# Flatness-Based Control: Kinematic Car

Tutorial 2: open-loop controller

D. Kastelan
d.kastelan@lsr.uni-saarland.de
Chair of Systems Theory and Control Engineering
Saarland University

In this tutorial, an open-loop controller and a suitable trajectory generator will be implemented for the kinematic car and included in the existing simulation from Tutorial 1. The open-loop controller (see notes §1.7, p. 10–14) written in terms of the reference flat output trajectory $t \mapsto (y_{1,r}(t), y_{2,r}(t))$ is given by

$$v_r(t) = \pm\sqrt{\dot{y}_{1,r}^2(t) + \dot{y}_{2,r}^2(t)} \tag{1.9a}$$

$$\varphi_r(t) = \arctan\left(\frac{l\left(\ddot{y}_{2,r}(t)\dot{y}_{1,r}(t) - \ddot{y}_{1,r}(t)\dot{y}_{2,r}(t)\right)}{v_r^3(t)}\right). \tag{1.9b}$$

Each component of the flat output trajectory may be implemented independently as a polynomial $[0, T] \ni t \mapsto y_i(t) = \sum_{j=0}^{n} c_{i,j} t^j$, $i = 1, 2$ given values $(y_i(t), \dot{y}_i(t), \ddot{y}_i(t)), i = 1, 2$ at $t = 0$ and $t = T$.

## Tasks

1. What minimum polynomial degree $n$ is required such that the system input $(v, \varphi) = (v_r(t), \varphi_r(t))$ is smooth?

2. Calculate the polynomial coefficients $c_{i,j}$, $j = 0, \ldots, n$ by numerically inverting an appropriate system of equations. Implement this calculation in the constructor for the Matlab class `PolyRef` in provided file `common/PolyRef.m`. Use the `polyder` function to compute the polynomial coefficients for the first two time derivatives as well. Save these in properties `p_y`, `p_dy`, `p_ddy`, respectively. Complete the `eval` method to evaluate the trajectory and its first two time derivatives. The `plot` method may be used to check your results.

3. (∗) The spline interpolation function `spapi` in Matlab's curve fitting toolbox may be used to quickly generate polynomial trajectories. Derivatives thereof may be calculated using function `fnder`. Use this approach to compute the trajectory parameters `ps_y`, `ps_dy`, `ps_ddy` in the `PolyRef` constructor. Evaluate the trajectory and its first two time derivatives in `eval_spline` (see the function `spval`). Compare your results with those from the previous task using the `plot_spline` method.

4. Implement the open-loop (feed-forward) controller from (1.9) in the `eval` method of the new Matlab class `KinematicCarControl`. Consider the positive branch of the square root function only. The state $(y_{1,r}, y_{2,r}, \theta_r)$ corresponding to the reference trajectory should also be computed. Make sure that your calculations accept vector-valued variables by performing element-wise operations (e.g., `a.*b` or `a./b` rather than `a*b` or `a/b` for multiplication and division, respectively). Initialize the controller in the main file and test your implementation using the references

```
r1=PolyRef(t0,t1,y10,5,1,0.5,0,0);
r2=PolyRef(t0,t1,y20,2,0,0,0,0);
```

for $t \mapsto y_1(t)$ and $t \mapsto y_2(t)$, respectively and an initial condition $(y_1(0), y_2(0), \theta(0)) = (0, 0, 0)$.

5. The trajectories in the previous task are compatible with the system's constraints and its initial conditions since they are tracked by the ideal kinematic car. Find polynomial trajectories compatible with the initial conditions $(y_1(0), y_2(0), \theta(0)) = (0, 0, \pi/2)$.

6. The relations (1.9) are singular for $v_r(t) = 0$. Command a reference trajectory containing this singularity and describe its effect on the reference and simulated variables. Are the results as you expect?

7. Consider now the negative branch of the square root function in (1.9). Simulate using the polynomial references from task 4 and the initial condition $(y_1(0), y_2(0), \theta(0)) = (0, 0, \pi)$. What effect does $v_r(t) < 0$ have? Why does $\theta_r = \arctan \frac{\dot{y}_{2,r}}{\dot{y}_{1,r}}$ appear to return the "wrong" value when $v_r(t) < 0$?