Part II. Calculations and Calculus with *Mathematica*

Section 3. Functions as Transformations

Functions are introduced in calculus in the general form $y = f(x)$. The letters x and y are called the independent and dependent variables respectively and the letter f is often referred to as the "function rule". In this manual we will refer to f as a function or sometimes a function rule or a transformation rule or as a transformation. Many applications of calculus are easier to handle when the relevant functions are entered as transformations.

Using a transformation to define a function

Consider the function $y = x^2 \sin(x)$ that was introduced in the calculus example in Section 2.

Another calculus example: Find the tangent line to the graph of this function at $x = 2$ and plot it.

Begin with the definition of the function as a transformation using what is called "function notation". Read the input as "f transforms x to x^2*sin(x)".

```
f[x_] := x^2 Sin[x]
```

Note that the function is defined using a combination of a color and an equals sign. We will refer to this as the "assignment operator". (No space between the colon and the equals sign, please.) The function is named f, but any name can be used. Likewise, although the letter x was chosen to denote the independent variable, any other symbol would serve exactly the same purpose. The input x on the left of the definition appears with an underscore, the appearances of x are not underscored.

Having done this, the value of f when x = a is obtained by entering f[a]. See the following examples.

```
{ f[0], f[1], f[2], f[3] }
N[%]
```

$\{0, \text{Sin}[1], 4\,\text{Sin}[2], 9\,\text{Sin}[3]\}$

$\{0., 0.841471, 3.63719, 1.27008\}$

The following plot shows the graph of f over the interval [0,3].

```
Plot[ f[x], {x,0,3} ]
```
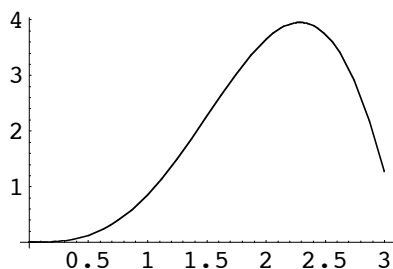

The derivative function, also as a transformation, is obtained with the entry **f '**, the second derivative is **f ''** and so on.

```
f'[x]
```

$x^2\,\text{Cos}[x] + 2\,x\,\text{Sin}[x]$

```
f''[x]
```
$4 x \, \text{Cos}[x] + 2 \, \text{Sin}[x] - x^2 \, \text{Sin}[x]$

Here is the derivative of f at x = 2, computed exactly and then approximately.

```
f'[2]
N[%]
```
$4 \, \text{Cos}[2] + 4 \, \text{Sin}[2]$

1.9726

This is the slope of the tangent line to the graph of f at the point (2 , f(2)).

The function whose graph is the tangent line at (2 , f(2)) is defined in the next entry. We name it T.

```
T[x_] := f[2] + f'[2]*(x - 2)
```

The entry T[x] will display the formula for the line in exact terms. Apply N to see the formula in decimal form.

```
T[x]
N[%]
```
$4 \, \text{Sin}[2] + (-2 + x) \, (4 \, \text{Cos}[2] + 4 \, \text{Sin}[2])$
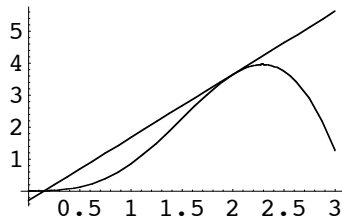
$3.63719 + 1.9726 \, (-2. + x)$

The next plot shows the graph of f and T together over the interval from 0 to 3.

```
Plot[ {f[x],T[x]}, {x,0,3} ]
```


Tangent lines are usually plotted over shorter intervals. This can be accomplished by applying Show to two plots.

Note. As was the case with Show in the last section the following entry produced three plots. We deleted the first two. As an alternative you can make the plots with DisplayFunction -> Identity and then add DisplayFunction -> $DisplayFunction to the Show entry. We will not say this anymore.

```
Show[ Plot[ f[x], {x,0,3} ], Plot[ T[x], {x,1.5,2.5}] ]
```
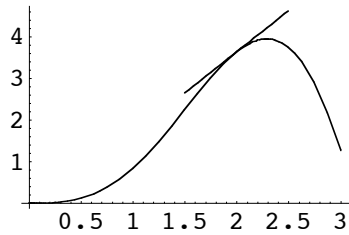
Plot several tangent lines

Several tangent lines can be plotted by defining the function L so that L[a,x] is the formula for the tangent line to the graph of f at the point (a,f(a)). L is a function of two variables. See below.

```
L[a_,x_] := f[a] + f'[a]*(x - a)
```

The Table function can then be used to make a list of tangent line plots at points evenly spaced along the curve. The list is named Lines (output suppressed). There are four of them. (Here we use DisplayFunction -> Identity.)

```
Lines = Table[ Plot[ L[a,x], {x,a-0.3,a+0.3},
                      DisplayFunction -> Identity ],
                 {a,0,3} ];
```

Now put Lines into the Show function (along with the graph of f(x)). The additional PlotRange specification is needed to force display of the combined plot over its full range.

```
Show[ {Plot[f[x],{x,0,3}], Lines},
          DisplayFunction -> $DisplayFunction,
          PlotRange->{{0,3},{0,4.5}} ]
```
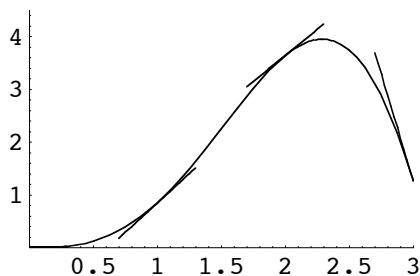

The next plot is a little fancier. The axes have been removed (Axes -> None) and the option PlotStyle -> Thickness[0.01] draws the four tangent line segments twice as thick. A title has also been added to the plot. The text for a title must be placed inside of double quotes.

```
NewLines = Table[ Plot[ L[a,x], {x,a-0.3,a+0.3},
                        PlotStyle -> Thickness[0.01],
                        DisplayFunction -> Identity ],
                  {a,0,3} ];
Show[ {Plot[f[x],{x,0,3}], NewLines},
           DisplayFunction -> $DisplayFunction,
           PlotRange->{{0,3},{0,4.5}},
           Axes -> None, PlotLabel → "Four Tangent Lines" ]
```

Four Tangent Lines
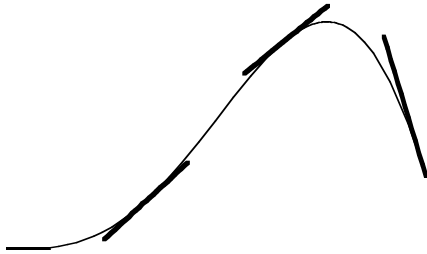
Find the length of the curve

The integral for the length of a curve defined by $y = f(x)$ over the interval $x = a$ to $x = b$ is

$$\int_a^b \sqrt{1 + (f'(x))^2}\ dx$$

Arc length integrals are notoriously difficult to evaluate exactly, and this one is no exception.

```
ArcLength = Integrate[Sqrt[1 + f'[x]^2], {x,0,3}]

$Aborted
```

As good as *Mathematica* is with integration it is unable to find an antiderivative Here is a 6 digit approximation.

```
ArcLength = NIntegrate[Sqrt[1 + f'[x]^2], {x,0,3}]

7.65897
```

Note that the numerical integration function is named **NIntegrate**.

Let's see how this compares to the sum of the lengths of the three secant line segments determined by the four points (k,f(k)), k = 0, 1, 2, 3. The kth segment is the hypotenuse of a right triangle with base 1 and side length

$$| f(k+1) - f(k) |, k = 0, 1, 2.$$

Add the lengths using the Sum function, then evaluate in floating point form.

```
SecantApproximation = Sum[ Sqrt[1 + (f[k+1]-f[k])^2], {k,0,2}];
N[%]
```
  6.84579

Now approximate the arc length with the sum of the lengths of three tangent line segments. The kth segment will be the tangent line at the point (0.5+k,f(0.5+k)) stretching over a unit interval on the x axis. Right triangles with base 1 can still be used, but the side of the triangle has length (plus or minus) f'(0.5+k). You may recognize the following addition formula as a midpoint approximation to the integral.

```
TangentLineApproximation = Sum[ Sqrt[1 + f'[0.5+k]^2], {k,0,2}];
N[%]
```
 6.77577

Both approximations are too small, but they are fairly close to one another. Let's try to see why.

The following picture shows the three tangent line segments used for TangentLineApprox.

```
TanLines = Table[ Plot[ L[0.5+k,x], {x,k,k+1},
                        PlotStyle -> Thickness[0.01],
                        DisplayFunction -> Identity ],
                  {k,0,3} ];
Show[ {Plot[f[x],{x,0,3}], TanLines},
             DisplayFunction -> $DisplayFunction,
             PlotRange->{{-0,3},{-0.5,4.5}},
             Axes -> None, PlotLabel → "Tangent Lines" ]
```


Tangent Lines
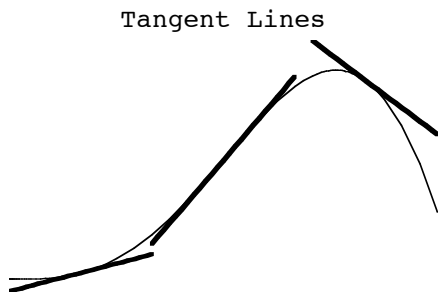
The three secant lines are shown next. They are drawn using ListPlot with the option PlotJoined -> True. The kth line joins (k,f(k)) to (k+1,f(k+1)), k=0, 1, 2.

```
SecLines = ListPlot[ Table[{k,f[k]}, {k,0,3}], PlotJoined → True,
                           PlotStyle → Thickness[0.01],
                           DisplayFunction -> Identity ];
Show[ {Plot[f[x],{x,0,3}], SecLines},
           DisplayFunction -> $DisplayFunction,
           PlotRange->{{-0.5,3},{0,4.5}},
           Axes -> None, PlotLabel → "Secant Lines" ]
```

Secant Lines
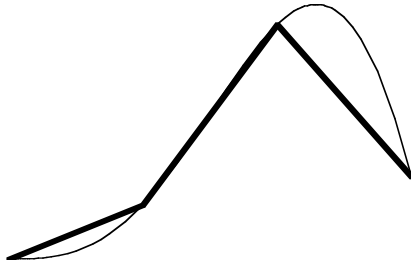
The pictures make it clear why the approximations are too small and are roughly the same.

An area calculation

Find the area between the graph of f and the graph of the secant line joining the endpoints of the graph over the interval from 0 *to* 3.
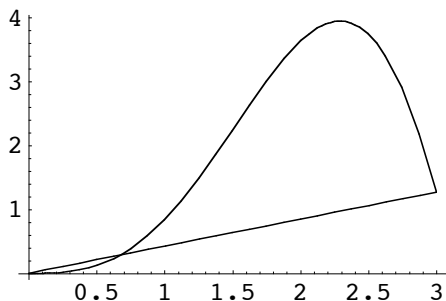
The function defined by the secant line will be named S. It's graph is a line through the origin with slope

$$\frac{f(3) - f(0)}{3 - 0}$$

```
S[x_] := (f[3] - f[0])/(3 - 0)*x
```

We want to find the area of the region between the two curves shown below.

```
Plot[ {f[x], S[x]}, {x,0,3} ]
```


The area equals the integral of the absolute value of f(x) - S(x) from x = 0 to x = 3. *Mathematica* uses Abs[x] for the absolute value of x. The integral is calculated numerically.

```
Area = NIntegrate[ Abs[f[x] - S[x]], {x,0,3}]
```
    3.96581

In an attempt to find an exact area formula we might try to get the exact value for the x coordinate of the point of intersection of the two graphs near x = 0.5. Try the Solve function..

```
Solve[ f[x] == S[x], x]
```

    Solve::ifun :
     Inverse functions are being used by Solve, so some solutions may
        not be found; use Reduce for complete solution information. More…
    {{x → 0}}

Mathematica found the solution x = 0. It did not find x = 3 nor did it find the exact x value we want (according to the graph a little to the right of x = 0.5). Try FindRoot with a specified value for x.

```
FindRoot[ f[x] == S[x], {x,0.5}]
```
    {x → 0.676348}

Let's name the point b.

```
b = x/.%
```
    0.676348

and use b to do the integration "exactly".

```
Area2 = Integrate[S[x]-f[x],{x,0,b}] + Integrate[ f[x]-S[x],{x,b,3}]
```
    3.96581

Mathematica was able to integrate exactly, but the answer is still approximate because *b* is in floating point form.

Higher Derivatives

The prime notation for derivatives can become tedious for order higher than 3. *Mathematica* provides a way to obtain higher derivatives using the D operator. Here, for example, is how to obtain the formula for the fourth derivative of the function f.

```
D[f[x],{x,4}]
```
    $-8 x \cos[x] - 12 \sin[x] + x^2 \sin[x]$

As an example using higher derivatives consider the following formula for the function P that is the second order Taylor polynomial approximation to the function f at the point a = 2.

```
P[x_] := f[2] + f'[2]*(x - 2) + 1/2*f''[2]*(x - 2)^2
```

This is referred to as a quadratic approximation. Enter P(x) to see the exact formula and apply N to see the quadratic in floating point form.

```
P[x]
N[%]
```
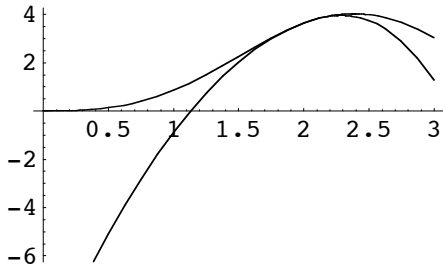
$$\frac{1}{2} (-2 + x)^2 (8 \cos[2] - 2 \sin[2]) + 4 \sin[2] + (-2 + x) (4 \cos[2] + 4 \sin[2])$$

$$3.63719 + 1.9726 (-2. + x) - 2.57388 (-2. + x)^2$$

The quadratic approximation plots like this.

```
Plot[ {f[x], P[x]}, {x,0,3}]
```


Near $x = 2$ the approximation is very good. The next plot uses the option Frame -> True to obtain the kind of picture that is popular in engineering texts. The function and the approximation are plotted using the dashing PlotStyle. Note that Dashing[{0,0}] is just a solid line. (I do not know why there is a vertical line at x = 1.6.)

```
Plot[ {f[x], P[x]}, {x,1.5,2.5}, Frame -> True,
      PlotStyle → {Dashing[{0,0}],Dashing[{0.03,0.03}]}]
```
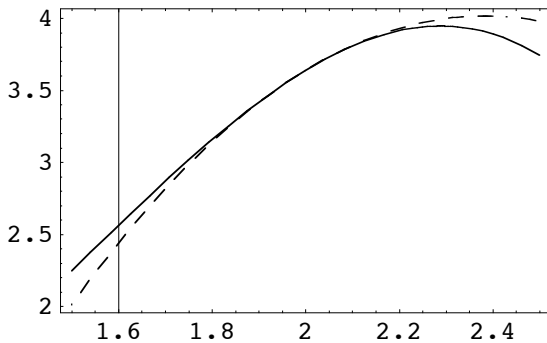

The function called Series will output the Taylor series and an error term for the function f. The syntax is shown below. The Normal function removes the error term and N evaluates the coefficients numerically..

```
Series[ f[x], {x,2,2} ]
Normal[%]
N[%]
```

$4 \, \text{Sin}[2] + (4 \, \text{Cos}[2] + 4 \, \text{Sin}[2]) \, (x - 2) + (4 \, \text{Cos}[2] - \text{Sin}[2]) \, (x - 2)^2 + O[x - 2]^3$

$(-2 + x)^2 \, (4 \, \text{Cos}[2] - \text{Sin}[2]) + 4 \, \text{Sin}[2] + (-2 + x) \, (4 \, \text{Cos}[2] + 4 \, \text{Sin}[2])$

$3.63719 + 1.9726 \, (-2. + x) - 2.57388 \, (-2. + x)^2$

Unevaluated derivatives

If an undefined function, say y(x), is differentiated, then *Mathematica* simply returns the derivative formula. See below.

```
y'[x]
```
$y'[x]$

```
D[y[x],x]
```
$y'[x]$

The same is true for higher order derivatives.

```
y''''[x]
```
$y^{(4)}[x]$

```
D[y[x],{x,4}]
```
$y^{(4)}[x]$

The chain rule

Mathematica is smart about differentiation. For example, the following entry shows that *Mathematica* knows the chain rule.

```
D[g[x[t]],t]
```
$g'[x[t]] \, x'[t]$

The output reads as "the derivative of g evaluated at x(t), multiplied by the derivative of x(t) with respect to t". This is the chain rule. The next entry illustrates one of many multivariable chain rule formulas: g is now a function of two variables, x and y are each functions of one variable.

```
D[g[x[t],y[t]],t]
```
$y'[t] \, g^{(0,1)}[x[t], \, y[t]] + x'[t] \, g^{(1,0)}[x[t], \, y[t]]$

The expression $g^{(0,1)}[x[t], y[t]]$ in the output is *Mathematica*'s way of denoting the partial derivative of g with respect to its second variable evaluated at the point (x(t) , y(t)). Similarly, $g^{(1,0)}[x[t], y[t]]$ is the partial derivative of g with respect to its first variable evaluated at the point (x(t) , y(t)).

Here is one more example: g is a function of 3 variables. x and p are both functions of one variable. The input asks for the derivative of g(x(p(t)), p(t), t) with respect to t. The output is the chain rule formula for this derivative.

```
D[ g[x[p[t]],p[t],t],t]
```

$g^{(0,0,1)}[x[p[t]], p[t], t] + p'[t] g^{(0,1,0)}[x[p[t]], p[t], t] +$
$\quad p'[t] x'[p[t]] g^{(1,0,0)}[x[p[t]], p[t], t]$

Just a little bit about differential equations

Unevaluated derivatives are exactly what are needed to define a differential equation. The next entry defines what is called a first order, linear, differential equation. The equation is given the name DE. Note the use of the double equals signifying that the equation is actually an identity, presumed valid for all t in some open interval.

```
DE = y'[t] + y[t] == t
```

$y[t] + y'[t] == t$

The unknown function is *y*, it is a function of the variable *t* and everywhere it appears in the equation it must be entered as y[t].

The next equation, named DE2, is called a Bernoulli equation. The unknown function is *x*, it is also a function of *t* and must be entered as x[t].

```
DE2 = x'[t] + x[t] == t^2*x[t]^3
```

$x[t] + x'[t] == t^2 x[t]^3$

Mathematica knows all about these equations. For example, it can solve both of them symbolically using a function called DSolve. To obtain the solution to DE just enter

DSolve[DE. y, t]

```
DSolve[ DE, y, t ]
```

$\{\{y \rightarrow \text{Function}[\{t\}, -1 + t + e^{-t} C[1]]\}\}$

The output is the solution as a "pure function" in a form suitable for substitution. The symbol C[1] denotes an arbitrary constant that arises in the solution process.

As a "pure function" the output can be substituted into expressions and turned into a named function easily. See below.

The next entry asks for the solution to DE2 satisfying the condition x(0) = 1. The solution equation is named soln.
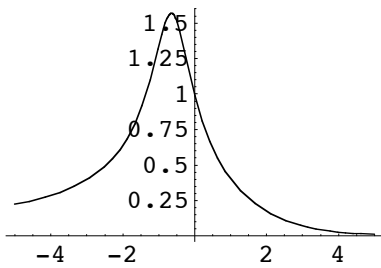
```
soln = DSolve[ { DE2, x[0]==1 }, x, t ]
```

DSolve::bvnul : For some branches of the general solution,
    the given boundary conditions lead to an empty solution. More…

$\left\{\left\{x \rightarrow \text{Function}\left[\{t\}, \dfrac{\sqrt{2}}{\sqrt{1 + e^{2t} + 2t + 2t^2}}\right]\right\}\right\}$

Here is the graph of the solution. Note the use of the /. construction to substitute the solution into x[t].

```
Plot[ x[t]/.soln, {t,-5,5} ]
```

Working with the solution

The solution to DE2 satisfying $x(0) = 1$ is a transformation rule that can be used to plot the solution curve and obtain solution values. For example, to get the value of the solution when $t = 2$ we can substitute $t \to 2$ into the solution expression.

```
x[t]/.soln/.t->2
N[%]
```

$$\{\sqrt{\frac{2}{13 + e^4}}\}$$

$$\{0.172008\}$$

Getting at the solution formula

Since soln is a list containing a list, soln[[1]] represents the inner list and soln[[1]][[1]] will be the transformation rule itself. This is the same as soln[[1,1]]. See below.

```
soln[[1,1]]
```

$$x \to \text{Function}\left[\{t\}, \frac{\sqrt{2}}{\sqrt{1 + e^{2t} + 2t + 2t^2}}\right]$$

Moreover, the first term in soln[[1,1]] is x and the second term is the solution in "pure function" form.
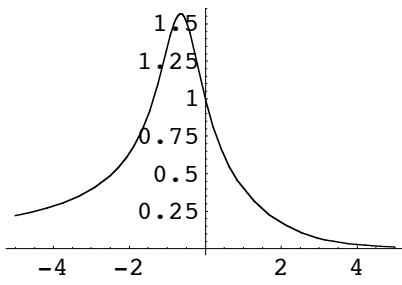
```
soln[[1,1,2]]
```

$$\text{Function}\left[\{t\}, \frac{\sqrt{2}}{\sqrt{1 + e^{2t} + 2t + 2t^2}}\right]$$

Using this, a function g can be made as follows.

```
g := soln[[1,1,2]]
```

`Plot[ g[t], {t,-5,5}]`

With g, and the graph above, let's find the positive *t* value such that $g(t) = 0.4$.

`FindRoot[ g[t]==0.4, {t,2}]`

$\{t \rightarrow 1.00532\}$

And here we check the solution.

`g[t]/.%`

`0.4`