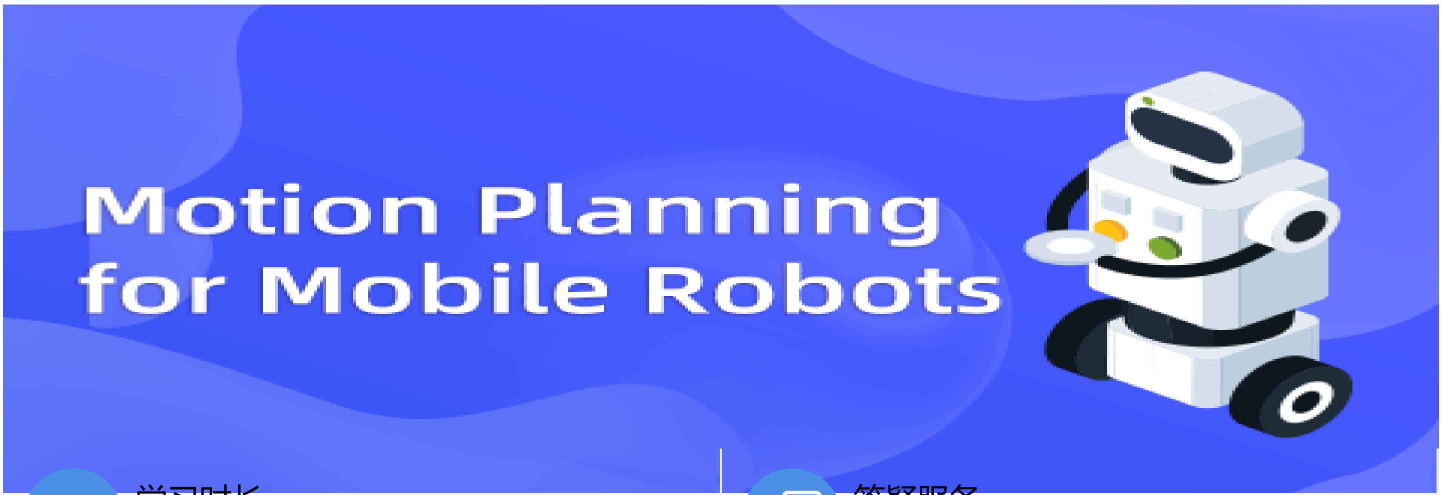
(/)

搜索

课程中心 (/courselist) 公开课 (/open/course/exp



学习时长

八周
建议每周至少六小时

答疑服务

专属微信答疑群
讲师助教均参与

作业批改

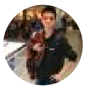
每章节设计作业
助教及时批改评优

课程奖学金

1万元YOGO Robot奖学金

返回讨论区 (/course/188/threads/show) / 话题

第五章作业讲评

 (/user/47491)

By 朱江超 (/user/47491) • 12-09 • 212次浏览

深蓝学院
shenlanxueyuan.com

Lecture 5

MINIMUM SNAP TRAJECTORY GENERATION 作业讲解

主讲人 朱江超

Implement minimum snap in Matlab and C++/ROS

Homework 1.1: In matlab, use the quadprog QP solver, write down a minimum snap trajectory generator

Homework 1.2: In matlab, generate minimum snap trajectory based on the closed form solution

Homework 2.1: In C++/ROS, use the OOQP solver, write down a minimum snap trajectory generator

Homework 2.2: In C++/ROS, use Eigen, generate minimum snap trajectory based on the closed form solution

符号约定

d_order: 优化目标的阶数

n_order: 多项式的阶数

n_seg: 轨迹的段数

K: 导数的阶数计数器

j: 轨迹计数器

i: 多项式系数计数器

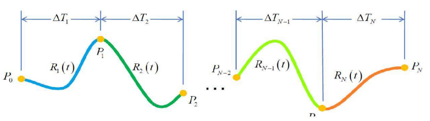
How to determine the trajectory order?

- Ensure smoothness at an order.
- Ensure continuity at an order.
- Minimize control input at an order.

This three items are not coupled!

- Minimum degree polynomial to ensure smoothness for one-segment trajectory:
  - Minimum jerk: N = 2 * 3(jerk) - 1 = 5
  - Minimum snap: N = 2 * 4(snap) - 1 = 7

- Minimum degree polynomial to ensure smoothness for k-segment trajectory:



Minimum jerk:

Constraints num: 3 + 3 + (k-1) = k + 5

Unknowns num: (N+1) * k

$(N+1) * k = k + 5 \qquad N = \frac{5}{k}$

HW 1.1

代码流程

```
function poly_coef = MinimumSnapQPSolver(waypoints, ts, n_seg, d_order)
    start_cond = [waypoints(1), 0, 0, 0];
    end_cond   = [waypoints(end), 0, 0, 0];

    %#################################################
    % STEP 1: compute Q of p'Qp
    Q = getQ(n_seg, d_order, ts);

    %#################################################
    % STEP 2: compute Aeq and beq
    [Aeq, beq] = getAbeq(n_seg, d_order, waypoints, ts, start_cond, end_cond);

    f = zeros(size(Q,1),1);
    poly_coef = quadprog(Q,f,[],[],Aeq, beq);
end
```

HW 1.1

getQ()

$$f(t)=\sum_i p_i t^i$$
$$\Longrightarrow f^{(4)}(t)=\sum_{i\geq 4} i(i-1)(i-2)(i-3)t^{i-4}p_i$$
$$\Longrightarrow \left(f^{(4)}(t)\right)^2=\sum_{i\geq 4,l\geq 4} i(i-1)(i-2)(i-3)l(l-1)(l-2)(l-3)t^{i+l-8}p_i\,p_l$$
$$\Longrightarrow J(T)=\int_0^T \left(f^4(t)\right)^2 dt=\sum_{i\geq 4,l\geq 4}\frac{i(i-1)(i-2)(i-3)l(l-1)(l-2)(l-3)}{i+l-7}T^{i+l-7}p_i\,p_l$$
$$\Longrightarrow J(T)=\int_0^T \left(f^4(t)\right)^2 dt$$
$$=\begin{bmatrix}\vdots\\ p_i\\ \vdots\end{bmatrix}^T\begin{bmatrix} & \vdots & \\ \cdots & \frac{i(i-1)(i-2)(i-3)l(l-1)(l-2)(l-3)}{i+l-7}T^{i+l-7} & \cdots\\ & \vdots & \end{bmatrix}\begin{bmatrix}\vdots\\ p_l\\ \vdots\end{bmatrix}$$
$$\Longrightarrow J_j(T)=\mathbf{p}_j^T\mathbf{Q}_j\mathbf{p}_j$$

```
% d_order = 4;
% n_order = 7;
Q = [];
for j = 0:n_seg-1
    Qj = zeros(8,8);
    for i = 4:7
        for l = i:7
            Qj(i+1, l+1) = factorial(i)/factorial(i-4)* ...
                           factorial(l)/factorial(l-4)* ...
                           ts(j+1)^(i+l-7)/(i+l-7);
            if i ~= l
                Qj(l+1, i+1) = Qj(i+1, l+1);
            end
        end
    end
    Q = blkdiag(Q, Qj);
end
```

HW 1.1

getAbeq()

- Derivative constraint for one polynomial segment
  - Also models waypoint constraint (0^{th} order derivative)

$$f_j^{(k)}(T_j)=x_j^{(k)}$$
$$\Rightarrow \sum_{i\geq k}\frac{i!}{(i-k)!}T_j^{i-k}p_{j,i}=x_{T,j}^{(k)}$$
$$\Rightarrow \begin{bmatrix}\cdots & \frac{i!}{(i-k)!}T_j^{i-k} & \cdots\end{bmatrix}\begin{bmatrix}\vdots\\ p_{j,i}\\ \vdots\end{bmatrix}=x_{T,j}^{(k)}$$
$$\Rightarrow \begin{bmatrix}\cdots & \frac{i!}{(i-k)!}T_{j-1}^{i-k} & \cdots\\ \cdots & \frac{i!}{(i-k)!}T_j^{i-k} & \cdots\end{bmatrix}\begin{bmatrix}\vdots\\ p_{j,i}\\ \vdots\end{bmatrix}=\begin{bmatrix}x_{0,j}^{(k)}\\ x_{T,j}^{(k)}\end{bmatrix}$$
$$\Rightarrow \mathbf{A}_j\mathbf{p}_j=\mathbf{d}_j$$

```
%#####################################################
% derivative constraints of start point
Aeq_start = zeros(4, n_all_poly);
for k = 0:3
    Aeq_start(k+1, k+1) = factorial(k);
end
beq_start = start_cond';

%#####################################################
% derivative constraints of end point
Aeq_end = zeros(4, n_all_poly);
start_idx_2 = (n_order+1)*(n_seg-1);
for k = 0:3
    for i = k:7
        Aeq_end(k+1, start_idx_2+i+1) = factorial(i)/factorial(i-k)...
                                        *ts(n_seg)^(i-k);
    end
end
beq_end = end_cond';

%#####################################################
% position constraints of middle waypoints
Aeq_wp = zeros(n_seg-1, n_all_poly);
beq_wp = zeros(n_seg-1, 1);
for j = 0:n_seg-2
    start_idx_2 = (n_order+1)*(j+1);
    Aeq_wp(j+1, start_idx_2+1) = 1.0;
    beq_wp(j+1, 1) = waypoints(j+2);
end
```

HW 1.1

getAbeq()

- Continuity constraint between two segments:
  - Ensures continuity between trajectory segments when no specific derivatives are given

$$f_j^{(k)}(T_j) = f_{j+1}^{(k)}(T_j)$$
$$\Rightarrow \sum_{i \geq k} \frac{i!}{(i-k)!} T_j^{i-k} p_{j,i} - \sum_{l \geq k} \frac{l!}{(l-k)!} T_j^{l-k} p_{j+1,l} = 0$$
$$\Rightarrow \begin{bmatrix} \cdots & \frac{i!}{(i-k)!} T_j^{i-k} & \cdots & -\frac{l!}{(l-k)!} T_j^{l-k} & \cdots \end{bmatrix} \begin{bmatrix} \vdots \\ p_{j,i} \\ \vdots \\ p_{j+1,l} \\ \vdots \end{bmatrix} = 0$$
$$\Rightarrow \begin{bmatrix} \mathbf{A}_j & -\mathbf{A}_{j+1} \end{bmatrix} \begin{bmatrix} \mathbf{p}_j \\ \mathbf{p}_{j+1} \end{bmatrix} = 0$$

```
%#####################################################
% continuity constraints of middle waypoints
Aeq_con = zeros((n_seg-1)*d_order, n_all_poly);
beq_con = zeros((n_seg-1)*d_order, 1);
for k = 0:3
    for j = 0:n_seg-2
        for i = k:7
            start_idx_1 = (n_seg-1)*k;
            start_idx_2 = (n_order+1)*j;
            Aeq_con(start_idx_1+j+1, start_idx_2+i+1) = ...
                                 factorial(i)/factorial(i-k)*ts(j+1)^(i-k);
            if i == k
                Aeq_con(start_idx_1+j+1, start_idx_2+(n_order+1)+i+1) =...
                                                          -factorial(i);
            end
            % the same as the following expression
            % Aeq_con(start_idx_1+j+1, start_idx_2+(n_order+1)+i+1) = ...
            %                     -factorial(i)/factorial(i-k)*0^(i-k);
        end
    end
end
```

HW 1.1

visualize

```
% display the trajectory
X_n = [];
Y_n = [];
k = 1;
tstep = 0.01;
for i=0:n_seg-1
    %#####################################################
    % STEP 3: get the coefficients of i-th segment of both x-axis
    % and y-axis
    start_idx = n_poly_perseg * i;
    Pxi = poly_coef_x(start_idx+1 : start_idx+n_poly_perseg, 1);
    Pxi = flipud(Pxi);
    Pyi = poly_coef_y(start_idx+1 : start_idx+n_poly_perseg, 1);
    Pyi = flipud(Pyi);
    for t=0:tstep:ts(i+1)
        X_n(k)  = polyval(Pxi, t);
        Y_n(k)  = polyval(Pyi, t);
        k = k+1;
    end
end
```

HW 1.2

代码流程

- We have $\boldsymbol{M}\mathbf{p}=\mathbf{d}$, where $\boldsymbol{M}$ is a mapping matrix that maps polynomial coefficients to derivatives
- Use a selection matrix $\mathbf{C}$ to separate free ($\mathbf{d}_P$) and constrained ($\mathbf{d}_F$) variables
  - Free variables : derivatives unspecified, only enforced by continuity constraints

$$\begin{bmatrix}\mathbf{d}_1\\ \vdots \\ \mathbf{d}_M\end{bmatrix} = \mathbf{C}^T\begin{bmatrix}\mathbf{d}_F\\ \mathbf{d}_P\end{bmatrix} \qquad J = \begin{bmatrix}\mathbf{d}_F\\ \mathbf{d}_P\end{bmatrix}^T \underbrace{\mathbf{C}\boldsymbol{M}^{-T}\mathbf{Q}\boldsymbol{M}^{-1}\mathbf{C}^T}_{\mathbf{R}} \begin{bmatrix}\mathbf{d}_F\\ \mathbf{d}_P\end{bmatrix} = \begin{bmatrix}\mathbf{d}_F\\ \mathbf{d}_P\end{bmatrix}^T \begin{bmatrix}\mathbf{R}_{FF} & \mathbf{R}_{FP}\\ \mathbf{R}_{PF} & \mathbf{R}_{PP}\end{bmatrix}\begin{bmatrix}\mathbf{d}_F\\ \mathbf{d}_P\end{bmatrix}$$

- Turned into an unconstrained quadratic programming that can be solved in closed form:

$$J = \mathbf{d}_F^T\mathbf{R}_{FF}\mathbf{d}_F + \mathbf{d}_F^T\mathbf{R}_{FP}\mathbf{d}_P + \mathbf{d}_P^T\mathbf{R}_{PF}\mathbf{d}_F + \mathbf{d}_P^T\mathbf{R}_{PP}\mathbf{d}_P$$

$$\mathbf{d}_P^* = -\mathbf{R}_{PP}^{-1}\mathbf{R}_{FP}^T\mathbf{d}_F$$

HW 1.2

getM()

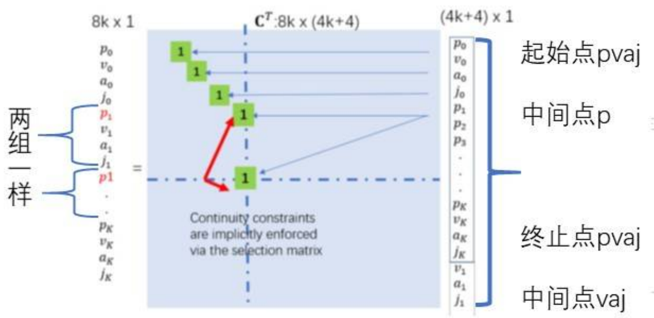
$$\boldsymbol{M}_{\mathrm{j}}\mathbf{p}_j = \mathbf{d}_j$$

$$f_j^{(k)}(T_j) = x_j^{(k)}$$
$$\Rightarrow \sum_{i \geq k} \frac{i!}{(i-k)!} T_j^{i-k} p_{j,i} = x_{T,j}^{(k)}$$

```
M = [];
d_order = 4;
n_order = 7;
for j = 0:n_seg-1
    Mj = zeros(8, 8);
    for k = 0:3
        Mj(k+1, k+1) = factorial(k);
        for i = k:7
            Mj(4+k+1, i+1) = factorial(i)/factorial(i-k)*ts(j+1)^(i-k);
        end
    end
    M = blkdiag(M, Mj);
end
```

HW 1.2

getCt()




```
%########################################################
% Ct for start point
d_order = 4;
Ct_start = zeros(d_order, d_order*(n_seg+1));
Ct_start(:, 1:d_order) = eye(d_order);

%########################################################
% Ct for middle points
Ct_mid = zeros(2*d_order*(n_seg-1), d_order*(n_seg+1));
for j = 0:n_seg-2
    Cj = zeros(d_order, d_order*(n_seg+1));% (j+1)th traj's pos
    Cj(1, d_order+j+1) = 1;
    start_idx_2 = 2*d_order+n_seg-1+3*j;% (j+1)th traj's vel, acc, jerk
    Cj(2:d_order, start_idx_2+1:start_idx_2+3) = eye(d_order-1);
    start_idx_1 = 2*d_order*j;
    Ct_mid(start_idx_1+1:start_idx_1+2*d_order, :) = [Cj;Cj];
end

%########################################################
% Ct for end point
Ct_end = zeros(d_order, d_order*(n_seg+1));
Ct_end(:, d_order+n_seg:2*d_order+n_seg-1) = eye(d_order);
```

HW 2.1

OOQP库的安装

参考：https://github.com/HKUST-Aerial-Robotics/Teach-Repeat-Replan

需要的文件：切换到experiment分支 installation/OOQP.zip ma27-1.0.0.tar.gz

安装步骤：见master分支下的README.md

调用方法：参考OOQP/examples/QpGen/cplusplus/example.c

或Teach-Repeat-Replan 工程中的代码

ROS package: ooqp_eigen_interface

代码：https://github.com/zhujiangchao/ooqp_eigen_interface

安装步骤：build OOQP and ma27 with the -fPIC option

修改CmakeList.txt

调用方法：bool result = ooqpei::OoqpEigenInterface::solve(Q, c, A, b, l, u, x);

贡献学员：caomuqing

4 回复

(/user/45805) suhang_1982 • 12-10

哪里可以下载到源码?

回复

(/user/47491) 朱江超 • 12-10

@suhang_1982 (/user/45805) 什么的源码?

回复

(/user/45805) suhang_1982 • 12-16

C++的源码实现

回复

(/user/47491) 朱江超 • 12-17

@suhang_1982 (/user/45805) C++源码没有开源

回复

添加回复

源码

添加回复