Optimization Method & Optimal Guidance

Haichao Hong

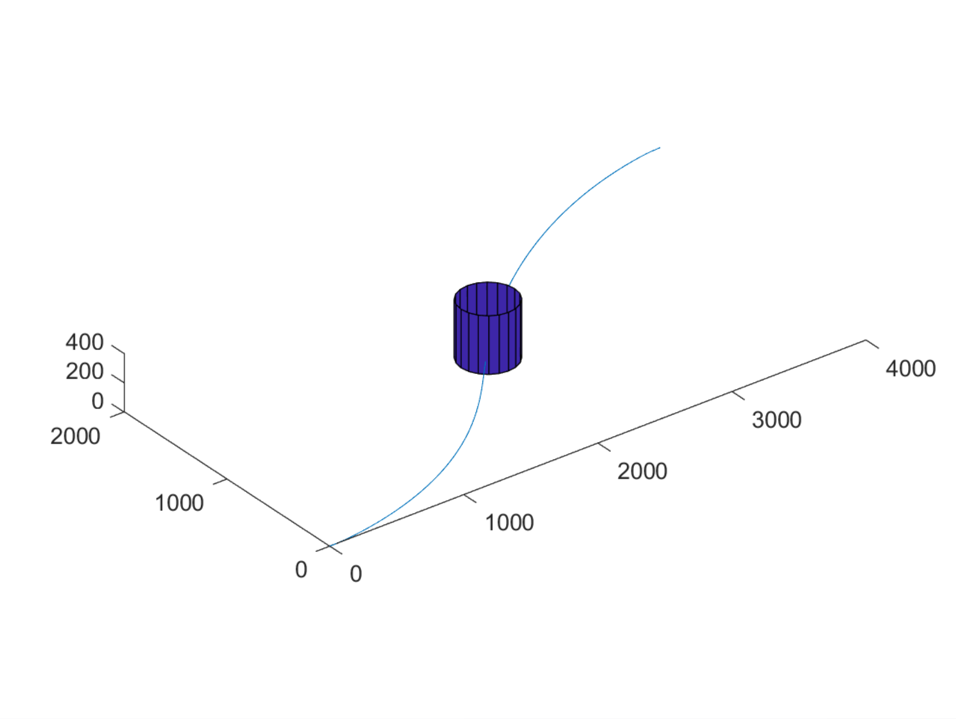
AE8120

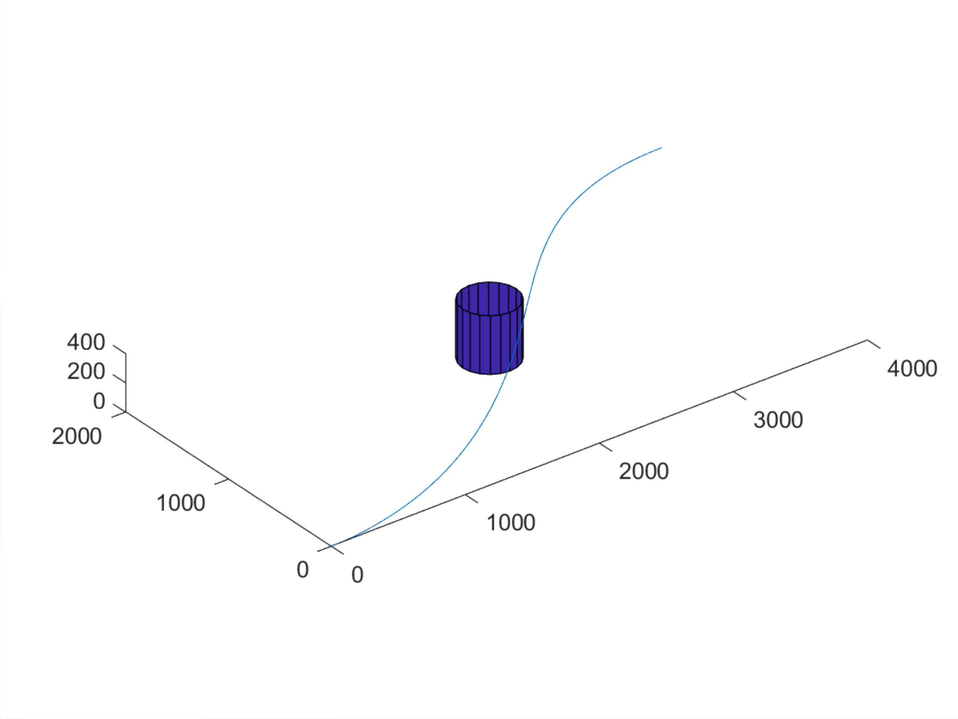
Contents

Flight Trajectory Optimization 3D - Path

Task:

- **Distance to (2100,1200) of a minimum 200 m**





FALCON.m – problem structure

Jacobian Matrix – MATLAB symbolic toolbox

```
syms g m rho S T_max k CD0
syms %add states here
syms %add controls here

states = [];
controls = [];




% Calculate aerodynamics
C_D = CD0 + k * C_L^2;
D = [];
L = [];

% implement state derivatives here



states_dot = [x_dot; y_dot; z_dot; V_dot; chi_dot; gamma_dot];

dFdX = jacobian(states_dot,states)
dFdU = jacobian(,)
```

Tasks:

- **Locate the syms_calculate_jacobians.m**
- **Use the dynamics in source_aircraft3D.m**
- **Calculate the analytical expressions of the Jacobian matrices**

Sequential Convex Optimization

Sequential Convex Optimization: Mission A

To climb 500 m instead of 400 m

Task part I:

- **Open main_fixed.m**
- **Read the code**
- **Complete the forward Euler integration**
- **Complete the lower-triangular matrix**
- **Complete calculate_jacobians.m**

$$\boldsymbol{B}_j^k = \left[\frac{\partial F_k}{\partial \boldsymbol{x}_k}\right]\left[\frac{\partial F_{k-1}}{\partial \boldsymbol{x}_{k-1}}\right]\cdots\left[\frac{\partial F_{j+1}}{\partial \boldsymbol{x}_{j+1}}\right]\left[\frac{\partial F_j}{\partial \boldsymbol{u}_j}\right] \quad \text{for } j = 1, 2, \ldots, k-2$$

$$\boldsymbol{B}_{k-1}^k = \left[\frac{\partial F_k}{\partial \boldsymbol{x}_k}\right]\left[\frac{\partial F_{k-1}}{\partial \boldsymbol{u}_{k-1}}\right]$$

$$\boldsymbol{B}_k^k = \frac{\partial F_k}{\partial \boldsymbol{u}_k}$$

$$d\boldsymbol{x}_{k+1} = \sum_{j=1}^{k} \boldsymbol{B}_j^k d\boldsymbol{u}_j$$

Hence, we have

$$\begin{bmatrix} d\boldsymbol{x}_2 \\ d\boldsymbol{x}_3 \\ \vdots \\ d\boldsymbol{x}_N \end{bmatrix} = \begin{bmatrix} \boldsymbol{B}_1^1 & 0 & \cdots & 0 \\ \boldsymbol{B}_1^2 & \boldsymbol{B}_2^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{B}_1^{N-1} & \boldsymbol{B}_2^{N-1} & \cdots & \boldsymbol{B}_{N-1}^{N-1} \end{bmatrix} \begin{bmatrix} d\boldsymbol{u}_1 \\ d\boldsymbol{u}_2 \\ \vdots \\ d\boldsymbol{u}_{N-1} \end{bmatrix}$$

This means that we can adjust the state variables by correcting the controls.

The computation of the sensitivity matrix $\boldsymbol{B}$ can be significantly simplified

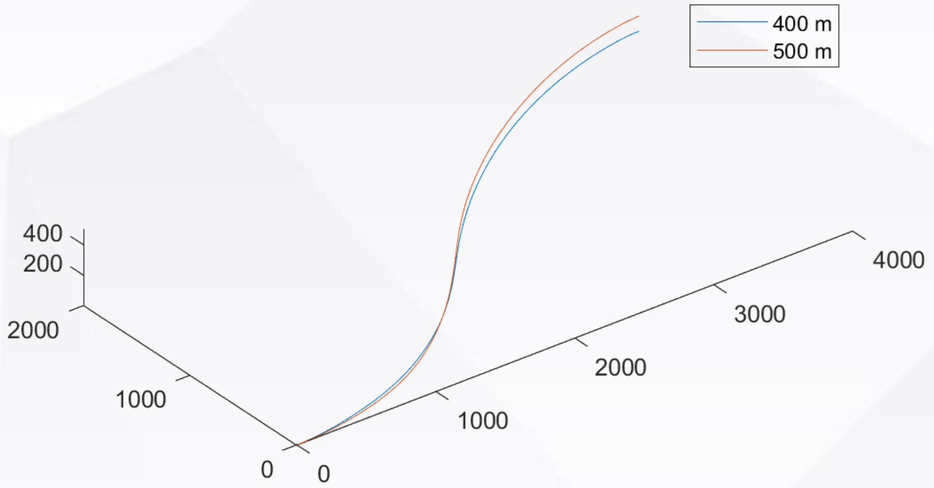
$$\left(\boldsymbol{B}_k^k\right)^0 = \boldsymbol{I}_n$$

$$\left(\boldsymbol{B}_j^k\right)^0 = \left(\boldsymbol{B}_{j+1}^k\right)^0 \left[\frac{\partial F_{j+1}}{\partial \boldsymbol{x}_{j+1}}\right], \, j = k-1, k-2, \ldots, 1$$

$$\boldsymbol{B}_j^k = \left(\boldsymbol{B}_j^k\right)^0 \left[\frac{\partial F_j}{\partial \boldsymbol{u}_j}\right], \qquad j = k, k-1, \ldots, 1\,.$$

Sequential Convex Optimization: Mission A

To climb 500 m instead of 400 m



Task part I:

- **Open main_fixed.m**
- **Read the code**
- **Complete the forward Euler integration**
- **Complete the lower-triangular matrix**

Task part II:

- **Open scp_solve_fixed.m**
- **Convert the box constraints**
- **Run the optimization**
- **Plot the velocity profile**

$$\boldsymbol{u}_{a\min} \leq \boldsymbol{u}_a \leq \boldsymbol{u}_{a\max}$$

$$\boldsymbol{u}_{a\min} - \boldsymbol{u}_a^p \leq d\boldsymbol{u}_a \leq \boldsymbol{u}_{a\max} - \boldsymbol{u}_a^p$$

Sequential Convex Optimization: Mission B

To climb 500 m instead of 400 m with Velocity constraint

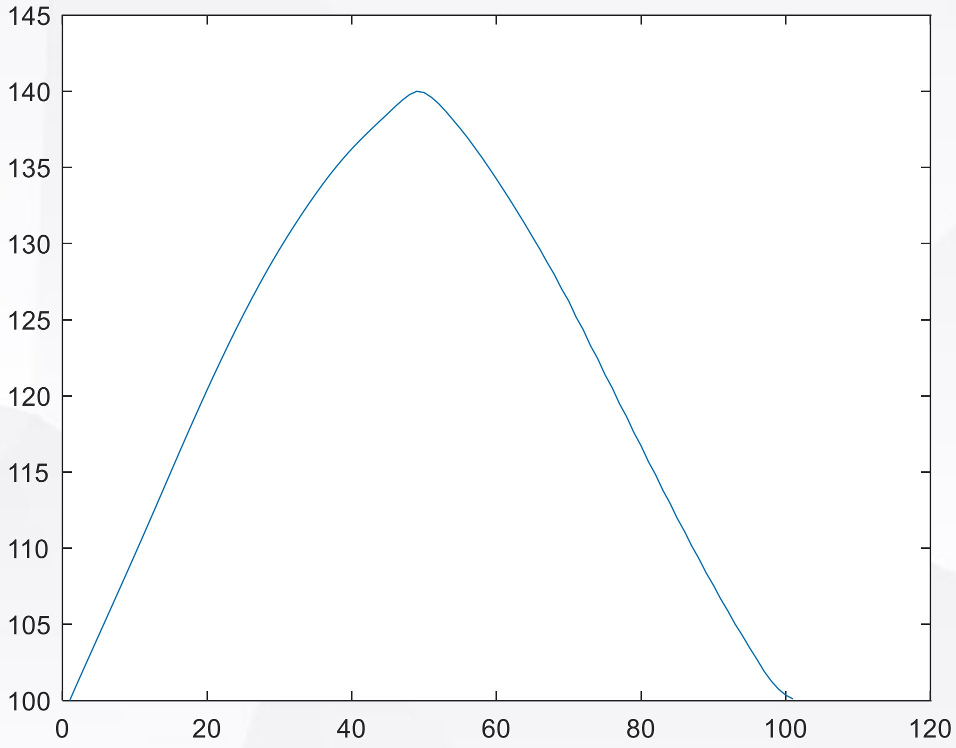
Task part I:

- **Uncomment the lines in main_fixed.m and scp_solve_fixed.m concerning V**
- **Run the optimization**
- **Plot the Velocity profile**

Task part II:

- **Set V max to 135 m/s**
- **Run the optimization**

Hint: Ctrl + C can stop the loop



Sequential Convex Optimization: Mission C

To climb 500 m instead of 400 m with Velocity constraint

NOW WE ARE WORKING ON **FREE FINAL TIME** SOLUTION

Task part I:

- **Open main_free.m**
- **Read the code**
- **Complete the forward Euler integration**
- **Complete the lower-triangular matrix**
- **Complete the new B matrix**

$$d\boldsymbol{x}_a = \boldsymbol{B} d\boldsymbol{u}_a$$

where

$$\boldsymbol{B} = \begin{bmatrix} \boldsymbol{B}_1^1 & 0 & 0 & \dots & 0 & \boldsymbol{C}_1^1 \\ \boldsymbol{B}_1^2 & \boldsymbol{B}_2^2 & 0 & \dots & 0 & \sum_{j=1}^{2} \boldsymbol{C}_j^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \boldsymbol{B}_1^{N-1} & \boldsymbol{B}_2^{N-1} & \boldsymbol{B}_3^{N-1} & \dots & \boldsymbol{B}_{N-1}^{N-1} & \sum_{j=1}^{N-1} \boldsymbol{C}_j^{N-1} \end{bmatrix} \in \mathbb{R}^{(N-1)n \times \{(N-1)m+1\}}$$

Task part II:

- **Complete scp_solve_free.m**
- **Complete update dt in main_free.m**
- **Run the optimization**

Expanding $d\boldsymbol{x}_{k+1}$ for $j = k,\ k-1, \ldots, 1$ gives

$$
\begin{aligned}
d\boldsymbol{x}_{k+1} &= \left[\frac{\partial F_k}{\partial \boldsymbol{x}_k}\right] d\boldsymbol{x}_k + \left[\frac{\partial F_k}{\partial \boldsymbol{u}_k}\right] d\boldsymbol{u}_k + \dot{\boldsymbol{x}}_k dh \\
&= \left[\frac{\partial F_k}{\partial \boldsymbol{x}_k}\right]\left[\frac{\partial F_{k-1}}{\partial \boldsymbol{x}_{k-1}}\right] d\boldsymbol{x}_{k-1} + \left[\frac{\partial F_k}{\partial \boldsymbol{x}_k}\right]\left[\frac{\partial F_{k-1}}{\partial \boldsymbol{u}_{k-1}}\right] d\boldsymbol{u}_{k-1} + \left[\frac{\partial F_k}{\partial \boldsymbol{x}_k}\right] \dot{\boldsymbol{x}}_{k-1} dh \\
&\quad + \left[\frac{\partial F_k}{\partial \boldsymbol{u}_k}\right] d\boldsymbol{u}_k + \dot{\boldsymbol{x}}_k dh \\
&\ \ \vdots \\
&= \boldsymbol{A}^k d\boldsymbol{x}_1 + \boldsymbol{B}_1^k d\boldsymbol{u}_1 + \boldsymbol{B}_2^k d\boldsymbol{u}_2 + \ldots + \boldsymbol{B}_k^k d\boldsymbol{u}_k + \left(\boldsymbol{C}_1^k + \boldsymbol{C}_2^k + \ldots + \boldsymbol{C}_k^k\right) dh
\end{aligned}
$$