# Mini-Matlab Lesson 11: Matlab's ODE Solvers

## Contents



Matlab has its own suite of built-in ODE solvers which are very effective. In most cases, you will not need to program your own algorithm.

For more information on Matlab's suite of solvers, try typing in the command

```
doc ode23
```

## ODE45

The ODE45 function is essentially a variable-step Runge-Kutta algorithm which can ensure that the error is less than a specified tolerance. It is basically the first solver you should try.

Its standard call requires

```
[TOUT,YOUT] = ODE45(ODEFUN,TSPAN,Y0,OPTIONS)
```

TOUT = independent variable output YOUT = dependent variable output

ODEFUN = two variable function of the form y' = f(x, y) TSPAN = a vector [t0 t1] which specifies the domain Y0 = an initial vector OPTIONS = extra options set using 'optimset'

**Note 1:** The function ODEFUN should return an N x 1 column vector **Note 2:** Consequently, Y0 should be an N x 1 column vector **Note 3:** The final solution TOUT will be, for example, a vector of length 'm' (m is determined by the error control). This means that YOUT is a matrix of size N x m.

## Airy equation

We seek to solve y'' = xy with the initial conditions

$$y(0) = \frac{1}{3^{2/3}\Gamma(2/3)} \text{ and } y'(0) = -\frac{1}{3^{1/3}\Gamma(1/3)}$$

With these initial conditions, the solution should be the well-known Airy function $y = Ai(x)$.

Transforming the ODE into a system, using $y_1 = y$ and $y_2 = y'$ gives

$$y_1' = y_2$$
$$y_2' = xy_1$$

This is used to create the function airyfunc (see incl. handout)

```
yinit = [3^(-2/3)/gamma(2/3); -3^(-1/3)/gamma(1/3)];

[xpos, ypos] = ode45(@airyfunc, [0 5], yinit);  % Solve for x > 0
[xneg, yneg] = ode45(@airyfunc, [0 -10], yinit); % Solve for x < 0

% Combine solutions
x = [flipud(xneg); xpos];
```

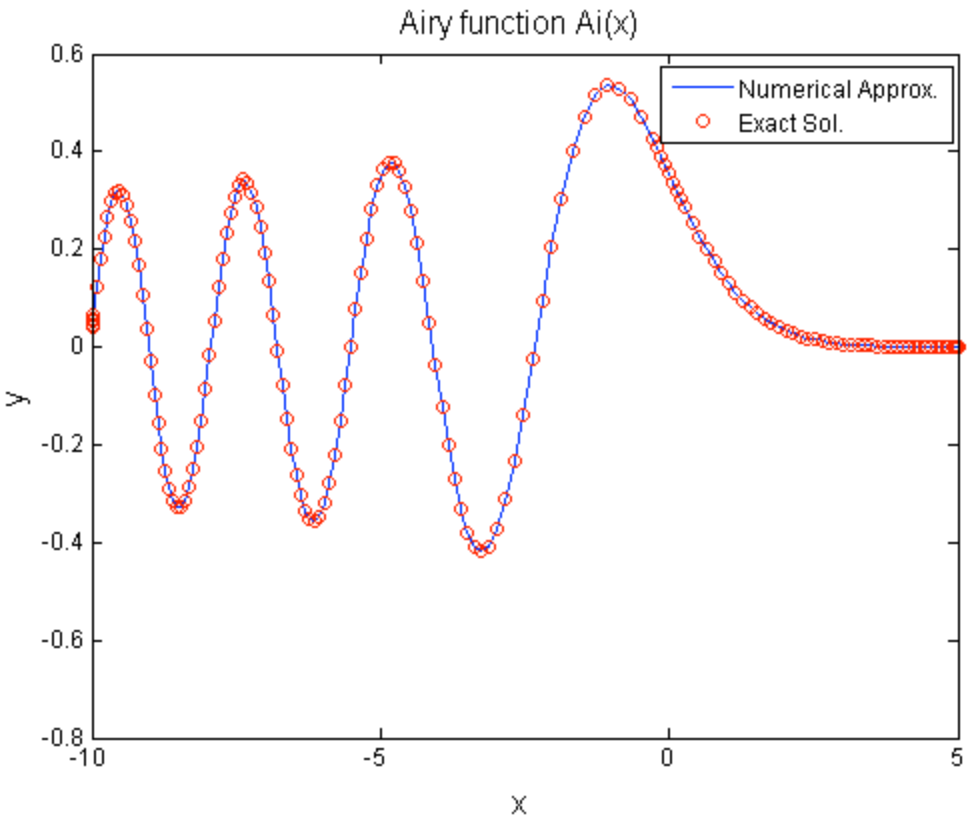
```
y = [flipud(yneg); ypos];

plot(x, y(:,1), 'b');

yexact = airy(x);
hold on
plot(x, real(yexact), 'ro');    % The 'real' is to ignore imaginary errors
hold off;
xlabel('x', 'FontSize', 16);
ylabel('y', 'FontSize', 16);
title('Airy function Ai(x)', 'FontSize', 16);
legend('Numerical Approx.', 'Exact Sol.');

disp(['Error (inf norm) = ', num2str(norm(yexact - y(:,1), inf))]);
```

```
Error (inf norm) = 0.0011336
```




## Changing the options

We can change the options for ODE45 by using the odeset command. The two more important options concern the accuracy. When you perform an optimization, you need to decide when to stop. One way to check for whether your solution is good enough is to check whether the solution is still changing significantly. There are two ways to measure how much a solution changes: relative change (i.e. % change), or absolute change.

**RelTol** is the tolerance on the relative error. Let x(i) be the solution at the ith step. RelTol concerns the condition

$$\frac{|x(i) - x(i-1)|}{|x(i-1)|} < \text{RelTol}$$

That is, how much as the current solution increased, relative to the previous? As long as the solution is not near zero, then RelTol is a good measure of the error. However, if the solution is near zero, then we have x/0.

**AbsTol** is the absolute tolerance, so this one concerns

$$|x(i) - x(i-1)| < \text{AbsTol}$$

This will be used when the solution is small.

```
options = odeset('RelTol', 1e-6, 'AbsTol', 1e-6);

[xpos, ypos] = ode45(@airyfunc, [0 5], yinit, options);
[xneg, yneg] = ode45(@airyfunc, [0 -10], yinit, options);
x = [flipud(xneg); xpos];
y = [flipud(yneg); ypos];
yexact = airy(x);
disp(['Error (inf norm) = ', num2str(norm(yexact - y(:,1), inf))]);
```

```
Error (inf norm) = 9.6066e-06
```

*Published with MATLAB® 7.12*