MECH 479/587

Computational Fluid Dynamics

Module 2 (a): Solution of ODEs

Rajeev K. Jaiman
Winter Term 1, 2022

Overview

- ❑ Introduction to ODE
- ❑ Numerical solution of ODEs
  - ► Approximation concepts for numerical discretization and integration
- ❑ Discussion on error analysis, stability and accuracy
- ❑ Show the implementation of numerical algorithms into computer codes

Measurable Learning Outcomes

- ❑ Define $\underline{\text{initial value problem}}$ of ODE system
- ❑ Approximate a simple ODE system
- ❑ Verify numerical integration schemes of ODE with analytical solutions
- ❑ Understand the $\underline{\text{local error analysis}}$
- ❑ Linking between numerical integration of ODE with Taylor series expansion

$\rightarrow$ High-order methods

$$\left.\begin{array}{l}\text{- Runge-Kutta methods}\\ \text{- Adams family}\end{array}\right\}$$

What is ODE?

- A differential equation containing one or more functions of one independent variable and the derivatives of those functions

Let y be a dependent variable and t independent variable, and $y = f(t)$ is unknown function of t or x.

A general implicit form of ODE is:

$$\underline{F\left(t, y, y', y'', y''', \ldots, y^{(n)}\right) = 0}$$

where

$$y' = \frac{dy}{dt}, \quad y'' = \frac{d^2 y}{dt^2}, \; y''' = \frac{d^3 y}{dt^3}, \; y^{(n)} = \frac{d^n y}{dt^n}$$

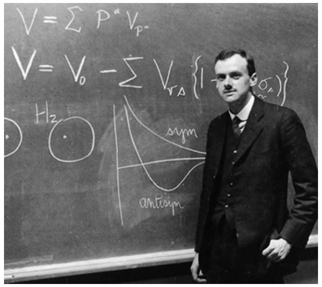
- Example: Newton's 2nd law of motion

$ma = F$

Numerical Solution of ODEs

- ❑ Analytical solution of ODE gives closed-form formula
  - ► That can be evaluated at any point
- ❑ Numerical solution of ODE is table of approximate values of solution function at discrete set of points
  - ► Starting at t_0 with given initial value y_0, we track trajectory dictated by ODE
  - ► Approximate solution values are generated step by step in increments moving across interval in which solution is sought
  - ► During stepping or integration, from one discrete point to another, there exist some error
  - ► Stability or instability of solutions determine whether errors are magnified or diminished with time

Approximations


$V = \sum P^{\alpha} V_{p^{\alpha}}$
$V = V_0 - \sum V_{rA} \{ 1 \ldots \sigma_r \}$
H_2
sym
antisym

I've learned that, in the description of Nature, one has to tolerate approximations, and that work with approximations can be interesting and can sometimes be beautiful.

P. A. M. Dirac

Errors in Numerical Solution of ODEs

- ❑ Two distinct types of errors:
  - ► Rounding error, which is due to finite precision of floating-point arithmetic
  - ► Truncation error (discretization error), which is due to approximation method used and would remain even if all arithmetic were exact
    - ◇ In practice, truncation error is dominant factor determining accuracy of numerical solutions of ODEs, so we will henceforth ignore rounding error

Errors in Numerical Solution of ODEs

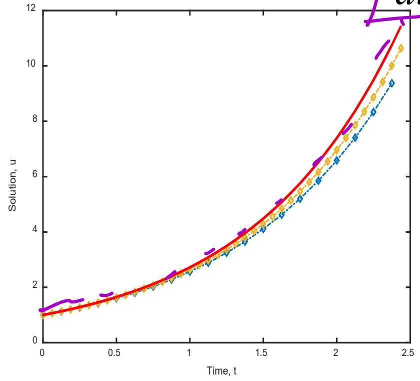
- Truncation error can be broken into:
  - Global error: difference between computed solution and true solution passing through initial point
  - Local error: error made in one step of numerical integration

Coding Example

$$\frac{du}{dt} = \underbrace{g(t,u)}_{u}$$

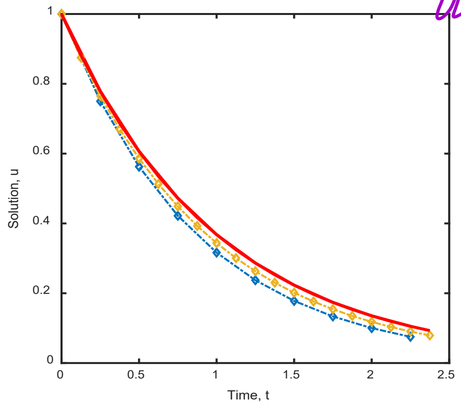
$u(t)=e^{t}$

- ❑ Unstable ODE $\boxed{\frac{du}{dt} = u}$



- ❑ Stable ODE $\frac{du}{dt} = -u$

$u=e^{-t}$



```
% MECH 479 - CFD
% EX1: A simple code for error analysis
nstep=20;
dt=0.125;
u1=zeros(nstep,1);
uex=zeros(nstep,1);
t=zeros(nstep,1);
t(1)=0;u1(1)=1;uex(1)=1;
for i=2:nstep
    u1(i)=u1(i-1)-dt*u1(i-1); %Forward Euler
    t(i)=t(i-1)+dt;
    uex(i)=exp(-t(i));
end;
plot(t,u1,'-.d','linewidt',2);
hold on;
plot(t,uex, 'r', 'linewidt',3);
set(gca,'fontsize',16,'linewidt',2);
xlabel("Time, t");
ylabel("Solution, u");
hold on;
```

Review: Order of Accuracy and Stability of ODE Solution

$h_k = \Delta t$

Order of accuracy of numerical method is *p* if

$$e_k = O(h_k^{\,p+1}) \qquad \text{where } h_k = \Delta t$$

$O(h_k^{\,p+1})$

Local error per unit step: $e_k / h_k = O(h_k^{\,p})$

$O(h_k^{\,p})$

Numerical method is stable if small perturbations do not cause resulting numerical solutions to diverge from each other without bound

- Such divergence of numerical solutions could be caused by inherent instability of solutions to ODE, but can also be due to <u>numerical method itself</u>, even when solutions to ODE are stable

General first-order ODE system

Integrating a $\underline{\text{first-order ODE system}}$

$$\frac{du}{dt} = g(t,u)$$

The initial condition must be given as:

$$\underline{u(t_o) = u^0}$$

This is called initial value problem (IVP). Infinite family of functions satisfies ODE. To single out particular solution, initial value of solution function must be specified at initial time t_o

Examples: Family of solutions for different ODEs

Numerical Discretization

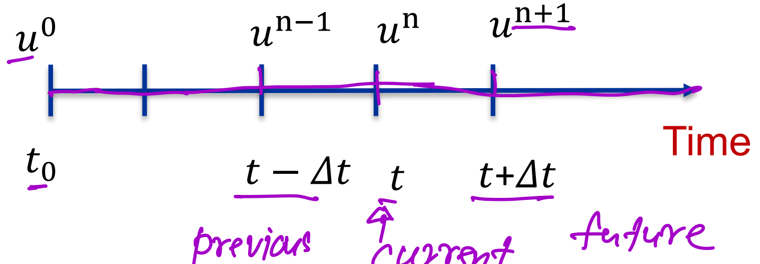
A numerical solution of

$$\boxed{\frac{du}{dt} = g(t,u)}$$

Consists of discrete values of u at discrete times

$$u^0 = u(t_0) \qquad u^1 = u(t_0 + \Delta t) \qquad u^2 = u(t_0 + 2\Delta t)$$

$$u^{n-1} = u(t - \Delta t) \qquad u^n = u(t) \qquad u^{n+1} = u(t + \Delta t)$$



Numerical Integration

Consider

$$\underline{\frac{du}{dt} = g(t,u)}$$

$$g(t,u) = \frac{du}{dt}$$

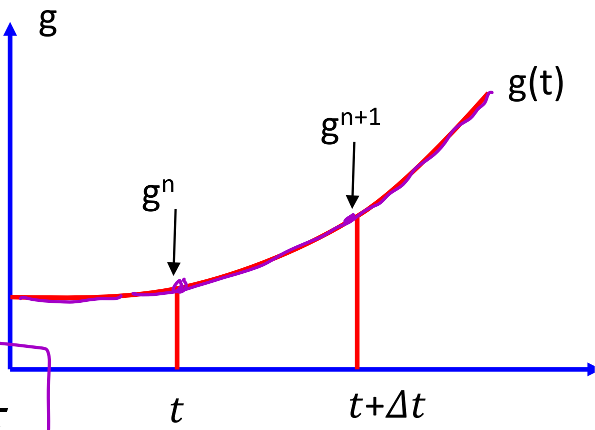
Rearrange

$$du = g(\tau,u)\,d\tau$$

Integrate

$$\int_t^{t+\Delta t} du = \int_t^{t+\Delta t} g(\tau,u)\,d\tau$$

$$\Rightarrow \boxed{u^{n+1} - u^{n} = \int_t^{t+\Delta t} g(\tau,u)\,d\tau}$$



Forward Euler

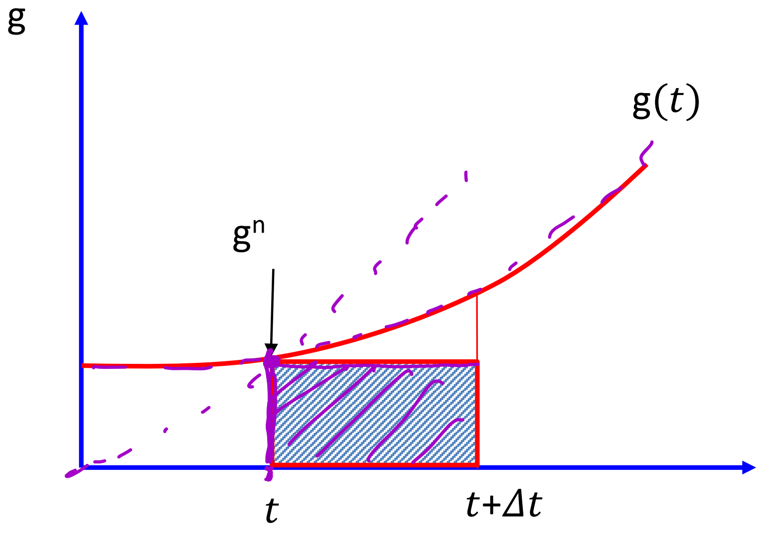
Approximate:

$$\int_t^{t+\Delta t} du = \int_t^{t+\Delta t} g(\tau,u)\,d\tau \approx g(t)\Delta t = g^n\Delta t$$

$$\Rightarrow \boxed{u^{n+1} = u^n + g^n\Delta t}$$

$$\boxed{u^{n+1} = u^n + g^n \Delta t}$$

Leonhard Euler



- Euler's method is single-step method because it depends on information at only one point in time to advance to next point
  - Euler's method advances solution by extrapolating along straight line whose slope is given

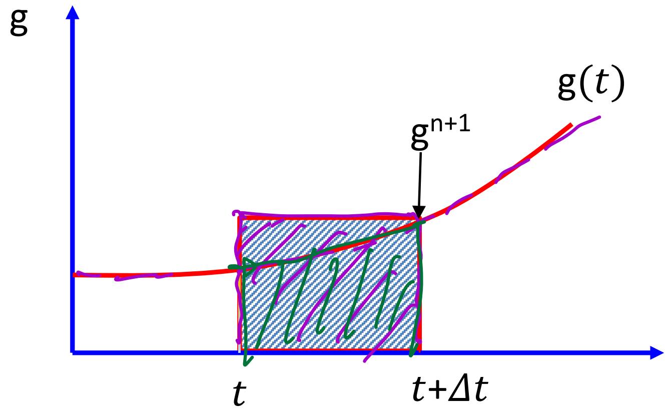
Backward Euler

$$\frac{du}{dt} = g(t,u)$$

Approximate:

$$\int_t^{t+\Delta t} du = \underline{\int_t^{t+\Delta t} g(\tau,u)\,d\tau} \approx \underline{g(t+\Delta t)}\Delta t = \underline{g}^{n+1}\Delta t$$

$$\Rightarrow \boxed{u^{n+1} = u^n + g^{n+1}\Delta t}$$

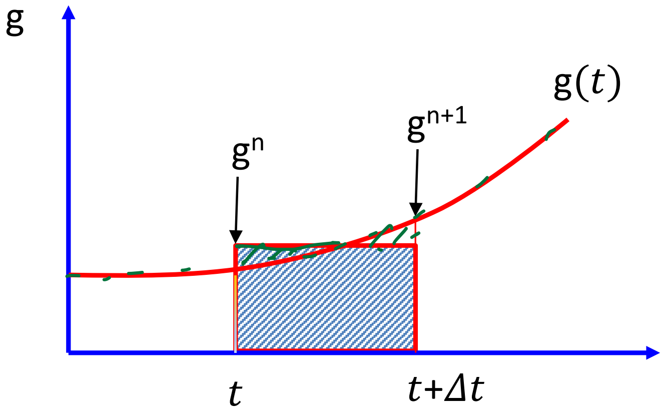


Trapezoidal Rule

Approximate:

$$\int_{t}^{t+\Delta t} g(\tau,u)\,d\tau \approx \frac{1}{2}\left(\underline{g(t)}+\underline{g(t+\Delta t)}\right)\Delta t = \underline{\frac{1}{2}\left(g^{n}+g^{n+1}\right)\Delta t}$$

$$\Rightarrow \boxed{u^{n+1} = u^{n} + \frac{1}{2}\left(g^{n}+g^{n+1}\right)\Delta t}$$



Recap

$$\frac{du}{dt} = g(t,u)$$

$$u(t_0) = u_0$$

- Forward Euler

$$\underline{u^{n+1} = u^n + g^n \Delta t}$$

$\uparrow$ current

- Backward Euler

$$\underline{u^{n+1} = u^n + g^{n+1} \Delta t}$$

$\uparrow$ future

- Trapezoidal rule

$$u^{n+1} = u^n + \frac{1}{2}\underbrace{(g^n + g^{n+1})}_{\text{Average}}\Delta t$$

Exact Solution

Take

$$\underline{g(t,u) = -u}$$

Then

$$\boxed{du = -u\,dt}$$

With initial condition $u(0) = 1$

The exact solution (Ground-Truth) is:

$$\boxed{u(t) = e^{-t}}$$

Accuracy Analysis

$$\frac{du}{dt} = -u^{\swarrow} \Rightarrow u(t) = e^{-t}$$

- Forward Euler

$$u^{n+1} = u^n - u^n \Delta t \Rightarrow u^{n+1} = (\underbrace{1-\Delta t}_{}) u^n$$

- Backward Euler

$$u^{n+1} = u^n - u^{n+1} \Delta t \Rightarrow u^{n+1} = \frac{u^n}{(1+\Delta t)}$$

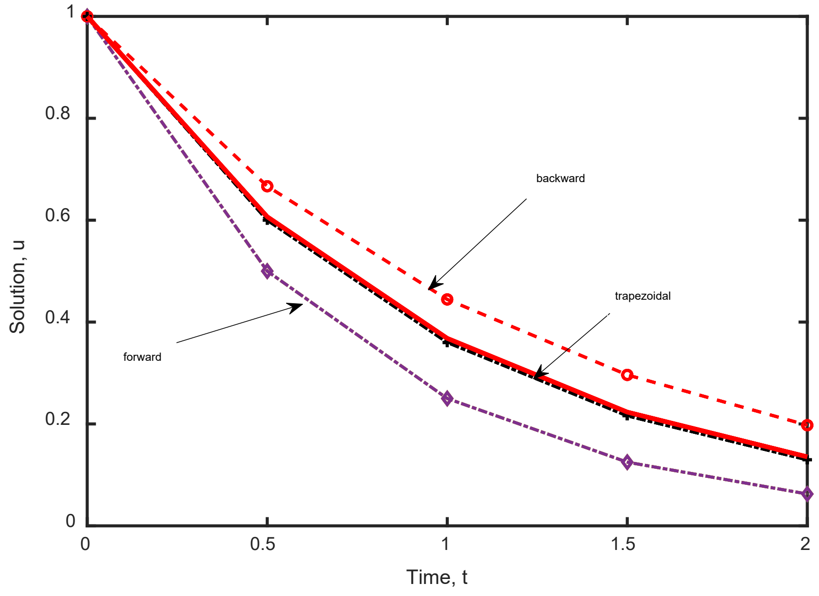
- Trapezoidal rule

$$u^{n+1} = u^n - \frac{\Delta t}{2}(u^n + u^{n+1}) \Rightarrow u^{n+1} = \frac{\left(1-\frac{\Delta t}{2}\right)}{\left(1+\frac{\Delta t}{2}\right)} u^n$$

An Example Code

```
% MECH 479 - CFD
% A sample code for three integration methods
nstep=5;
dt=0.5;
u1=zeros(nstep,1);
u2=zeros(nstep,1);
u3=zeros(nstep,1);
uex=zeros(nstep,1);
t=zeros(nstep,1);
t(1)=0;u1(1)=1;u2(1)=1;u3(1)=1;uex(1)=1;
for i=2:nstep
    u1(i)=u1(i-1)-dt*u1(i-1); %Forward Euler
    u2(i)=u2(i-1)/(1.0+dt); %Backward Euler
    u3(i)=u3(i-1)*(1.0-0.5*dt)/(1.0+0.5*dt); %Trapezoidal Rule
    t(i)=t(i-1)+dt;
    uex(i)=exp(-t(i));
end;
plot(t,u1);hold on;plot(t,u2,'r');plot(t,u3,'k');
plot(t,uex, 'r', 'linewidt',3);
set(gca,'fontsize',24,'linewidt',2);
```

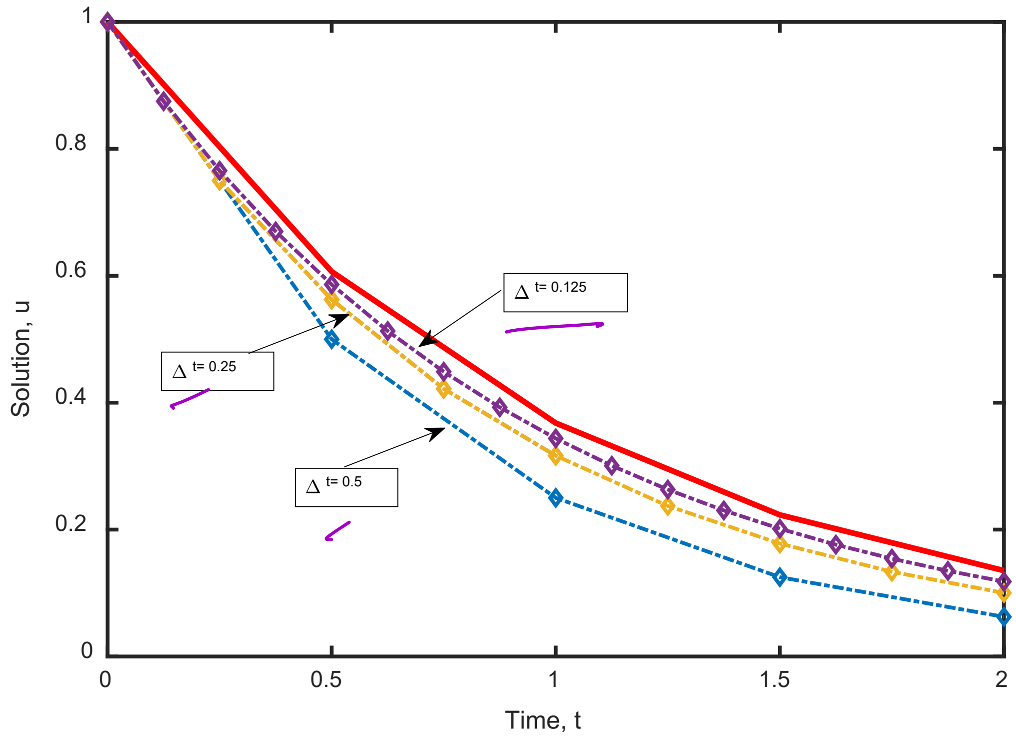
Numerical Analysis



- Overall, numerical solutions have a similar behavior as the exact solution
  - The trapezoidal rule results in numerical values that are approximately the same.
- Can we improve the accuracy of the forward and backward Euler method?
- Re-run the forward Euler method with smaller time steps

Comparison with Exact Solution

- ❑ Forward Euler
  - ▶ Dependence of solution on time step size
  - ▶ Detailed error analysis

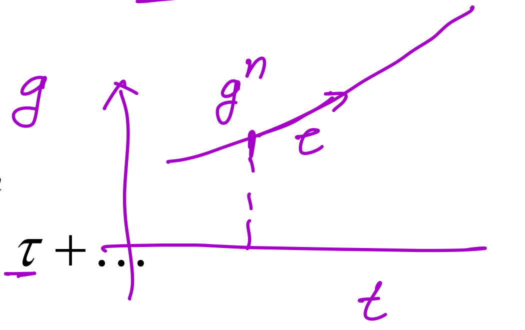


Error Analysis

- ❑ Now examine the error in computing the integral using the Taylor series

Expand

$$\underline{g} = \underline{g^n} + \left(\frac{dg}{dt}\right)^n \underline{\tau} + \ldots$$



Then

$$\underline{\int_t^{t+\Delta t} g\,d\tau} = \underline{\int_t^{t+\Delta t} g^n\,d\tau} + \underline{\int_t^{t+\Delta t} \left(\frac{dg}{dt}\right)^n \tau\,d\tau} + \ldots$$

Integrate

$$* \quad \boxed{\int_t^{t+\Delta t} g\,d\tau = g^n \Delta t + \left(\frac{dg}{dt}\right)^n \frac{\Delta t^2}{2} + L}$$

Error Analysis

- Exact

$$\boxed{u^{n+1} = u^n + \int_t^{t+\Delta t} g(t,u)\,d\tau}$$

- Approximate integral

$$\int_t^{t+\Delta t} g\,d\tau = g^n \Delta t + \left(\frac{dg}{dt}\right)^n \frac{\Delta t^2}{2} + L$$

$$\Rightarrow u^{n+1} = u^n + \underline{g^n \Delta t} + \boxed{\underbrace{\left(\frac{dg}{dt}\right)^n \frac{\boxed{\Delta t^2}}{2} + L}_{O(\Delta t^2)}}$$

$$\Rightarrow u^{n+1} = u^n + g^n \underline{\Delta t} + \underline{\boxed{O(\Delta t^2)}}$$

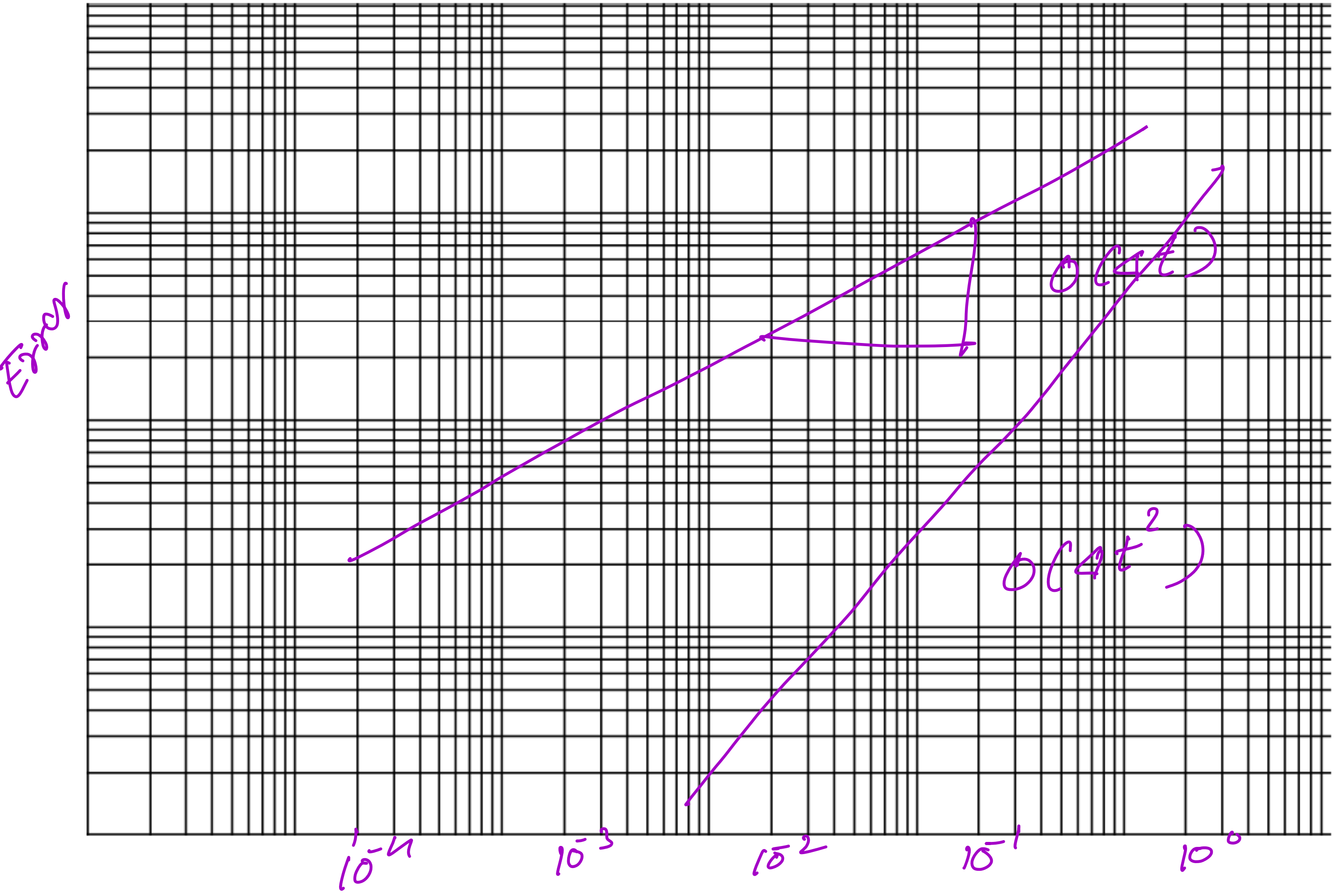
Error at each time step

$p = 1$

$O(\Delta t^1)$

$$\Rightarrow \frac{u^{n+1} - u^n}{\Delta t} = g^n + \frac{O(\cancel{\Delta t}^2)}{\cancel{\Delta t}}$$

Local order of accuracy: convergence rate of truncation error



Summary (1)

- ❑ Introduction of ODEs in time
  - ► Conceptual understanding can be extended to Heun, Runge-Kutta and other advanced multistep techniques.
- ❑ Implementation in a MATLAB code
- ❑ Error analysis

Finite Difference Approximation of the Derivatives

$$\frac{du}{dt} = g(t,u)$$

- The time integration by forward Euler can be formulated by a Taylor series as well

$$\frac{u(t_0+\Delta t) - u(t_0)}{\Delta t}$$

- Time derivative
  - ▶ Expansion at t_0

$$u(t_0+\Delta t) = u(t_0) + (\Delta t)\left.\frac{du}{dt}\right|_{t=t_0} + \frac{1}{2!}(\Delta t)^2 \left.\frac{d^2u}{dt^2}\right|_{t=t_0} + \frac{1}{3!}(\Delta t)^3 \left.\frac{d^3u}{dt^3}\right|_{t=t_0} + \cdots$$

- Solving for the time derivative gives

$$\Rightarrow \boxed{\frac{du}{dt} = \frac{u(t_0+\Delta t) - u(t_0)}{\Delta t} - \frac{1}{2}(\Delta t)\frac{d^2u}{dt^2} + \ldots}$$

$$\Rightarrow \frac{du}{dt} = \boxed{\frac{u^{n+1} - u^n}{\Delta t}} + O(\Delta t)$$

Application to First-Order ODE

Brook Taylor

- The original equation at time level n is

$$\left(\frac{du}{dt}\right) = g^n$$

- Applying time derivative result

$$\boxed{\frac{u^{n+1} - u^n}{\Delta t} = g^n + O(\Delta t)}$$

- This is exactly the first order forward Euler method.

High-Order Methods

- Runge-Kutta methods
  - A class of methods which judiciously uses the information on the slope or derivative at more than one point to extrapolate the solution to the future time step.
  - Derivation of second order RK (see handout)

- Adams Methods
  - Adams methods are based on the idea of approximating the integrand with a polynomial within time interval
  - Second order Adams-Bashforth (AB2) (see handout)

$$u_{n+1} = u_n + \frac{(g_n + g_{n+1})}{2}$$

Runge-Kutta Methods

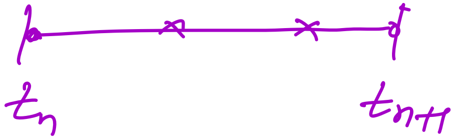
$$\frac{du}{dt} = g(t, u)$$

$$\boxed{h = \Delta t}$$

① consider the following approx:

$$u_{n+1} = u_n + aK_1 + bK_2$$

where K_1 & K_2 are:



$$K_1 = h\, \underline{g}\,(u_n, t_n)$$

$$K_2 = h\, g\,(u_n + \underline{\ }\beta K_1,\ t_n + \underline{\alpha} h)$$

$\underline{a, b, \alpha, \beta}$ coef's need to be determined!

Second-Order RK Method

② $u_{n+1} = u_n + h \left.\frac{du}{dt}\right|_{t_n} + \underbrace{\frac{h^2}{2} \left.\frac{d^2u}{dt^2}\right|}_{t_n} + O(h^3)$

(~~Taylor~~ series)

Using $\frac{du}{dt} = g(t,u)$

✓ $\frac{d^2u}{dt^2} = \frac{dg}{dt} = \frac{\partial g}{\partial t} + \frac{\partial g}{\partial u} \boxed{\frac{\partial u}{\partial t}} \swarrow g$

$= \frac{\partial g}{\partial t} + g \frac{\partial g}{\partial u}$ ③

Substitute in ②

(4) $$u_{n+1} = u_n + h\, g(u_n, t_n) + \frac{h^2}{2}\left[\frac{\partial g}{\partial t} + g\frac{\partial g}{\partial u}\right](u_n, t_n) + O(h^3)$$

Consider the expansion

$$K_2 = h\, \vec{g}\,(t_n + \alpha h,\ u_n + \beta K_1)$$

Using Taylor series:

$$\Rightarrow K_2 = h\left[g(t_n, u_n) + \alpha h \frac{\partial g}{\partial t} + \beta K_1 \frac{\partial g}{\partial u}\right] + O(h^3)$$

Substitute in Eq. (1)

(5) $$u_{n+1} = u_n + (\underbrace{a+b}_{=}) h\, g(u_n, t_n) + b h^2\left(\alpha\frac{\partial g}{\partial t} + \beta g \frac{\partial g}{\partial u}\right) + O(h^3).$$

Compare it with Eq. (4)
(Taylor series)

$$a + b = 1, \quad \alpha b = \frac{1}{2}, \quad \beta b = \frac{1}{2}$$

There are many possibilites to select for α, β, a, b

Let's choose : $\alpha = \beta = 1$

$\Rightarrow \quad a = \frac{1}{2}, \quad b = \frac{1}{2}$

With this choice, we have the classical second-order RK method

$$k_1 = h\, g(t_n, y_n)$$

$$k_2 = h\, g(t_n + h, y_n + k_1)$$

$$\boxed{u_{n+1} = u_n + \frac{(k_1 + k_2)}{2}}$$

Adams Methods

$$\frac{du}{dt} = g(t,u)$$

$$u_{n+1} = u_n + \int_{t_n}^{t_{n+1}} \frac{du}{dt}\,dt = u_n + \int_{t_n}^{t_{n+1}} g(u,t)\,dt$$

Adams family methods use the idea of approximating the integrand with a polynomial within the interval (t_n, t_{n+1}).

$\Rightarrow$ Using kth order polynomial $(k+1)$th order method can be constructed!

Adams-Bashforth (AB2) Method

AB 2 method for : $\frac{du}{dt} = g(t,u)$

$u(t=0) = u_0$

Step size : $h = \Delta t$

$$u_{n+1} = u_n + \frac{3}{2} h\, g(t_n, u_n) - \frac{1}{2} h\, u(t_{n-1}, u_{n-1})$$

This is explicit two-step AB method.

$\Rightarrow$ Derivation is given in Handout & will be reviewed in the class!

Rajeev K. Jaiman

Email: rjaiman@mech.ubc.ca