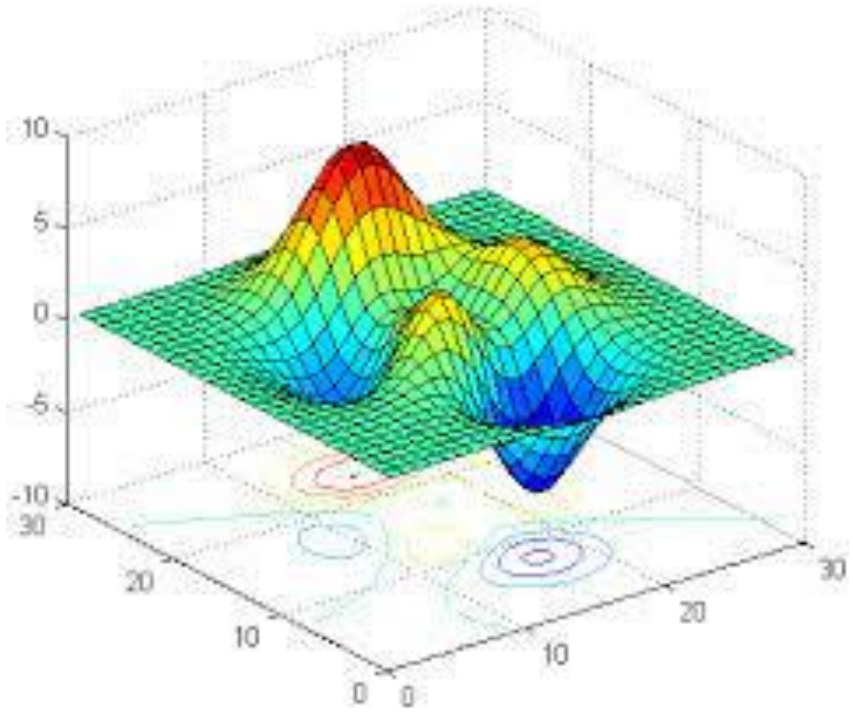
Khemis Miliana University
Faculty of Sciences of Matter and Computer Science
Department of Physics

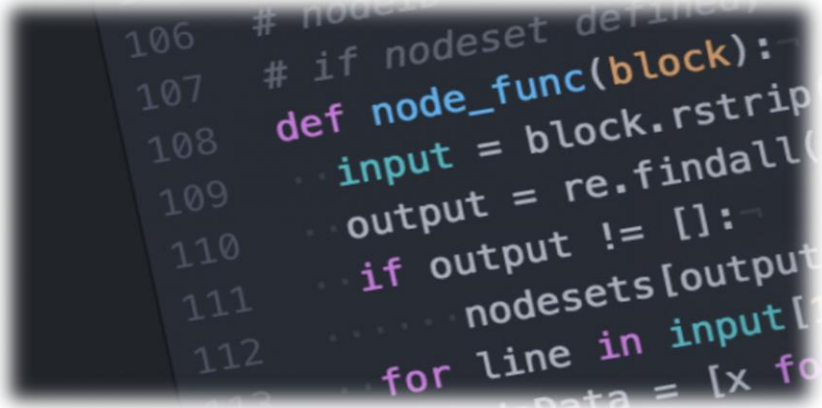


Numerical Methods & Scientific Programming

Dr. Salah-Eddine BENTRIDI
s.bentridi@univ-dbkm.dz
Univ. Khemis-Miliana

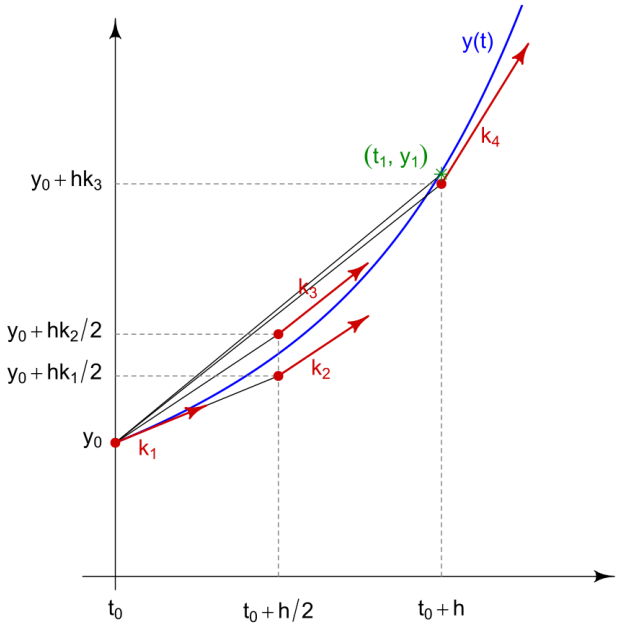
Content of the program

- Chapter 01: ***Initiation to a programming language (Python)***
  - ***Hands-on-Pythn; Basics of Python***
- Chapter 02: ***Numerical Integration***
  - ***Trapezoidal rule; Simpson's method***
- Chapter 03: ***Numerical Solution of equations – Root finding***
  - ***Bisection method; Newton's Method***
- Chapter 04: ***Numerical resolution of differential equations***
  - ***Euler's method; Runge-Kutta method***
- Chapter 05: ***Numerical resolution of linear equations system***
  - ***Gauss method, Gauss-Seidel method***



Chapter 03: Numerical resolution of differential equations

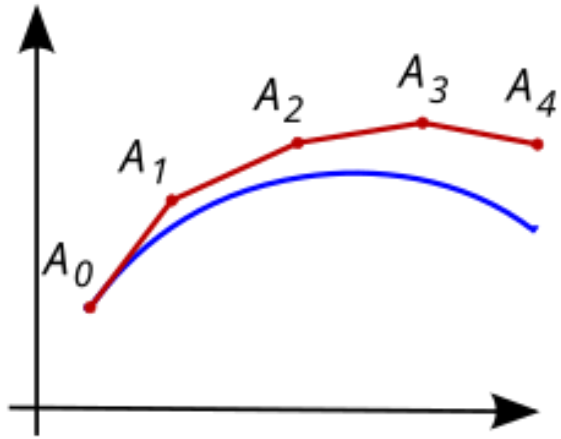
Euler's method and Runge-Kutta method



Dr. Salah-Eddine BENTRIDI

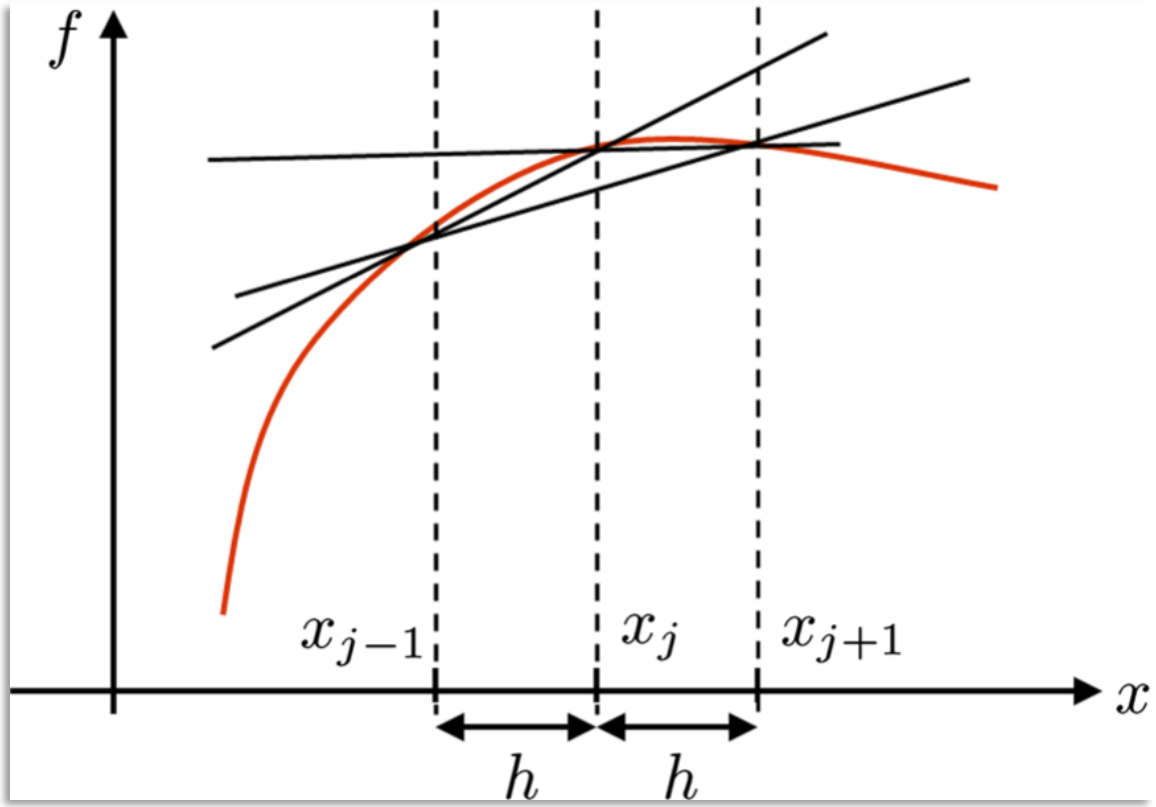
s.bentridi@univ-dbkm.dz

Khemis-Miliana University



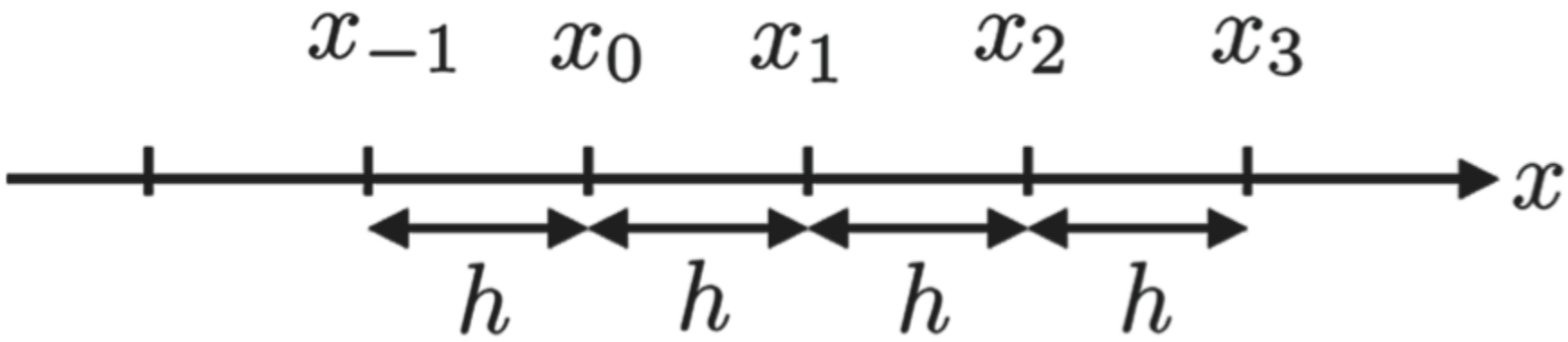
Outline

- Numerical Differentiation
- ODE statement problem
- Euler Method
- Runge-Kutta Method



I. Numerical Differentiation

A numerical grid is an evenly spaced set of points over the domain of a function (i.e., the independent variable), over some interval. The spacing or step size of a numerical grid is the distance between adjacent points on the grid. If x is a numerical grid, then x_j is the j^{th} point in the numerical grid and h is the spacing between x_{j-1} and x_j.



I. Numerical Differentiation

- ***Whether f is an analytic function or a discrete representation of one, we would like to derive methods of approximating the derivative of f over a numerical grid and determine their accuracy.***
- ***The derivative $f'(x)$ of a function $f(x)$ at the point $x = a$ is defined as:***

$$f'(a) = \lim_{x \to a} \frac{f(x) - f(a)}{x - a}$$

- **Geometrically, the derivative at $x = a$ represents the slope at this point.**

I. Numerical Differentiation

Finite Difference Approximating Derivatives

- *In finite difference approximations of this slope, we can use values of the function in the neighborhood of the point $x = a$ to achieve the goal.*
- *There are various finite difference formulas used in different applications, and three of these, where the derivative is calculated using the values of two points, are presented below:*

The forward difference is to estimate the slope of the function at x_j using the line that connects $(x_j, f(x_j))$ and $(x_{j+1}, f(x_{j+1}))$:

$$f'(x_j) = \frac{f(x_{j+1}) - f(x_j)}{x_{j+1} - x_j}$$

The backward difference is to estimate the slope of the function at x_j using the line that connects $(x_{j-1}, f(x_{j-1}))$ and $(x_j, f(x_j))$:

$$f'(x_j) = \frac{f(x_j) - f(x_{j-1})}{x_j - x_{j-1}}$$

The central difference is to estimate the slope of the function at x_j using the line that connects $(x_{j-1}, f(x_{j-1}))$ and $(x_{j+1}, f(x_{j+1}))$:

$$f'(x_j) = \frac{f(x_{j+1}) - f(x_{j-1})}{x_{j+1} - x_{j-1}}$$

I. Numerical Differentiation

Finite Difference Approximating Derivatives

The forward difference :

$$f'(x_j) = \frac{f(x_{j+1}) - f(x_j)}{x_{j+1} - x_j}$$

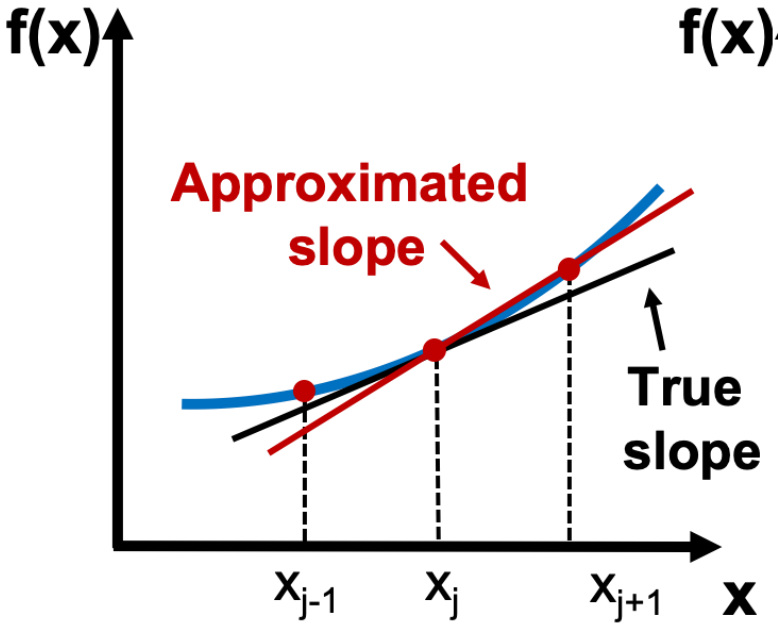
The backward difference :

$$f'(x_j) = \frac{f(x_j) - f(x_{j-1})}{x_j - x_{j-1}}$$

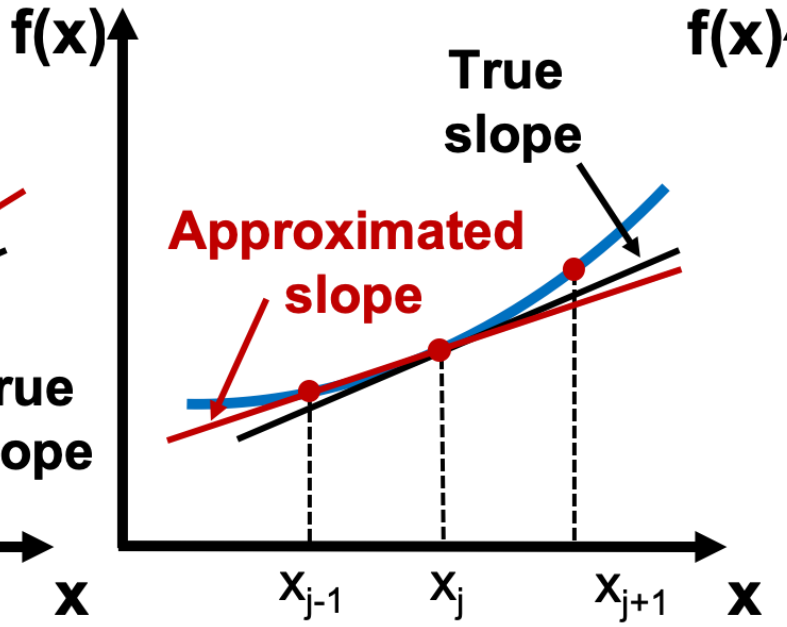
The central difference :

$$f'(x_j) = \frac{f(x_{j+1}) - f(x_{j-1})}{x_{j+1} - x_{j-1}}$$

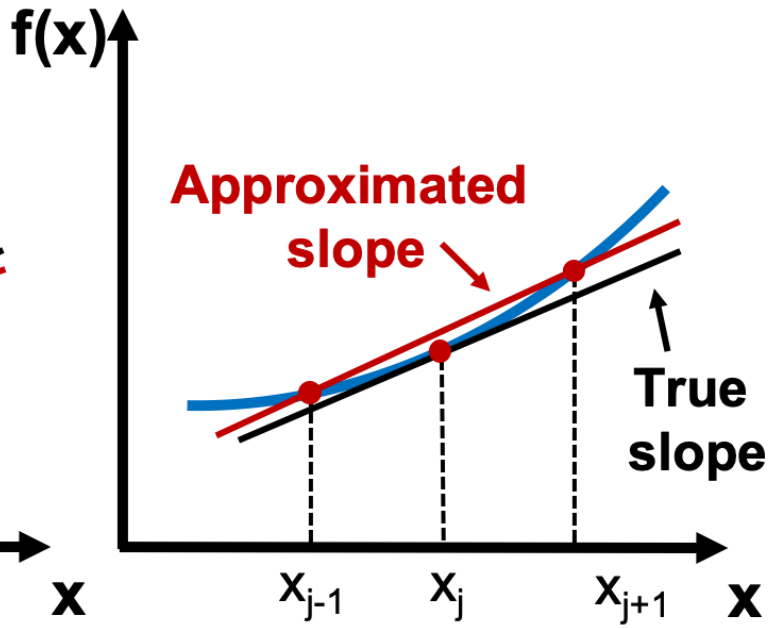
Forward difference



Backward difference



Central difference



I. Numerical Differentiation

Finite Difference Approximating Derivatives

The forward difference :

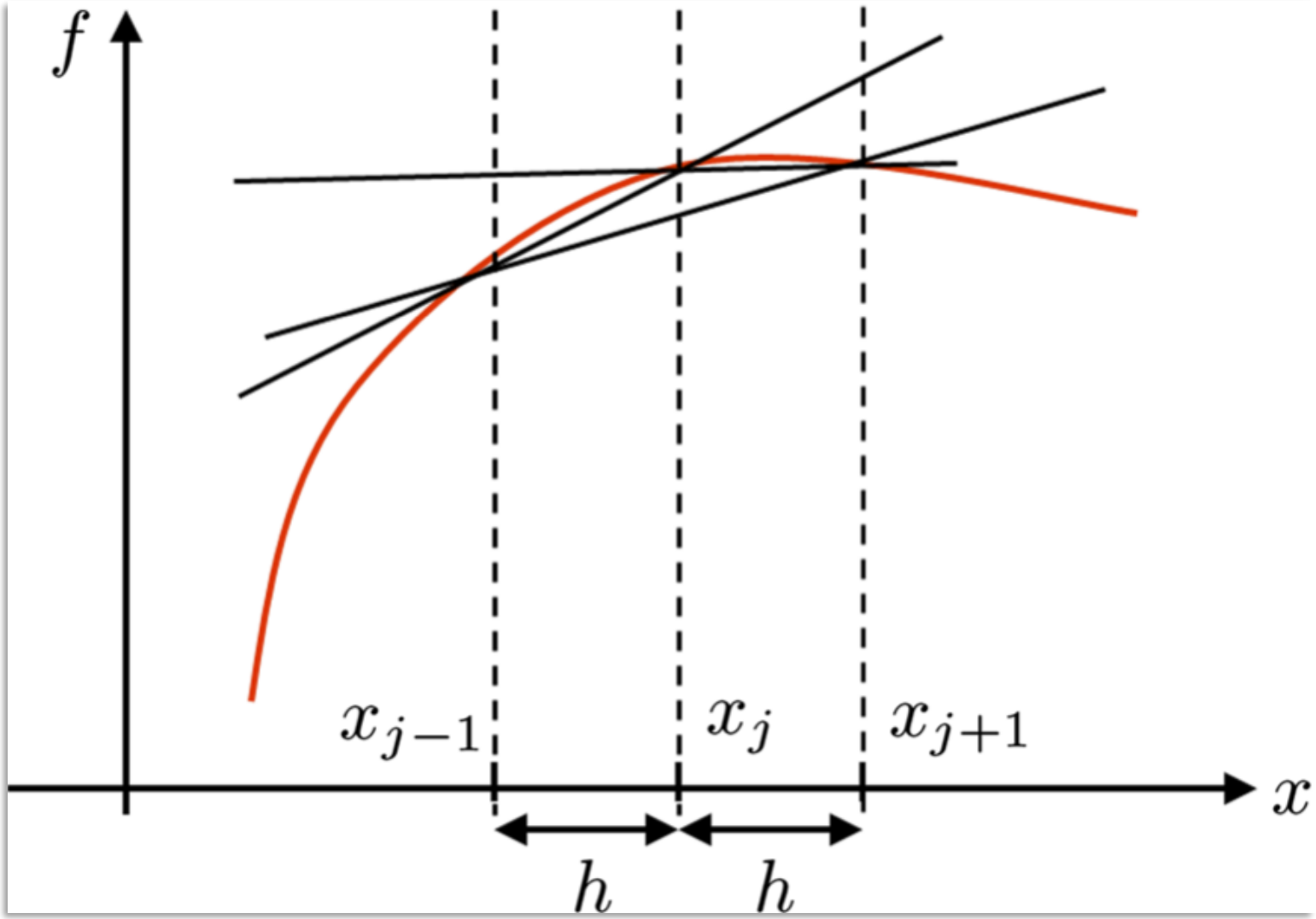
$$f'(x_j) = \frac{f(x_{j+1}) - f(x_j)}{h}$$

The backward difference :

$$f'(x_j) = \frac{f(x_j) - f(x_{j-1})}{h}$$

The central difference :

$$f'(x_j) = \frac{f(x_{j+1}) - f(x_{j-1})}{2h}$$

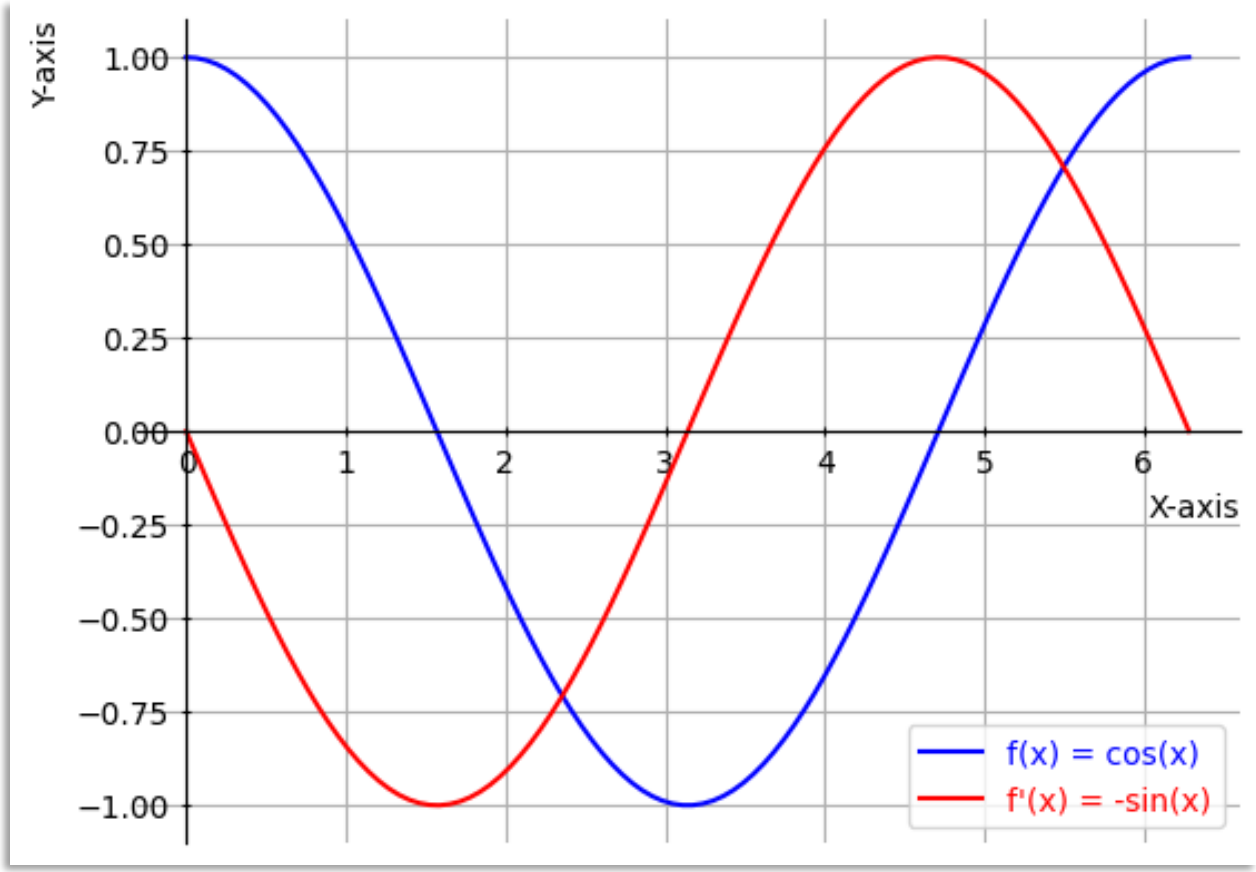


I. Numerical Differentiation

Finite Difference Approximating Derivatives

Example:

Consider the function $f(x) = \cos(x)$. ***We know the derivative of*** $\cos(x)$ ***is*** $-\sin(x)$. ***Although in practice we may not know the underlying function we are finding the derivative for, we use the simple example to illustrate the aforementioned numerical differentiation methods and their accuracy. The following code computes the derivatives numerically.***



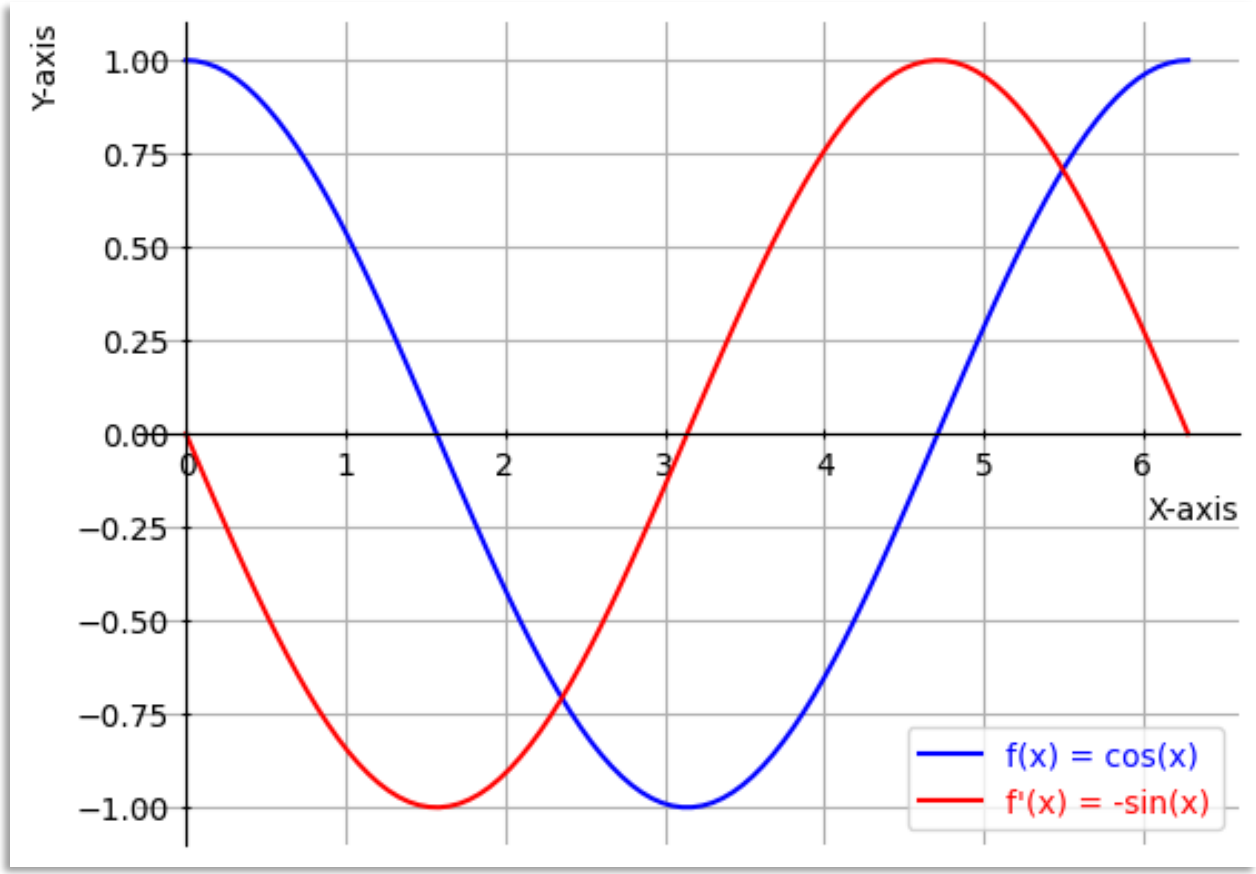
I. Numerical Differentiation

Finite Difference Approximating Derivatives

Example:

1. *Let's apply the first method "Forward difference" on* $f(x) = \cos(x)$.
2. *Consider the short interval* $[-h, \pi]$ *divided on ten small spacing* $h = \frac{\pi}{10}$
3. *Consequently, we use:*

$$f'(x_j) = \frac{f(x_{j+1}) - f(x_j)}{h}$$



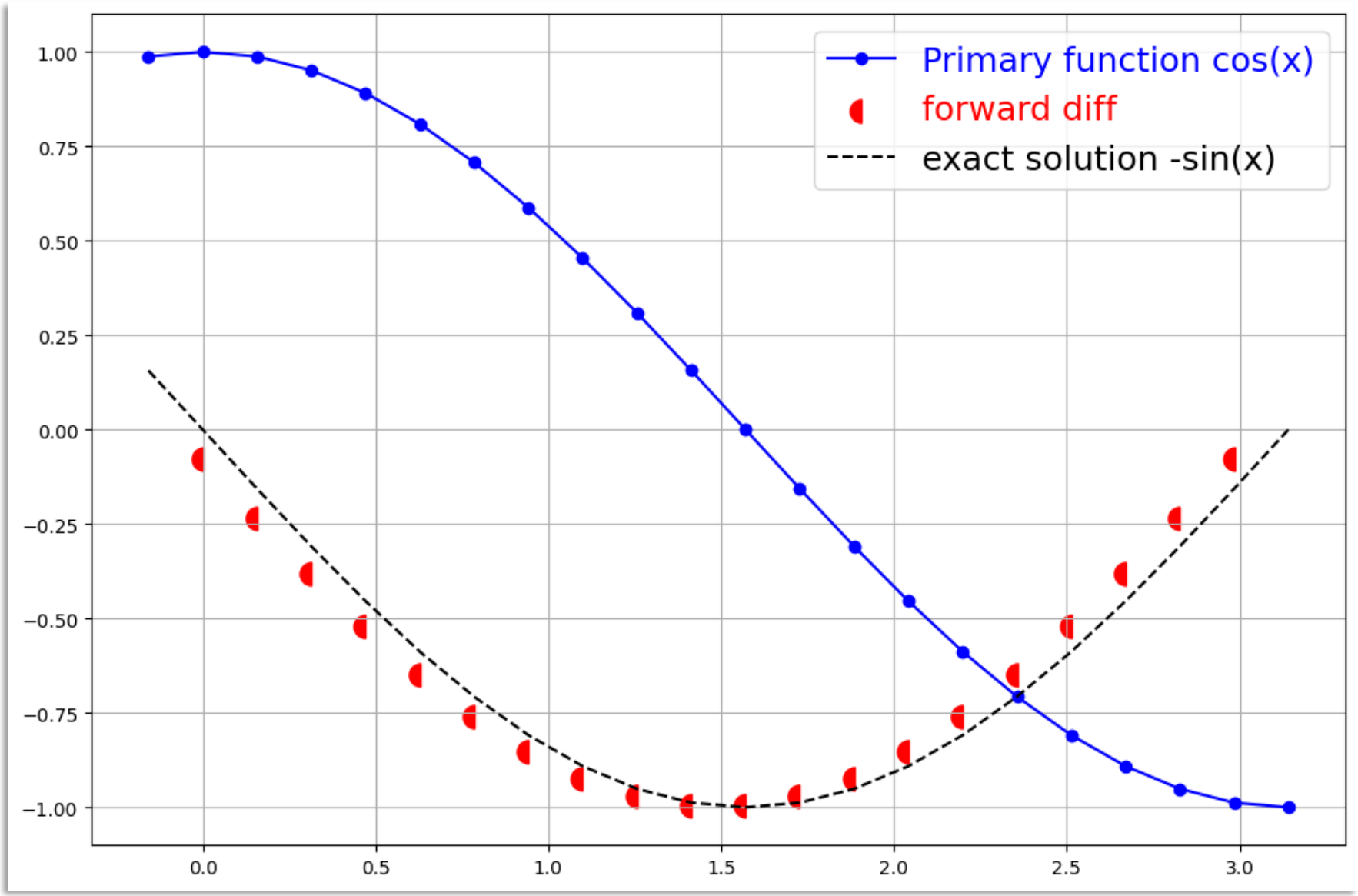
I. Numerical Differentiation

Finite Difference Approximating Derivatives

```
x = [-0.15707963  0.          0.15707963  0.31415927  0.4712389   0.62831853
  0.78539816  0.9424778   1.09955743  1.25663706  1.41371669  1.57079633
  1.72787596  1.88495559  2.04203522  2.19911486  2.35619449  2.51327412
  2.67035376  2.82743339  2.98451302  3.14159265]
x_i = [0.          0.15707963 0.31415927 0.4712389  0.62831853 0.78539816
 0.9424778  1.09955743 1.25663706 1.41371669 1.57079633 1.72787596
 1.88495559 2.04203522 2.19911486 2.35619449 2.51327412 2.67035376
 2.82743339 2.98451302]
forward = [-0.07837846 -0.23320544 -0.38229012 -0.52196156 -0.64878057 -0.75962445
 -0.85176385 -0.92293    -0.97137055 -0.99589274 -0.99589274 -0.97137055
 -0.92293    -0.85176385 -0.75962445 -0.64878057 -0.52196156 -0.38229012
 -0.23320544 -0.07837846]
```

I. Numerical Differentiation

Finite Difference Approximating Derivatives



I. Numerical Differentiation

Finite Difference Approximating Derivatives

Exam

4. Re ***ame function***

$f(x)$

```
x = [-0.15707963  0.          0.15707963  0.31415927  0.4712389   0.62831853
  0.78539816  0.9424778   1.09955743  1.25663706  1.41371669  1.57079633
  1.72787596  1.88495559  2.04203522  2.19911486  2.35619449  2.51327412
  2.67035376  2.82743339  2.98451302  3.14159265]
x_i = [0.         0.15707963 0.31415927 0.4712389  0.62831853 0.78539816
 0.9424778  1.09955743 1.25663706 1.41371669 1.57079633 1.72787596
 1.88495559 2.04203522 2.19911486 2.35619449 2.51327412 2.67035376
 2.82743339 2.98451302]
forward = [-0.07837846 -0.23320544 -0.38229012 -0.52196156 -0.64878057 -0.75962445
 -0.85176385 -0.92293    -0.97137055 -0.99589274 -0.99589274 -0.97137055
 -0.92293    -0.85176385 -0.75962445 -0.64878057 -0.52196156 -0.38229012
 -0.23320544 -0.07837846]
backward = [ 0.07837846 -0.07837846 -0.23320544 -0.38229012 -0.52196156 -0.64878057
 -0.75962445 -0.85176385 -0.92293    -0.97137055 -0.99589274 -0.99589274
 -0.97137055 -0.92293    -0.85176385 -0.75962445 -0.64878057 -0.52196156
 -0.38229012 -0.23320544]
central = [ 0.         -0.15579195 -0.30774778 -0.45212584 -0.58537106 -0.70420251
 -0.80569415 -0.88734692 -0.94715028 -0.98363164 -0.99589274 -0.98363164
 -0.94715028 -0.88734692 -0.80569415 -0.70420251 -0.58537106 -0.45212584
 -0.30774778 -0.15579195]
```

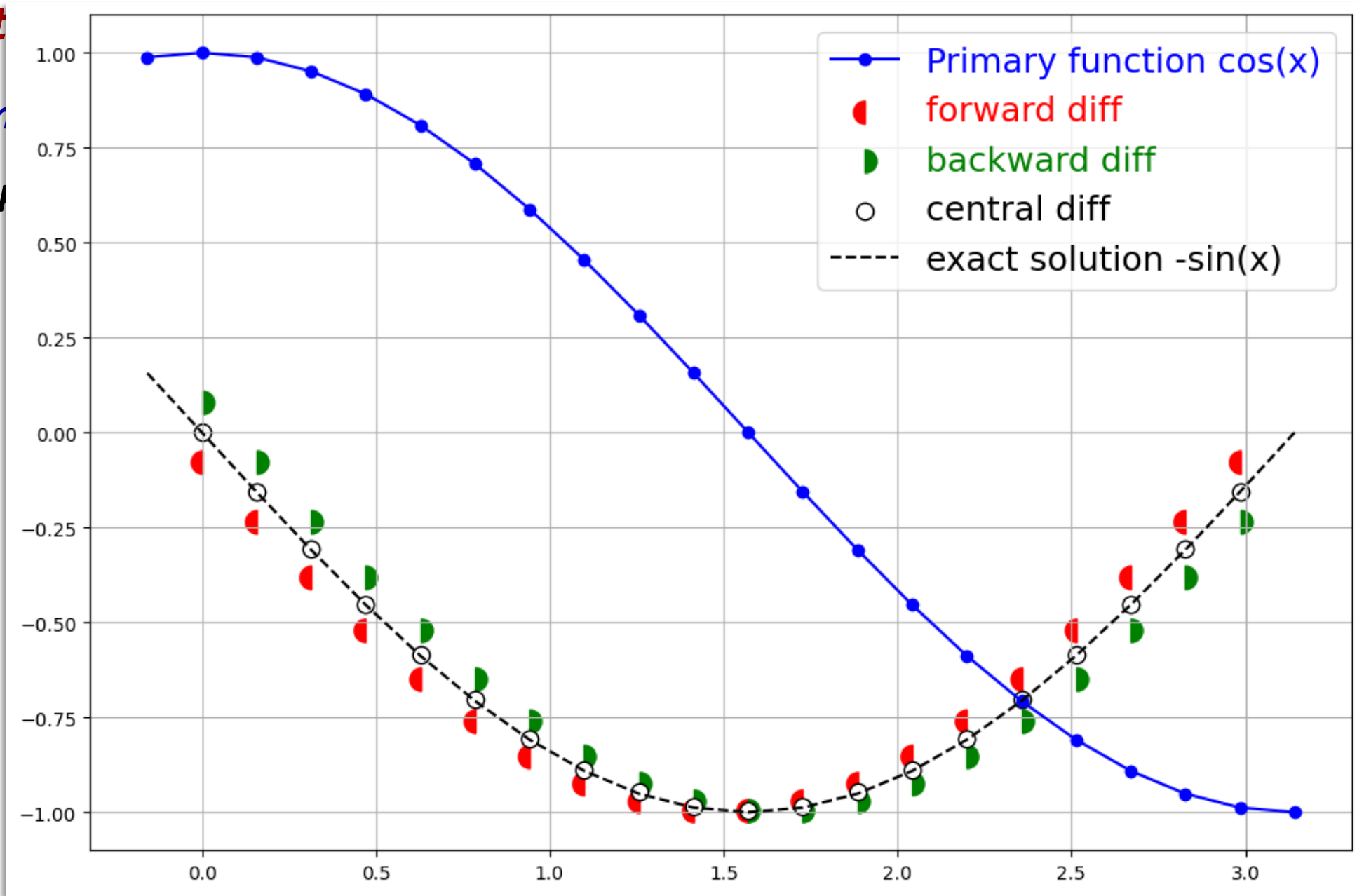
I. Numerical Differentiation

Finit

Exam

4. Re ***ume function***

$f(x)$



II. ODE statement problem

A differential equation is a relationship between a function, $f(x)$, its independent variable, x, and any number of its derivatives. An ordinary differential equation or ODE is a differential equation where the independent variable, and therefore also the derivatives, is in one dimension. We assume that an ODE can be written

$$F\left(x, f(x), \frac{df(x)}{dx}, \frac{d^2f(x)}{dx^2}, \ldots, \frac{d^{n-1}f(x)}{dx^{n-1}}\right) = \frac{d^nf(x)}{dx^n}$$

where F is an arbitrary function that incorporates one or all of the input arguments, and n is the order of the differential equation. This equation is referred to as an n^{th} order ODE.

II. ODE statement problem

A common set of known values for an ODE solution is the initial value. For an ODE of order n *, the initial value is a known value for the* 0^{th} *to* $(n-1)^{th}$ *derivatives at:* $x = 0, f(0), f'(0), f''(0), \dots, f^{(n)}(0)$. *For a certain class of ordinary differential equations, the initial value is sufficient to find a unique particular solution. Finding a solution to an ODE given an initial value is called the initial value problem.*

Many numerical methods for solving initial value problems are designed specifically to solve first-order differential equations. To make these solvers useful for solving higher order differential equations, we must often reduce the order of the differential equation to first order:

$$dS(t)/dt = F(t, S(t))$$

III. Euler method

Let $\frac{dS(t)}{dt} = F(t,\ S(t))$ *be an explicitly defined first order ODE. That is,* F *is a function that returns the derivative, or change, of a state given a time and state value. Also, let* t *be a numerical grid of the interval* $[t_0, t_f]$ *with spacing* h*. Without loss of generality, we assume that* $t_0 = 0$ *, and that* $t_f = Nh$ *for some positive integer,* N*.*

The linear approximation of $S(t)$ *around* t_j *at* t_{j+1} *is*

$$S(t_{j+1}) = S(t_j) + (t_{j+1} - t_j)\frac{dS(t_j)}{dt}$$

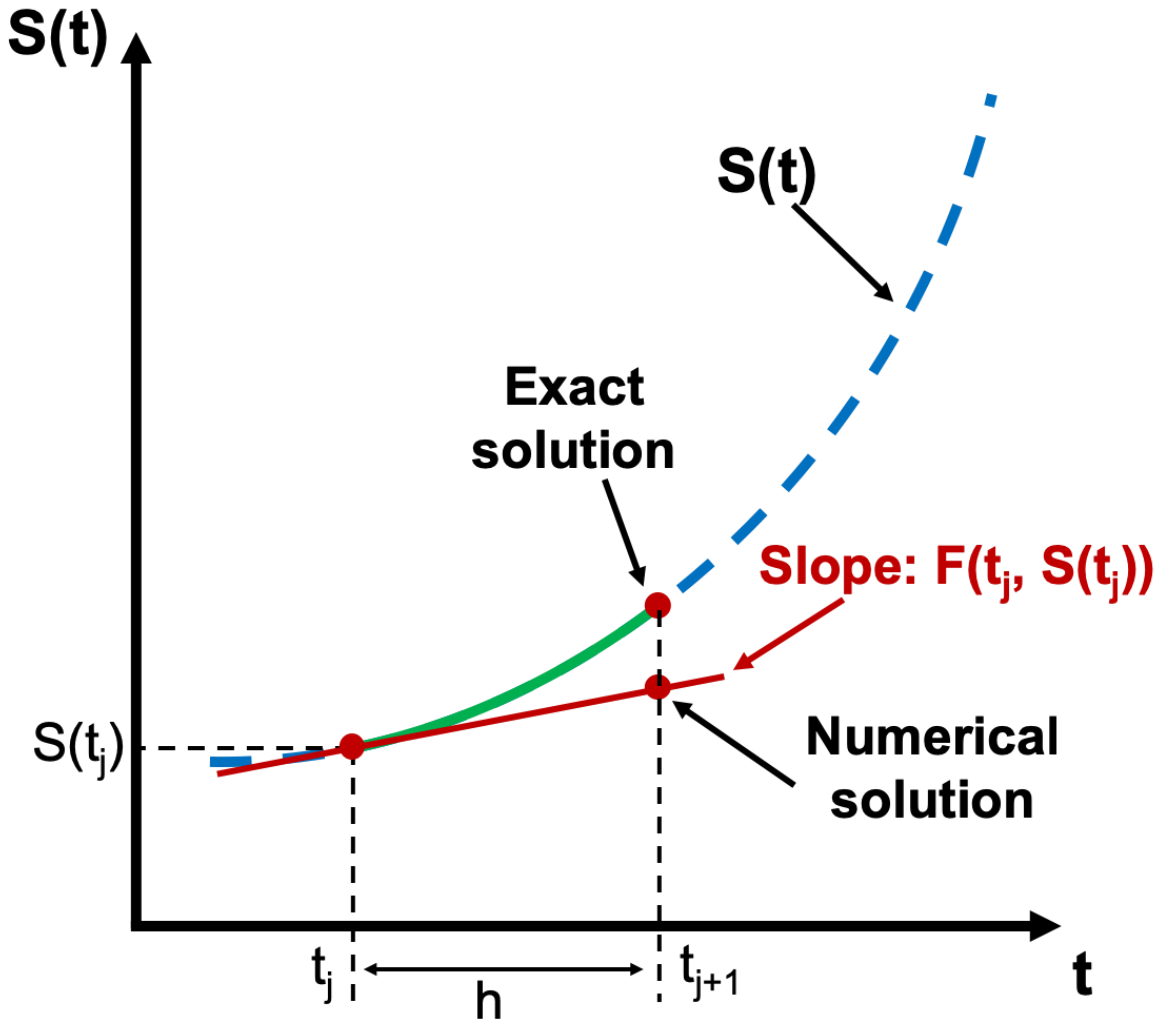
which can also be written

$$S(t_{j+1}) = S(t_j) + hF(t_j, S(t_j))$$

This formula is called the Explicit Euler Formula, and it allows us to compute an approximation for the state at $S(t_{j+1})$ *given the state at* $S(t_j)$*.*

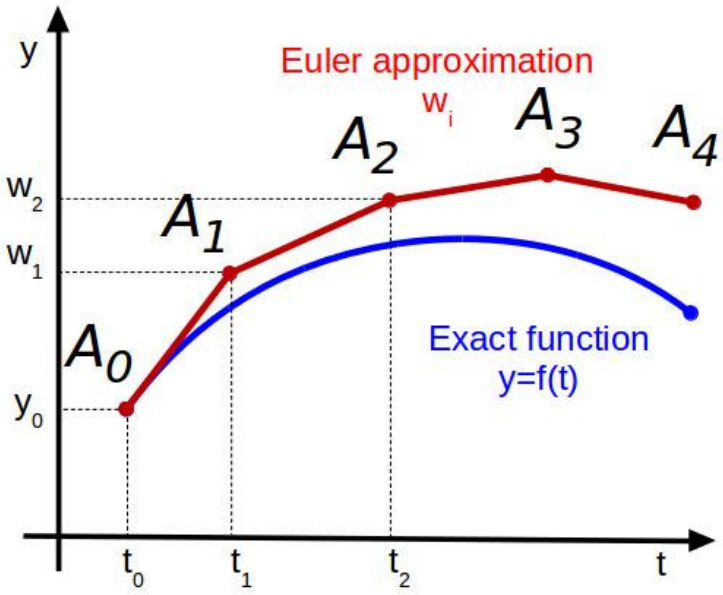
III. Euler method

Starting from an initial value of $S_0 = S(t_0)$, *we can use this formula to integrate the states up to* $S(t_f)$; *these* $S(t)$ *values are then an approximation for the solution of the differential equation. The Explicit Euler formula is the simplest and most intuitive method for solving initial value problems. At any state* $(t_j, S(t_j))$ *it uses* F *at that state to "point" toward the next state and then moves in that direction a distance of* h.



III. Euler method

Assume we are given a function $F(t, S(t))$ that computes $\frac{dS(t)}{dt}$, a numerical grid, t, of the interval, $[t_0, t_f]$, and an initial state value $S_0 = S(t_0)$. We can compute $S(t_j)$ for every t_j in t using the following steps.



1. Store $S_0 = S(t_0)$ in an array, S.
2. Compute $S(t_1) = S_0 + hF(t_0, S_0)$.
3. Store $S_1 = S(t_1)$ in S.
4. Compute $S(t_2) = S_1 + hF(t_1, S_1)$.
5. Store $S_2 = S(t_1)$ in S.
6. $\cdots$
7. Compute $S(t_f) = S_{f-1} + hF(t_{f-1}, S_{f-1})$.
8. Store $S_f = S(t_f)$ in S.
9. S is an approximation of the solution to the initial value problem.

When using a method with this structure, we say the method integrates the solution of the ODE.

IV. Runge-Kutta method

Runge Kutta (RK) methods are one of the most widely used methods for solving ODEs.

Recall that the Euler method uses the first two terms in Taylor series to approximate the numerical integration, which is linear: $S(t_{j+1}) = S(t_j + h) = S(t_j) + h.S'(t_j)$.

We can greatly improve the accuracy of numerical integration if we keep more terms of the series in:

$$S(t_{j+1}) = S(t_j + h) = S(t_j) + S'(t_j).h + \frac{1}{2!}S''(t_j).h^2 + \cdots + \frac{1}{n!}S^{(n)}(t_j).h^n$$

In order to get this more accurate solution, we need to derive the expressions of $S''(t_j), S'''(t_j), \cdots, S^{(n)}(t_j)$. *This extra work can be avoided using the RK methods, which is based on truncated Taylor series, but not require computation of these higher derivatives.*

IV. Runge-Kutta method: *2nd order RK method*

Let us first derive the second order RK method. Let $\frac{dS(t)}{dt} = F(t, S(t))$***, then we can assume an integration formula the form of***

$$S(t+h) = S(t) + c_1 F\big(t, S(t)\big).h + c_2 F\big[t + ph, S(t) + qhF\big(t, S(t)\big)\big].h \quad (*)$$

We can attempt to find these parameters c_1, c_2, p, q ***by matching the above equation to the second-order Taylor series, which gives us***

$$S(t+h) = S(t) + S'(t)h + \frac{1}{2!}S''(t_j).h^2 = S(t) + F\big(t, S(t)\big)h + \frac{1}{2!}F'\big(t, S(t)\big).h^2 \ (**)$$

Noting that $F'\big(t, s(t)\big) = \frac{\partial F}{\partial t} + \frac{\partial F}{\partial S}.\frac{\partial S}{\partial t} = \frac{\partial F}{\partial t} + \frac{\partial F}{\partial S}F$

Therefore, the previous equation $(**)$ ***can be written as:***

$$S(t+h) = S + Fh + \frac{1}{2!}\left(\frac{\partial F}{\partial t} + \frac{\partial F}{\partial S}F\right)h^2$$

IV. Runge-Kutta method: *2nd order RK method*

Accordingly, in equation $(*)$ we can rewrite the last term by applying Taylor series in several variables, which gives us:

$$F[t+ph, S+qhF)] = F + \frac{\partial F}{\partial t}ph + qh\frac{\partial F}{\partial S}F$$

Thus equation $(*)$ becomes:

$$S(t+h) = S + (c_1 + c_2)Fh + c_1\left[\frac{\partial F}{\partial t}p + q\frac{\partial F}{\partial S}F\right]h^2$$

Comparing last equations, will lead to:

$$c_1 + c_2 = 1, c_2 p = \frac{1}{2}, c_2 q = \frac{1}{2}$$

Since we have four unknowns and only three equations, we can assign any value to one of the parameters and get the rest of the parameters. One popular choice is:

$$c_1 = \frac{1}{2}, c_2 = \frac{1}{2}, p = 1, q = 1$$

IV. Runge-Kutta method: *2nd order RK method*

We can also define:

$$k_1 = F(tj, S(tj))$$

$$k_2 = F(t_j + ph, S(t_j) + qhk_1)$$

Where we will have:

$$S(t_{j+1}) = S(t_j) + \frac{1}{2}(k_1 + k_2)h$$

This formula is known as 2nd order Runge-Kutta method.

IV. Runge-Kutta method: *4th order RK method*

A classical method for integrating ODEs with a high order of accuracy is the Fourth Order Runge Kutta (RK4) method. It is obtained from the Taylor series using similar approach we just discussed in the second-order method. This method uses four points k_1, k_2, k_3, and k_4. A weighted average of these is used to produce the approximation of the solution. The formula is as follows.

$$\begin{cases} k_1 = F\big(tj, S(tj)\big) \\ k_2 = F(t_j + \dfrac{h}{2}, S(t_j) + \dfrac{1}{2}hk_1) \\ k_3 = F(t_j + \dfrac{h}{2}, S(t_j) + \dfrac{1}{2}hk_2) \\ k_4 = F\big(t_j + h, S\big(t_j\big) + hk_3\big) \end{cases}$$

Therefore, we will have:

$$S(t_{j+1}) = S(t_j) + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

As indicated by its name, the RK4 method is fourth-order accurate, or $O(h4)$.