Solving Boundary Value Problems in MathCad

(Dr. Tom Co 11/3/2008)

Introduction

There are problems defined by differential equations known as boundary value problems (BVP), where some conditions are specified at the initial point while the rest are specified at the end point.

Example:

$$\frac{d^2y}{dx^2}+\frac{1}{2}\frac{dy}{dx}+y=5 \quad \textbf{(1)}$$

$$y(0)=1 \quad \text{and} \quad y(2)=2 \quad \textbf{(2)}$$

Unfortunately, most numerical methods such as Runge-Kutta solve only initial value problems (IVP), where all the conditions are given at the initial point. Thus, to be able to use the ODE-IVP solvers, we need to change the problem by first finding the missing initial conditions. One such method is known as the "shooting method" which tries different values for the missing initial conditions until the prescribed end conditions are satisfied. Other methods include finite difference methods. [1] In MathCad, the function **sbval( )** is tasked with finding the missing initial values.

MathCad Procedure:

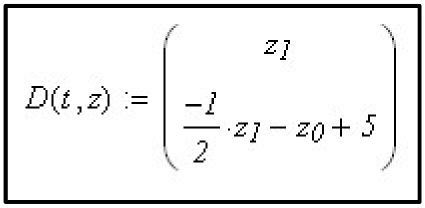
In the steps below, the discussion will proceed using the example given by (1) and (2):

1. Put the differential equation into the "state space" formulation, i.e. a set of first-order ODEs in which each first-order derivatives are in the left-hand sides and the right-hand sides contain only the state-variables (variables having derivatives) and process parameters.

[1] Not all boundary value problems have solutions. In some cases, there can even be multiple solutions. The conditions for the existence and/or uniqueness of the solutions of BVP are beyond the scope of this brief lecture.

$$\frac{dz_0}{dt} = z_1$$
$$\frac{dz_1}{dt} = -\frac{1}{2}z_1 - z_0 + 5$$

2. Collect the right hand side elements into a vector



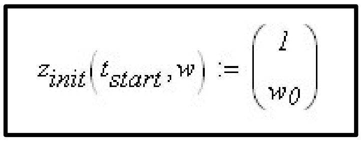
$$D(t,z) := \begin{pmatrix} z_1 \\ \frac{-1}{2}\cdot z_1 - z_0 + 5 \end{pmatrix}$$

Remarks:

a) Include the independent variable (t in our case) as the first argument of $D(\,)$, even if it does not appear in the vector of equations

b) Use the index-subscript for z, i.e. use **(ctrl [ )**.

3. Define the vector of initial conditions as a function of the start value of the independent variable, e.g. t_{start}, and a vector of unknown initial conditions, e.g. w



$$z_{init}(t_{start}, w) := \begin{pmatrix} 1 \\ w_0 \end{pmatrix}$$

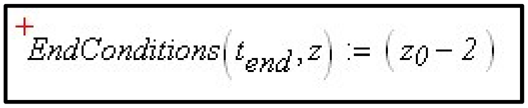
Remarks:

c) w is required to be a vector, including a 1×1 vector.

d) You can choose any other name instead of t_{start}. However, it has to be the first argument of the initial vector function, $z_{init}(\,)$.

e) Later, the vector of initial conditions needed by **RkAdapt( )** will be a vector of constants, not a function such as in $z_{init}(\,)$ above.

4. Define the vector of end conditions as a function of the end values of the independent variable, e.g. t_{end}, and the state vector. The entries of this vector contain conditions that needed to be zeroed.



$$EndConditions(t_{end}, z) := \begin{pmatrix} z_0 - 2 \end{pmatrix}$$

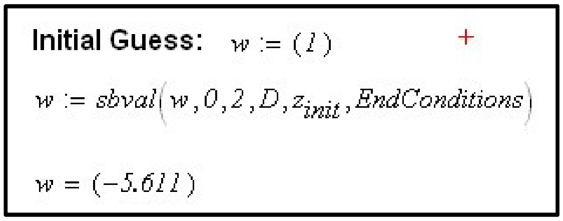
Remark:

f) The right hand side must be a vector, including the case of a 1 × 1 vector as in the example above.

5. Implement the **sbval( )** function. The syntax is given by

$$soln \coloneqq sbval(\,w_{guess}, t_{start}, t_{end}, D, InitFuncVec, EndCondVec)$$

where w_{guess} is a vector of initial guesses for the unknown initial values. t_{start} and t_{end} are scalar values for start and end values of the independent variables, respectively. D is the vector defined earlier, $InitFuncVec$ and $EndCondVec$ are vectors of initial functions and end conditions, respectively. For the example above,



Initial Guess: $w := (1)$

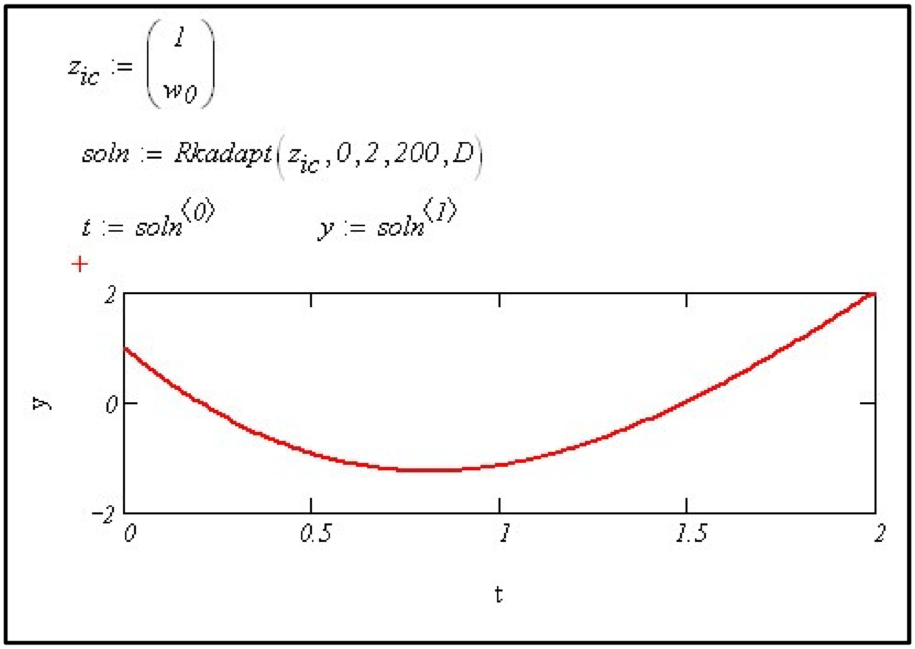
$w := sbval(w, 0, 2, D, z_{init}, EndConditions)$

$w = (-5.611)$

Remark:

g) w as before is required to be a vector, including the case of a 1 × 1 vector.

6. Using the results from **sbval( )**, define the constant vector of initial conditions to be applied with **Rkadapt( )** and simulate as before.



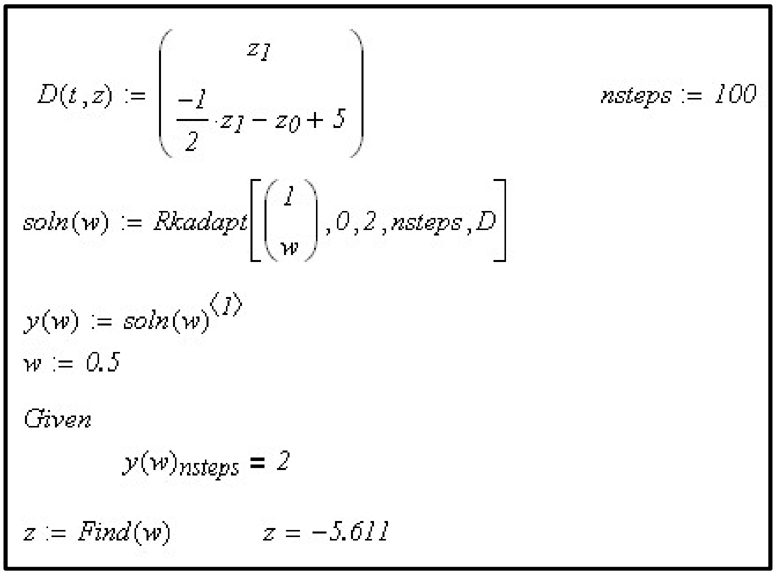
$z_{ic} := \begin{pmatrix} 1 \\ w_0 \end{pmatrix}$

$soln := Rkadapt(z_{ic}, 0, 2, 200, D)$

$t := soln^{\langle 0 \rangle}$ $y := soln^{\langle 1 \rangle}$

Alternative MathCad procedure:

1. Instead of **sbval( )**, one could use the **(Given…Find)** block to perform a “shooting method” to find the missing initial conditions.



$$D(t,z) := \begin{pmatrix} z_1 \\ \frac{-1}{2}\cdot z_1 - z_0 + 5 \end{pmatrix}$$

$nsteps := 100$

$$soln(w) := Rkadapt\left[\begin{pmatrix} 1 \\ w \end{pmatrix}, 0, 2, nsteps, D\right]$$

$y(w) := soln(w)^{\langle 1 \rangle}$

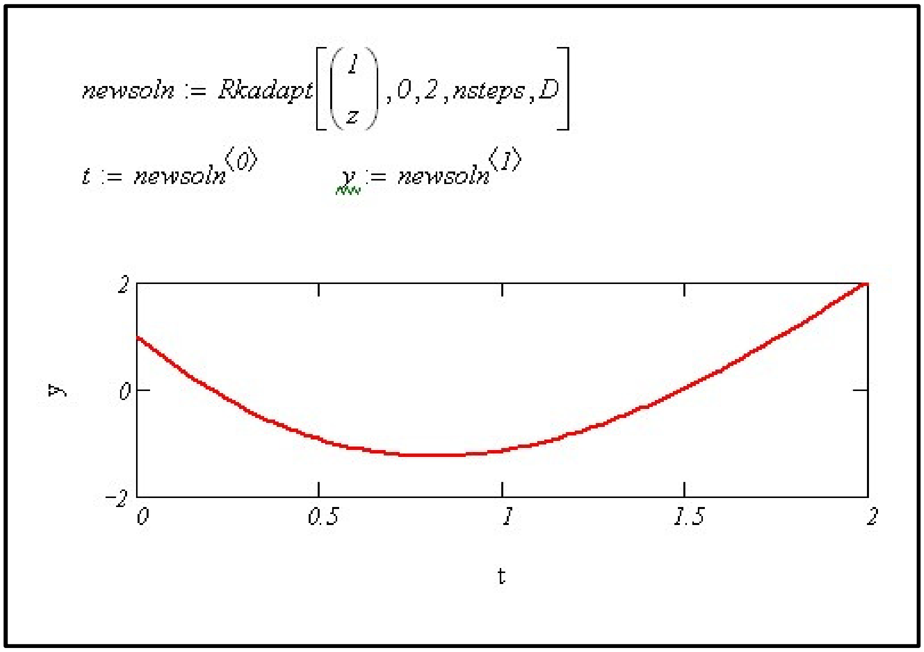
$w := 0.5$

Given

$y(w)_{nsteps} = 2$

$z := Find(w)$ $\qquad$ $z = -5.611$

2. After the missing initial conditions are found, solve the ODE once more to obtain the desired solution.



$$newsoln := Rkadapt\left[\begin{pmatrix} 1 \\ z \end{pmatrix}, 0, 2, nsteps, D\right]$$

$t := newsoln^{\langle 0 \rangle}$ $\qquad$ $y := newsoln^{\langle 1 \rangle}$