Runge-Kutta method

The formula for the fourth order Runge-Kutta method (RK4) is given below. Consider the problem

$$\begin{cases} y' = f(t, y) \\ y(t_0) = \alpha \end{cases}$$

Define h to be the time step size and $t_i = t_0 + ih$. Then the following formula

$$w_0 = \alpha$$

$$k_1 = hf(t_i,\ w_i)$$

$$k_2 = hf\left(t_i + \frac{h}{2},\ w_i + \frac{k_1}{2}\right)$$

$$k_3 = hf\left(t_i + \frac{h}{2},\ w_i + \frac{k_2}{2}\right)$$

$$k_4 = hf(t_i + h,\ w_i + k_3)$$

$$w_{i+1} = w_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

computes an approximate solution, that is $w_i \approx y(t_i)$.

Let us look at an example:

$$\begin{cases} y' = y - t^2 + 1 \\ y(0) = 0.5 \end{cases}$$

The exact solution for this problem is $y = t^2 + 2t + 1 - \frac{1}{2}e^t$, and we are interested in the value of y for $0 \leq t \leq 2$.

1. We first solve this problem using RK4 with $h = 0.5$. From $t = 0$ to $t = 2$ with step size $h = 0.5$, it takes 4 steps: $t_0 = 0$, $t_1 = 0.5$, $t_2 = 1$, $t_3 = 1.5$, $t_4 = 2$.

Step 0 $t_0 = 0$, $w_0 = 0.5$.

Step 1 $t_1 = 0.5$

$$\begin{aligned}
k_1 &= hf(t_0, w_0) = 0.5f(0, 0.5) = 0.75 \\
k_2 &= hf(t_0 + h/2, w_0 + k_1/2) = 0.5f(0.25, 0.875) = 0.90625 \\
K_3 &= hf(t_0 + h/2, w_0 + k_2/2) = 0.5f(0.25, 0.953125) = 0.9453125 \\
K_4 &= hf(t_0 + h, w_0 + K_3) = 0.5f(0.5, 1.4453125) = 1.09765625 \\
w_1 &= w_0 + (k_1 + 2k_2 + 2k_3 + k_4)/6 = 1.425130208333333
\end{aligned}$$

Step 2 $t_2 = 1$

$$\begin{aligned}
k_1 &= hf(t_1, w_1) = 0.5f(0.5, 1.425130208333333) = 1.087565104166667 \\
k_2 &= hf(t_1 + h/2, w_1 + k_1/2) = 0.5f(0.75, 1.968912760416667) = 1.203206380208333 \\
K_3 &= hf(t_1 + h/2, w_1 + k_2/2) = 0.5f(0.75, 2.0267333984375) = 1.23211669921875 \\
K_4 &= hf(t_1 + h, w_1 + K_3) = 0.5f(1, 2.657246907552083) = 1.328623453776042 \\
w_2 &= w_1 + (k_1 + 2k_2 + 2k_3 + k_4)/6 = 2.639602661132812
\end{aligned}$$

Step 3 $t_3 = 1.5$

$$k_1 = hf(t_2, w_2) = 0.5f(1, 2.639602661132812) = 1.319801330566406$$
$$k_2 = hf(t_2 + h/2, w_2 + k_1/2) = 0.5f(1.25, 3.299503326416016) = 1.368501663208008$$
$$K_3 = hf(t_2 + h/2, w_2 + k_2/2) = 0.5f(1.25, 3.323853492736816) = 1.380676746368408$$
$$K_4 = hf(t_2 + h, w_2 + K_3) = 0.5f(1.5, 4.020279407501221) = 1.385139703750610$$
$$w_3 = w_2 + (k_1 + 2k_2 + 2k_3 + k_4)/6 = 4.006818970044454$$

Step 4 $t_4 = 2$

$$k_1 = hf(t_3, w_3) = 0.5f(1.5, 4.006818970044454) = 1.378409485022227$$
$$k_2 = hf(t_3 + h/2, w_3 + k_1/2) = 0.5f(1.75, 4.696023712555567) = 1.316761856277783$$
$$K_3 = hf(t_3 + h/2, w_3 + k_2/2) = 0.5f(1.75, 4.665199898183346) = 1.301349949091673$$
$$K_4 = hf(t_3 + h, w_3 + K_3) = 0.5f(2, 5.308168919136127) = 1.154084459568063$$
$$w_4 = w_3 + (k_1 + 2k_2 + 2k_3 + k_4)/6 = 5.301605229265987$$

Now let's compare what we got with the exact solution

t_i	Exact solution $y(t_i)$	Numerical solution w_i	Error $\lvert w_i - y(t_i)\rvert$
0.0	0.5	0.5	0
0.5	1.425639364649936	1.425130208333333	0.000509156316603
1.0	2.640859085770477	2.639602661132812	0.001256424637665
1.5	4.009155464830968	4.006818970044454	0.002336494786515
2.0	5.305471950534675	5.301605229265987	0.003866721268688

All this can be done by using Matlab:

```
function rungekutta
  h = 0.5;
  t = 0;
  w = 0.5;
  fprintf('Step 0: t = %12.8f, w = %12.8f\n', t, w);
  for i=1:4
    k1 = h*f(t,w);
    k2 = h*f(t+h/2, w+k1/2);
    k3 = h*f(t+h/2, w+k2/2);
    k4 = h*f(t+h, w+k3);
    w = w + (k1+2*k2+2*k3+k4)/6;
    t = t + h;
    fprintf('Step %d: t = %6.4f, w = %18.15f\n', i, t, w);
  end

%%%%%%%%%%%%%%%%%%
function v = f(t,y)
  v = y-t^2+1;
```

2. Solve the problem using RK4 with $h = 0.2$. All you need to do is to replace `h = 0.5;` and `for i=1:4` in the above Matlab program into `h = 0.2` and `for i=1:10`. Then we have

t_i	Exact solution $y(t_i)$	Numerical solution w_i	Error $\|w_i - y(t_i)\|$
0.0	0.5	0.5	0
0.2	0.829298620919915	0.829293333333333	0.000005287586582
0.4	1.214087651179365	1.214076210666667	0.000011440512698
0.6	1.648940599804746	1.648922017041600	0.000018582763146
0.8	2.127229535753766	2.127202684947944	0.000026850805823
1.0	2.640859085770477	2.640822692728752	0.000036393041726
1.2	3.179941538631726	3.179894170232231	0.000047368399496
1.4	3.732400016577663	3.732340072854980	0.000059943722683
1.6	4.283483787802442	4.283409498318406	0.000074289484036
1.8	4.815176267793527	4.815085694579435	0.000090573214092
2.0	5.305471950534674	5.305363000692655	0.000108949842019

3. Solve the problem using RK4 with $h = 0.05$. Again, all need to do is to set `h = 0.05` and `for i=1:40`. Then we have

t_i	Exact solution $y(t_i)$	Numerical solution w_i	Error $\|w_i - y(t_i)\|$
0.0	0.5	0.5	0
0.05	0.576864451811988	0.576864446614583	0.000000005197405
0.10	0.657414540962176	0.657414530368210	0.000000010593966
...	...	...	...
1.80	4.815176267793529	4.815175898599096	0.000000369194433
1.85	4.942590238699086	4.942589852008494	0.000000386690592
1.90	5.067052778860367	5.067052374183828	0.000000404676539
1.95	5.188156209705356	5.188155786548850	0.000000423156505
2.00	5.305471950534677	5.305471508400809	0.000000442133868

4. Comparing the results By comparing the above results, we can see that

$$\begin{aligned}
&\text{for } h = 0.5, &&\text{the error at } t = 2 \text{ is } 0.003866721268688, &&\text{using } 4 \text{ steps}\\
&\text{for } h = 0.2, &&\text{the error at } t = 2 \text{ is } 0.000108949842019, &&\text{using } 10 \text{ steps}\\
&\text{for } h = 0.05, &&\text{the error at } t = 2 \text{ is } 0.000000442133868, &&\text{using } 40 \text{ steps}
\end{aligned}$$

This brings out a question. How to find the most appropriate step size, if we want to make sure the error is less than a given ε, for example, $\varepsilon = 0.00001$?

5. Adaptive step size control and the Runge-Kutta-Fehlberg method The answer is, we will use adaptive step size control during the computation. The idea is to start with a moderate step size. When we detect the expected error is larger than ε, reduce the step size and recalculate the current step. When we detect the expected error is less than ε, keep the current step and slightly enlarge the step size in the next step. This requires us to have a good estimation of the "expected error".

Here's the formula for the Runge-Kutta-Fehlberg method (RK45).

$$
\begin{aligned}
w_0 &= \alpha \\
k_1 &= hf(t_i,\, w_i) \\
k_2 &= hf\left(t_i + \frac{h}{4},\, w_i + \frac{k_1}{4}\right) \\
k_3 &= hf\left(t_i + \frac{3h}{8},\, w_i + \frac{3}{32}k_1 + \frac{9}{32}k_2\right) \\
k_4 &= hf\left(t_i + \frac{12h}{13},\, w_i + \frac{1932}{2197}k_1 - \frac{7200}{2197}k_2 + \frac{7296}{2197}k_3\right) \\
k_5 &= hf\left(t_i + h,\, w_i + \frac{439}{216}k_1 - 8k_2 + \frac{3680}{513}k_3 - \frac{845}{4104}k_4\right) \\
k_6 &= hf\left(t_i + \frac{h}{2},\, w_i - \frac{8}{27}k_1 + 2k_2 - \frac{3544}{2565}k_3 + \frac{1859}{4104}k_4 - \frac{11}{40}k_5\right) \\
w_{i+1} &= w_i + \frac{25}{216}k_1 + \frac{1408}{2565}k_3 + \frac{2197}{4104}k_4 - \frac{1}{5}k_5 \\
\tilde{w}_{i+1} &= w_i + \frac{16}{135}k_1 + \frac{6656}{12825}k_3 + \frac{28561}{56430}k_4 - \frac{9}{50}k_5 + \frac{2}{55}k_6 \\
R &= \frac{1}{h}|\tilde{w}_{i+1} - w_{i+1}| \\
\delta &= 0.84\left(\frac{\varepsilon}{R}\right)^{1/4}
\end{aligned}
$$

if $R \le \varepsilon$ keep w as the current step solution
and move to the next step with step size δh

if $R > \varepsilon$ recalculate the current step with step size δh

The Matlab program is given below:

```
function rk45
  epsilon = 0.00001;
  h = 0.2;
  t = 0;
  w = 0.5;
  i = 0;
  fprintf('Step %d: t = %6.4f, w = %18.15f\n', i, t, w);
  while t<2
    h = min(h, 2-t);

    k1 = h*f(t,w);
```

```
    k2 = h*f(t+h/4, w+k1/4);
    k3 = h*f(t+3*h/8, w+3*k1/32+9*k2/32);
    k4 = h*f(t+12*h/13, w+1932*k1/2197-7200*k2/2197+7296*k3/2197);
    k5 = h*f(t+h, w+439*k1/216-8*k2+3680*k3/513-845*k4/4104);
    k6 = h*f(t+h/2, w-8*k1/27+2*k2-3544*k3/2565+1859*k4/4104-11*k5/40);
    w1 = w + 25*k1/216+1408*k3/2565+2197*k4/4104-k5/5;
    w2 = w + 16*k1/135+6656*k3/12825+28561*k4/56430-9*k5/50+2*k6/55;
    R = abs(w1-w2)/h;
    delta = 0.84*(epsilon/R)^(1/4);

    if R<=epsilon
      t = t+h;
      w = w1;
      i = i+1;
      fprintf('Step %d: t = %6.4f, w = %18.15f\n', i, t, w);
      h = delta*h;
    else
      h = delta*h;
    end
  end

%%%%%%%%%%%%%%%%%%
function v = f(t,y)
  v = y-t^2+1;
```

Run this program, we see that starting with $h = 0.2$, the program outputs

```
Step 0: t = 0.0000, w =  0.500000000000000
Step 1: t = 0.2000, w =  0.829299076923077
Step 2: t = 0.4353, w =  1.287432405787216
Step 3: t = 0.6766, w =  1.827289794651997
Step 4: t = 0.9264, w =  2.448301479233138
Step 5: t = 1.1902, w =  3.153049280338359
Step 6: t = 1.4806, w =  3.955581050460808
Step 7: t = 1.8537, w =  4.952039512278185
Step 8: t = 2.0000, w =  5.305486816572746
```

Notice the error is

$$|y(2) - w_8| = 0.0000148660380707710\,3$$