Lecture 10

Oscillations : Computational Approach

ODEs using MATLAB

SHM:

- ODE
- Initial condition
- Euler's method; Finite Difference

<u>Pendulam →</u> particle of mass m connected by a massless string to a rigid support (<u>goal:</u> to understand particles trajectory)

<u>Case 1:</u> simple case without friction

Write down the equation of motion.

Forces

Tension and gravity.

Component parallel and perpendicular to string.

Parallel force add to zero !!; force perpendicular to the string is

$$F = - m\, g \sin(\theta)$$

Force is always opposite to the displacement from the position theta=0 → -ve sign.

Newtons law.

$$F_\theta = m d^2 s / dt^2$$

displacement $s = \ell\,\theta$

Assuming angle theta is very small, sin(theta) ~ theta

$$\frac{d^2\theta}{dt^2} = -\frac{g}{\ell}\theta$$

Characteristic time scale!!

First order ODEs

$$\frac{d\omega}{dt} = -\frac{g}{\ell}\theta ,$$

$$\frac{d\theta}{dt} = \omega ,$$

$$\omega_{i+1} = \omega_i - \frac{g}{\ell}\theta_i \Delta t$$

$$\theta_{i+1} = \theta_i + \omega_i \Delta t .$$

Angular displacement and angular velocity .

MATLAB code

Conservation of energy over one cycle !!

Eulers method !! → unstable

Not suitable for this problem.

Euler-Cromer method→

- **previous value of omega and theta to calculate new value of omega.**
- **But new value of omega to calculate new value of theta.**

Knowledge of numerical methods/Numerical approaches

ODEs in MATLAB

ODEs : MATLAB

General form ODE : **du/dt=F(u)**

System with 3 unknown functions x,y,z of time “t”

$$\frac{d}{dt}\begin{pmatrix} u_1(t) \\ u_2(t) \\ u_3(t) \end{pmatrix} = \begin{pmatrix} f_1(u_1, u_2, u_3) \\ f_2(u_1, u_2, u_3) \\ f_3(u_1, u_2, u_3) \end{pmatrix}.$$

$$u_1 \equiv x$$
$$u_2 \equiv y$$
$$u_3 \equiv z$$

ODEs : MATLAB

General form ODE : **du/dt=F(u)**

$$\frac{d}{dt}\begin{pmatrix} u_1(t) \\ u_2(t) \\ u_3(t) \end{pmatrix} = \begin{pmatrix} f_1(u_1,u_2,u_3) \\ f_2(u_1,u_2,u_3) \\ f_3(u_1,u_2,u_3) \end{pmatrix}.$$

$$u_1 \equiv x$$
$$u_2 \equiv y$$
$$u_3 \equiv z$$

In our previous example of SHM

$$\frac{du_1}{dt} =$$
$$\frac{du_2}{dt} =$$

→ **ODEs**

$$\mathbf{u} = \begin{pmatrix} x \\ v \end{pmatrix}.$$

And the corresponding vector F(u)

ODE solvers in MATLAB

Matlab has its own DE solver → more accurate.

1. Define the RHS function for your problem (set of 1st order DE) → in a M file (myfunction.m)
2. Choose the beginning and ending time to pass to the matlab ODE function.
3. Put the initial column vector “u” in the variable “u0” to define the initial conditions of your problem.
4. Choose the ODE solver control options (Matlab’s odeset function).

ODE solvers in MATLAB

5. Ask Matlab to give → a column of times *t* and a matrix of **u**-values by calling one of the ode solvers

[t,u]=ode45(@myfunction,[tstart,tfinal],u0,options);

6. <u>The DE solver then returns:</u>
→ A column vector "t" of the discrete times between tstart and tfinal (accuracy defined via "odeset" or "tstart:dt:tfinal")
→ A matrix u with as many columns as you have unknowns in your set of ode's.

ODEs : MATLAB

Ode45 : Explicit one step RK medium order solver.
Moderate accuracy; non-stiff problem.
Typically the first solver to try for a new problem.

Syntax

[T,Y] = *solver*(odefun,tspan,y0)
[T,Y] = *solver*(odefun,tspan,y0,options)

ODEs : MATLAB