ME242 – MECHANICAL ENGINEERING SYSTEMS

LECTURE 24:

- Numerical Methods - Matlab

INTEGRATION OF ODE's

Given a scalar, first-order differential equation of the form

$$\frac{dy}{dt} = f(y,t) \qquad \text{with} \qquad y(t_o) = y_o$$

Goals:

- Advancement of the system in time by integration of ODE
- The quantity being integrated, f, is itself a function of the result of the integration, y.

Method:

- We seek to approximate the solution $y(t)$ at timestep $t_{n+1}=t_n+h$ given t_o and the solution at the previously-computed timesteps t_1 to t_n

Note:

- ODE's of higher order are generalization of this first order case

INTEGRATION OF ODE's - STIFFNESS

A stiff ODE is an ordinary differential equation that has a transient region whose behavior is on a different scale from that outside this transient region.

An important characteristic of a stiff system is that the equations are always stable, meaning that they converge to a solution. The following example clarifies this characteristic:

$$\begin{bmatrix} u' \\ v' \end{bmatrix} = \begin{bmatrix} 998 & 1998 \\ -999 & -1999 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}, \quad \begin{bmatrix} u(0) \\ v(0) \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

Dominates the solution

$$u(x) = 2e^{-x} + e^{-1000x}$$
$$v(x) = -e^{-x} + e^{-1000x}$$

Has negligible effect on the solution.
Restricts the step size of the numerical solver in order to make it stable

INTEGRATION OF ODE's - CLASSIFICATION

Number of steps:

Multi-Step

Single-Step
To compute y_n we only need the immediately preceding time poin y_{n-1}

Size of steps:

Variable-Step

Fixed-Step
The value of h is constant

NOTE: We have focused on Single-Step, Fixed-Step, both explicit and implicit methods

INTEGRATION OF ODE's – MATLAB

Solvers for Non-Stiff Problems:

ODE45: Based on an explicit Runge-Kutta (4,5) formula, the Dormand-Prince pair. It is a *one-step* solver.In general, ode45 is the best function to apply as a "first try" for most problems.

ODE23: Based on an explicit Runge-Kutta (2,3) pair of Bogacki and Shampine. It may be more efficient than ode45 at crude tolerances and in the presence of mild stiffness. Like ode45, ode23 is a one-step solver.

ODE113: Variable order Adams-Bashforth-Moulton PECE solver. It may be more efficient than ode45 at stringent tolerances and when the ODE function is particularly expensive to evaluate. ode113 is a *multistep* solver.

INTEGRATION OF ODE's – MATLAB

Solvers for Stiff Problems:

ODE15s: Variable-order solver based on the numerical differentiation formulas (NDFs). Optionally it uses the backward differentiation formulas, BDFs, (also known as Gear's method). Like ode113, ode15s is a multistep solver. If you suspect that a problem is stiff or if ode45 failed or was very inefficient, try ode15s.

ODE23s: Based on a modified Rosenbrock formula of order 2. Because it is a one-step solver, it may be more efficient than ode15s at crude tolerances. It can solve some kinds of stiff problems for which ode15s is not effective.

ODE23t: An implementation of the trapezoidal rule using a "free" interpolant. Use this solver if the problem is only moderately stiff and you need a solution without numerical damping.

ODE23tb: An implementation of TR-BDF2, an implicit Runge-Kutta formula with a first stage that is a trapezoidal rule step and a second stage that is a backward differentiation formula of order 2. Like ode23s, this solver may be more efficient than ode15s at crude tolerances.

INTEGRATION OF ODE's – MATLAB

Syntax:

```
[T,Y] = solver(odefun,tspan,y0)
[T,Y] = solver(odefun,tspan,y0,options)
[T,Y] = solver(odefun,tspan,y0,options,P1,P2,…)
```

[T,Y] = *solver*(odefun,tspan,y0) with tspan=[t0,tf]. integrates the system of differential equations $y'=f(y,t)$ from time t0 to tf with initial conditions y0. odefun is a function handle. Function f = odefun(t,y), for a scalar t and a column vector y, must return a column vector f corresponding to $f(y,t)$. Each row in the solution array Y corresponds to a time returned in column vector T. To obtain solutions at the specific times t0, t1,...,tf (all increasing or all decreasing), use tspan = [t0,t1,...,tf].

INTEGRATION OF ODE's – MATLAB

Syntax:

```
[T,Y] = solver(odefun,tspan,y0)
[T,Y] = solver(odefun,tspan,y0,options)
[T,Y] = solver(odefun,tspan,y0,options,P1,P2,…)
```

odefun: A function handle that evaluates the right side of the differential equations $y'=f(y,t)$.

tspan: A vector specifying the interval of integration, $[t_0,t_f]$.

y0: A vector of initial conditions.

options: Structure of optional parameters that change the default integration properties. You can change this parameter using the "odeset" function.

Pi: Additional parameters for the odefun.

INTEGRATION OF ODE's – MATLAB

Example – Pendulum equations:

$$\frac{dp}{dt} = -mg\frac{L}{2}\sin\phi \qquad \frac{d\phi}{dt} = \frac{1}{I}p$$

```
% solve_pendulum.m
clear all
[t,x]=ode23('pendulum',[0 3],[0 pi/2]);
plot(t,x(:,2)*180/pi)
```

```
% pendulum.m
function f=pendulum(t,x)
L=1; W=1;g=32.2; I=L^2*W/(3*g);

f(1)=-W*(L/2)*sin(x(2));
f(2)=x(1)/I;
f=f';
```