Runge-Kutta Methods
Review of Heun’s Method  
(Derivation from Integration)

The idea is to construct an algorithm to solve the IVP ODE

\[ y'(x) = f(x, y(x)) \quad (9.1) \]

over \([x_0, x_1]\) with \(y(x_0) = y_0\).

To obtain the solution point \((x_1, y_1)\) we can use the fundamental theorem of calculus and integrate \(y'(x)\) over \([x_0, x_1]\) to get

\[
\int_{x_0}^{x_1} f(x, y(x)) \, dx = \int_{x_0}^{x_1} y'(x) \, dx = y(x_1) - y(x_0) \quad (9.2)
\]
A numerical integration method can be used to approximate the definite integral in equation (9.3). If the trapezoidal rule (Lecture 6) is used with the step size $h = x_1 - x_0$ the result is

$$y(x_1) = y(x_0) + \int_{x_0}^{x_1} f(x, y(x)) \, dx$$  \hspace{1cm} (9.3)$$

When equation (9.2) is solved from $y(x_1)$ the result is

$$y(x_1) \approx y(x_0) + \frac{h}{2}(f(x_0, y(x_0)) + f(x_1, y(x_1)))$$  \hspace{1cm} (9.4)$$
Review of Heun’s Method
(Derivation from Integration)

Notice that the formula on the right hand side of (9.4) involves the yet to be determined value $y(x_1)$. To estimate $y(x_1)$ we use Euler’s forward method

$$y(x_{i+1}) = y(x_i) + f(x_i, y(x_i))h$$

or, for our case

$$y(x_1) = y(x_0) + f(x_0, y(x_0))h$$  \hspace{1cm} (9.5)

Substituting (9.5) into (9.4), the resulting formula is called Heun’s method, given as

$$y(x_1) = y(x_0) + \frac{h}{2} (f(x_0, y_0) + f(x_1, y_0 + hf(x_0, y_0)))$$  \hspace{1cm} (9.6)
In Heun’s method, at each step:
(a) the Euler’s (forward) method is used as a prediction, and then,
(b) the trapezoidal rule is used to make a correction to obtain the final value.

The general step for Heun’s method is

\[
\begin{align*}
\text{Predictor:} & \quad p_{i+1} = y_i + hf(x_i, y_i) \\
\text{Corrector:} & \quad y_{i+1} = y_i + \frac{h}{2}(f(x_i, y_i) + f(x_{i+1}, p_{i+1})) \\
\end{align*}
\]

\[
x_{i+1} = x_i + h
\]
Example 9.1

Use Heun’s method to solve the IVP

\[ y' = \frac{x - y}{2} \]

on \([0, 3]\) with \(y(0) = 1\). Compare solutions for \(h = \frac{1}{2}, \frac{1}{4}, \frac{1}{8}\).

Solution:

(1) For \(h = \frac{1}{2}\)

at \(x_0 = 0, \ y_0 = 1\) we use to calculate

\[ y_0' = \frac{x_0 - y_0}{2} = -\frac{1}{2} \]
Example 9.1

At $x_1 = 0.5$, the predictor:

$$p_1 = y_0 + hy'_0 = 0.75$$

which we use to calculate

$$f(x_1, p_1) = f(0.5, 0.75)$$

$$= \frac{0.5 - 0.75}{2}$$

$$= -0.125$$

Thus, the corrector:

$$y_1 = y_0 + \frac{h}{2}(y'_0 + f(x_1, p_1))$$

$$= 0.84375$$
## Example 9.1

<table>
<thead>
<tr>
<th>$x_i$</th>
<th>$y_i$</th>
<th>$y_i$</th>
<th>$y_i$</th>
<th>$y_{\text{exact}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$h = 0.5$</td>
<td>$h = 0.25$</td>
<td>$h = 0.125$</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>0.125</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.25</td>
<td>0.898438 ($0.1055%$)</td>
<td>0.897717 ($0.0252%$)</td>
<td>0.897491</td>
<td></td>
</tr>
<tr>
<td>0.375</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.5</td>
<td>0.84375 ($0.8785%$)</td>
<td>0.838074 ($0.1999%$)</td>
<td>0.836801 ($0.0477%$)</td>
<td>0.836402</td>
</tr>
<tr>
<td>0.75</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0.831055 ($1.3986%$)</td>
<td>0.822196 ($0.3177%$)</td>
<td>0.820213 ($0.0758%$)</td>
<td>0.819592</td>
</tr>
<tr>
<td>1.5</td>
<td>0.930511 ($1.4623%$)</td>
<td>0.920143 ($0.3318%$)</td>
<td>0.917825 ($0.0791%$)</td>
<td>0.917100</td>
</tr>
<tr>
<td>2</td>
<td>1.117587 ($1.2639%$)</td>
<td>1.106800 ($0.2865%$)</td>
<td>1.104392 ($0.0683%$)</td>
<td>1.103638</td>
</tr>
<tr>
<td>2.5</td>
<td>1.373115 ($1.0004%$)</td>
<td>1.362593 ($0.2265%$)</td>
<td>1.360248 ($0.054%$)</td>
<td>1.359514</td>
</tr>
<tr>
<td>3</td>
<td>1.682121 ($0.7626%$)</td>
<td>1.672269 ($0.1725%$)</td>
<td>1.670076 ($0.0411%$)</td>
<td>1.669390</td>
</tr>
</tbody>
</table>
MATLAB Implementations

```matlab
function [x,y] = HeunMethod(f,xinit,xend,yinit,h)

% Number of iterations
N = (xend-xinit)/h;

x = zeros(1, N+1);
y = zeros(1, N+1);

y(1) = yinit;
x(1) = xinit;

for i=1:N
    x(i+1) = x(i)+h;
    y0 = y(i) + feval(f,x(i),y(i))*h;
    pred1 = feval(f,x(i),y(i));
    pred2 = feval(f,x(i+1),y0);
    y(i+1) = y(i) + h*(pred1+pred2)/2;
end

function dydx = Example_HeunMethod(x,y)

dydx = (x-y)/2;
end
```
MATLAB Implementations

Heun method with $h=0.5$

Heun method with $h=0.25$

Heun method with $h=0.125$

Euler forward method with $h=0.125$
Runge-Kutta Method

Here we see that Heun's method is an improvement over the rather simple Euler’s (forward) method. Although the method uses Euler's method as a basis, it goes beyond it, it attempts to compensate for the Euler method's failure to take the curvature of the solution curve into account. Heun's method is one of the simplest of a class of methods called predictor-corrector algorithms. One of the most powerful predictor-corrector algorithms of all—one which is so accurate, that most computer packages designed to find numerical solutions for differential equations will use it by default—is the fourth order Runge-Kutta method.
Second-Order Runge-Kutta Methods

The 2\textsuperscript{nd} order Runge-Kutta method simulates the accuracy of the Taylor series method of order 2. Recall the Taylor series formula for \( y(x + h) \):

\[
y(x + h) = y(x) + y'(x)h + \frac{y''(x)}{2}h^2 + C_Th^3 + \ldots \tag{9.8}
\]

Where \( C_T \) is a constant involving the third derivative of \( y(x) \) and the other terms in the series involve powers of \( h^n \) for \( n > 3 \).

The derivatives \( y'(x) \) and \( y''(x) \) must be expressed in terms of \( f(x, y) \) and its partial derivatives. Recall that

\[
y'(x) = f(x, y) \tag{9.9}
\]
Second-Order Runge-Kutta Methods

Equation (9.9) can be differentiated using the chain rule for differentiating a function of two variables, for example:

\[ z = f(u, v), \quad \text{where} \quad u = u(x) \text{ and } v = v(x) \]

then

\[ \frac{dz}{dx} = \frac{\partial f}{\partial u} \frac{du}{dx} + \frac{\partial f}{\partial v} \frac{dv}{dx} \]

Back to the equation (9.9),

\[ \frac{dy}{dx} = y'(x) = f(x, y) \]

\[ \frac{d^2y}{dx^2} = y''(x) = f'(x, y) = \frac{\partial f}{\partial x} \frac{dx}{dx} + \frac{\partial f}{\partial y} \frac{dy}{dx} \]

\[ = \frac{\partial f}{\partial x} + \frac{\partial f}{\partial y} f(x, y) \]
Second-Order Runge-Kutta Methods

Another notation for partial derivatives:

\[
\frac{\partial f}{\partial x} = f_x(x, y), \quad \frac{\partial f}{\partial y} = f_y(x, y)
\]

hence

\[
y''(x) = f'(x, y) = f_x(x, y) + f_y(x, y)f(x, y) \tag{9.10}
\]

Substitute the derivatives (9.9) and (9.10) into the Taylor series (9.8):

\[
y(x + h) = y(x) + hf(x, y) + \frac{h^2}{2!} f_x(x, y)
\]

\[
+ \frac{h^2}{2!} f_y(x, y)f(x, y) + C_T h^3 + \ldots \tag{9.11}
\]
Second-Order Runge-Kutta Methods

For second order, equation (9.11) can be rewritten as a linear combination of two function values to express \( y(x + h) \):

\[
y(x + h) = y(x) + Ahf_0 + Bhf_1 \quad (9.12)
\]

where

\[
f_0 = f(x, y)
\]

\[
f_1 = f_x(x, y) + f_y(x, y)f(x, y)
\]

\[
= f_x(x, y) + f_y(x, y)f_0
\]

\[
= f(x + Ph, y + Qhf_0)
\]

Next, we determine the values of A, B, P and Q.
Second-Order Runge-Kutta Methods

Recall Taylor series for functions more than one variable (Lecture 1):

\[ f(x_{i+1}, z_{i+1}) = f(x_i, z_i) + \frac{\partial f}{\partial x}(x_{i+1} - x_i) + \frac{\partial f}{\partial z}(z_{i+1} - z_i) \]
\[ + \frac{1}{2!} \left( \frac{\partial^2 f}{\partial x^2}(x_{i+1} - x_i)^2 + 2 \frac{\partial^2 f}{\partial x \partial z}(x_{i+1} - x_i)(z_{i+1} - z_i) \right. \]
\[ \left. + \frac{\partial^2 f}{\partial z^2}(z_{i+1} - z_i)^2 \right) + \ldots \]

and use it to expand \( f_1 \) in equation (9.13) to get

\[ f_1 = f(x, y) + f_x(x, y)Ph + f_y(x, y)Qhf_0 + \ldots \] (9.14)
Second-Order Runge-Kutta Methods

Then (9.14) is used in (9.12) to get the second-order Runge-Kutta expression for \( y(x + h) \):

\[
y(x + h) = y(x) + Ahf(x, y) + Bh[f(x, y) + f_x(x, y)Ph \\
+ f_y(x, y)Qhf(x, y) + \ldots]
\]
or

\[
y(x + h) = y(x) + (A + B)hf(x, y) + BPh^2f_x(x, y) \\
+ BQh^2f_y(x, y)f(x, y) + \ldots
\]  

(9.15)
Second-Order Runge-Kutta Methods

Comparing similar terms in equations (9.11) and (9.15):

\[ hf(x, y) \equiv (A + B)hf(x, y) \]

implies that \( 1 = A + B \)

\[ \frac{h^2}{2!} f_x(x, y) \equiv BPh^2 f_x(x, y) \]

implies that \( \frac{1}{2} = BP \)

\[ \frac{h^2}{2!} f_y(x, y)f(x, y) \equiv BQh^2 f_y(x, y)f(x, y) \]

implies that \( \frac{1}{2} = BQ \)
Second-Order Runge-Kutta Methods

Hence, we require that $A$, $B$, $P$, and $Q$ satisfy the relations

$$A + B = 1 \quad BP = \frac{1}{2} \quad BQ = \frac{1}{2}$$  \hspace{1cm} (9.16)

The second-order Runge-Kutta method in (9.15) will have the same order of accuracy as the Taylor’s method in (9.11).

Now, there are 4 unknowns with only three equations, hence the system of equations (9.16) is undetermined, and we are permitted to choose one of the coefficients.

**Case (i):** Choose $A = \frac{1}{2}$

This choice leads to $B = \frac{1}{2}$, $P = 1$, and $Q = 1$. 
Second-Order Runge-Kutta Methods

Substituting these values into equation (9.12) and (9.13) yields

\[ y(x + h) = y(x) + \frac{h}{2} [f(x, y) + f(x + h, y + hf(x, y))] \]  

(9.17)

If we break it down, eqn. (9.17) can be written as

\[ k_1 = f(x_i, y_i) \]

\[ k_2 = f(x_i + h, y_i + h k_1) \]

\[ y_{i+1} = y_i + \frac{h}{2} (k_1 + k_2) \]

This method is also called Heun’s method with a single corrector.
Case (ii): Choose $A = 0$

This choice leads to $B = 1$, $P = \frac{1}{2}$, and $Q = \frac{1}{2}$.

Substituting these values into equation (9.12) and (9.13) yields

$$y(x + h) = y(x) + hf \left( x + \frac{h}{2}, y + \frac{h}{2}f(x, y) \right) \quad (9.18)$$

or

$$k_1 = f(x_i, y_i)$$

$$k_2 = f \left( x_i + \frac{1}{2}h, y_i + \frac{1}{2}hk_1 \right)$$

$$y_{i+1} = y_i + hk_2$$
Case (iii): Choose $A = \frac{1}{3}$

This choice leads to $B = \frac{2}{3}$ and $P = Q = \frac{3}{4}$.

Substituting these values into equation (9.12) and (9.13) yields

$$y(x + h) = y(x) + \frac{h}{3} f(x, y) + \frac{2h}{3} f(x + \frac{3h}{4}, y + \frac{3h}{4} f(x, y))$$

or

$$k_1 = f(x_i, y_i)$$

$$k_2 = f \left( x_i + \frac{3}{4} h, y_i + \frac{3}{4} h k_1 \right)$$

$$y_{i+1} = y_i + \frac{h}{3} (k_1 + 2k_2)$$
Example 9.2

MATLAB implementations of 2nd order Runge-Kutta methods for IVP

\[ y' = 1 + y^2 \]

with \( y(0) = 0 \) on interval \([0, 1.4]\).

\[ h = 0.1 \]
Example 9.2

MATLAB implementations of 2nd order Runge-Kutta methods for IVP

\[ y' = 1 + y^2 \]

with \( y(0) = 0 \) on \([0, 0.14] \).

\[ h = 0.01 \]
Example 9.3

Write a Matlab solver using third-order Runge-Kutta method with $A = 0$, $A = 1/2$, and $A = 1/3$, and apply to the following problem

$$\frac{dy}{dx} = -2x^3 + 12x^2 - 20x + 8.5$$

from $x = 0$ to $x = 4$ using a step size of 5. The initial condition at $x = 0$ is $y = 0$. Compare the results. Refine the step size to 0.25 and 0.1 and calculate the relative error.
Third-Order Runge-Kutta Methods

The general formula is

$$y_{i+1} = y_i + \frac{h}{6}(k_1 + 4k_2 + k_3)$$ \hspace{1cm} (9.20)

where

$$k_1 = f(x_i, y_i)$$

$$k_2 = f\left(x_i + \frac{1}{2}h, y_i + \frac{1}{2}hk_1\right)$$

$$k_3 = f(x_i + h, y_i - hk_1 + 2hk_2)$$
Fourth-Order Runge-Kutta Methods

The classical fourth-order Runge-Kutta method

\[ y_{i+1} = y_i + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \]  \hspace{1cm} (9.21)

where

\[ k_1 = f(x_i, y_i) \]

\[ k_2 = f \left( x_i + \frac{1}{2}h, y_i + \frac{1}{2}hk_1 \right) \]

\[ k_3 = f \left( x_i + \frac{1}{2}h, y_i + \frac{1}{2}hk_2 \right) \]

\[ k_4 = f(x_i + h, y_i + hk_3) \]
Higher-Order Runge-Kutta Methods

(1) Butcher’s fifth-order Runge-Kutta method:

\[ y_{i+1} = y_i + \frac{h}{90} \left( 7k_1 + 32k_3 + 12k_4 + 32k_5 + 7k_6 \right) \]

where

\[ k_1 = f(x_i, y_i) \]

\[ k_2 = f \left( x_i + \frac{1}{4}h, y_i + \frac{1}{4}hk_1 \right) \]

\[ k_3 = f \left( x_i + \frac{1}{4}h, y_i + \frac{1}{8}hk_1 + \frac{1}{8}hk_2 \right) \]

\[ k_4 = f \left( x_i + \frac{1}{2}h, y_i - \frac{1}{2}hk_2 + hk_3 \right) \]
Higher-Order Runge-Kutta Methods

(1) Butcher’s fifth-order Runge-Kutta method:

\[ k_5 = f \left( x + \frac{3}{4}h, y + \frac{3}{16}hk_1 + \frac{9}{16}hk_4 \right) \]

\[ k_6 = f \left( x + h, y - \frac{3}{7}hk_1 + \frac{2}{7}hk_2 + \frac{12}{7}hk_3 - \frac{12}{7}hk_4 + \frac{8}{7}hk_5 \right) \]
Example 9.4

MATLAB implementations of 2\textsuperscript{nd} order, 3\textsuperscript{rd} order, and 4\textsuperscript{th} order Runge-Kutta methods for IVP

\[ y' = e^{-2x} - 2y \]

with \( y(0) = \frac{1}{10} \) on \([0, 2]\) and step size \( h = 0.2 \) and \( h = 0.1 \).

The exact solution is

\[ y(x) = \frac{1}{10} e^{-2x} + x e^{-2x} \]
$h = 0.2$

2nd Order Runge-Kutta $A=1/2$

2nd Order Runge-Kutta $A=0$

3rd Order Runge-Kutta

4th Order Runge-Kutta
$h = 0.1$

**2nd Order Runge-Kutta $A = \frac{1}{2}$**

**2nd Order Runge-Kutta $A = 0$**

**3rd Order Runge-Kutta**

**4th Order Runge-Kutta**
Precision of the Runge-Kutta Method

Assume that $y(x)$ is the solution of the IVP. If $y(x) \in C^5[x_0, b]$ and $\{(x_i, y_i)\}_{i=0}^n$ is the sequence of approximations generated by the Runge-Kutta method of order 4, then

$$|\epsilon_i| = |y(x_i) - y_i| = O(h^4)$$
$$|\epsilon_{i+1}| = |y(x_{i+1}) - y_i - hT_n(x_i, y_i)| = O(h^5)$$

In particular, the global error at the end of the interval will satisfy

$$E(y(b), h) = |y(b) - y_n| = O(h^4)$$
Precision of the Runge-Kutta Method

If approximations are computed using the step size $h$ and $h/2$, we should have

$$E(y(b), h) \approx C h^4$$

$E(y(b), \frac{h}{2}) \approx C \frac{h^4}{16} \approx E(y(b), h)$

The idea is that, if the step size in Runge-Kutta order 4 is reduced by a factor of $\frac{1}{2}$, we can expect that the global error will be reduced by a factor of $\frac{1}{16}$. 
Precision of the Runge-Kutta Method

A termination criterion for convergence of the corrector of all these methods is provided by

\[ |\varepsilon_a| = \left| \frac{y_{i+1}^{j} - y_{i+1}^{j-1}}{y_{i+1}^{j}} \right| \times 100\% \]  \hspace{1cm} (9.23)

where \( y_{i+1}^{j-1} \) is the result from the prior iteration of the corrector, and \( y_{i+1}^{j} \) is the result from the present iteration of the corrector.
Adaptive Runge-Kutta Methods

Methods that have been presented so far employ a constant step size. For a significant number of problems, this can represent a serious limitation.

Why do we need adaptive methods? In this figure, for most of the range, the solution changes gradually. We may use a fairly large step for such behavior. However, there is a range that undergoes abrupt change, in the region between $x = 1.75$ to $x = 2.25$. Consequently, smaller step size is required.
Adaptive Runge-Kutta Methods

If a constant step-size algorithm were employed, we would have to apply the smaller step size for entire the region, which would not be necessary and be wasted on the regions of gradual change.

Adaptive methods that adjust step size can give great advantage because they “adapt” to the solution’s trajectory. Implementation of such approaches requires that an estimate of the local truncation error (LTE) be obtained at each step, which serves as a basis for adjusting step size.
Adaptive Runge-Kutta Methods

Two primary approaches:

1) the error is estimated as the difference between two predictions using the same-order Runge-Kutta method but with different step sizes.

2) the local truncation error (LTE) is estimated as the difference between two predictions using different-order Runge-Kutta methods.
Adaptive Runge-Kutta Methods

(1) Step-halving method

This step involves taking each step twice, that is:
- once as a full step
- independently as two half steps

The difference in the results represents an estimate of the local truncation error. If $y_1$ designates the single-step prediction and $y_2$ designates the prediction using the two half steps, the error can be represented as

$$\varepsilon = y_2 - y_1$$

(9.24)
Equation (9.24) can serve two purposes:
- to provide a criterion for step-size control
- to correct the $y_2$ prediction

For the fourth-order Runge-Kutta version, the corrections is

$$y_2 \leftarrow y_2 + \frac{\varepsilon}{15}$$  

(9.25)
Adaptive Runge-Kutta Methods

(2) Runge-Kutta-Fehlberg (RKF45) method

Each step requires the use of the following six values

\[ k_1 = f(x_i, y_i) \]
\[ k_2 = f \left(x_i + \frac{1}{5} h, y_i + \frac{1}{5} h k_1\right) \]
\[ k_3 = f \left(x_i + \frac{3}{10} h, y_i + \frac{3}{40} h k_1 + \frac{9}{40} h k_2\right) \]
\[ k_4 = f \left(x_i + \frac{3}{5} h, y_i + \frac{3}{10} h k_1 - \frac{9}{10} h k_2 + \frac{6}{5} h k_3\right) \]
Adaptive Runge-Kutta Methods

\[ k_5 = f \left( x_i + h, y_i - \frac{11}{54}hk_1 + \frac{5}{2}hk_2 - \frac{70}{27}hk_3 + \frac{35}{27}hk_4 \right) \]

\[ k_6 = f \left( x_i + \frac{7}{8}h, y_i + \frac{1631}{55296}hk_1 + \frac{175}{512}hk_2 + \frac{575}{13824}hk_3 \right. \]

\[ + \frac{44275}{110592}hk_4 + \frac{253}{4096}hk_5 \left. \right) \]

An approximation using a fourth-order Runge-Kutta method is given by:

\[ y_{i+1} = y_i + h \left( \frac{37}{378}k_1 + \frac{250}{621}k_3 + \frac{125}{594}k_4 + \frac{512}{1771}k_6 \right) \]

(9.26)
Adaptive Runge-Kutta Methods

A better value for the solution is determined by using a fifth-order Runge-Kutta method which approximates the value of the local truncation error (at a specific node):

$$y_{i+1} = y_i + h \left( \frac{2825}{27648} k_1 + \frac{18575}{48384} k_3 + \frac{13525}{55296} k_4 
+ \frac{277}{14336} k_5 + \frac{1}{4} k_6 \right) \quad (9.27)$$
Example 9.5

Use the step-halving method to integrate

\[ \frac{dy}{dx} = 4e^{0.8x} - 0.5y \]

from \( x = 0 \) to \( 2 \) using \( h = 2 \) and initial condition \( y(0) = 2 \).
Example 9.5

Solution:
The differential equation can be integrated analytically, where the exact/true solution is

\[ y(x) = \frac{4}{1.3} \left( e^{0.8x} - e^{-0.5x} \right) + 2e^{-0.5x} \]

where at \( x = 0 \), \( y(0) = 2 \) and at \( x = 2 \), \( y(2) = 14.84392 \).

We start by calculating the four components of the fourth-order Runge-Kutta method as given in eqn. (9.21) at \( x = 0 \) with \( y(0) = 2 \):

\[ k_1 = f(x, y) = f(0, 2) = 4e^{0.8(0)} - 0.5(2) = 3 \]

\[ k_2 = f(x + \frac{h}{2}, y + \frac{h}{2}k_1) = f(1, 5) = 6.40216 \]
Example 9.5

\[ k_3 = f\left(x + \frac{h}{2}, y + \frac{h}{2}k_2\right) = f(1, 8.40216) = 4.70108 \]

\[ k_4 = f\left(x + h, y + k_3h\right) = f(2, 11.40216) = 14.11105 \]

Then the single prediction with step \( h \) at \( x = 2 \) is computed as:

\[ y(x + h) = y(x) + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \]

\[ y(2) = y(0) + \frac{2}{6}(3 + 2(6.40216 + 4.70108) + 14.11105) \]

\[ = 15.10584 \]
Example 9.5

For the two half-step where now \( h = 1 \), at \( x = 1 \):

\[
k_1 = f(x, y) = f(0, 2) = 4e^{0.8(0)} - 0.5(2) = 3
\]

\[
k_2 = f\left(0 + \frac{1}{2}, 2 + \frac{1}{2}\right) = 4.21730
\]

\[
k_3 = f\left(0 + \frac{1}{2}, 2 + \frac{1}{2}\cdot4.21370\right) = 3.91297
\]

\[
k_4 = f\left(0 + 1, 2 + (3.91297)(1)\right) = 5.945681
\]

the prediction is:

\[
y(1) = y(0) + \frac{1}{6}(3 + 2(4.21730 + 3.91297) + 5.945681)
\]

\[
= 6.20104
\]
Example 9.5

and at \( x = 2 \) (with \( h = 1 \), and using \( y(1) = 6.20104 \)):

\[
\begin{align*}
k_1 &= f(1, 6.20104) = 5.80164 \\
k_2 &= f\left(1 + \frac{1}{2}, 6.20104 + \frac{1}{2}5.80164\right) = 8.72954 \\
k_3 &= f\left(1 + \frac{1}{2}, 6.20104 + \frac{1}{2}8.72954\right) = 7.99756 \\
k_4 &= f\left(1 + 1, 6.20104 + (7.99756)(1)\right) = 12.71283
\end{align*}
\]

the prediction is:

\[
\begin{align*}
y(2) &= y(1) + \frac{1}{6}(5.80164 + 2(8.72954 + 7.99756) + 12.71283) \\
&= 14.86249
\end{align*}
\]
Example 9.5

Using eqn. (9.25), the approximate error is

\[ E_a = \frac{14.86249 - 15.10584}{15} = -0.01622 \]

and the true error

\[ E_t = 14.84392 - 14.86249 = -0.01857 \]

To correct the prediction, we use the error estimate in eqn. (9.25):

\[ y(2) = 14.86249 - 0.01622 = 14.84627 \]