Simulation of a tennis ball throwing machine

Lars Schiller E-10 Simulation GmbH

July 2, 2015

1 Introduction and task description

We want to calculate the dropping velocity which is needed for a tennis ball to reach a certain point y_b after a certain time t_f. In doing so we taking to account the gravity force and the resistance force due to air drag. The following parameters are known:

y_a ($\text{Pos}_{t=0}$)	y_b ($\text{Pos}_{t=1.5}$)	g	v_w	ρ_T	ρ_L	c_w	r	t_f
(0,1)	(14,0)	$9.81\frac{\text{m}}{\text{s}^2}$	$-10\frac{\text{m}}{\text{s}}$	$408\frac{\text{kg}}{\text{m}^3}$	$1.225\frac{\text{kg}}{\text{m}^3}$	0.47	0.034 m	1.5 s

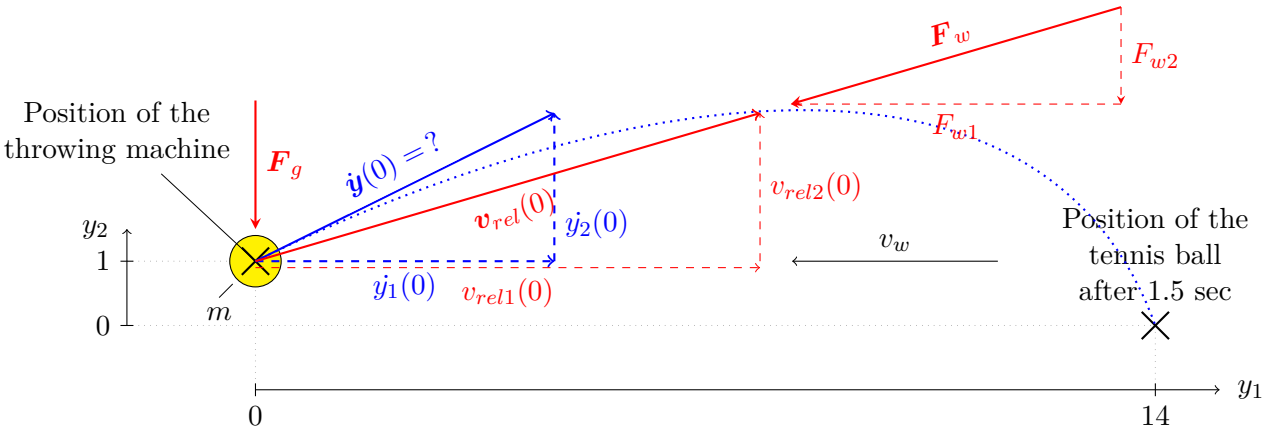

Figure 1: principle sketch of the problem to solve

2 Mathematical model of the task

In order to obtain a system of ordinary differential equations from the given problem one can use *Newtons* 2nd law

$$m\ddot{y} = \sum_i F_i. \tag{1}$$

It is known, that only the gravity force $\boldsymbol{F}_g$ and the force generated by air drag $\boldsymbol{F}_w$ are applied to the flying ball. We can calculate the gravity force to

$$F_g = mg, \tag{2}$$

where $m = V\rho_T = \frac{4}{3}\pi r^3 \rho_T$ is the mass of the tennis ball. We can approximate the drag force with the following approach[1]:

$$F_w = kv_{rel}^2 = \frac{1}{2}A_K c_w \rho_L v_{rel}^2, \tag{3}$$

[1] http://www.leifiphysik.de/themenbereiche/reibung-und-fortbewegung

where $A_K = \pi r^2$ is the cross-sectional area of the ball, c_w is the drag coefficient, ρ_L is the density of the air and v_{rel} the relative velocity of the ball. These forces are illustrated in figure 1. With this information we can rewrite equation 1 as:

$$\begin{bmatrix} \ddot{y}_1 \\ \ddot{y}_2 \end{bmatrix} = \begin{bmatrix} -\frac{k}{m}|v_{rel}|\left(\dot{y}_1 - v_w\right) \\ -\frac{k}{m}|v_{rel}|\dot{y}_2 - g \end{bmatrix}, \tag{4}$$

with $|v_{rel}| = \sqrt{\left(\dot{y}_1 - v_w\right)^2 + \dot{y}_2^2}$. By defining $y_3 = \dot{y}_1$, $y_4 = \dot{y}_2$ and the state vector $\boldsymbol{y} = [y_1 \;\; y_2 \;\; y_3 \;\; y_4]^T$ we can reformulate equation 4 to a system of 1st order (non linear) differential equations:

$$\dot{\boldsymbol{y}} := f(\boldsymbol{y}) = \begin{bmatrix} y_3 \\ y_4 \\ -\frac{k}{m}|v_{rel}|\left(y_3 - v_w\right) \\ -\frac{k}{m}|v_{rel}|y_4 - g \end{bmatrix}. \tag{5}$$

3 Choice and description of suited numerical solvers

Since we are not able to solve boundary value problems (RWP) with the known methods, but initial value problems (AWP), the task is to transform this RWP into an AWP. We already know the the initial value of the first and the second state $\boldsymbol{y}_a = [y_1(0) \;\; y_2(0)]^T$ and we can guess an initial value for the both other states $\boldsymbol{\eta} = [y_3(0) \;\; y_4(0)]^T$. We also know the final value of the 1st and 2nd state $\boldsymbol{y}_b = [y_1(t_f) \;\; y_2(t_f)]$. With this informations we can set up the residual function $\boldsymbol{F}$:

$$\boldsymbol{F}(\boldsymbol{\eta}) := \boldsymbol{y}_{\boldsymbol{\eta}}(t_f) - \boldsymbol{y}_b, \tag{6}$$

where $\boldsymbol{y}_{\boldsymbol{\eta}}(t_f)$ is the evaluation at the final time t_f of the solution of the AWP given in equation 5 with the initial condition $\boldsymbol{y}(0) = [\boldsymbol{y}_a \;\; \boldsymbol{\eta}]^T$. If we can find a $\boldsymbol{\eta}$ such that $||\boldsymbol{F}||$ goes to zero, we have found a solution for the RWP.

3.1 *Newton* - method

One way for finding a zero of a certain function is the *Newton*-method. We guess an initial $\boldsymbol{\eta}_0$ and calculate the following within a loop while $||\boldsymbol{F}(\boldsymbol{\eta}_k)|| > \texttt{tol}$:

$$\boldsymbol{\eta}_{k+1} = \boldsymbol{\eta}_k - \left(\boldsymbol{JF}(\boldsymbol{\eta}_k)\right)^{-1}\boldsymbol{F}(\boldsymbol{\eta}), \tag{7}$$

where $\boldsymbol{JF}$ is the Jacobian of $\boldsymbol{F}$. So we have to solve in each iteration step $n+1$ initial value problems, where n is the dimension of $\boldsymbol{F}$ (in this case $n = 2$). This is quite expensive.

3.2 *Broyden* - method

With a given $\boldsymbol{\eta}_0$ and a given approximation to the Jacobian $\boldsymbol{JF}_0$ we can set up the following iteration loop:

$$\boldsymbol{s}_k = -\left(\boldsymbol{JF}_k\right)^{-1}\boldsymbol{F}(\boldsymbol{\eta}_k) \tag{8}$$

$$\boldsymbol{\eta}_{k+1} = \boldsymbol{\eta}_k + \boldsymbol{s}_k \tag{9}$$

$$\boldsymbol{JF}_{k+1} = \boldsymbol{JF}_k + \frac{\boldsymbol{F}(\boldsymbol{\eta}_{k+1})\boldsymbol{s}_k}{||\boldsymbol{s}_k||^2}. \tag{10}$$

This is called the *Broyden* - method and needs to solve the initial value problem only once per iteration step, which is much cheaper than *Newton* (eq. 7).

3.3 *Dormand* and *Price* method

To solve the initial value problem the *Dormand* and *Price* method (DOPRI5) is used in both cases *Newton* and *Broyden*. This is an explicit 7-staged *Runge-Kutta* method (RKV) with an order of error $p = 5$ and an embedded step size control. The general form of a RKV looks like :

$$\begin{aligned} \boldsymbol{k}_i &= f\left(x_n + a_i h_n, \boldsymbol{y}_n + h_n \sum_{j=1}^{i-1} b_{ij}\boldsymbol{k}_j\right) \quad (i = 1, ..., s) \\ \boldsymbol{y}_{n+1} &= \boldsymbol{y}_n + h_n \sum_{i=1}^{s} c_i \boldsymbol{k}_i \end{aligned}$$

with a *Butcher*-scheme (the values can be looked up in the script p. 25) $\begin{array}{c|c} \boldsymbol{a} & \boldsymbol{B} \\ \hline p = 5 & \boldsymbol{c}^T \\ \hline \hat{p} = 4 & \hat{\boldsymbol{c}}^T \end{array}$.

Now the trick is that there exists a 2nd coefficient vector $\hat{\boldsymbol{c}}$ to the same $\boldsymbol{a}$ and $\boldsymbol{B}$ which results in an order of error $\hat{p} = 4$. So its very cheap to build an error estimator φ to control the step size h_n :

$$\varphi = \left| h_n \left(\sum_{i=1}^{s} c_i \boldsymbol{k}_i - \sum_{i=1}^{s} \hat{c}_i \boldsymbol{k}_i \right) \right| \tag{11}$$

So we can increase the step size if φ is small and decrease if the estimator exceeds the tolerance.

4 Description and evaluation of the results

Figure 2 illustrates the trajectory of the tennis ball and shows the (scaled) initial velocity $\boldsymbol{\eta}$. Both methods result in nearly the same outcome. The difference $err = \left|\boldsymbol{\eta}_{\text{Newton}} - \boldsymbol{\eta}_{\text{Broyden}}\right| \approx$ `1e-5` is for practical applications negligible. Another thing to observe is that *Newton* converges a bit faster but needs more total function evaluations.

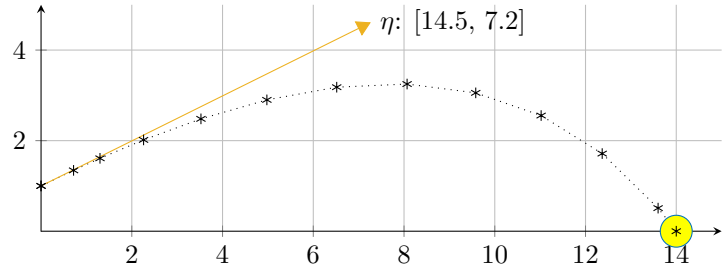

Figure 2: Illustration of the result

In table 1 some statistics of the solving process are listed. It is easy to see that the *Broyden*-method overcomes in almost all categories. And we can expect that if the dimension of the problem and the residual function respectively would be higher, the difference between the total function evaluations and thereby the computational times would be very much larger as well. Since *Newton* needs $n + 1$, *Broyden* only 1 solution of the AWP per iteration step.
So it is reasonable to prefer the *Broyden*-method to solve the boundary value problem.

method	iteration steps	total fun evals	computational time [sec]
Newton	3	1010	0.0444
Broyden	4	509	0.0238

Table 1: Evaluation of the results