Boyce/DiPrima/Meade 11th ed, Ch 8.1: Numerical Methods:The Euler or Tangent Line Method

- The methods we have discussed for solving differential equations have emphasized analytical techniques, such as integration or series solutions, to find exact solutions.
- However, there are many important problems in engineering and science, especially nonlinear ones, to which these methods either do not apply or are complicated to use.
- In this chapter we discuss the use of numerical methods to approximate the solution of an initial value problem.
- We study these methods as applied to single first order equations, the simplest context to learn the methods.
- These procedures extend to systems of first order equations, and this is outlined in Chapter 8.5.

Euler's Method

- We will concentrate on the first order initial value problem

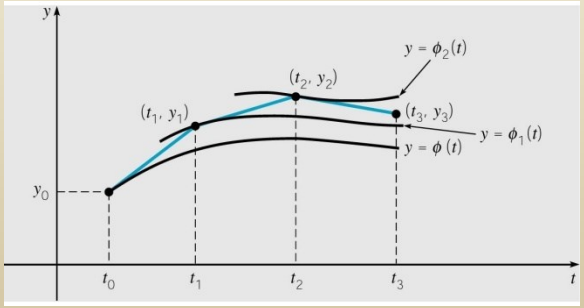
$$\frac{dy}{dt} = f(t, y), \quad y(t_0) = y_0$$

- Recall that if f and f_y are continuous, then this IVP has a unique solution $y = \phi(t)$ in some interval about t_0.
- In Chapter 2.7, Euler's method was formulated as

$$y_{n+1} = y_n + f_n \cdot (t_{n+1} - t_n), \quad n = 0, 1, 2, \ldots$$

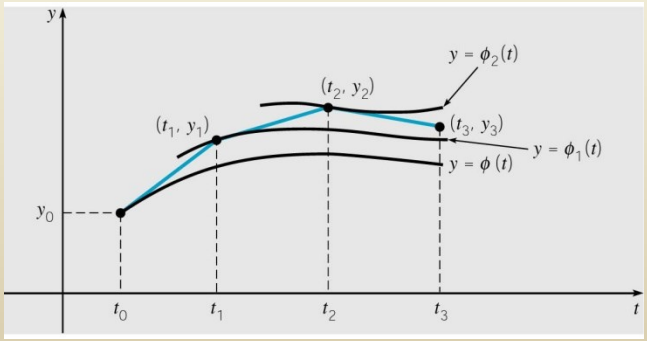
where $f_n = f(t_n, y_n)$. For a uniform step size $h = t_n - t_{n\text{-}1}$, Euler's method becomes

$$y_{n+1} = y_n + f_n h, \quad n = 0, 1, 2, \ldots$$



Euler's Method: Programming Outline

- A computer program for Euler's method with a uniform step size will have the following structure.
  - Step 1. Define $f(t,y)$
  - Step 2. Input initial values t0 and y0
  - Step 3. Input step size h and number of steps n
  - Step 4. Output t0 and y0
  - Step 5. For j from 1 to n do
  - Step 6. $k1 = f(t,y)$
    $y = y + h*k1$
    $t = t + h$
  - Step 7. Output t and y
  - Step 8. End



Initial Value Problem & Exact Solution (1 of 2)

- Throughout this chapter, we will use the initial value problem below to illustrate and compare different numerical methods.

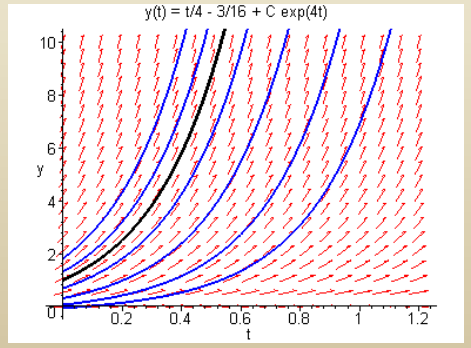
$$y' = 1 - t + 4y, \quad y(0) = 1$$

- Using the methods of Chapter 2.1, it can be shown that the general solution to the differential equation is

$$y(t) = \frac{1}{4}t - \frac{3}{16} + Ce^{4t},$$

while the solution to the initial value problem is

$$y = \phi(t) = \frac{1}{4}t - \frac{3}{16} + \frac{19}{16}e^{4t}$$



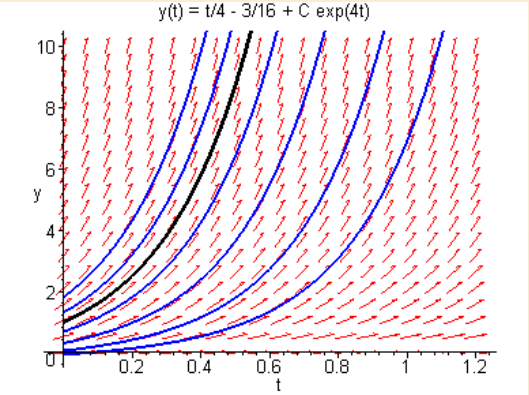
Exact Solution and Integral Curves (2 of 2)

- Thus the exact solution of

$$y' = 1 - t + 4y, \quad y(0) = 1$$

is given by

$$y = \phi(t) = \frac{1}{4}t - \frac{3}{16} + \frac{19}{16}e^{4t}$$



- In the graph above, the direction field for the differential equation is given, along with the solution curve for the initial value problem (black) and several integral curves (blue) for the general solution of the differential equation.
- The integral curves diverge rapidly from each other, and thus it may be difficult for our numerical methods to approximate the solution accurately. However, it will be relatively easy to observe the benefits of the more accurate methods.

$$y' = 1 - t + 4y, \quad y(0) = 1$$

Example 1: Euler's Method (1 of 3)

- The table below compares the results of Euler's method, with step sizes $h = 0.05, 0.025, 0.01, 0.001$, and the exact solution values, on the interval $0 \le t \le 2$. We have used the formula

$$\text{Relative Error} = \left| \frac{y_{exact} - y_{approx}}{y_{exact}} \right| \times 100$$

t	h = 0.05	h = 0.025	h = 0.01	h = 0.001	Exact	Rel Error h = 0.05	Rel Error h = 0.025	Rel Error h = 0.01	Rel Error h = 0.001
0.00	1.0000	1.0000	1.0000	1.0000	1.0000	0.00	0.00	0.00	0.00
0.10	1.5475	1.5761	1.5953	1.6076	1.6090	3.82	2.04	0.85	0.09
0.20	2.3249	2.4080	2.4646	2.5011	2.5053	7.20	3.88	1.63	0.17
0.30	3.4334	3.6144	3.7390	3.8207	3.8301	10.36	5.63	2.38	0.25
0.40	5.0185	5.3690	5.6137	5.7755	5.7942	13.39	7.34	3.12	0.32
0.50	7.2902	7.9264	8.3767	8.6771	8.7120	16.32	9.02	3.85	0.40
1.00	45.5884	53.8079	60.0371	64.3826	64.8978	29.75	17.09	7.49	0.79
1.50	282.0719	361.7595	426.4082	473.5598	479.2592	41.14	24.52	11.03	1.19
2.00	1745.6662	2432.7878	3029.3279	3484.1608	3540.2001	50.69	31.28	14.43	1.58

Example 1: Discussion of Accuracy (2 of 3)

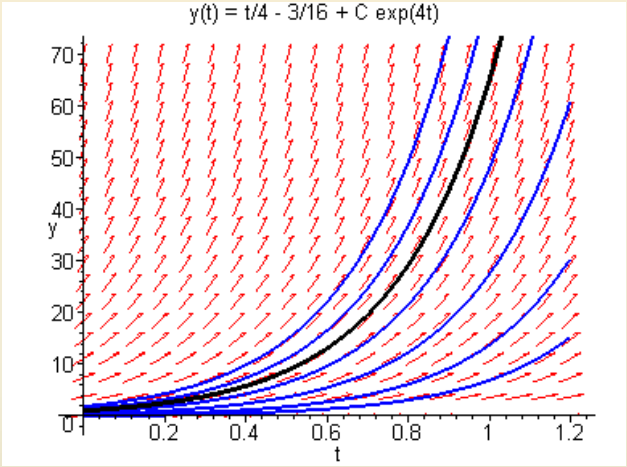
- The errors are reasonably small when $h = 0.001$. However, 2000 steps are required to traverse interval from $t = 0$ to $t = 2$. Thus considerable computation is needed to obtain reasonably good accuracy for Euler's method.
- We will see later in this chapter that with other numerical approximations it is possible to obtain comparable or better accuracy with larger step sizes and fewer computational steps.

t	h = 0.05	h = 0.025	h = 0.01	h = 0.001	Exact	Rel Error h = 0.05	Rel Error h = 0.025	Rel Error h = 0.01	Rel Error h = 0.001
0.00	1.0000	1.0000	1.0000	1.0000	1.0000	0.00	0.00	0.00	0.00
0.10	1.5475	1.5761	1.5953	1.6076	1.6090	3.82	2.04	0.85	0.09
0.20	2.3249	2.4080	2.4646	2.5011	2.5053	7.20	3.88	1.63	0.17
0.30	3.4334	3.6144	3.7390	3.8207	3.8301	10.36	5.63	2.38	0.25
0.40	5.0185	5.3690	5.6137	5.7755	5.7942	13.39	7.34	3.12	0.32
0.50	7.2902	7.9264	8.3767	8.6771	8.7120	16.32	9.02	3.85	0.40
1.00	45.5884	53.8079	60.0371	64.3826	64.8978	29.75	17.09	7.49	0.79
1.50	282.0719	361.7595	426.4082	473.5598	479.2592	41.14	24.52	11.03	1.19
2.00	1745.6662	2432.7878	3029.3279	3484.1608	3540.2001	50.69	31.28	14.43	1.58

Example 1: Table and Graph (3 of 3)

- The table of numerical results along with the graphs of several integral curves are given below for comparison.

t	h = 0.05	h = 0.025	h = 0.01	h = 0.001	Exact
0.00	1.0000	1.0000	1.0000	1.0000	1.0000
0.10	1.5475	1.5761	1.5953	1.6076	1.6090
0.20	2.3249	2.4080	2.4646	2.5011	2.5053
0.30	3.4334	3.6144	3.7390	3.8207	3.8301
0.40	5.0185	5.3690	5.6137	5.7755	5.7942
0.50	7.2902	7.9264	8.3767	8.6771	8.7120
1.00	45.5884	53.8079	60.0371	64.3826	64.8978
1.50	282.0719	361.7595	426.4082	473.5598	479.2592
2.00	1745.6662	2432.7878	3029.3279	3484.1608	3540.2001



Alternative 1: Forward Difference Quotient

- To begin to investigate errors in numerical approximations, and to suggest ways to construct more accurate algorithms, we examine some alternative ways to look at Euler's method.
- Let $y = \phi(t)$ be the solution of $y' = f(t,y)$. At $t = t_n$, we have

$$\phi' = f\left(t_n, \phi(t_n)\right)$$

- Using a forward difference quotient for ϕ', it follows that

$$\frac{\phi(t_{n+1}) - \phi(t_n)}{t_{n+1} - t_n} \cong f\left(t_n, \phi(t_n)\right)$$

- Replacing $\phi(t_{n+1})$ and $\phi(t_n)$ by their approximate values y_{n+1} and y_n, and then solving for y_{n+1}, we obtain Euler's formula:

$$y_{n+1} = y_n + f(t_n, y_n) \cdot \left(t_{n+1} - t_n\right), \quad n = 0,1,2,\ldots$$

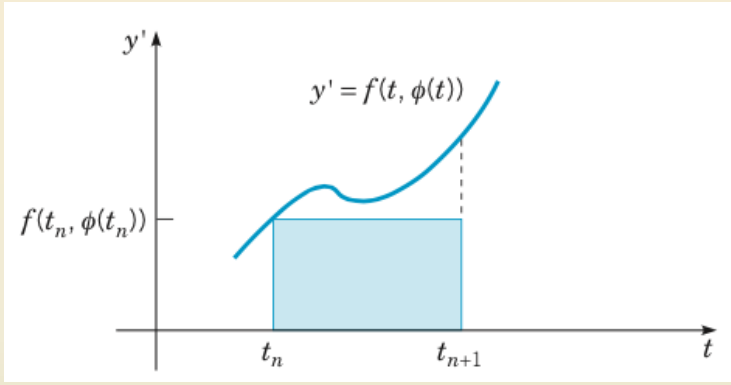
Alternative 2: Integral Equation

- We could write problem as an integral equation. That is, since $y = \phi(t)$ is a solution of $y' = f(t,y)$, $y(t_0) = y_0$, we have

$$\int_{t_n}^{t_{n+1}} \phi'(t)dt = \int_{t_n}^{t_{n+1}} f\big(t,\phi(t)\big)dt$$

or

$$\phi(t_{n+1}) = \phi(t_n) + \int_{t_n}^{t_{n+1}} f\big(t,\phi(t)\big)dt$$



- Approximating the above integral (see figure on right), we obtain

$$\phi(t_{n+1}) \cong \phi(t_n) + f\big(t_n,\phi(t_n)\big)\big(t_{n+1} - t_n\big)$$

- Replacing $\phi(t_{n+1})$ and $\phi(t_n)$ by their approximate values y_{n+1} and y_n, we obtain Euler's formula:

$$y_{n+1} = y_n + f(t_n, y_n)\cdot\big(t_{n+1} - t_n\big), \quad n = 0,1,2,\ldots$$

Alternative 3: Taylor Series

- A third approach is to assume the solution $y = \phi(t)$ has a Taylor series about $t = t_n$. Then

$$\phi(t) = \phi(t_n) + \phi'(t_n)(t - t_n) + \frac{1}{2!}\phi''(t_n)(t - t_n)^2 + \cdots$$

- Since $h = t_{n+1} - t_n$ and $\phi' = f(t, \phi)$, it follows that

$$\phi(t_{n+1}) = \phi(t_n) + f(t_n, \phi(t_n))h + \frac{1}{2!}\phi''(t_n)h^2 + \cdots$$

- If the series is terminated after the first two terms, and if we replace $\phi(t_{n+1})$ and $\phi(t_n)$ by their approximations y_{n+1} and y_n, then once again we obtain Euler's formula:

$$y_{n+1} = y_n + f(t_n, y_n) \cdot (t_{n+1} - t_n), \quad n = 0, 1, 2, \ldots$$

- Further, using a Taylor series with remainder, we can estimate the magnitude of error in this formula (later in this section).

Backward Euler Formula

- The backward Euler formula is derived as follows. Let $y = \phi(t)$ be the solution of $y' = f(t,y)$. At $t = t_n$, we have
$$\phi' = f\left(t_n, \phi(t_n)\right)$$
- Using a backward difference quotient for ϕ', it follows that
$$\frac{\phi(t_n) - \phi(t_{n-1})}{t_n - t_{n-1}} \cong f\left(t_n, \phi(t_n)\right)$$
- Replacing $\phi(t_n)$ and $\phi(t_{n-1})$ by their approximations y_n and y_{n-1}, and solving for y_n, we obtain the backward Euler formula
$$y_n = y_{n-1} + f(t_n, y_n) \cdot h \quad \Leftrightarrow \quad y_{n+1} = y_n + f(t_{n+1}, y_{n+1}) \cdot h$$
- Note that this equation implicitly defines y_{n+1}, and must be solved in order to determine the value of y_{n+1}.

Example 2: Backward Euler Formula (1 of 4)

- For our initial value problem

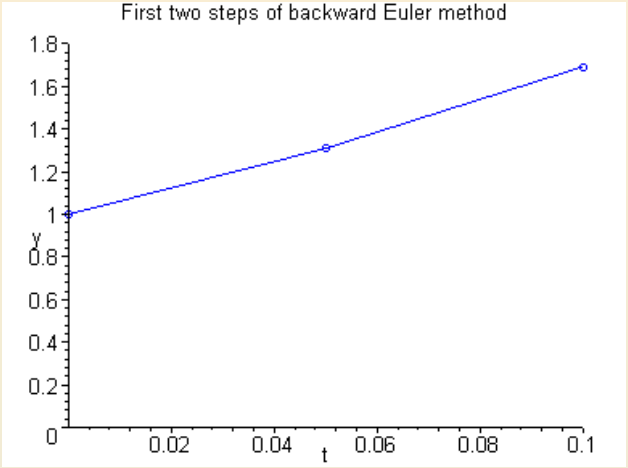
$$y' = 1 - t + 4y, \quad y(0) = 1,$$

the backward Euler formula

$$y_{n+1} = y_n + f(t_{n+1}, y_{n+1}) \cdot h$$

becomes

$$y_{n+1} = y_n + (1 - t_{n+1} + 4y_{n+1})h$$



- For $h = 0.05$ on the interval $0 \le t \le 2$, our first two steps are:

$$y_1 = 1 + (1 - 0.05 + 4y_1)(0.05) \Rightarrow y_1 \cong 1.3094$$

$$y_2 = 1.3094 + (1 - 0.1 + 4y_2)(0.05) \Rightarrow y_2 \cong 1.6930$$

- The results of these first two steps of the backward Euler method are graphed above.

$$y' = 1 - t + 4y, \quad y(0) = 1$$

Example 2: Numerical Results (2 of 4)

- The table below compares the results of the backward Euler method, with step sizes $h = 0.05, 0.025, 0.01, 0.001$, and the exact solution values, on the interval $0 \le t \le 2$.

t	h = 0.05	h = 0.025	h = 0.01	h = 0.001	Exact	Rel Error h = 0.05	Rel Error h = 0.025	Rel Error h = 0.01	Rel Error h = 0.001
0.00	1.0000	1.0000	1.0000	1.0000	1.0000	0.00	0.00	0.00	0.00
0.10	1.6930	1.6474	1.6237	1.6104	1.6090	5.22	2.39	0.91	0.09
0.20	2.7617	2.6211	2.5491	2.5096	2.5053	10.23	4.62	1.75	0.17
0.30	4.4175	4.0921	3.9286	3.8396	3.8301	15.34	6.84	2.57	0.25
0.40	6.9906	6.3210	5.9908	5.8131	5.7942	20.65	9.09	3.39	0.33
0.50	10.9970	9.7050	9.0801	8.7473	8.7120	26.23	11.40	4.23	0.41
1.00	103.0617	80.4028	70.4524	65.4200	64.8978	58.81	23.89	8.56	0.80
1.50	959.4424	661.0073	542.1243	485.0583	479.2592	100.19	37.92	13.12	1.21
2.00	8934.0696	5435.7294	4172.7228	3597.4478	3540.2001	152.36	53.54	17.87	1.62

Example 2: Discussion of Accuracy (3 of 4)

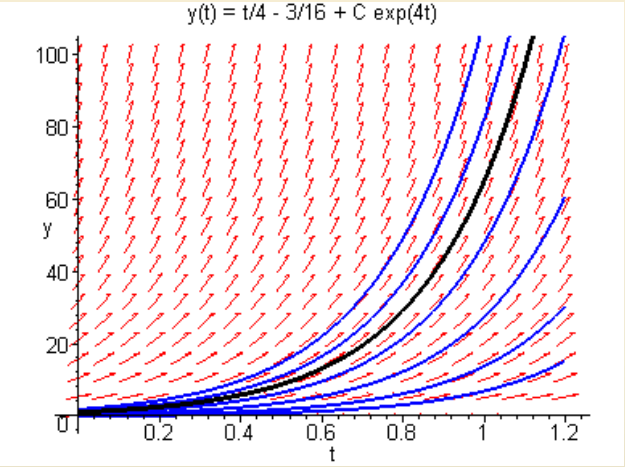
- The errors here are larger than for regular Euler method, although for small values of *h* the differences are small.
- The approximations consistently overestimate exact values, while Euler method approximations underestimated them.
- We will see later in this chapter that the backward Euler method is the simplest example of backward differentiation methods, which are useful for certain types of equations.

t	h = 0.05	h = 0.025	h = 0.01	h = 0.001	Exact	Rel Error h = 0.05	Rel Error h = 0.025	Rel Error h = 0.01	Rel Error h = 0.001
0.00	1.0000	1.0000	1.0000	1.0000	1.0000	0.00	0.00	0.00	0.00
0.10	1.6930	1.6474	1.6237	1.6104	1.6090	5.22	2.39	0.91	0.09
0.20	2.7617	2.6211	2.5491	2.5096	2.5053	10.23	4.62	1.75	0.17
0.30	4.4175	4.0921	3.9286	3.8396	3.8301	15.34	6.84	2.57	0.25
0.40	6.9906	6.3210	5.9908	5.8131	5.7942	20.65	9.09	3.39	0.33
0.50	10.9970	9.7050	9.0801	8.7473	8.7120	26.23	11.40	4.23	0.41
1.00	103.0617	80.4028	70.4524	65.4200	64.8978	58.81	23.89	8.56	0.80
1.50	959.4424	661.0073	542.1243	485.0583	479.2592	100.19	37.92	13.12	1.21
2.00	8934.0696	5435.7294	4172.7228	3597.4478	3540.2001	152.36	53.54	17.87	1.62

Example 2: Table and Graph (4 of 4)

- The table of numerical results along with the graphs of several integral curves are given below for comparison.

t	h = 0.05	h = 0.025	h = 0.01	h = 0.001	Exact
0.00	1.0000	1.0000	1.0000	1.0000	1.0000
0.10	1.6930	1.6474	1.6237	1.6104	1.6090
0.20	2.7617	2.6211	2.5491	2.5096	2.5053
0.30	4.4175	4.0921	3.9286	3.8396	3.8301
0.40	6.9906	6.3210	5.9908	5.8131	5.7942
0.50	10.9970	9.7050	9.0801	8.7473	8.7120
1.00	103.0617	80.4028	70.4524	65.4200	64.8978
1.50	959.4424	661.0073	542.1243	485.0583	479.2592
2.00	8934.0696	5435.7294	4172.7228	3597.4478	3540.2001



Errors in Numerical Approximations

- The use of a numerical procedure, such as Euler's formula, to solve an initial value problem raises questions that must be answered before the approximate numerical solution can be accepted as satisfactory.
- For example, as the step size h tends to zero, do the values y_1, y_2, …, y_n, … converge to the values of the actual solution?
- Also, an estimation of error in computing y_n is important. Two fundamental sources of error are the following.
  - Global truncation error, due to approximate formulas used to determine the values of y_n, and approximate data input into these formulas.
  - Round-off error, due to finite precision arithmetic.

Convergence

- As the step size h tends to zero, do the values $y_1, y_2, \ldots, y_n, \ldots$ converge to the values of the actual solution, for each t?
- If the approximations converge to the solution, how small a step size is needed to guarantee a given level of accuracy?
  - We want to use a step size that is small enough to ensure the required accuracy, but not too small.
  - An unnecessarily small step size slows down calculations, makes them more expensive, and in some cases may even cause a loss of accuracy.

t	h = 0.05	h = 0.025	h = 0.01	h = 0.001	Exact	Rel Error h = 0.05	Rel Error h = 0.025	Rel Error h = 0.01	Rel Error h = 0.001
0.00	1.0000	1.0000	1.0000	1.0000	1.0000	0.00	0.00	0.00	0.00
0.10	1.5475	1.5761	1.5953	1.6076	1.6090	3.82	2.04	0.85	0.09
0.20	2.3249	2.4080	2.4646	2.5011	2.5053	7.20	3.88	1.63	0.17
0.30	3.4334	3.6144	3.7390	3.8207	3.8301	10.36	5.63	2.38	0.25
0.40	5.0185	5.3690	5.6137	5.7755	5.7942	13.39	7.34	3.12	0.32
0.50	7.2902	7.9264	8.3767	8.6771	8.7120	16.32	9.02	3.85	0.40
1.00	45.5884	53.8079	60.0371	64.3826	64.8978	29.75	17.09	7.49	0.79
1.50	282.0719	361.7595	426.4082	473.5598	479.2592	41.14	24.52	11.03	1.19
2.00	1745.6662	2432.7878	3029.3279	3484.1608	3540.2001	50.69	31.28	14.43	1.58

Global and Local Truncation Error

- Assume here that we can carry out all computations with complete accuracy. That is, we can retain an infinite number of decimal places with no round-off error.
- At each step in a numerical method, the solution value $\phi(t_n)$ is approximated by the value y_n.
- The **global truncation error** is defined as

$$E_n = \phi(t_n) - y_n$$

- This error arises from two causes:
  1. At each step we use an approximate formula to determine y_{n+1}.
  2. The input data at each step are only approximately correct, since $\phi(t_n)$ in general does not equal y_n.
- If we assume that $y_n = \phi(t_n)$ at step n, then the only error at step $n+1$ is due to the use of an approximate formula. This error is known as the **local truncation error** e_n.

Round-Off Error

- Round-off error occurs from carrying out computations in arithmetic with only a finite number of digits.
- As a result, the value of y_n, derived from an approximation formula, is in turn approximated by its computed value Y_n.
- Thus **round-off error** is defined as

$$R_n = y_n - Y_n.$$

- Round-off error is somewhat random in nature. It depends on type of computer used, the sequence in which computations are carried out, the method of rounding off, etc. Therefore, an analysis of round-off error is beyond the scope of this course.

Total Error

- From the discussion on the previous slides, we see that at each step the solution value $\phi(t_n)$ is approximated by the value y_n, which in turn is approximated by its computed value Y_n.
- The total error can therefore be taken as $T_n = \phi(t_n) - Y_n$.
- From the triangle inequality, $|a + b| \leq |a| + |b|$, it follows that

$$\begin{aligned} |\phi(t_n) - Y_n| &= |\phi(t_n) - y_n + y_n - Y_n| \\ &\leq |\phi(t_n) - y_n| + |y_n - Y_n| \\ &\leq |E_n| + |R_n| \end{aligned}$$

- Thus the total error is bounded by the sum of the absolute values of the truncation and round-off errors.
- We will limit our discussion primarily to local truncation error.

Local Truncation Error for Euler Method (1 of 2)

- Assume that $y = \phi(t)$ is a solution to $\phi' = f(t, \phi)$, $y(t_0) = y_0$ and that f, f_t and f_y are continuous. Then ϕ'' is continuous, where

$$\phi''(t) = f_t\big(t, \phi(t)\big) + f_y\big(t, \phi(t)\big) f\big(t, \phi(t)\big)$$

- Using a Taylor polynomial with a remainder to expand $\phi(t)$ about $t = t_n$, we have

$$\phi(t) = \phi(t_n) + \phi'(t_n)\big(t - t_n\big) + \frac{1}{2!}\phi''(\bar{t}_n)\big(t - t_n\big)^2$$

where $\bar{t}_n$ is some point in the interval $t_n < \bar{t}_n < t_{n+1}$.

- Since $h = t_{n+1} - t_n$ and $\phi' = f(t, \phi)$, it follows that

$$\phi(t_{n+1}) = \phi(t_n) + f\big(t_n, \phi(t_n)\big)h + \frac{1}{2!}\phi''(\bar{t}_n)h^2$$

Local Truncation Error for Euler Method (2 of 2)

- From the previous slide, we have

$$\phi(t_{n+1}) = \phi(t_n) + f\big(t_n, \phi(t_n)\big)h + \frac{1}{2!}\phi''(\bar{t}_n)h^2$$

- Recalling the Euler formula

$$y_{n+1} = y_n + f(t_n, y_n)h,$$

it follows that

$$\phi(t_{n+1}) - y_{n+1} = \big[\phi(t_n) - y_n\big] + \big[f\big(t_n, \phi(t_n)\big) - f(t_n, y_n)\big]h + \frac{1}{2!}\phi''(\bar{t}_n)h^2$$

- To compute the local truncation error e_{n+1}, we take $y_n = \phi(t_n)$ and hence

$$e_{n+1} = \phi(t_{n+1}) - y_{n+1} = \frac{1}{2!}\phi''(\bar{t}_n)h^2$$

Uniform Bound for Local Truncation Error

- Thus the local truncation error is proportional to the square of the step size h, and the proportionality constant depends on ϕ''.

$$e_{n+1} = \phi(t_{n+1}) - y_{n+1} = \frac{1}{2!}\phi''(\bar{t}_n)h^2$$

- Thus e_{n+1} depends on n, and hence is typically different for each step. A uniform bound, valid on an interval $[a, b]$, is

$$|e_n| \le \frac{M}{2}h^2, \quad M = \max\{\phi''(t) : a < t < b\}$$

- This bound represents the worst possible case, and may well be a considerable overestimate of the actual truncation error in some parts of the interval $[a, b]$.

Step Size and Local Truncation Error Bound

- From the previous slide we have

$$|e_n| \le \frac{M}{2} h^2, \quad M = \max\left\{ |\phi''(t)| : a < t < b \right\}$$

- One use of this bound is to choose a step size h that will result in a local truncation error no greater than some given tolerance level ε. That is, we choose h such that

$$\frac{M}{2} h^2 \le \varepsilon \;\Rightarrow\; h \le \sqrt{\frac{2\varepsilon}{M}}$$

- It can be difficult estimating $|\phi''(t)|$ or M. However, the central fact is that e_n is proportional to h^2. Thus if h is reduced by a factor of ½, then the error is reduced by ¼, and so on.

Estimating Global Truncation Error

- Using the local truncation error e_n, we can make an intuitive estimate for the global truncation error E_n at a fixed $T > t_0$.
- Taking n steps, from t_0 to $T = t_0 + nh$, the error at each step is at most $Mh^2/2$, and hence error in n steps is at most $nMh^2/2$.
- Thus the global truncation error E_n for the Euler method is

$$|E_n| \le \frac{nM}{2} h^2 = (T - t_0)\frac{M}{2} h$$

- This argument is not complete since it does not consider the effect an error at one step will have in succeeding steps.
- Nevertheless, it can be shown that for a finite interval, E_n is bounded by a constant times h, and hence Euler's method is a **first order method**.

Example 2 Extension: Local Truncation Error (1 of 4)

- Consider again our initial value problem

$$y' = 1 - t + 4y, \quad y(0) = 1; \quad 0 \le t \le 2$$

- Using the solution $\phi(t)$, we have

$$\phi(t) = \left(4t - 3 + 19e^{4t}\right)/16 \;\Rightarrow\; \phi''(t) = 19e^{4t}$$

- Thus the local truncation error e_{n+1} at step $n + 1$ is given by

$$e_{n+1} = \frac{\varphi''(\bar{t}_n)}{2}h^2 = \frac{19e^{4\bar{t}_n}}{2}h^2, \; t_n < \bar{t}_n < t_n + h$$

- The presence of the factor 19 and the rapid growth of e^{4t} explains why the numerical approximations in this section with $h = 0.05$ were not very accurate.

Example 2 Extension: Error in First Step (2 of 4)

- For $h = 0.05$, the error in the first step is

$$e_1 = \frac{19e^{4\bar{t}_0}}{2}(0.0025),\ 0 < \bar{t}_0 < 0.05$$

- Since $1 < e^{4\bar{t}_0} < e^{4(0.05)} = e^{0.02}$, it follows that

$$0.02375 \cong \frac{19}{2}(0.0025) < e_1 < \frac{19e^{0.2}}{2}(0.0025) \cong 0.02901$$

- It can be shown that the actual error is 0.02542.
- Similar computations give the following bounds:

$$1.0617 < e_{20} < 1.2967 \quad (0.95 < t < 1.0)$$

$$57.96 < e_{40} < 70.80 \quad (1.95 < t < 2.0)$$

Example 2 Extension: Error Bounds & Step Size (3 of 4)

- We have the following error bounds e_n for $h = 0.05$:

$$0.02375 < e_1 < 0.02901 \quad (0 < t < 0.05)$$

$$57.96 < e_{40} < 70.80 \quad (1.95 < t < 2.0)$$

- Note that the error near $t = 2$ is close to 2500 times larger than it is near $t = 0$.
- To reduce local truncation error throughout $0 \le t \le 2$, we must choose a step size based on an analysis near $t = 2$.
- For example, to achieve $e_n < 0.01$ throughout $0 \le t \le 2$, note that $M = 19e^{4(2)}$, and hence the required step size h is

$$h \le \sqrt{2\varepsilon / M} \cong 0.00059$$

Example 2 Extension: Error Tolerance & Uniform Step Size (4 of 4)

- Thus, in order to achieve $e_n < 0.01$ throughout $0 \le t \le 2$, the required step size $h = 0.00059$. Comparing this with a similar calculation over $0 \le t \le 0.05$, we obtain $h = 0.02936$.
- Some disadvantages in using a uniform step size is that h is much smaller than necessary over much of the interval, and the numerical method will then require more time and calculations than necessary. Also, as a result, there is a possibility of more unacceptable round-off errors.
- Another to approach to keeping within error tolerance is to gradually decrease h as t increases. Such a procedure is called an **adaptive method**, and is discussed in Chapter 8.2.

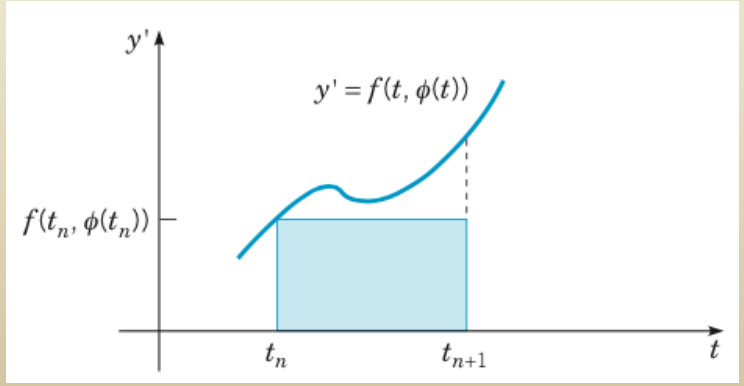
Boyce/DiPrima/Meade 11th ed, Ch 8.2: Improvements on the Euler Method

- Consider the initial value problem $y' = f(t, y)$, $y(t_0) = y_0$, with solution $\phi(t)$.
- For many problems, Euler's method requires a very small step size to produce sufficiently accurate results. In the next three sections, we will discuss several more efficient methods.
- In this section, we examine the **Improved Euler Formula**, or **Heun formula.** This method better approximates the integral introduced in Chapter 8.1, where we had

$$\int_{t_n}^{t_{n+1}} \phi'(t)dt = \int_{t_n}^{t_{n+1}} f\big(t, \phi(t)\big)dt$$

$$\phi(t_{n+1}) = \phi(t_n) + \int_{t_n}^{t_{n+1}} f\big(t, \phi(t)\big)dt$$

$$\phi(t_{n+1}) \cong \phi(t_n) + f\big(t_n, \phi(t_n)\big)\big(t_{n+1} - t_n\big)$$



Improved Euler Method

- Consider again the integral equation

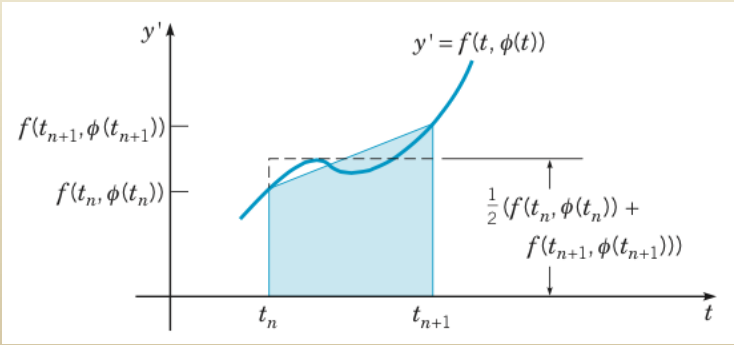
$$\phi(t_{n+1}) = \phi(t_n) + \int_{t_n}^{t_{n+1}} f\big(t_n, \phi(t_n)\big)dt$$

- Approximating the integrand $f\,(t_n, \phi(t_n))$ with the average of its values at the two endpoints (see graph below), we obtain

$$\phi(t_{n+1}) \cong \phi(t_n) + \frac{f\big(t_n, \phi(t_n)\big) + f\big(t_{n+1}, \phi(t_{n+1})\big)}{2}\big(t_{n+1} - t_n\big)$$

- Replacing $\phi(t_{n+1})$ and $\phi(t_n)$ by y_{n+1} and y_n, we obtain

$$y_{n+1} = y_n + \frac{f\big(t_n, y_n\big) + f\big(t_{n+1}, y_{n+1}\big)}{2}h$$



Improved Euler Method

- Our formula defines y_{n+1} implicitly, instead of explicitly:

$$y_{n+1} = y_n + \frac{f(t_n, y_n) + f(t_{n+1}, y_{n+1})}{2} h$$

- Replacing y_{n+1} by its value from the Euler formula

$$y_{n+1} = y_n + f(t_n, y_n)h,$$

we obtain the **improved Euler formula**

$$y_{n+1} = y_n + \frac{f_n + f(t_n + h, y_n + f_n h)}{2} h,$$

where $f_n = f(t_n, y_n)$ and $t_{n+1} = t_n + h$.

Error Estimates

- The improved Euler formula is

$$y_{n+1} = y_n + \frac{h}{2}\left[f_n + f\left(t_n + h, y_n + f_n h\right)\right]$$

- It can be shown that the local truncation error is proportional to h^3. This is an improvement over the Euler method, where the local truncation error is proportional to h^2.
- For a finite interval, it can be shown that the global truncation error is bounded by a constant times h^2. Thus the improved Euler method is a second order method, whereas Euler's method is a first order method.
- This greater accuracy comes at expense of more computational work, as it is now necessary to evaluate $f(t, y)$ twice in order to go from t_n to t_{n+1}.

Comparison of Euler and Improved Euler Computations

- The Euler and improved Euler formulas are given by, respectively,

$$y_{n+1} = y_n + f(t_n, y_n)h$$

$$y_{n+1} = y_n + (0.5h)\left[f_n + f\left(t_n + h, y_n + f_n h\right)\right]$$

- The improved Euler method requires two evaluations of *f* at each step, whereas the Euler method requires only one.
- This is significant because typically most of the computing time in each step is spent evaluating *f*.
- Thus, for a given step size *h*, the improved Euler method requires twice as many evaluations of *f* as the Euler method.
- Alternatively, the improved Euler method for step size *h* requires the same number of evaluations of *f* as the Euler method with step size *h*/2.

Trapezoid Rule

- The improved Euler formula is

$$y_{n+1} = y_n + \frac{f_n + f\left(t_n + h, y_n + f_n h\right)}{2} h$$

- If $f(t, y)$ depends only on t and not on y, then we have

$$y_{n+1} - y_n = \frac{h}{2}\left[f\left(t_n\right) + f\left(t_n + h\right)\right],$$

which is the trapezoid rule for numerical integration.

$$y_{n+1} = y_n + \frac{h}{2}\left[f_n + f\left(t_n + h, y_n + f_n h\right)\right]$$

Example 1: Improved Euler Method (1 of 3)

- For our initial value problem

$$y' = 1 - t + 4y, \quad y(0) = 1,$$

we have

$$f_n = f(t_n, y_n) = 1 - t_n + 4y_n,$$
$$f(t_n + h, y_n + hf_n) = 1 - \left(t_n + h\right) + 4\left(y_n + hf_n\right)$$

- Further,

$$t_0 = 0,\ y_0 = 1 \ \Rightarrow\ f_0 = 1 - t_0 + 4y_0 = 5$$

- For $h = 0.025$, it follows that

$$f(t_0 + h, y_0 + hf_0) = 1 - 0.025 + 4\left(1 + (0.025)(5)\right) = 5.475$$

- Thus

$$y_1 = 1 + (0.5)(0.025)(5 + 5.475) = 1.1309375$$

$$y_{n+1} = y_n + \frac{h}{2}\left[f_n + f\left(t_n + h, y_n + f_n h\right)\right]$$

Example 1: Second Step (2 of 3)

- For the second step, we have

$$t_1 = 0.025,\ y_1 = 1.1309375,\ \text{and}$$
$$f_1 = 1 - 0.025 + 4(1.1309375) = 5.49875$$

- Also,

$$y_1 + hf_1 = 1.1309375 + (0.025)(5.49875) = 1.26840625$$

and

$$f(t_1, y_1 + h f_1) = 1 - 0.05 + 4(1.26840625) = 6.023625$$

- Therefore

$$y_2 = 1.1309375 + (0.5)(0.025)(5.49875 + 6.023625)$$
$$= 1.2749671875$$

$$y' = 1 - t + 4y, \quad y(0) = 1$$

Example 1: Numerical Results (3 of 3)

- Recall that for a given step size h, the improved Euler formula requires the same number of evaluations of f as the Euler method with step size $h/2$.
- From the table below, we see that the improved Euler method is more efficient, and yields substantially better results or requiring much less total computing effort, or both.

	Euler		Improved	Euler		Euler		Improved	Euler
t	h = 0.01	h = 0.001	h = 0.025	h = 0.01	Exact	Rel Error h = 0.01	Rel Error h = 0.001	Rel Error h = 0.025	Rel Error h = 0.01
0.00	1.0000	1.0000	1.0000	1.0000	1.0000	0.00	0.00	0.00	0.00
0.10	1.5953	1.6076	1.6079	1.6089	1.6090	0.85	0.09	0.07	0.01
0.20	2.4646	2.5011	2.5021	2.5048	2.5053	1.63	0.17	0.13	0.02
0.30	3.7390	3.8207	3.8223	3.8289	3.8301	2.38	0.25	0.20	0.03
0.40	5.6137	5.7755	5.7797	5.7918	5.7942	3.12	0.32	0.25	0.04
0.50	8.3767	8.6771	8.6849	8.7075	8.7120	3.85	0.40	0.31	0.05
1.00	60.0371	64.3826	64.4979	64.8307	64.8978	7.49	0.79	0.62	0.10
1.50	426.4082	473.5598	474.8340	478.5159	479.2592	11.03	1.19	0.92	0.16
2.00	3029.3279	3484.1608	3496.6702	3532.8789	3540.2001	14.43	1.58	1.23	0.21

Programming Outline

- A computer program for the improved Euler's method with a uniform step size will have the following structure.
  - Step 1. Define $f(t,y)$
  - Step 2. Input initial values t0 and y0
  - Step 3. Input step size h and number of steps n
  - Step 4. Output t0 and y0
  - Step 5. For j from 1 to n do
  - Step 6. $k1 = f(t, y)$

    $k2 = f(t + h, y + h*k1)$

    $y = y + (h/2)*(k1 + k2)$

    $t = t + h$
  - Step 7. Output t and y
  - Step 8. End

Variation of Step Size (1 of 2)

- In Chapter 8.1 we mentioned that adaptive methods adjust the step size as calculations proceed, so as to maintain the local truncation error at roughly a constant level. We outline such a method here.
- Suppose that after n steps, we have reached the point (t_n, y_n).
- We next choose an h and then calculate y_{n+1}.
- To estimate the error in calculating y_{n+1}, we can use a more accurate method to calculate y_{n+1} starting from (t_n, y_n). For example, if we used the Euler method for original calculation, then we might repeat it with the improved Euler method.
- The difference of the two calculated values for y_{n+1} is an estimate d_{n+1} of the error in the original method.

Variation of Step Size (2 of 2)

- If d_{n+1} is different than the error tolerance ε, then we adjust the step size and repeat the calculation of y_{n+1}.
- The key to making this adjustment efficiently is knowing how the local truncation error e_{n+1} depends on the step size h.
- For the Euler method, e_{n+1} is proportional to h^2, so to bring the estimated error d_{n+1} down (or up) to the tolerance level ε, we must multiply the original step size h by the factor $\sqrt{\varepsilon / d_{n+1}}$.
- Modern adaptive codes for solving differential equations adjust the step size in very much this way as they proceed, although more accurate formulas than the Euler and improved Euler formulas are used. Consequently, efficiency and accuracy are achieved by using very small steps only where really needed.

Extension of Example 1: Adaptive Euler Method (1 of 3)

- Consider our initial value problem

$$y' = 1 - t + 4y, \;\; y(0) = 1$$

- To prepare for the Euler and improved Euler methods,

$$t_0 = 0, \; y_0 = 1 \;\Rightarrow\; f_0 = 1 - t_0 + 4y_0 = 5$$

- Thus after one step of the Euler method for $h = 0.1$, we have

$$y_1 = y_0 + f_0 h = 1 + (5)(0.1) = 1.5,$$

while for the improved Euler method,

$$f(t_0 + h, y_0 + hf_0) = 1 - 0.1 + 4\big(1 + (0.1)(5)\big) = 6.9$$

and hence

$$y_1 = 1 + (0.5)(0.1)(5 + 6.9) = 1.595$$

- Thus $d_1 = 1.595 - 1.5 = 0.095$.

Extension of Example 1: Adaptive Euler Method (2 of 3)

- Thus $d_1 = 0.095$. If our error tolerance is $\varepsilon = 0.05$, then we adjust the step size $h = 0.1$ downward by the factor $\sqrt{\varepsilon / d_{n+1}}$

$$h = \left(\sqrt{0.05/0.095}\right)(0.1) \cong 0.073$$

- To be safe, we round downward and take $h = 0.07$. Then from the Euler formula, we obtain

$$y_1 = y_0 + f_0 h = 1 + (5)(0.07) = 1.35,$$

while for the improved Euler method,

$$f(t_0 + h, y_0 + hf_0) = 1 - 0.07 + 4\big(1 + (0.07)(5)\big) = 6.33$$

and hence

$$y_1 = 1 + (0.5)(0.07)(5 + 6.33) = 1.39655$$

Extension of Example 1: Adaptive Euler Method (3 of 3)

- We can follow this same procedure at each step, and thereby keep the local truncation error roughly constant throughout the entire numerical process.
- Note $d_1 = 0.04655$ is an estimate of the error in computing y_1, which in our case can be computed using the exact solution

$$\phi(t) = \left(4t - 3 + 19e^{4t}\right)/16$$

- Since $t_0 = 0$ and $h = 0.07$, it follows that $t_1 = 0.07$. We then compute $\phi(0.07) \leq 1.4012$, and hence the actual error is

$$\phi(0.07) - y_1 \cong 1.4012 - 1.35 = 0.0512$$

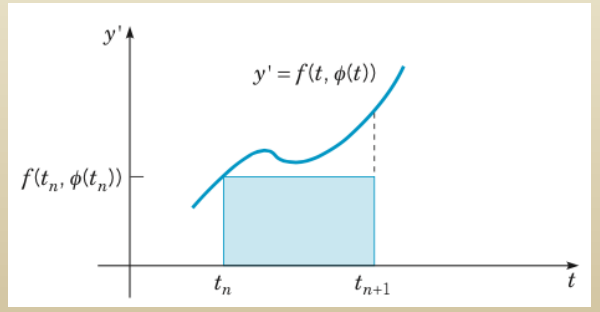
Boyce/DiPrima/Meade 11th ed, Ch 8.3: The Runge-Kutta Method

- Consider the initial value problem $y' = f(t, y)$, $y(t_0) = y_0$, with solution $\phi(t)$.
- We have seen that the local truncation errors for the Euler, backward Euler, and improved Euler methods are proportional to h^2, h^2, and h^3, respectively.
- In this section, we examine the **Runge-Kutta** method, whose local truncation error is proportional to h^5.
- As with the improved Euler approach, this method better approximates the integral introduced in Ch 8.1, where we had

$$\int_{t_n}^{t_{n+1}} \phi'(t)dt = \int_{t_n}^{t_{n+1}} f\big(t, \phi(t)\big)dt$$

$$\phi(t_{n+1}) \cong \phi(t_n) + f\big(t_n, \phi(t_n)\big)\big(t_{n+1} - t_n\big)$$

$$y_{n+1} = y_n + f_n h$$



Runge-Kutta Method

- The Runge-Kutta formula approximates the integrand $f(t_n, \phi(t_n))$ with a weighted average of its values at the two endpoints and at the midpoint. It is given by

$$y_{n+1} = y_n + \frac{h}{6}\left(k_{n1} + 2k_{n2} + 2k_{n3} + k_{n4}\right)$$

where

$$k_{n1} = f(t_n, y_n)$$
$$k_{n2} = f(t_n + h/2, y_n + k_{n1}h/2)$$
$$k_{n3} = f(t_n + h/2, y_n + k_{n2}h/2)$$
$$k_{n4} = f(t_n + h, y_n + k_{n3}h)$$

- Global truncation error is bounded by a constant times h^4 for a finite interval, with local truncation error proportional to h^5.

Simpson's Rule

- The Runge-Kutta formula is

$$y_{n+1} = y_n + \frac{h}{6}(k_{n1} + 2k_{n2} + 2k_{n3} + k_{n4})$$

where

$$k_{n1} = f(t_n, y_n)$$
$$k_{n2} = f(t_n + h/2, y_n + k_{n1}h/2)$$
$$k_{n3} = f(t_n + h/2, y_n + k_{n2}h/2)$$
$$k_{n4} = f(t_n + h, y_n + k_{n3}h)$$

- If $f(t, y)$ depends only on t and not on y, then we have

$$y_{n+1} - y_n = \frac{h}{6}[f(t_n) + 4f(t_n + h/2) + f(t_n + h)],$$

which is Simpson's rule for numerical integration.

Programming Outline: Runge-Kutta Method

- Step 1. Define $f(t,y)$
- Step 2. Input initial values t0 and y0
- Step 3. Input step size h and number of steps n
- Step 4. Output t0 and y0
- Step 5. For j from 1 to n do
- Step 6.

$$k1 = f(t, y)$$
$$k2 = f(t + 0.5*h, y + 0.5*h*k1)$$
$$k3 = f(t + 0.5*h, y + 0.5*h*k2)$$
$$k4 = f(t + h, y + h*k3)$$
$$y = y + (h/6)*(k1 + 2*k2 + 2*k3 + k4)$$
$$t = t + h$$

- Step 7. Output t and y
- Step 8. End

Example 1: Runge-Kutta Method (1 of 2)

- Recall our initial value problem

$$y' = 1 - t + 4y, \quad y(0) = 1,$$

- To calculate y_1 in the first step of the Runge-Kutta method for $h = 0.2$, we start with

$$k_{01} = f(0,1) = 5; \quad \frac{1}{2} h k_{01} = 0.5,$$

$$k_{02} = f(0 + 0.1, 1 + 0.5) = 6.9; \quad \frac{1}{2} h k_{02} = 0.69,$$

$$k_{03} = f(0 + 0.1, 1 + 0.69) = 7.66; \quad h k_{03} = 1.532,$$

$$k_{04} = f(0 + 0.2, 1 + 1.532) = 10.928.$$

- Thus

$$y_1 = 1 + \frac{0.2}{6}\left(5 + 2(6.9) + 2(7.66) + 10.928\right) = 2.5016$$

$$y' = 1 - t + 4y, \quad y(0) = 1$$

Example 1: Numerical Results (2 of 2)

- The Runge-Kutta method ($h = 0.05$) and the improved Euler method ($h = 0.025$) both require a total of 160 evaluations of f. However, we see that the Runge-Kutta is far more accurate.
- The Runge-Kutta method ($h = 0.2$) requires 40 evaluations of f and the improved Euler method requires 160 evaluations of f, and yet the accuracy at $t = 2$ is similar.

t	Improved Euler h = 0.025	Runge-Kutta h = 0.2	Runge-Kutta h = 0.1	Runge-Kutta h = 0.05	Exact	Improved Euler Rel Error h = 0.025	Runge - Kutta Rel Error h = 0.2	Runge - Kutta Rel Error h = 0.1	Runge - Kutta Rel Error h = 0.05
0.00	1.0000	1.0000	1.0000	1.0000	1.0000	0.0000	0.0000	0.0000	0.0000
0.10	1.6079		1.6089	1.6090	1.6090	0.0684		0.0062	0.0000
0.20	2.5021	2.5016	2.5050	2.5053	2.5053	0.1277	0.1477	0.0120	0.0000
0.30	3.8223		3.8294	3.8301	3.8301	0.2037		0.0183	0.0000
0.40	5.7797	5.7776	5.7928	5.7941	5.7942	0.2503	0.2865	0.0242	0.0017
0.50	8.6849		8.7093	8.7118	8.7120	0.3111		0.0310	0.0023
1.00	64.4979	64.4416	64.8581	64.8949	64.8978	0.6162	0.7030	0.0612	0.0045
1.50	474.8340		478.8193	479.2267	479.2592	0.9233		0.0918	0.0068
2.00	3496.6702	3490.5574	3535.8667	3539.8804	3540.2001	1.2296	1.4023	0.1224	0.0090

Adaptive Runge-Kutta Methods

- The Runge-Kutta method with a fixed step size can suffer from widely varying local truncation errors.
- That is, a step size small enough to achieve satisfactory accuracy in some parts of the interval of interest may be smaller than necessary in other parts of the interval.
- Adaptive Runge-Kutta methods have been developed, resulting in a substantial gain in efficiency.
- Adaptive Runge-Kutta methods are a very powerful and efficient means of approximating numerically the solutions of a large class of initial value problems, and are widely available in commercial software packages.

Boyce/DiPrima/Meade 11th ed, Ch 8.4: Multistep Methods

- Consider the initial value problem $y' = f(t, y)$, $y(t_0) = y_0$, with solution $\phi(t)$.
- So far we have studied numerical methods in which data at the point t_n is used to approximate $\phi(t_{n+1})$. Such methods are called **one-step methods**.
- **Multistep methods** use previously obtained approximations of $\phi(t)$ to find the next approximation of $\phi(t)$. That is, the approximations $y_1, \ldots, y_n$ at $t_1, \ldots, t_n$, respectively, may be used to find y_{n+1} at t_{n+1}.
- In this section we discuss two types of multistep methods: Adams methods and backward differentiation formulas.
- For simplicity, we will assume the step size h is constant.

Adams Methods

- Recall that

$$\phi(t_{n+1}) - \phi(t_n) = \int_{t_n}^{t_{n+1}} \phi'(t)dt$$

- The basic idea of an Adams method is to approximate $\phi'(t)$ in the above integral by a polynomial $P_k(t)$ of degree k.
- The coefficients of $P_k(t)$ are determined by using the $k+1$ previously calculated data points.
- For example, for $P_1(t) = At + B$, we use (t_{n-1}, y_{n-1}) and (t_n, y_n), with $P_1(t_{n-1}) = f(t_{n-1}, y_{n-1}) = f_{n-1}$ and $P_1(t_n) = f(t_n, y_n) = f_n$.
- Then

$$\left.\begin{aligned} At_{n-1} + B &= f_{n-1} \\ At_n + B &= f_n \end{aligned}\right\} \Rightarrow A = \frac{1}{h}(f_n - f_{n-1}),\ B = \frac{1}{h}(f_{n-1}t_n - f_n t_{n-1})$$

Second Order Adams-Bashforth Formula

- From the discussion on the previous slide, it follows that

$$\phi(t_{n+1}) - \phi(t_n) = \int_{t_n}^{t_{n+1}} \phi'(t)dt$$

  evaluates to

$$\phi(t_{n+1}) - \phi(t_n) = (A/2)\left(t_{n+1}^2 - t_n^2\right) + B\left(t_{n+1} - t_n\right)$$

- After simplifying, we obtain

$$y_{n+1} = y_n + (3/2)hf_n - (1/2)hf_{n-1}$$

- This equation is the **second order Adams-Bashforth** formula. It is an explicit formula for y_{n+1} in terms of y_n and y_{n-1}, and has local truncation error proportional to h^3.
- We note that when a constant polynomial $P_0(t) = A$ is used, the first order Adams-Bashforth formula is just Euler's formula

Fourth Order Adams-Bashforth Formula

- More accurate Adams formulas can be obtained by using a higher degree polynomial $P_k(t)$ and more data points.
- For example, the coefficients of a 3rd degree polynomial $P_3(t)$ are found using (t_n, y_n), (t_{n-1}, y_{n-1}), (t_{n-2}, y_{n-2}), (t_{n-3}, y_{n-3}).
- As before, $P_3(t)$ then replaces $\phi'(t)$ in the integral equation

$$\phi(t_{n+1}) - \phi(t_n) = \int_{t_n}^{t_{n+1}} \phi'(t)dt$$

to obtain the fourth order Adams-Bashforth formula

$$y_{n+1} = y_n + (h/24)\left(55f_n - 59f_{n-1} + 37f_{n-2} - 9f_{n-3}\right)$$

- The local truncation error of this method is proportional to h^5.

Second Order Adams-Moulton Formula

- A variation on the Adams-Bashforth formulas gives another set of formulas called the Adams-Moulton formulas.
- We begin with the second order case, and use a first degree polynomial $Q_1(t) = \alpha t_n + \beta$ to approximate $\phi'(t)$.
- To determine α and β, we now use (t_n, y_n) and (t_{n+1}, y_{n+1}):

$$\left.\begin{aligned} \alpha t_n + \beta &= f_n \\ \alpha t_{n+1} + \beta &= f_{n+1} \end{aligned}\right\} \Rightarrow \alpha = \frac{1}{h}(f_{n+1} - f_n),\ \beta = \frac{1}{h}(f_n t_{n+1} - f_{n+1} t_n)$$

- As before, $Q_1(t)$ replaces $\phi'(t)$ in the integral equation to obtain the second order Adams-Moulton formula

$$y_{n+1} = y_n + (h/2)(f_n + f_{n+1}(t_{n+1}, y_{n+1})) = y_n + (h/2)(f_n + f_{n+1})$$

- Note that this equation implicitly defines y_{n+1}. The local truncation error of this method is proportional to h^3.

Fourth Order Adams-Moulton Formula

- The first order Adams-Moulton formula is just the backwards Euler formula.
- More accurate higher order formulas can be obtained using a polynomial of higher degree.
- For example, the fourth order Adams-Moulton formula is

$$y_{n+1} = y_n + \frac{h}{24}\left(9f_{n+1} + 19f_n - 5f_{n-1} + f_{n-2}\right)$$

- The local truncation error of this method is proportional to h^5.

Comparison of Methods

- The Adams-Bashforth and Adams-Moulton formulas both have local truncation errors proportional to the same power of h, but moderate order Adams-Moulton formulas are more accurate.
- For example, for the fourth order methods, the proportionality constant on h^5 for the Adams-Moulton formula is less than 1/10 that of the Adams-Bashforth formula.
- The Adams-Bashforth formula explicitly defines y_{n+1} and thus is faster than the more accurate Adams-Moulton formula, which implicitly defines y_{n+1}.
- Which method to use depends on whether, by using the more accurate method, the step size can be increased to reduce the number of computations required.
- A **predictor-corrector method** combines both approaches.

Predictor-Corrector Method

- Consider the fourth order Adams-Bashforth and Adams-Moulton formulas, respectively:

$$y_{n+1} = y_n + (h/24)\left(55f_n - 59f_{n-1} + 37f_{n-2} - 9f_{n-3}\right)$$

$$y_{n+1} = y_n + (h/24)\left(9f_{n+1} + 19f_n - 5f_{n-1} + f_{n-2}\right)$$

- Once $y_{n-3}, y_{n-2}, y_{n-1}, y_n$ are known, we compute $f_{n-3}, f_{n-2}, f_{n-1}, f_n$ and use Adams-Bashforth formula (predictor) to obtain y_{n+1}.
- We then compute f_{n+1}, and use the Adams-Bashforth formula (corrector) to obtain an improved value of y_{n+1}.
- We can continue to use corrector formula if the change in y_{n+1} is too large. However, if it is necessary to use the corrector formula more than once or perhaps twice, the step size h is likely too large and should be reduced.

Starting Values for Multistep Methods

- In order to use any of the multistep methods, it is necessary to first to calculate a few y_k by some other method.
- For example, the fourth order Adams-Moulton method requires values for y_1 and y_2, while the fourth order Adams-Bashforth method also requires a value for y_3.
- One way to proceed is to use a one-step method of comparable order to calculate the necessary starting values.
- For example, for a fourth order multistep method, use a fourth order Runge-Kutta method to calculate the starting values.
- Another approach is to use a low order method with a very small h to calculate y_1, and then to increase gradually both the order and step size until enough starting values are obtained.

Example 1: Initial Value Problem (1 of 6)

- Recall our initial value problem

$$y' = 1 - t + 4y, \quad y(0) = 1$$

- With a step size of $h = 0.1$, we will use the methods of this section to approximate the solution solution $\phi(t)$ at $t = 0.4$.
- We use the Runge-Kutta method to find y_1, y_2 and y_3. These values are given in Table 8.3.1. The corresponding values for $f(t, y) = 1 - t - 4y$ can then be computed, with results below.

$$\begin{aligned} y_0 &= 1, & f_0 &= 5, \\ y_1 &= 1.6089333, & f_1 &= 7.3357332, \\ y_2 &= 2.5050062, & f_2 &= 10.820025, \\ y_3 &= 3.8294145, & f_3 &= 16.017658. \end{aligned}$$

Example 1: Adams-Bashforth Method (2 of 6)

- The values of f_k from the previous page are

  $$f_0 = 5,\ f_1 = 7.3357332,\ f_2 = 10.820025,\ f_3 = 16.017658$$

- Using the fourth order Adams-Bashforth formula, we have

  $$y_4 = y_3 + (0.1/24)\left(55f_3 - 59f_2 + 37f_1 - 9f_1\right) = 5.7836305$$

- The exact value of $\phi(0.4)$ can be found using the solution,

  $$\phi(t) = \left(4t - 3 + 19e^{4t}\right)/16 \ \Rightarrow\ \phi(0.4) \cong 5.7942260$$

  and hence the error in this case is -0.0105955, with a relative error of 0.183%.

Example 1: Adams-Moulton Method (3 of 6)

- Recall the fourth order Adams-Moulton formula:

$$y_{n+1} = y_n + (h/24)\left(9f_{n+1} + 19f_n - 5f_{n-1} + f_{n-2}\right)$$

- Using the previously calculated values of f_k

$$f_0 = 5,\ f_1 = 7.3357332,\ f_2 = 10.820025,\ f_3 = 16.017658,$$

the fourth order Adams-Moulton formula reduces to

$$y_4 = 4.9251275 + 0.15y_4$$

- Solving this linear implicit equation for y_4, we obtain

$$y_4 = 5.7942676$$

- Recall that the exact value to seven decimal places is

$$\phi(0.4) = 5.7942260$$

- The error in this case is therefore 0.0000416, with a relative error of 0.0072%.

Example 1: Predictor-Corrector Method (4 of 6)

- Recall our fourth order equations:

$$y_4 = y_3 + (0.1/24)\left(55f_3 - 59f_2 + 37f_1 - 9f_0\right) \quad \text{(predictor)}$$

$$y_4 = y_3 + (0.1/24)\left(9f_4 + 19f_3 - 5f_2 + f_1\right) \quad \text{(corrector)}$$

- Using the first equation, we predict $y_4 = 5.7836305$, as before.
- Then $f_4 = 1 - 0.4 + 4(5.7836305) = 23.734522$.
- Using the second equation as a corrector, we obtain

$$y_4 = 5.7926721$$

- The error is –0.0015539, with a relative error of 0.02682%.
- The error for the corrected y_4 has been reduced by a factor of approximately 7 when compared to the error of predicted y_4.

Example 1: Summary of Results (5 of 6)

- The Adams-Bashforth method is the simplest and fastest of these methods, but is also the least accurate.
- Using the Adams-Moulton formula as a corrector increases the amount of calculation required, but still is explicit in y_4.
- For this problem, the error in corrected value of y_4 is reduced by a factor of 7 when compared to the error in predicted y_4.
- The Adams-Moulton method yields the best result, with an error that is about 1/40 the error of predictor-corrector result.
- The Adams-Moulton method is implicit in y_4, and hence an equation must be solved at each step. For this problem, the equation was linear with y_4 easily found. In other problems, this part of the procedure may be more time consuming.

Example 1: Comparison with Runge-Kutta Method (6 of 6)

- The Runge-Kutta method for $h = 0.1$ gives $y_4 = 5.7927853$, as seen in Table 8.3.1.
- The corresponding error is –0.0014407, with a relative error of 0.02686%.
- Thus the Runge-Kutta method is comparable in accuracy to the predictor-corrector method for this example.

Backward Differentiation Formulas

- Another type of multistep method uses a polynomial $P_k(t)$ to approximate the solution $\phi(t)$ instead of its derivative $\phi'(t)$.
- We then differentiate $P_k(t)$ and set $P_k'(t_{n+1}) = f(t_{n+1}, y_{n+1})$ to obtain an implicit formula for y_{n+1}.
- These are called **backward differentiation formulas**.
- The simplest case uses a first degree $P_1(t) = At + B$.
- The values of A and B are chosen to match the computed solution values y_n and y_{n+1}:

$$At_n + B = y_n$$

$$At_{n+1} + B = y_{n+1}$$

- Also, we set $P_k'(t_{n+1}) = A = f(t_{n+1}, y_{n+1})$, as mentioned above.

Backward Differentiation: First Order Formula

- We thus have $A = f(t_{n+1}, y_{n+1})$ and

$$\left.\begin{aligned} At_n + B &= y_n \\ At_{n+1} + B &= y_{n+1} \end{aligned}\right\} \Rightarrow \quad A = \frac{1}{h}(y_{n+1} - y_n)$$

- From these two equations for A, it follows that

$$y_{n+1} = y_n + hf(t_{n+1}, y_{n+1})$$

- Note that this is the backward Euler formula.

Higher Order Formulas

- By using higher order polynomials and correspondingly more data points, backward differentiation formulas of any order can be obtained.
- The second order formula is

$$y_{n+1} = \frac{1}{3}\left(4y_n - y_{n-1} + 2hf\left(t_{n+1}, y_{n+1}\right)\right)$$

- The local truncation error of this method is proportional to h^3.
- The fourth order formula is

$$y_{n+1} = \frac{1}{25}\left(48y_n - 36y_{n-1} + 16y_{n-2} - 3y_{n-3} + 12hf\left(t_{n+1}, y_{n+1}\right)\right)$$

- The local truncation error of this method is proportional to h^5.

Example 2: Fourth Order Backward Differentiation Method (1 of 2)

- Recall our initial value problem

$$y' = 1 - t + 4y, \quad y(0) = 1$$

- Use the fourth order backward differentiation formula with $h = 0.1$ to approximate the solution solution $\phi(t)$ at $t = 0.4$.
- From Example 1, we have the following data:

$$y_0 = 1, \quad y_1 = 1.6089333, \quad y_2 = 2.5050062, \quad y_3 = 3.8294145$$

- Thus

$$\begin{aligned} y_4 &= \left[48y_3 - 36y_2 + 16y_1 - 3y_0 + 12(0.1) f(t_4, y_4)\right]/25 \\ &= 4.6837842 + 0.192y_4 \end{aligned}$$

and hence

$$y_4 = 5.7967626$$

Example 2: Results (2 of 2)

- Our fourth order backward differentiation approximation is

$$y_4 = 5.7967626$$

- Recall that the exact value to seven decimal places is

$$\phi(0.4) = 5.7942260$$

- The error in this case is therefore 0.0025366, with a relative error of 0.0438%.
- These results are somewhat better than the Adams-Bashforth method, but not as good as using the predictor-corrector method, and not nearly as good as the result using the Adams-Moulton method.

Comparison of One-Step and Multistep Methods (1 of 2)

- In comparing methods, we first consider the number of evaluations of f at each step:
  - The fourth order Runge-Kutta method requires four calculations of f.
  - The fourth order Adams-Bashforth method, once past the starting values, requires only one evaluation of f.
  - The predictor-corrector method requires two evaluations of f.
- Thus, for a given step size h, the latter two methods may be faster than Runge-Kutta. However, if Runge-Kutta is more accurate and can therefore use fewer steps, then the difference in speed will be reduced and perhaps eliminated.
- The Adams-Moulton and backward differentiation formulas also require that the difficulty in solving the implicit equation at each step be taken into account.

Comparison of One-Step and Multistep Methods (2 of 2)

- All multistep methods have the possible disadvantage that errors in earlier steps can feed back into later calculations.
- On the other hand, the underlying polynomial approximations in multistep methods make it easy to approximate the solution at points between the mesh points, if desirable.
- Multistep methods have become popular largely because it is relatively easy to estimate the error at each step and adjust the order or the step size to control it.

Boyce/DiPrima/Meade 11th ed, Ch 8.5: Systems of First Order Equations

- Recall from Section 7.1 that a higher order equation can always be reduced to a system of first order equations.
- In this section, we examine how the numerical methods of this chapter can be applied to systems of first order equations.
- For simplicity, we consider the system below.

$$x' = f(t,x,y),\ y' = g(t,x,y);\ \ x(t_0) = x_0,\ y(t_0) = y_0$$

- The functions f and g are assumed to satisfy the conditions of Theorem 7.1.1, so that the initial value problem above has a unique solution in some interval of the t-axis containing t_0.
- We seek approximate values x_n and y_n of the solution $x = \phi(t)$, $y = \psi(t)$, at the points $t_n = t_0 + nh$, for $n \geq 1$.

Vector Notation

- In vector notation, the initial value problem

$$x' = f(t,x,y),\ y' = g(t,x,y);\ \ x(t_0) = x_0,\ y(t_0) = y_0$$

can be written as

$$\mathbf{x}' = \mathbf{f}(t,\mathbf{x}),\quad \mathbf{x}(t_0) = \mathbf{x}_0$$

where

$$\mathbf{x} = \begin{pmatrix} x(t) \\ y(t) \end{pmatrix},\ \mathbf{f} = \begin{pmatrix} f(t,x,y) \\ g(t,x,y) \end{pmatrix},\quad \mathbf{x}_0 = \begin{pmatrix} x_0 \\ y_0 \end{pmatrix}$$

- The numerical methods of the previous sections can be readily generalized to handle systems of two or more equations.
- To accomplish this, we simply replace x and f in the numerical formulas with **x** and **f**, as illustrated in the following slides.

Euler Method

- Recall the Euler method for a uniform step size h:

$$y_{n+1} = y_n + hf(t_n, y_n)$$

- In vector notation, this can be written as

$$\mathbf{x}_{n+1} = \mathbf{x}_n + h\mathbf{f}_n$$

or

$$\begin{pmatrix} x_{n+1} \\ y_{n+1} \end{pmatrix} = \begin{pmatrix} x_n \\ y_n \end{pmatrix} + h\begin{pmatrix} f(t_n, x_n, y_n) \\ g(t_n, x_n, y_n) \end{pmatrix}$$

- The initial conditions are used to determine $\mathbf{f}_0$, the tangent vector to the graph of the solution $\mathbf{x} = \Phi(t)$ in the xy-plane.
- We move in the direction of this tangent vector $\mathbf{f}_0$ for a time step h in order to find $\mathbf{x}_1$, then find a new tangent vector $\mathbf{f}_1$, move along it for a time step h in order to find $\mathbf{x}_2$, and so on.

Runge-Kutta Method

- In a similar way, the Runge-Kutta method can be extended to a system of equations. The vector formula is given by

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \frac{h}{6}\left(\mathbf{k}_{n1} + 2\mathbf{k}_{n2} + 2\mathbf{k}_{n3} + \mathbf{k}_{n4}\right)$$

where

$$\mathbf{k}_{n1} = \mathbf{f}\left(t_n, \mathbf{x}_n\right), \qquad \mathbf{k}_{n2} = \mathbf{f}\left(t_n + h/2, \mathbf{x}_n + (h/2)\mathbf{k}_{n1}\right),$$
$$\mathbf{k}_{n3} = \mathbf{f}\left(t_n + h/2, \mathbf{x}_n + (h/2)\mathbf{k}_{n2}\right), \qquad \mathbf{k}_{n4} = \mathbf{f}\left(t_n + h, \mathbf{x}_n + h\mathbf{k}_{n3}\right).$$

- To help better understand the notation here, observe that

$$\mathbf{x}_n = \begin{pmatrix} x_n \\ y_n \end{pmatrix}, \quad \mathbf{z} = \begin{pmatrix} z_1 \\ z_2 \end{pmatrix}, \quad \mathbf{f} = \begin{pmatrix} f(t,x,y) \\ g(t,x,y) \end{pmatrix}$$

$$\Rightarrow \mathbf{f}\left(t_n + r, \mathbf{x}_n + r\,\mathbf{z}\right) = \begin{pmatrix} f(t_n + r, x_n + r\,z_1, y_n + r\,z_2) \\ g(t_n + r, x_n + r\,z_1, y_n + r\,z_2) \end{pmatrix}$$

Example 1: Exact Solution (1 of 4)

- Consider the initial value problem

$$x' = x - 4y, \quad y' = -x + y,$$
$$x(0) = 1, \qquad y(0) = 0.$$

- We will use Euler's method with $h = 0.1$ and the Runge-Kutta method with $h = 0.2$ to approximate the solution at $t = 0.2$, and then compare results with the exact solution:

$$x = \phi(t) = \frac{e^{-t} + e^{3t}}{2}, \quad y = \psi(t) = \frac{e^{-t} - e^{3t}}{4}$$

- For this problem, note that $f_n = x_n - 4y_n$ and $g_n = -x_n + y_n$, and that f and g are independent of t, with

$$\mathbf{f} = \begin{pmatrix} f(t,x,y) \\ g(t,x,y) \end{pmatrix} = \begin{pmatrix} f(x,y) \\ g(x,y) \end{pmatrix} = \begin{pmatrix} x - 4y \\ -x + y \end{pmatrix}$$

Example 1: Euler Method (2 of 4)

- We have $x_0 = 1$, $y_0 = 0$, and

$$f_n = x_n - 4y_n, \qquad g_n = -x_n + y_n,$$
$$f_0 = 1 - 4(0) = 1, \quad g_0 = -1 - 4(0) = -1.$$

- Using the Euler formulas with $h = 0.1$,

$$x_1 = x_0 + (0.1)f_0 = 1.1, \qquad y_1 = y_0 + (0.1)g_0 = -0.1,$$
$$x_2 = x_1 + (0.1)f_1 = 1.25, \quad y_2 = y_1 + (0.1)g_1 = -0.22.$$

- The values of the exact solution, correct to eight digits, are

$$\phi_1(0.2) = 1.3204248, \quad \psi_1(0.2) = -0.25084701$$

- Thus the Euler method approximation errors are 0.0704 and 0.0308, respectively, with corresponding relative errors of about 5.3% and 12.3%.

Example 1: Runge-Kutta Method (3 of 4)

- Using the Runge-Kutta formulas for $\mathbf{k}_{ni}$,

$$\mathbf{k}_{n1} = \mathbf{f}(t_n, \mathbf{x}_n), \qquad \mathbf{k}_{n2} = \mathbf{f}(t_n + h/2, \mathbf{x}_n + (h/2)\mathbf{k}_{n1}),$$
$$\mathbf{k}_{n3} = \mathbf{f}(t_n + h/2, \mathbf{x}_n + (h/2)\mathbf{k}_{n2}), \quad \mathbf{k}_{n4} = \mathbf{f}(t_n + h, \mathbf{x}_n + h\mathbf{k}_{n3}),$$

and recalling that $x_0 = 1$, $y_0 = 0$, we obtain, for $h = 0.2$,

$$\mathbf{k}_{01} = \begin{pmatrix} f(1,0) \\ g(1,0) \end{pmatrix} = \begin{pmatrix} 1 \\ -1 \end{pmatrix}, \quad \mathbf{k}_{02} = \begin{pmatrix} f(1.1,-0.1) \\ g(1.1,-0.1) \end{pmatrix} = \begin{pmatrix} 1.5 \\ -1.2 \end{pmatrix},$$

$$\mathbf{k}_{03} = \begin{pmatrix} f(1.15,-0.12) \\ g(1.15,-0.12) \end{pmatrix} = \begin{pmatrix} 1.63 \\ -1.27 \end{pmatrix}, \quad \mathbf{k}_{04} = \begin{pmatrix} f(1.326,-0.254) \\ g(1.326,-0.254) \end{pmatrix} = \begin{pmatrix} 2.342 \\ -1.580 \end{pmatrix},$$

$$\mathbf{x}_1 = \mathbf{x}_0 + \frac{0.2}{6}(\mathbf{k}_{01} + 2\mathbf{k}_{02} + 2\mathbf{k}_{03} + \mathbf{k}_{04}) = \begin{pmatrix} 1.320067 \\ -0.25066667 \end{pmatrix}$$

- The errors for x_1 and y_1 are 0.000358 and 0.000180, respectively, with relative errors of about 0.0271% and 0.072%.

Example 1: Summary (4 of 4)

- This example again illustrates the great gains in accuracy that are possible by using a more accurate approximation method, such as the Runge-Kutta method.
- In the calculations for this example, the Runge-Kutta method requires only twice as many function evaluations as the Euler method, but the error in the Runge-Kutta method is about 200 times less than in the Euler method.

Boyce/DiPrima/Meade 11th ed, Ch 8.6: More on Errors; Stability

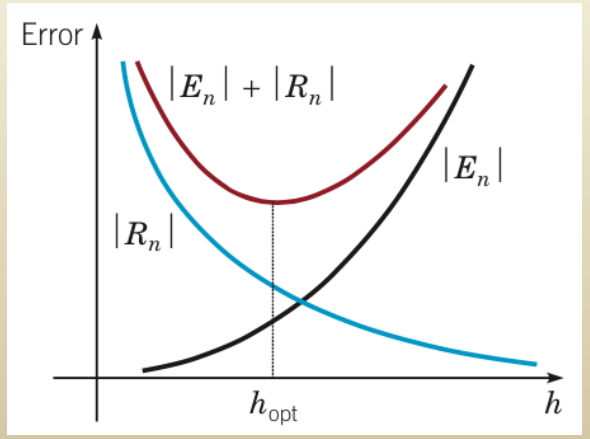
- In Section 8.1 we discussed some ideas related to the errors that can occur in a numerical approximation to the solution of the initial value problem $y' = f(t, y), y(t_0) = y_0$.
- In this section we continue that discussion and also point out some other difficulties that can arise.
- Some of the points are difficult to treat in detail, so we will illustrate them by means of examples.

Truncation and Round-off Errors

- Recall that for the Euler method, the local truncation error is proportional to h^2, and that for a finite interval the global truncation error is at most a constant times h.
- In general, for a method of order p, the local truncation error is proportional to h^{p+1}, and the global truncation error on a finite interval is bounded by a constant times h^p.
- To achieve a high accuracy we normally use a numerical procedure for which p is fairly large, perhaps 4 or higher.
- As p increases, the formula used in computing y_{n+1} normally becomes more complicated, and more calculations are required at each step. This is usually not a serious problem unless $f(t,y)$ is complicated, or if the calculation must be repeated many times.

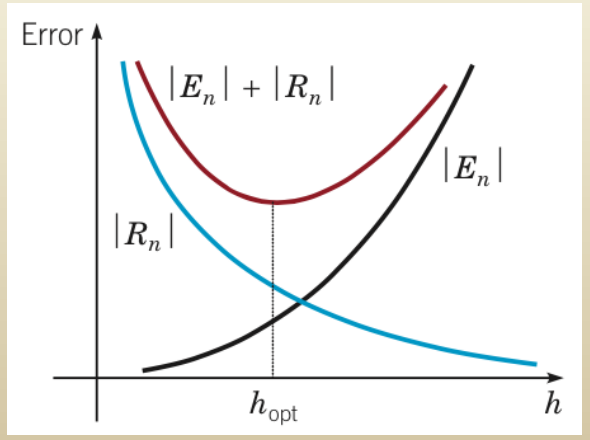
Truncation and Round-off Errors

- If the step size h is decreased, the global truncation error is decreased by the same factor raised to the power p.
- However, if h is very small, then many steps will be required to cover a fixed interval, and the global round-off error may be larger than the global truncation error.
- This situation is shown schematically below, where R_n is the round-off error, and E_n the truncation error, at step n.
- See next slide for more discussion.



Truncation and Round-off Errors

- We assume R_n is proportional to the number of computations, and thus is inversely proportional to h.
- We also assume E_n is proportional to a positive power of h.
- From Section 8.1, the total error is bounded by $|R_n| + |E_n|$. Thus we want to choose h so as to minimize this quantity.
- This optimum value of h occurs when the rate of increase of E_n (as h increases) is balanced by the rate of decrease of R_n.



Example 1: Euler Method Results (1 of 4)

- Consider the initial value problem

$$y' = 1 - t + 4y, \quad y(0) = 1; \quad 0 \le t \le 1$$

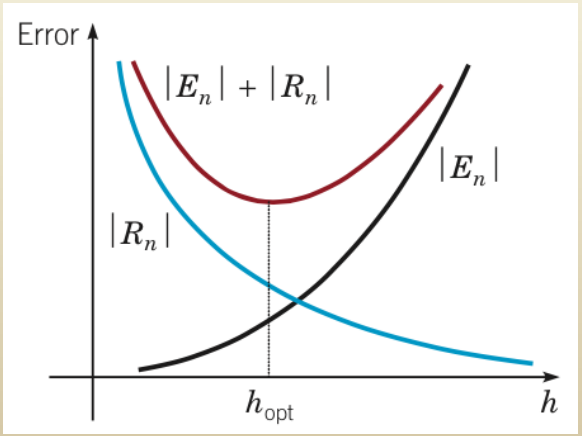
- In the table below, the values $y_{N/2}$ and y_N are Euler method approximations to $\phi(0.5) = 8.712$, $\phi(1) = 64.90$, respectively, for different step sizes h.
- The number of steps N required to traverse [0, 1] are given, as are the errors between approximations and exact values.

h	N	y[N/2]	Error	y[N]	Error
0.010000	100	8.39	–0.322	60.12	–4.78
0.005000	200	8.551	–0.161	62.51	–2.39
0.002000	500	8.633	–0.079	63.75	–1.15
0.001000	1000	8.656	–0.056	63.94	–0.96
0.000800	1250	8.636	–0.076	63.78	–1.12
0.000625	1600	8.616	–0.096	64.35	–0.55
0.000500	2000	8.772	0.060	64.00	–0.9
0.000400	2500	8.507	0.205	63.40	–1.5
0.000250	4000	8.231	0.481	56.77	–8.13

Example 1: Error and Step Size (2 of 4)

- For relatively large step sizes, round-off error R_n is much less than global truncation error E_n. Thus total error $\leq E_n$, which for Euler's method is bounded by a constant times h.
- Thus as step size is reduced, error is reduced proportionately. The first three lines of the table show this behavior.
- For $h = 0.001$, error is reduced, but less than proportionally, and hence round-off error is becoming important.

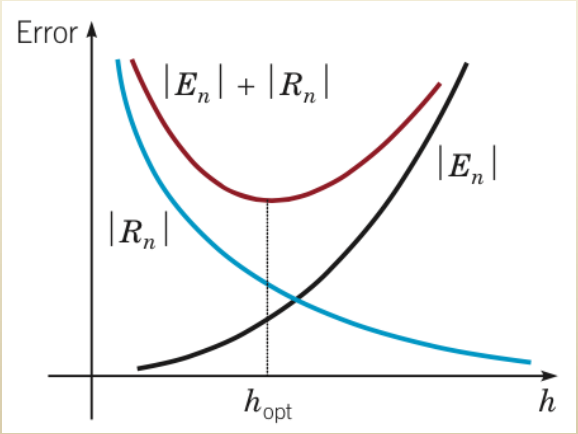
h	N	y[N/2]	Error	y[N]	Error
0.010000	100	8.39	–0.322	60.12	–4.78
0.005000	200	8.551	–0.161	62.51	–2.39
0.002000	500	8.633	–0.079	63.75	–1.15
0.001000	1000	8.656	–0.056	63.94	–0.96
0.000800	1250	8.636	–0.076	63.78	–1.12
0.000625	1600	8.616	–0.096	64.35	–0.55
0.000500	2000	8.772	0.060	64.00	–0.9
0.000400	2500	8.507	0.205	63.40	–1.5
0.000250	4000	8.231	0.481	56.77	–8.13



Example 1: Optimal Step Size (3 of 4)

- As h is reduced further, the error begins to fluctuate.
- For values of $h < 0.0005$ the error increases, and hence the round-off error is now the dominant part of the error.
- For this problem it is best to use an N between 1000 and 2000. In the table, the best result at $t = 0.5$ occurs for $N = 1000$, while at $t = 1$ the best result is for $N = 1600$.

h	N	y[N/2]	Error	y[N]	Error
0.010000	100	8.39	–0.322	60.12	–4.78
0.005000	200	8.551	–0.161	62.51	–2.39
0.002000	500	8.633	–0.079	63.75	–1.15
0.001000	1000	8.656	–0.056	63.94	–0.96
0.000800	1250	8.636	–0.076	63.78	–1.12
0.000625	1600	8.616	–0.096	64.35	–0.55
0.000500	2000	8.772	0.060	64.00	–0.9
0.000400	2500	8.507	0.205	63.40	–1.5
0.000250	4000	8.231	0.481	56.77	–8.13



Example 1: Truncation and Round-Off Error Discussion (4 of 4)

- Optimal ranges for h and N depend on differential equation, numerical method, and number of digits retained.
- It is generally true that if too many steps are required, then eventually round-off error is likely to accumulate to the point where it seriously degrades accuracy of the procedure.
- For many problems this is not a concern, as the fourth order methods discussed in Sections 8.3 and 8.4 will produce good results with a number of steps far less than the level at which round-off error becomes important.
- For some problems round-off error becomes vitally important, and the choice of method may become crucial, and adaptive methods advantageous.

Example (Vertical Asymptote): Euler's Method (1 of 5)

- Consider the initial value problem

$$y' = t^2 + y^2, \quad y(0) = 1; \quad 0 \le t \le 1$$

- Since this differential equation is nonlinear, the existence and uniqueness theorem (Theorem 2.4.2) guarantees only that there is a solution $\phi(t)$ in *some* interval about $t = 0$.
- Using the Euler method, we obtain the approximate values of the solution at $t = 1$ shown in the table below.
- The large differences among the computed values suggest we use a more accurate method, such as the Runge-Kutta method.

h	t = 1
0.10	7.189548
0.05	12.320930
0.01	90.755510

Example (Vertical Asymptote): Runge-Kutta Method (2 of 5)

- Using the Runge-Kutta method, we obtain the approximate solution values at $t = 0.90$ and $t = 1$ shown in the table below.
- It may be reasonable to conclude that $\phi(0.9) \cong 14.305$, but it is not clear what is happening between $t = 0.90$ and $t = 1$.
- To help clarify this, we examine analytical approximations to the solution $\phi(t)$. This will illustrate how information can be obtained by a combination of analytical and numerical work.

h	t = 0.90	t = 1
0.100	14.02182	735.09910
0.050	14.27117	1.75863 x 10^5
0.010	14.30478	2.0913 x 10^2893
0.001	14.30486	

Example (Vertical Asymptote): Analytical Bounds (3 of 5)

- Recall our initial value problem

$$y' = t^2 + y^2, \;\; y(0) = 1; \;\; 0 \le t \le 1$$

and its solution $\phi(t)$. Note that

$$y^2 \le t^2 + y^2 \le 1 + y^2, \;\; \text{for } 0 \le t \le 1.$$

- It follows that the solution $\phi_1(t)$ of

$$y' = 1 + y^2, \;\; y(0) = 1$$

is an upper bound for $\phi(t)$, and the solution $\phi_2(t)$ of

$$y' = y^2, \;\; y(0) = 1$$

is an lower bound for $\phi(t)$. That is,

$$\phi_2(t) \le \phi(t) \le \phi_1(t),$$

as long as the solutions exist.

Example (Vertical Asymptote): Analytical Results (4 of 5)

- Using separation of variables, we can solve for $\phi_1(t)$ and $\phi_2(t)$:

$$y' = 1 + y^2,\ y(0) = 1 \ \Rightarrow\ \varphi_1(t) = \tan\left(t + \frac{\pi}{4}\right)$$

$$y' = y^2,\quad y(0) = 1 \ \Rightarrow\ \varphi_2(t) = (1-t)^{-1}$$

- Note that

$$\lim_{t \to \pi/4} \phi_1(t) = \infty,\ \lim_{t \to 1} \phi_2(t) = \infty$$

- Recall from previous slide that $\phi_2(t) \le \phi(t) \le \phi_1(t)$, as long as the solutions exist. It follows that
  (1) $\phi(t)$ exists for at least $0 \le t < \pi/4 \le 0.785$, and at most for $0 \le t < 1$.
  (2) $\phi(t)$ has a vertical asymptote for some t in $\pi/4 \le t \le 1$.
- Our numerical results suggest that we can go beyond $t = \pi/4$, and probably beyond $t = 0.9$.

Example: Vertical Asymptote (5 of 5)

- Assuming that the solution $y = \phi(t)$ of our initial value problem

$$y' = t^2 + y^2, \quad y(0) = 1; \quad 0 \le t \le 1$$

exists at $t = 0.9$, with $\phi(0.9) = 14.305$, we can obtain a more accurate appraisal of what happens for larger t by solving

$$y' = 1 + y^2, \quad y(0.9) = 14.305 \quad \Rightarrow \quad \phi_1(t) = \tan(t + 0.60100)$$

$$y' = y^2, \qquad y(0.9) = 14.305 \quad \Rightarrow \quad \phi_2(t) = (0.96991 - t)^{-1}$$

- Note that

$$\lim_{t \to 0.96980} \phi_1(t) = \infty, \qquad \lim_{t \to 0.96991} \phi_2(t) = \infty,$$

where $0.96980 \cong \pi/2 - 0.60100$.

- We conclude that the vertical asymptote of $\phi(t)$ lies between $t = 0.96980$ and $t = 0.96991$.

Stability (1 of 2)

- Stability refers to the possibility that small errors introduced in a procedure die out as the procedure continues. Instability occurs if small errors tend to increase.
- In Section 2.5 we identified equilibrium solutions as (asymptotically) stable or unstable, depending on whether solutions that were initially near the equilibrium solution tended to approach it or depart from it as t increased.
- More generally, the solution of an initial value problem is asymptotically stable if initially nearby solutions tend to approach the solution, and unstable if they depart from it.
- Visually, in an asymptotically stable problem, the graphs of solutions will come together, while in an unstable problem they will separate.

Stability (2 of 2)

- When solving an initial value problem numerically, it will at best mimic the actual solution behavior. We cannot make an unstable problem a stable one by solving it numerically.
- However, a numerical procedure can introduce instabilities that are not part of the original problem. This can cause trouble in approximating the solution.
- Avoidance of such instabilities may require restrictions on the step size h.

Example: Stability & Euler Methods (1 of 5)

- Consider the equation and its general solution,

$$y' = ry, \;\; \phi(t) = Ce^{rt}$$

- Suppose that in solving this equation we have reached the point (t_n, y_n). The exact solution passing through this point is

$$y = y_n e^{r(t-t_n)}$$

- With $f(t,y) = ry$, the numerical approximations obtained from the Euler and backward Euler methods are, respectively,

$$y_{n+1} = y_n + hf(t_n, y_n) = y_n(1+rh)$$

$$y_{n+1} = y_n + hf(t_{n+1}, y_{n+1}) = y_n + rhy_{n+1}$$

- From the backward Euler and geometric series formulas,

$$y_{n+1} = \frac{y_n}{1-rh} = y_n\left[1+(rh)+(rh)^2+\cdots\right]$$

Example (Stability): Order of Error (2 of 5)

- The exact solution at t_{n+1} is

$$y(t_{n+1}) = y_n e^{r(t_{n+1}-t_n)} = y_n e^{rh} = y_n\left[1+(rh)+\frac{1}{2}(rh)^2+\cdots\right]$$

- From the previous slide, the Euler and backward Euler approximations are, respectively,

$$y_{n+1} = y_n(1+rh)$$

$$y_{n+1} = y_n\left[1+(rh)+(rh)^2+\cdots\right]$$

- Thus the errors for Euler and backward Euler approximations are of order h^2, as the theory predicts.

Example (Stability): Error Propagation and Stability of Problem (3 of 5)

- Now suppose that we change y_n to $y_n + \delta$, where we think of δ as the error that has accumulated by the time we reach $t = t_n$.
- The question is then whether this error increases or decreases in going one more step to t_{n+1}.
- From the exact solution, the change in $y(t_{n+1})$ due to the change in y_n is δe^{rh}, as seen below.

$$y(t_{n+1}) = (y_n + \delta)e^{rh} = y_n e^{rh} + \delta e^{rh}$$

- Note that $|\delta e^{rh}| < |\delta|$ if $e^{rh} < 1$, which occurs for $r < 0$.
- This confirms our conclusion from Chapter 2 that the equation

$$y' = ry$$

is asymptotically stable if $r < 0$, and is unstable if $r > 0$.

Example:
Stability of Backward Euler Method (4 of 5)

- For the backward Euler method, the change in y_{n+1} due to the change in y_n is $\delta/(1 - rh)$, as seen below.

$$y_{n+1} = \frac{y_n + \delta}{1 - rh}$$

- Note that $0 < |\delta/(1 - rh)| < |\delta|$ for $r < 0$.
- Thus if the differential equation

$$y' = ry$$

is stable, then so is the backwards Euler method.

Example: Stability of Euler Method (5 of 5)

- For the Euler method, the change in y_{n+1} due to the change in y_n is $\delta(1 + rh)$, as seen below.

$$y_{n+1} = (y_n + \delta)(1 + rh)$$

- Note that $0 < |\ \delta(1 + rh)|\ < |\ \delta|$ for $r < 0$ and $|1 + rh| < 1$.
- From this it follows that h must satisfy $h < 2/|r|$, as follows:

$$|1 + rh| < 1 \Leftrightarrow -1 < 1 + rh < 1 \Leftrightarrow -2 < rh < 0$$

$$\Leftrightarrow 2 > (-r)h > 0 \Leftrightarrow h < 2/|r|, \quad r < 0$$

- Thus Euler's method is not stable unless h is sufficiently small.
- Note: Requiring $h < 2/|r|$ is relatively mild, unless r is large.

Stiff Problems

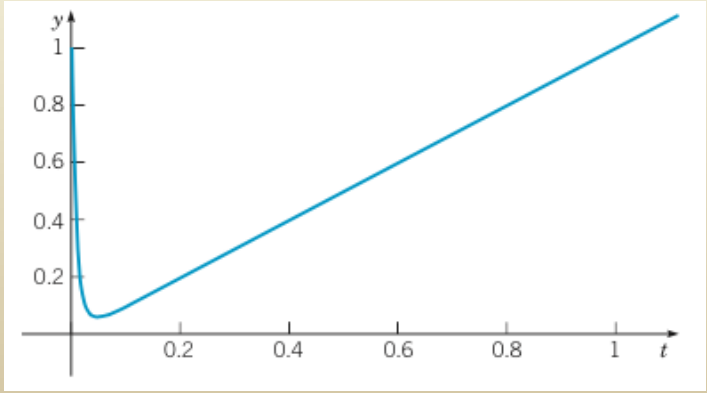
- The previous example illustrates that it may be necessary to restrict h in order to achieve stability in the numerical method, even though the problem itself is stable for all values of h.
- Problems for which a much smaller step size is needed for stability than for accuracy are called **stiff**.
- The backward differentiation formulas of Section 8.4 are popular methods for solving stiff problems, and the backward Euler method is the lowest order example of such methods.

Example 2: A Stiff Problem (1 of 4)

- Consider the initial value problem

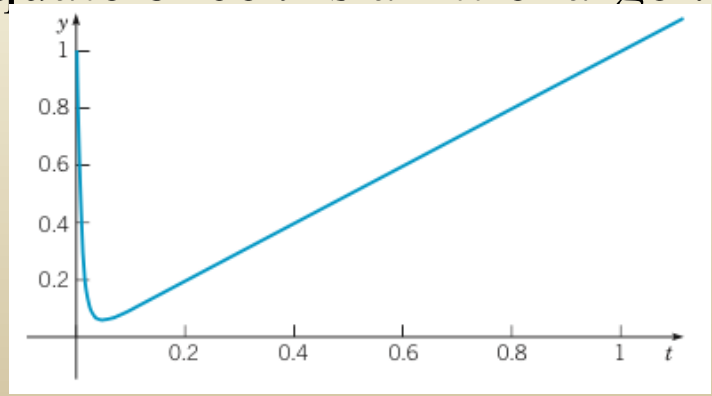
$$y' = -100y + 100t + 1, \quad y(0) = 1; \quad 0 \le t \le 1$$

- Since the equation is linear, with solution $\phi(t) = e^{-100t} + t$.
- The graph of this solution is given below. There is a thin layer (boundary layer) to the right of $t = 0$ in which the exponential term is significant and the values of the solution vary rapidly. Once past this layer, $\phi(t) \cong t$ and the graph is essentially a line.
- The width of the boundary layer is somewhat arbitrary, but it is certainly small. For example, at $t = 0.1$, $e^{-100t} \cong 0.000045$.



Example 2: Error Analysis (2 of 4)

- Numerically, we might expect that a small step size will be needed only in the boundary layer. To make this more precise, consider the following.
- The local truncation errors for the Euler and backward Euler methods are proportional to $\phi''(t)$. Here, $\phi''(t) = 10{,}000e^{-100t}$, which varies from 10,000 at $t = 0$ to nearly zero for $t > 0.2$.
- Thus a very small step size is needed for accuracy near $t = 0$, but a much larger step size is adequate once t is a little larger.

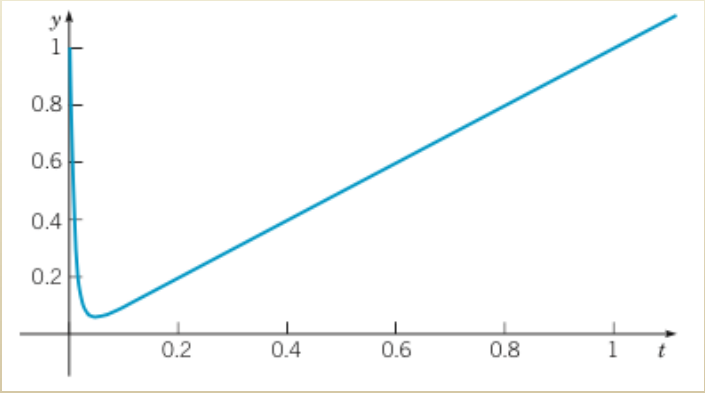


Example 2: Stability Analysis (3 of 4)

- Recall our initial value problem:

$$y' = -100y + 100t + 1, \quad y(0) = 1; \quad 0 \le t \le 1$$

- Comparing this equation with the stability analysis equations, we take $r = -100$ here.
- It follows that for Euler's method, we require $h < 2/|r| = 0.02$.
- There is no corresponding restriction on h for the backward Euler method.



Example 2: Numerical Results (4 of 4)

- The Euler results for $h = 0.025$ are worthless from instability, while results for $h = 1/60 = 0.0166\ldots$ are reasonably accurate for $t \geq 0.2$. Comparable accuracy is obtained for $h = 0.1$ using backward Euler method.
- The Runge-Kutta method is unstable for $h = 1/30 = 0.0333\ldots$ in this problem, but stable for $h = 0.025$.
- Note that a smaller step size is needed for the boundary layer.

		Euler	Euler	Runge-Kutta	Runge-Kutta	Backward Euler
t	Exact	h = 0.025	h = 0.0167	h = .0333	h = 0.025	h = 0.1
0.00	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
0.05	0.056738	2.300000	-0.246296		0.470471	
0.10	0.100045	5.162500	0.187792	10.652700	0.276796	0.190909
0.20	0.200000	25.828900	0.207707	111.559000	0.231257	0.208264
0.40	0.400000	657.241000	0.400059	1.24 x 10^4	0.400977	0.400068
0.60	0.600000	1.68 x 10^4	0.600000	1.38 x 10^6	0.600031	0.600001
0.80	0.800000	4.31 x 10^5	0.800000	1.54 x 10^8	0.800001	0.800000
1.00	1.000000	1.11 x 10^7	1.000000	1.71 x 10^10	1.000000	1.000000

Example 3 (Numerical Dependence): First Set of Solutions (1 of 6)

- Consider the second order equation

$$y'' - 10\pi^2 y = 0, \quad t > 0$$

- Two linearly independent solutions are

$$\phi_1(t) = \cosh\left(\sqrt{10}\pi t\right), \; \phi_2(t) = \sinh\left(\sqrt{10}\pi t\right)$$

where $\phi_1(t)$ and $\phi_2(t)$ satisfy the respective initial conditions

$$\phi_1(0) = 1, \; \phi_1'(0) = 0; \quad \phi_2(0) = 0, \; \phi_2'(0) = \sqrt{10}\pi$$

- Recall that

$$\cosh(t) = \frac{e^t + e^{-t}}{2}, \; \sinh(t) = \frac{e^t - e^{-t}}{2}$$

- It follows that for large t, $\phi_1(t) \cong \phi_2(t)$.

Example 3 (Numerical Dependence): Numerical Dependence (2 of 6)

- Our two linearly independent solutions are

$$\phi_1(t) = \cosh\left(\sqrt{10}\pi t\right),\ \phi_2(t) = \sinh\left(\sqrt{10}\pi t\right)$$

- For large t, $\phi_1(t) \cong \phi_2(t)$, and hence these two solutions will look the same if only a fixed number of digits are retained.
- For example, at $t = 1$ and using 8 significant figures, we have

$$\cosh\left(\sqrt{10}\pi\right) = \sinh\left(\sqrt{10}\pi\right) = 10{,}315.894$$

- If the calculations are performed on an eight digit machine, the two solutions will be identical on $t \geq 1$. Thus even though the solutions are linearly independent, their numerical tabulation would be the same.
- This phenomenon is called **numerical dependence**.

Example 3 (Numerical Dependence): Second Set of Solutions (3 of 6)

- We next consider two other linearly independent solutions,

$$\phi_3(t) = e^{\sqrt{10}\pi t}\ ,\ \phi_4(t) = e^{-\sqrt{10}\pi t}$$

where $\phi_3(t)$ and $\phi_4(t)$ satisfy the respective initial conditions

$$\phi_3(0) = 1,\ \phi_3'(0) = \sqrt{10}\pi;\quad \phi_4(0) = 1,\ \phi_4'(0) = -\sqrt{10}\pi$$

- Due to truncation and round-off errors, at any point t_n the data used in going to t_{n+1} are not precisely $\phi_4(t_n)$ and $\phi_4'(t_n)$.
- The solution of the initial value problem with these data at t_n involves $e^{-\text{sqrt}(10)\pi t}$ and $e^{\text{sqrt}(10)\pi t}$.
- Because the error at t_n is small, $e^{\text{sqrt}(10)\pi t}$ appears with a small coefficient, but nevertheless will eventually dominate, and the calculated solution will be a multiple of $\phi_3(t)$.

Example 3 (Numerical Dependence): Runge-Kutta Method (4 of 6)

- Consider then the initial value problem

$$y'' - 10\pi^2 y = 0;\ y(0) = 1,\ y'(0) = -\sqrt{10}\pi;\quad t > 0$$

- Using the Runge-Kutta method with eight digits of precision and $h = 0.01$, we obtain the following numerical results.
- The numerical values deviate from exact values as t increases due to presence, in the numerical approximation, of a small component of the exponentially growing solution $\phi_3(t)$.

t	Runge-Kutta	Exact
0.00	1.0	1.0
0.25	8.3439 x 10^(-2)	8.3438 x 10^(-2)
0.50	6.9623 x 10^(-3)	6.9620 x 10^(-3)
0.75	5.8409 x 10^(-4)	5.8089 x 10^(-4)
1.00	8.6688 x 10^(-5)	4.8469 x 10^(-5)
1.50	5.4900 x 10^(-3)	3.3744 x 10^(-7)
2.00	7.8852 x 10^(-1)	2.3492 x 10^(-9)
2.50	1.1326 x 10^2	1.6355 x 10^(-11)
3.00	1.6268 x 10^4	1.1386 x 10^(-13)
3.50	2.3368 x 10^6	7.9272 x 10^(-16)
4.00	3.3565 x 10^8	5.5189 x 10^(-18)
4.50	4.8211 x 10^10	3.8422 x 10^(-20)
5.00	6.9249 x 10^12	2.6749 x 10^(-22)

Example 3 (Numerical Dependence): Round-Off Error (4 of 6)

- With eight-digit arithmetic, we can expect a round-off error of the order 10^{-8} at each step. Since $e^{\text{sqrt}(10)\pi t}$ grows by a factor of 3.7×10^{21} from $t = 0$ to $t = 5$, an error of 10^{-8} near $t = 0$ can produce an error of order 10^{13} at $t = 5$, even if no further errors are introduced in the intervening calculations.
- From the results in the table, we see that this is what happens.

t	Runge-Kutta	Exact
0.00	1.0	1.0
0.25	8.3439 x 10^(-2)	8.3438 x 10^(-2)
0.50	6.9623 x 10^(-3)	6.9620 x 10^(-3)
0.75	5.8409 x 10^(-4)	5.8089 x 10^(-4)
1.00	8.6688 x 10^(-5)	4.8469 x 10^(-5)
1.50	5.4900 x 10^(-3)	3.3744 x 10^(-7)
2.00	7.8852 x 10^(-1)	2.3492 x 10^(-9)
2.50	1.1326 x 10^2	1.6355 x 10^(-11)
3.00	1.6268 x 10^4	1.1386 x 10^(-13)
3.50	2.3368 x 10^6	7.9272 x 10^(-16)
4.00	3.3565 x 10^8	5.5189 x 10^(-18)
4.50	4.8211 x 10^10	3.8422 x 10^(-20)
5.00	6.9249 x 10^12	2.6749 x 10^(-22)

Example 3 (Numerical Dependence): Unstable Problem (6 of 6)

- Our second order equation

$$y'' - 10\pi^2 y = 0, \quad t > 0$$

  is highly unstable.
- The behavior shown in this example is typical of unstable problems.
- One can track the solution accurately for a while, and the interval can be extended by using smaller step sizes or more accurate methods.
- However, the instability of the problem itself eventually takes over and leads to large errors.

Summary: Step Size

- The methods we have examined in this chapter have primarily used a uniform step size. Most commercial software allows for varying the step size as the calculation proceeds.
- Too large a step size leads to inaccurate results, while too small a step size will require more time and can lead to unacceptable levels of round-off error.
- Normally an error tolerance is prescribed in advance, and the step size at each step must be consistent with this requirement.
- The step size must also be chosen so that the method is stable. Otherwise small errors will grow and the results worthless.
- Implicit methods require than an equation be solved at each step, and the method used to solve the equation may impose additional restrictions on step size.

Summary: Choosing a Method

- In choosing a method, one must balance accuracy and stability against the amount of time required to execute each step.
- An implicit method, such as the Adams-Moulton method, requires more calculations for each step, but if its accuracy and stability permit a larger step size, then this may more than compensate for the additional computations.
- The backward differentiation methods of moderate order (four, for example) are highly stable and are therefore suitable for stiff problems, for which stability is the controlling factor.

Summary: Higher Order Methods

- Some current software allow the order of the method to be varied, as well as step size, as the method proceeds. The error is estimated at each step, and the order and step size are chosen to satisfy the prescribed tolerance level.
- In practice, Adams methods up to order twelve, and backward differentiation methods up to order five, are in use.
- Higher order backward differentiation methods are unsuitable because of lack of stability.
- The smoothness of f, as measured by the number of continuous derivatives that it possesses, is a factor in choosing the order of the method to be used. Higher order methods lose some of their accuracy if f is not smooth to a corresponding order.