MATHEMATICS

Help with using the Runge-Kutta 4th order method on a system of 2 first order ODE's.

Asked 6 years, 2 months ago Active 1 year, 4 months ago Viewed 97k times

28

21

The original ODE I had was

$$\frac{d^2y}{dx^2} + \frac{dy}{dx} - 6y = 0$$

with $y(0) = 3$ and $y'(0) = 1$. Now I can solve this by hand and obtain that $y(1) = 14.82789927$. However I wish to use the 4th order Runge-Kutta method, so I have the system:

$$\begin{cases} \frac{dy}{dx} = z \\ \frac{dz}{dx} = 6y - z \end{cases}$$

With $y(0) = 3$ and $z(0) = 1$.

Now I know that for two general 1st order ODE's

$$\frac{dy}{dx} = f(x, y, z)$$
$$\frac{dz}{dx} = g(x, y, z)$$

The 4th order Runge-Kutta formula's for a system of 2 ODE's are:

$$y_{i+1} = y_i + \frac{1}{6}(k_0 + 2k_1 + 2k_2 + k_3)$$

Where

$$k_0 = hf(x_i, y_i, z_i)$$
$$k_1 = hf(x_i + \frac{1}{2}h, y_i + \frac{1}{2}k_0, z_i + \frac{1}{2}l_0)$$
$$k_2 = hf(x_i + \frac{1}{2}h, y_i + \frac{1}{2}k_1, z_i + \frac{1}{2}l_1)$$
$$k_3 = hf(x_i + h, y_i + k_2, z_i + l_2)$$

and

$$l_0 = hg(x_i, y_i, z_i)$$
$$l_1 = hg(x_i + \frac{1}{2}h, y_i + \frac{1}{2}k_0, z_i + \frac{1}{2}l_0)$$
$$l_2 = hg(x_i + \frac{1}{2}h, y_i + \frac{1}{2}k_1, z_i + \frac{1}{2}l_1)$$
$$l_3 = hg(x_i + h, y_i + k_2, z_i + l_2)$$

My problem is I am struggling to apply this method to my system of ODE's so that I can program a method that can solve any system of 2 first order ODE's using the formulas above, I would like for someone to please run through one step of the method, so I can understand it better.

ordinary-differential-equations numerical-methods systems-of-equations runge-kutta-methods

edited Mar 23 '18 at 23:42
Rodrigo de Azevedo
15.3k 4 27 71

asked Mar 21 '14 at 12:21
Michael
365 1 4 10

For reference, see this answer on SO. – ja72 Mar 24 '18 at 15:22

4 Answers

Active Oldest Votes

37

I will outline the process and you can fill in the calculations.

We have our system as:

$$\begin{cases} \frac{dy}{dx} = z \\ \frac{dz}{dx} = 6y - z \end{cases}$$

With $y(0) = 3$ and $z(0) = 1$.

We must do the calculations in a certain order as there are dependencies between the numerical calculations. This order is:

- $k_0 = hf(x_i, y_i, z_i)$
- $l_0 = hg(x_i, y_i, z_i)$
- $k_1 = hf(x_i + \frac{1}{2}h, y_i + \frac{1}{2}k_0, z_i + \frac{1}{2}l_0)$
- $l_1 = hg(x_i + \frac{1}{2}h, y_i + \frac{1}{2}k_0, z_i + \frac{1}{2}l_0)$
- $k_2 = hf(x_i + \frac{1}{2}h, y_i + \frac{1}{2}k_1, z_i + \frac{1}{2}l_1)$

- $k_3 = hf(x_i + h, y_i + k_2, z_i + l_2)$
- $l_3 = hg(x_i + h, y_i + k_2, z_i + l_2)$
- $y_{i+1} = y_i + \frac{1}{6}(k_0 + 2k_1 + 2k_2 + k_3)$
- $z_{i+1} = z_i + \frac{1}{6}(l_0 + 2l_1 + 2l_2 + l_3)$

We typically need some inputs for the algorithm:

- A range that we want to do the calculations over: $a \le t \le b$, lets use $a = 0, b = 1$.
- The number of steps N, say $N = 10$.
- The steps size $h = \dfrac{b-a}{N} = \dfrac{1}{10}$

The system we are solving is:

$$\frac{dy}{dx} = f(x, y, z) = z$$
$$\frac{dz}{dx} = g(x, y, z) = 6y - z$$

Doing the calculations using the above order for the first time step $i = 0, t_0 = 0 = x_0$, yields:

- $k_0 = hf(x_0, y_0, z_0) = \dfrac{1}{10}(z_0) = \dfrac{1}{10}(1) = \dfrac{1}{10}$
- $l_0 = hg(x_0, y_0, z_0) = \dfrac{1}{10}(6y_0 - z_0) = \dfrac{1}{10}(6 \times 3 - 1) = \dfrac{1}{10}(17)$
- $k_1 = hf(x_0 + \frac{1}{2}h, y_0 + \frac{1}{2}k_0, z_0 + \frac{1}{2}l_0) = \dfrac{1}{10}(1 + \dfrac{1}{2}\dfrac{1}{10}(17))$ (You please continue the calcs.)
- $l_1 = hg(x_0 + \frac{1}{2}h, y_i + \frac{1}{2}k_0, z_0 + \frac{1}{2}l_0)$
- $k_2 = hf(x_0 + \frac{1}{2}h, y_0 + \frac{1}{2}k_1, z_0 + \frac{1}{2}l_1)$
- $l_2 = hg(x_0 + \frac{1}{2}h, y_0 + \frac{1}{2}k_1, z_0 + \frac{1}{2}l_1)$
- $k_3 = hf(x_0 + h, y_0 + k_2, z_0 + l_2)$
- $l_3 = hg(x_0 + h, y_0 + k_2, z_0 + l_2)$
- $y_1 = y_0 + \frac{1}{6}(k_0 + 2k_1 + 2k_2 + k_3)$
- $z_1 = z_0 + \frac{1}{6}(l_0 + 2l_1 + 2l_2 + l_3)$

You now have x_1 and z_1 which you need for the next time step after all of the intermediate (in order again). Now, we move on to the next time step $i = 1, t_1 = t_0 + h = \dfrac{1}{10} = x_1$, so we have:

- $k_0 = hf(x_1, y_1, z_1) = \dfrac{1}{10}(z_1)$
- $l_0 = hg(x_1, y_1, z_1) = \dfrac{1}{10}(6y_1 - z_1)$
- $k_1 = hf(x_1 + \frac{1}{2}h, y_1 + \frac{1}{2}k_0, z_1 + \frac{1}{2}l_0)$
- $l_1 = hg(x_1 + \frac{1}{2}h, y_1 + \frac{1}{2}k_0, z_1 + \frac{1}{2}l_0)$
- $k_2 = hf(x_1 + \frac{1}{2}h, y_1 + \frac{1}{2}k_1, z_1 + \frac{1}{2}l_1)$
- $l_2 = hg(x_1 + \frac{1}{2}h, y_1 + \frac{1}{2}k_1, z_1 + \frac{1}{2}l_1)$

- $l_3 = hg(x_1 + h, y_1 + k_2, z_1 + l_2)$
- $y_2 = y_1 + \frac{1}{6}(k_0 + 2k_1 + 2k_2 + k_3)$
- $z_2 = z_1 + \frac{1}{6}(l_0 + 2l_1 + 2l_2 + l_3)$

Continue this for 10 time steps. Your final result should match closely (assuming the numerical algorithm is stable for this problem) to the exact solution. You will compare z_{10} to the exact result. The exact solution is:

$$y(x) = e^{-3x} + 2e^{2x}$$

If we find $y(1) = \dfrac{1}{e^3} + 2e^2 = 14.827899266229164397440 1973\ldots$.

edited Mar 22 '14 at 0:25 answered Mar 21 '14 at 15:36

Amzoti
53.5k 12 64 103

Thanks for your thorough response, seeing the start of it I now understand it better. Thanks! – Michael Mar 21 '14 at 15:49

1 Recall, in your new system, the first equation $y' = z$ is just a dummy variable in order to use RK4 methods. Regards – Amzoti Mar 21 '14 at 15:53

1 @Michael: Also, you will clearly see when you calculate y_i, z_i, which is the correct final result. – Amzoti Mar 21 '14 at 16:02

1 Late reply but, is y_i or z_i the solution to the original ODE? Comparing values it seems like the solution is given by y_i, but I'm not sure. – Erik Vesterlund May 4 '16 at 15:47

1 @ErikVesterlund if I got this right, then z would be the solution for the derivative of y and y is the solution to the original ODE – lucidbrot Jan 22 '17 at 16:11

10

Although this answer contains the same content as Amzoti's answer, I think it's worthwhile to see it another way.

In general consider if you had m first-order ODE's (after appropriate decomposition). The system looks like

$$\frac{dy_1}{dx} = f_1(x, y_1, \ldots, y_m)$$
$$\frac{dy_2}{dx} = f_2(x, y_1, \ldots, y_m)$$
$$\vdots$$
$$\frac{dy_m}{dx} = f_m(x, y_1, \ldots, y_m)$$

Define the vectors $\vec{Y} = (y_1, \ldots, y_m)$ and $\vec{f} = (f_1, \ldots, f_m)$, then we can write the system as

$$\frac{d}{dx}\vec{Y} = \vec{f}(x, \vec{Y})$$

Now we can generalize the RK method by defining

$$\vec{k}_1 = h\vec{f}\,(x_n, \vec{Y}(x_n))$$
$$\vec{k}_2 = h\vec{f}\,(x_n + \tfrac{1}{2}h, \vec{Y}(x_n) + \tfrac{1}{2}\vec{k}_1)$$
$$\vec{k}_3 = h\vec{f}\,(x_n + \tfrac{1}{2}h, \vec{Y}(x_n) + \tfrac{1}{2}\vec{k}_2)$$
$$\vec{k}_4 = h\vec{f}\,(x_n + h, \vec{Y}(x_n) + \vec{k}_3)$$

and the solutions are then given by

$$\vec{Y}(x_{n+1}) = \vec{Y}(x_n) + \tfrac{1}{6}(\vec{k}_1 + 2\vec{k}_2 + 2\vec{k}_3 + \vec{k}_4)$$

with m initial conditions specified by $\vec{Y}(x_0)$. When writing a code to implement this one can simply use arrays, and write a function to compute $\vec{f}\,(x, \vec{Y})$

For the example provided, we have $\vec{Y} = (y, z)$ and $\vec{f} = (z, 6y - z)$. Here's an example in Fortran90:

```
program RK4
    implicit none
    integer , parameter :: dp = kind(0.d0)
    integer , parameter :: m = 2 ! order of ODE
    real(dp) :: Y(m)
    real(dp) :: a, b, x, h
    integer :: N, i

    ! Number of steps
    N = 10

    ! initial x
    a = 0
    x = a

    ! final x
    b = 1

    ! step size
    h = (b-a)/N

    ! initial conditions
    Y(1) = 3 ! y(0)
    Y(2) = 1 ! y'(0)

    ! iterate N times
    do i = 1,N
        Y = iterate(x, Y)
        x = x + h
    end do

    print*, Y


contains

    ! function f computes the vector f

    function f(x, Yvec) result (fvec)
        real(dp) :: x
        real(dp) :: Yvec(m), fvec(m)

        fvec(1) = Yvec(2) !z
        fvec(2) = 6*Yvec(1) - Yvec(2) !6y-z

    end function
```

```
    function iterate(x, Y_n) result (Y_nplus1)
        real(dp) :: x
        real(dp) :: Y_n(m), Y_nplus1(m)
        real(dp) :: k1(m), k2(m), k3(m), k4(m)

        k1 = h*f(x, Y_n)
        k2 = h*f(x + h/2, Y_n + k1/2)
        k3 = h*f(x + h/2, Y_n + k2/2)
        k4 = h*f(x + h, Y_n + k3)

        Y_nplus1 = Y_n + (k1 + 2*k2 + 2*k3 + k4)/6

    end function

end program
```

This can be applied to any set of m first order ODE's, just change `m` in the code and change the function `f` to whatever is appropriate for the system of interest. Running this code as-is yields

14.827578509968953 29.406156886687729

The first value is $y(1)$, the second $z(1)$, correct to the third decimal point with only ten steps.

edited Nov 4 '18 at 0:19

answered Mar 23 '18 at 23:29

Kai

495 4 10

1 you should use x_n instead of t_n – tnt235711 Nov 3 '18 at 20:39

Good catch, fixed it – Kai Nov 4 '18 at 0:20

Fantastic answer @Kai +1. Would give +50 if possible! Many people struggle with systems of ODE's and RK methods. I have a question though regarding your Fortran implementation. If you wanted to be fancy you could write your k_i's using a for loop correct? Essentially placing them in an array? So you would have an array $k(i, n)$ where i was the number of stages and n was the dimension of your state vector? Are you aware of any documentation that does this in Fortran? I am writing something similar at the minute and am a bit stumped!! – Rumplestillskin Feb 10 '19 at 0:03

4

A Matlab implementation is given below:

```
% It calculates ODE using Runge-Kutta 4th order method
% Author Ido Schwartz
% Originally available form:
http://www.mathworks.com/matlabcentral/fileexchange/29851-runge-kutta-4th-order-
ode/content/Runge_Kutta_4.m
% Edited by Amin A. Mohammed, for 2 ODEs(April 2016)

clc;                                                 % Clears the screen
clear all;

h=0.1;                                               % step size
x = 0:h:1;                                           % Calculates upto y(1)
y = zeros(1,length(x));
z = zeros(1,length(x));
y(1) = 3;                                            % initial condition
z(1) = 1;                                            % initial condition
% F_xy = @(t,r) 3.*exp(-t)-0.4*r;                    % change the function as you
desire
F_xyz = @(x,y,z) z;                                    % change the function as you
desire
G_xyz = @(x,y,z) 6*y-z;
```

```
    L_1 = G_xyz(x(i),y(i),z(i));
    k_2 = F_xyz(x(i)+0.5*h,y(i)+0.5*h*k_1,z(i)+0.5*h*L_1);
    L_2 = G_xyz(x(i)+0.5*h,y(i)+0.5*h*k_1,z(i)+0.5*h*L_1);
    k_3 = F_xyz((x(i)+0.5*h),(y(i)+0.5*h*k_2),(z(i)+0.5*h*L_2));
    L_3 = G_xyz((x(i)+0.5*h),(y(i)+0.5*h*k_2),(z(i)+0.5*h*L_2));
    k_4 = F_xyz((x(i)+h),(y(i)+k_3*h),(z(i)+L_3*h)); % Corrected
    L_4 = G_xyz((x(i)+h),(y(i)+k_3*h),(z(i)+L_3*h));

    y(i+1) = y(i) + (1/6)*(k_1+2*k_2+2*k_3+k_4)*h;  % main equation
    z(i+1) = z(i) + (1/6)*(L_1+2*L_2+2*L_3+L_4)*h;  % main equation

end
```

edited Jul 3 '16 at 10:50

Community ♦ 1

answered Apr 30 '16 at 8:09

AlFagera 143 5

0

A Fortran code shown below:

```
!Runge-Kutta Forth Order Method
!For 2nd Order Differentiation Equation
!First you have to define the function
F(x,y,z) = z !dy/dx
G(x,y,z) = 6*y-z !dz/dx = d2y/dx2
 INTEGER :: n,i
 REAL :: k1,l1,k2,l2,k3,l3,k4,l4 !Most Important
 Write (*,*) "Given Equation '(y2)-6(y1)+(y0)=0'"
 Write (*,*) "Xo=0, Yo=3, Zo=Y'o=1, Xn=1, n=?"
 Xo=0    !Given Condition
 Yo=3    !Given Condition
 Zo=1    !Given Condition
 Xn=1    !Given Condition
 Read (*,*) n   !n=number of Intercept
 h=(Xn-Xo)/n
 DO i=1,n   !you have to do the CAlculation 'n' times
    k1 = h*F(Xo,Yo,Zo)
    l1 = h*G(Xo,Yo,Zo)
    k2 = h*F(Xo+h/2,Yo+k1/2,Zo+l1/2)
    l2 = h*G(Xo+h/2,Yo+k1/2,Zo+l1/2)
    k3 = h*F(Xo+h/2,Yo+k2/2,Zo+l2/2)
    l3 = h*G(Xo+h/2,Yo+k2/2,Zo+l2/2)
    k4 = h*F(Xo+h,Yo+k3,Zo+l3)
    l4 = h*G(Xo+h,Yo+k3,Zo+l3)
    !Sum Up
    Yn = Yo+(k1+2*k2+2*k3+k4)/6
    Zn = Zo+(l1+2*l2+2*l3+l4)/6
    !Operaion for Next calculation
    Xo=Xo+h       !(+h) than previous Term
    Yo=Yn      !Now Yn becomes Yo
    Zo=Zn      !Now Zn becomes Zo
 End Do
 Write (*,*) "Xn,Yn =",Xo,Yo
 Stop
 End
```

produces the following result

```
C:\Program Files (x86)\Silverfrost\FTN95\Plato.exe
 Given Equation '(y2)-6(y1)+(y0)=0'
 Xo=0, Yo=3, Zo=Y'o=1, Xn=1, n=?


10
 Xn,Yn =      1.00000         14.8276
```

```
!Runge-Kutta Fourth Order Method

!For 2nd Order Differentiation Equation

!First you have to define the function

F(x,y,z) = z !dy/dx

G(x,y,z) = 6*y-z !dz/dx = d2y/dx2

INTEGER :: n,i

REAL :: k1,l1,k2,l2,k3,l3,k4,l4    !Most Important

Write (*,*) "Given Equation '(y2)-6(y1)+(y0)=0'"

Write (*,*) "Xo=0, Yo=3, Zo=Y'o=1, Xn=1, n=?"

Xo=0    !Given Condition

Yo=3    !Given Condition

Zo=1    !Given Condition

Xn=1    !Given Condition

read (*,*) n    !n=number of Intercept

h=(Xn-Xo)/n

do i=1,n    !you have to do the Calculation 'n' times

k1 = h*F(Xo,Yo,Zo)

l1 = h*G(Xo,Yo,Zo)

k2 = h*F(Xo+h/2,Yo+k1/2,Zo+l1/2)

l2 = h*G(Xo+h/2,Yo+k1/2,Zo+l1/2)

k3 = h*F(Xo+h/2,Yo+k2/2,Zo+l2/2)

l3 = h*G(Xo+h/2,Yo+k2/2,Zo+l2/2)

k4 = h*F(Xo+h,Yo+k3,Zo+l3)

l4 = h*G(Xo+h,Yo+k3,Zo+l3)

!Sum Up

Yn = Yo+(k1+2*k2+2*k3+k4)/6

Zn = Zo+(l1+2*l2+2*l3+l4)/6

!Operation for Next calculation

Xo=Xo+h     !(+h) than previous Term

Yo=Yn    !Now Yn becomes Yo

Zo=Zn    !Now Zn becomes Zo

End Do

Write (*,*) "Xn,Yn =",Xo,Yo

Stop
```

edited Jan 31 '19 at 19:34
dantopa
7,707 8 29 63

answered Jan 31 '19 at 19:02
Raihan Ahamad
1

Can your clarify your question? – dantopa Jan 31 '19 at 19:39