

Numerical methods for solving second order differential equations
of the form
 $y'' + p(x, y)y' + q(x, y)y = r(x)$
with initial values

We have so far developed methods for solving first order differential equations, namely the Euler, Modified Euler and Runge Kutta order 4 (the last two being predictor-corrector methods).

In calculating the analytical solution of a second order differential equation, two arbitrary constants appear due to integrating twice. These constants can be found from the **initial** or **boundary** values that define the differential equation. If the differential equation $y'' + p(x, y)y' + q(x, y)y = r(x)$ has a solution $y(x)$ then the *initial* value values take the form $y(x_0) = y_0$ and $y'(x_0) = z_0$. So, we know the starting co-ordinate of the solution (as we did in the first order case) and we also know it's initial gradient. This initial information is essential for solving second order differential equations numerically.

We also mentioned *boundary* value problems, these differ from initial value problems in that were we are not given a starting gradient. Instead we are given a second co-ordinate through which the solution passes. Boundary value problems need special treatment, numerically, and we shall study some of these methods later.

Recasting

The first step in solving a second order initial value problem is to recast the second order equation into two simultaneous first order differential equations. For example, if the differential equation is defined as

$$y'' - 3xy' + \sin x y = e^x \text{ where } y(0) = 0 \text{ and } y'(0) = 1$$

then we recast this equation by first setting

$$y' = z \text{ which when differentiated with respect to } x \text{ gives } y'' = z'$$

and replacing y' and y'' in the differential equation with z and z' , respectively.

So we have $y' = z$

$$z' - 3xz + \sin x y = e^x$$

finally we rearrange the equations such that y' and z' are written as functions of x , y , and z .

$$y' = z$$

$$z' = 3xz - \sin x y + e^x$$

Exercise

Recast these differential equation into sets of first order differential equations.

(a) $y'' - \sin(xy)y' = y - e^x$ (b) $xy'' - (1 + x^2)y' + y = e^x$

Solving Simultaneous First Order ODE's

The methods of Euler and Ring Kutta for first order odes can be applied to simultaneous first order differential equations. In the first instance we will use Euler's method to demonstrate the technique.

We are given two first order differential equations

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

$$\frac{dz}{dx} = g(x, y, z), \text{ (notice 3 variables 2 equations)}$$

with initial values x_0, y_0, z_0 .

We will applying Euler's method to obtain estimates for y_1 and z_1 ,

$$y_1 = y_0 + hf(x_0, y_0, z_0)$$

$$z_1 = z_0 + hg(x_0, y_0, z_0)$$

were h is the step size.

This of course leads us to the more general recurrence relations

$$y_{n+1} = y_n + hf(x_n, y_n, z_n)$$

$$z_{n+1} = z_n + hg(x_n, y_n, z_n)$$

DERIVE ACTIVITY 8A

- (A) Develop a function that will calculate the Euler method approximations to two simultaneous first order ODEs, with: starting values x_0, y_0, z_0 ; step size h ; n applications.

Author

```
#1: f(x, y, z):=
#2: g(x, y, z):=
#3: [x0:=, y0:=, z0:=]
#4: [x+h, y+hf(x, y, z), z+hg(x, y, z)]
#5: ITERATES(#4,[x, y, z],[x0, y0, z0],n)
#6: EULER_SIM(x, y, z, x0, y0, z0, h, n):=#5
```

Remove #3 to #5 and Save as EULERSIM.MTH

- (B) Use the function EULER_SIM to approximate the solution of

$$\frac{dy}{dx} = 4y + 2z$$

$$\frac{dz}{dx} = 3y - z$$

with initial values [0,1,1] from $x=0$ to $x=1$ with step size 0.1

Author

```
#4: f(x, y, z):= 4y + 2z
#5: g(x, y, z):= 3y - z
#6: EULER_SIM(x, y, z, 0, 1, 1, 0.1, 10)
```

approX #6

to give

#7:	<table style="border-collapse: collapse; width: 100%; text-align: center;"> <tr><td style="padding: 2px 10px;">0</td><td style="padding: 2px 10px;">1</td><td style="padding: 2px 10px;">1</td></tr> <tr><td style="padding: 2px 10px;">0.1</td><td style="padding: 2px 10px;">1.6</td><td style="padding: 2px 10px;">1.2</td></tr> <tr><td style="padding: 2px 10px;">0.2</td><td style="padding: 2px 10px;">2.48</td><td style="padding: 2px 10px;">1.56</td></tr> <tr><td style="padding: 2px 10px;">0.3</td><td style="padding: 2px 10px;">3.784</td><td style="padding: 2px 10px;">2.148</td></tr> <tr><td style="padding: 2px 10px;">0.4</td><td style="padding: 2px 10px;">5.7272</td><td style="padding: 2px 10px;">3.0684</td></tr> <tr><td style="padding: 2px 10px;">0.5</td><td style="padding: 2px 10px;">8.63176</td><td style="padding: 2px 10px;">4.47972</td></tr> <tr><td style="padding: 2px 10px;">0.6</td><td style="padding: 2px 10px;">12.980408</td><td style="padding: 2px 10px;">6.621276</td></tr> <tr><td style="padding: 2px 10px;">0.7</td><td style="padding: 2px 10px;">19.49682639</td><td style="padding: 2px 10px;">9.853270799</td></tr> <tr><td style="padding: 2px 10px;">0.8</td><td style="padding: 2px 10px;">29.26621111</td><td style="padding: 2px 10px;">14.71699163</td></tr> <tr><td style="padding: 2px 10px;">0.9</td><td style="padding: 2px 10px;">43.91609389</td><td style="padding: 2px 10px;">22.02515581</td></tr> <tr><td style="padding: 2px 10px;">1</td><td style="padding: 2px 10px;">65.88756261</td><td style="padding: 2px 10px;">32.99746839</td></tr> </table>	0	1	1	0.1	1.6	1.2	0.2	2.48	1.56	0.3	3.784	2.148	0.4	5.7272	3.0684	0.5	8.63176	4.47972	0.6	12.980408	6.621276	0.7	19.49682639	9.853270799	0.8	29.26621111	14.71699163	0.9	43.91609389	22.02515581	1	65.88756261	32.99746839
0	1	1																																
0.1	1.6	1.2																																
0.2	2.48	1.56																																
0.3	3.784	2.148																																
0.4	5.7272	3.0684																																
0.5	8.63176	4.47972																																
0.6	12.980408	6.621276																																
0.7	19.49682639	9.853270799																																
0.8	29.26621111	14.71699163																																
0.9	43.91609389	22.02515581																																
1	65.88756261	32.99746839																																

- (C) The second order differential equation $y'' + 3y' + 2y = e^x$ into two simultaneous ODEs. The ode has initial conditions $y(0) = 0$ and $y'(0) = 1$, use Euler's method to approximate the solution between $x=0$ and $x=1$, with step size 0.1. Compare this approximate solution with the analytical solution, which can be calculated using the function DSOLVE2_IV() found in the utility file ODE2.MTH.

Recasting we get

$$\begin{aligned} y' &= z \\ z' &= e^x - 3z - 2y \end{aligned}$$

Author

- #8: $f(x, y, z) := z$
 #9: $g(x, y, z) := e^x - 3z - 2y$ (**Alt e** for the exponential e)
 #10: $EULER_SIM(x, y, z, 0, 0, 1, 0.1, 10)$
 approx #10 to give

```

#11:
      0      0      1
0.1   0.1   0.8
0.2   0.18  0.6505170918
0.3  0.2450517091  0.5415022400
0.4  0.2992019331  0.4650271069
0.5  0.3457046438  0.4148610580
0.6  0.3871907496  0.3861339388
0.7  0.4258041435  0.3750674873
0.8  0.4633108923  0.3787616831
0.9  0.5011870606  0.3950250926
1    0.5406895698  0.4222404638

```

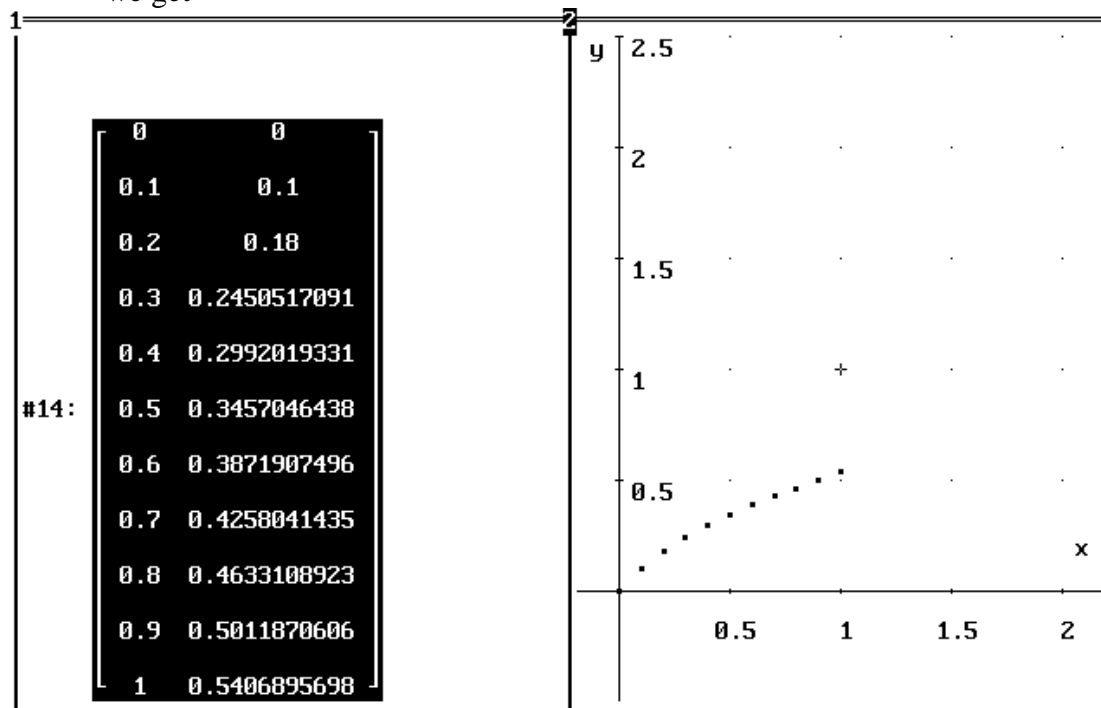
We can plot y against x by extracting the first two columns of this matrix, the following function will do this for us.

```
#12:  EXTRACT(m):= [m`sub1,m`sub2]
```

```
#13:  EXTRACT(#11)
```

approX #13 , Plot (Beside at 40), Centre , Zoom in (once: press F9).

we get



COMMAND: **Algebra** Center Delete Help Move Options Plot Quit Range Scale Transfer
Window axes Zoom

Enter option

Cross x:1

y:1

Scale x:0.5

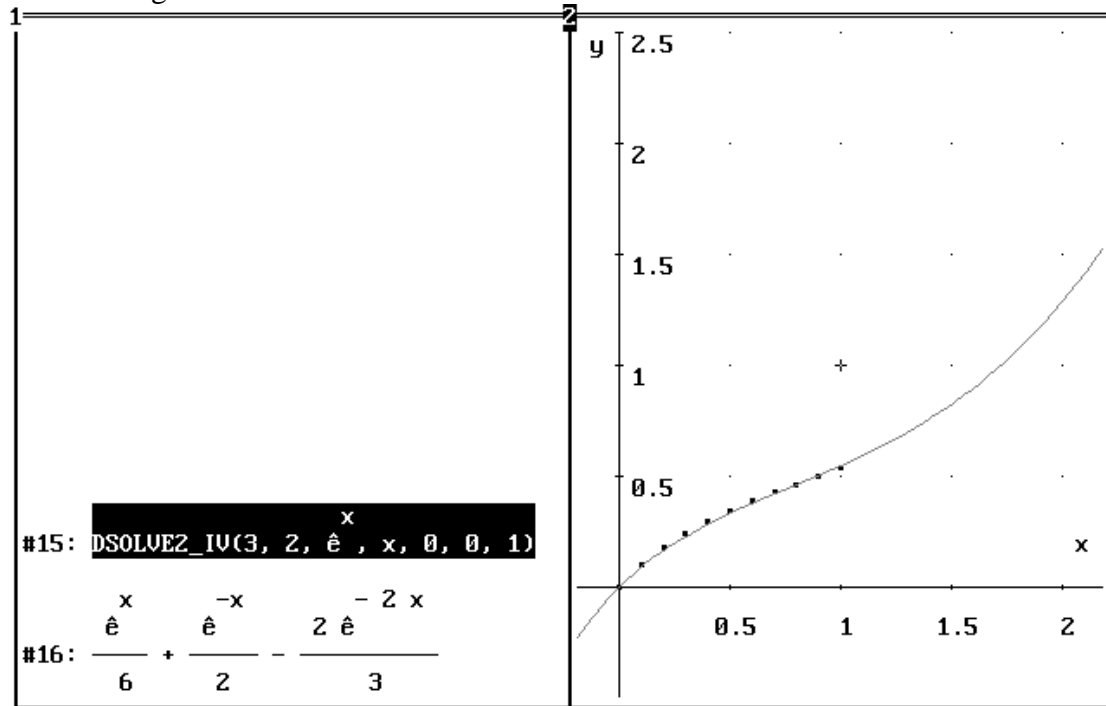
y:0.5

Derive 2D-plot

Transfer Load Utility ODE2.MTH

Author#15: $DSOLVE2_IV(3,2,e^x,x,0,0,1)$ **Simplify #15 and Plot #16**

we get



COMMAND: **Algebra** Center Delete Help Move Options Plot Quit Range Scale Transfer Window axes Zoom

Enter option

Cross x:1

y:1

Scale x:0.5

y:0.5

Derive 2D-plot

Which shows that, as one would expect, the Euler method is 'peeling' away from the solution as x increases.

The Modified Euler Method

In the single first order case, the modified Euler method is defined as

$$y_{n+1} = y_n + \frac{h}{2}(k_1 + k_2) \quad \text{where } k_1 = f(x, y) \text{ and } k_2 = f(x+h, y+hk_1).$$

As in the Euler method, we have a third variable, z , which is also changing as x steps along the x axis. So we adjust the modified Euler method to account for the third variable thus:

Let $f(x, y, z) = k_1$ and $g(x, y, z) = l_1$ (we use l as it is next to k in the alphabet)

and also let

$$k_2 = f(x+h, y+hk_1, z+hl_1) \quad \text{and} \quad l_2 = g(x+h, y+hk_1, z+hl_1).$$

We can now define the recurrence equations for y and z as,

$$y_{n+1} = y_n + \frac{h}{2}(k_1 + k_2) \quad \text{and} \quad z_{n+1} = z_n + \frac{h}{2}(l_1 + l_2)$$

respectively.

Example 8A

Find an approximate value for $y(1.1)$ and $z(1.1)$, where $y(x)$ and $z(x)$ satisfy the simultaneous differential equations,

$$\frac{dy}{dx} = x + yz \quad \text{and} \quad \frac{dz}{dx} = x - z$$

with initial values $x_0 = 1$, $y_0 = 0$ and $z_0 = 0$ and a step size of $h=0.1$.

So $k_1 = x + yz$ and $l_1 = x - z$. Using the initial values we have that

$$k_1 = 1 + 0 \cdot 0 = 1 \quad \text{and} \quad l_1 = 1 - 0 = 1$$

So, as $k_2 = f(x+h, y+hk_1, z+hl_1)$ and $l_2 = g(x+h, y+hk_1, z+hl_1)$, then $k_2 = 1.1 + (0 + 0.1 \cdot 1)(0 + 0.1 \cdot 1) = 1.11$ and $l_2 = 1.1 - (0 + 0.1 \cdot 1) = 1$

Which leads to,

$$y(1.1) = y_1 = y_0 + \frac{h}{2}(k_1 + k_2) = 0 + \frac{0.1}{2}(1 + 1.1) = 0.1055 \quad \text{and}$$

$$z(1.1) = z_1 = z_0 + \frac{h}{2}(l_1 + l_2) = 0 + \frac{0.1}{2}(1 + 1) = 0.1.$$

Whilst not overcomplicated in its computation, one wouldn't wish to be doing this 1000 times!

DERIVE ACTIVITY 8B

(A) Write a function that will automate the modified Euler method for two simultaneous first order differential equations $\frac{dy}{dx} = f(x, y, z)$ and

$$\frac{dz}{dx} = g(x, y, z), \text{ with initial values } x_0, y_0, z_0 \text{ and step size } h.$$

Author

```
#1: f(x, y, z):=
#2: g(x, y, z):=
#3: [x0:=, y0:=, z0:=]
#4: k1:= f(x, y, z)
#5: l1:= g(x, y, z)
#6: k2:= f(x+h, y+hk1, z+hl1)
#7: l2:= g(x+h, y+hk1, z+hl1)
#8: y + h/2(k1+k2)
#9: z + h/2(l1+l2)
#10: [x+h, #8, #9]
```

Simplify #10**Author**

```
#12: ITERATES(#11, [x, y, z], [x0, y0, z0], n)
#13: RK2_SIM(x, y, z, x0, y0, z0, h, n):=#12
```

Remove #4 to #12

Transfer Save Derive as RK2_SIM.MTH

- (B) Use the RK2_SIM() function to find $y(1)$ and $z(1)$ for the differential equation in example 8A with step size $h=0.1$.

```
#5: f( , y, z):= yz
```

```
#6: x y x-z
```

```
#7: RK SIM y z
```

```
#6: G(x, y, z) := x - z
```

```
#7: RK2_SIM(x, y, z, 1, 0, 0, 0.1, 10)
```

```
#8: [ 1      0      0
    1.1    0.1055  0.1
    1.2 0.2231930500 0.2
    1.3 0.3556398341 0.3
    1.4 0.5059006122 0.4
    1.5 0.6776720404 0.5
    1.6 0.8754605107 0.6
    1.7 1.104803910 0.7
    1.8 1.372557655 0.8
    1.9 1.687266263 0.9
    2   2.059649256 1 ]
```

```
COMMAND: Author Build Calculus Declare Expand Factor Help Jump solve Manage
Options Plot Quit Remove Simplify Transfer Unremove move Window approx
Compute time: 0.4 seconds
Approx(#7)          C:1212.MTH          Free:99%  Ins          Derive Algebra
```

Exercise 8A

- (1) approximate 2 first order simultaneous equations using the Runge Kutta order 4 method. Use the RK2_SIM() function for different step size h equation $y' + 3y + 2z = 0$, $z' + 3z + 2y = 0$, $y(0) = 1, z(0) = 0$ to show that the modified Euler method has global error order h^2 .

Using the RK function in the DERIVE™ utility file ODE_APPR.MTH

There is a utility file that is supplied with DERIVE called ODE_APPR.MTH that contains a function that will perform Runge Kutta order 4 approximations for any number of simultaneous first order differential equations. The function is of the form $RK(r, v, v_0, h, n)$ where: r is a vector whose elements are the right hand side of each of the differential equation (in the form $y' = r(x, y, \dots)$); v is a vector whose elements are the variables; v_0 is a vector whose elements are the initial values for the variable; h is

the step size; n is the number of iterations. As ever, an example will describe best how to use this function.

DERIVE ACTIVITY 8C

- (A) Use the RK4() function in ODE_APPR.MTH to solve the following system of first order differential equations with the given initial values;

$$\frac{dw}{dx} = x - w + y, \quad \frac{dy}{dx} = wx + y + z, \quad \frac{dz}{dx} = wxyz$$

where $x_0 = 0$, $w_0 = 1$, $y_0 = 2$, $z_0 = 3$, $h=0.1$ and $n=10$.

So r is the vector $[x - w + y, wx + y + z, wxyz]$; v is the vector $[x, w, y, z]$; v_0 is the vector $[0, 1, 2, 3]$.

The order of elements in v and v_0 is very important! As the derivatives are with respect to x , then x is the first element, as the first differential equation is $\frac{dw}{dx} = x - w + y$ then w is the next element, etc. To get RK() to the desired

produce the results:

Transfer Load Utility ODE_APPR.MTH

Author

#1: $[x - w + y, wx + y + z, wxyz]$

#2: $[x, w, y, z]$

#3: $[0, 1, 2, 3]$

#4: $RK(\#1, \#2, \#3, 0.1, 10)$

approx #4 to give

	0	1	2	3
	0.1	1.125226406	2.532656237	3.038344405
	0.2	1.302386881	3.144603349	3.202225852
	0.3	1.535496334	3.868079966	3.630266412
	0.4	1.832463093	4.764484812	4.687422553
	0.5	2.209461937	5.976786968	7.640042258
#5:	0.6	2.706622704	7.986411547	19.39890339
	0.7	3.487332173	14.19072784	143.6171366
	0.8	8.005613130	371.7328509	2.585249712 ⁵ 10
	0.9	7.679312710 ⁸ 10	6.816133813 ¹⁷ 10	4.386745322 ³³ 10
	1	6.114744330 ¹⁰³ 10	3.291317200 ¹⁹² 10	6.238564842 ³⁵⁴ 10

- (B) Use the RK() function in ODE_APPR.MTH to approximate the solution to the second order differential equation $y'' + 5y' + 6y = e^{-x} \sin x$ with initial values $y(0) = 0$ and $y'(0) = 1$, between $x=0$ and $x=2$.

First recast the second order differential equation into two first order equations, we have

$$\frac{dy}{dx} = z$$

$$\frac{dz}{dx} = e^{-x} \sin x - 5z - 6y$$

Author

- #6: $[z, e^{-x} \sin x - 5z - 6y]$
 #7: $[x, y, z]$
 #8: $[0, 0, 1]$
 #9: $RK(\#6, \#7, \#8, 0.1, 20)$

approX #9

We now have a large matrix of points for x , y and y' , if we wish to plot x against y then we need to extract the first two columns.

When ODE_APPR.MTH is loaded a function called EXTRACT_2_COLUMNS($m, C1, C2$) is entered into DERIVE's function memory, this extracts columns $C1$ and $C2$ from the matrix m . We can use this to extract the first two columns of the matrix in expression 10 to produce a plot of the solution.

Author

- #11: $\text{Extract_2_Columns}(\#10, 1, 2)$

approX #11**Plot, Beside, at 40, #12**

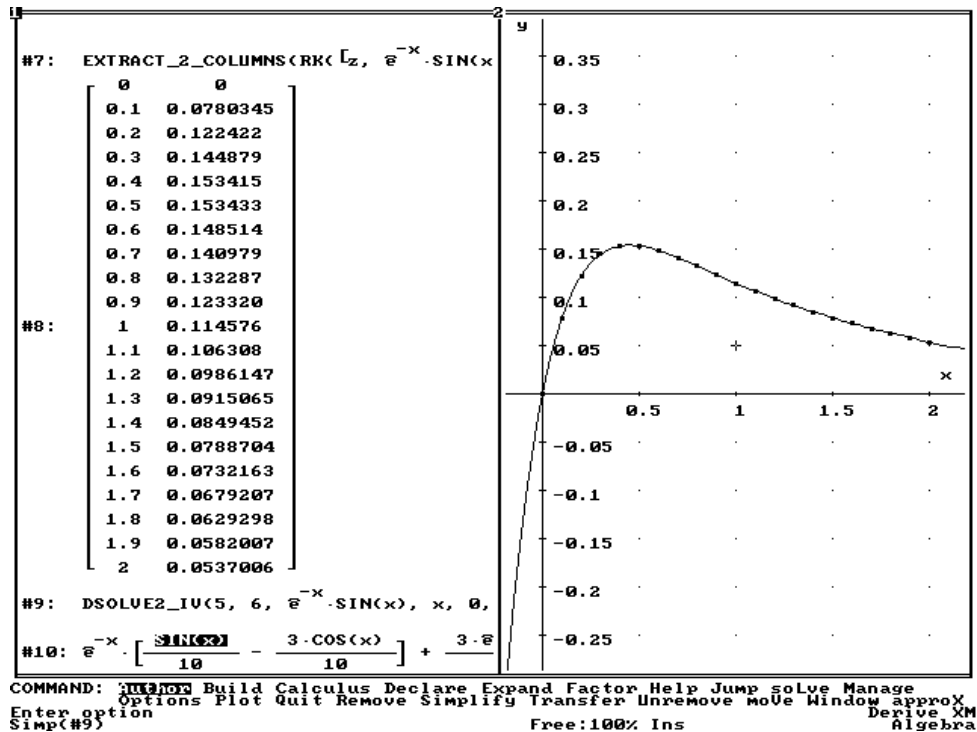
This particular differential equation can, in fact, be solved analytically. We can use the utility file ODE2.MTH which contains the function DSOLVE2_IV($p, q, r, x, x0, y0, v0$) that solves second order differential equations of the form $y'' + p(x)y' + q(x)y = r(x)$ with initial values $y(x0) = y0$ and $y'(x0) = v0$ (where possible). We shall use this function here!

Transfer Load Utility ODE2.MTH

- #13: $DSOLVE2_IV(5, 6, e^{-x} \sin x, x, 0, 0, 1)$

Simplify #13**Plot #14 Scale x: 0.5 y: 0.05**

The following screen dump demonstrates the DERIVE output



RK and DSOLVE2_IV in action

(line = analytic solution : dots = numeric solution)

The Shooting Method for Boundary Value Problems

In the previous examples the differential equations were initial value problems, so called because an initial value for position and gradient are supplied to solve the equations. However, it is quite common for a differential equation to be defined in terms of Boundary Values, i.e. an initial value for the position and another value for the position. In other words we are given two points through which the solution passes instead of an initial position and gradient. Clearly this will cause problems for solution numerically because the methods described previously are reliant on the knowledge of an initial gradient.

An example of a boundary value differential equation is

$$y'' + 3y' + 2y = -5\cos x \text{ where } y(0) = 0 \text{ and } y(1) = 0.$$

Of course this particular second order differential equation has an analytical solution, hence the boundary value nature of the ODE does not cause a problem as the boundary values can determine the two arbitrary constants that arise from the general solution.

However, let's imagine that we do not know the analytical solution and we wish to solve the equation using say the Runge Kutta method order four.

The title Shooting Method is descriptive of the method, it is very much like shooting at a target. We do not know the initial gradient at which to shoot at the target but we know where the target is. So we make a guess (not very scientific but there it is!) and

see how close we get to the target, we then adjust the initial gradient until we hit the target. There you have it, the Shooting Method as employed by the Artillery for centuries. Lets see this method in action!

DERIVE ACTIVITY 8D

We will use the example $y'' + 3y' + 2y = -5\cos x$ where $y(0) = 0$ and $y(1) = 0$.

Recasting the second order differential equation into two simultaneous first order differential equations we have:

$$\begin{aligned} y' &= z \\ z' &= -3z - 2y - 5\cos x \end{aligned}$$

Transfer Load the **Utility** File ODE_APPR.MTH

The initial value for the position is (0,0) and we will take the first guess at the initial value of the gradient to be 1, this will mean that we have an initial starting vector of [0,0,1]. If we use a step size of $h=0.1$, we will need 10 iteration to reach (1,0)

Author

```
#1: [z,-3z-2y-5cosx]
#2: RK(#1,[x,y,z],[0,0,1],0.1,10)
approX #2
```

we get

```
#1: [z, - 3 · z - 2 · y - 5 · COS(x)]
#2: RK([z, - 3 · z - 2 · y - 5 · COS(x)], [x, y, z], [0, 0, 1], 0.1, 10)
```

```
#3: [
  0      0      1
0.1  0.0634776  0.302882
0.2  0.0665454 -0.214371
0.3  0.0254618 -0.585199
0.4 -0.0465451 -0.836985
0.5 -0.138731  -0.992196
0.6 -0.242394 -1.06931
0.7 -0.350510 -1.08357
0.8 -0.457445 -1.04761
0.9 -0.558716 -0.971911
  1  -0.650802 -0.865246
```

```
COMMAND: Author Build Calculus Declare Expand Factor Help Jump soLve Manage
          Options Plot Quit Remove Simplify Transfer Unremove moVe Window approX
Compute time: 0.6 seconds
User                               Free:62%  Ins           Derive Algebra
```

From inspection, we can see that we have missed the mark but not by a great deal. A picture would be helpful at this stage to see what happened. The function `EXTRACT_2_COLUMNS(m,C1,C2)` was automatically loaded with the utility file `ODE_APPR.MTH`. If we now employ this function on #3 we can plot our attempt at the solution.

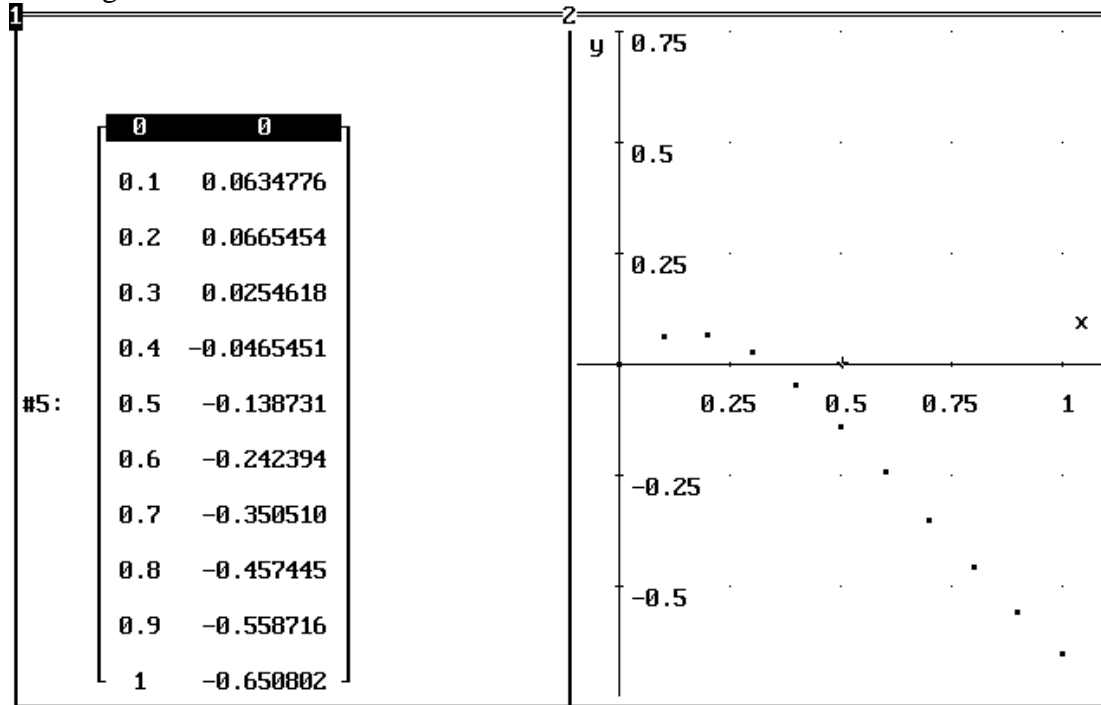
Author

#4: `EXTRACT_2_COLUMNS(#3,1,2)`

approx #4

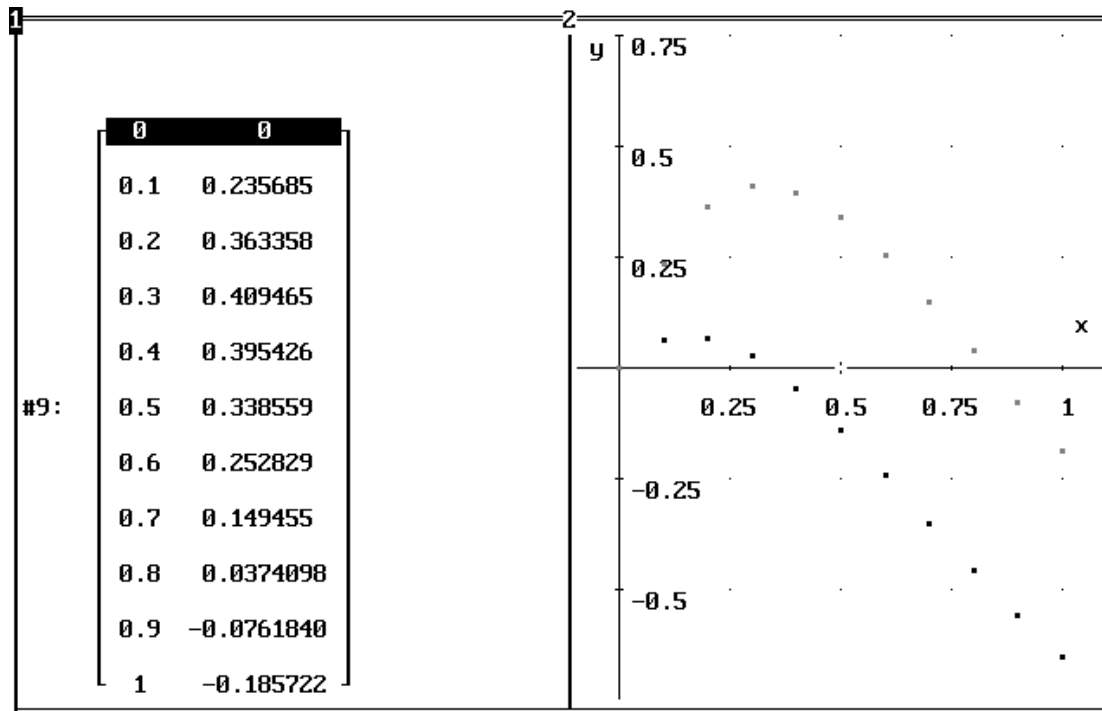
Plot, Beside, at Column 40 #5 Scale x:0.25 y:0.25 Centre (0,0.5)

resulting in



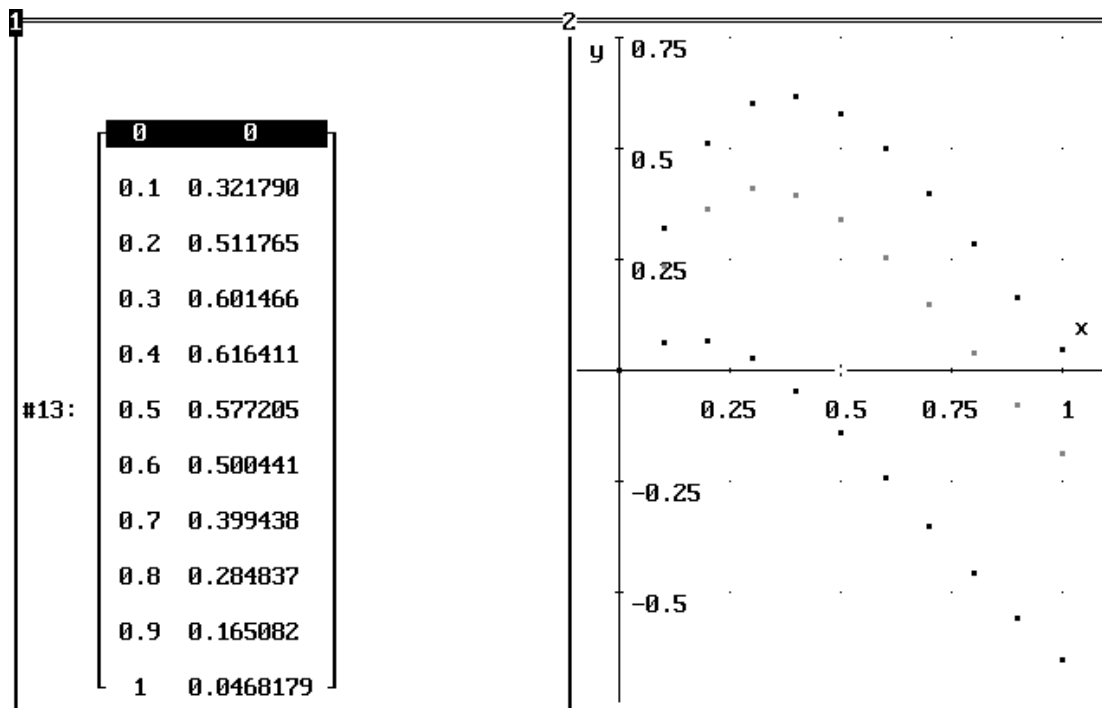
COMMAND: **Author** Build Calculus Declare Expand Factor Help Jump solve Manage
Options Plot Quit Remove Simplify Transfer Unremove move Window approx
Enter option
Approx(#4) Free:95% Ins Derive Algebra

Intuitively, the initial gradient needs to be much higher. Edit #3 by highlighting it, **Author F3** and adjust the starting vector to $[0,0,3]$, using a starting gradient of 3 instead of 1. approx and extract the first two columns and plot (as before) to give



COMMAND: **Author** Build Calculus Declare Expand Factor Help Jump solve Manage
 Options Plot Quit Remove Simplify Transfer Unremove move Window approx
 Enter option
 Approx(#8) Free:72% Ins Derive Algebra

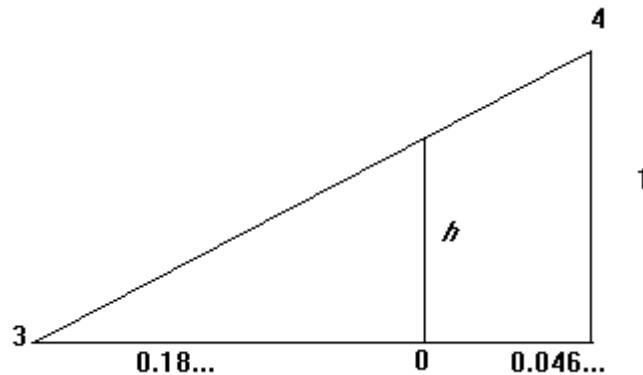
As we can see the initial gradient was still not high enough, so we repeat the whole process with a starting vector [0,0,4].



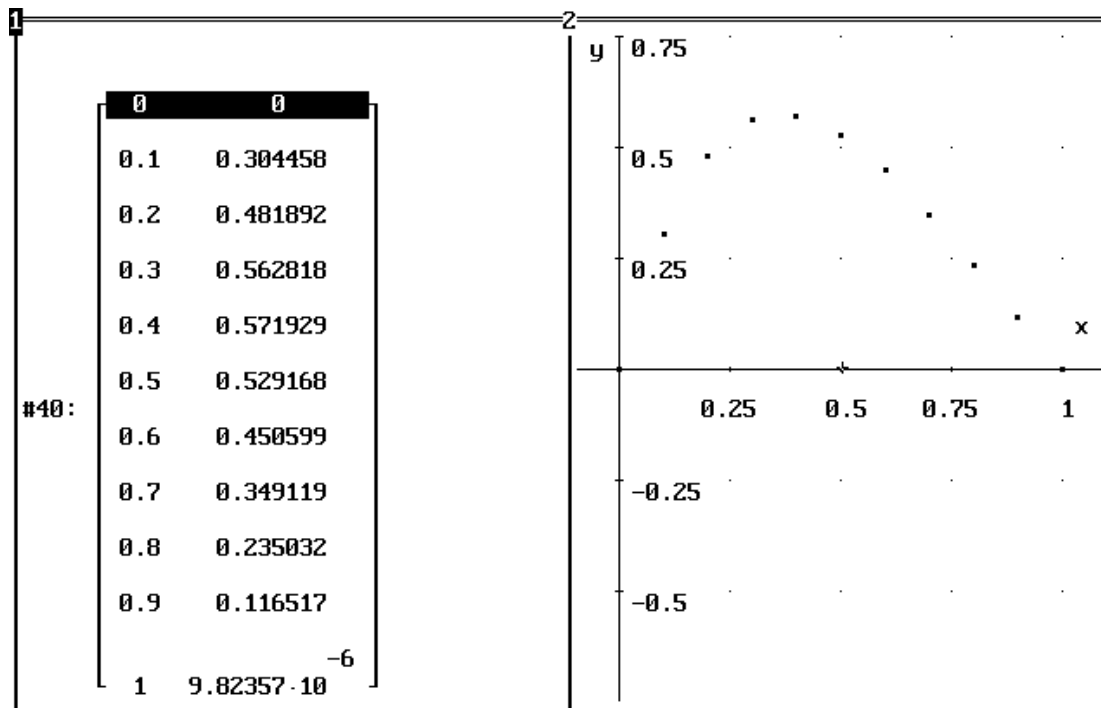
COMMAND: **Author** Build Calculus Declare Expand Factor Help Jump solve Manage
 Options Plot Quit Remove Simplify Transfer Unremove move Window approx
 Enter option
 Approx(#12) Free:53% Ins Derive Algebra

As you can see we have now over shot the mark, so our starting gradient is in the region [3,4]. Once we know an interval in which the initial gradient lies we can use Linear Interpolation to find a more accurate starting gradient.

When we used 3 as a starting gradient we had -0.185722 as our value for $y(1)$ and when we used 4 as a starting gradient we had 0.0468179 as our value for $y(1)$. Using the method of linear interpolation with approximate values



we have $\frac{h}{0.185772} = \frac{1}{0.185772 + 0.0468179} \Rightarrow h \approx 0.798710$. Which implies that a starting value for the gradient of 3.79871 should be very close to the mark!



COMMAND: **Author** Build Calculus Declare Expand Factor Help Jump soLve Manage
Options Plot Quit Remove Simplify Transfer Unremove moVe Window approx

Enter option

Approx(#39)

Free:59% Ins

Derive Algebra

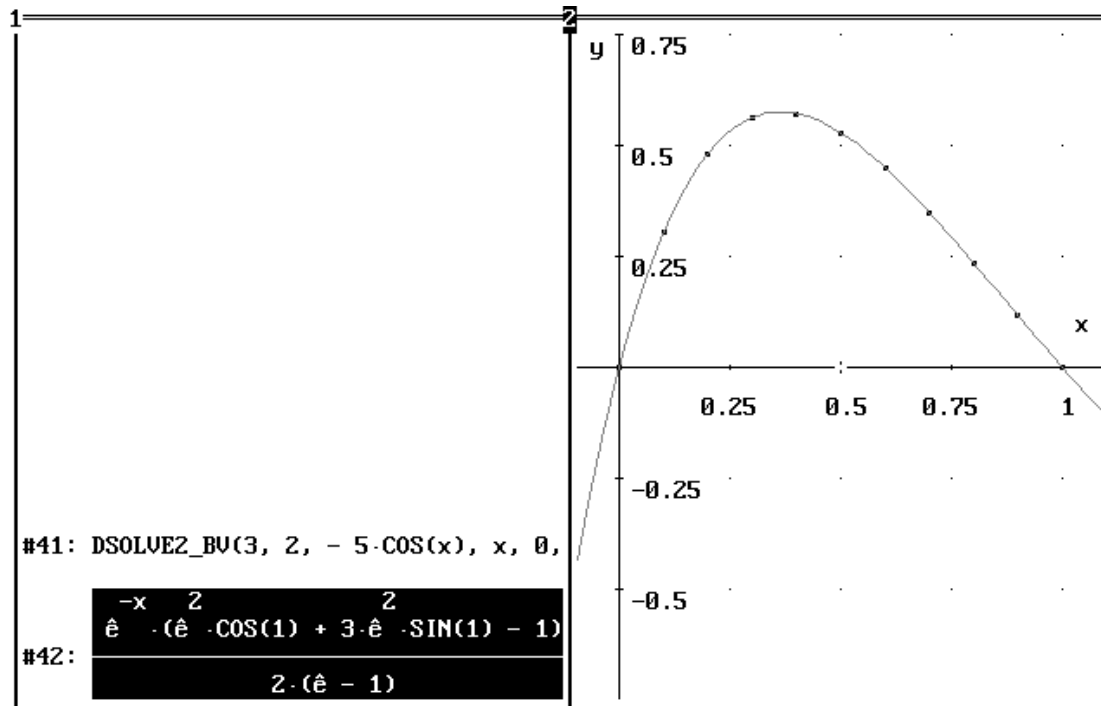
which is very close!!

Finally, we will solve this equation analytically (only because we can) and see what the analytical solution looks like.

Transfer Load Utility File ODE2.MTH

#n: *DSOLVE2_BV*(3,2,-5cos x,x,0,0,1,0)

Simplify #n and Plot #(n+1)



COMMAND: **Algebra** Center Delete Help Move Options Plot Quit Range Scale Transfer Window axes Zoom

Enter option

Cross x:0.5 y:0 Scale x:0.25 y:0.25 Derive 2D-plot

Of course, the luxury of such verification is not always available, but it demonstrates the accuracy of the method.

Exercise

Solve the following boundary value second order differential equations.

1. $y'' + xyy' + 3xy = e^{3x}$ when $y(0) = 1$ and $y(1) = 1$
2. $y'' + e^{5xy} y' = 6xy \cos x$ when $y(0) = 1$ and $y(3) = 0$