The 4th -order Runge-Kutta method for a 2nd order ODE

By Gilberto E. Urroz, Ph.D., P.E.
January 2010

Problem description

Consider the 2nd-order ODE: $y'' + y \cdot y' + 3 \cdot y = \sin(x)$

subject to the initial conditions: $y(0) = -1$ $y'(0) = 1$

Variable substitution to form a system of ODEs:

This 2nd-order ODE can be converted into a system of two 1st-order ODEs by using the following variable substitution:

$$u_1 = y \quad u_2 = y'$$

with initial conditions:

$$u_1 = -1 \quad \text{and} \quad u_2 = 1 \quad \text{at} \quad x = 0\ .$$

The variable substitution $u_2 = y'$ is equivalent to:

$$\frac{d}{dx} u_1 = u_2 \qquad \text{[Eq. 1]}$$

while the ODE is re-written as: $y'' = -y \cdot y' - 3 \cdot y + \sin(x)$

or:

$$\frac{d}{dx} u_2 = -u_1 \cdot u_2 - 3 \cdot u_1 + \sin(x) \qquad \text{[Eq. 2]}$$

The system of equations [Eq. 1] and [Eq. 2] is transformed into the vector ODE:

$$\frac{d}{dx}\begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} u_2 \\ -u_1 \cdot u_2 - 3 \cdot u_1 + \sin(x) \end{pmatrix}$$

or,

$$\frac{d}{dx} u = f(x, u)\ , \quad \text{where} \quad u = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} \quad \text{and}$$

$$f(x, u) = \begin{pmatrix} u_2 \\ -u_1 \cdot u_2 - 3 \cdot u_1 + \sin(x) \end{pmatrix}$$

The initial conditions are $us = \begin{pmatrix} -1 \\ 1 \end{pmatrix}$ at $xs = 0$

We'll solve the ODEs in the interval: $0 \le x \le 20$ using 100 intervals.

Solution (version 1):

First, define the vector function f(x,u):

$$f(x, u) := \begin{pmatrix} u_2 \\ -u_1 \cdot u_2 - 3 \cdot u_1 + \sin(x) \end{pmatrix}$$

The initial conditions are: $xs := 0$ $\quad us := \begin{pmatrix} -1 \\ 1 \end{pmatrix}$

The end of the solution interval is: $xe := 20$

Use 100 intervals: $n := 100$

Calculate the increment size, Δx:

$$\Delta x := \mathrm{eval}\left(\frac{xe - xs}{n}\right) \qquad \Delta x = 0.2$$

Create the x solution vector: $xsol := \mathrm{eval}(xs, xs + \Delta x .. xe)$

The y-solution vector gets initialized as follows:

$$usol := us \qquad usol = \begin{pmatrix} -1 \\ 1 \end{pmatrix}$$

The following "for" loop calculates the Runge-Kutta algorithm (version 1) to produce the solution:

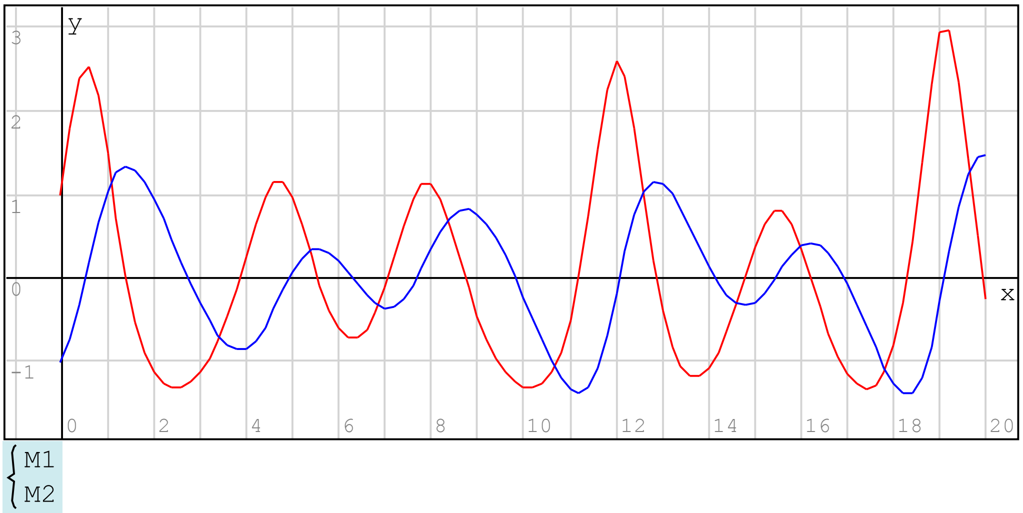
```
for k ∈ 1 .. n
   x0 := eval(xsol_k)
   u0 := eval(col(usol, k))
   xM := eval(x0 + (1/2)·Δx)
   K1 := eval(Δx·f(x0, u0))
   uM := eval(u0 + (1/2)·K1)
   K2 := eval(Δx·f(xM, uM))
   uM := eval(u0 + (1/2)·K2)
   K3 := eval(Δx·f(xM, uM))
   u1 := eval(u0 + K3)
   x1 := eval(xsol_(k+1))
   K4 := eval(Δx·f(x1, u1))
   u1 := eval(u0 + (1/6)·(K1 + 2·K2 + 2·K3 + K4))
   usol := augment(usol, u1)
```

After completing the iterative process, the solution is stored in a row vector called "ysol". This vector can be transposed to put together the graph of the two solutions as illustrated here:

$$usol := usol^T$$

M1 := augment(xsol, col(usol, 1))

M2 := augment(xsol, col(usol, 2))



The blue line represents u[1]=y while the red line represents u[2] = dy/dx.

Solution (version 2):

First, define the vector function f(x,y):

$$f(x, u) := \begin{pmatrix} u_2 \\ -u_1 \cdot u_2 - 3 \cdot u_1 + \sin(x) \end{pmatrix}$$

The initial conditions are: xs := 0 $us := \begin{pmatrix} -1 \\ 1 \end{pmatrix}$

The end of the solution interval is: xe := 20

Use 100 intervals: n := 100

Calculate the increment size, Δx:

$$\Delta x := \text{eval}\left(\frac{xe - xs}{n}\right)$$

Δx = 0.2

Create the x solution vector: xsol := eval(xs, xs + Δx .. xe)

The y-solution vector gets initialized as follows:

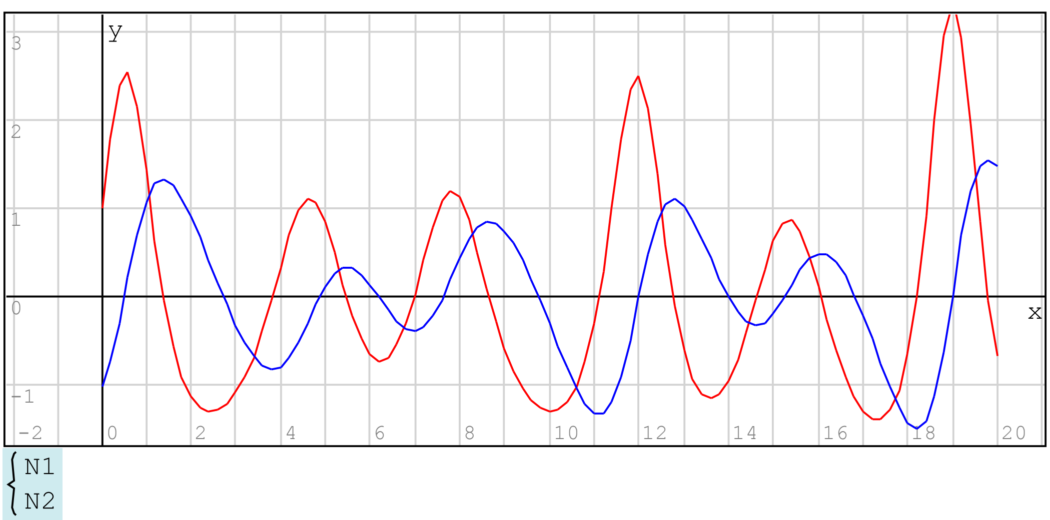
usol := us $usol = \begin{pmatrix} -1 \\ 1 \end{pmatrix}$

The following "for" loop calculates the Runge-Kutta algorithm (version 1) to produce the solution:

```
for k ∈ 1 .. n
  | x0 := eval(xsol_k)
  | u0 := eval(col(usol, k))
  | x13 := eval(x0 + (1/3)·Δx)
  | x23 := eval(x0 + (2/3)·Δx)
  | K1 := eval(Δx·f(x0, u0))
  | u13 := eval(u0 + (1/3)·K1)
  | K2 := eval(Δx·f(x13, u13))
  | u23 := eval(u13 + (1/3)·K2)
  | K3 := eval(Δx·f(x23, u23))
  | u1 := eval(u0 + K1 − K2 + K3)
  | x1 := eval(xsol_(k+1))
  | K4 := eval(Δx·f(x1, u1))
  | u1 := eval(u0 + (1/8)·(K1 + 3·K2 + 3·K3 + K4))
  | usol := augment(usol, u1)
```

After completing the iterative process, the solution is stored in a row vector called "ysol". This vector can be transposed to put together the graph of the two solutions as illustrated here:

```
usol := usol^T
N1 := augment(xsol, col(usol, 1))
N2 := augment(xsol, col(usol, 2))
```



The blue line represents u[1]=y while the red line represents u[2] = dy/dx.