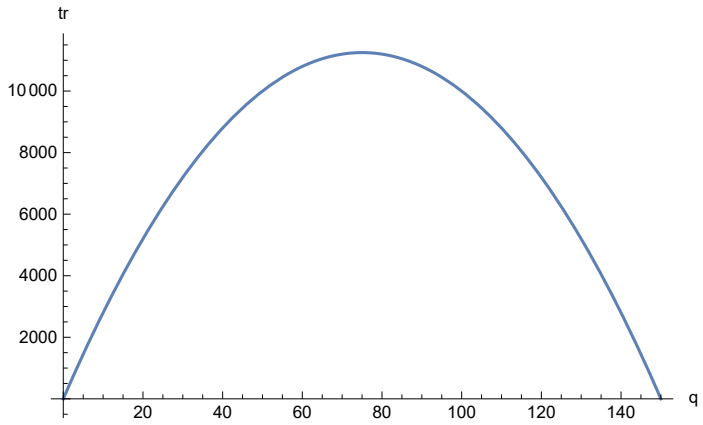
Given the inverse demand function, we can write total revenue as a function of quantity.

```
tr = p * q /. inverseDemand // Expand
```

$$300\ q - 2\ q^2$$

If we plot total revenue, we see that it reaches a maximum around 75.

```
Plot[tr, {q, 0, 150}, AxesLabel → {"q", "tr"}]
```



Let us find an exact maximizer (by incantation).

```
Maximize[tr, q]
```

$$\{11250, \{q \to 75\}\}$$

If the monopolist had no costs, revenue maximization would be a reasonable goal. How does this change in the face of rising marginal costs of production? In order to explore this, we will work with the following total-cost function.

```
tc = 5 + 40 * q + q^2 / 2; (* cost depends on quantity *)
```

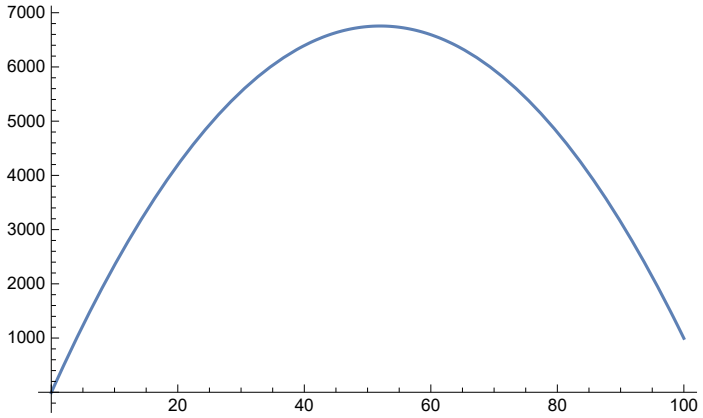
Since we have already characterized total revenue as a function of quantity, we can now express profits as a function of quantity: the difference between total revenues and total costs.

```
profits = tr - tc // Simplify
```

$$-\frac{5}{2}\left(2-104\,q+q^2\right)$$

A quick plot of the profit function suggests the monopolist should be producing around 50 units of its product.

```
Plot[profits, {q, 0, 100}]
```



Let us confirm this by maximizing by incantation. Since we are interested in a global extremum, we can use `Maximize`.

```
Maximize[profits, q]
```

```
{6755, {q → 52}}
```

We find that the profit maximizing quantify is less than the revenue maximizing quantity. Let us explore this a bit further based on our understanding of extrema. First, produce the necessary first-order condition. Then solve the first-order condition for the stationary point. Finally, check the curvature at the stationary point. (In this case the curvature is constant, so it is the same at the stationary point as everywhere else.) We find it is negative, so we have found a maximum.

Expression	Result
`profits`	$-\frac{5}{2}\left(2-104\,q+q^2\right)$
`foc=D[profits,q]==0`	$-\frac{5}{2}\left(-104+2\,q\right)==0$
`solns=Solve[foc,q]`	`{{q → 52}}`
`D[profits,{q,2}]`	`-5`

Average Cost and Marginal Cost

Expression	Result	Comment
tc=q^3-3*q^2+5*q+1	$1 + 5\,q - 3\,q^2 + q^3$	total cost
ac=tc/q//Apart	$5 + \frac{1}{q} - 3\,q + q^2$	average cost
mc=D[tc,q]	$5 - 6\,q + 3\,q^2$	marginal cost

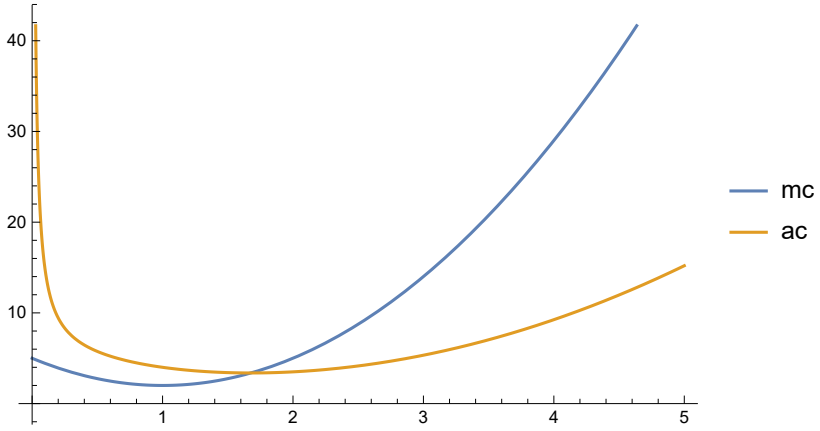


Figure 19: Average Cost and Marginal Cost

```
NSolve[mc == ac, q, Reals]
```

$$\{\{q \to 1.67765\}\}$$

```
NSolve[D[ac, q] == 0 && q > 0, q]
NMinimize[{ac, q > 0}, q]
```

$$\{\{q \to 1.67765\}\}$$

$$\{3.37763, \{q \to 1.67765\}\}$$

Let us explore this further with a different approach.

```
mc == ac // Simplify
q`intersect = Solve[mc == ac, q, Reals]
```

$$2\,q^2 == \frac{1}{q} + 3\,q$$

$$\left\{\left\{q \to \text{Root}\left[-1 - 3\,\#1^2 + 2\,\#1^3\ \&,\ 1\right]\right\}\right\}$$

We can request an exact expression for this.

```
ToRadicals[q`intersect]
```

$$\left\{\left\{q \to \frac{1}{6}\left(3 + \left(81 - 54\sqrt{2}\right)^{1/3} + 3\left(3 + 2\sqrt{2}\right)^{1/3}\right)\right\}\right\}$$

Alternatively, we can request a numerical value.

```
N[q`intersect]
```

```
{{q → 1.67765}}
```

If we want a numerical value, we can more simply start out by requesting a numerical solution.

```
NSolve[mc == ac, q, Reals]
```

```
{{q → 1.67765}}
```

Next, let us consider the minimum of the average cost curve. It looks like average cost reaches a minimum where the two curves intersect. Let us check. We can use the Minimize command, as long as we are careful to impose the positivity constraint on q.

```
Minimize[{ac, q ≥ 0}, q] // Simplify
N[%]
```

$$\left\{5 - 3\,\text{Root}\left[-4 - 3\,\#1^2 + \#1^3\ \&,\ 1\right] + \frac{3}{4}\,\text{Root}\left[-4 - 3\,\#1^2 + \#1^3\ \&,\ 1\right]^2,\ \left\{q \to \text{Root}\left[-1 - 3\,\#1^2 + 2\,\#1^3\ \&,\ 1\right]\right\}\right\}$$

```
{3.37763, {q → 1.67765}}
```

Let us try this again, this time by being explicit about the first order condition.

```
q`minac = Solve[D[ac, q] == 0, Reals]
q`minac == q`intersect
```

$$\left\{\left\{q \to \text{Root}\left[-1 - 3\,\#1^2 + 2\,\#1^3\ \&,\ 1\right]\right\}\right\}$$

```
True
```

Two more times, this time using NSolve and NMinimize.

```
NSolve[D[ac, q] == 0 && q > 0, q]
NMinimize[{ac, q > 0}, q]
```

```
{{q → 1.67765}}
```

```
{3.37763, {q → 1.67765}}
```

Bottle of Wine

`Maximize` or `Minimize` work especially well with polynomials, but can work with many functions.

```
exp01 = e^(-0.05*t) * e^Sqrt[t/2] ;
Maximize[{exp01, t > 0}, t]
```

```
{12.1825, {t → 50.}}
```

A strictly increasing transformation of the objective function naturally produces the same maximizer. Of course, the value of the objective function will be transformed.

```
exp02 = Log[exp01] // PowerExpand
Maximize[{exp02, t > 0}, t]
```

$$\frac{\sqrt{t}}{\sqrt{2}} - 0.05\,t$$

```
{2.5, {t → 50.}}
```

While the use of Maximize is convenient, we should also be able to approach the problem from first principles.

```
tmax = Solve[D[exp02, t] == 0, t] // Simplify
f2max02 /. tmax
```

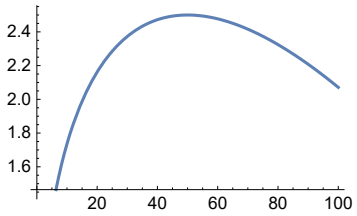
```
{{t → 50.}}
```

```
{f2max02}
```

Of course we should consider the curvature of the function as well. We can check that the second derivative is negative, or we can just look at a plot:

```
D[exp02, {t, 2}] < 0 /. tmax // Simplify
Plot[exp02, {t, 0, 100}, ImageSize → Small]
```

```
{True}
```



Let us quickly repeat the process using an explicit function rather than an expression in *t*.

Expression	Result	Comment
f=t⟼√(t/2) -t/20;		define maximand
foc=f'[t]==0//Simplify	$\frac{5\sqrt{2}}{\sqrt{t}} == 1$	first-order condition
soln=First@Solve[foc,t]	{t → 50}	stationary point
0>f''[t]/.soln	True	confirm maximum

Minimizing a Loss Function

```
L = -0.5 * (pi - pie) + pi^2
dL = D[L, pi]
Solve[dL == 0, pi]
D[dL, pi]
```

```
pi^2 - 0.5 (pi - pie)
```

```
-0.5 + 2 pi
```

```
{{pi → 0.25}}
```

```
2
```

```
pi^2 - 0.5 * (pi - pie)
```

```
pi^2 - 0.5 (pi - pie)
```

5.7 Univariate Integral Calculus

From a differentiable function $F[x]$ we can produce a derivative function $F'[x]$. The function $F[x]$ is called an antiderivative (or primitive) of $F'[x]$.

Indefinite Integral

Finding an antiderivative for a function f is called "integration" of f. Roughly speaking, integration is the inverse of differentiation. Note however that if $F[x]$ is an antiderivative of a function $f[x]$, then so is $F[x] + C$ for any constant C. The arbitrariness of this "constant of integration" suggests that some information is irrecoverably lost during the process of differentiation, and as a result the process of integration suggests an entire set of possible antiderivatives. We represent this set of antiderivatives by $\int f[x]\, dx$, an expression know as the indefinite integral of f. In WL, we use the `Integrate` command to produce an indefinite integral.

```
ClearAll[f]
Integrate[f[x], x]
```

$$\int f[x]\, dx$$

The function f is called the integrand, and the variable x in this expression is called the variable of integration. Note that the variable of integration is a bound variable: it is "used up" in the expression. For example, we could equally well express the indefinite integral as $\int f[z]\, dz$. Here are a few examples of indefinite integrals.

Expression	Result
Integrate[3x^2,x]	x^3
Integrate[3z^2,z]	z^3
Integrate[x^3,x]	$\frac{x^4}{4}$
Integrate[x^(-1),x]	Log[x]
Integrate[E^x,x]	e^x

We can always check that we have indeed found an antiderivative of a function *f* by differentiating it and confirming that this yields *f*.

Definite Integral

A definite integral includes limits of integration. For example, we write the definite integral of *f* over the interval [*a* .. *b*] as $\int_a^b f[x]\,dx$.

```
Integrate[f[x], {x, a, b}]
```

$$\int_a^b f[x]\,dx$$

Note the nice display that Mathematica gives to this expression. In fact, we can enter our integrals using this notation. (See the documentation of `Integrate` for details.)

We refer to *a* and *b* as the lower and upper limits of integration, and when this definite integral exists, we say that *f* is integrable over [*a* .. *b*]. When *f* is a real function, we interpret the definite integral as the (signed) area between the function graph and the *x*-axis. (More on that in a moment.) The first fundamental theorem of calculus ensures that we can compute this area in terms of any antiderivative *F*.

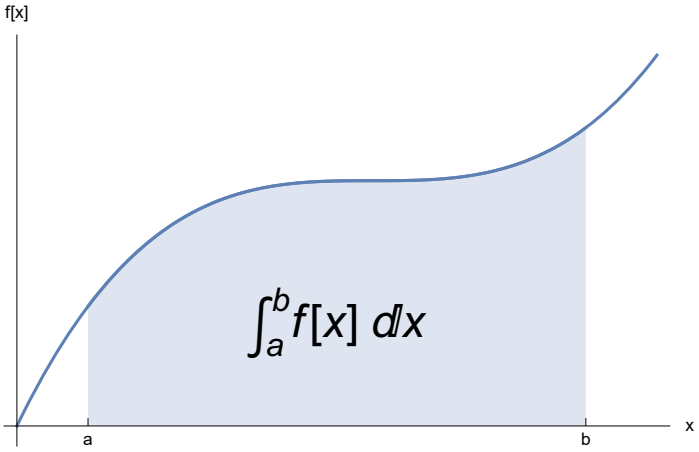
$$\int_a^b f[x]\,dx = F[b] - F[a] \tag{15}$$

To illustrate, we will work with a particular function.

```
f = x ↦ 125 + Power[x - 5, 3];
```

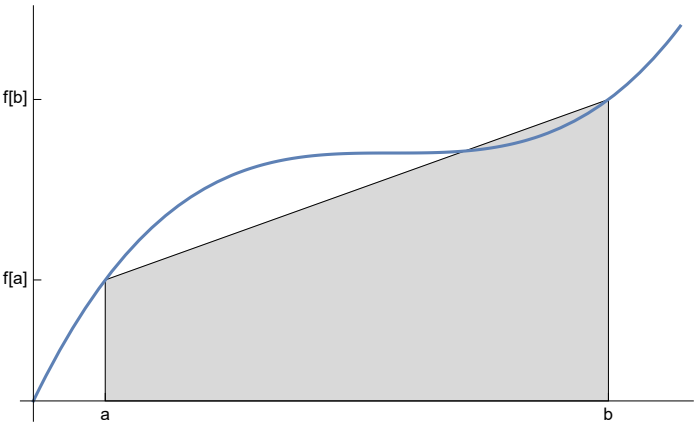
We are going to think about the (signed) area between the function graph and the *x*-axis, over an interval [*a* .. *b*]. This area is the definite integral, where *a* and *b* are the limits of integration. To keep things simple for now, we are going to focus on an interval in which the function takes on positive values.

```
{a, b} = {1, 8};
g1 = Plot[f[x], {x, 0, 9}, AxesLabel → {"x", "f[x]"},
    Ticks → {{{a, "a"}, {b, "b"}}, None}];
g2 = Plot[f[x], {x, a, b}, Filling → Axis, AxesOrigin → {0, 0}];
Show[g1, g2, Epilog → {Text[Style["∫ₐᵇf[x] dx", Large], Scaled[{0.5, 0.3}]]}]
```



Our first attempt to measure the area will be to approximate it with a right trapezoid that has vertexes (a, 0), (a, $f[a]$), (b, $f[b]$), and (b, 0). (Recall that a trapezoid, sometimes called a trapezium, is a convex quadrilateral with two parallel sides.) This requires two function evaluations: $f[a]$ and $f[b]$. In order to separate out concerns, we are going to create a function that describes a light-gray trapezoid.

```
ClearAll[fTrapezoid]
fTrapezoid = {f, a, b} ↦ With[{pts = {{a, 0}, {a, f[a]}, {b, f[b]}, {b, 0}}},
     {LightGray, EdgeForm[{Thin, Black}], Polygon[pts]}];
g1 = Plot[f[x], {x, 0, 9},
   Ticks → {{{a, "a"}, {b, "b"}}, {{f[a], "f[a]"}, {f[b], "f[b]"}}},
   Prolog → fTrapezoid[f, a, b]]
```



We know from basic geometry that the resulting area is $(b - a)\,(f[b] + f[a])/2$. This serves as an approximation to the area under the curve.

```
(b - a) * (f[b] + f[a]) / 2.
```

```
745.5
```

If our function were affine, this would give us an exact answer. But we can see from our plot that it is only an approximation. If we are satisfied, we can stop with this approximation. But we can also try to improve on our estimate by subdividing [*a* .. *b*], creating a trapezoid over each segment. Once we have all the subintervals, we can construct a right trapezoid for each subinterval. As we construct increasingly fine partitions of the interval [*a* .. *b*], it looks like the implied area approximation is approaching the true area. This is indeed the case.

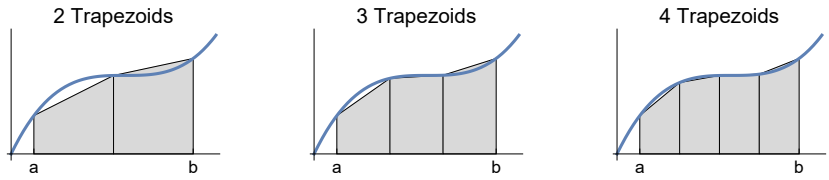


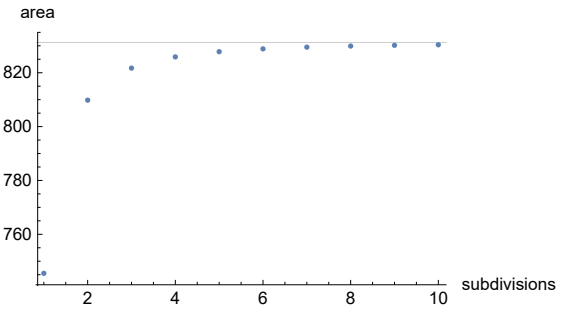
Figure 20: Area Approximations

Let us look at this numerically for our particular function and interval. We know how to compute the area of each trapezoid, so let us compute the total area of all of our trapezoids, which provides an approximation to area under the curve. This approach to approximating the area is called the composite trapezoid rule. Here we implement this rule using the `Sum` command, which sums values of its first argument implied by the iterator specification that is its second argument.

```
compositeTrapezoidRule = {f, a, b, n} ↦ With[{dx = (b - a) / n},
    ((f[a] + f[b]) / 2. + Sum[f[a + i * dx], {i, 1, n - 1}]) * dx];
```

In in this simple case, we do not need a very fine partition to get a good approximation to the area. To drive this point home, we will plot a few outcomes, using the `GridLines` option to `ListPlot` in order to include in our plot a grid line at the value of the actual integral.

```
estimates = compositeTrapezoidRule[f, a, b, #] & /@ Range[10];
ListPlot[estimates, AxesLabel → {"subdivisions", "area"},
 GridLines → {None, {Integrate[f[x], {x, a, b}]}},
 PlotRange → All, ImageSize → 288]
```



Note that the composite trapezoid rule can be thought of as the arithmetic average of two related rules: the left Riemann sum, and the right Riemann sum. These can also be considered approximations to the area under the curve.

```
leftRiemannSum = {f, a, b, n} ↦ With[{dx = (b - a) / n},
    Sum[f[a + i * dx], {i, 0, n - 1}] * dx];
rightRiemannSum = {f, a, b, n} ↦ With[{dx = (b - a) / n},
    Sum[f[a + i * dx], {i, 1, n}] * dx];
```

If we use a very fine partition, it does not matter which of these approximations we use. In the limit, they all approach the same value (for any piecewise continuous function), and that limiting value defines the area under the curve. That is, that value is the definite integral of the function over the interval under consideration.

Application: Probability

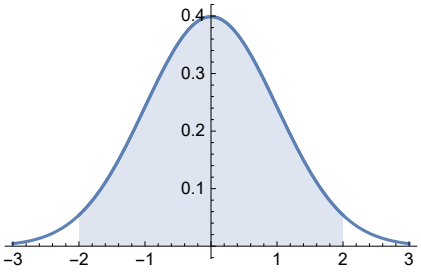
For a continuous distribution, the probability of falling in an interval is the associated area under the probability density function. Consider the PDF of the standard normal distribution.

```
pdfStandardNormal = PDF[NormalDistribution[0, 1]];
pdfStandardNormal[x]
```

$$\frac{e^{-\frac{x^2}{2}}}{\sqrt{2\,\pi}}$$

The probability that a normally distributed variable will fall within two standard deviations of its mean is represented by the following shaded area.

```
Show[Plot[pdfStandardNormal[x], {x, -3, 3}],
 Plot[pdfStandardNormal[x], {x, -2, 2}, Filling → Axis],
 ImageSize → 216]
```



Let us try to compute this using `Integrate`. The result may appear a bit obscure.

```
Integrate[pdfStandardNormal[x], {x, -2, 2}]
```

$$\text{Erf}\left[\sqrt{2}\right]$$

The problem is that there is not possible to express this integral in terms of familiar (elementary) functions. We sometimes say that there is no “closed form” for this integral. However, this and related results are needed often enough that they are often expressed in terms of a function made up specifically for this purpose, called the error function. (That is what the name `Erf` stands for.) If we want a numerical result, we simply use the `N` command to produce it.

```
N@Erf[√2]
```

```
0.9545
```

So a normally distributed variable falls within two standard deviations of its mean more than 95% of the time. Applying our composite trapezoid rule with 100 trapezoids yields a similar result. (Naturally, so do the left and right Riemann sums.)

```
compositeTrapezoidRule[pdfStandardNormal, -2, 2, 100]
```

```
0.954471
```

Application: Consumer Surplus

We often loosely interpret a demand curve characterizing a marginal willingness to pay: if a consumer demands q_0 at some price p_0, she must have been willing to pay p_0 for the last unit purchased. In this sense, along a demand curve, price paid measures marginal willingness to pay. The difference between total willingness to pay and the amount actually paid is called consumer surplus. We will find this by integration.

Let us explore this with a numerical example. We begin with a demand function, q_d, which represents quantity demanded as a function of price. Without worrying for now about how it was determined, we also set a price p_0 that is faced by consumers. At that price there is a corresponding quantity demanded, which we call q_0. Naturally, $q_0 = q_d[p_0]$.

```
qd = p ↦ 180 - 3 p / 4; (* qty demanded as a fn of price *)
p0 = 120; (* arbitrary price *)
```

Looking at the figure below, we see that we can compute the consumer surplus as an area under the demand function.

```
q0 = qd[p0]; (* qty demanded at p0 *)
pmax = p /. First@Solve[qd[p] == 0, p];
Integrate[qd[p], {p, p0, pmax}]
```

```
5400
```

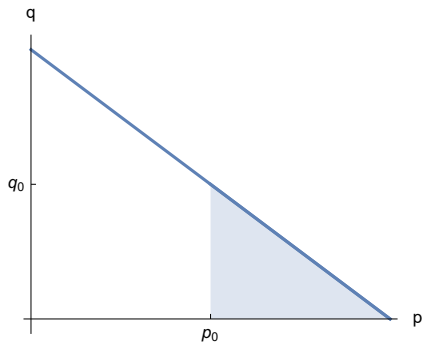


Figure 21: Consumer Surplus: Area under a Demand Function

For historical reason, this is not the most common representation of consumer surplus. Instead of working directly with the demand function, it is common to work with its inverse.

```
pd = InverseFunction[qd];
```

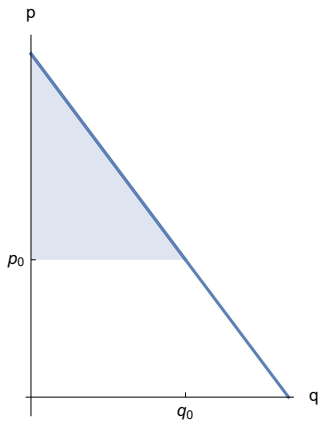


Figure 22: Consumer Surplus: Willingness to Pay Less Amount Paid

From this perspective, we find consumer surplus by finding the area under the demand curve, which represents the total willingness to pay, and subtracting out the amount actually paid, which is $p_0\, q_0$.

```
Integrate[pd[q], {q, 0, q0}] - p0 * q0
```

```
5400
```

Improper Integrals

"Monstrous indeed is the madness of men, who desire to subject the immeasurable to the puny measure of their own reason!" -- John Calvin

An improper integral is basically a definite integral with one or more problematic infinities in its description. There are two basic types of improper integrals: infinite bounds of integration, or singularities (approaching infinite function values) within the bounds of integration. In either case, we approach evaluation of the integral by describing the limit of a sequence of proper integrals. If this sequence converges, we say the improper integral converges. Otherwise, we say it diverges.

Example: evaluate $\int_1^\infty f[x]\,dx$ where $f[x] = x^{-2}$ by finding the antiderivative $F[x] = -x^{-1}$, evaluating $F[b] - F[1] = 1 - 1/b$, and then noting that this expression approaches a limiting value of 1.

```
Integrate[x^-2, {x, 1, ∞}]
```

```
1
```

An improper integral may well diverge.

Example: evaluate $\int_1^\infty f[x]\,dx$ where $f[x] = x^{-1}$ by finding the antiderivative $F[x] = \text{Log}[x]$, evaluating $F[b] - F[1] = \text{Log}[b]$, and then noting that this expression is unbounded in b. If we use WL to request an integral in this situation, we get back a warning.

```
Integrate[x^-1, {x, 1, ∞}]
```

```
Integrate: Integral of 1/x does not converge on {1, ∞}.
```

$$\int_1^\infty \frac{1}{x}\,dx$$

An improper integral may converge even though it both limits of integration are unbounded. For example, any continuous univariate probability density function must have an integral of 1 over the reals.

```
Integrate[PDF[NormalDistribution[0, 1]][x], {x, -∞, ∞}]
```

```
1
```

Application: Discounted Present Value

Consider a continuous income stream arriving at rate $f[t]$ for $t \in [0, \tau]$. We want to compute the present value of this income stream, given a rate of discount r. To do so, we compute the definite integral of $e^{-rt} f[t]$ over $[0, \tau]$. We will measure time in years, and we first consider an income stream arriving at a constant rate of $1000 per year.

```
ClearAll[f]
```

```
pv = ∫_0^τ ℯ^(-r t) 1000 𝕕t
```

$$\frac{1000 \left(1 - e^{-r\, \text{Function}\left[x, \frac{x^{b-1}}{b}\right]}\right)}{r}$$

We see that the value depends on the length of the stream. A longer income stream naturally has a larger present value. We can consider the value of a perpetuity as the limiting value of this this expression.

```
Limit[pv, τ → ∞, Assumptions → {r > 0}]
```

$$\frac{1000}{r}$$

Let us next consider a 10-year income stream.

```
pv10 = pv /. {τ → 10}
```

$$\frac{1000 \left(1 - e^{-10\, r}\right)}{r}$$

The value of the income stream also depends critically on the discount rate.

```
Plot[pv10, {r, 0, 0.10}, ImageSize → 216]
```