

Runge-Kutta algorithm example

This Maple document, and the mirror Matlab document, have equivalent code for solving initial value problems using the Runge-Kutta method. By comparing the code you can see some of the main syntax differences between Maple and Matlab. In all examples, we are approximating the solution to the IVP for $y(x)$,

$$\begin{aligned} y'(x) &= y \\ y(0) &= 1 \end{aligned}$$

which has exact solution $y(x) = e^x$. You can execute commands successively by placing your cursor in the first math field and then hitting <enter> repeatedly to move through the worksheet. Or, you can use the Edit/Execute/Worksheet option at the top of the Maple Window. (And, you can also remove all output from your worksheet using the Edit/Remove Output/From Worksheet option.)

```
> restart : #clear all memory
> Digits := 6 : #decide how many digits to use in numerical computations

> N := 5 : # number of steps (subintervals)
  x0 := 0. : #choose initial x-value
  xf := 1.0 : #final x-value Note that xf-x0 must be divisible by h in order to end exactly at xf
  h := (xf - x0) / N : # step size
>

> X[0] := 0 : #We choose to use a vector "X" to hold x-values, and this is the initial one
  Y[0] := 1 :
  #We choose to use a vector "Y" to hold y-values, and this is the initial one for our IVP
> frhs := (x, y) -> y : # slope function f(x,y) for our example

> for i from 0 to N - 1 do #Runge Kutta loop
  k1 := frhs(X[i], Y[i]) : #left slope estimate
  k2 := frhs(X[i] + h/2, Y[i] + h/2 * k1) : #first midpoint slope estimate
  k3 := frhs(X[i] + h/2, Y[i] + h/2 * k2) : #second midpoint slope estimate
  k4 := frhs(X[i] + h, Y[i] + h * k3) : # right endpoint slope estimate
  k := (k1 + 2 * k2 + 2 * k3 + k4) / 6 :
  # "Simpson's rule" average slope estimate motivates Runge Kutta
  X[i + 1] := X[i] + h :
  Y[i + 1] := Y[i] + h * k
end do:
>
> for i from 0 to N do #to print out results
```

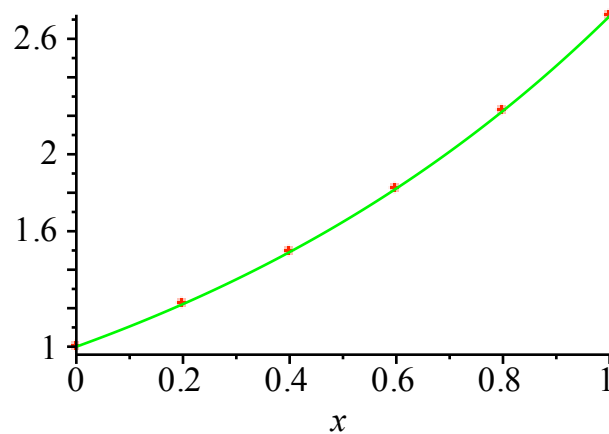
```
print(i, X[i], Y[i]);
end do;
```

```
0, 0, 1
1, 0.200000, 1.22140
2, 0.400000, 1.49182
3, 0.600000, 1.82211
4, 0.800000, 2.22552
5, 1.00000, 2.71825
```

(1)

```
> Yexact := x → ex : #exact solution for this IVP
> with(plots) : #load plotting package
plot1 := pointplot( {seq( [X[i], Y[i]], i = 0 .. N) }, color = red ) : #Euler points
plot2 := plot(Yexact(x), x = 0 .. 1, color = green) : #exact solution graph
display( {plot1, plot2}, title = `Runge-Kutta approximation and exact` );
```

Runge-Kutta approximation and exact solution



```
> Error := abs(Yexact(xf) - Y[N]); #absolute error at final x-value
RelativeError :=  $\frac{\text{Error}}{\text{abs}(Yexact(xf))}$ ; #relative error
```

```
Error := 0.00003
```

```
RelativeError := 0.0000110364
```

(2)