Numerical Optimal Control, August 2014

Exercise 4: Solving BVPs with Newtons Method

Joel Andersson Joris Gillis Greg Horn Rien Quirynen Moritz Diehl

University of Freiburg – IMTEK, August 5th, 2014

A two-point boundary-value problem

Consider the following two-point boundary-value problem, describing a person throwing a ball against a target:

$$\left\{\begin{array}{rcl} \dot{p}_x &=& v_x \\ \dot{v}_x &=& -\alpha\, v_x \sqrt{v_x^2+v_y^2} \\ \dot{p}_y &=& v_y \\ \dot{v}_y &=& -\alpha\, v_y \sqrt{v_x^2+v_y^2} - g_0 \end{array}\right. \qquad \left\{\begin{array}{rcl} p_x(0) &=& 0 \\ v_x(0) &=& v_{x,0} \\ p_y(0) &=& h \\ v_y(0) &=& v_{y,0} \end{array}\right. \qquad \left\{\begin{array}{rcl} p_x(T) &=& d \\ v_x(T) &=& v_{x,T} \\ p_y(T) &=& 0 \\ v_y(T) &=& v_{y,T} \end{array}\right. \tag{1}$$

The ball leaves the hand of the thrower with a velocity $(v_{x,0}, v_{y,0})$ a distance $h = 1.5$ m above the ground. It then follows an unguided trajectory determined by standard gravity $g_0 = 9.81$ m/s^2 and air friction $\alpha = 0.02$ hitting a target on the ground $d = 20$ m away after $T = 3$ s. The problem is to determine $(v_{x,0}, v_{y,0})$.

Tasks:

4.1 Use the RK4 integrator scheme from Exercise 3 with 20 steps to simulate the trajectory of the ball assuming assuming $v_{x,0} = v_{y,0} = 5$ m/s.

4.2 Rewrite the integrator using only CasADi symbolics in order to get an `MXFunction` that given $v_0 := (v_{x,0}, v_{y,0})$ returns $p_T := (p_{x,T}, p_{y,T})$. Do it as follows:

- Start by forming an `SXFunction` instance which takes one input x and returns one output $\dot{x}$, i.e.

```
px = SX.sym("px");    vx = SX.sym("vx")
py = SX.sym("py");    vy = SX.sym("vy")
x = vertcat([px,vx,py,vy])
px_dot = ...
...
x_dot = vertcat([px_dot,vx_dot,py_dot,vy_dot])
f = SXFunction([x],[x_dot])
f.init()   ← do not forget!
```

- Then create an `MXFunction` instance with one input (v_0) and one output (p_T) that contains *call* to `f` as described in Section 4.3 of the user guide. Your code should look something like this:

```
v0 = MX.sym("v0",2)
px_0 = 0.; vx_0 = v0[0]; py_0 = 1.5.; vy_0 = v0[1]
x = vertcat([px_0,vx_0,py_0,vy_0])
for k in range(N):
    # RK4
    [k1] = f([x])    ← creates a function call to f!
    [k2] = ...
    ...
    x = ...
pT = x[0::2]
F = MXFunction([v0],[pT])
```

- Evaluate this function numerically, as described in Section 4.1 of the user guide. Make sure that the result is consistent with what you obtained in Task 4.1.

4.3 Formulate the two-point boundary-value problem as an NLP as in Exercise 2. Solve with IPOPT to determine v_0.

4.4 Use algorithmic differentiation in CasADi to calculate the Jacobian $\frac{\partial p_T}{\partial v_0}$.

Tip: `F.jacobian()` will generate a new function for calculating the Jacobian.

4.5 Write a full-step Newton method with 10 iterations to solve the root-finding problem

$$p_T = F(v_0). \tag{2}$$

Verify the result by simulating the trajectory as in Task 4.1.

Tip: To solve the lineararized system, use `numpy.linalg.solve` or the fact that:

$$\begin{bmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{bmatrix}^{-1} = \frac{1}{a_{1,1}\,a_{2,2} - a_{1,2}\,a_{2,1}} \begin{bmatrix} a_{2,2} & -a_{1,2} \\ -a_{2,1} & a_{1,1} \end{bmatrix} \tag{3}$$

4.6 **Extra**: Replace the quadratic friction terms $\alpha\, v_x \sqrt{v_x^2 + v_y^2}$ and $\alpha\, v_y \sqrt{v_x^2 + v_y^2}$ in (1) with the linear terms $\alpha\, v_x$ and $\alpha\, v_y$. How does this influence the number of Newton-iterations needed to solve the problem?

4.7 **Extra**: Replace the RK4 integrator with CVODES as in Task 3.4 of Exercise 3. Do you get the same result?