Solution to Simultaneous/Higher Order Differential Equations Using MATLAB

Numerical integration programs are typically designed to handle first order derivative systems only. To get them to handle higher order systems, we need to reduce their order to first. We do this be creating a system of first order equations. For example, if we consider the second order system (FYI, this is the Van der Pol equation):

$$X'' - X' \bullet (1 - X^2) - X = 0$$

where: $X(0) = 0.25$
$X'(0) = 1$

We can define

$$X_1 = X$$

and

$$X_2 = X'$$

Noting that that

$$X'' = (X')'$$

We can write the second order differential equation as a pair of simultaneous first order differential equations of the form:

$$X_1' = X_2$$

and, isolating the second derivative term,

$$X_2' = X_2 \bullet (1 - X_1^2) - X_1$$

with the initial conditions

$$X_1(0) = 0.25$$

and

$$X_2(0) = 1$$

These equation can now be solved in MATLAB

1. Open MATLAB

2. Convert the right-hand side of the two equations into MATLAB code:

```
x(2)
```

and

```
1-x(1)^2 * x(2) - x(1)
```

These equations will be used in step 4.

3. From within MATLAB, generate an M-file by opening a new file entitled `vdpol2.m` using the file menu.

4. Enter the following code in that file:

```
function xdot=vdpol2(t,x)
xdot = [x(2); 1-x(1)^2 * x(2) - x(1)];
```

Note that the first equation is typed before the semicolen and the second equation is typed after the semicolon.

5. Save this file.

6. Return to the "Command Window."

7. Enter the following code at the >> prompt:

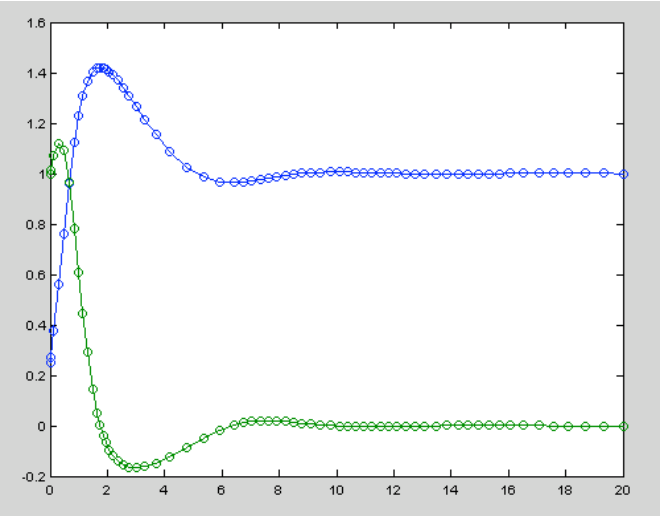
```
ode23('vdpol2',[0 20],[0.25 1]');
```

`[0 20]` refers to the simulation run time from 0 to 20 seconds

`[0.25 1]` refers to our initial conditions for X_1 and X_2. Note the transpose indicated by the `'` symbol.

The help text for the function `ode23` is attached.

8. This should yield the following plot:



ODE23 Solve non-stiff differential equations, low order method. [TOUT,YOUT] = ODE23(ODEFUN,TSPAN,Y0) with TSPAN = [T0 TFINAL] integrates the system of differential equations y' = f(t,y) from time T0 to TFINAL with initial conditions Y0. ODEFUN is a function handle. For a scalar T and a vector Y, ODEFUN(T,Y) must return a column vector corresponding to f(t,y). Each row in the solution array YOUT corresponds to a time returned in the column vector TOUT. To obtain solutions at specific times T0,T1,...,TFINAL (all increasing or all decreasing), use TSPAN = [T0 T1 ... TFINAL]. [TOUT,YOUT] = ODE23(ODEFUN,TSPAN,Y0,OPTIONS) solves as above with default integration properties replaced by values in OPTIONS, an argument created with the ODESET function. See ODESET for details. Commonly used options are scalar relative error tolerance 'RelTol' (1e-3 by default) and vector of absolute error tolerances 'AbsTol' (all components 1e-6 by default). If certain components of the solution must be non-negative, use ODESET to set the 'NonNegative' property to the indices of these components. ODE23 can solve problems M(t,y)*y' = f(t,y) with mass matrix M that is nonsingular. Use ODESET to set the 'Mass' property to a function handle MASS if MASS(T,Y) returns the value of the mass matrix. If the mass matrix is constant, the matrix can be used as the value of the 'Mass' option. If the mass matrix does not depend on the state variable Y and the function MASS is to be called with one input argument T, set 'MStateDependence' to 'none'. ODE15S and ODE23T can solve problems with singular mass matrices.

[TOUT,YOUT,TE,YE,IE] = ODE23(ODEFUN,TSPAN,Y0,OPTIONS) with the 'Events' property in OPTIONS set to a function handle EVENTS, solves as above while also finding where functions of (T,Y), called event functions, are zero. For each function you specify whether the integration is to terminate at a zero and whether the direction of the zero crossing matters. These are the three column vectors returned by EVENTS: [VALUE,ISTERMINAL,DIRECTION] = EVENTS(T,Y). For the I-th event function: VALUE(I) is the value of the function, ISTERMINAL(I)=1 if the integration is to terminate at a zero of this event function and 0 otherwise. DIRECTION(I)=0 if all zeros are to be computed (the default), +1 if only zeros where the event function is increasing, and -1 if only zeros where the event function is decreasing. Output TE is a column vector of times at which events occur. Rows of YE are the corresponding solutions, and indices in vector IE specify which event occurred. SOL = ODE23(ODEFUN,[T0 TFINAL],Y0...) returns a structure that can be used with DEVAL to evaluate the solution or its first derivative at any point between T0 and TFINAL. The steps chosen by ODE23 are returned in a row vector SOL.x. For each I, the column SOL.y(:,I) contains the solution at SOL.x(I). If events were detected, SOL.xe is a row vector of points at which events occurred. Columns of SOL.ye are the corresponding solutions, and indices in vector SOL.ie specify which event occurred.

Example

```
[t,y]=ode23(@vdp1,[0 20],[2 0]);
plot(t,y(:,1));
```

solves the system y' = vdp1(t,y), using the default relative error tolerance 1e-3 and the default absolute tolerance of 1e-6 for each component, and plots the first component of the solution.

Class support for inputs TSPAN, Y0, and the result of ODEFUN(T,Y): float: double, single

See also

other ODE solvers:	ode45, ode113, ode15s, ode23s, ode23t, ode23tb
implicit ODEs:	ode15i
options handling:	odeset, odeget
output functions:	odeplot, odephas2, odephas3, odeprint
evaluating solution:	deval
ODE examples:	rigidode, ballode, orbitode
function handles:	function_handle