Dynamic Programming for Trajectory through Doors of a UAV

叶子 27.03.2019

"uav_time_opt.m" implements a dynamic optimization for UAV trajectory with the open source library CASADI. To run this program, you need to import the library first. You might have to modify the following lines.

```
addpath('../../../casadi')
import casadi.*;
```

The input variables are the thrust forces at four rotors. The state variables are position (x, y, z), velocity (x, y, z), angles (roll, pitch, yaw), and angular rates (roll, pitch, yaw).

$$x = [p_x\ p_y\ p_z\ v_x\ v_y\ v_z\ \phi\ \psi\ \theta\ \omega_x\ \omega_y\ \omega_z]^T$$

The objective is to minimize the weighted cost of consumed energy by rotors, under constraints of UAV dynamics, bounds of states and inputs, starting and final conditions. The constraints also take the way points into consideration. In this example, there are two way points. The UAV flies pass these way points at a given state (given position, velocity etc. accordingly). These way points separate the overall time into time intervals. The objective functional is accumulated in each of the intervals.

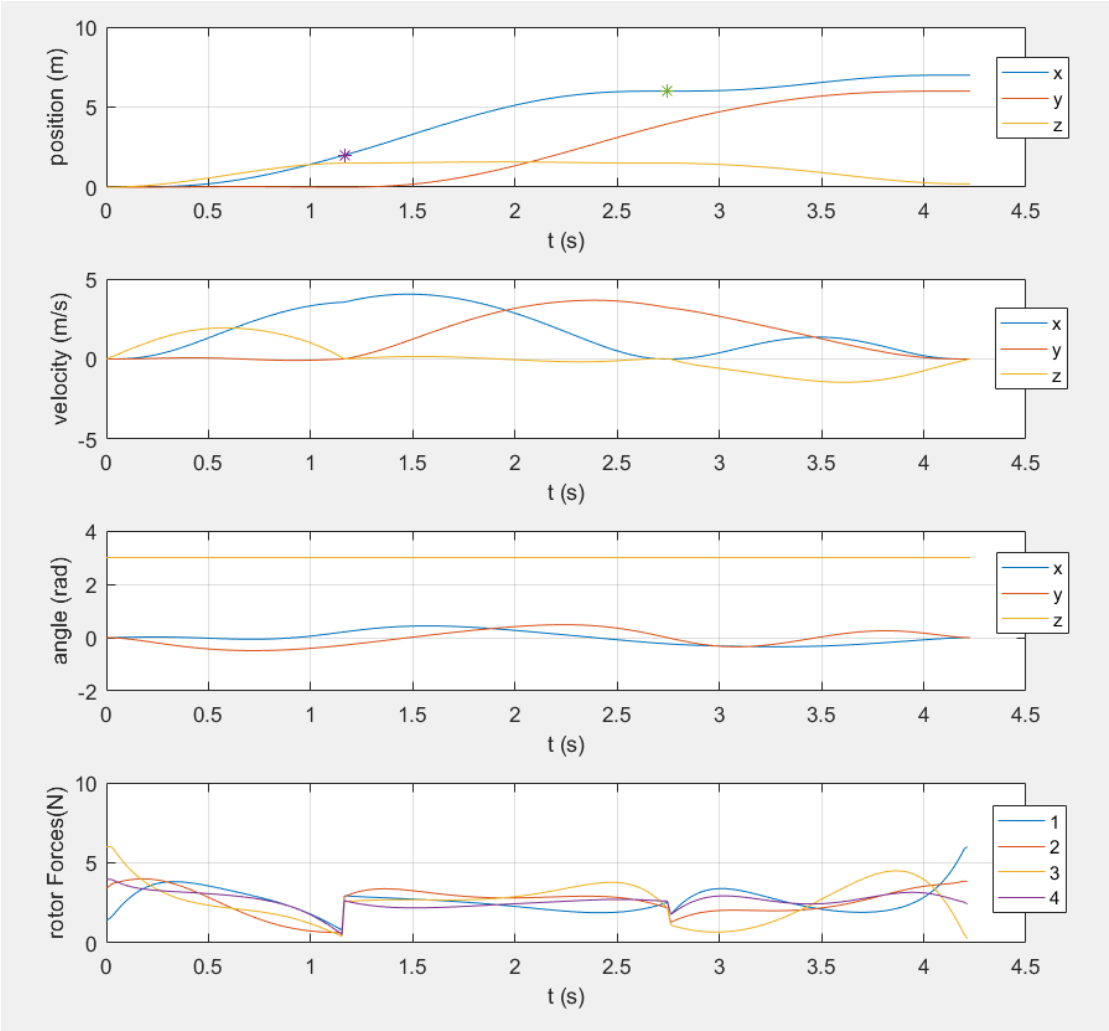
$$J = \int_{t_0}^{t_1} u^T(t)Ru(t)\mathrm{d}t + \int_{t_1}^{t_2} u^T(t)Ru(t)\mathrm{d}t + \int_{t_2}^{t_3} u^T(t)Ru(t)\mathrm{d}t$$

s.t.

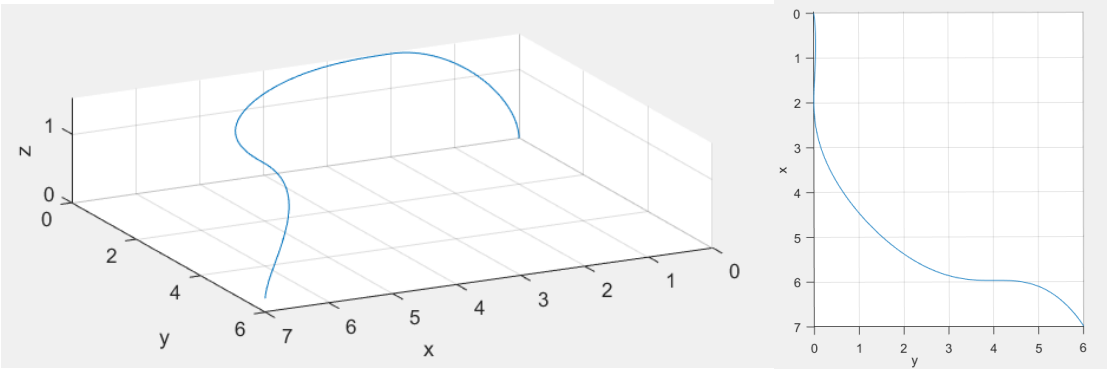
$$\frac{dx(t)}{dt} = f(t, x(t), u(t))$$
$$u(\cdot) \in [0, u_{max}]$$
$$v_{x,y,z}(\cdot) \in [-v_{max}, v_{max}]$$
$$\phi(\cdot) \in [-\phi_{max}, \phi_{max}]$$
$$\psi(\cdot) \in [-\psi_{max}, \psi_{max}]$$
$$x(t_0) = [p_{x0}\ p_{y0}\ p_{z0}\ 0\ 0\ 0\ 0\ 0\ \theta_0\ 0\ 0\ 0]^T$$
$$p(t_1) = p_1$$
$$v(t_1) = v_1$$
$$p(t_2) = p_2$$
$$v(t_2) = v_2$$
$$x(t_3) = [p_{x3}\ p_{y3}\ p_{z3}\ 0\ 0\ 0\ 0\ 0\ \theta_3\ 0\ 0\ 0]^T$$

For this free end time problem, time transformation is applied for $i = 1,2,3$.

$$t_i = \tau t_{fi}$$
$$\tau \in [0, 1]$$
$$\frac{dx(\tau)}{d\tau} = t_{fi} f(\tau, x(\tau), u(\tau))$$



The two way points are marked once in the position plot with "*". To be noticed is that the output changes rapidly at these two points, which might cause instability!



The figures above are the 3D view and top-down view of the trajectory with given starting states and end states, passing through way points [2;0;1.5] and [7;6;0.2] with desired velocity directions

The next move is to implement these in python, which is compatible in ROS system. The program

should then be able to publish control set points including position, velocity, acceleration (attitude), and angular velocity. It can then be simulated at real time in GAZEBO. These have been realized. https://github.com/waltYeh/rotors_simulator

Dynamic Programming for Grasping of a UAV

"uav_grasp_opt_var_mass.m" implements a dynamic optimization for UAV grasping with the open source library CASADI.

The model is built in a plane, in which the front and rear rotor of the UAV lie. The arm can rotate only in this plane. Another simplification is that the joint between arm and UAV is at the center of mass of this UAV.

The input variables are the total thrust forces of four rotors T, the torque generated by the front and rear rotor M, and the torque of the servo at the joint τ.

$$u = [T, M, \tau]^T$$

The state variables are horizontal position of UAV, vertical position of UAV, pitch angle of UAV, angle of arm with respect to horizon, and also their derivatives.

$$x = [x_q \ z_q \ \theta \ \beta \ \dot{x}_q \ \dot{z}_q \dot{\theta} \ \dot{\beta}]^T$$

The objective is to minimize the weighted cost of consumed energy of inputs, and the angle between the UAV and arm should not be far away from 90 degrees. The dynamics of the system can be obtained from Lagrange mechanics. Bear in mind that the payload $m_P(t)$ is added only in the second interval. Other constraints include bounds of states and inputs, starting and final conditions. The constraints also take the state of grasping into consideration. When the UAV flies pass the object, the end point of arm should meet the object at the absolute velocity of zero. The objective functional is accumulated in the intervals before and after grasping.

$$J = \int_{t_0}^{t_1} u^T(t)Ru(t) + (\beta - \theta - \frac{\pi}{2})^T \mathrm{Q}(\beta - \theta - \frac{\pi}{2})\mathrm{dt} + \int_{t_1}^{t_2} u^T(t)Ru(t) + (\beta - \theta - \frac{\pi}{2})^T \mathrm{Q}(\beta - \theta - \frac{\pi}{2})\mathrm{dt}$$

s.t.

$$\frac{dx(t)}{dt} = f(t, x(t), u(t), m_P(t))$$

$$T(\cdot) \in [0, T_{max}]$$

$$M(\cdot) \in [-M_{max}, M_{max}]$$

$$\tau(\cdot) \in [-\tau_{max}, \tau_{max}]$$

$$\gamma = \beta - \theta \in [0, \pi]$$

$$\dot{x}_g(\cdot) \in [-v_{max}, v_{max}]$$

$$\dot{z}_g(\cdot) \in [-v_{max}, v_{max}]$$

$$\theta(\cdot) \in [-\theta_{max}, \theta_{max}]$$

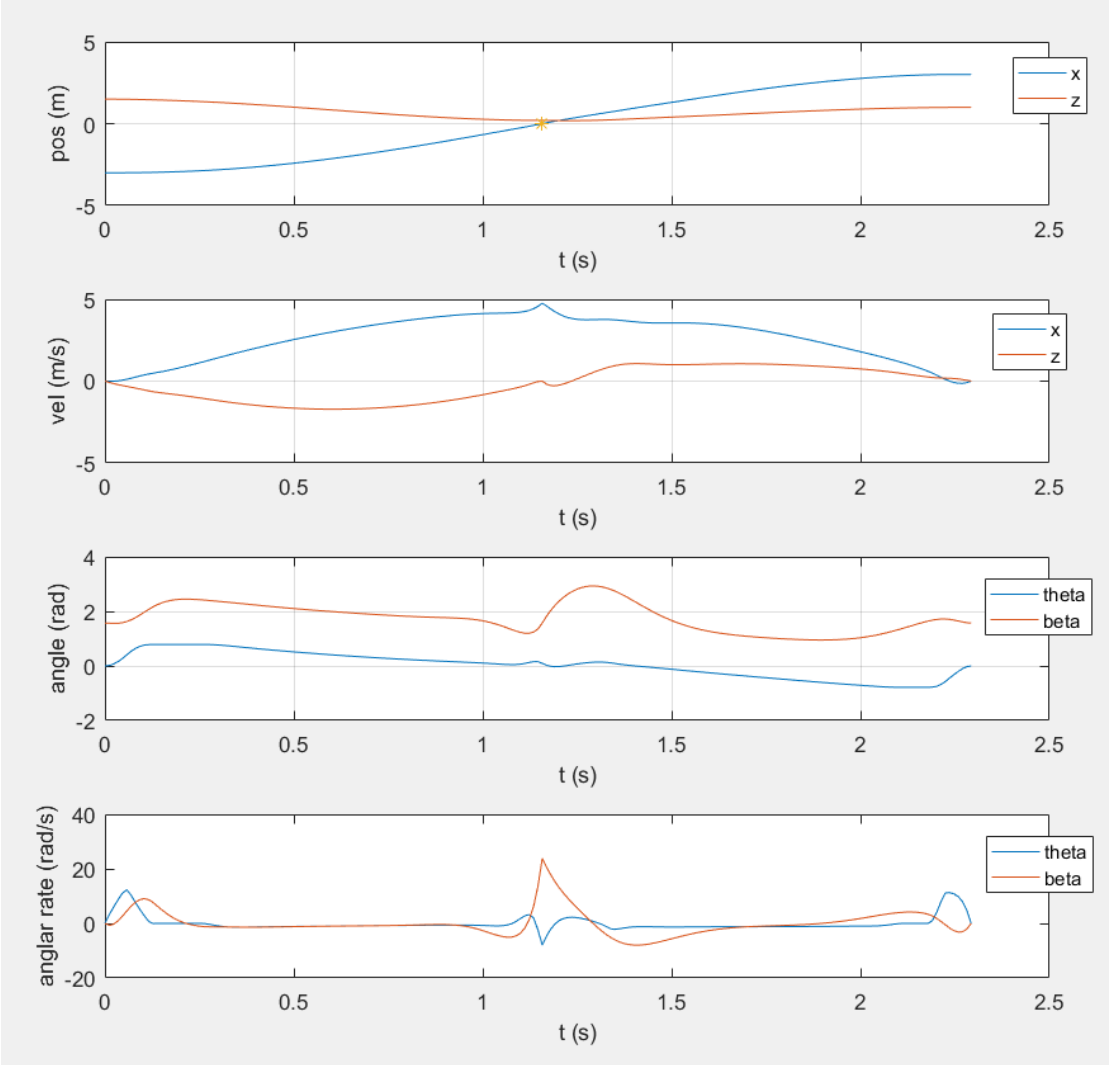
$$x(t_0) = [x_{q0} \ z_{q0} \ 0 \ \frac{\pi}{2} \ 0 \ 0 \ 0 \ 0]^T$$

$$x(t_2) = [x_{q2} \ z_{q2} \ 0 \ \frac{\pi}{2} \ 0 \ 0 \ 0 \ 0]^T$$

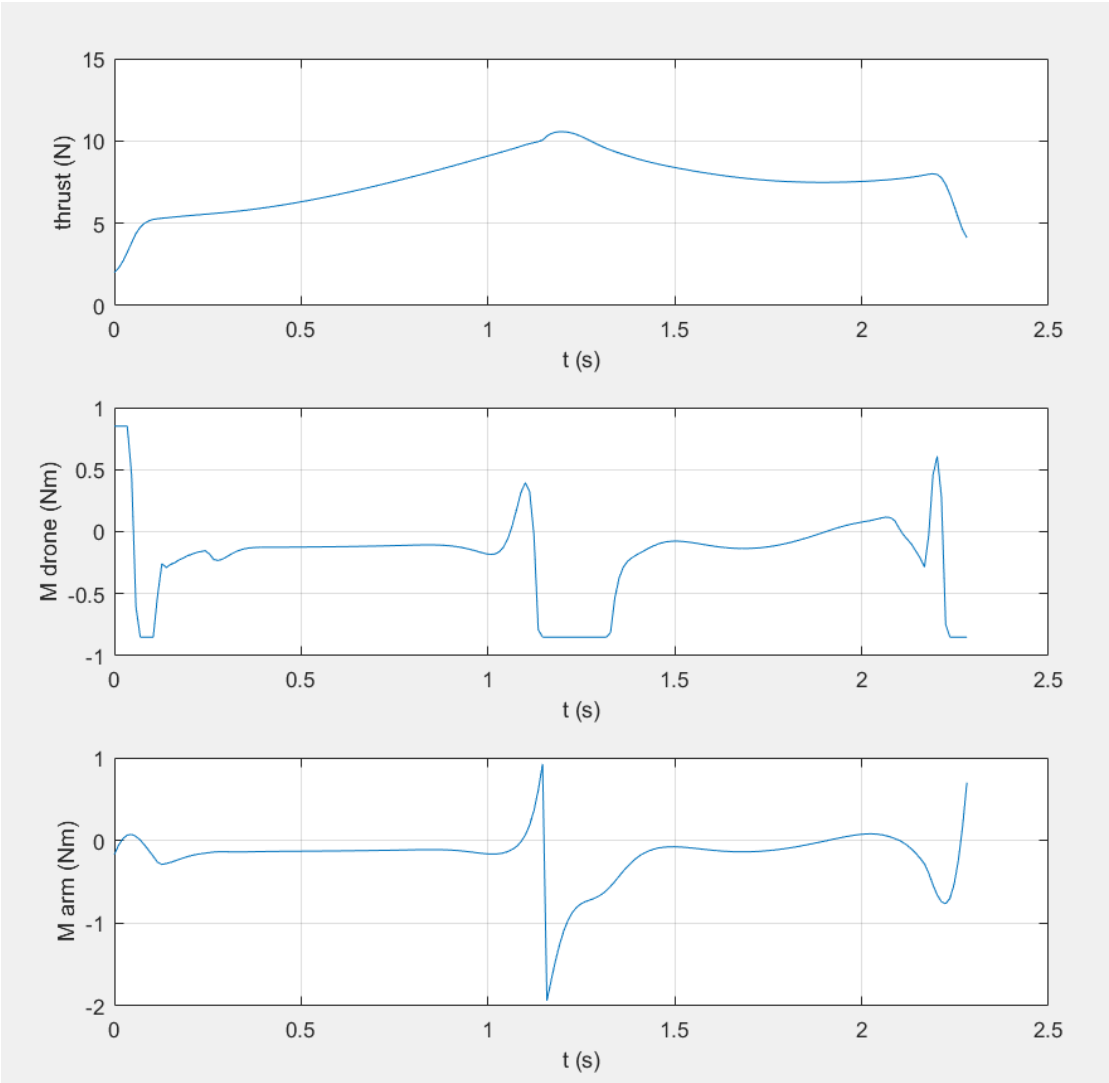
$$x_g(t_1) = x_q(t_1) + L_{arm}\cos(\beta(t_1)) = x_{obj}$$
$$z_g(t_1) = z_q(t_1) - L_{arm}\sin(\beta(t_1)) = z_{obj}$$
$$\dot{x}_g(t_1) = \dot{x}_q(t_1) - \dot{\beta}(t_1)L_{arm}\sin(\beta(t_1)) = 0$$
$$\dot{z}_g(t_1) = \dot{z}_q(t_1) - \dot{\beta}(t_1)L_{arm}\cos(\beta(t_1)) = 0$$

For this free end time problem, time transformation is applied for $i = 1,2$.

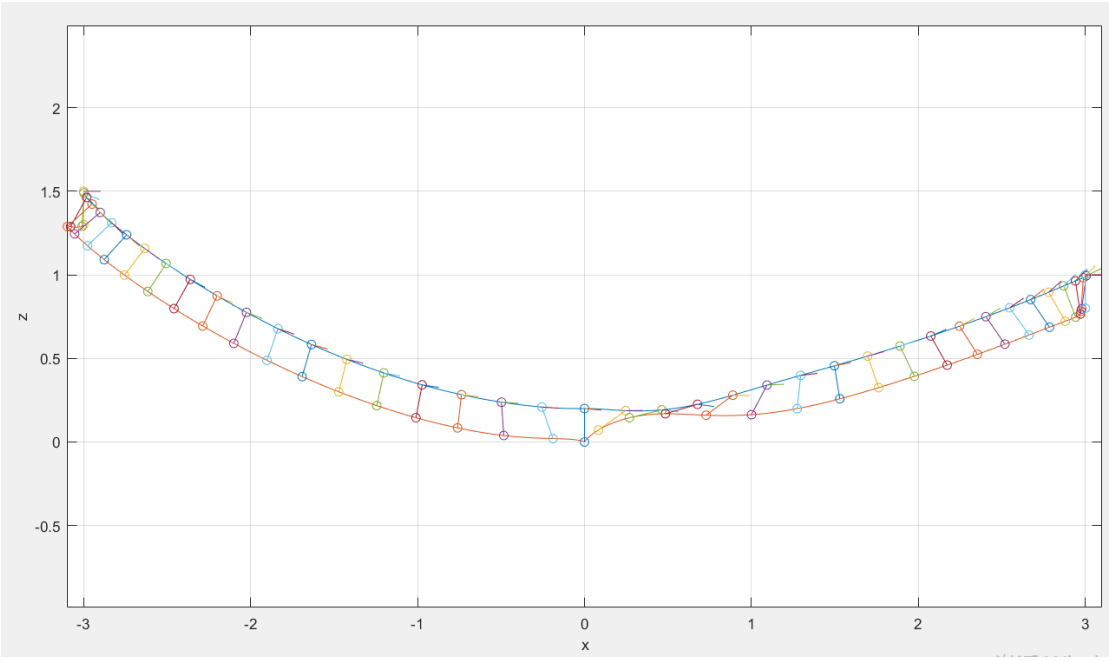
$$t_i = \tau t_{fi}$$
$$\tau \in [0,1]$$
$$\frac{dx(\tau)}{d\tau} = t_{fi} f(\tau, x(\tau), u(\tau))$$



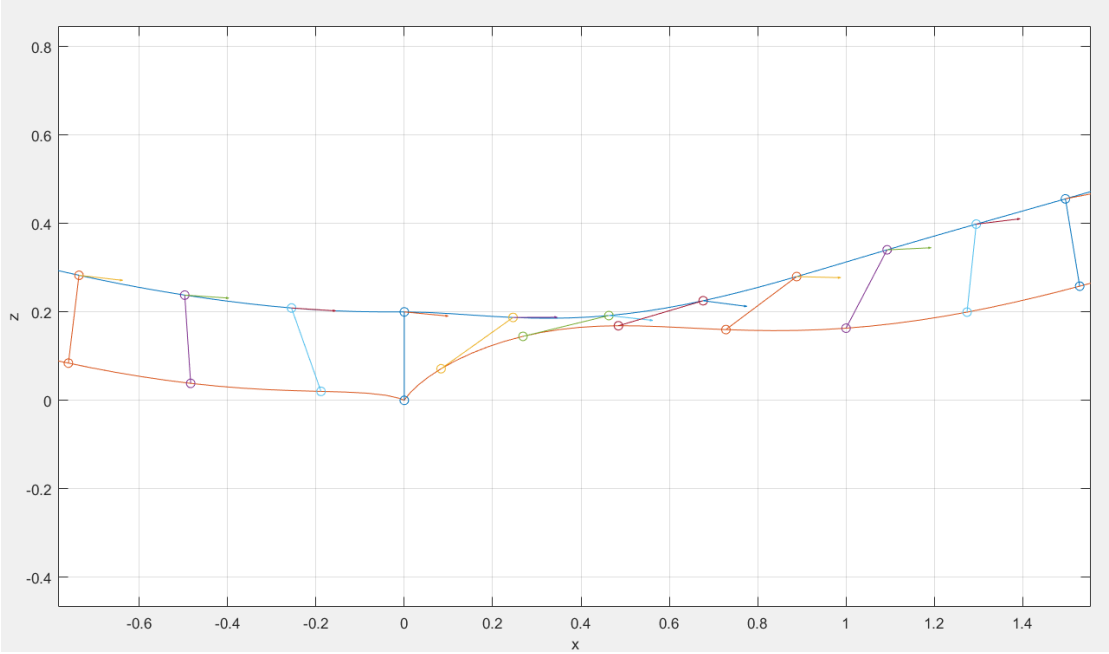
The states with respect to time. The grasping state is marked once in the position plot with "*".



The plotting of system inputs.



The figure above is the trajectory of UAV and the trajectory of the end point of arm. The quivers show the incline angles of the UAV at some time points, and the line connecting two trajectories shows the poses of arm at some time points.



Detailed view of grasping at x=0, z=0

The next move is to implement these in python, which is compatible in ROS system. Also important is to build a model in GAZEBO. It can then be simulated at real time.