Worksheet: Usage of MATLAB's **ode45** and other ODE IVP solvers

Below is a screenshot from `www.mathworks.com/help/matlab/ref/ode45.html`, MATLAB's documentation page on the `ode45` solver. The question is, how do you use it, or other high-quality solvers like the stiff solver `ode23s`? They work for scalar ODEs or systems of ODEs; note that systems are written using column vectors. The first line of the Description tells you much of what you need to know, but it is reasonable to go see the whole page, which includes examples.

ode45

R2019b

Solve nonstiff differential equations — medium order method

collapse all in page

Syntax

```
[t,y] = ode45(odefun,tspan,y0)
[t,y] = ode45(odefun,tspan,y0,options)
[t,y,te,ye,ie] = ode45(odefun,tspan,y0,options)
sol = ode45( ___ )
```

Description

`[t,y] = ode45(odefun,tspan,y0)`, where `tspan = [t0 tf]`, integrates the system of differential equations $y' = f(t, y)$ from `t0` to `tf` with initial conditions `y0`. Each row in the solution array `y` corresponds to a value returned in column vector `t`. example

All MATLAB® ODE solvers can solve systems of equations of the form $y' = f(t, y)$, or problems that involve a mass matrix, $M(t, y)y' = f(t, y)$. The solvers all use similar syntaxes. The `ode23s` solver only can solve problems with a mass matrix if the mass matrix is constant. `ode15s` and `ode23t` can solve problems with a mass matrix that is singular, known as differential-algebraic equations (DAEs). Specify the mass matrix using the `Mass` option of `odeset`.

`ode45` is a versatile ODE solver and is the first solver you should try for most problems. However, if the problem is stiff or requires high accuracy, then there are other ODE solvers that might be better suited to the problem. See Choose an ODE Solver for more information.

`[t,y] = ode45(odefun,tspan,y0,options)` also uses the integration settings defined by `options`, which is an argument created using the `odeset` function. For example, use the `AbsTol` and `RelTol` options to specify absolute and relative error tolerances, or the `Mass` option to provide a mass matrix. example

A. Show how to use `ode45` to solve the scalar ODE IVP

$$y' = t - 10e^{-t}\sin(10y), \qquad y(1) = -2$$

to compute $y(3)$. Then show how to plot the solution $y(t)$ on the interval $1 \le t \le 3$ using labeled t- and y-axes and indicating the points along the solution which were computed by `ode45`.

```
>>

>>

>>

>>

>>

>>
```

B. The system of ODEs

$$R' = (1 - 0.01F)R$$
$$F' = (0.02R - 1)F$$

is an example of a "Lotka-Volterra predator-prey model" as described in Example 11.0.3 (pp. 253–254) in the textbook. Assume initial conditions $R(0) = 20$, $F(0) = 20$. Use `ode45` to solve for, and plot, $R(t)$ and $F(t)$ on the interval $[0, 15]$.

```
>>

>>

>>

>>
```

C. Suppose that for part **B** you write a right-hand side function

```
function dydy = lotka(t,y)
```

in a separate file `lotka.m`. What changes in how you use `ode45`? Why?

D. Here is another system:

$$y_1' = -1000y_1 + y_2$$
$$y_2' = y_1^2 - y_2$$

Assume $\mathbf{y}(0) = \mathbf{y}_0$ is a given vector. Someone claims that this ODE system is stiff so you should use `ode23s`. Show how to solve and plot it on the interval $t \in [0, T]$ if T is given.

```
>>

>>

>>
```

E. To reset the tolerances the best way is to compute an "options structure" using `odeset`, and then give it to `ode45` as an argument. For example, the defaults correspond to doing

```
>> s = odeset('RelTol',1.0e-3,'AbsTol',1.0e-6);
>> [tt,yy] = ode45(f,[t0,tf],y0,s);
```

Redo part **A** with very tight tolerances: `RelTol` $= 10^{-10}$ and `AbsTol` $= 10^{-14}$. Replot the solution; does it look different? How many more points (steps) were used?

```
>>

>>

>>

>>
```