# PHY425: Computational Methods in Physics II

January – April 2020

Vishal Bhardwaj

*Monday, Tuesday 10-10:55 am  AB1-1A*
*Thursday 2-4:55 pm CP-2*

# *Pre-requisites*

Knowledge of programming in **FORTRAN** or **C/C++** or ***python***
Computational Methods in Physics I
or
Numerical Methods in Chemistry

Aim of the course:
To make you more comfortable with the computational techniques

First and only useful advice (I can give or I ever got)

**K.I.S.S. → Keep it Simple and Stupid**

Use any language (C++ or python) for coding, as long as you are comfortable with it.

**PHY425: Computational methods in physics II**

**[Cr:4, Lc:3, Tt:0, Lb:3]**

*Knowledge of the content of* **PHY422** *is essential to follow this course.*

**Course Outline**

- Solution of Ordinary Differential Equations, Euler's method, Runge-Kutta methods, Initial and Boundary Value Problems.
- Fast Fourier Transforms, relation with Fourier series and Fourier transforms.
- Partial differential equations: Diffusion equation, Wave equation, Poisson equation. Finite element and relaxation methods.
- Parallel computing: Decomposition of problems, functional, data and domain decomposition. Shared memory and distributed memory parallelization. Optimization with co-processors and GP-GPUs.
- Topics in Numerical Simulations (At least two topics to be covered):
  1. Many particle simulations with short range interactions.
  2. Many particle simulations with long range interactions.
  3. Computational Fluid Dynamics.
  4. Simulations of spin systems.
  5. Simulations of quantum mechanical scattering.
  6. Parameter estimation for $n > 3$ unknown parameters from experimental/observational data.

**Recommended Reading**

- H. M. Antia, *Numerical Methods For Scientists And Engineers*, 02nd edition, Birkhauser Basel (2002).
- *Numerical Recipes in C: The Art of Scientific Computing*, W. H. Press, S. A. Teukolsky, W. T. Vellerling and B. P. Flannery, Cambridge University Press (1992).
- *Programming Massively Parallel Processors: A Hands-on Approach*, by David B. Kirk, Wen-mei W. Hwu, Publisher: Morgan Kaufmann; 3rd edition (2016)
- *Computational Physics, J. M. Thijssen, Cambridge University Press (1999)*

I will be following random books, online tutorials.
You can follow any book.

If time permits, will try to do one simple TensorFlow NN example *[but don't think]*

# Evaluation

| Type | Duration | Total weightage |
|---|---|---|
| Assignments | Deadline | 20 |
| Lab quiz-1 | 2 hours (after 1st Mid Sem) | 10 |
| Lab quiz-2 | 2 hours (after 23 March) | 10 |
| Mid-sem exam (theory) | 1 hour | 15 |
| End-sem exam (theory) | 2 hour | 25 |
| End-sem (Lab) | 1 hour | 20 |

# *Some tips to be a good Programmer*

*Few instructions, bare minimum*

- Simple and Stupid
- Be careful and try to be correct.
- Try to write algorithm or flow-chart before writing a code.
- Build your code in steps and test step by step.
- Include comments, useful when you are dealing with 10K line of codes.
- Variable names should be reasonable:
  Eg. Arrays, Lists : *Plular names*, single variable: *singular name*
- Avoid infite loops at any cost.
- Test your code using few cases.
- Display intermediate results.
- Include useful Warnings and Error messages.
- Precision: Use single/double precision properly.

Thumb rule:
*Don't use 1000cc when 60cc can do same job*

Use a good editor to edit your code.
I like Emacs: for C++, python, latex
One can use it inside terminal and also GUI.

http://www.gnu.org/software/emacs/

# Simple mathematical modeling

A mathematical model can be broadly defined as formulation or equation that expressses the essential features of a physical system or process in mathematical terms.

$$\text{Dependent variable} = f\,(\text{independent variable}, \text{parameters}, \text{forcing functions})$$

*Dependent variable* is characteristice that usualy reflects the behaviour or state of the system.
*Independent variables* are usually dimensions, such as time and space, along which system behavior is being determined
*Parameters* are reflective of system's properties
*Forcing functions* are external influences acting upon the system

$$a = \frac{F}{m}$$

a= dependent variable reflecting the system behavior
F= forcing function
m = parameter represeting system property

# Ordinary differential Equations (d.E.)

A simple first order differential equation has general form

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

$\frac{dy}{dt}$ change in *y* with respect to time $f(y,t)$ any function of *y* and time

An ordinary differential equation is a differential equation in which the dependent variable *y* depends only on one independent variable say *t* (so that derivatives of *y* are ordinary derivaties).

$$F(t, y, \frac{dy}{dt}, \frac{d^2y}{dt^2}, ..., \frac{d^ny}{dt^n}) = 0$$

$n^{th}$ order differential equation

Many analytical technique exist for solving such equations, but the methods can be applied to solve only selected class of differential equations.

In Numerical methods, one do not find a relation between variables.

*But find numerical values of the dependent variable for certain values of independent variables*

Solution is usually obtained in form of a table; the functional form of solution reamins unknown in so far as a specific forumlae is concern.

Divide as

**Initial Value Problems (IVP)**

Problems in which all the conditions are specified at the initial point only

**Boundary Value Problems (BVP)**

Problems involving second and higher differential equations, in which conditions at two or more points are specified.

To obtain a unique solution of $n^{th}$ order ordinary differential equation

→ *It is necessary to specify n values of dependent variable and/or its derivative at specific values of independent variable*.

# Classification of methods

Numerical solutions are obtained step-by-step through a series of equal intervals in the independent variable so that as soon as the solution $y$ has been obtained at $x=x_i$, the next step consist of evaluating $y_{i+1}$ *at* $x= x_{i+1}$

## *Single Step Methods*

Requires only the numerical value $y_i$ in order to compute the next $y_{i+1}$ for solving eqn.

*Taylor, Picard, Euler*

## *Multi Step Methods*

Requires not only the numerical value $y_i$ but also at least one of the past valyes $y_{i-1}$, $y_{i-2}$ , ....

*Adams-Bashforth method*

Single step method has obvious advantage over multi-step:
Don't need previous values ($y_n$, $y_{n-1}$, ..., $y_{n-p}$) which require initial value ($y_1$, $y_2$, ..., $y_n$) that have to be calculated by another method.

Disadvantage of single step:
Use many more evaluations of derivative to attain same degree of accuracy compared to multi-step methods

# Various numerical methods of solution of O.D.E.

Yield solution in one of the two forms

Series of *y* in terms of powers of *x* from which the value of *y* can be obtained by direct substitution

***Taylor's method***
***Picard's method***

Set of tabulated values of *x* and *y*

***Euler's method***
***Runge-Kutta method***
***Adams-Bashforth method***
***Milne's method***

- Euler's and Runge-Kutta methods, interval *h* sould be kept small, hence they can be applied for tabulating y only over a limted range.
- To get functional over wider range, Adams-Bashforth, Milne, Adams-Moulton methods may be applied, and require starting values, usually obtained by Taylor's series or Runge-Kutta methods.

# Taylor series method

Taylor series generated by *f* at *x*=*a* is

$$\sum_{k=0}^{\infty} \frac{f^{(k)}(a)}{k!}(x-a)^k = f(a) + (x-a)f'(a) + \frac{(x-a)^2}{2!}f''(a) + \ldots + \frac{(x-a)^{n-1}}{(n-1)!}f^{(n-1)}(a) + \frac{(x-a)^n}{n!}f^{(n)}(a) + \ldots$$

In most of the cases, Taylor's series converges to *f(x)* at every *x* and we often write the Taylor's series at *x*=*a* as

$$f(x) = f(a) + (x-a)f'(a) + \frac{(x-a)^2}{2!}f''(a) + \ldots$$

Or one can choose to write as:

$$y(x) = y(x_0) + (x-x_0)y'(a) + \frac{(x-x_0)^2}{2!}y''(x_0) + \ldots$$

# Taylor's Series Method

Solution of first order ODE IVP

$$\frac{dy}{dx} = f(x,y)$$ With initial condition $$y(x_0) = y_0$$

$$y(x) = y(x_0) + (x-x_0)y'(x_0) + \frac{(x-x_0)^2}{2!}y''(x_0)$$
$$\frac{(x-x_0)^3}{3!} + y'''(x_0) + ... + \frac{(x-x_0)^n}{n!}y^{(n)}(x_0) + ...$$

*y(x)* at any point can be calculated, provided $h=x-x_0$ is small

If the values of $y'(x_0)$, $y''(x_0)$ , ... are known then the above =n gives a power series for *y*

$$y'' = f' = \frac{df}{dx} = \frac{\partial f}{\partial x} + \left(\frac{\partial f}{\partial x}\right)y'$$

$$y''' = f'' = \frac{\partial^2 f}{dx^2} + \frac{\partial f}{\partial x}\frac{\partial f}{\partial y} + f\frac{\partial^2 f}{\partial x \partial y} + f\frac{\partial^2 f}{\partial y \partial x} + f^2\frac{\partial^2 f}{\partial y^2}$$ And so on.

When $x=x_0$ , $y=y_0$ *and* $x= x_1$ $(=x_0+h)$, *we get*

$$y_1 = y(x_1) = y_0 + hy_0' + \frac{h^2}{2!}y_0'' + \frac{h^3}{3!}y_0''' + ... + \frac{h^n}{n!}y_0^{(n)} + ...$$

$$y_2 = y(x_1+h) = y_1 + hy_1' + \frac{h^2}{2!}y_1'' + \frac{h^3}{3!}y_1''' + ... + \frac{h^n}{n!}y_1^{(n)} + ...$$

# Picards iteration method

Consider hthe initil value problem

$$y' = f(x, y), y(x_0) = y_0$$

$$dy = f(x, y)dx$$

$$\int_{y_0}^{y} dy = \int_{x_0}^{x} f(x, y)dx$$

or $$y(x) = y_0 + \int_{x_0}^{x} f(x, y)dx$$

$$y^{(1)}(x) = y_0 + \int_{x_0}^{x} f(x, y_0)dx$$ First approxiation

$$y^{(2)}(x) = y_0 + \int_{x_0}^{x} f(x, y^{(1)}(x))dx$$ Second approxiation

$$y^{(n)}(x) = y_0 + \int_{x_0}^{x} f(x, y^{(n-1)}(x))dx$$ $N^{th}$ approxiation

Process is stopped when two values of *y; $y^{(n-1)}$* and *$y^{(n)}$* are same to desigred degree of accuracy.

$$y' = f(x, y), y(x_0) = y_0$$

$$y^{(n)} = y_0 + \int_{x_0}^{x} f(x, y^{(n-1)})dx \qquad (n = 1, 2, 3, ...)$$

# Euler's method

Euler's method is simplest one-step method and has a limited application because of its low accuracy.

In this method, we determine the change Δy in y corresponding to small increase in the x

Let the d.E. to be $\frac{dy}{dx} = f(x, y), \qquad y(x_0) = y_0$

Let $y = g(x)$ be the solution of the equation above

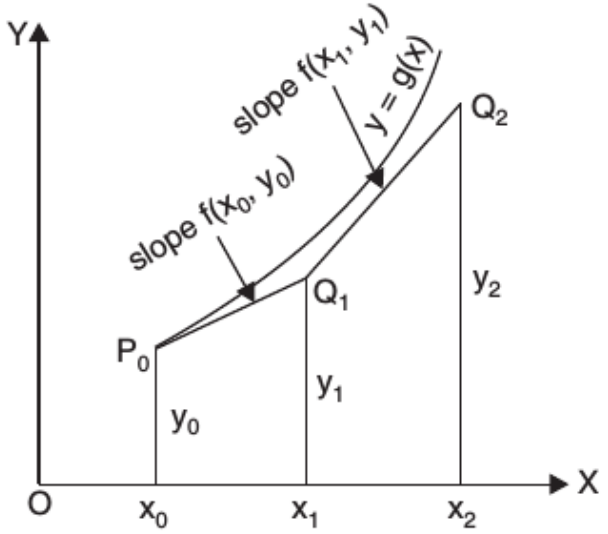
In this method, we use the property that in small interval, a curve is almost a straight line.

At point ($x_0$,$y_0$), we approximate the curve by tanget at the point ($x_0$,$y_0$)

$$y - y_0 = \left(\frac{dy}{dx}\right)_{x_0, y_0} (x - x_0)$$

$$y - y_0 = f(x_0, y_0)(x - x_0)$$

$$y = y_0 + (x - x_0) f(x_0, y_0)$$



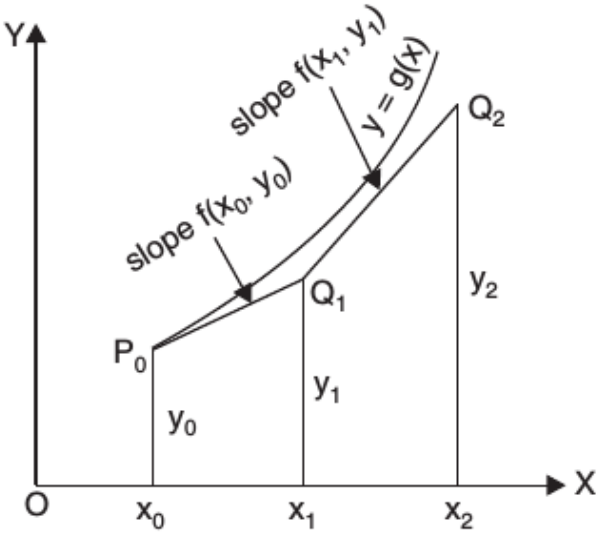
$$y = y_0 + (x - x_0) f(x_0, y_0)$$

$$\text{at } x \Rightarrow x_1 = x_0 + h$$

$$y_1 = y_0 + h f(x_0, y_0)$$

$$x_2 = x_1 + h$$

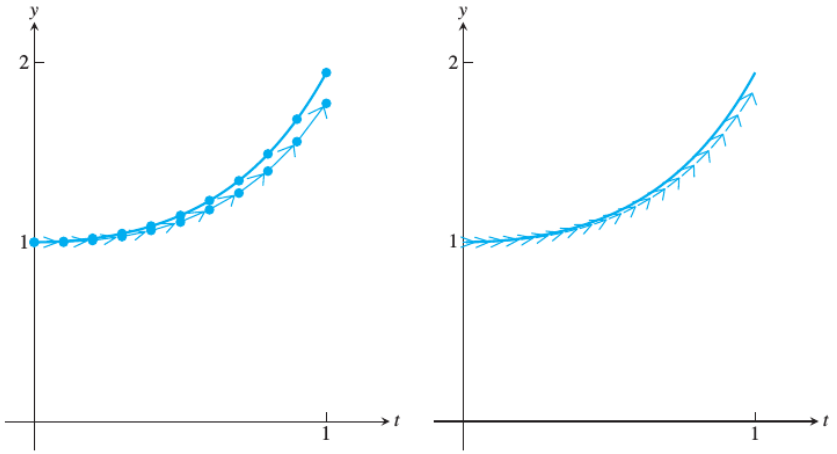
$$y_2 = y_1 + h f(x_1, y_1)$$




In general, one obtain

$$y_n = y_{n-1} + h f(x_{n-1}, y_{n-1}), \text{ where } x_n = x_{n-1} + h$$

This is called Euler's Formula



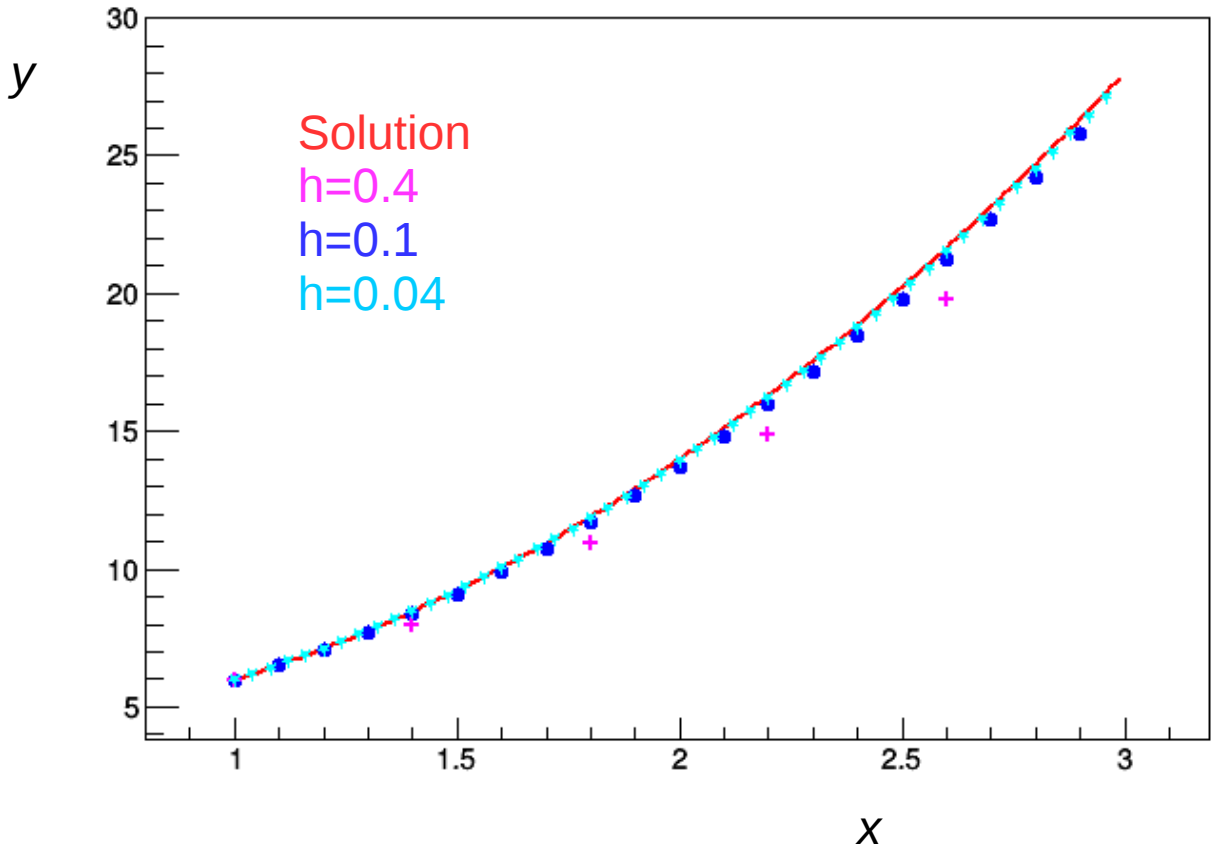

Disadvantage: if dy/dx changes rapidly over an interval; its value at the beginning of the interval may give a poor approximation.
The errors accumulate in succeeding intervals and value of y become errorneous.

Curve of actual solution is approximated by the h.
If h is not properly chosen, the numerical solution deviate significantly from the actual solution.

# Example A:

$$\frac{dy}{dx} = 6x - 1 \quad \text{with IC } y(1) = 6$$

Solution : $y = 3x^2 - x + 4$

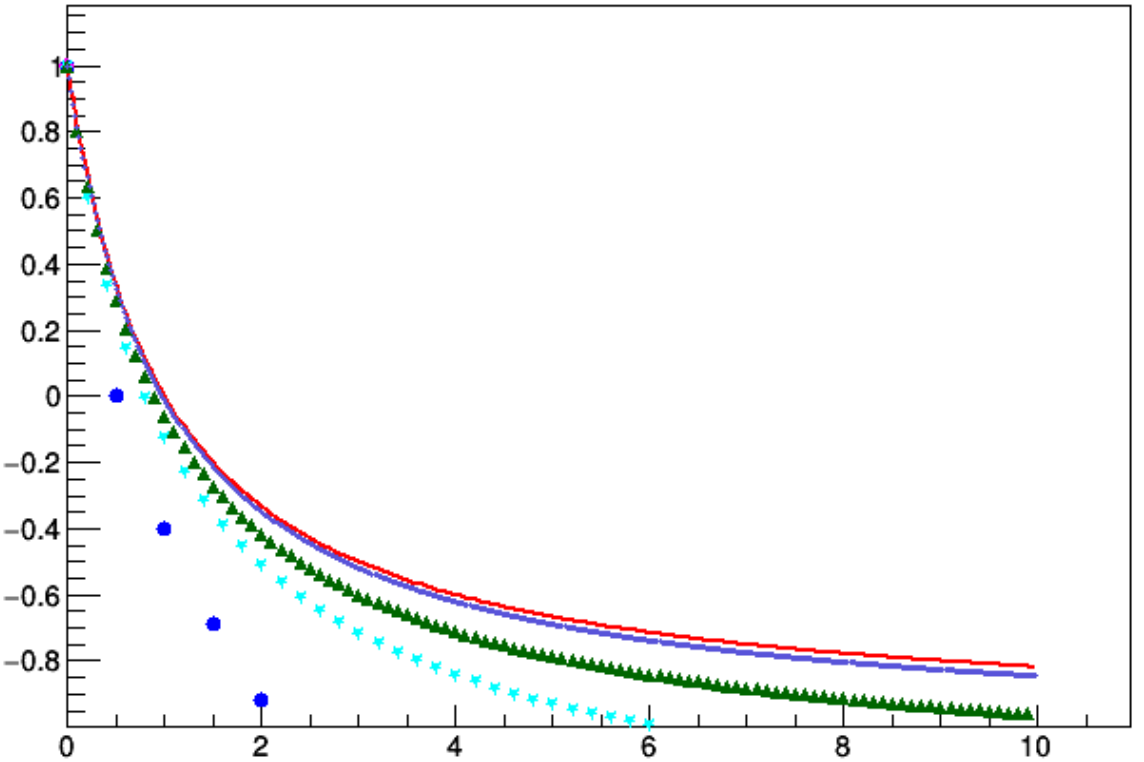


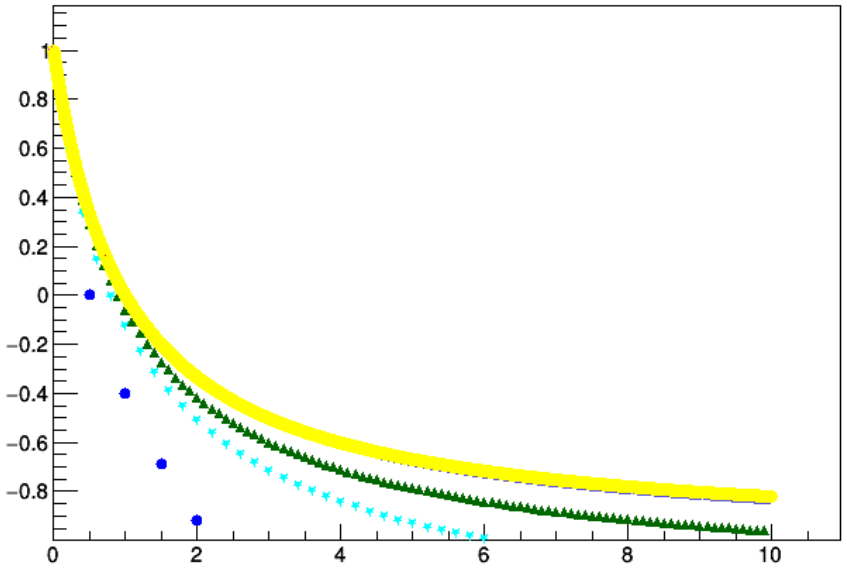
# Example B:

$$\frac{dy}{dx} = -\frac{1+y^2}{1+x^2} \text{ with IC } y(0) = 1$$

$$\text{Solution : } y = \frac{1-x}{1+x}$$

Solution
h=0.5
h=0.2
h=0.1
h=0.02
h=0.002

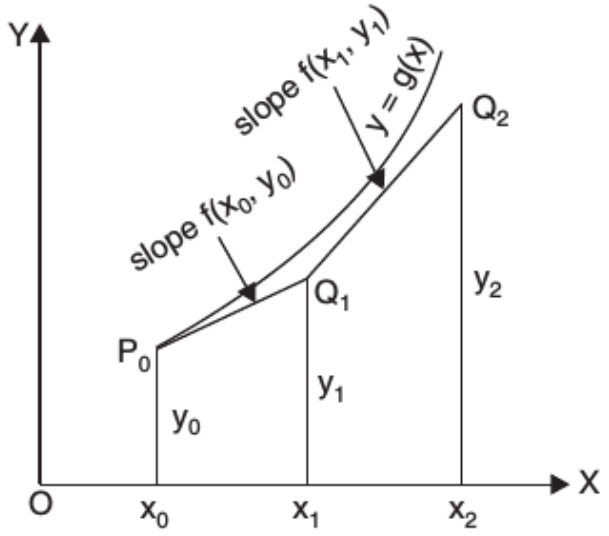


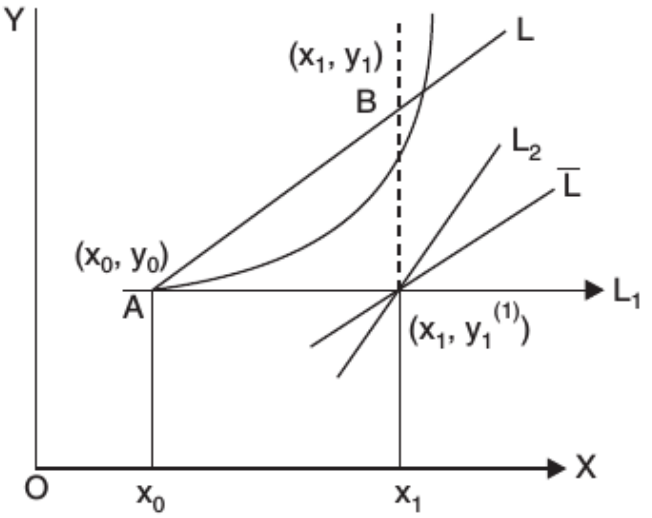
# Modified Euler's Methods

As seen, if *h* is not correctly choosen. Numerical value deviate from actual solution.

Modified Euler's method gives greater improvement in accuracy over the original Euler's method




$$y_1^{(1)} = y_0 + hf(x_0,y_0)$$




$L_1$ is tanget at $(x_0,y_0)$
$L_2$ is line through $(x_1, y_1^{(1)})$ of slope $f(x_1,y_1^{(1)})$
Lbar is slope equal to avaerage of $f(x_0,y_0)$ and $f(x_1,y_1^{(1)})$

$$y_1 = y_0 + (x_1 - x_0)\left(\frac{f(x_0,y_0) + f(x_1,y_1^{(1)})}{2}\right)$$

$$= y_0 + h\left(\frac{f(x_0,y_0) + f(x_1,y_1^{(1)})}{2}\right)$$

*Also called Huen's Method*

# Predictor-Corrector Approach

Predictor

$$y_n^0 = y_{n-1} + hf(x_{n-1}, y_{n-1}), \text{ where } x_n = x_{n-1} + h$$

$$y_n = y_{n-1} + h\frac{f(x_{n-1}, y_{n-1}) + f(x_n, y_n^0)}{2}, \text{ where } x_n = x_{n-1} + h$$

Corrector

This iterative process does not necessarily converge on the true answer but will converge on an estimate with a finite truncation error.

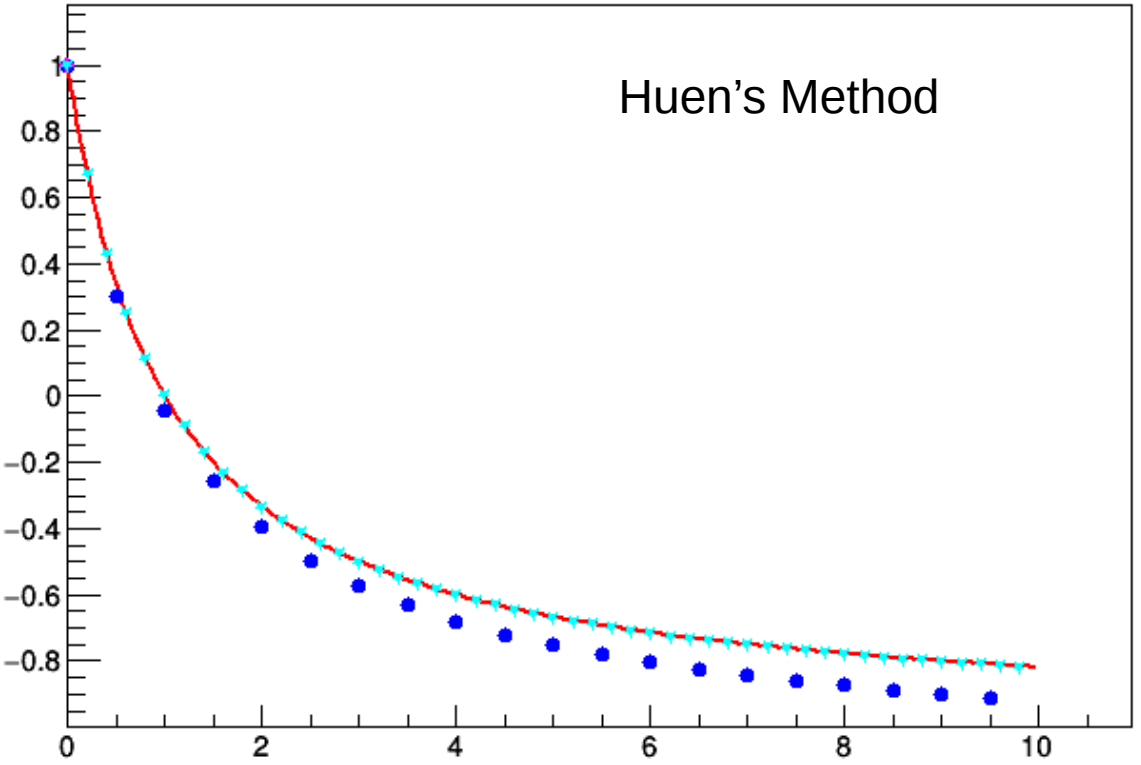
$$|\epsilon_{y_n}| = \left|\frac{y_n^k - y_n^{k-1}}{y_n^k}\right| \times 100\%$$

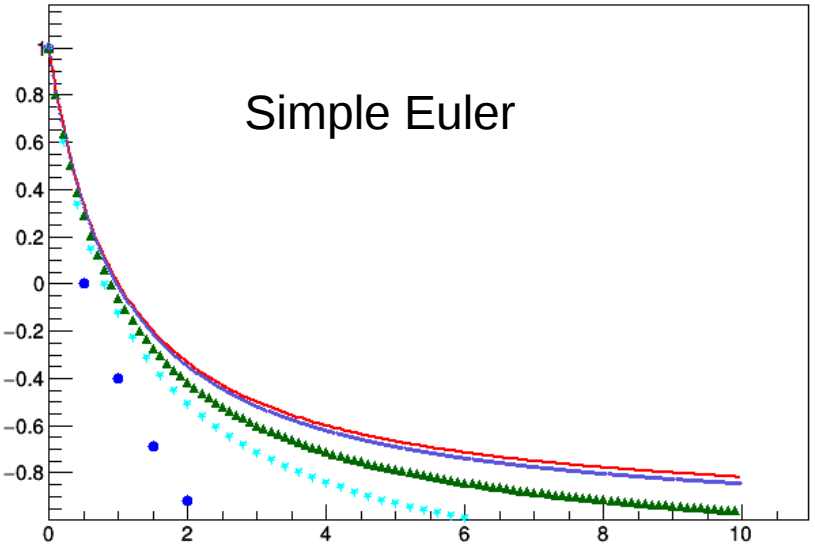
# Example Bh:

$$\frac{dy}{dx} = -\frac{1+y^2}{1+x^2} \text{ with IC } y(0) = 1$$

$$\text{Solution : } y = \frac{1-x}{1+x}$$

Solution
h=0.5
h=0.2
h=0.1
h=0.02



# Midpoint Method

$$y_{n+1/2} = y_n + \frac{h}{2} f(x_n, y_n)$$

Then this predicted value is used to calculate a slope at the midpoint

$$y'_{n+1/2} = f(x_{n+1/2}, y_{n+1/2})$$

This slope is then used to extrapolate linearly from $x_n$ to $x_{n+1}$

$$y'_{n+1} = y_n + h f(x_{n+1/2}, y_{n+1/2})$$

Midpoint method is superior to normal Euler's method because it utilizes a slope estimate at the midpoint of the pediction interval.

# Example Bm:

$$\frac{dy}{dx} = -\frac{1+y^2}{1+x^2} \text{ with IC } y(0) = 1$$

$$\text{Solution : } y = \frac{1-x}{1+x}$$

Solution
h=0.5
h=0.2
h=0.1
h=0.02