Euler’s Methods
(a family of Runge-Kutta methods)
ODE IVP

An Ordinary Differential Equation (ODE) is an equation that contains a function having one independent variable:

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

The equation is coupled with an initial value/condition \( y(0) = y_0 \) (i.e., value of \( y \) at \( x = 0 \)).

Hence, such equation is usually called **ODE Initial Value Problem (IVP)**.

Different notations for derivatives:

Space/position: \( y' \equiv y^{(1)} \equiv \frac{dy}{dx} \)

Time: \( y_t \equiv \dot{y} \equiv \frac{dy}{dt} \)
Recall Taylor Series (Lecture 1)

If the y-axis and origin are moved $a$ units to the left, the equation of the same curve relative to the new axis becomes $y = f(a+x)$ and the function value at P is $f(a)$.

At point Q:

$$f(a + h) = f(a) + hf'(a) + \frac{h^2}{2!}f''(a) + \ldots$$
Euler’s Method from Taylor Series

The approximation used with Euler’s method is to take only the first two terms of the Taylor series:

$$f(a + h) = f(a) + hf'(a)$$

In general form:

new value = old value + step size \times slope

If $f(a + h) = y_{i+1}$ and $f(a) = y_i$ as well as $f'(a) = y'_i$, then

$$y_{i+1} = y_i + hy'_i$$

(8.1)

with $i = 0, 1, 2, \ldots, N - 1$. 
Euler’s (Forward) Method

Alternatively, from step size $h = x_{i+1} - x_i$ and rearrange to $x_{i+1} = x_i + h$ we use the Taylor series to approximate the function $f(x_{i+1}) = y_{i+1}$ around $f(x_i) = y_i$ with step size $h = x_{i+1} - x_i$. Taking only the first derivative:

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

(8.2)

where $f(x_i, y_i)$ is the differential equation evaluated at $x_i$ and $y_i$.

This formula is referred to as Euler’s forward method, or explicit Euler’s method, or Euler-Cauchy method, or point-slope method.
Example 8.1

Obtain a numerical solution of the differential equation

\[ \frac{dy}{dx} = 3(1 + x) - y \]

given the initial conditions that \( x = 1 \) when \( y = 4 \), for the range \( x = 1 \) to \( x = 2 \) with intervals of 0.2.
Example 8.1

Obtain a numerical solution of the differential equation

\[ \frac{dy}{dx} = 3(1 + x) - y \]

given the initial conditions that \( x = 1 \) when \( y = 4 \), for the range \( x = 1 \) to \( x = 2 \) with intervals of 0.2.

Solution:

\[ \frac{dy}{dx} = y' = 3(1 + x) - y \]

with \( x_0 = 1 \) and \( y_0 = 4 \), then \( y'_0 = 3(1 + 1) - 4 = 2 \).

By Euler’s method:

\[ y_1 = y_0 + hy'_0 \]

\[ y_1 = 4 + (0.2)(2) = 4.4 \]
Example 8.1

At $x_1 = x_0 + h = 1 + 0.2 = 1.2$ and $y_1 = 4.4$ where

$$y'_1 = 3(1 + x_1) - y_1$$

$$y'_1 = 3(1 + 1.2) - 4.4 = 2.2$$

then

$$y_2 = y_1 + hy'_1$$

$$y_2 = 4.4 + (0.2)(2.2) = 4.84$$

If the step by step Euler’s method is continued, we can present the results in a table:

<table>
<thead>
<tr>
<th>$x_i$</th>
<th>$y_i$</th>
<th>$y'_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>1.2</td>
<td>4.4</td>
<td>2.2</td>
</tr>
<tr>
<td>1.4</td>
<td>4.84</td>
<td>2.36</td>
</tr>
<tr>
<td>1.6</td>
<td>5.312</td>
<td>2.488</td>
</tr>
<tr>
<td>1.8</td>
<td>5.8096</td>
<td>2.5904</td>
</tr>
<tr>
<td>2</td>
<td>6.32768</td>
<td></td>
</tr>
</tbody>
</table>
Example 8.1

To compare the approximations from Euler’s method with the exact solution, the ODE

\[
\frac{dy}{dx} = 3(1 + x) - y
\]

can be solved analytically using **integrating factor** method. Rearrange the equation into the form

\[
\frac{dy}{dx} + Py = Q \quad (8.3)
\]

so now \( \frac{dy}{dx} + y = 3 + 3x \) where \( P = 1 \) and \( Q = 3 + 3x \).

Take the integrating factor \( e^{\int P \, dx} \) and with \( P = 1 \) thus

\[
e^{\int P \, dx} = e^{\int dx} = e^x
\]

The general solution of the equation (8.3) is:

\[
ye^{\int P \, dx} = \int e^{\int P \, dx} Q \, dx \quad (8.4)
\]
Example 8.1

from which

\[ ye^x = \int e^x (3 + 3x) \, dx \]

\[ ye^x = 3 \int e^x \, dx + 3 \int xe^x \, dx \]

The second term on the right hand side is solved using integration by parts. Hence

\[ ye^x = 3xe^x + C \]

When \( x = 1, y = 4 \) thus \( 4e = 3e + C \) from which \( C = e \). Therefore

\[ ye^x = 3xe^x + e \]
or:

\[ y = 3x + e^{(1-x)} \]
Example 8.1

- **Exact solution**
- **Euler's method**
function [x,y] = EulerForward(f,xinit,xend,yinit,h)

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

% Initialize arrays
% The first elements take xinit and yinit, correspondingly,
% the rest fill with 0s.
x = zeros(1, N+1);
y = zeros(1, N+1);

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

for i=1:N
    x(i+1) = x(i)+h;
    y(i+1) = y(i) + h*feval(f,x(i),y(i));
end
end
MATLAB Implementations

```matlab
function dydx = EulerFunction(x,y)
    dydx = 3*(1+x)-y;
end

a = 1; b = 2; ya = 4; h = 0.2;
N = (b-a)/h;
t = a:h:b;

[x,y] = EulerForward('EulerFunction', a, b, ya, h);

ye = y; % Numerical solution from using Euler's forward method
yi = 3*t+exp(1-t); % Exact solution

hold on;
plot(t,yi,'r','LineWidth', 2);
plot(t,ye,'b','LineWidth', 2); hold off;
box on;
```
Error Analysis

The numerical solution of ODEs involves two types of error:

1. Truncation, or discretization errors caused by the nature of the techniques used to approximate values of $y$.
2. Round-off errors, caused by the limited numbers of significant digits that can be retained by a computer.

The truncation errors are composed of two parts:

a. Local truncation error $\rightarrow$ from an application of the method in question over a single step.

b. Propagated truncation error $\rightarrow$ from approximations produced during the previous step.

The sum of (a) and (b) is the **global truncation error**.
Error Analysis

The local truncation error is the difference between the numerical solution after one step $y_1$ and the exact solution at $x_1 = x_0 + h$.

Using equation (8.2), the numerical solution is given by:

$$y_1 = y_0 + f(x_0, y_0)h$$  \hspace{1cm} (8.5)

where $f(x_0, y_0) = y'_0$

For the exact solution, we use the Taylor expansion of the function $y$ around $x_0$:

$$y(x_0 + h) = y_0 + y'_0 h + \frac{y'’_0}{2!} h^2 + O(h^3)$$  \hspace{1cm} (8.6)
Error Analysis

Subtracting equation (8.6) from (8.5) yields

$$y(x_0 + h) - y_1 = \frac{y''}{2!} h^2 + \mathcal{O}(h^3)$$

The difference between the exact and the numerical solutions is the true local truncation error $E_t$:

$$E_t = \frac{f'(x_0, y_0)}{2!} h^2 + \mathcal{O}(h^3) \quad (8.7)$$

The general form of the true local truncation error is given by:

$$E_t = \frac{f'(x_i, y_i)}{2!} h^2 + \frac{f''(x_i, y_i)}{3!} h^3 + \ldots + \mathcal{O}(h^{n+1}) \quad (8.8)$$
Error Analysis

The expressions (8.7) and (8.8) show that for sufficiently small $h$, the errors usually decrease as the order increases, the result is represented as the approximate local truncation error, formulated as

$$E_a = \frac{f'(x_0, y_0)}{2!} h^2$$  \hspace{1cm} (8.9)

or

$$E_a = \mathcal{O}(h^2)$$  \hspace{1cm} (8.10)
Error Analysis

Because Euler’s method uses straight-line segments to approximate the solution, hence the method is also referred to as a first-order method.

From error estimates shown in Lecture 7, then the global truncation error $\propto O(h)$.

In general, if the local truncation error is $O(h^{n+1})$ the global truncation error is $O(h^n)$.
Example 8.2

Use eqn. (8.8) to estimate the error of the first step of the following equation when integrating from $x = 0$ to $x = 4$ with step size $h = 0.5$. Use the results to determine the error due to each higher-order term of the Taylor series expansion.

Solution:
For the above problem, eqn. (8.8) can be written in:

$$
E_t = \frac{f'(x_i, y_i)}{2!} h^2 + \frac{f''(x_i, y_i)}{3!} h^3 + \frac{f'''(x_i, y_i)}{4!} h^4
$$
Example 8.2

where

\[
  f'(x_i, y_i) = -6x^2 + 24x - 20
\]

\[
  f''(x_i, y_i) = -12x + 24
\]

\[
  f'''(x_i, y_i) = -12
\]

with derivative of order higher than 3 is equal to zero.

The error due to truncation of the second-order term can be calculated as

\[
  E_{t,2} = \frac{(-6)(0) + (24)(0) - 20}{2} (0.5)^2 = -2.5
\]
Example 8.2

The error due to truncation of the third-order term can be calculated as

$$E_{t,3} = \frac{(-12)(0) + 24}{6} (0.5)^3 = 0.5$$

and the error due to truncation of the fourth-order term can be calculated as

$$E_{t,4} = \frac{-12}{24} (0.5)^4 = -0.03125$$

The three results can be added to yield the total truncation error:

$$E_t = E_{t,2} + E_{t,3} + E_{t,4} = -2.03125$$
Example 8.2

If the problem in question is equipped with additional information, i.e., initial condition where at $x = 0$, $y = 1$, the global error can be calculated using formula

$$E_t = \text{true} - \text{approximate}$$

We use eqn. (8.2):\n
$$y_{i+1} = y_i + f(x_i, y_i)h$$

where $y(0) = 1$ and the slope at $x = 0$ is

$$f(0, 1) = -2(0)^3 + 12(0)^2 - 20(0) + 8.5 = 8.5$$

due to

$$y(0.5) = y(0) + f(0, 1)0.5 = 5.25$$
Example 8.2

The true solution at $x = 0.5$ is

$$y = -0.5x^4 + 4x^3 - 10x^2 + 8.5x + 1$$

$$= 3.21875$$

thus the global error is

$$E_t = 3.21875 - 5.25 = -2.03125$$
Effect of Reduced Step on Euler’s Method

From the previous example

\[ \frac{dy}{dx} = 3(1 + x) - y \]

Reducing the interval or step size to 0.1 within the range \( x = 1 \) to \( x = 2 \) we get:

<table>
<thead>
<tr>
<th>( x_i )</th>
<th>( y_i )</th>
<th>( y'_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>1.1</td>
<td>4.2</td>
<td>2.1</td>
</tr>
<tr>
<td>1.2</td>
<td>4.41</td>
<td>2.19</td>
</tr>
<tr>
<td>1.3</td>
<td>4.629</td>
<td>2.271</td>
</tr>
<tr>
<td>1.4</td>
<td>4.8561</td>
<td>2.3439</td>
</tr>
<tr>
<td>1.5</td>
<td>5.09049</td>
<td>2.40951</td>
</tr>
<tr>
<td>1.6</td>
<td>5.331441</td>
<td>2.468559</td>
</tr>
<tr>
<td>1.7</td>
<td>5.5782969</td>
<td>2.5217031</td>
</tr>
<tr>
<td>1.8</td>
<td>5.83046721</td>
<td>2.56953279</td>
</tr>
<tr>
<td>1.9</td>
<td>6.087420489</td>
<td>2.612579511</td>
</tr>
<tr>
<td>2</td>
<td>6.3486784401</td>
<td>2.6513215599</td>
</tr>
</tbody>
</table>
Effect of Reduced Step on Euler’s Method

![Graph showing the effect of Euler's method compared to the exact solution with reduced step size.]
(1) Heun’s Method

Recall in Euler’s method:

\[ y'_i = f(x_i, y_i) \]  \hspace{1cm} (8.11)

is used to extrapolate linearly to \( y_{i+1} \)

\[ y^0_{i+1} = y_i + f(x_i, y_i)h \]  \hspace{1cm} (8.12)

which is called a predictor equation. It provides an estimate of \( y_{i+1} \) that allows the calculation of an estimated slope at the end of the interval:

\[ y'_{i+1} = f(x_{i+1}, y^0_{i+1}) \]  \hspace{1cm} (8.13)
Improvements of Euler’s Method

Thus, the two slopes in (8.11) and (8.13) can be combined to obtain an average slope

\[ \bar{y}' = \frac{y'_i + y'_{i+1}}{2} = \frac{f(x_i, y_i) + f(x_{i+1}, y^0_{i+1})}{2} \]

The average slope is used to extrapolate linearly from \( y_i \) to \( y_{i+1} \) using Euler’s method:

\[ y_{i+1} = y_i + \frac{f(x_i, y_i) + f(x_{i+1}, y^0_{i+1})}{2} h \]

Or

Predictor: \( y^0_{i+1} = y_i + f(x_i, y_i)h \) \hspace{1cm} (8.14)

Corrector: \( y_{i+1} = y_i + \frac{f(x_i, y_i) + f(x_{i+1}, y^0_{i+1})}{2} h \) \hspace{1cm} (8.15)
Improvements of Euler’s Method

(2) Midpoint Method

To predict a value of $y$ at the midpoint of the interval:

$$y_{i+\frac{1}{2}} = y_i + f(x_i, y_i) \frac{h}{2}$$

This predicted value is used to calculate a slope at the midpoint:

$$y'_1 + \frac{1}{2} = f(x_1 + \frac{1}{2}, y_1 + \frac{1}{2})$$
Improvements of Euler’s Method

The slope is then used to extrapolate linearly from \( x_i \) to \( x_{i+1} \) to get

\[
y_{i+1} = y_i + f(x_{i+\frac{1}{2}}, y_{i+\frac{1}{2}})h
\]  

(8.16)
Stability

The (explicit) forward Euler’s method is easy to implement. The drawback arises from the limitations on the step size to ensure numerical stability.

To see this, let’s examine a linear IVP given by

\[ \frac{dy}{dx} = -ay \]  

(8.17)

with \( y(0) = 1 \) and \( a > 0 \).

The exact solution of (8.17) is \( y = e^{-ax} \), which is a stable and a very smooth solution with \( y(0) = 1 \) and \( y(\infty) = 0 \).
Stability

Applying the (explicit) forward Euler’s method:

\[ y_{i+1} = y_i - ah y_i = (1 - ah)y_i = (1 - ah)^{i+1}y_0 \]

The solution is decaying (stable) if

\[ |1 - ah| < 1 \]

To prevent the amplification of the errors in the iteration process, we require, for stability of the forward Euler’s method:

\[ h < \frac{2}{a} \quad (8.18) \]
The backward method computes the approximations using

\[ y_{i+1} = y_i + f(x_{i+1}, y_{i+1})h \]  \hspace{1cm} (8.19)

which is an implicit method, in the sense that in order to find \( y_{i+1} \) the nonlinear equation (8.19) has to be solved.

Using equation (8.17) gives us

\[ y_{i+1} = \frac{1}{1 + ah} y_i = \left( \frac{1}{1 + ah} \right)^{i+1} y_0 \]  \hspace{1cm} (8.20)

and the solution is decaying (stable) if \( |1 + ah| > 1 \).
Example 8.3

Using the same example as in the forward method with $h = 0.2$

$$\frac{dy}{dx} = 3(1 + x) - y$$

Solution: $$\frac{dy}{dx} = y' = 3(1 + x) - y$$

with $x_0 = 1$ and $y_0 = 4$, then $y'_0 = 3(1 + 1) - 4 = 2$.

By Euler’s backward method:

$$y_1 = y_0 + y'_1 h \quad (8.21)$$

where $y'_1 = 3(1 + x_1) - y_1$. Substitute this into (8.21) and
Example 8.3

Now with $x_1 = x_0 + h = 1 + 0.2 = 1.2$, we solve for $y_1$:

$$y_1 = y_0 + (3(1 + x_1) - y_1)h$$

$$y_1 = 4 + (3(1 + 1.2) - y_1)(0.2)$$

$$\frac{y_1 - 4}{0.2} = 3(1 + 1.2) - y_1$$

$$5y_1 - 20 = 6.6 - y_1$$

$$y_1 = 4.433$$

And we use it to get the slope

$$y'_1 = 3(1 + x_1) - y_1$$

$$y'_1 = 3(1 + 1.2) - 4.433 = 2.167$$
The next iteration is
\[ y_2 = y_1 + y_2' h \]
where \( y_2' = 3(1 + x_2) - y_2 \) and the next point of iteration \( x_2 = x_1 + h = 1.2 + 0.2 = 1.4 \) we solve for \( y_2 \) :
\[ y_2 = y_1 + (3(1 + x_2) - y_2)h \]
\[ y_2 = 4.433 + (3(1 + 1.4) - y_2)(0.2) \]
\[ y_2 = 4.8942 \]
And continue the iteration until \( x = 2 \).
Example 8.3
Write Matlab solver to approximate the following function using Heun’s Method

\[ \frac{1}{x} \frac{dy}{dx} + 4y = 2 \]

given the initial conditions \( x = 0 \) when \( y = 4 \) within the range \( x = 0 \) to \( x = 2 \) with intervals of 0.1.
function [x,y] = HeunsMethod(f,xinit,xend,yinit,h)

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

% Initialize arrays
% The first elements take xinit and yinit, correspondingly, 
% the rest fill with 0s.
x = [xinit zeros(1, N)];
y = [yinit zeros(1, N)];

for i=1:N
    x(i+1) = x(i)+h;
    % Predictor
    ynew = y(i) + h*feval(f,x(i),y(i));
    % Corrector
    y(i+1) = y(i) + h*feval(f,x(i),y(i))/2 + h*feval(f,x(i+1),ynew)/2;
end
end
MATLAB Implementation

```matlab
function dydx = HeunFunction(x,y)
    dydx = 2*x - 4*x*y;
end

clear all
close all

a = 0; b = 2; y0 = 4; h = 0.1;

t = a:h:b;

[x, y] = HeunsMethod('HeunFunction', a, b, y0, h);

ye = y; % Numerical solution from using Euler’s backward method
yi = 0.5*(1+7*exp(-2*x.^2)); % Exact solution, for comparison
err = abs(yi - ye)*100/yi; % relative error

hold on;
plot(t, yi, 'r', 'LineWidth', 2);
plot(t, ye, 'b', 'LineWidth', 2);
hold off;
box on;
```