Project: Minimum Time to Climb

Teerthaa Parakh

In this project, an optimization problem, minimum time to climb for an aircraft is solved. For this problem, the initial state and the final values of velocity, flight path angle, altitude are given. The goal is to find a sequence of angle of attack values such that the final state values are reached in minimum total time while satisfying the dynamics equations. To solve this problem, the system states are discretized using number of time-steps and the angle of attack using number of control points. Also in this trajectory optimization problem, gradient of the constraints is required, which is found using Adjoint method. The results from the adjoint method is cross-checked with Complex-Step method and the relative error is found to be in order of $1e-15$. Then the optimization problem is solved using Trust-Region Constrained Algorithm available in Python library SciPy. To get the trajectory from a given sequence of angle of attack, Newton's method is used. From optimization it is found that the amount of time taken to go from initial state to final is 131s. After this different number of time steps and number of control points are tested, and it is found that if the number of time-steps or number of control points are increased, the optimal solution remained the same and time taken by the algorithm to converge increases. If number of control points is increased, time taken to converge increases by a huge amount as compared to the case in which number of time-steps is increased. To make the model more realistic, atmospheric model is introduced in which the density is changing with altitude. In this case the amount of time taken to go from initial state to final is 124s, which is lower than the case in which atmospheric model is not considered.

I. Nomenclature

v	=	velocity
γ	=	flight path angle
h	=	altitude
r	=	distance flown
m	=	mass
T	=	Thrust
D	=	Drag
L	=	Lift
α	=	angle of attack
q	=	state vector, $[v, \gamma, h, r, m]^T$
N	=	number of time steps
M	=	number of control points
P	=	number of design variables
t_f	=	final time
g	=	9.81ms^{-2}

II. Problem Formulation

The dynamics equations for this system are :

$$\begin{aligned} \frac{dv}{dt} &= \frac{T}{m}\cos\alpha - \frac{D}{m} - g\sin\gamma \\ \frac{d\gamma}{dt} &= \frac{T}{mv}\sin\alpha + \frac{L}{mv} - \frac{g\cos\gamma}{v} \\ \frac{dh}{dt} &= v\sin\gamma \\ \frac{dr}{dt} &= v\cos\gamma \\ \frac{dm}{dt} &= \frac{-T}{gI_{sp}} \end{aligned} \tag{1}$$

First the problem is discritized in time. The time interval is divided between 0 and t_f into N time intervals. Therefore,

$$t_k = kt_f/N$$

The design variable vector consists of the angle of attack inputs at each point in time (x_α) and final time ($x_t = t_f$), where $\alpha(t) = N(t)^T x_\alpha$

$$x = [x_\alpha, x_{t_f}] = [x_{\alpha_0}, x_{\alpha_1}, x_{\alpha_2}, ..., x_{\alpha_{(M-1)}}, x_{t_f}]$$

Therefore length of design variables vector is $P = (M+1)$

The state variable vector consists of all the system variables at each point in time, excluding the initial conditions, where $q_k = q(t_k)$

$$q = [q_1, q_2, q_3, ..., q_N]$$

Therefore q is a $5 \times N$ matrix.

The objective to minimize is

$$f_{obj} = \min e^T x \tag{2}$$

where $e = [0, 0, 0, ...0, 1]^T$ is a unit vector of length $(M+1)$ with 1 at the end.

Subject to the **initial and final conditions**:

$$q_0 = [136, 0, 100, 0, 19030]^T \tag{3}$$

$$q_N = [340.28, 0, 20000, None, None]^T \tag{4}$$

Also, the **states $q(t)$ must satisfy dynamics equations** such that,

$$R(\dot{q}, q, x, t) = \begin{bmatrix} \frac{dv}{dt} - (\frac{T}{m}\cos\alpha - \frac{D}{m} - g\sin\gamma) \\ \frac{d\gamma}{dt} - (\frac{T}{mv}\sin\alpha + \frac{L}{mv} - \frac{g\cos\gamma}{v}) \\ \frac{dh}{dt} - v\sin\gamma \\ \frac{dr}{dt} - v\cos\gamma \\ \frac{dm}{dt} - (\frac{-T}{gI_{sp}}) \end{bmatrix} = 0 \tag{5}$$

Equation(2-5) is the constrained optimization problem that is solved in this report.

III. Calculating Gradients

The constrained optimization problem is solved using Adjoint Method. In this, adjoint is required for constraints. The equation(4) can be formulated as constraint. Therefore:

$$c(x,q) = \begin{bmatrix} q_N[0] - 340.28 \\ q_N[1] - 0 \\ q_N[2] - 20000 \end{bmatrix} = 0 \tag{6}$$

Since v, γ, h has different order of magnitude, so a scaling of $[100.0, 1.0, 1000.0]$ is used.

The adjoint equations are:

$$\begin{aligned} \frac{\partial R}{\partial q}^T \psi &= -\frac{\partial c}{\partial q}^T \\ \psi &= -\left(\frac{\partial R}{\partial q}^T\right)^{-1} \frac{\partial c}{\partial q}^T \end{aligned} \tag{7}$$

$$\frac{dc}{dx} = \frac{\partial c}{\partial x} + \psi^T \frac{\partial R}{\partial x} \tag{8}$$

The system states are interpolated linearly and control inputs are interpolated using Lagrange interpolation.

$$\begin{aligned} q &\approx \frac{q_k + q_{k-1}}{2} \\ \dot{q} &\approx \frac{q_k - q_{k-1}}{\Delta t} \\ x &\approx N(t_k)^T x_\alpha \end{aligned} \tag{9}$$

A. Adjoint Algorithm

$\frac{dc}{dx}$ is a $3 \times P$ matrix. $\frac{dc_i}{dx} = \frac{dc}{dx}[i,:]$, where $0 =< i <= 2$ for three constraints in equation(6). Each component of $\frac{dc}{dx}$ is found separately using the following method.

There are total N ψ's, and those are found by marching backwards from $k = N$ to $k = 1$.

$$\frac{\partial R_N}{\partial q_N}^T \psi_N = -\frac{\partial c_i}{\partial q_N}^T \tag{10}$$

for $k \in [1, N-1]$,

$$\frac{\partial R_k}{\partial q_k}^T \psi_k = -\frac{\partial c_i}{\partial q_k}^T - \frac{\partial R_{k+1}}{\partial q_{k+1}}^T \psi_{k+1} \tag{11}$$

ψ_k is a 5×1 vector for $k \in [1, N]$.

$\frac{\partial c_i}{\partial q_N}^T$ is a unit vector of shape 5×1 with 1 at index i.

$\frac{\partial c_i}{\partial q_k}^T$ is a zero vector for $k \in [1, N-1]$

$\frac{\partial R_k}{\partial q_k}^T$ is a 5×5 matrix for $k \in [1, N]$, such that

$$\frac{\partial R_k}{\partial q_k} = \frac{1}{2}\frac{\partial R}{\partial q} + \frac{1}{\Delta t}\frac{\partial R}{\partial \dot{q}} \tag{12}$$

The R.H.S of equation(12) is calculated using Complex Step method with step-size 10^{-30}.

Now,

$$\frac{dc_i}{dx} = \frac{\partial c_i}{\partial x} + \psi^T \frac{\partial R}{\partial x} \tag{13}$$

Or

$$\frac{dc_i}{dx} = \frac{\partial c_i}{\partial x} + \sum_{k=1}^{N} \psi_k^T \frac{\partial R_k}{\partial x} \tag{14}$$

$\frac{\partial c_i}{\partial x}$ is a zero vector.

Since $x = [x_\alpha, x_t]$, so $\frac{\partial R_k}{\partial x} = [\frac{\partial R_k}{\partial x_\alpha} \ \frac{\partial R_k}{\partial x_t}]$ and similarly $\frac{dc_i}{dx} = [\frac{dc_i}{dx_\alpha} \ \frac{dc_i}{dx_t}]$.

$\frac{\partial R_k}{\partial x_\alpha} = \frac{\partial R_k}{\partial \alpha_{k-0.5}} \frac{\alpha_{k-0.5}}{\partial x_\alpha}$ is a $5 \times M$ matrix because $\frac{\partial R_k}{\partial \alpha_{k-0.5}}$ is 5×1 vector and is calculated using Complex Step method with step-size 10^{-30} and $\frac{\alpha_{k-0.5}}{\partial x_\alpha}$ is $1 \times M$ vector

$\frac{\partial R_k}{\partial x_t} = \frac{\partial R_k}{\partial \dot{q}} \frac{\partial \dot{q}}{\partial \Delta t} \frac{\partial \Delta t}{\partial x_t}$ is a 5×1 vector because $\frac{\partial R_k}{\partial \dot{q}}$ is an identity matrix of size 5×5, $\frac{\partial \dot{q}}{\partial \Delta t}$ is a 5×1 vector with value of all the elements is $\frac{-1}{(\Delta t^2)}$ and $\frac{\partial \Delta t}{\partial x_t} = \frac{1}{N}$

B. Cross-checking the constraint jacobian with Complex-Step

Let p be some random direction, $x0$ is the design variable of length $1 \times P$ with a smooth distribution, h be a small complex-step perturbation $1e-30$ and tolerance of newton method is $1e-11$.
Then the complex-step is:

$$\frac{\text{imag } c(x0 + \text{j}hp)}{h} \tag{15}$$

The adjoint method is:

$$\nabla c(x0)^T p \tag{16}$$

The figure below shows the result obtained from adjoint method, and the complex-step method and the relative error for all the three constraints in equation(6).

```
print('Adjoint check')
print('{0:>25s} {1:>25s} {2:>25s}'.format('Adjoint', 'Complex step', 'Relative error'))
for adjoint, cs in zip(adjoint_result, cs_result):
    print('{0:25.15e} {1:25.15e} {2:25.15e}'.format(adjoint, cs, (adjoint - cs)/cs))
```

```
Adjoint check
                  Adjoint                Complex step            Relative error
-8.329833316912871e-04    -8.329833316912874e-04    -3.904767831131061e-16
 4.291314110929382e+01     4.291314110929380e+01     3.311539157436398e-16
 2.137909579696859e+01     2.137909579696867e+01    -3.655893667153745e-15
```

Fig. 1 Comparison of Adjoint method and Complex Step method

As we can see from Fig. 1, the relative error is atleast 1e − 15.

IV. Solving the Optimization Problem

In Scipy library of python, a gradient-based optimization Algorithm, Trust-Region Constrained is chosen to solve the problem. In Trust Region methods, given some point in the domain of objective function, a region is selected around that point and in that region the objective function is approximated to a simpler function for example quadratic function. Now, within that region an improvement point is to be found. Again a region is selected around this improved point and the above process repeats until convergence. With this method, there is always an upper bound on step-length.
In this, the gradient of the constraints are calculated using adjoint method, described in Section - III.A and Newton's

method is used to find the trajectory using a sequence of control points. Tolerance for newton method is $1e-11$ and for trust-region method it is $1e-6$, maxiter is fixed to 10000 because if the number of control point or number of time steps are increased, then the trust-region algorithm might require larger number of iterations to converge. Lower bound of -5 deg and and upper bound of 5 deg is put on angle of attack. Lower bound of 100s is put on final time t_f

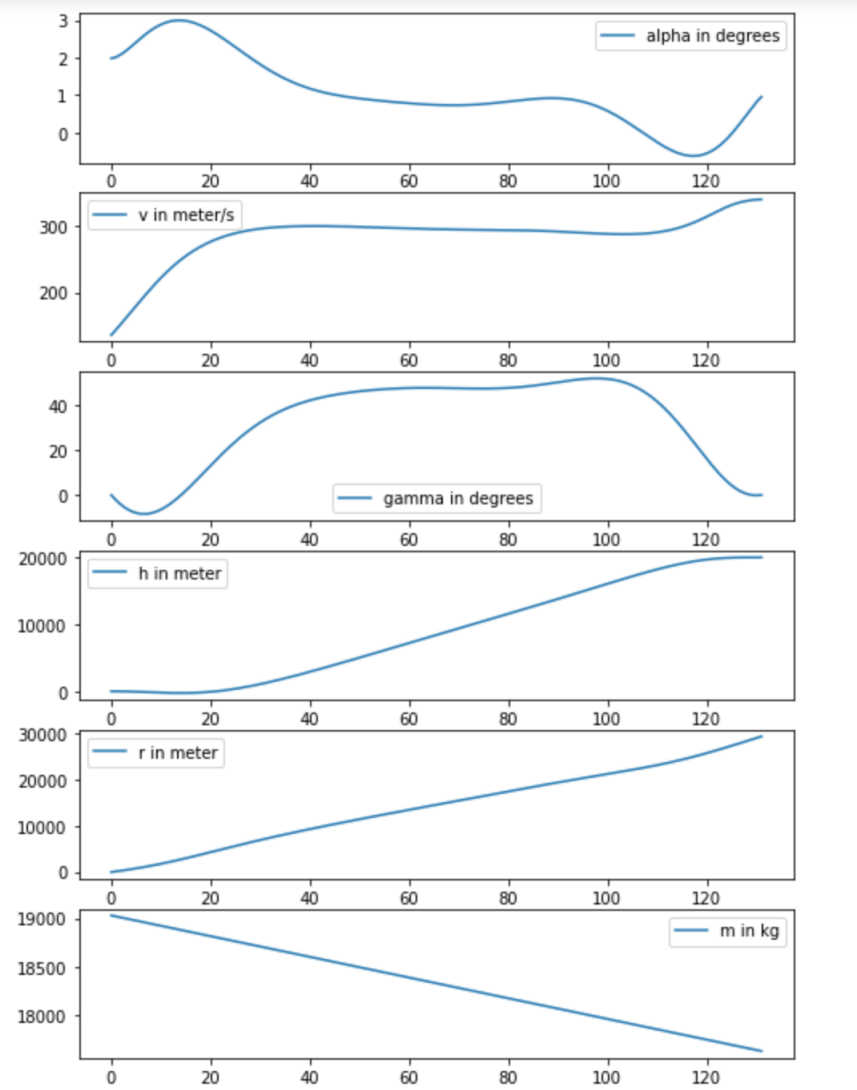

Fig. 2 state variables vs time plot

```
           message: '`gtol` termination condition is satisfied.'
            method: 'tr_interior_point'
              nfev: 675
              nhev: 0
                nit: 660
              niter: 660
              njev: 675
        optimality: 9.94329127045436e-07
            status: 1
           success: True
         tr_radius: 1.0
                 v: [array([-278.59276337,   -0.91905798,   -4.5175882 ]), array([ 1.77810657e-07,  7.84035058e-08,
8.02892284e-08,  5.27815111e-08,
        1.92354936e-08,  4.59004012e-09,  1.65504452e-08, -6.68695538e-09,
        1.11301255e-08,  1.90743368e-08, -8.24017785e-09])]
                 x: array([  1.98287801,   2.29163576,   2.97638205,   1.59645492,
         0.84961013,   0.7827099 ,   0.68237775,  -0.60399626,
         0.30523625,   0.95561811, 131.06729552])
```

Fig. 3 optimization algorithm output

Fig. 2 shows plots of v, γ, h, r and m vs time. From Fig. 3, the final time is $t_f = 131.07$s and trust-region method has taken 660 iterations to solve this problem. From Fig. 2, it can be seen that $-5 \leq \alpha \leq 5$, $v(t_f) = 340.29$m/s, $\gamma(t_f) = -7.31e-07$ deg and $h(t_f) = 1.99999999e+04 \approx 2.0000e+04$m. So, the constraints in equation(6) are satisfied.

A. Changing Number of Time Steps and Control Points

No. of Control points (M)	No. of Time Steps (N)	No. of iterations taken (niter)	No. of fun eval (nfev)	No. of Constr Gradient eval (cg_niter)	total time taken (in s) to converge (exec_t)	Result (final time in s) (tf)
10	**600**	635	654	1626	1900	131.077
10	**300**	660	675	1607	1244	131.06
10	**150**	675	711	1630	680	131.026
10	**50**	727	753	1687	200	130.55
20	**300**	1905	1953	4452	5900	131.00
20	**150**	2276	2302	5391	3475	130.966
5	**50**	245	238	300	55	132.68

Fig. 4 Comparison of optimal result for different Number of Time Steps and Control Points

From the table shown in Fig. 4, it can be concluded that, for M = 5 and N = 50, final time is 132.68 s, so due to poor dicretization in both control points and time steps, the solution obtained is higher than the optimal solution. For M = 10, as N is decreased from 600 to 50, niter and nfev are increasing slowly because the discretization has reduced so higher number of iterations are required for convergence. At the same time, exec time is decreasing very fast, this is because as mentioned in Section III.A ψ_k's are found by marching backwards from $k = N$ to $k = 1$ and also in newton's method the trajectory is found by marching forward from 1 to N, so as N is decreasing, the time required per iteration is also decreasing and hence the total execution time is also decreasing. If we keep the number of time-steps N to be 300 and increase M from 10 to 20, then there is drastic increase in niter and also in total execution time because the number of variables (whose values we want to determine) has increased.

V. Atmospheric Model

An atmospheric model [1] is used to make the system dynamics more accurate. In this atmospheric model, temperature and pressure varies with altitude, so the density also varies with altitude. Following is the atmospheric model:

$$T(\text{in K}) = \begin{cases} 141.94 + 0.00299h & \text{if } h > 25000\text{m} \\ 216.69 & \text{if } 11000\text{m} < h < 25000\text{m} \\ 288.19 - 0.00649h & \text{if } h < 11000\text{m} \end{cases}$$

$$p(\text{in kpa}) = \begin{cases} 2.488\left(\frac{T}{216.6}\right)^{-11.388} & \text{if } h > 25000\text{m} \\ 22.65e^{1.73-0.000157h} & \text{if } 11000\text{m} < h < 25000\text{m} \\ 101.29\left(\frac{T}{288.08}\right)^{5.256} & \text{if } h < 11000\text{m} \end{cases}$$

$$\rho = \frac{p}{0.2869T}\text{kgm}^{-3}$$

$$q_N = [295, 0, 20000, None, None]^T$$

After incorporating atmospheric model, if lower bound of -5 deg and and upper bound of 5 deg is put on angle of attack then the final time $t_f = 124.73$s, which is about 7s less than the final time obtained in non-atmospheric system model. Also if initial guess t_{guess} is taken to be 300s, then it is taking longer time to converge, so the t_{guess} is taken to be 135s, which is approximately the final time obtained from non-atmospheric dynamics model.

```
           message: '`gtol` termination condition is satisfied.'
            method: 'tr_interior_point'
              nfev: 724
              nhev: 0
               nit: 718
             niter: 718
              njev: 724
       optimality: 9.79898890178121e-07
            status: 1
           success: True
        tr_radius: 1664968.7926524323
                 v: [array([17.623878  ,  0.93411956, -5.8463113 ]), array([ 8.47920694e-08,  6.66724109e-08,  8.0705
9704e-08,  7.65997675e-08,
        5.81411090e-08,  1.48003884e-09, -5.11881280e-01, -1.07235722e+00,
       -9.93678214e-01, -1.66416593e-01, -1.03522501e-08])]
                 x: array([  0.27639187,   2.11584896,   2.90415375,   2.70576367,
        2.01243318,   0.30780792,  -4.99999912,  -4.99999987,
       -4.99999966,  -4.99999888, 124.72892789])
```

Fig. 5 optimization algorithm output after incorporating atmospheric model

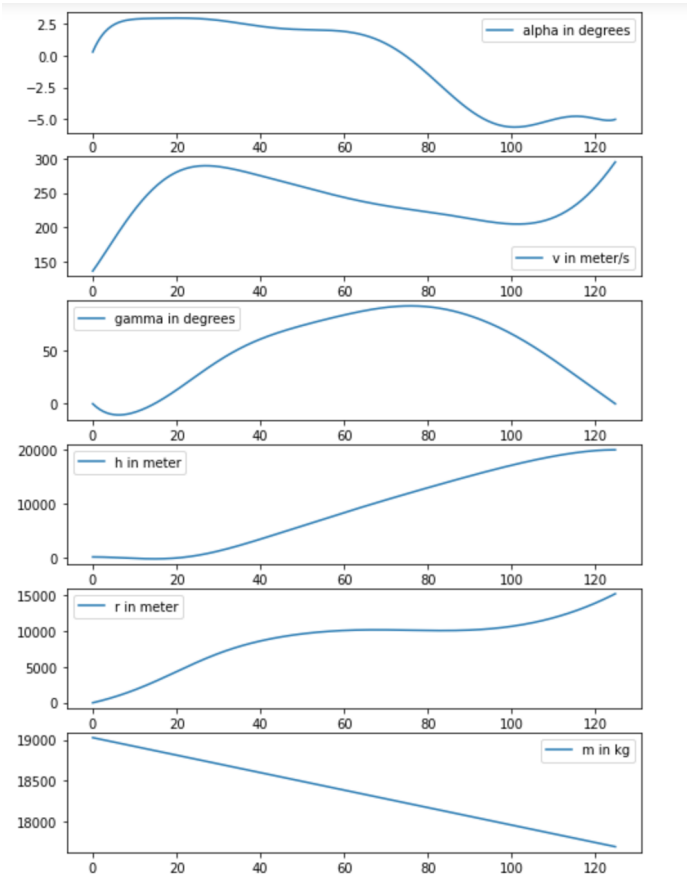

Fig. 6 State variables vs time plot after incorporating atmospheric model

As can be seen from Fig 5 and Fig 5, at the end of 124.73s, $v = 295\text{ms}^{-1}$, $\gamma = 0$ deg and $h = 20000$m. The angle of attack at the end reaches the lower bound of -5 deg.
If the lower bound on angle of attack is changed to -10 deg and upper bound to 10 deg, then the final time $t_f = 118.39$s. The angle of attack at the end reaches the lower bound of -10 deg.

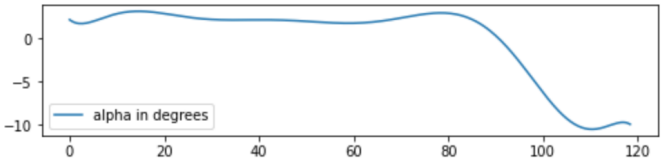

Fig. 7 If the lower bound is decreased to -10 deg, the total time t_f is 118.39s

References

[1] Grc.nasa.gov. 2022. Earth Atmosphere Model - Metric Units. [online] Available at: https://www.grc.nasa.gov/www/k-12/airplane/atmosmet.html [Accessed 28 April 2022].