

Numerical Solution of Ordinary Differential Equations

Mark A. Austin

University of Maryland

austin@umd.edu

ENCE 201, Fall Semester 2023

July 14, 2023

Overview

- 1 Problem Space (Ordinary Differential Equations)
 - Problem Space, Analytic and Numerical Solutions
- 2 Euler Method
 - Mathematical Theory, Examples
- 3 Modified Euler Method
 - Mathematical Theory, Examples
- 4 Python Code Listings

Ordinary Differential Equations

Ordinary Differential Equations

Example 1: Oscillatory mechanical systems:

$$[m]y''(t) + [c]y'(t) + [k]y(t) = f(t). \quad (1)$$

Example 2: Basic equation for beam deflection:

$$[EI] \frac{d^4y}{dx^4} + w(x) = 0. \quad (2)$$

Example 3: Equation for cable suspension:

$$\frac{d^2y}{dx^2} = \frac{w}{H} \sqrt{1 + \left(\frac{dy}{dx}\right)^2} \quad (3)$$

Ordinary Differential Equations

Initial Value Problems (IVP): First order equation:

$$\frac{dy}{dx} = f(x, y) \quad \text{with} \quad y(x_0) = y_0. \quad (4)$$

Note: In general, an **n-th order differential equation**, i.e.,

$$Ay^n(t) + By^{n-1}(t) + \dots = f(t). \quad (5)$$

can be written as a **family of 1st order differential equations**:

$$\left. \begin{aligned} y_1'(x) &= f_1(x, y_1, y_2, \dots, y_n) \\ y_2'(x) &= f_2(x, y_1, y_2, \dots, y_n) \\ &\dots \\ y_n'(x) &= f_n(x, y_1, y_2, \dots, y_n) \end{aligned} \right\}. \quad (6)$$

Ordinary Differential Equations

Key Challenge: Sometimes analytic solutions are unobtainable, even for seemingly simple problems:

$$\frac{dy}{dx} = x^2 + y^2. \quad (7)$$

Discrete Approximation: Abandon idea of computing solution for all values of x .

Analytic Solution

————— x axis

Discrete Approximation

—|—|—|— x axis
x₀ x₁ x₂

Computational Strategy: Compute solutions to equation 4 at a discrete number of points.

Ordinary Differential Equations

Numerical Solution of Initial Value Problem:

$$\frac{dy}{dx} = f(x, y) \iff y(x_{i+1}) = y(x_i) + \int_{x_i}^{x_{i+1}} f(x, y(x)) dx. \quad (8)$$

Can approximate integral with Trapezium Rule:

$$y(x_{i+1}) = y(x_i) + \frac{h}{2} [f(x_i, y_i(x_i)) + f(x_{i+1}, y(x_{i+1}))]. \quad (9)$$

where $x_{i+1} = x_i + h$.

Equation 9 is a **discrete approximation**, **implicit** in the value of $y(x_{i+1})$.

Euler Method

Euler Method

Mathematical Theory. Consider the IVP:

$$\begin{cases} \frac{dy}{dx} = f(x, y) \\ y(0) = y_o. \end{cases} \quad (10)$$

Let:

$$\frac{dy}{dx} = \left[\frac{y_{i+1} - y_i}{h} \right] = f(y_i, x_i). \quad (11)$$

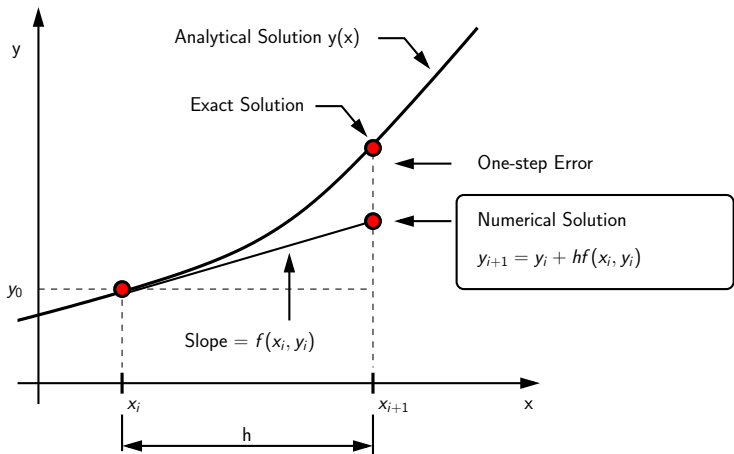
Rearranging equation 11 gives the Euler update:

$$y_{i+1} = y_i + hf(y_i, x_i). \quad (12)$$

One-step truncation error: $O(h^2)$.

Euler Method

Discrete Update for Euler's Method:



Euler Method

Example 1. Let:

$$\frac{dy}{dx} = f(x, y) = x + y \quad \text{with} \quad y(0) = 0. \quad (13)$$

Analytic Solution:

$$y(x) = e^x - x - 1. \quad (14)$$

Large Step Size: $h = 0.5$

x coordinate	Euler Method	Exact $y(x)$
0.0000000e+00	0.0000000e+00	0.0000000e+00
5.0000000e-01	0.0000000e+00	1.4872127e-01
1.0000000e+00	2.5000000e-01	7.1828183e-01

Poor accuracy ...

Euler Method

Small Step Size: $h = 0.10$

x coordinate	Euler Method	Exact y(x)
0.0000000e+00	0.0000000e+00	0.0000000e+00
1.0000000e-01	0.0000000e+00	5.1709181e-03
2.0000000e-01	1.0000000e-02	2.1402758e-02
3.0000000e-01	3.1000000e-02	4.9858808e-02
4.0000000e-01	6.4100000e-02	9.1824698e-02
.....		
9.0000000e-01	4.5794769e-01	5.5960311e-01
1.0000000e+00	5.9374246e-01	7.1828183e-01

Still not very accurate ...

Modified Euler Method

Modified Euler Method

Computational Strategy: Base update on **average of gradients** at **successive points**:

Prediction Step:

$$y_{i+1}^* = y_i + hf(x_i, y_i) \quad (15)$$

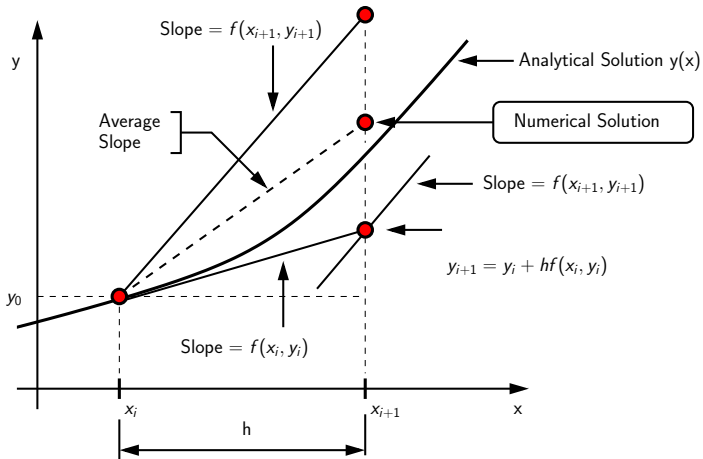
Correction Step:

$$y_{i+1} = y_i + \frac{h}{2} [f(x_i, y_i) + f(x_{i+1}, y_{i+1}^*)] \quad (16)$$

One-step truncation error: $O(h^3)$.

Modified Euler Method

Discrete Update for Modified Euler's Method:



Modified Euler Method

Example 1. Let:

$$\frac{dy}{dx} = f(x, y) = x + y \quad \text{with} \quad y(0) = 0. \quad (17)$$

Analytic Solution:

$$y(x) = e^x - x - 1. \quad (18)$$

Predictor-Corrector Algorithm:

$$y_{i+1}^* = y_i + h[x_i + y_i] \quad (19)$$

$$y_{i+1} = y_i + \frac{h}{2}[x_i + y_i + x_{i+1} + y_{i+1}^*] \quad (20)$$

Modified Euler Method

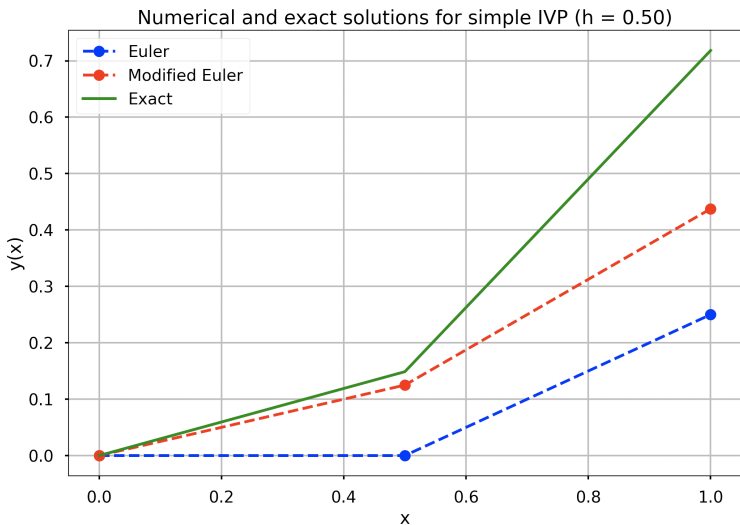
Large Step Size: $h = 0.5$

X coordinate	Euler Method	Modified Euler	Exact $y(x)$
0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
5.000000e-01	0.000000e+00	1.250000e-01	1.4872127e-01
1.000000e+00	2.500000e-01	4.375000e-01	7.1828183e-01

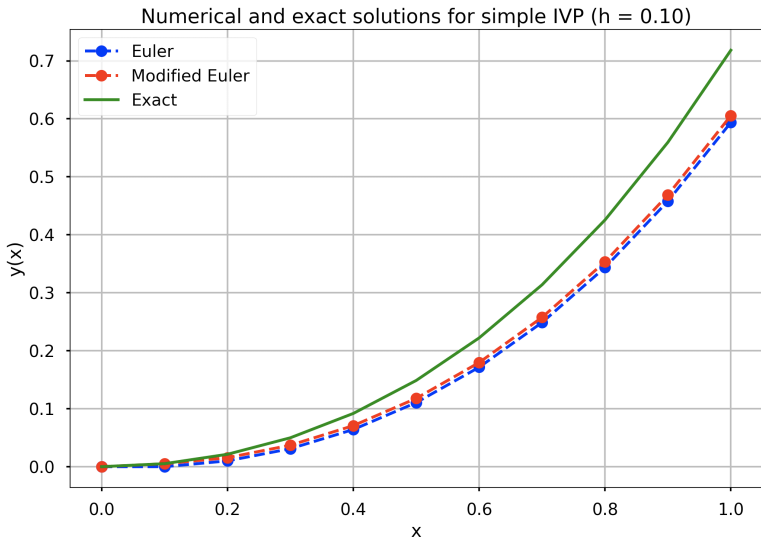
Small Step Size: $h = 0.1$

X coordinate	Euler Method	Modified Euler	Exact $y(x)$
0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
1.000000e-01	0.000000e+00	5.000000e-03	5.1709181e-03
2.000000e-01	1.000000e-02	1.550000e-02	2.1402758e-02
3.000000e-01	3.100000e-02	3.705000e-02	4.9858808e-02
4.000000e-01	6.410000e-02	7.075500e-02	9.1824698e-02
.....			
9.000000e-01	4.5794769e-01	4.6866564e-01	5.5960311e-01
1.000000e+00	5.9374246e-01	6.0553220e-01	7.1828183e-01

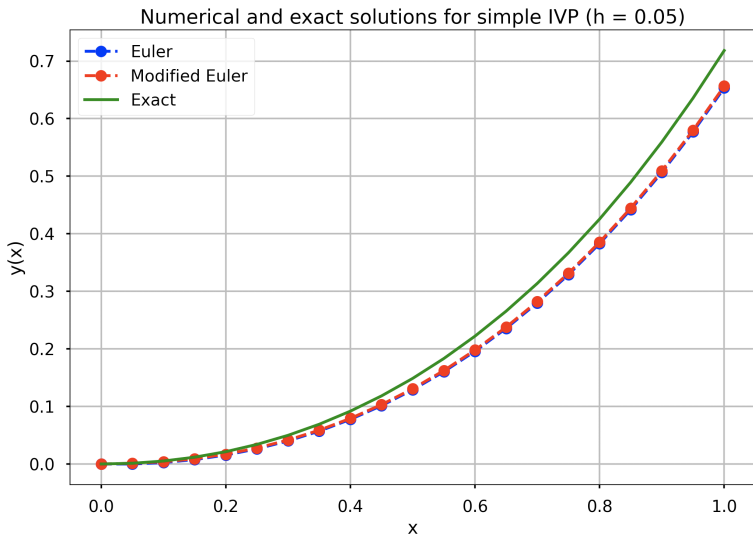
Modified Euler Method



Modified Euler Method



Modified Euler Method



Python Code Listings

Code 1: Modified Euler Method

```

1  # =====
2  # TestOdeModifiedEuler01.py: Use Modified Euler Method to solve IVP.
3  #
4  # Written By: Mark Austin                                     July 2023
5  # =====
6
7  import math
8  import numpy as np
9  import matplotlib.pyplot as plt
10
11 import LinearMatrixEquations as lme
12
13 # Define mathematical functions ...
14
15 def f1(x,y):
16     return x + y
17
18 def f2(x):
19     return math.exp(x) - x - 1.0
20
21 # main method ...
22
23 def main():
24     print("--- Enter TestOdeModifiedEuler01.main()      ... ");
25     print("--- ===== ... ");
26
27     # Define parameters

```

Code 1: Modified Euler Method

```

29     s0 = 0.0                                # <-- Initial Condition
30     h  = 0.50                                # <-- Step size
31     x  = np.arange(0, 1 + h, h) # <-- Create numerical grid on [0,1] ...
32
33     # Setup array for numerical and exact solutions ...
34
35     euler01 = np.zeros(len(x)) # <-- store euler method ...
36     euler02 = np.zeros(len(x)) # <-- store improved euler method ..
37     exact01 = np.zeros(len(x)) # <-- store exact solution ...
38
39     # Initialize array values ...
40
41     euler01[0] = s0                          # <-- Euler estimate ....
42     euler02[0] = s0                          # <-- Modified Euler estimate ...
43     exact01[0] = s0                          # <-- Exact solution ...
44
45     # Compute iterations of Euler and Modified Euler, Exact solution ...
46
47     for i in range(0, len(x) - 1):
48         euler01[i+1] = euler01[i] + h*f1( x[i], euler01[i] )
49         euler02[i+1] = euler01[i] + (h/2)*( f1( x[i], euler01[i] ) + f1( x[i+1], euler01[
50         exact01[i+1] = f2( x[i+1] )
51
52     # Print x, euler, improved euler, and y(x) vectors ...
53
54     lme.printvector("(x coordinate)", x )
55     lme.printvector("(euler method)",          euler01 )
56     lme.printvector("(improved euler method)", euler02 )

```

Code 1: Modified Euler Method

```

57     lme.printvector("y(x) (exact solution)",    exact01 )
58
59     # Plot output ...
60
61     plt.style.use('seaborn-poster')
62     plt.figure(figsize = (12, 8))
63     plt.plot( x, euler01,'bo--', label='Euler')
64     plt.plot( x, euler02,'ro--', label='Modified Euler')
65     plt.plot( x, exact01,   'g', label='Exact')
66     plt.title('Numerical and exact solutions for simple IVP (h = 0.50)')
67     plt.xlabel('x')
68     plt.ylabel('y(x)')
69     plt.grid()
70     plt.legend(loc='upper left')
71     plt.show()
72
73     print("--- ===== ... ");
74     print("--- Leave TestOdeModifiedEuler01.main() ... ");
75
76     # call the main method ...
77
78     main()

```