School of Mathematics and Statistics
MAST30028 Numerical Methods & Scientific Computing
Week 11

Drag and drop the folder `Week11` from `L: \MAST30028` to `C:\...\MATLAB` and include it in the path. Now MATLAB knows how to find the files in `Week11`.

Exercise Set 1: Runge-Kutta fixed step codes

a. Run the script `myShowRK()` to see the error for fixed-step Runge-Kutta methods of orders 1–5 in solving the problem

$$y' = -y$$

The method of order 1 is Euler's Method.

Have a look at the code in the underlying files `RKStep` and `FixedRK` to see how a Runge-Kutta code could be written.

b. As another example, examine and run the file `blobtrack2` which solves part of a Thermofluids assignment, using fixed-step classical RK4.

Exercise Set 2: Embedded RK methods

a. Have a look through the published demo (an html file) obtained by

`Documentation -> MATLAB -> Examples -> Numerical integration of differential equations`. Do NOT open as Live Script.

It explains how to use `ode23`, `ode45` to solve the Lotka-Volterra (1925) equations, a simple model for predator-prey dynamics.

b. Now try `ode23, ode45` on the problem from Week 10:

$$y' = y^2$$

where $y(0) = 1$, over the interval $[0, 2]$; and see what happens.

Which of fixed-step or variable step RK is more helpful on this problem?

c. A simplified version of `ode23`, called `ode23tx` is explained in Ch. 7 of Moler (which can be downloaded or found in `Week11/odes.pdf`).

To explore its features, work through Section 7.7 of Moler.

d. The file `Kepler.m` contains the functions required to solve the problem of a planet orbiting the Sun (the Kepler problem). The equations describing this are Newton's equations for the x and y components of acceleration:

$$\ddot{x} = -x/r^3; \quad \ddot{y} = -y/r^3; \quad r = \sqrt{x^2 + y^2}$$

Look at `Kepler.m` to see how a system of ODEs is coded in MATLAB for input to the MATLAB solvers. In this case, the system of 2nd order ODEs has to be turned into a system of 1st order ODEs.

The script `ShowKepler` illustrates the use of the built-in Embedded RK solvers `ode23` and `ode45` to solve the Kepler problem.

The output figures illustrate:

- the use of `tspan` to specify the time interval or force output at specified times
- how to plot orbits (phase plots) or solution components
- the performance of methods with different order
- the effect of tighter tolerances
- the range of different timesteps used with these variable-step solvers

Another way to run these solvers is to plot as you solve — *dynamic plot syntax*.

Try `ode23(@Kepler, tSpan, uInitial);` . By default, it plots all the components versus time.

e. In all the MATLAB solvers, you can specify options for the solvers using the command `odeset`, using the template:

```
options = odeset('option_name', option_value,'option_name', option_value .....) ;
[t,y] = solver_name(@funcname, tspan, y0,options);
```


Sometimes the value is a string e.g. `'on'`; sometimes it's a numerical value e.g. `1.e-4`. As an example, one output option is `Stats`, which is off by default. If you set the value to `'on'` the solver reports the numbers of function evaluations etc. during the solution. A useful input option is `'RelTol'`, which sets the relative tolerance you ask for in each component; it is `1.e-3` by default.

Edit `rigidode` to include the following code: e.g. use `edit rigidode` then save to a new M-file or use `type rigidode` then copy & paste to a new M-file.


```
options = odeset('Stats', 'on') ;
ode45(@funcname, tspan, y0,options);
```

How many function evaluations are needed? Now try

```
options = odeset('Stats', 'on','RelTol',1.e-4) ;
ode45(@funcname, tspan, y0,options);
```

Exercise Set 3: Other MATLAB solvers

a. MATLAB has a variable-step variable-order nonstiff solver: `ode113`, an implementation of the Adams-Bashforth-Moulton (ABM) method, using methods of orders 1–13.

   Try `ode113` on the `rigidode` problem, and see which solver becomes more efficient as you specify a smaller tolerance e.g. down to `1.e-7`.

b. MATLAB has several stiff solvers:

   - `ode15s`, a multistep variable-order method of orders 1–5 similar to the BDF methods of Gear ; this is the one I've used the most.
   - `ode23t`, `ode23tb`, which implement various implicit RK methods.
   - `ode23s`, which uses another kind of method not mentioned in MAST30028.

   The Van der Pol oscillator

   $$y'' + y = \mu y'(1 - y^2)$$

   becomes increasingly stiff as the parameter μ becomes large.

   Run `stiffdemo()` which solves this problem with $\mu = 1000$, first with a stiff solver `ode15s`, then with a nonstiff solver `ode45`.

c. Run the code `vdpode` for various values of input argument `mu` to see the effect of μ

   Edit `vdpode` :

   e.g. use `edit vdpode` then save to a new M-file or use `type vdpode` then copy & paste to a new M-file.

   to try `ode45` and `ode113` (nonstiff solvers) on this stiff problem.