GNG1106
Fundamentals of Engineering Computation

Instructor: **Hitham Jleed**

University of Ottawa

Fall 2023 ~

Outline

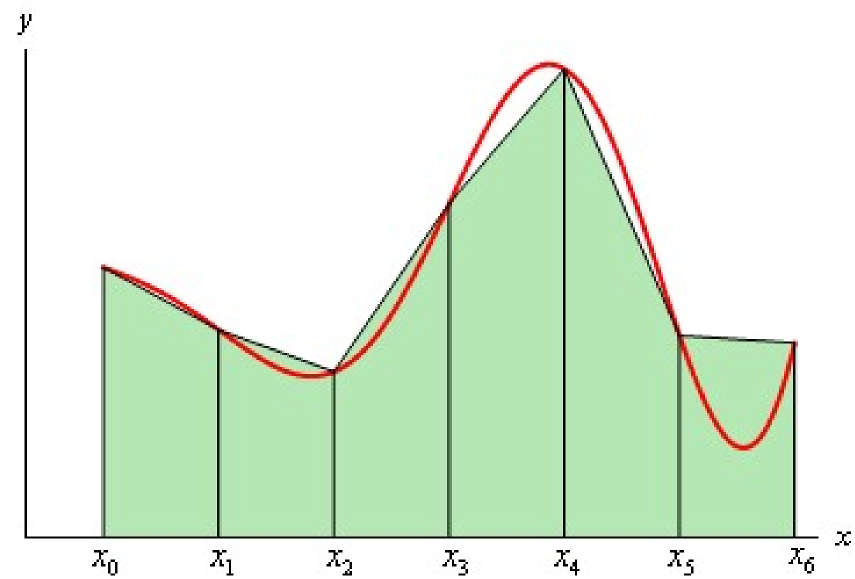
1. Teaching Evaluation
2. Numerical Method II: Trapezoidal Rule For Integration
3. Numerical Method III: Euler's Method For Solving DE

Outline

- Computing the integral of a function, say $f(x)$ is often needed.
- For many functions $f(x)$, there is no analytic solution.
- We now consider numerical computation of the integral

$$\int_a^b f(x)dx$$

- Idea: Cut the range $[a, b]$ into n equal subintervals and approximate the integral as the sum of the areas of n trapezoids.



Highlight

The area of a trapezoid is equal to:

$$\frac{\text{Base1} + \text{Base2}}{2} \times \text{Height}$$

- The smaller are the subintervals, the more accurate is the approximation.
- Larger range $[a, b]$ results in decreased approximation accuracy.

Algorithm

- Input: integration limits a and b, subinterval size Δ
- Output: integral value
- Let $n = \text{Round}(\frac{b-a}{\Delta})$. Cut $[a, b]$ into n subintervals $(a_0, a_1), (a_1, a_2), \ldots, (a_{n-1}, a_n)$, where $a_i - a_{i-1} = \Delta$. Then

$$\begin{aligned} \text{Intergal} &\approx \frac{f(a_0)+f(a_1)}{2}\Delta + \frac{f(a_1)+f(a_2)}{2}\Delta + \ldots + \frac{f(a_{n-1})+f(a_n)}{2}\Delta \\ &= \Delta \cdot \left(\frac{f(a_0)}{2} + \sum_{i=1}^{n-1} f(a_i) + \frac{f(a_n)}{2} \right) \end{aligned}$$

- Algorithm/Code: Do it yourself.

Outline

Differential Equations (DE)

- Engineering applications often need to solve differential equations.
- We consider the first-order DE of the following form.

$$\frac{dy}{dt} = f(y, t) \quad (1)$$

$$y(t_0) = y_0 \quad (2)$$

where

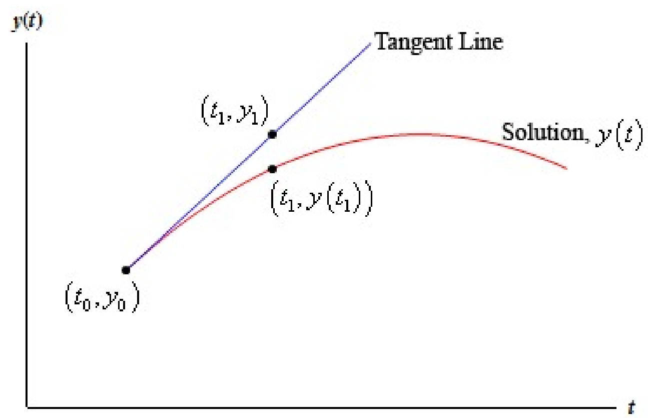
  - y, or written as $y(t)$, is an unknown function of time t.
  - f is a given function of y and t.
  - t_0 is a given initial time and y_0 is the given initial value of y at $t = t_0$.
- Solving this DE means finding function $y(t)$ for $t > t_0$.

- For example, when an object falls with air friction/resistance, we have

$$\begin{aligned} \frac{dv}{dt} &= \frac{mg - cv}{m} \\ v(t_0) &= v_0 \end{aligned}$$

  where m is object mass, g is gravitational acceleration, c is some constant, and v is the object falling speed.
- In some cases, analytic solutions are attainable using calculus.
- In other cases, analytic solutions are difficult to obtain or do not exist.
- We would like to solve such a DE numerically, i.e, instead of obtaining an analytic expression of the $y(t)$, we aim at obtaining a numerical solution: for a list of t values with $t > t_0$, we find their corresponding $y(t)$ values.

Euler's Method for Solving DE



$$\frac{dy}{dt} = \lim_{\Delta \to 0} \frac{y(t+\Delta) - y(t)}{\Delta}$$
$$\approx \frac{y(t+\Delta) - y(t)}{\Delta} \quad \text{(for small } \Delta)$$
$$f(y,t) \approx \frac{y(t+\Delta) - y(t)}{\Delta} \quad \text{(for small } \Delta)$$

Highlight

If we select a set of equally space time points $\{t_1, t_2, \ldots, t_n\}$ with $t_i - t_{i-1} = \Delta$, we have

$$y(t_i) \approx y(t_{i-1}) + f(y(t_{i-1}), t_{i-1})\Delta$$

The Algorithm

- Input: t_0, y_0, Δ, n
- Output: $(t_1, t_2, \ldots, t_n)$ and $(y_1, y_2, \ldots, y_n)$
  (y_i is the short-hand notation for $y(t_i)$).
- Algorithm (not the code):

  $t = t_0$;

  $y = y_0$;

  for (i=0; i<n; i++)

  {

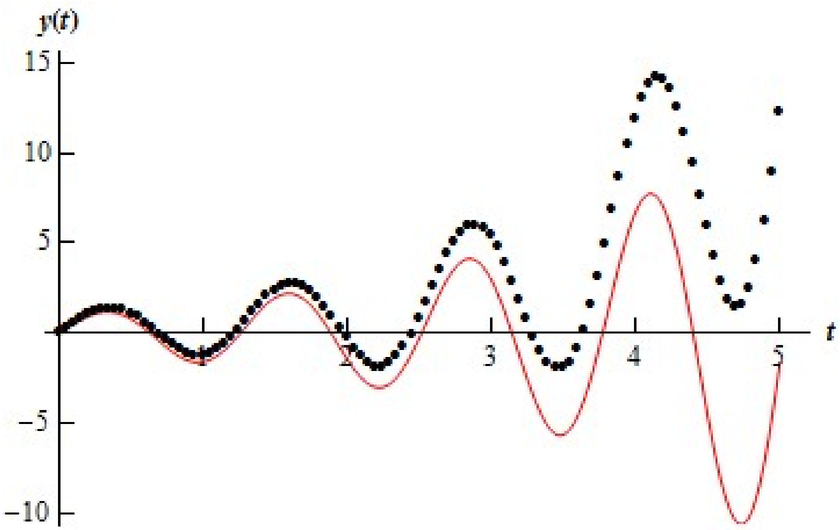
  $\quad y = y + f(y, t)\Delta$;

  $\quad t = t + \Delta$;

  $\quad$ print (t, y, "\n");

  }

$$\frac{dy}{dt} = y - \frac{1}{2}e^{t/2}\sin(5t) + 5e^{t/2}\cos(5t)$$



- The smaller the step size Δ, the more accurate is the approximation.
- As t increases, the approximation usually gets less accurate.

In-Class Exercise: