Sections 5.2 and 5.3 - Euler method and Taylor methods

Solving Initial Value Problems with DSolve

All of the methods we will see in chapter 5 are numerical methods for finding approximate solutions to differential equations. To help us judge how good those methods are we will find it useful to be able to find the exact solutions to those equations. Fortunately, *Mathematica* has a very powerful function called DSolve that can solve many differential equations exactly. Here I apply DSolve to solve the initial value problem I will be using for all of the examples below.

```
soln = DSolve[{y'[t] == (y[t])^2 + t^2 , y[0] == 0.5}, y, t]
```

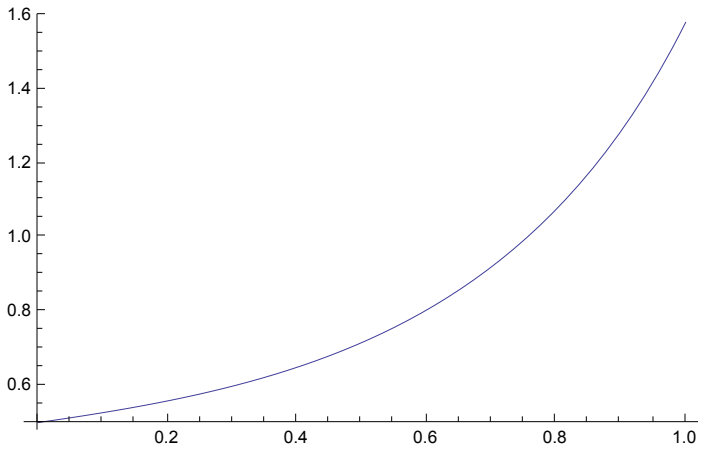
$$\left\{\left\{y \to \text{Function}\left[\{t\},\ \left(0.5\ \left(1.35196\ t^2\ \text{BesselJ}\left[-\frac{5}{4},\ \frac{t^2}{2}\right] - 2.\ t^2\ \text{BesselJ}\left[-\frac{3}{4},\ \frac{t^2}{2}\right] + 1.35196\ \text{BesselJ}\left[-\frac{1}{4},\ \frac{t^2}{2}\right] - 1.35196\ t^2\ \text{BesselJ}\left[\frac{3}{4},\ \frac{t^2}{2}\right]\right)\right) \Big/ \left(t\ \left(-1.35196\ \text{BesselJ}\left[-\frac{1}{4},\ \frac{t^2}{2}\right] + \text{BesselJ}\left[\frac{1}{4},\ \frac{t^2}{2}\right]\right)\right)\right]\right\}\right\}$$

DSolve returns a result in the form of one or more replacement rules. To do something useful with this result it is best to apply the replacement rule to define a function Y[t] that is the exact solution we seek.

```
Y := y /. soln[[1]]
```

Here is a plot of that exact solution. This will allow us to make simple visual comparisons between the exact solution and the approximate solutions we generate below.

```
plot1 = Plot[Y[t], {t, 0, 1}]
```



Direction Fields

Before we look at our first approximation method, lets take a look at a simple graphical technique that can help us to understand why solutions to differential equations behave the way they do. The differen-

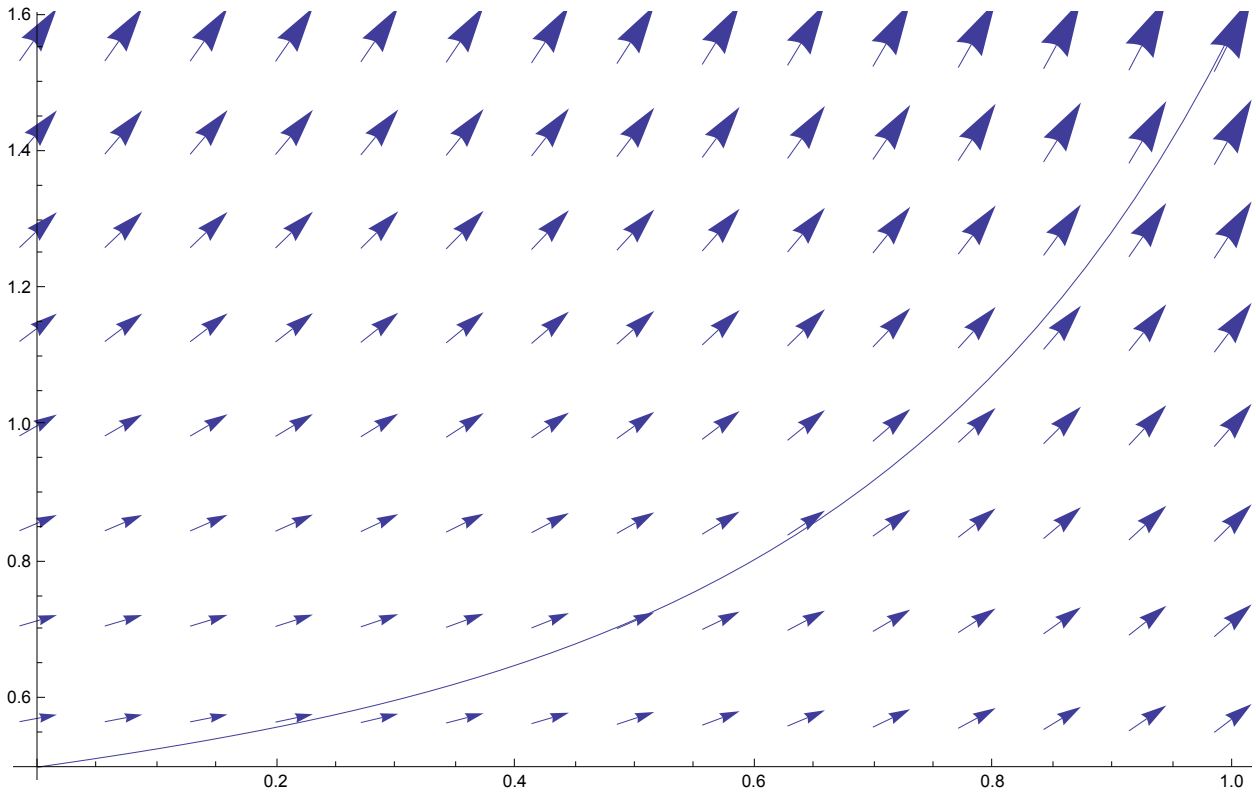
tial equation y'[t] = f[t,y] allows us to compute the slope of the solution at any point (t,y). By making a special plot called a direction field plot we can visual these slopes.
To make a plot of a direction field we use the *Mathematica* VectorPlot function.

```
f[t_, y_] := y^2 + t^2
```

Here is the direction field for this particular f.

```
plot2 = VectorPlot[{1, f[t, y]}, {t, 0, 1}, {y, 0, 2}];
```

```
Show[plot1, plot2]
```



The direction field is useful for two things. It is useful for explaining why the graph of the exact solution behaves the way it does, because the solution function has to follow the direction field. Later the direction field will also be useful for helping us to understand just why and how approximate solutions will diverge from the exact solution. Approximate solutions tend to diverge from the exact solution because once you make a mistake in estimate y[t] for a particular t the direction field will tend to push that approximate solution even further away from the exact solution.

Euler's Method

The following function implements a single step in the Euler method iteration derived in section 5.2.

```
euler[{t_, y_}] := {t + h, y + h * f[t, y]}
```

```
f[t_, y_] := y^2 + t^2
```

```
h = 0.1
```

```
0.1
```

Using the single step Euler function with NestList generates a sequence of approximate solution points.

```
points = NestList[euler, {0, 0.5}, 10];
```

To get a measure of the quality of the solution produced, we can examine the y value of the last point generated. That point corresponds to t = 1, so we can compare that approximate y with the exact y at 1.0.

```
yEst = points[[-1, 2]]
```

```
1.34466
```

```
yActual = Y[1.0]
```

```
1.58008
```

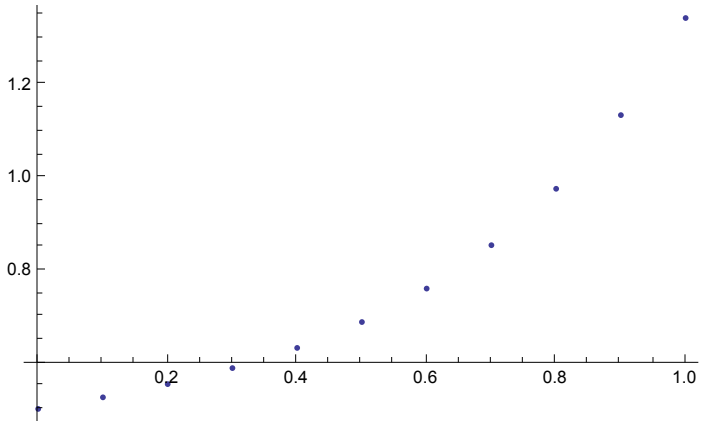
The difference between these two is quite substantial. This is not surprising, since the Euler method is a relatively crude method.

```
yActual - yEst
```

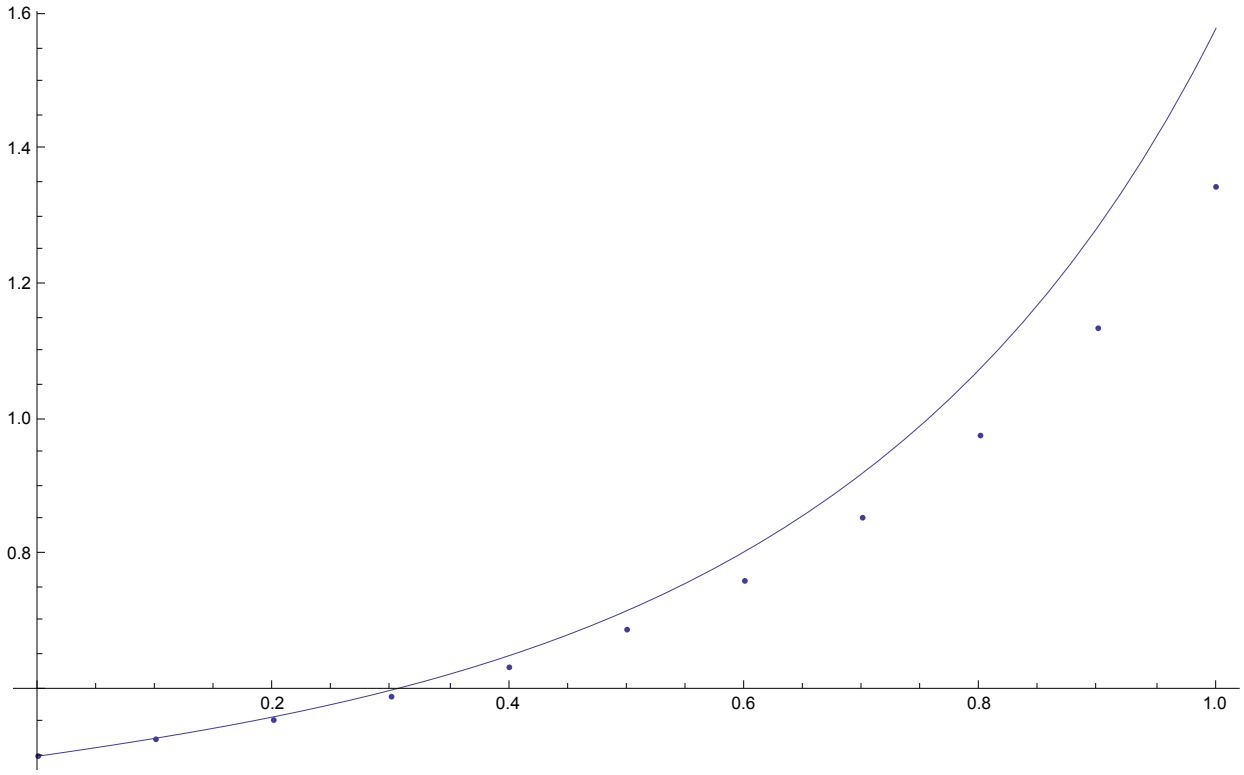
```
0.235423
```

We can see the crudeness of the Euler estimate by plotting those estimated points against the exact solution.

```
plot3 = ListPlot[points]
```



```
Show[plot1, plot3]
```



The only effective way to improve the quality of the Euler estimate is to use a smaller step size and compute more points. This second iteration through the method uses a step size 1/4 as small as above and computes 4 times as many points.

```
h = 0.025
```

```
0.025
```

```
points = NestList[euler, {0, 0.5}, 40];
```

This is the new error estimate for t = 2.0. Note that the size of the error has decreased by roughly a factor of 4, just as the step size has decreased. This is what we expect, becuase the Euler method has an O[h] error estimate.

```
yEst = points[[-1, 2]]
```

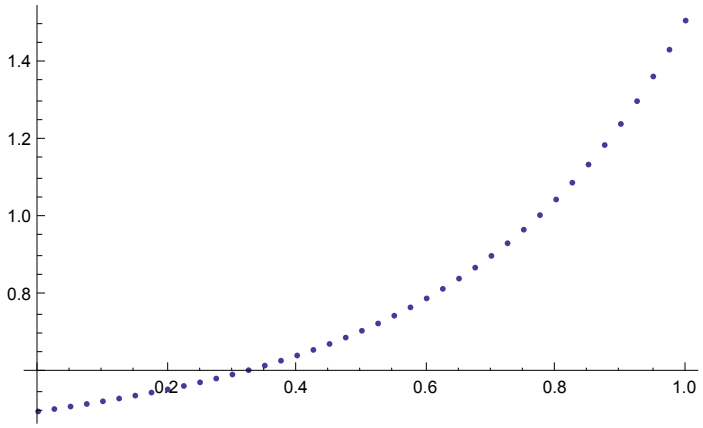
```
1.50806
```

```
yActual - yEst
```

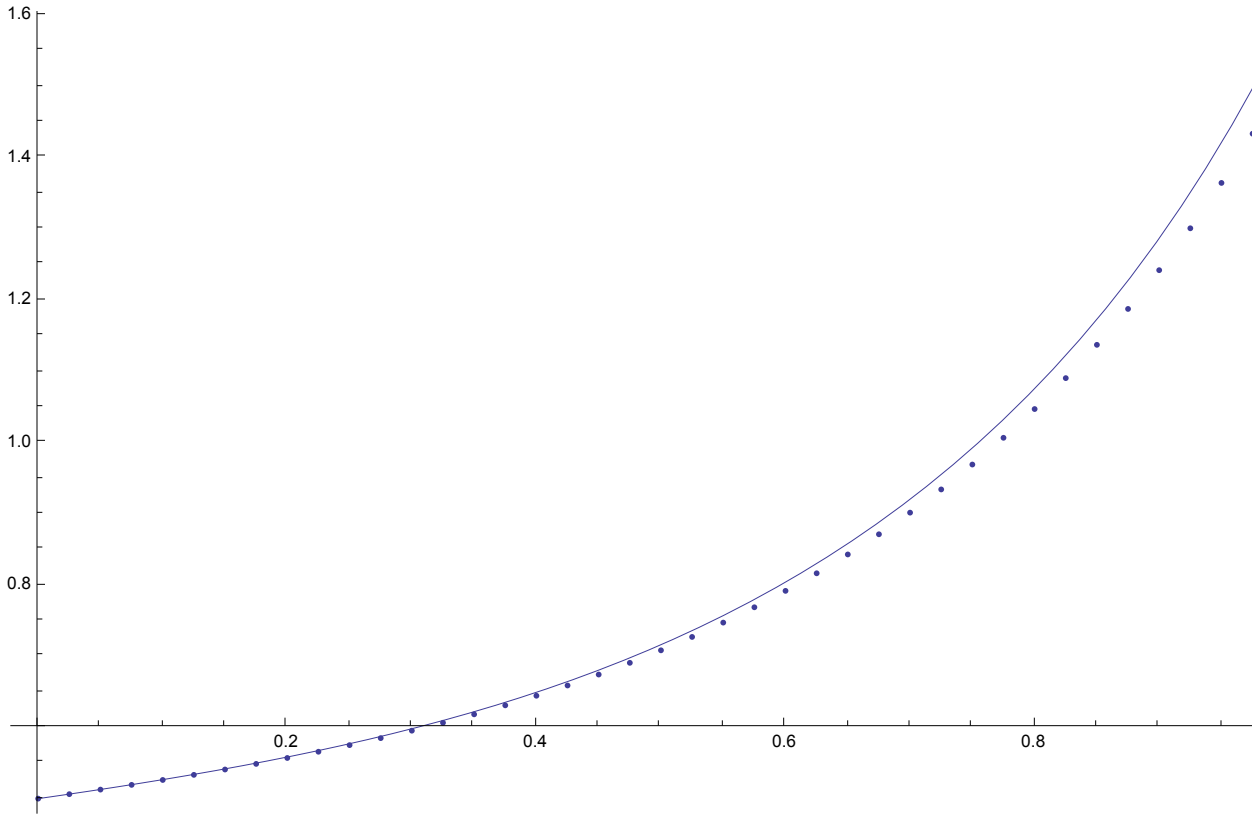
```
0.0720264
```

Here is a plot of the Euler estimate with smaller step size.

```
plot3 = ListPlot[points]
```




```
Show[plot1, plot3]
```



Higher Order Taylor Methods

The Euler method

$y_{i+1} = y_i + h\, f(t_i, y_i)$

can be written

$y(t_{i+1}) = y(t_i) + y'(t_i)\,(t_{i+1} - t_i)$

which looks like a Taylor expansion for $y(t)$ out to the first derivative. If a Taylor expansion to one derivative is useful, perhaps it might be useful to expand this estimate out to higher derivatives.

For example, using a Taylor estimate out to the third derivative produces the estimate

$$y(t_{i+1}) = y(t_i) + y'(t_i)(t_{i+1} - t_i) + \frac{y''(t_i)}{2!}(t_{i+1} - t_i)^2 + \frac{y^{(3)}(t_i)}{3!}(t_{i+1} - t_i)^3$$

As before, we can begin to simplify this by substituting $f(t_i, y_i)$ for $y'(t_i)$, but what do we do with the higher order derivatives? The simple solution is to note that the second derivative is the derivative with respect to t of the first derivative, the third derivative is the derivative of the second derivative with respect to t, and so on. Further, to simplify these derivatives as we compute them we will want to substitute $f(t, y)$ for $y'(t)$ whereever it occurs.

The following two statements show how to implement this process in *Mathematica*. The first statement computes an nth order Taylor expansion for $y(t)$ about the point $t = t_k$. The second statement goes into that expansion and replaces all of the derivative terms with $f(t, y)$.

```
Clear[f, h]
```

```
exp = Normal[Series[y[t], {t, tk, 4}]]
```

$$y[tk] + (t - tk)\, y'[tk] + \frac{1}{2}(t - tk)^2 y''[tk] + \frac{1}{6}(t - tk)^3 y^{(3)}[tk] + \frac{1}{24}(t - tk)^4 y^{(4)}[tk]$$

```
exp //. {Derivative[n_][y][tk] → (Dt[f[t, y[t]], {t, n - 1}]) /. {t → tk}}
```

$$\begin{aligned}
&(t - tk)\, f[tk, y[tk]] + y[tk] + \\
&\frac{1}{2}(t - tk)^2 \left(f[tk, y[tk]]\, f^{(0,1)}[tk, y[tk]] + f^{(1,0)}[tk, y[tk]]\right) + \\
&\frac{1}{6}(t - tk)^3 \left(f^{(0,1)}[tk, y[tk]] \left(f[tk, y[tk]]\, f^{(0,1)}[tk, y[tk]] + f^{(1,0)}[tk, y[tk]]\right) + \right. \\
&\quad f[tk, y[tk]]\, f^{(1,1)}[tk, y[tk]] + \\
&\quad \left. f[tk, y[tk]] \left(f[tk, y[tk]]\, f^{(0,2)}[tk, y[tk]] + f^{(1,1)}[tk, y[tk]]\right) + f^{(2,0)}[tk, y[tk]]\right) + \\
&\frac{1}{24}(t - tk)^4 \left(\left(f[tk, y[tk]]\, f^{(0,1)}[tk, y[tk]] + f^{(1,0)}[tk, y[tk]]\right) f^{(1,1)}[tk, y[tk]] + \right. \\
&\quad 2 \left(f[tk, y[tk]]\, f^{(0,1)}[tk, y[tk]] + f^{(1,0)}[tk, y[tk]]\right) \\
&\quad \left(f[tk, y[tk]]\, f^{(0,2)}[tk, y[tk]] + f^{(1,1)}[tk, y[tk]]\right) + f^{(0,1)}[tk, y[tk]] \\
&\quad \left(f^{(0,1)}[tk, y[tk]] \left(f[tk, y[tk]]\, f^{(0,1)}[tk, y[tk]] + f^{(1,0)}[tk, y[tk]]\right) + \right. \\
&\quad\quad f[tk, y[tk]]\, f^{(1,1)}[tk, y[tk]] + f[tk, y[tk]] \left(f[tk, y[tk]]\, f^{(0,2)}[tk, y[tk]] + \right. \\
&\quad\quad \left. \left. f^{(1,1)}[tk, y[tk]]\right) + f^{(2,0)}[tk, y[tk]]\right) + f[tk, y[tk]]\, f^{(2,1)}[tk, y[tk]] + \\
&\quad f[tk, y[tk]] \left(f[tk, y[tk]]\, f^{(1,2)}[tk, y[tk]] + f^{(2,1)}[tk, y[tk]]\right) + \\
&\quad f[tk, y[tk]] \left(f^{(0,2)}[tk, y[tk]] \left(f[tk, y[tk]]\, f^{(0,1)}[tk, y[tk]] + f^{(1,0)}[tk, y[tk]]\right) + \right. \\
&\quad\quad f[tk, y[tk]]\, f^{(1,2)}[tk, y[tk]] + f[tk, y[tk]] \left(f[tk, y[tk]]\, f^{(0,3)}[tk, y[tk]] + \right. \\
&\quad\quad \left. \left. \left. f^{(1,2)}[tk, y[tk]]\right) + f^{(2,1)}[tk, y[tk]]\right) + f^{(3,0)}[tk, y[tk]]\right)
\end{aligned}$$

The second statement uses the *Mathematica* replace repeatedly operator, //. This operator says to apply a replacement rule to an expression repeatedly until the expression no longer changes. This is what is required to fully replace all the derivatives of $y(t)$ with $f(t, y)$.

Now that we know how to do this, we can consolidate these two statements into a convenient function

definition.

```
taylorFormula[t_, w_, n_] := Module[{s, exp, subs},
  exp = Normal[Series[y[s], {s, t, n}]];
  subs = exp //. {Derivative[k_][y][t] → (Dt[f[s, y[s]], {s, k - 1}]) /. {s → t}};
  subs /. {y[t] → w, s → t + h}]
```

Here is what one of these estimates looks like. This is the Taylor estimate of order 2.

```
taylorFormula[t, y, 2]
```

$$y + h\, f[t, y] + \frac{1}{2} h^2 \left(f[t, y]\, f^{(0,1)}[t, y] + f^{(1,0)}[t, y]\right)$$

When the Taylor estimate gets applied in practice, it is most often used out to order 4. The following function sets up the iteration to develop a series of point estimates with the Taylor method of order 4.

```
taylor4[{t_, y_}] = {t + h, (taylorFormula[p, q, 4]) /. {p → t, q → y}}
```

$$\left\{h + t,\ y + h\, f[t, y] + \frac{1}{2} h^2 \left(f[t, y]\, f^{(0,1)}[t, y] + f^{(1,0)}[t, y]\right) + \right.$$
$$\frac{1}{6} h^3 \left(f^{(0,1)}[t, y] \left(f[t, y]\, f^{(0,1)}[t, y] + f^{(1,0)}[t, y]\right) + f[t, y]\, f^{(1,1)}[t, y] + \right.$$
$$\left. f[t, y] \left(f[t, y]\, f^{(0,2)}[t, y] + f^{(1,1)}[t, y]\right) + f^{(2,0)}[t, y]\right) +$$
$$\frac{1}{24} h^4 \left(\left(f[t, y]\, f^{(0,1)}[t, y] + f^{(1,0)}[t, y]\right) f^{(1,1)}[t, y] + \right.$$
$$2 \left(f[t, y]\, f^{(0,1)}[t, y] + f^{(1,0)}[t, y]\right) \left(f[t, y]\, f^{(0,2)}[t, y] + f^{(1,1)}[t, y]\right) +$$
$$f^{(0,1)}[t, y] \left(f^{(0,1)}[t, y] \left(f[t, y]\, f^{(0,1)}[t, y] + f^{(1,0)}[t, y]\right) + f[t, y]\, f^{(1,1)}[t, y] + \right.$$
$$\left. f[t, y] \left(f[t, y]\, f^{(0,2)}[t, y] + f^{(1,1)}[t, y]\right) + f^{(2,0)}[t, y]\right) +$$
$$f[t, y]\, f^{(2,1)}[t, y] + f[t, y] \left(f[t, y]\, f^{(1,2)}[t, y] + f^{(2,1)}[t, y]\right) +$$
$$f[t, y] \left(f^{(0,2)}[t, y] \left(f[t, y]\, f^{(0,1)}[t, y] + f^{(1,0)}[t, y]\right) + f[t, y]\, f^{(1,2)}[t, y] + \right.$$
$$\left.\left. f[t, y] \left(f[t, y]\, f^{(0,3)}[t, y] + f^{(1,2)}[t, y]\right) + f^{(2,1)}[t, y]\right) + f^{(3,0)}[t, y]\right)\Big\}$$

Once we specify what $f(t, y)$ is and supply a value to use for h, this becomes quite a bit more tractable.

```
f[t_, y_] := y^2 + t^2
```

```
h = 0.1
```

```
0.1
```

```
taylor4[{t, y}]
```

$$\left\{0.1 + t,\ y + 0.1 \left(t^2 + y^2\right) + 0.005 \left(2\, t + 2\, y \left(t^2 + y^2\right)\right) + \right.$$
$$0.000166667 \left(2 + 2 \left(t^2 + y^2\right)^2 + 2\, y \left(2\, t + 2\, y \left(t^2 + y^2\right)\right)\right) + 4.16667 \times 10^{-6}$$
$$\left.\left(6 \left(t^2 + y^2\right) \left(2\, t + 2\, y \left(t^2 + y^2\right)\right) + 2\, y \left(2 + 2 \left(t^2 + y^2\right)^2 + 2\, y \left(2\, t + 2\, y \left(t^2 + y^2\right)\right)\right)\right)\right\}$$

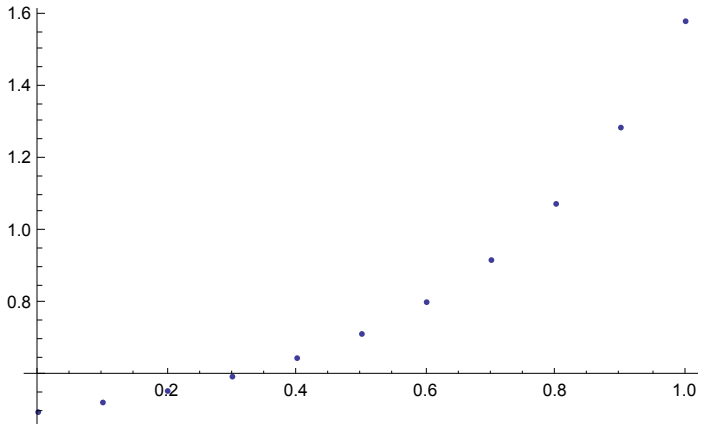
Here now is the result of iterating the fourth order Taylor formula starting from our usual initial conditions.

```
taylorPts = NestList[taylor4, {0., 0.5}, 10]
```

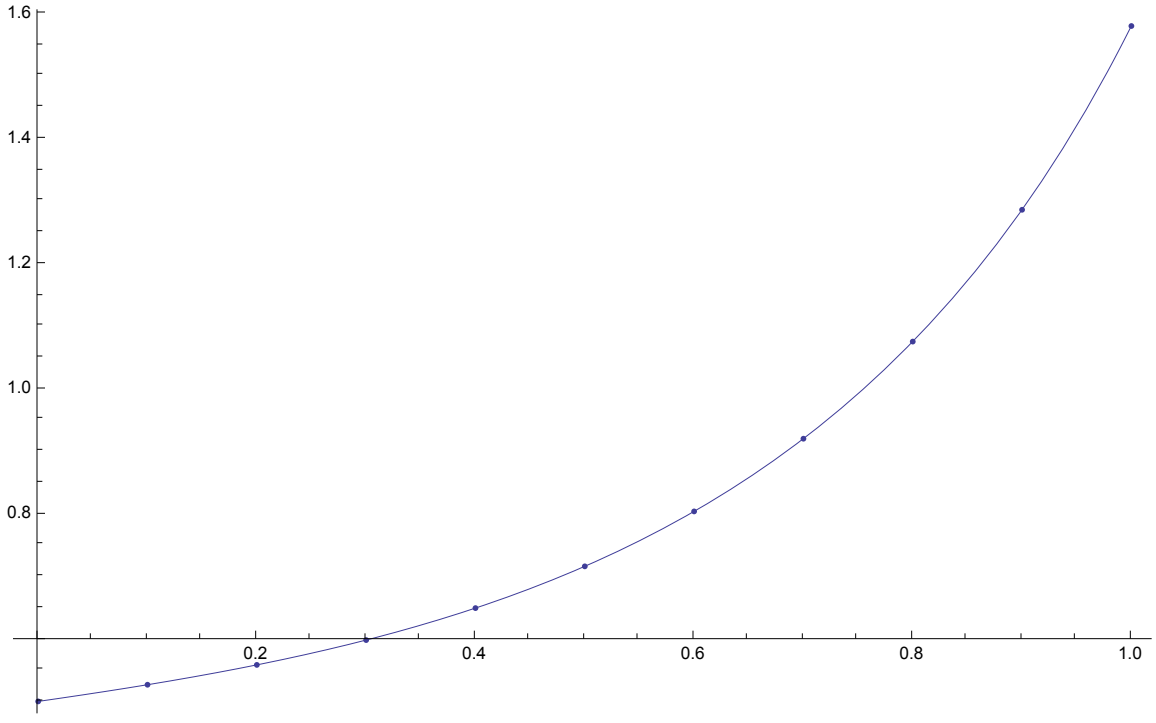
```
{{0., 0.5}, {0.1, 0.526657}, {0.2, 0.558372},
 {0.3, 0.598058}, {0.4, 0.649161}, {0.5, 0.715906}, {0.6, 0.80372},
 {0.7, 0.919956}, {0.8, 1.07522}, {0.9, 1.28593}, {1., 1.5797}}
```

Plotting these estimated points against the exact solution shows that this is a pretty good estimate.

```
plot4 = ListPlot[taylorPts]
```




```
Show[plot1, plot4]
```



Section 5.4 - Runge-Kutta rules

Runge-Kutta type rules are very important solution methods for differential equations which are simpler and easier to apply than Taylor methods. In this section I will derive some of these Runge-Kutta

type rules.

My approach here is somewhat different than the approach used in the text. This is intentional, because the text's discussion of this subject is rather vague and difficult to follow.

Multi-stage methods

We have seen that the Euler method, which uses a single derivative estimate, is fairly crude and ineffective. The Taylor method is more effective because it uses higher order derivatives to refine the estimate. Another way to refine the estimate is to use multiple estimates for the first derivative developed at different points between t_k and $t_k + h$ instead of just the single first derivative that Euler's method uses.

A *multi-stage method* computes a series of different estimates for the derivative $y'(t)$ developed at different points to improve on the crude, single derivative term that Euler's method uses.

In a multi-stage method with p stages we compute a series of derivative estimates k_i. These estimates are constructed recursively via the following scheme:

$$k_1 = f(t_k, y_k)$$
$$k_i = f\left(t_k + \alpha_i h, y_k + h \sum_{j=1}^{i-1} \beta_{i,j} k_j\right)$$

We then construct our $\phi(t_k, y_k)$ as a weighted average of the derivative estimates:

$$\phi(t_k, y_k) = \sum_{i=1}^{p} \gamma_i k_i$$

As you can see, the main problem we will confront when constructing a multi-stage method is in picking the appropriate set of constants α_i, $\beta_{i,j}$, and γ_i. In the next section we will use a sophisticated technique designed to determine the correct constants to use in the multi-stage formula.

Matching a multi-stage method to a Taylor method

The only effective methods we have seen so far are Taylor methods. Taylor methods have one weakness, and that is the difficulty in constructing the Taylor formulas. This is no big deal if we have *Mathematica* to do the heavy symbolic work for us, but nonetheless Taylor methods are not the most convenient methods around.

Here once again is the function we developed above to help us construct $T^{(n)}(t_k, y_k)$

```
taylorFormula[t_, w_, n_] := Module[{s, exp, subs},
  exp = Normal[Series[y[s], {s, t, n}]];
  subs = exp //. {Derivative[k_][y][t] → (Dt[f[s, y[s]], {s, k - 1}]) /. {s → t}};
  subs /. {y[t] → w, s → t + h}]

bigT[n_] := Simplify[(taylorFormula[tk, yk, n] - yk) / h]
```

And here again for your edification is an example of a typical Taylor formula. In this case, the Taylor formula of order 4.

```
bigT[4]
```

$$\frac{1}{24}\left(h^3\, f[tk, yk]^3\, f^{(0,3)}[tk, yk] + h^2\, f[tk, yk]^2 \left(4\left(1 + h\, f^{(0,1)}[tk, yk]\right) f^{(0,2)}[tk, yk] + 3\, h\, f^{(1,2)}[tk, yk]\right) + f[tk, yk]\left(24 + 4\, h^2\, f^{(0,1)}[tk, yk]^2 + h^3\, f^{(0,1)}[tk, yk]^3 + 3\, h^3\, f^{(0,2)}[tk, yk]\, f^{(1,0)}[tk, yk] + 8\, h^2\, f^{(1,1)}[tk, yk] + h\, f^{(0,1)}[tk, yk]\left(12 + 5\, h^2\, f^{(1,1)}[tk, yk]\right) + 3\, h^3\, f^{(2,1)}[tk, yk]\right) + h\left(f^{(1,0)}[tk, yk]\left(12 + 4\, h\, f^{(0,1)}[tk, yk] + h^2\, f^{(0,1)}[tk, yk]^2 + 3\, h^2\, f^{(1,1)}[tk, yk]\right) + h\left(\left(4 + h\, f^{(0,1)}[tk, yk]\right) f^{(2,0)}[tk, yk] + h\, f^{(3,0)}[tk, yk]\right)\right)\right)$$

Our goal is to replace this nasty thing with an equivalent multi-stage method. As you will see below, multi-stage methods are much easier to compute in practice than Taylor methods. Provided we can get the constants in the multi-stage method right, we will have found a better approach.

Since we have a bunch of constants to give values to, we should set ourselves a goal that helps us to know when we have the 'right' set of constants for the multi-stage method. Our goal here will simply be to make sure that the $\phi(t_k, y_k)$ that the multi-stage method computes is as close as possible to the $\phi(t_k, y_k)$ that comes from the Taylor method we are trying to replace. In particular, for a Taylor method of order n, which has truncation error $O(h^n)$, we will set ourselves the goal of making the $\phi(t_k, y_k)$ of the multi-stage method differ from the Taylor $\phi(t_k, y_k)$ by an error of size $O(h^n)$.

Our first challenge in this program is that the two methods do not look much like each other. The Taylor method, as you can see above, involves a tangled mass of partial derivatives of $f(t, y)$ and factors of h. The multi-stage method involves nested applications of $f(t, y)$, but no partial derivatives. How do we reconcile these?

The key to reconciling these is a multi-variate Taylor expansion of the function $f(t, y)$. Just as we can form Taylor expansions of functions of one variable, we can make Taylor expansions of multivariate functions. In *Mathematica*, the same Series function we use to develop Taylor series in a single variable can also be used to make multivariate Taylor expansions for functions of two or more variables.

Here is an example. The next statement expands $f(t, y)$ as a power series in t and y up to order 2 about (t_k, y_k).

```
Normal[Series[f[t, y], {t, tk, 2}, {y, yk, 2}]]
```

$$f[tk, yk] + (t - tk)\, f^{(1,0)}[tk, yk] + \frac{1}{2}(t - tk)^2\, f^{(2,0)}[tk, yk] + (y - yk)\left(f^{(0,1)}[tk, yk] + (t - tk)\, f^{(1,1)}[tk, yk] + \frac{1}{2}(t - tk)^2\, f^{(2,1)}[tk, yk]\right) + (y - yk)^2\left(\frac{1}{2} f^{(0,2)}[tk, yk] + \frac{1}{2}(t - tk)\, f^{(1,2)}[tk, yk] + \frac{1}{4}(t - tk)^2\, f^{(2,2)}[tk, yk]\right)$$

Lots of lovely partial derivatives here. Furthermore, if we simply replace the $t - t_k$ here with h terms, we start to get something that looks a little more like the Taylor expression above. If we can replace all of the $f(t, y)$ terms in $\phi(t_k, y_k) = \sum_{i=1}^{p} \gamma_i k_i$ with their power series expansions, we stand a pretty good chance of being able to construct an expression that we can match to the Taylor expression.

A first example

As our first example of this matching process, we will pick a fairly modest problem. Our goal is to construct a two-stage method that matches the Taylor formula of order 2.

```
bigT[2]
```

$$\frac{1}{2}\left(f[tk, yk]\left(2 + h\, f^{(0,1)}[tk, yk]\right) + h\, f^{(1,0)}[tk, yk]\right)$$

Here is how we would render the 2-stage method in *Mathematica*.

```
k[1] := f[tk, yk]
```

```
k[2] := f[tk + a[2] h, yk + h b[2, 1] × k[1]]
```

```
phi  = c[1] × k[1] + c[2] × k[2]
```

```
c[1] × f[tk, yk] + c[2] × f[tk + h a[2], yk + h b[2, 1] × f[tk, yk]]
```

We will accomplish this matching process by replacing every *f* in sight in phi with its power series expansion in *t* and *y* up to order 1. This will give us a set of partial derivative and *h* terms that we can then try to match to the corresponding terms in the Taylor formula.

First we construct a generic series expansion of $f(t, y)$.

```
fSeries = Normal[Series[f[t, y], {t, tk, 1}, {y, yk, 1}]]
```

$$f[tk, yk] + (t - tk)\, f^{(1,0)}[tk, yk] + (y - yk)\left(f^{(0,1)}[tk, yk] + (t - tk)\, f^{(1,1)}[tk, yk]\right)$$

We then construct equivalents for k[1] and k[2] where we use the series expansion to estimate $f(t, y)$ for t near t_k and y near y_k.

```
k1 = f[tk, yk]
```

```
f[tk, yk]
```

```
k2 = fSeries /. {t → tk + a[2] h, y → yk + b[2, 1] h k1}
```

```
f[tk, yk] + h a[2] f^(1,0)[tk, yk] +
 h b[2, 1] × f[tk, yk] (f^(0,1)[tk, yk] + h a[2] f^(1,1)[tk, yk])
```

```
phi = c[1] k1 + c[2] k2
```

```
c[1] × f[tk, yk] + c[2] (f[tk, yk] + h a[2] f^(1,0)[tk, yk] +
     h b[2, 1] × f[tk, yk] (f^(0,1)[tk, yk] + h a[2] f^(1,1)[tk, yk]))
```

Our goal now is to make the difference between bigT[2] and phi as small as possible. In particular, we want to make this difference be $O(h^2)$.

```
diff = Expand[bigT[2] - phi ]
```

$$f[tk, yk] - c[1] \times f[tk, yk] - c[2] \times f[tk, yk] + \frac{1}{2} h\, f[tk, yk]\, f^{(0,1)}[tk, yk] - h\, b[2, 1] \times c[2] \times f[tk, yk]\, f^{(0,1)}[tk, yk] + \frac{1}{2} h\, f^{(1,0)}[tk, yk] - h\, a[2] \times c[2]\, f^{(1,0)}[tk, yk] - h^2\, a[2] \times b[2, 1] \times c[2] \times f[tk, yk]\, f^{(1,1)}[tk, yk]$$

Right away we see what we have to do to make the terms containing only f[tk,yk] vanish. We have to pick c[1] and c[2] to satisfy c[1] + c[2] == 1.

The remaining terms involve various partial derivatives of *f*. Let us isolate those terms by using the *Mathematica* Coefficient function to isolate the coefficients of specific terms in diff.

```
set1 = Coefficient[diff, Derivative[0, 1][f][tk, yk]]
```

$$\frac{1}{2} h\, f[tk, yk] - h\, b[2, 1] \times c[2] \times f[tk, yk]$$

```
set2 = Coefficient[diff, Derivative[1, 0][f][tk, yk]]
```

$$\frac{h}{2} - h\, a[2] \times c[2]$$

We can read off from this two equations that will have to be satisfied.

```
eqns1 = {Coefficient[set1, h f[tk, yk]] == 0}
```

$$\left\{\frac{1}{2} - b[2, 1] \times c[2] == 0\right\}$$

```
eqns2 = {Coefficient[set2, h ] == 0}
```

$$\left\{\frac{1}{2} - a[2] \times c[2] == 0\right\}$$

Finally, we assemble all of the required equations into one set and solve.

```
allEqns = Join[{c[1] + c[2] == 1}, eqns1, eqns2]
```

$$\left\{c[1] + c[2] == 1,\ \frac{1}{2} - b[2, 1] \times c[2] == 0,\ \frac{1}{2} - a[2] \times c[2] == 0\right\}$$

```
soln = Solve[allEqns, {c[1], c[2], a[2], b[2, 1]}]
```

```
Solve::svars : Equations may not give solutions for all "solve" variables. ≫
```

$$\left\{\left\{c[1] \to 1 - c[2],\ a[2] \to \frac{1}{2\, c[2]},\ b[2, 1] \to \frac{1}{2\, c[2]}\right\}\right\}$$

Since the equations are under-determined (we have three equations in four unknowns), the solution comes back containing a free parameter. Basically, we are free to choose c[2] to be any number between 0 and 1, inclusive. Once we have selected c[2] the remaining constants in the multi-stage methods are set.

Two special cases of the two stage method are commonly used. The *midpoint rule* uses c[2] = 1, and the

modified Euler rule uses c[2] = 1/2.

Here is the phi for the midpoint rule.

```
c[2] = 1
```

```
1
```

```
phi = (c[1] × k[1] + c[2] × k[2]) /. soln[[1]]
```

$$f\left[\frac{h}{2} + tk,\ yk + \frac{1}{2}\, h\, f[tk,\ yk]\right]$$

and the phi for the modified Euler rule.

```
c[2] = 1 / 2
```

$$\frac{1}{2}$$

```
phi = (c[1] × k[1] + c[2] × k[2]) /. soln[[1]]
```

$$\frac{1}{2}\, f[tk,\ yk] + \frac{1}{2}\, f[h + tk,\ yk + h\, f[tk,\ yk]]$$

Both of these methods have the great advantage of being much easier to work with in practice than the Taylor method of order 2 they replace.

```
bigT[2]
```

$$\frac{1}{2}\left(f[tk,\ yk]\left(2 + h\, f^{(0,1)}[tk,\ yk]\right) + h\, f^{(1,0)}[tk,\ yk]\right)$$

Order 4 Multi-step methods

The order 2 multi-stage methods we developed above are an improvement on Euler's method, but they are not strong enough for industrial use. To get acceptable results, we have to go to at least order 4 in the Taylor method. The details are largely similar to what we saw in the section above, but the algebra does scale up quite a bit.

We start by defining k terms for a four-stage process.

```
k[1] := f[tk, yk]
```

```
k[2] := f[tk + a[2] h, yk + h b[2, 1] × k[1]]
```

```
k[3] := f[tk + a[3] h, yk + h b[3, 1] × k[1] + h b[3, 2] × k[2]]
```

```
k[4] := f[tk + a[4] h, yk + h b[4, 1] × k[1] + h b[4, 2] × k[2] + h b[4, 3] × k[3]]
```

```
phi = c[1] × k[1] + c[2] × k[2] + c[3] × k[3] + c[4] × k[4]
```

```
c[1] × f[tk, yk] + c[2] × f[tk + h a[2], yk + h b[2, 1] × f[tk, yk]] + c[3] × f[tk + h a[3],
    yk + h b[3, 1] × f[tk, yk] + h b[3, 2] × f[tk + h a[2], yk + h b[2, 1] × f[tk, yk]]] +
 c[4] × f[tk + h a[4], yk + h b[4, 1] × f[tk, yk] +
     h b[4, 2] × f[tk + h a[2], yk + h b[2, 1] × f[tk, yk]] + h b[4, 3] × f[tk + h a[3],
       yk + h b[3, 1] × f[tk, yk] + h b[3, 2] × f[tk + h a[2], yk + h b[2, 1] × f[tk, yk]]]]
```

Our goal is to match the Taylor method of order 4.

```
bigT[4]
```

$$\frac{1}{24}\left(h^3\, f[tk, yk]^3\, f^{(0,3)}[tk, yk] + h^2\, f[tk, yk]^2 \left(4\left(1 + h\, f^{(0,1)}[tk, yk]\right) f^{(0,2)}[tk, yk] + 3\, h\, f^{(1,2)}[tk, yk]\right) + f[tk, yk]\left(24 + 4\, h^2\, f^{(0,1)}[tk, yk]^2 + h^3\, f^{(0,1)}[tk, yk]^3 + 3\, h^3\, f^{(0,2)}[tk, yk]\, f^{(1,0)}[tk, yk] + 8\, h^2\, f^{(1,1)}[tk, yk] + h\, f^{(0,1)}[tk, yk]\left(12 + 5\, h^2\, f^{(1,1)}[tk, yk]\right) + 3\, h^3\, f^{(2,1)}[tk, yk]\right) + h\left(f^{(1,0)}[tk, yk]\left(12 + 4\, h\, f^{(0,1)}[tk, yk] + h^2\, f^{(0,1)}[tk, yk]^2 + 3\, h^2\, f^{(1,1)}[tk, yk]\right) + h\left(\left(4 + h\, f^{(0,1)}[tk, yk]\right) f^{(2,0)}[tk, yk] + h\, f^{(3,0)}[tk, yk]\right)\right)\right)$$

Note that this taylor function includes derivatives of up to order 2. To generate those derivatives from our multi-step formula we will have to expand *f*(*t*, *y*) to order 2 in both variables.

```
fSeries = Normal[Series[f[t, y], {t, tk, 2}, {y, yk, 2}]]
```

$$f[tk, yk] + (t - tk)\, f^{(1,0)}[tk, yk] + \frac{1}{2}(t - tk)^2\, f^{(2,0)}[tk, yk] + (y - yk)\left(f^{(0,1)}[tk, yk] + (t - tk)\, f^{(1,1)}[tk, yk] + \frac{1}{2}(t - tk)^2\, f^{(2,1)}[tk, yk]\right) + (y - yk)^2\left(\frac{1}{2} f^{(0,2)}[tk, yk] + \frac{1}{2}(t - tk)\, f^{(1,2)}[tk, yk] + \frac{1}{4}(t - tk)^2\, f^{(2,2)}[tk, yk]\right)$$

Next we construct expansions for k[1] through k[4].

```
k1 = f[tk, yk]
```

```
f[tk, yk]
```

```
k2 = fSeries /. {t → tk + a[2] h, y → yk + b[2, 1] h k1}
```

$$f[tk, yk] + h\, a[2]\, f^{(1,0)}[tk, yk] + \frac{1}{2} h^2\, a[2]^2\, f^{(2,0)}[tk, yk] + h\, b[2, 1] \times f[tk, yk]\left(f^{(0,1)}[tk, yk] + h\, a[2]\, f^{(1,1)}[tk, yk] + \frac{1}{2} h^2\, a[2]^2\, f^{(2,1)}[tk, yk]\right) + h^2\, b[2, 1]^2\, f[tk, yk]^2\left(\frac{1}{2} f^{(0,2)}[tk, yk] + \frac{1}{2} h\, a[2]\, f^{(1,2)}[tk, yk] + \frac{1}{4} h^2\, a[2]^2\, f^{(2,2)}[tk, yk]\right)$$

```
k3 = fSeries /. {t → tk + a[3] h, y → yk + b[3, 1] h k1 + b[3, 2] h k2}
```

$$f[tk, yk] + h\, a[3]\, f^{(1,0)}[tk, yk] + \frac{1}{2} h^2 a[3]^2 f^{(2,0)}[tk, yk] + \left(f^{(0,1)}[tk, yk] + h\, a[3]\, f^{(1,1)}[tk, yk] + \frac{1}{2} h^2 a[3]^2 f^{(2,1)}[tk, yk]\right) \left(h\, b[3, 1] \times f[tk, yk] + h\, b[3, 2] \left(f[tk, yk] + h\, a[2]\, f^{(1,0)}[tk, yk] + \frac{1}{2} h^2 a[2]^2 f^{(2,0)}[tk, yk] + h\, b[2, 1] \times f[tk, yk] \left(f^{(0,1)}[tk, yk] + h\, a[2]\, f^{(1,1)}[tk, yk] + \frac{1}{2} h^2 a[2]^2 f^{(2,1)}[tk, yk]\right) + h^2 b[2, 1]^2 f[tk, yk]^2 \left(\frac{1}{2} f^{(0,2)}[tk, yk] + \frac{1}{2} h\, a[2]\, f^{(1,2)}[tk, yk] + \frac{1}{4} h^2 a[2]^2 f^{(2,2)}[tk, yk]\right)\right)\right) + \left(\frac{1}{2} f^{(0,2)}[tk, yk] + \frac{1}{2} h\, a[3]\, f^{(1,2)}[tk, yk] + \frac{1}{4} h^2 a[3]^2 f^{(2,2)}[tk, yk]\right) \left(h\, b[3, 1] \times f[tk, yk] + h\, b[3, 2] \left(f[tk, yk] + h\, a[2]\, f^{(1,0)}[tk, yk] + \frac{1}{2} h^2 a[2]^2 f^{(2,0)}[tk, yk] + h\, b[2, 1] \times f[tk, yk] \left(f^{(0,1)}[tk, yk] + h\, a[2]\, f^{(1,1)}[tk, yk] + \frac{1}{2} h^2 a[2]^2 f^{(2,1)}[tk, yk]\right) + h^2 b[2, 1]^2 f[tk, yk]^2 \left(\frac{1}{2} f^{(0,2)}[tk, yk] + \frac{1}{2} h\, a[2]\, f^{(1,2)}[tk, yk] + \frac{1}{4} h^2 a[2]^2 f^{(2,2)}[tk, yk]\right)\right)\right)^2$$

```
k4 = fSeries /. {t → tk + a[4] h, y → yk + b[4, 1] h k1 + b[4, 2] h k2 + b[4, 3] h k3}
```

$$f[tk, yk] + h\, a[4]\, f^{(1,0)}[tk, yk] + \frac{1}{2} h^2 a[4]^2 f^{(2,0)}[tk, yk] + \left(f^{(0,1)}[tk, yk] + h\, a[4]\, f^{(1,1)}[tk, yk] + \frac{1}{2} h^2 a[4]^2 f^{(2,1)}[tk, yk]\right) \left(h\, b[4, 1] \times f[tk, yk] + h\, b[4, 2] \left(f[tk, yk] + h\, a[2]\, f^{(1,0)}[tk, yk] + \frac{1}{2} h^2 a[2]^2 f^{(2,0)}[tk, yk] + h\, b[2, 1] \times f[tk, yk] \left(f^{(0,1)}[tk, yk] + h\, a[2]\, f^{(1,1)}[tk, yk] + \frac{1}{2} h^2 a[2]^2 f^{(2,1)}[tk, yk]\right) + h^2 b[2, 1]^2 f[tk, yk]^2 \left(\frac{1}{2} f^{(0,2)}[tk, yk] + \frac{1}{2} h\, a[2]\, f^{(1,2)}[tk, yk] + \frac{1}{4} h^2 a[2]^2 f^{(2,2)}[tk, yk]\right)\right) + h\, b[4, 3] \left(f[tk, yk] + h\, a[3]\, f^{(1,0)}[tk, yk] + \frac{1}{2} h^2 a[3]^2 f^{(2,0)}[tk, yk] + \left(f^{(0,1)}[tk, yk] + h\, a[3]\, f^{(1,1)}[tk, yk] + \frac{1}{2} h^2 a[3]^2 f^{(2,1)}[tk, yk]\right) \left(h\, b[3, 1] \times f[tk, yk] + h\, b[3, 2] \left(f[tk, yk] + h\, a[2]\, f^{(1,0)}[tk, yk] + \right.\right.\right.$$

$$\frac{1}{2} h^2 a[2]^2 f^{(2,0)}[tk, yk] + h b[2, 1] \times f[tk, yk] \left(f^{(0,1)}[tk, yk] + h a[2] f^{(1,1)}[tk, yk] + \frac{1}{2} h^2 a[2]^2 f^{(2,1)}[tk, yk]\right) + h^2 b[2, 1]^2 f[tk, yk]^2 \left(\frac{1}{2} f^{(0,2)}[tk, yk] + \frac{1}{2} h a[2] f^{(1,2)}[tk, yk] + \frac{1}{4} h^2 a[2]^2 f^{(2,2)}[tk, yk]\right)\right)\right) + \left(\frac{1}{2} f^{(0,2)}[tk, yk] + \frac{1}{2} h a[3] f^{(1,2)}[tk, yk] + \frac{1}{4} h^2 a[3]^2 f^{(2,2)}[tk, yk]\right) \left(h b[3, 1] \times f[tk, yk] + h b[3, 2] \left(f[tk, yk] + h a[2] f^{(1,0)}[tk, yk] + \frac{1}{2} h^2 a[2]^2 f^{(2,0)}[tk, yk] + h b[2, 1] \times f[tk, yk] \left(f^{(0,1)}[tk, yk] + h a[2] f^{(1,1)}[tk, yk] + \frac{1}{2} h^2 a[2]^2 f^{(2,1)}[tk, yk]\right) + h^2 b[2, 1]^2 f[tk, yk]^2 \left(\frac{1}{2} f^{(0,2)}[tk, yk] + \frac{1}{2} h a[2] f^{(1,2)}[tk, yk] + \frac{1}{4} h^2 a[2]^2 f^{(2,2)}[tk, yk]\right)\right)^2\right)\right) + \left(\frac{1}{2} f^{(0,2)}[tk, yk] + \frac{1}{2} h a[4] f^{(1,2)}[tk, yk] + \frac{1}{4} h^2 a[4]^2 f^{(2,2)}[tk, yk]\right) \left(h b[4, 1] \times f[tk, yk] + h b[4, 2] \left(f[tk, yk] + h a[2] f^{(1,0)}[tk, yk] + \frac{1}{2} h^2 a[2]^2 f^{(2,0)}[tk, yk] + h b[2, 1] \times f[tk, yk] \left(f^{(0,1)}[tk, yk] + h a[2] f^{(1,1)}[tk, yk] + \frac{1}{2} h^2 a[2]^2 f^{(2,1)}[tk, yk]\right) + h^2 b[2, 1]^2 f[tk, yk]^2 \left(\frac{1}{2} f^{(0,2)}[tk, yk] + \frac{1}{2} h a[2] f^{(1,2)}[tk, yk] + \frac{1}{4} h^2 a[2]^2 f^{(2,2)}[tk, yk]\right)\right) + h b[4, 3] \left(f[tk, yk] + h a[3] f^{(1,0)}[tk, yk] + \frac{1}{2} h^2 a[3]^2 f^{(2,0)}[tk, yk] + \left(f^{(0,1)}[tk, yk] + h a[3] f^{(1,1)}[tk, yk] + \frac{1}{2} h^2 a[3]^2 f^{(2,1)}[tk, yk]\right) \left(h b[3, 1] \times f[tk, yk] + h b[3, 2] \left(f[tk, yk] + h a[2] f^{(1,0)}[tk, yk] + \frac{1}{2} h^2 a[2]^2 f^{(2,0)}[tk, yk] + h b[2, 1] \times f[tk, yk] \left(f^{(0,1)}[tk, yk] + h a[2] f^{(1,1)}[tk, yk] + \frac{1}{2} h^2 a[2]^2 f^{(2,1)}[tk, yk]\right) + h^2 b[2, 1]^2 f[tk, yk]^2 \left(\frac{1}{2} f^{(0,2)}[tk, yk] + \frac{1}{2} h a[2] f^{(1,2)}[tk, yk] + \frac{1}{4} h^2 a[2]^2 f^{(2,2)}[tk, yk]\right)\right)\right) + \left(\frac{1}{2} f^{(0,2)}[tk, yk] + \frac{1}{2} h a[3] f^{(1,2)}[tk, yk] + \frac{1}{4} h^2 a[3]^2 f^{(2,2)}[tk, yk]\right) \left(h b[3, 1] \times f[tk, yk] + h b[3, 2] \left(f[tk, yk] + h a[2] f^{(1,0)}[tk, yk] + \frac{1}{2} h^2 a[2]^2$$

$$f^{(2,0)}[tk, yk] + h\, b[2, 1] \times f[tk, yk] \left(f^{(0,1)}[tk, yk] + h\, a[2]\, f^{(1,1)}[tk, yk] + \frac{1}{2} h^2\, a[2]^2\, f^{(2,1)}[tk, yk]\right) + h^2\, b[2, 1]^2\, f[tk, yk]^2 \left(\frac{1}{2} f^{(0,2)}[tk, yk] + \frac{1}{2} h\, a[2]\, f^{(1,2)}[tk, yk] + \frac{1}{4} h^2\, a[2]^2\, f^{(2,2)}[tk, yk]\right)\right)^2\right)^2$$

```
phi = c[1] k1 + c[2] k2 + c[3] k3 + c[4] k4;
```

Our goal now is the minimize the difference between phi and bigT[4]. Specifically, we want all terms containing powers of *h* lower than 4 to vanish.

Here is the difference to minimize. This is a very large expression, so I have suppressed printing of it.

```
diff = Expand[bigT[4] - phi];
```

Just as in the last example, diff contains terms depending on f[tk,yk] and terms containing the two first derivatives of *f*. The first set of terms tell us that

```
cEqn = {c[1] + c[2] + c[3] + c[4] == 1}
```

```
{c[1] + c[2] + c[3] + c[4] == 1}
```

The other terms of interest are the terms containing first derivatives of *f*. We can isolate those by using Coefficient again. Once again, this is a very large set of terms, so I have suppressed printing of the terms.

```
set1 = Coefficient[diff, Derivative[0, 1][f][tk, yk]];
```

The parts of set1 that interest us are the terms involving powers of *h* less than 4. There is a relatively modest number of those terms, so we can use Coefficient to peel them off one by one for study.

```
Coefficient[set1, h]
```

$$\frac{1}{2} f[tk, yk] - b[2, 1] \times c[2] \times f[tk, yk] - b[3, 1] \times c[3] \times f[tk, yk] - b[3, 2] \times c[3] \times f[tk, yk] - b[4, 1] \times c[4] \times f[tk, yk] - b[4, 2] \times c[4] \times f[tk, yk] - b[4, 3] \times c[4] \times f[tk, yk]$$

```
Coefficient[set1, h^2]
```

$$\frac{1}{6} f^{(1,0)}[tk, yk] - a[2] \times b[3, 2] \times c[3]\, f^{(1,0)}[tk, yk] - a[2] \times b[4, 2] \times c[4]\, f^{(1,0)}[tk, yk] - a[3] \times b[4, 3] \times c[4]\, f^{(1,0)}[tk, yk]$$

```
Coefficient[set1, h^3]
```

$$\frac{1}{6} f[tk, yk]^2 f^{(0,2)}[tk, yk] - \frac{1}{2} b[2, 1]^2 b[3, 2] \times c[3] f[tk, yk]^2 f^{(0,2)}[tk, yk] -$$
$$b[2, 1] \times b[3, 1] \times b[3, 2] \times c[3] f[tk, yk]^2 f^{(0,2)}[tk, yk] -$$
$$b[2, 1] b[3, 2]^2 c[3] f[tk, yk]^2 f^{(0,2)}[tk, yk] -$$
$$\frac{1}{2} b[2, 1]^2 b[4, 2] \times c[4] f[tk, yk]^2 f^{(0,2)}[tk, yk] -$$
$$b[2, 1] \times b[4, 1] \times b[4, 2] \times c[4] f[tk, yk]^2 f^{(0,2)}[tk, yk] -$$
$$b[2, 1] b[4, 2]^2 c[4] f[tk, yk]^2 f^{(0,2)}[tk, yk] -$$
$$\frac{1}{2} b[3, 1]^2 b[4, 3] \times c[4] f[tk, yk]^2 f^{(0,2)}[tk, yk] -$$
$$b[3, 1] \times b[3, 2] \times b[4, 3] \times c[4] f[tk, yk]^2 f^{(0,2)}[tk, yk] -$$
$$\frac{1}{2} b[3, 2]^2 b[4, 3] \times c[4] f[tk, yk]^2 f^{(0,2)}[tk, yk] -$$
$$b[3, 1] \times b[4, 1] \times b[4, 3] \times c[4] f[tk, yk]^2 f^{(0,2)}[tk, yk] -$$
$$b[3, 2] \times b[4, 1] \times b[4, 3] \times c[4] f[tk, yk]^2 f^{(0,2)}[tk, yk] -$$
$$b[2, 1] \times b[4, 2] \times b[4, 3] \times c[4] f[tk, yk]^2 f^{(0,2)}[tk, yk] -$$
$$b[3, 1] \times b[4, 2] \times b[4, 3] \times c[4] f[tk, yk]^2 f^{(0,2)}[tk, yk] -$$
$$b[3, 2] \times b[4, 2] \times b[4, 3] \times c[4] f[tk, yk]^2 f^{(0,2)}[tk, yk] -$$
$$b[3, 1] b[4, 3]^2 c[4] f[tk, yk]^2 f^{(0,2)}[tk, yk] -$$
$$b[3, 2] b[4, 3]^2 c[4] f[tk, yk]^2 f^{(0,2)}[tk, yk] + \frac{5}{24} f[tk, yk] f^{(1,1)}[tk, yk] -$$
$$a[2] \times b[2, 1] \times b[3, 2] \times c[3] \times f[tk, yk] f^{(1,1)}[tk, yk] -$$
$$a[3] \times b[2, 1] \times b[3, 2] \times c[3] \times f[tk, yk] f^{(1,1)}[tk, yk] -$$
$$a[2] \times b[2, 1] \times b[4, 2] \times c[4] \times f[tk, yk] f^{(1,1)}[tk, yk] -$$
$$a[4] \times b[2, 1] \times b[4, 2] \times c[4] \times f[tk, yk] f^{(1,1)}[tk, yk] -$$
$$a[3] \times b[3, 1] \times b[4, 3] \times c[4] \times f[tk, yk] f^{(1,1)}[tk, yk] -$$
$$a[4] \times b[3, 1] \times b[4, 3] \times c[4] \times f[tk, yk] f^{(1,1)}[tk, yk] -$$
$$a[3] \times b[3, 2] \times b[4, 3] \times c[4] \times f[tk, yk] f^{(1,1)}[tk, yk] -$$
$$a[4] \times b[3, 2] \times b[4, 3] \times c[4] \times f[tk, yk] f^{(1,1)}[tk, yk] +$$
$$\frac{1}{24} f^{(2,0)}[tk, yk] - \frac{1}{2} a[2]^2 b[3, 2] \times c[3] f^{(2,0)}[tk, yk] -$$
$$\frac{1}{2} a[2]^2 b[4, 2] \times c[4] f^{(2,0)}[tk, yk] - \frac{1}{2} a[3]^2 b[4, 3] \times c[4] f^{(2,0)}[tk, yk]$$

Examining these sets of terms gives us some equations that need to be solved.

```
eqns1 = {Coefficient[set1, h f[tk, yk]] == 0, Coefficient[set1, h^2 f^(1,0)[tk, yk]] == 0,
  Coefficient[set1, h^3 f[tk, yk]^2 f^(0,2)[tk, yk]] == 0,
  Coefficient[set1, h^3 f[tk, yk] f^(1,1)[tk, yk]] == 0,
  Coefficient[set1, h^3 f^(2,0)[tk, yk]] == 0}
```

$$\left\{\frac{1}{2} - b[2,1]\times c[2] - b[3,1]\times c[3] - b[3,2]\times c[3] - b[4,1]\times c[4] - b[4,2]\times c[4] - b[4,3]\times c[4] == 0,\right.$$

$$\frac{1}{6} - a[2]\times b[3,2]\times c[3] - a[2]\times b[4,2]\times c[4] - a[3]\times b[4,3]\times c[4] == 0,$$

$$\frac{1}{6} - \frac{1}{2} b[2,1]^2 b[3,2]\times c[3] - b[2,1]\times b[3,1]\times b[3,2]\times c[3] - b[2,1]\, b[3,2]^2 c[3] -$$
$$\frac{1}{2} b[2,1]^2 b[4,2]\times c[4] - b[2,1]\times b[4,1]\times b[4,2]\times c[4] - b[2,1]\, b[4,2]^2 c[4] -$$
$$\frac{1}{2} b[3,1]^2 b[4,3]\times c[4] - b[3,1]\times b[3,2]\times b[4,3]\times c[4] - \frac{1}{2} b[3,2]^2 b[4,3]\times c[4] -$$
$$b[3,1]\times b[4,1]\times b[4,3]\times c[4] - b[3,2]\times b[4,1]\times b[4,3]\times c[4] -$$
$$b[2,1]\times b[4,2]\times b[4,3]\times c[4] - b[3,1]\times b[4,2]\times b[4,3]\times c[4] -$$
$$b[3,2]\times b[4,2]\times b[4,3]\times c[4] - b[3,1]\, b[4,3]^2 c[4] - b[3,2]\, b[4,3]^2 c[4] == 0,$$

$$\frac{5}{24} - a[2]\times b[2,1]\times b[3,2]\times c[3] - a[3]\times b[2,1]\times b[3,2]\times c[3] -$$
$$a[2]\times b[2,1]\times b[4,2]\times c[4] - a[4]\times b[2,1]\times b[4,2]\times c[4] -$$
$$a[3]\times b[3,1]\times b[4,3]\times c[4] - a[4]\times b[3,1]\times b[4,3]\times c[4] -$$
$$a[3]\times b[3,2]\times b[4,3]\times c[4] - a[4]\times b[3,2]\times b[4,3]\times c[4] == 0,$$

$$\left.\frac{1}{24} - \frac{1}{2} a[2]^2 b[3,2]\times c[3] - \frac{1}{2} a[2]^2 b[4,2]\times c[4] - \frac{1}{2} a[3]^2 b[4,3]\times c[4] == 0\right\}$$

Next, we do the similar process with the other first derivative terms.

```
set2 = Coefficient[diff, Derivative[1, 0][f][tk, yk]];
```

```
Coefficient[set2, h]
```

$$\frac{1}{2} - a[2]\times c[2] - a[3]\times c[3] - a[4]\times c[4]$$

```
Coefficient[set2, h^2]
```

$$\frac{1}{6} f^{(0,1)}[tk, yk] - a[2]\times b[3,2]\times c[3]\, f^{(0,1)}[tk, yk] -$$
$$a[2]\times b[4,2]\times c[4]\, f^{(0,1)}[tk, yk] - a[3]\times b[4,3]\times c[4]\, f^{(0,1)}[tk, yk]$$

```
Coefficient[set2, h^3]
```

$$\frac{1}{24} f^{(0,1)}[tk, yk]^2 - a[2] \times b[3, 2] \times b[4, 3] \times c[4] f^{(0,1)}[tk, yk]^2 +$$
$$\frac{1}{8} f[tk, yk] f^{(0,2)}[tk, yk] - a[2] \times b[3, 1] \times b[3, 2] \times c[3] \times f[tk, yk] f^{(0,2)}[tk, yk] -$$
$$a[2] b[3, 2]^2 c[3] \times f[tk, yk] f^{(0,2)}[tk, yk] -$$
$$a[2] \times b[4, 1] \times b[4, 2] \times c[4] \times f[tk, yk] f^{(0,2)}[tk, yk] -$$
$$a[2] b[4, 2]^2 c[4] \times f[tk, yk] f^{(0,2)}[tk, yk] -$$
$$a[3] \times b[4, 1] \times b[4, 3] \times c[4] \times f[tk, yk] f^{(0,2)}[tk, yk] -$$
$$a[2] \times b[4, 2] \times b[4, 3] \times c[4] \times f[tk, yk] f^{(0,2)}[tk, yk] -$$
$$a[3] \times b[4, 2] \times b[4, 3] \times c[4] \times f[tk, yk] f^{(0,2)}[tk, yk] -$$
$$a[3] b[4, 3]^2 c[4] \times f[tk, yk] f^{(0,2)}[tk, yk] +$$
$$\frac{1}{8} f^{(1,1)}[tk, yk] - a[2] \times a[3] \times b[3, 2] \times c[3] f^{(1,1)}[tk, yk] -$$
$$a[2] \times a[4] \times b[4, 2] \times c[4] f^{(1,1)}[tk, yk] - a[3] \times a[4] \times b[4, 3] \times c[4] f^{(1,1)}[tk, yk]$$

```
eqns2 = {Coefficient[set2, h ] == 0, Coefficient[set2, h^2 f^(0,1)[tk, yk]] == 0,
   Coefficient[set2,  h^3 f^(0,1)[tk, yk]^2] == 0,
   Coefficient[set2, h^3 f[tk, yk] f^(0,2)[tk, yk]] == 0,
   Coefficient[set2, h^3 f^(1,1)[tk, yk]] == 0};
```

Here now are all of the equations we need to solve.

```
allEqns = Join[cEqn, eqns1, eqns2]
```

$$\Big\{c[1]+c[2]+c[3]+c[4]==1,\ \frac{1}{2}-b[2,1]\times c[2]-b[3,1]\times c[3]-$$
$$b[3,2]\times c[3]-b[4,1]\times c[4]-b[4,2]\times c[4]-b[4,3]\times c[4]==0,$$
$$\frac{1}{6}-a[2]\times b[3,2]\times c[3]-a[2]\times b[4,2]\times c[4]-a[3]\times b[4,3]\times c[4]==0,$$
$$\frac{1}{6}-\frac{1}{2}b[2,1]^2 b[3,2]\times c[3]-b[2,1]\times b[3,1]\times b[3,2]\times c[3]-b[2,1]\,b[3,2]^2 c[3]-$$
$$\frac{1}{2}b[2,1]^2 b[4,2]\times c[4]-b[2,1]\times b[4,1]\times b[4,2]\times c[4]-b[2,1]\,b[4,2]^2 c[4]-$$
$$\frac{1}{2}b[3,1]^2 b[4,3]\times c[4]-b[3,1]\times b[3,2]\times b[4,3]\times c[4]-\frac{1}{2}b[3,2]^2 b[4,3]\times c[4]-$$
$$b[3,1]\times b[4,1]\times b[4,3]\times c[4]-b[3,2]\times b[4,1]\times b[4,3]\times c[4]-$$
$$b[2,1]\times b[4,2]\times b[4,3]\times c[4]-b[3,1]\times b[4,2]\times b[4,3]\times c[4]-$$
$$b[3,2]\times b[4,2]\times b[4,3]\times c[4]-b[3,1]\,b[4,3]^2 c[4]-b[3,2]\,b[4,3]^2 c[4]==0,$$
$$\frac{5}{24}-a[2]\times b[2,1]\times b[3,2]\times c[3]-a[3]\times b[2,1]\times b[3,2]\times c[3]-$$
$$a[2]\times b[2,1]\times b[4,2]\times c[4]-a[4]\times b[2,1]\times b[4,2]\times c[4]-$$
$$a[3]\times b[3,1]\times b[4,3]\times c[4]-a[4]\times b[3,1]\times b[4,3]\times c[4]-$$
$$a[3]\times b[3,2]\times b[4,3]\times c[4]-a[4]\times b[3,2]\times b[4,3]\times c[4]==0,$$
$$\frac{1}{24}-\frac{1}{2}a[2]^2 b[3,2]\times c[3]-\frac{1}{2}a[2]^2 b[4,2]\times c[4]-\frac{1}{2}a[3]^2 b[4,3]\times c[4]==0,$$
$$\frac{1}{2}-a[2]\times c[2]-a[3]\times c[3]-a[4]\times c[4]==0,$$
$$\frac{1}{6}-a[2]\times b[3,2]\times c[3]-a[2]\times b[4,2]\times c[4]-a[3]\times b[4,3]\times c[4]==0,$$
$$\frac{1}{24}-a[2]\times b[3,2]\times b[4,3]\times c[4]==0,$$
$$\frac{1}{8}-a[2]\times b[3,1]\times b[3,2]\times c[3]-a[2]\,b[3,2]^2 c[3]-a[2]\times b[4,1]\times b[4,2]\times c[4]-$$
$$a[2]\,b[4,2]^2 c[4]-a[3]\times b[4,1]\times b[4,3]\times c[4]-a[2]\times b[4,2]\times b[4,3]\times c[4]-$$
$$a[3]\times b[4,2]\times b[4,3]\times c[4]-a[3]\,b[4,3]^2 c[4]==0,\ \frac{1}{8}-a[2]\times a[3]\times b[3,2]\times c[3]-$$
$$a[2]\times a[4]\times b[4,2]\times c[4]-a[3]\times a[4]\times b[4,3]\times c[4]==0\Big\}$$

Unfortunately, this is a difficult system of nonlinear equations. Even *Mathematica* will have a lot of trouble solving these equations. Consequently, we will have to give it some help.

One reasonable way to generate a solution to this system is to make two simple assumptions. The first assumption is that each step depends only on the step immediately before it. The second assumption is that step 2 looks like the midpoint rule.

```
b[n_, k_] := 0 /; k < n - 1;
a[2] = 1 / 2;
b[2, 1] = 1 / 2;
```

If we code these assumptions and then try again to solve the equations, we meet with success.

```
allEqns
```

$$\left\{c[1]+c[2]+c[3]+c[4]==1,\ \frac{1}{2}-\frac{c[2]}{2}-b[3,2]\times c[3]-b[4,3]\times c[4]==0,\right.$$
$$\frac{1}{6}-\frac{1}{2}b[3,2]\times c[3]-a[3]\times b[4,3]\times c[4]==0,$$
$$\frac{1}{6}-\frac{1}{8}b[3,2]\times c[3]-\frac{1}{2}b[3,2]^2 c[3]-\frac{1}{2}b[3,2]^2 b[4,3]\times c[4]-b[3,2]\,b[4,3]^2 c[4]==$$
$$0,\ \frac{5}{24}-\frac{1}{4}b[3,2]\times c[3]-\frac{1}{2}a[3]\times b[3,2]\times c[3]-a[3]\times b[3,2]\times b[4,3]\times c[4]-$$
$$a[4]\times b[3,2]\times b[4,3]\times c[4]==0,\ \frac{1}{24}-\frac{1}{8}b[3,2]\times c[3]-\frac{1}{2}a[3]^2 b[4,3]\times c[4]==0,$$
$$\frac{1}{2}-\frac{c[2]}{2}-a[3]\times c[3]-a[4]\times c[4]==0,\ \frac{1}{6}-\frac{1}{2}b[3,2]\times c[3]-a[3]\times b[4,3]\times c[4]==0,$$
$$\frac{1}{24}-\frac{1}{2}b[3,2]\times b[4,3]\times c[4]==0,\ \frac{1}{8}-\frac{1}{2}b[3,2]^2 c[3]-a[3]\,b[4,3]^2 c[4]==0,$$
$$\left.\frac{1}{8}-\frac{1}{2}a[3]\times b[3,2]\times c[3]-a[3]\times a[4]\times b[4,3]\times c[4]==0\right\}$$

```
soln = Solve[allEqns, {a[3], a[4], b[3, 2], b[4, 3], c[1], c[2], c[3], c[4]}]
```

$$\left\{\left\{c[1]\to\frac{1}{6},\ c[2]\to\frac{1}{3},\ a[4]\to 1,\ c[3]\to\frac{1}{3},\ a[3]\to\frac{1}{2},\ c[4]\to\frac{1}{6},\ b[4,3]\to 1,\ b[3,2]\to\frac{1}{2}\right\}\right\}$$

This produces the most commonly used Runge-Kutta formula of order 4. Slightly different assumptions will produce other possible order 4 formulas.

```
phi = (c[1] × k[1] + c[2] × k[2] + c[3] × k[3] + c[4] × k[4]) /. soln[[1]]
```

$$\frac{1}{6}f[tk,yk]+\frac{1}{3}f\left[\frac{h}{2}+tk,yk+\frac{1}{2}h\,f[tk,yk]\right]+$$
$$\frac{1}{3}f\left[\frac{h}{2}+tk,yk+\frac{1}{2}h\,f\left[\frac{h}{2}+tk,yk+\frac{1}{2}h\,f[tk,yk]\right]\right]+$$
$$\frac{1}{6}f\left[h+tk,yk+h\,f\left[\frac{h}{2}+tk,yk+\frac{1}{2}h\,f\left[\frac{h}{2}+tk,yk+\frac{1}{2}h\,f[tk,yk]\right]\right]\right]$$

Runge-Kutta method example

Here is the code to implement a single step in the Runge-Kutta method we derived above. For clarity, this version is written out in multi-stage form.

```
runge[{t_, y_}] := Module[{k1, k2, k3, k4},
  k1 = f[t, y];
  k2 = f[t + h / 2, y + k1 * h / 2];
  k3 = f[t + h / 2, y + k2 * h / 2];
  k4 = f[t + h, y + k3 * h];
  {t + h, y + (k1 + 2 * k2 + 2 * k3 + k4) * h / 6}]
```

Let's test this on the differential equation we've been using for examples.

```
f[t_, y_] := y^2 + t^2
```

```
h = 0.1
```

```
0.1
```

```
rungePts = NestList[runge, {0., 0.5}, 10]
```

```
{{0., 0.5}, {0.1, 0.526658}, {0.2, 0.558374},
 {0.3, 0.598062}, {0.4, 0.649168}, {0.5, 0.715919}, {0.6, 0.803744},
 {0.7, 0.92}, {0.8, 1.0753}, {0.9, 1.28611}, {1., 1.58009}}
```

The Runge-Kutta method is a fourth order method with error proportional to h^4. For a wide range of applications Runge-Kutta produces acceptable results with only a modest amount of calculation effort.

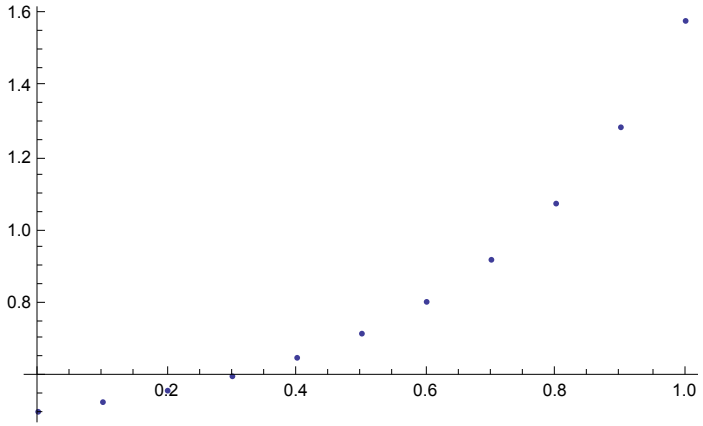
```
yEst = rungePts[[-1, 2]]
```

```
1.58009
```

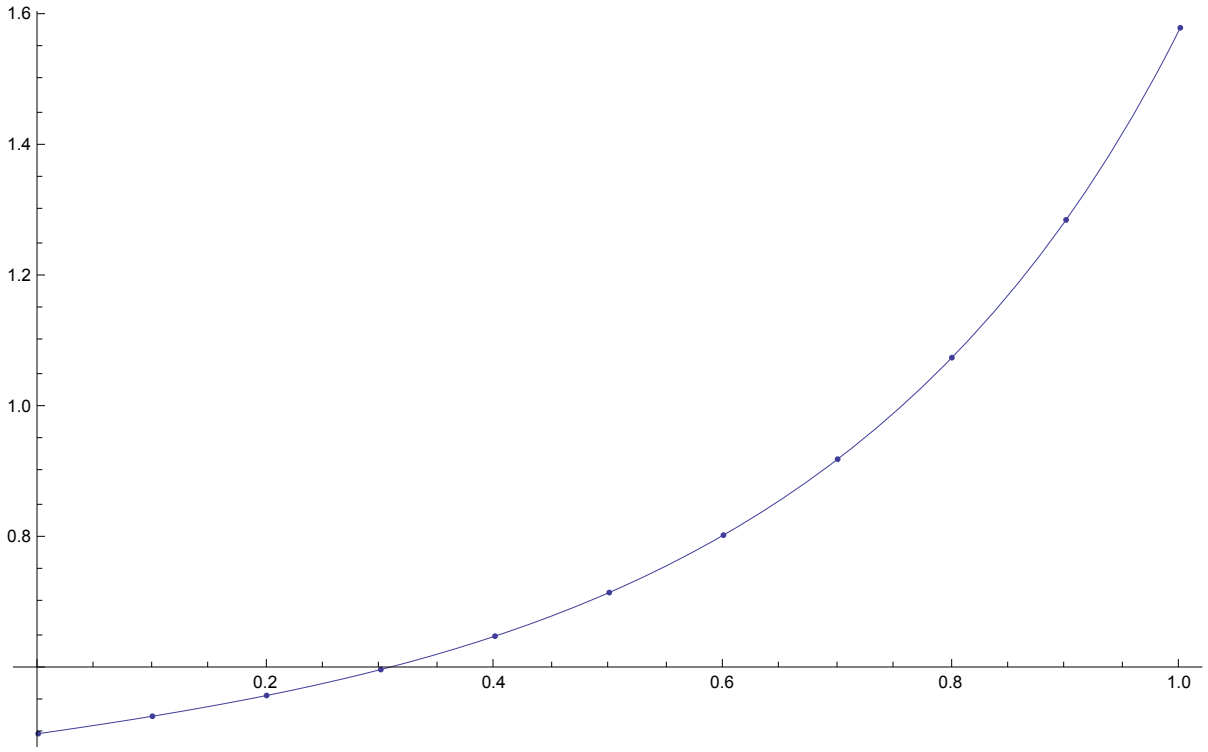
```
yActual - yEst
```

$$-3.02998 \times 10^{-6}$$

```
plot6 = ListPlot[rungePts]
```



```
Show[plot1, plot6]
```



Numerical Solutions from *Mathematica*

Now that we have seen a number of numerical methods for solving first order differential equations, it is interesting to note that *Mathematica* also has built-in methods for solving differential equations numerically.

Numerical methods have to be applied in situations where it is impossible to solve the differential equation exactly. Here is a simple example. The following demonstrates that there are relatively simple looking first order differential equations that are impossible for *Mathematica* to solve exactly.

```
DSolve[{y'[t] == (y[t])^(-2) + t^2, y[0.0] == 0.5}, y[t], t]
```

$$\text{DSolve}\left[\left\{y'[t] == t^2 + \frac{1}{y[t]^2}, y[0.] == 0.5\right\}, y[t], t\right]$$

When DSolve can not solve an equation, you can at least solve it numerically by using NDSolve, the built-in numerical differential equation solver in *Mathematica*.

```
soln = NDSolve[{y'[t] == (y[t])^(-2) + t^2, y[0.0] == 0.5}, y, {t, 0.0, 2.0}]
```

```
{{y → InterpolatingFunction[{{0., 2.}}, <>]}}
```

We can use this result to get an estimate for what y[2.0] is.

```
Evaluate[y[2.0] /. soln[[1]]]
```

```
4.23683
```

By way of comparison, here is the same equation solved by the Runge-Kutta method we developed above.

```
h = 0.01
```

```
0.01
```

```
f[t_, y_] := y^(-2) + t^2
```

```
rungePts = NestList[runge, {0., 0.5}, 200];
```

```
rungePts[[-1]]
```

```
{2., 4.23683}
```

Out to 6 digits the produces the same result as NDSolve. (This is actually not that surprising, since NDSolve is actually using a variant of the Runge-Kutta method to do its work.)

The Runge-Kutta-Fehlberg Method

This section of our notes will take you through the process of using the Runge-Kutta-Fehlberg variable step size method to solve a first order DE.

The exact solution

To make it easier to judge whether or not our methods are working correctly, we choose a differential equation that can be solved exactly. The problem we solve comes from exercise 3a at the end of section 5.5.

Here is the equation solved exactly with DSolve.

```
a = 1.0;
b = 3.0;
```

```
soln = DSolve[{y'[t] == y[t] / t - (y[t] / t)^2, y[a] == 1.}, y, t]
```

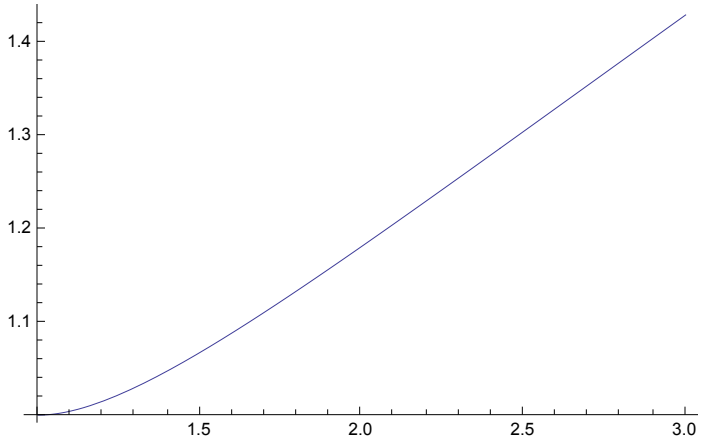
$$\left\{\left\{y \to \text{Function}\left[\{t\}, \frac{t}{1. + \text{Log}[t]}\right]\right\}\right\}$$

```
Y[t_] = Evaluate[ y[t] /. soln[[1]]]
```

$$\frac{t}{1. + \text{Log}[t]}$$

Here is a plot of the exact solution.

```
plot1 = Plot[Y[t], {t, a, b}]
```



Runge-Kutta method with fixed step

For comparison purposes, we first solve the initial value problem with the Runge-Kutta method with step size $h = 0.2$.

```
runge[{t_, y_}] := Module[{k1, k2, k3, k4},
  k1 = f[t, y];
  k2 = f[t + h / 2, y + k1 * h / 2];
  k3 = f[t + h / 2, y + k2 * h / 2];
  k4 = f[t + h, y + k3 * h];
  {t + h, y + (k1 + 2 * k2 + 2 * k3 + k4) * h / 6}]
```

```
f[t_, y_] := y / t - (y / t) ^ 2
```

```
h = 0.2
```

```
0.2
```

```
rungePts = NestList[runge, {a, 1.}, Round[(b - a) / h]]
```

```
{{1., 1.}, {1.2, 1.01495}, {1.4, 1.04753},
 {1.6, 1.08843}, {1.8, 1.13365}, {2., 1.18123}, {2.2, 1.2301},
 {2.4, 1.27967}, {2.6, 1.32957}, {2.8, 1.37956}, {3., 1.42951}}
```

Since we have the exact solution we can compute the actual error at $t = 3$:

```
rungeError = Y[b] - rungePts[[-1, 2]]
```

6.25765×10^{-6}

Runge-Kutta-Fehlberg

Here now is the code to do one step in the Runge-Kutta-Fehlberg method. This function takes as its input a list composed of a value for *t*, a suggested step size *h*, and a value for *y*. It returns a new *t* and *y* along with a new suggested step size to use for the next iteration. The function does its work by first computing an error estimate, *r*. Once the error *r* is known, we can use it to compute the new step size,

q h. If *r* is smaller than a preset tolerance, TOL, we go ahead and return the result of the estimate. If the error exceeds TOL, we redo the estimate with *q h* in place of *h* and return the result of that instead.

```
rkf[{t_, h_, y_}] := Module[{k1, k2, k3, k4, k5, k6, est1, est2, r, q, newH},
   k1 = h * f[t, y];
   k2 = h * f[t + h / 4, y + k1 / 4];
   k3 = h * f[t + 3 * h / 8, y + 3 * k1 / 32 + 9 * k2 / 32];
   k4 = h * f[t + 12 * h / 13, y + 1932 * k1 / 2197 - 7200 * k2 / 2197 + 7296 * k3 / 2197];
   k5 = h * f[t + h, y + 439 * k1 / 216 - 8 * k2 + 3680 * k3 / 513 - 845 * k4 / 4104];
   k6 =
    h * f[t + h / 2, y - 8 * k1 / 27 + 2 * k2 - 3544 * k3 / 2565 + 1859 * k4 / 4104 - 11 * k5 / 40];
   r = Abs[k1 / 360 - 128 * k3 / 4275 - 2197 * k4 / 75 240 + k5 / 50 + 2 * k6 / 55] / h;
   q = 0.84 * (TOL / r) ^ (1 / 4);
   newH = If[q ≤ 0.1, 0.1 * h, If[q ≥ 4, 4 * h, q * h]];
   If[r ≥ TOL, rkf[{t, newH, y}],
    {t + h, newH, y + 25 * k1 / 216 + 1408 * k3 / 2565 + 2197 * k4 / 4104 - k5 / 5}]];
```

We are now ready to apply this method. We have to provide a tolerance.

```
TOL = 10 ^ (-8)
```

$$\frac{1}{100\,000\,000}$$

A further complication is that we can not use NestList to iterate this method. The problem is that the step size will be changing constantly, so we can't predict in advance how many steps it will take to go from *a* to *b*. The solution is to use a variant of NestList, NestWhileList. NestWhileList takes as its third parameter a function. On each iteration it will apply that function to the result of the most recent iteration. If it returns true, NestWhileList will continue for another iteration. If it returns false, the iteration will stop. Since we want our iteration to continue until the *t* coordinate of the list returned by rkf passes $t = b$, we set up our test function like so:

```
continueTest[{t_, h_, y_}] = (t < b)
```

```
t < 3.
```

We are now ready to run the iteration.

```
rkfList = NestWhileList[rkf, {a, 0.2, 1.}, continueTest]
```

```
{{1., 0.2, 1.}, {1.03608, 0.0332876, 1.00061}, {1.06937, 0.0360897, 1.00216},
 {1.10546, 0.0388362, 1.00472}, {1.14429, 0.0418997, 1.00838},
 {1.18619, 0.0452993, 1.01319}, {1.23149, 0.0490819, 1.01926},
 {1.28057, 0.0533005, 1.02667}, {1.33387, 0.0580166, 1.03555},
 {1.39189, 0.0633017, 1.04601}, {1.45519, 0.0692396, 1.05822},
 {1.52443, 0.0759283, 1.07232}, {1.60036, 0.0834835, 1.08851},
 {1.68384, 0.0920416, 1.10701}, {1.77588, 0.101766, 1.12805},
 {1.87765, 0.112851, 1.15192}, {1.9905, 0.125534, 1.17894},
 {2.11604, 0.140104, 1.20948}, {2.25614, 0.156923, 1.24397},
 {2.41306, 0.176442, 1.28293}, {2.5895, 0.199247, 1.32695},
 {2.78875, 0.226106, 1.37676}, {3.01486, 0.258058, 1.43322}}
```

Here is the actual error at the last point computed. As you can see, the rkf method has met the desired error tolerance.

```
rkfError = Y[rkfList[[-1, 1]] - rkfList[[-1, 3]]
```

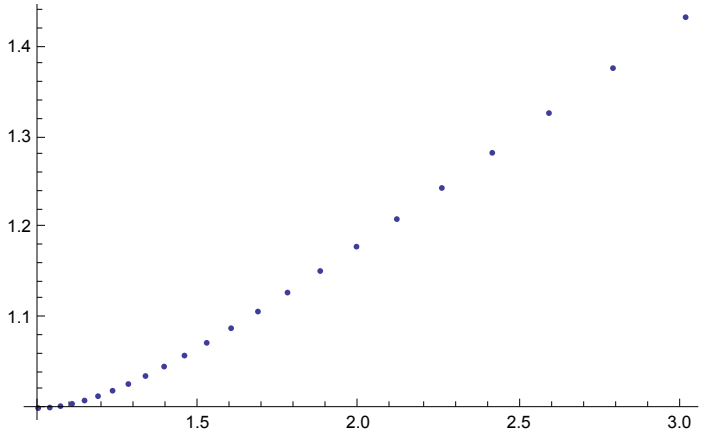
$$-4.40719 \times 10^{-9}$$

Finally, we convert the data generated by the iteration above into a sequence of points and plot them alongside the exact solution.

```
rkfPts = Table[{rkfList[[k, 1]], rkfList[[k, 3]]}, {k, 1, Length[rkfList]}]
```

```
{{1., 1.}, {1.03608, 1.00061}, {1.06937, 1.00216}, {1.10546, 1.00472},
 {1.14429, 1.00838}, {1.18619, 1.01319}, {1.23149, 1.01926},
 {1.28057, 1.02667}, {1.33387, 1.03555}, {1.39189, 1.04601}, {1.45519, 1.05822},
 {1.52443, 1.07232}, {1.60036, 1.08851}, {1.68384, 1.10701}, {1.77588, 1.12805},
 {1.87765, 1.15192}, {1.9905, 1.17894}, {2.11604, 1.20948}, {2.25614, 1.24397},
 {2.41306, 1.28293}, {2.5895, 1.32695}, {2.78875, 1.37676}, {3.01486, 1.43322}}
```

```
plot2 = ListPlot[rkfPts]
```



This shows quite clearly that the method was using a relatively small step size in regions where the solution was changing quickly and relaxing to a larger step size when the solution became more linear.

```
Show[plot1, plot2]
```