

Numerical Methods for ODE's II

1. Classical Runge-Kutta Method.

The classical Runge-Kutta method extends the idea of Heun's method. If an approximation of the slope at the right endpoint can be generated by the slope at the left endpoint, then why not use the information at the left endpoint to generate more slope values.

This method does just that. The slope value at the left endpoint usually called k_1 , is used to generate slope information at the midpoint of the interval. This slope is labeled k_2 . This approximation is in turn used to get another approximation at the midpoint. Call it k_3 , which is then used to get an approximation of the slope at the right endpoint, k_4 .

$$k_1 \rightarrow k_2 \rightarrow k_3 \rightarrow k_4$$

So the general formula for the classical 4th order Runge-Kutta (RK-4) method is:

$$y_i = y_{i-1} + h \left(\frac{k_1 + 2k_2 + 2k_3 + k_4}{6} \right)$$

where

$$k_1 = f(t_{i-1}, y_{i-1})$$

$$k_2 = f(t_{i-1} + h/2, y_{i-1} + (h/2)k_1)$$

$$k_3 = f(t_{i-1} + h/2, y_{i-1} + (h/2)k_2)$$

$$k_4 = f(t_{i-1} + h, y_{i-1} + hk_3)$$

Programming the Runge-Kutta method can also be done quickly. Again using the myeuler program as a template, the only section that needs to be changed is where

```
else
    y(i) = y(i-1) + ...
end
```

The values k1, k2, k3, and k4 should be appropriately defined and then

```
else
    k1 = ...
    k2 = ...
    k3 = ...
    k4 = ...
    y(i) = y(i-1) + ...
end
```

The RK-4 method requires 4 times as much work per step as Euler's method and twice as much work per step as Heun's method. Though the work per step is greater, the reduced discretization error makes the RK-4 method a more efficient way to achieve a desired accuracy in a solution.

2. Matlab's build in functions

Matlab has 2 build in functions to solve IVP problems. **ode23** and **ode45** are built in functions that use adaptive stepsize algorithms as well as Runge-Kutta Methods to solve an IVP. With a carefully chosen set of substeps in the interval of size h , these methods simultaneously obtain 2 solutions with different discretization errors at each step. This allows the user to monitor the accuracy of the solution. The **ode23** routine uses 2nd and 3rd order Runge-Kutta formulas to compute the solution and monitor the accuracy. Similarly, **ode45** uses 4th and 5th order Runge-Kutta formulas.

They may be called in the following ways:

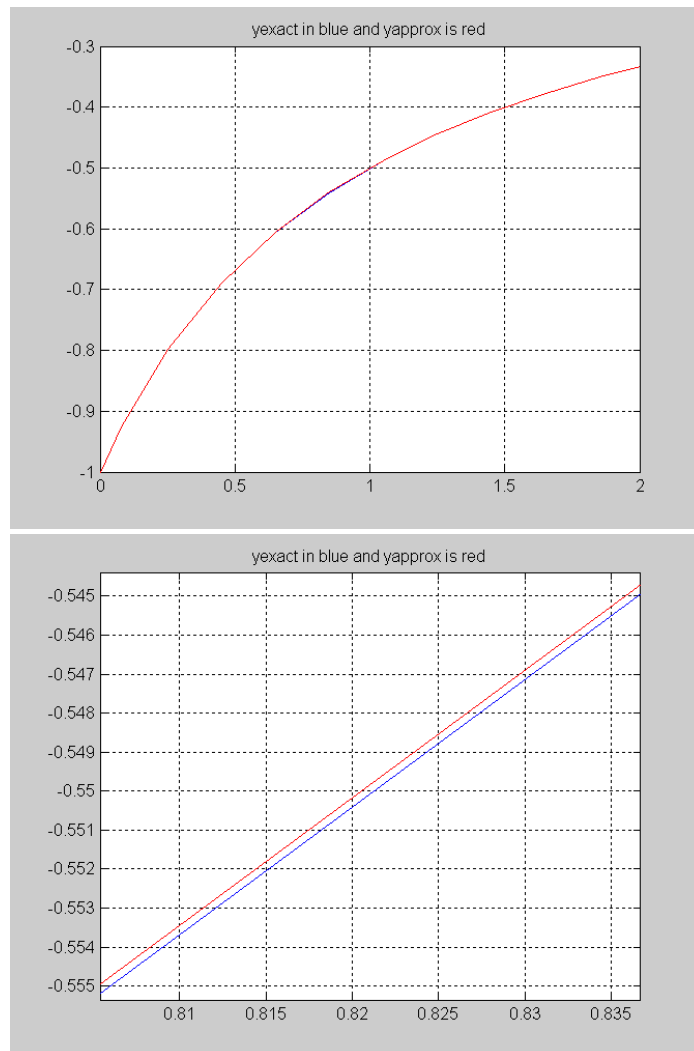
```
>> [t,y] = ode23(diffeq, tn, y0)
>> [t,y] = ode23(diffeq, [t0,tn], y0)
>> [t,y] = ode23(diffeq, [t0,tn], y0,options)
>> [t,y] = ode23(diffeq, [t0,tn], y0,options, arg1,arg2,...)
```

ode45 uses the same parameter sequence. **diffeq** is the name of an m-file that returns the right-hand side of the differential equation. The **t0** and **tn** define the overall interval upon which to find a solution. If **t0** is not given, then it is assumed to be zero. For more help on options and arguments, type **help ode45** or **help ode23**.

Example: Solve the IVP $y' = y^2$ over the interval $[0,2]$ using **ode23**:
 $y(0) = -1$

```
>> tf = 2; y0 = -1;
>> [t,y] = ode23('rhsfile',tf,y0);
>> yexact = -1./(t+1);

>> plot(t,yexact, 'b', t, y, 'r')
>> grid
>> title('yexact in blue and yapprox is red')
```



3. Systems of Differential Equations

a. System of 1st-order ODE's

So far, we have only talked about individual, first-order ODE's. We now extend to coupled **systems** of first-order ODE's:

$$\frac{dy_1}{dt} = f_1(t, y_1, y_2)$$

$$\frac{dy_2}{dt} = f_2(t, y_1, y_2)$$

Since the equation for y_1 explicitly depends on y_2 and vice versa, these equations are a coupled system of equations.

We will use the notation $k_{1,i}$ to denote the first slope for the i th equation, $k_{2,i}$ to denote the 2nd slope for the i th equation, etc.

Then for the ODE methods discussed earlier, you have to compute the intermediate slopes for both equations simultaneously. For example, in RK-4 to advance to the j th step (the values at step $j-1$ are known), first compute

$$k_{1,1} = f_1(t_j, y_{j,1}, y_{j,2})$$

$$k_{1,2} = f_2(t_j, y_{j,1}, y_{j,2})$$

then

$$k_{2,1} = f_1(t_j + (h/2), y_{j,1} + (h/2)k_{1,1}, y_{j,2} + (h/2)k_{1,2})$$

$$k_{2,2} = f_2(t_j + (h/2), y_{j,1} + (h/2)k_{1,1}, y_{j,2} + (h/2)k_{1,2})$$

and so on.

Note that both $k_{1,1}$ and $k_{1,2}$ must be computed before any of the terms of the form $k_{2,i}$ can be determined.

b. Single Higher Order ODE's

To solve a higher order ODE, the equation must be mathematically transformed into an equivalent system of first-order ODE's.

Example: Transform the 2nd-order ODE, $\frac{d^2z}{dt^2} + \alpha \frac{dz}{dt} + \beta z = f(t)$, into a system of 1st-order ODE's:

We introduce new variables: $y_1 \equiv z$ and $y_2 \equiv \frac{dz}{dt}$. Then $\frac{dy_1}{dt} = \frac{dz}{dt} = y_2$ and

$\frac{dy_2}{dt} = \frac{d^2z}{dt^2} = f(t) - \alpha \frac{dz}{dt} - \beta z = f(t) - \alpha y_2 - \beta y_1$. So the coupled system of 1st-order ODE's is

$$\frac{dy_1}{dt} = y_2$$

$$\frac{dy_2}{dt} = f(t) - \alpha y_2 - \beta y_1$$

This system can also be put into matrix form:

$$y' = f(t, y) \text{ where } y' = \begin{pmatrix} \frac{dy_1}{dt} \\ \frac{dy_2}{dt} \end{pmatrix} \text{ and } f(t, y) = \begin{pmatrix} f_1 \\ f_2 \end{pmatrix} = \begin{pmatrix} y_2 \\ f(t) - \alpha y_2 - \beta y_1 \end{pmatrix}$$

In this case, the initial condition would be of the form $y(0) = y_0$ where

$y(0) = \begin{pmatrix} y_1(0) \\ y_2(0) \end{pmatrix}$ and y_0 is a column vector containing the initial conditions for

equation 1 and equation 2.

4. pplane7

pplane7 is a program developed by John C. Polking from the Department of Mathematics at Rice University. Pplane7 is an interactive tool for studying systems of differential equations. When pplane7 is executed, a setup window is opened. The user may enter the differential equations and specify a display window using the interactive controls in the setup window. When the Proceed button is pressed on the setup window, a display window is opened, and a field of the type requested is displayed for the system. When the mouse button is depressed in the display window, the solution to the system with that initial condition is calculated and plotted. The initial conditions may also be input via a menu option. For more information visit his website at <http://math.rice.edu/~dfield/>.