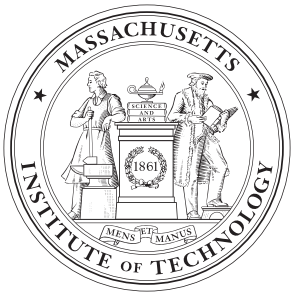
Optimal Dynamic Soaring of an Albatross

TERM PROJECT SPRING 2014
16.323 OPTIMAL CONTROL

Due: Tuesday, May 6, 2014



Author:
Daniel WIESE

Instructor:
Prof. Steven HALL

Abstract

This work presents an analysis and comparison of an optimal control solution for the dynamic soaring of an albatross with experimental data. Three different wind profiles were examined, and the problem of interest was maximizing the upwind progress an albatross could make during periodic, or traveling dynamic soaring. Representative geometrical and aerodynamic properties of a typical wandering albatross were gathered from the literature to use in the numerical simulation, where the solver GPOPS was used to obtain the optimal trajectory. This optimal solution when a linear, logarithmic, or power-law horizontal wind profile was used was compared to GPS data. The relevant data such as average groundspeed, peak altitude, and average cycle time were compared and agreed to well within one order of magnitude. It was determined that the albatross make forward progress into the wind with groundspeed approximately equal to the wind velocity. The trajectory verified the following dynamic soaring rule: climb into the wind, descend away from the wind. These solutions were found to be very sensitive to the wind gradient near the surface, as well as the lower altitude constraint limit.

1 List of Symbols

x	Position (east)
y	Position (north)
z	Altitude
V	Total airspeed
γ	Flight path angle
ψ	Heading angle (measured clockwise from north, or positive y-axis)
ϕ	Roll angle
p	Roll rate
m	Mass
ρ	Density
C_L	Lift coefficient
C_D	Drag coefficient
C_{D_0}	Parasitic drag coefficient
S	Wing area
E_{max}	Best lift-to-drag-ratio
L	Lift
D	Drag
k	Induced drag factor
W_x	Horizontal wind velocity

2 Introduction and Problem Statement

Soaring is defined as sustained flight which does not require external thrust. *Dynamic soaring* is a term used more specifically when the flight involves energy extraction from horizontal wind, as opposed to vertical wind components. It was first suggested by Lord Rayleigh in 1883 that birds may take advantage of dynamic soaring, and a substantial amount of research has been conducted since then.[1] We now understand that birds such as the wandering albatross have evolved to leverage this phenomenon, allowing them to make flights which last around 10 days, flying more than 1000 km per day. The information gained from learning about how the wandering albatross utilizes dynamic soaring could be used to develop bio-inspired UAVs

designed for long distance travel. Such UAVs could have applications in surveillance, search and rescue, and environmental monitoring, staying aloft for weeks at a time and covering thousands of miles with minimal energy consumption.[2]

2.1 The Wandering Albatross

The wandering Albatross is a large sea bird with 3.5 meter wing span, the largest of the animal kingdom. It has a special bone structure which allows it to "lock-in" its wings to the fixed, outstretched position without requiring the use of its muscles to flap its wings or hold them in place while flying. This means that the albatross maintains its flying configuration with essentially zero energy expenditure. It has an aspect ratio of about 16, and weighs between 7–10 kg. Some noteworthy figures are the range of the best lift-to-drag ratio of between 18–27[3–5] and maximum lift coefficient of 1.15–1.5.[6–8] The relevant quantities are listed in Table 1. This data was based on representative data[7–9] as compiled in Reference [10].

Figure 1: The wandering albatross. `http://www.stephenburch.com/`

Table 1: Albatross properties

Parameter	Symbol	Value	Unit
Mass	m	8.5	kg
Wing area	S	0.65	m^2
Wingspan	b	3.44	m
Parasitic drag coefficient	C_{D_0}	0.033	—
Maximum lift coefficient	$C_{L,\text{max}}$	1.5	—
Best lift-to-drag ratio	E_{max}	20	—

2.2 Problem Statement

The problem that was considered in this work is stated as follows: Given a typical boundary layer profile that might be observed over a flat field or calm sea, determine the optimal trajectory for an albatross to fly to maximize its distance travelled into the wind.

In Section 3 we provide the equations of motion which describe the albatross during dynamic soaring, and discuss the wind profile model used, the state constraints that must be satisfied while the albatross is soaring, and the assumptions that were made in deriving the model. In Section 4 we formulate the problem as an optimal control problem, and discuss how the solver GPOPS was used to numerically determine a solution.

In Section 5 we present the results of the numerical solution, and provide a discussion and interpretation of these results.

3 Equations of Motion

The anatomy of the albatross and its ability to keep its wings stretched makes it behave essentially like a fixed-wing glider, with some differences which will be discussed later. The equations of motion describing the albatross are the following[11]

$$\begin{aligned}
\dot{x} &= V\cos\gamma\sin\psi + W_x \\
\dot{y} &= V\cos\gamma\cos\psi \\
\dot{z} &= V\sin\gamma \\
\dot{V} &= -\frac{1}{m}D - g\sin\gamma - \dot{W}_x\cos\gamma\sin\psi \\
\dot{\gamma} &= \frac{\cos\phi}{mV}L - \frac{1}{V}g\cos\gamma + \frac{1}{V}\dot{W}_x\sin\gamma\sin\psi \\
\dot{\psi} &= \frac{\sin\phi}{mV\cos\gamma}L - \frac{\cos\psi}{V\cos\gamma}\dot{W}_x \\
\dot{\phi} &= p \\
\dot{C}_L &= -bC_L + bC_{L,\text{cmd}}
\end{aligned} \tag{1}$$

where x is the position corresponding to east, y is north, V is the airspeed, γ is the flight path angle, ψ is the heading angle, ϕ is the roll angle, and p is the roll rate. L and D are the lift and drag, and $W_x(z)$ and $\dot{W}_x(z)$ define the wind profile, and time rate of change in the wind experienced by the albatross during flight. The lift and drag are given by the following well-known equations

$$L = \frac{1}{2}\rho V^2 S C_L \tag{2}$$

$$D = \frac{1}{2}\rho V^2 S C_D \tag{3}$$

where ρ is the density of air, S is the planform area, and C_L and C_D are the lift and drag coefficients, respectively. The total drag coefficient is calculated as follows

$$C_D = C_{D_0} + kC_L^2 \tag{4}$$

where C_{D_0} is the parasitic drag coefficient, and k is the induced drag factor. The induced drag factor can be calculated as

$$k = \frac{1}{4E_{\text{max}}^2 C_{D_0}} \tag{5}$$

where E_{max} is the maximum lift-to-drag ratio, given as

$$E_{\text{max}} = \left(\frac{C_L}{C_D}\right)_{\text{max}} = \left(\frac{C_L}{C_{D_0} + kC_L^2}\right)_{\text{max}} \tag{6}$$

The inputs to this system of equations is the roll rate p and the commanded lift coefficient $C_{L,\text{cmd}}$. The lift coefficient was included as a state in (1) to capture the dynamics which govern the ability of the albatross to change its lift during flight. During dynamic soaring, it can be advantageous to abruptly change the lift generated, but with outstretched wings there is a finite rate at which the albatross can do this. By effectively filtering the lift coefficient as an input it required changes in the albatrosses lift coefficient to be smooth, as would be expected during flight. For the same reason the roll rate was used as an input instead of the roll angle.

3.1 State-Space Formulation

The system of equations in (1) can be represented as

$$\dot{X} = f(X, U)$$

where the state vector X and input U are given by

$$\begin{aligned} X &= \begin{bmatrix} x & y & z & V & \gamma & \psi & \phi & C_L \end{bmatrix} \\ U &= \begin{bmatrix} C_{L,\text{cmd}} & p \end{bmatrix} \end{aligned}$$

This is the representation of the dynamic constraint which will be inputed into the solver to obtain the optimal trajectory as discussed in the following section.

3.2 Wind Profile

The essence of dynamic soaring is the extraction of energy from a horizontal wind-shear gradient. Such a gradient is necessary for dynamic soaring, and so the particulars of the wind profile are of great importance. This gradient is strong within an altitude of about 20 meters above the surface, after which the wind speed remains essentially constant. The wind profile is accurately modeled using the following logarithmic function[7,8,12]

$$\textbf{Logarithmic} \qquad W_x(z) = W_{x,\text{ref}} \frac{\ln(z/z_0)}{\ln(z_{\text{ref}}/z_0)}$$

with derivative

$$\frac{dW_x}{dz} = \frac{W_x}{\ln(z_{\text{ref}}/z)} \frac{1}{z + z_0}$$

A more convenient and less accurate of the wind profile is to use a linear model, given by

$$\textbf{Linear} \qquad W_x(z) = \frac{W_{x,\text{ref}}}{z_{\text{ref}}} z$$

with derivative

$$\frac{dW_x}{dz} = \frac{W_{x,\text{ref}}}{z_{\text{ref}}}$$

This linear model is less representative of the actual wind profile, but can be more convenient when evaluating the numerical solution. Another wind model that is suggested is the power-law wind profile,[3] given below. This model is often preferred over the linear profile because it is a more accurate representation of the

actual wind profile. It is often preferred over the logarithmic profile due to the fact that the logarithmic profile has an infinite gradient at the surface, and the wind velocity never actually diminishes with an increase in altitude, whereas the following power-law profile has a finite gradient at the surface and diminishes

$$\textbf{Power-law} \qquad W_x(z) = \frac{W_{x,\text{ref}}}{1-e^{-a}}\left(1-e^{-a\frac{z}{z_{\text{ref}}}}\right)$$

with derivative

$$\frac{dW_x}{dz} = \left(\frac{a}{z_{\text{ref}}}\right)\left(\frac{W_{x,\text{ref}}}{1-e^{-a}}\right)e^{-a\frac{z}{z_{\text{ref}}}}$$

The time derivative $\dot{W}_x$ is evaluated using the chain rule as

$$\dot{W}_x = \frac{dW_x}{dt} = \frac{dW_x}{dz}\frac{dz}{dt} = \frac{dW_x}{dz}V\sin\gamma$$

Each of these three wind profiles is plotted in Figure 2 below.

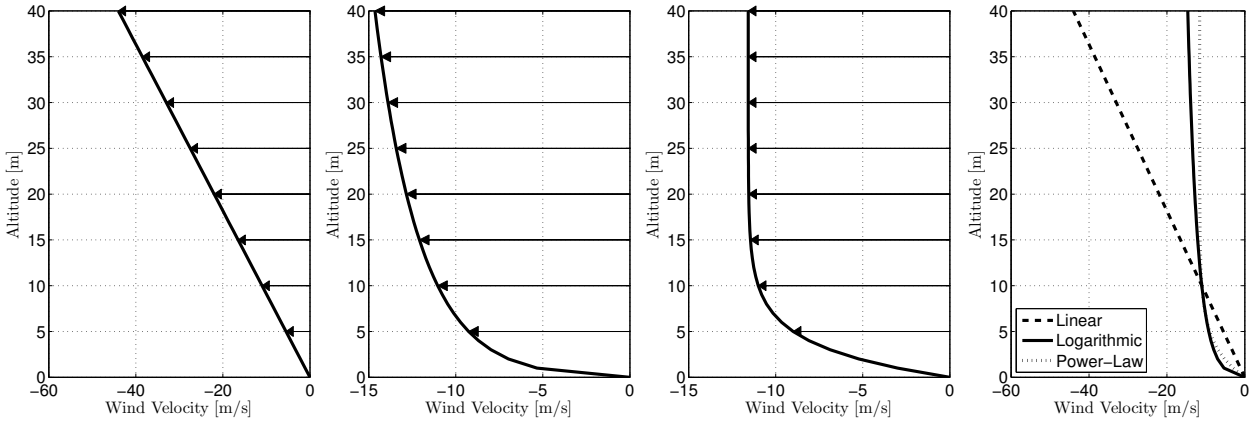


Figure 2: Plot of the wind over the surface for linear, logarithmic, and power-law profiles, from left to right. The fourth figure shows a comparison between each velocity profile.

3.3 State Constraints

In nearly any optimal control problem, there will exist constraints on the state, both at the initial and terminal times, as well as along the trajectory. To obtain an optimal control solution which resembles experimental data, it is important to accurately capture what the state constraints are that an albatross must satisfy during soaring flight.

During dynamic soaring, that is without flapping its wings, $v_{\text{min}} = 12$ m/s, is the minimum airspeed that the bird requires to maintain lift.[13,14] There is not much of a practical upper limit on airspeed[15] but observed data suggests $v_{\text{max}} = 47$ m/s is a reasonable upper limit. Observed bank angles are estimated to be in the range 55–70 degrees.[16,17] No data could be found on observed flight path angle limits for albatrosses. Because the wind gradient provides the energy for dynamic soaring, and because the wind will have a gradient which is largest at the surface, the minimum altitude at which the albatross flies greatly impacts its ability to use dynamic soaring. Observations have shown that it is not uncommon for the wingtips of the albatross to skim

the water when soaring. A lower limit of 1 meter was selected as the minimum altitude. A peak altitude of between 8–30 meters has been observed during typical soaring. 40 meters was selected as the upper limit.

4 Optimal Control Problem Formulation

In terms of an optimal control problem formulation, the problem statement in Section 2.2 is stated as: determine the optimal control inputs $C_{L,\text{cmd}}$ and p to minimize the following objective function, subject to the state dynamics, initial, terminal and path constraints. The objective function is given by

$$\min_{C_{L,\text{cmd}},p} J = -x(t_f)$$

indicating the desired objective is to maximize the distance travelled upwind. The boundary conditions are given as

$$\begin{aligned} x(t_0) &= 0 \\ y(t_0) &= 0 \\ z(t_0) &= 10 \\ V(t_0) &= 24 \end{aligned}$$

with the following conditions used to enforce periodicity of the flight path for what is called *traveling dynamic soaring*. That is, the albatross should complete a dynamic soaring maneuver repetitively to make forward progress into the wind.

$$\begin{aligned} z(t_0) &= z(t_f) \\ V(t_0) &= V(t_f) \\ \gamma(t_0) &= \gamma(t_f) \\ \psi(t_0) &= \psi(t_f) \\ \phi(t_0) &= \phi(t_f) \\ C_L(t_0) &= C_L(t_f) \end{aligned}$$

The path constraints are given by the following, where these values are representative of the observed data discussed in Section 3.3.

$$\begin{aligned} 1 \leq & z \\ 12 \leq & V \leq 47 \\ -75 \leq & \gamma \leq 75 \\ -80 \leq & \phi \leq 80 \\ -1.5 \leq & C_L \leq 1.5 \end{aligned}$$

where the altitude is expressed in [m], the velocity is expressed in [m/s], and the angles expressed in [deg]. In determining the optimal solution, the following numerical values were used for the wind profile.

Table 2: Environment properties

Parameter	Symbol	Value	Unit
Air density	ρ	1.225	kg/m^3
Reference windspeed	$W_{x,\text{max}}$	-11	m/s
Reference altitude	z_{ref}	10	m
Logarithmic wind parameter	z_0	0.15	m
Power-law wind parameter	a	3	—

4.1 Solution Method

After the optimal control problem was formulated, the solver GPOPS was used to numerically determine a solution to the dynamic soaring problem for the three different wind profiles described above. GPOPS uses collocation, which approximates trajectories using Lagrange interpolating polynomials to essentially represent the problem as a constrained function optimization problem, or nonlinear program. This is known as a transcription method. Once the problem has been cast as a nonlinear program, GPOPS has different NLP solvers which can be used.

5 Results and Discussion

Figures 3-5 show the trajectory of the albatross as it attempts to maximize its distance traveled in the positive x-direction against the wind for the three different wind profiles. The albatross starts the flight at the green circle, ending at the red x. Analysis of these profiles verifies the following rule, known as the dynamic soaring rule. Simply stated, this rule says to maximize dynamic soaring energy the albatross should climb into the wind and descend away from the wind.

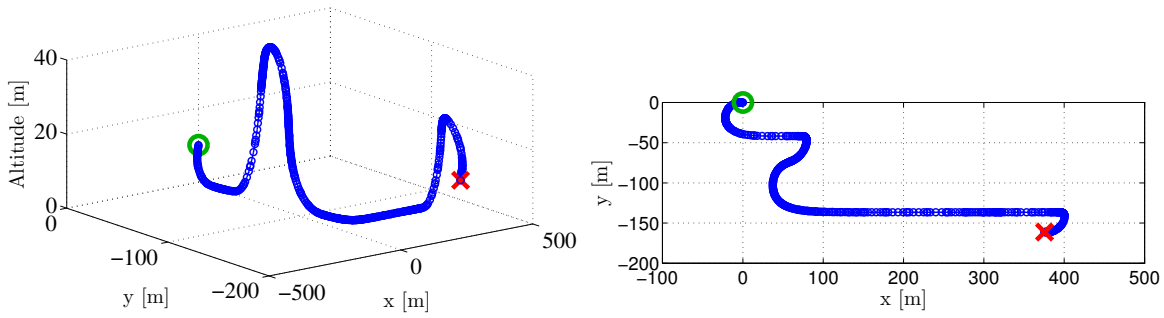


Figure 3: Optimal dynamic soaring trajectory of the albatross in the linear wind profile with $W_{x,\text{ref}} = -11$ m/s and $z_{\text{ref}} = 10$ m.

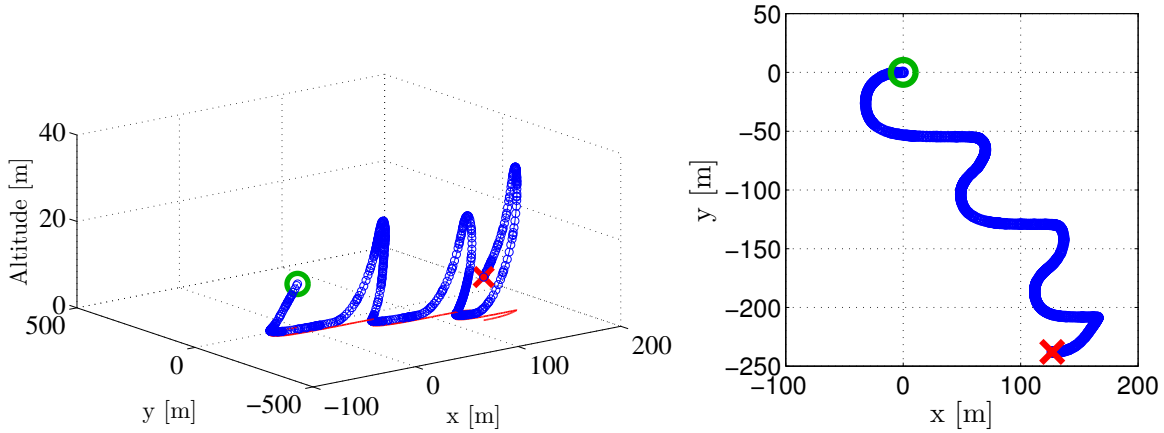


Figure 4: Optimal dynamic soaring trajectory of the albatross in the logarithmic wind profile with $W_{x,\text{ref}} = -11$ m/s, $z_{\text{ref}} = 10$ m, and $z_0 = 0.15$ m.

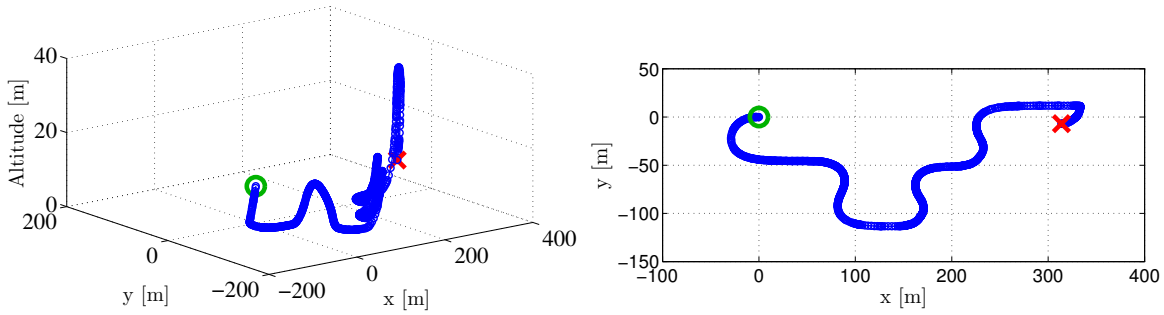


Figure 5: Optimal dynamic soaring trajectory of the albatross in the power-law wind profile with $W_{x,\text{ref}} = -11$ m/s, $z_{\text{ref}} = 10$ m, and $a = 3$.

5.1 Comparison to Experimental Data

The results of the dynamic soaring optimization described above were compared to some recently obtained data obtained for the flight of albatrosses equipped with a GPS sensor to monitor their flight path. It has been reported in many observations that the typical dynamic soaring cycle is around 10 seconds, with peak altitudes of around 15 m being fairly common during moderate 10 m/s winds when the albatross is flying upwind. Furthermore, upwind progress was observed to be between 4.6–7.6 m/s, depending on the strength of the wind. These observations were verified by some recently obtained GPS data which captured several days worth of dynamic soaring data of an albatross.[18,19] This GPS data is consistent with the results obtained in the optimization, well within one order of magnitude of all the relevant quantities such as average groundspeed, peak altitude, and cycle time. The relevant data for the optimization is collected in Table 3 below.

Table 3: Optimization results for different wind profiles when flying upwind

Wind Profile	Average groundspeed [m/s]	Average Airspeed [m/s]	Peak altitude [m]	Cycle time [s]
Linear	12.5	25.7	40.0	15.0
Logarithmic	5.4	24.3	29.5	7.5
Power-Law	10.5	26.0	34.0	7.5
GPS data	6.1	12.7	9.8	7.5

It was found that the solution was highly dependent on the wind profile and the lower altitude constraint limit. This is expected, as the velocity gradient is the mechanism through which the albatross extracts energy from the wind, and the gradient becomes increasingly large very near the surface. Changing the wind profile parameters a and z_0 very slightly resulted in relatively large differences in the solution. Changing the lower altitude limit an order of 0.1 meters also had a noticeable impact on the solution. Furthermore, the solution was sensitive to the constraints on roll angle and roll rate as well. The more aggressively the albatross maneuvers, the more efficiently it can use dynamic soaring. However, while gliders can be made to sustain g-loads of 30 or more, the albatross cannot tolerate more than about 3. While the results overall matched well, it is noted that the power-law and logarithmic profiles are still a significant simplification of the profile that is experienced by an albatross flying over the ocean. Waves greatly change the wind profile near the surface, causing eddies between wave crests, and updrafts of around 2 m/s.

6 Conclusions and Future Work

The optimal dynamic soaring trajectory for an albatross which maximizes distance traveled into the wind was determined using optimal control. Using representative data describing the mass, geometry, and aerodynamic characteristics of an albatross, as well as a simplified wind model, the optimal trajectory is fairly close to GPS data taken from real albatrosses while performing dynamic soaring. However, there are several major limitations that must be considered when comparing the optimal control solution to GPS data. The first limitation is that the simplified wind profile model neglects wind eddies between wave crests, and the significant updrafts which are often present on the ocean. Second, the GPS data which is recorded does not provide information about even the average wind during the flight of an albatross. Lastly, the optimal control solution is very sensitive to small changes in the wind profile and the lower constraint limit on altitude, and so solutions can differ noticeably for small changes in these values. This can make it hard to obtain an accurate optimal control solution and fairly compare it with the GPS data. However, even with these limitations the optimal control solution for all three wind profiles is still a decent representation of the soaring observed in albatrosses.

In order to improve the optimal control solution and more accurately compare it with experimental data, it would be beneficial to improve the wind model. To do this, more information should be compiled to determine a good representation of the average waves and wind profile on the open ocean on an average day. Then, more GPS data should be collected with additional information gathered about the local wind speed near the albatross as it flies.

References

[1] Rayleigh, L., "The soaring of birds," *Nature*, Vol. 27, No. 701, 1883, pp. 534–535.

[2] Richardson, P. L., "High-speed robotic albatross: unmanned aerial vehicle powered by dynamic soaring," 2012.

[3] Barnes, J. P., "How Flies the Albatross–the Flight Mechanics of Dynamic Soaring," *Flight Dynamics*, Vol. 2013, pp. 04–03.

[4] Alexander, D. and Vogel, S., *Nature's Flyers: Birds, Insects, and the Biomechanics of Flight*, Nature's Flyers, Johns Hopkins University Press, 2004.

[5] Denny, M., "Dynamic soaring: aerodynamics for albatrosses," *European Journal of Physics*, Vol. 30, No. 1, 2009, pp. 75.

[6] Pennycuick, C., "Gliding flight of the fulmar petrel," *Journal of experimental Biology*, Vol. 37, No. 2, 1960, pp. 330–338.

[7] Wood, C., "The flight of albatrosses (a computer simulation)," *Ibis*, Vol. 115, No. 2, 1973, pp. 244–256.

[8] Pennycuick, C., "The flight of petrels and albatrosses (Procellariiformes), observed in South Georgia and its vicinity," *Philosophical Transactions of the Royal Society of London. B, Biological Sciences*, Vol. 300, No. 1098, 1982, pp. 75–106.

[9] Tucker, V. A. and Parrott, G. C., "Aerodynamics of gliding flight in a falcon and other birds," *Journal of Experimental Biology*, Vol. 52, No. 2, 1970, pp. 345–367.

[10] Sachs, G., "Minimum shear wind strength required for dynamic soaring of albatrosses," *Ibis*, Vol. 147, No. 1, 2005, pp. 1–10.

[11] Zhao, Y. J., "Optimal patterns of glider dynamic soaring," *Optimal Control Applications and Methods*, Vol. 25, No. 2, 2004, pp. 67–89.

[12] Cone, C. D., *A Mathematical Analysis of the Dynamic Soaring Flight of the Albatross: With Ecological Interpretations*, Virginia Institute of Marine Science Gloucester Point, Virgina, 1964.

[13] Wilson, J., "Sweeping flight and soaring by albatrosses," *Nature*, Vol. 257, 1975, pp. 307–308.

[14] Videler, J. J., *Avian flight*, Oxford University Press, 2005.

[15] Catry, P., Phillips, R. A., Croxall, J. P., and Burger, A., "Sustained fast travel by a gray-headed albatross (Thalassarche chrysostoma) riding an Antarctic storm," *The Auk*, Vol. 121, No. 4, 2004, pp. 1208–1213.

[16] Idrac, P. and Georgii, W., *Experimentelle Untersuchungen über den Segelflug, mitten im Fluggebiet grosser Segelnder Vögel (Geier, Albatros usw). Ihre Anwendung auf den Segelflug des Menschen...[Einleitung von Walter Georgii.]*., R. Oldenbourg, 1932.

[17] Pennycuick, C. J., "Gust soaring as a basis for the flight of petrels and albatrosses (Procellariiformes)," *Avian Science*, Vol. 2, No. 1, 2002, pp. 1–12.

[18] Sachs, G., Traugott, J., and Holzapfel, F., "Progress against the wind with dynamic soaring: results from in-flight measurements of albatrosses," *AIAA guidance, navigation, and control conference and exhibit, Portland, Oregon, USA*, 2011.

[19] Sachs, G., Traugott, J., Nesterova, A., and Bonadonna, F., "Experimental verification of dynamic soaring in albatrosses," *The Journal of experimental biology*, Vol. 216, No. 22, 2013, pp. 4222–4232.

7 Appendix: Optimal Trajectory Plots

7.1 Linear Wind Profile

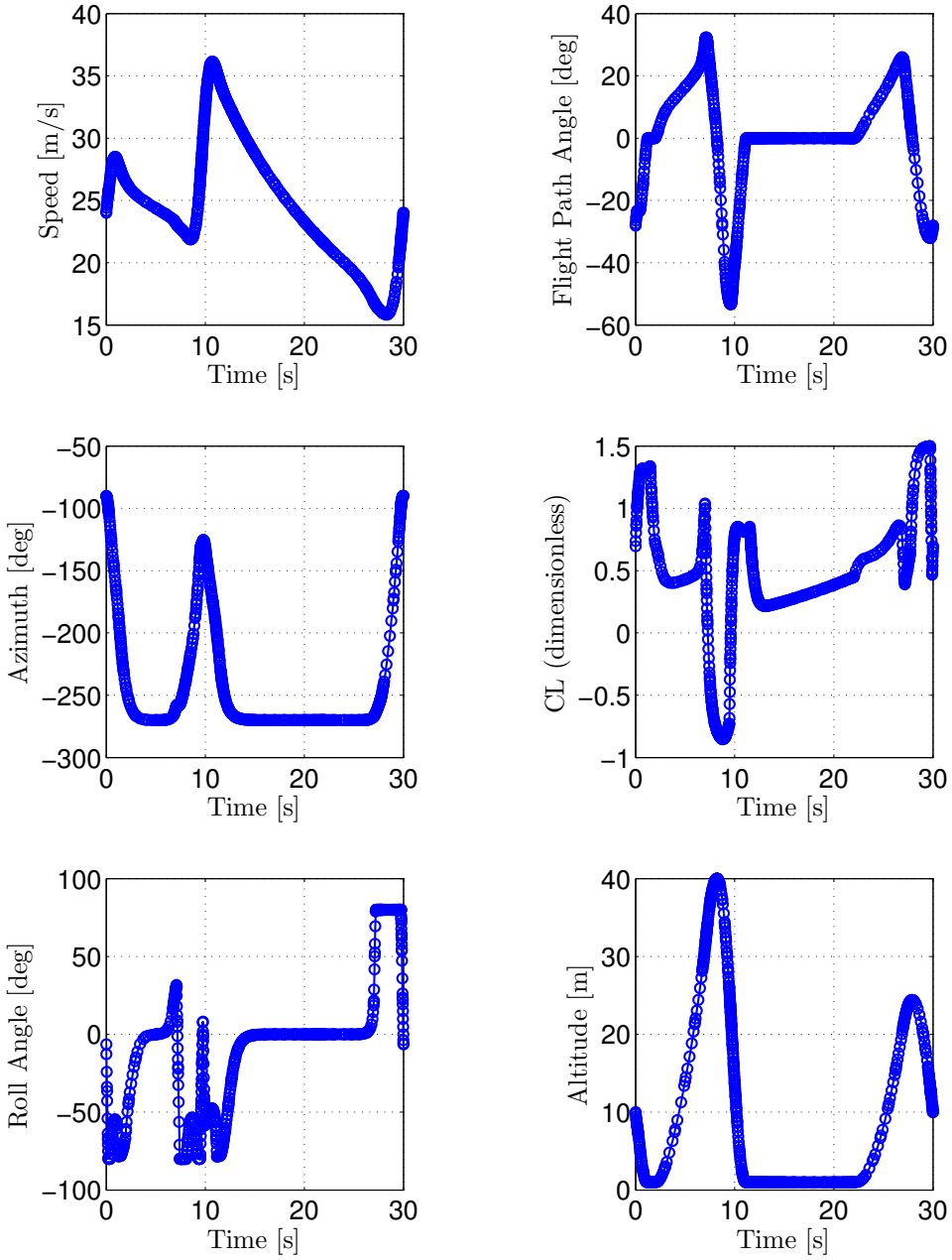


Figure 6: optimal dynamic soaring trajectory

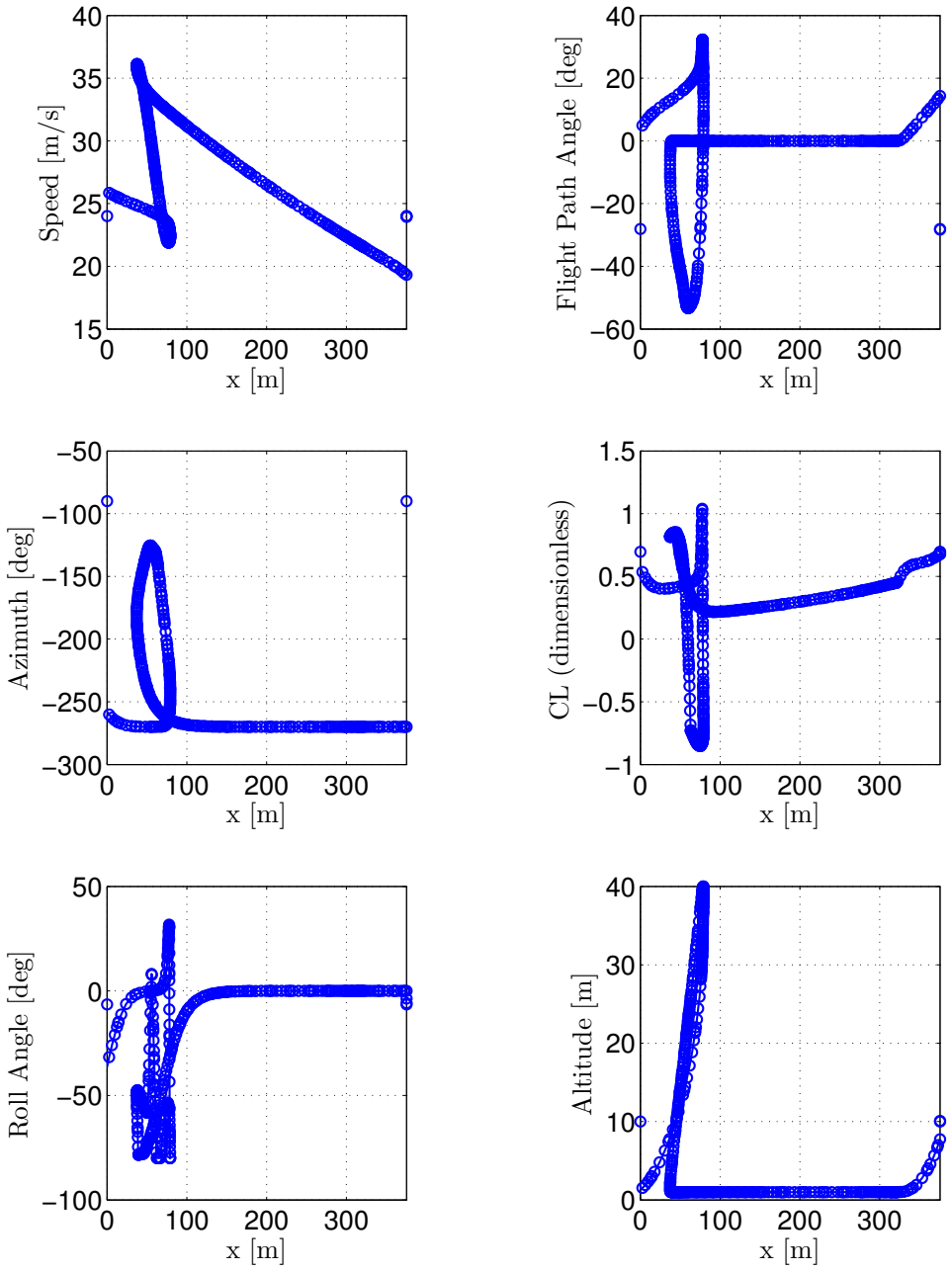


Figure 7: optimal dynamic soaring trajectory

7.2 Logarithmic Wind Profile

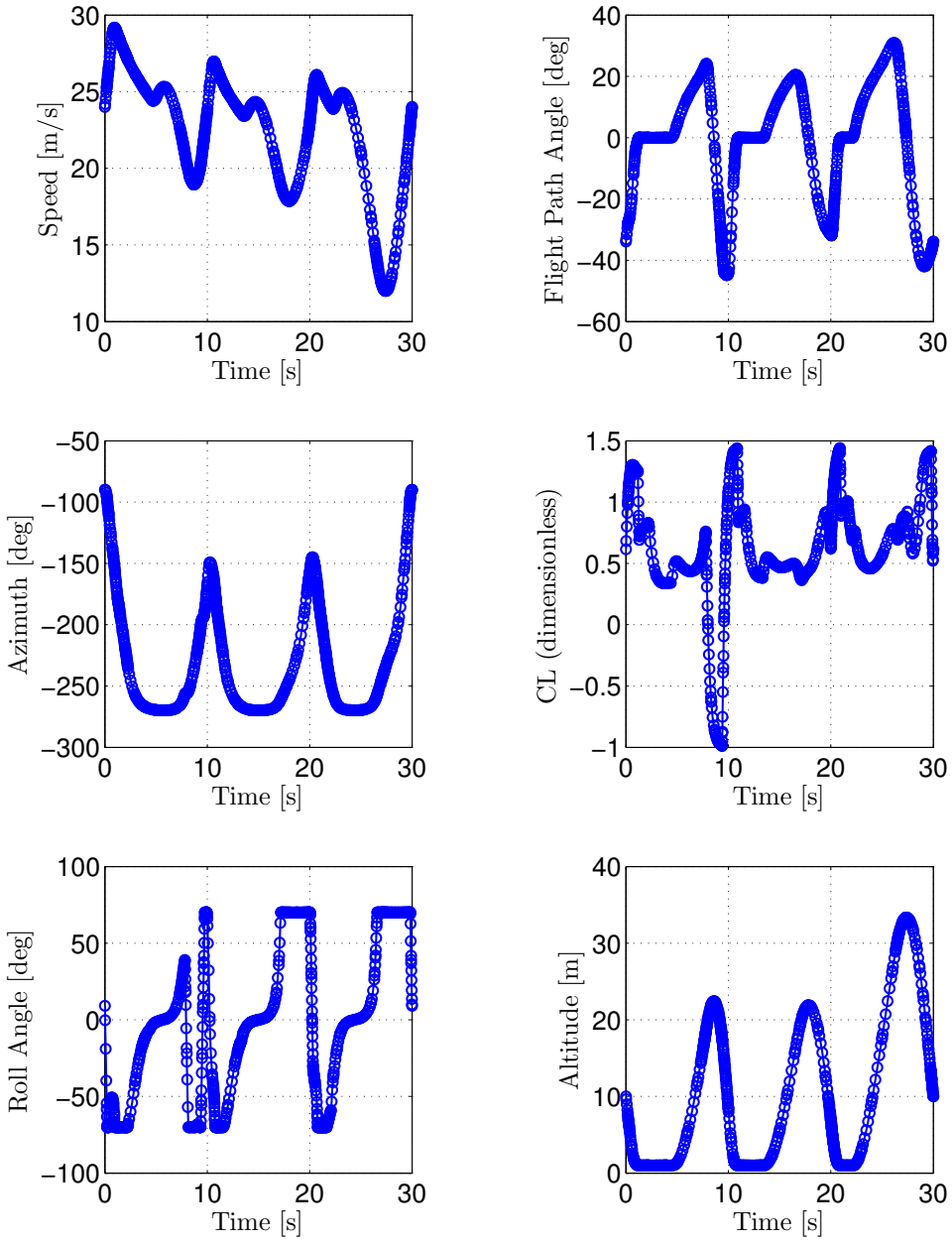


Figure 8: optimal dynamic soaring trajectory

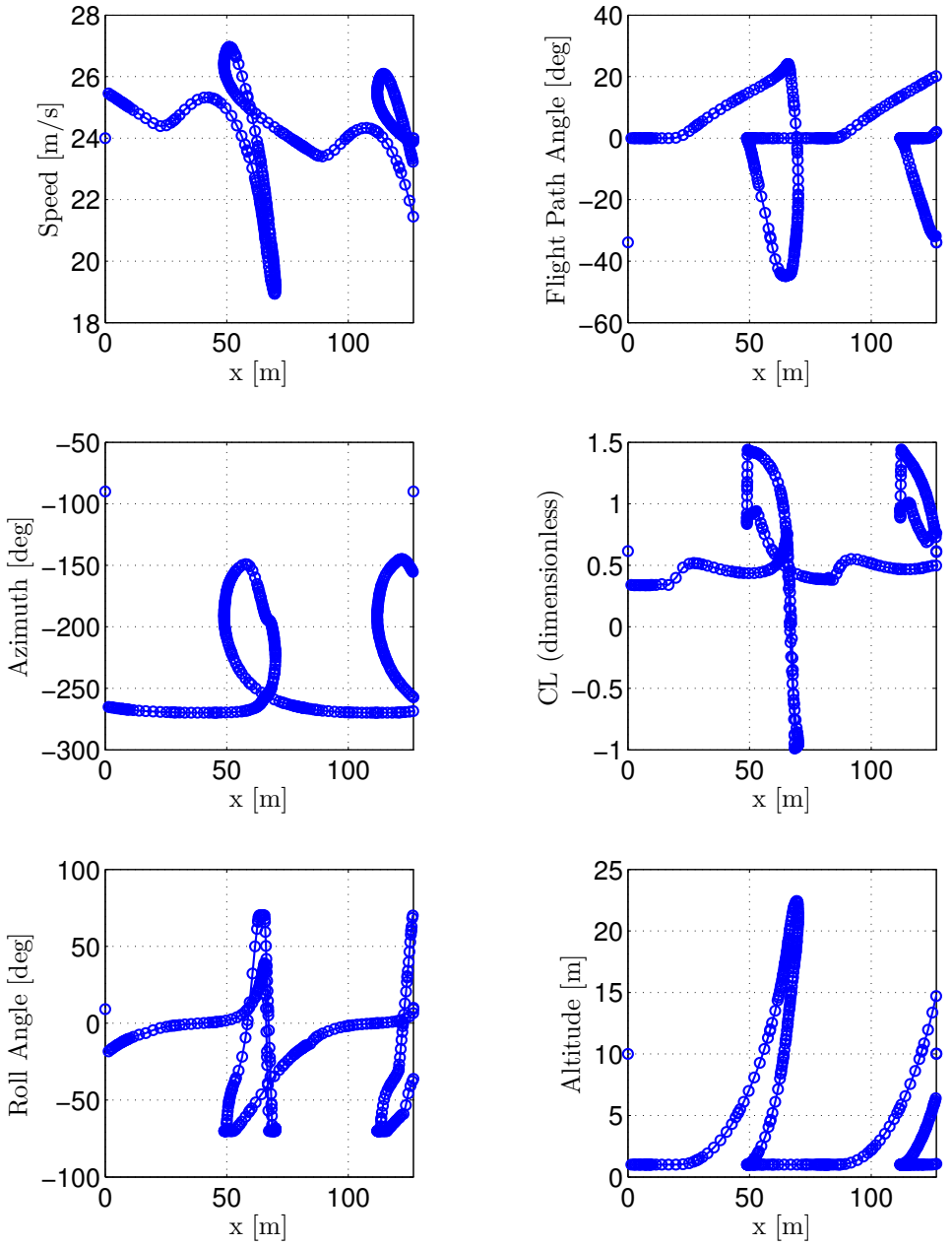


Figure 9: optimal dynamic soaring trajectory

7.3 Power-Law Wind Profile

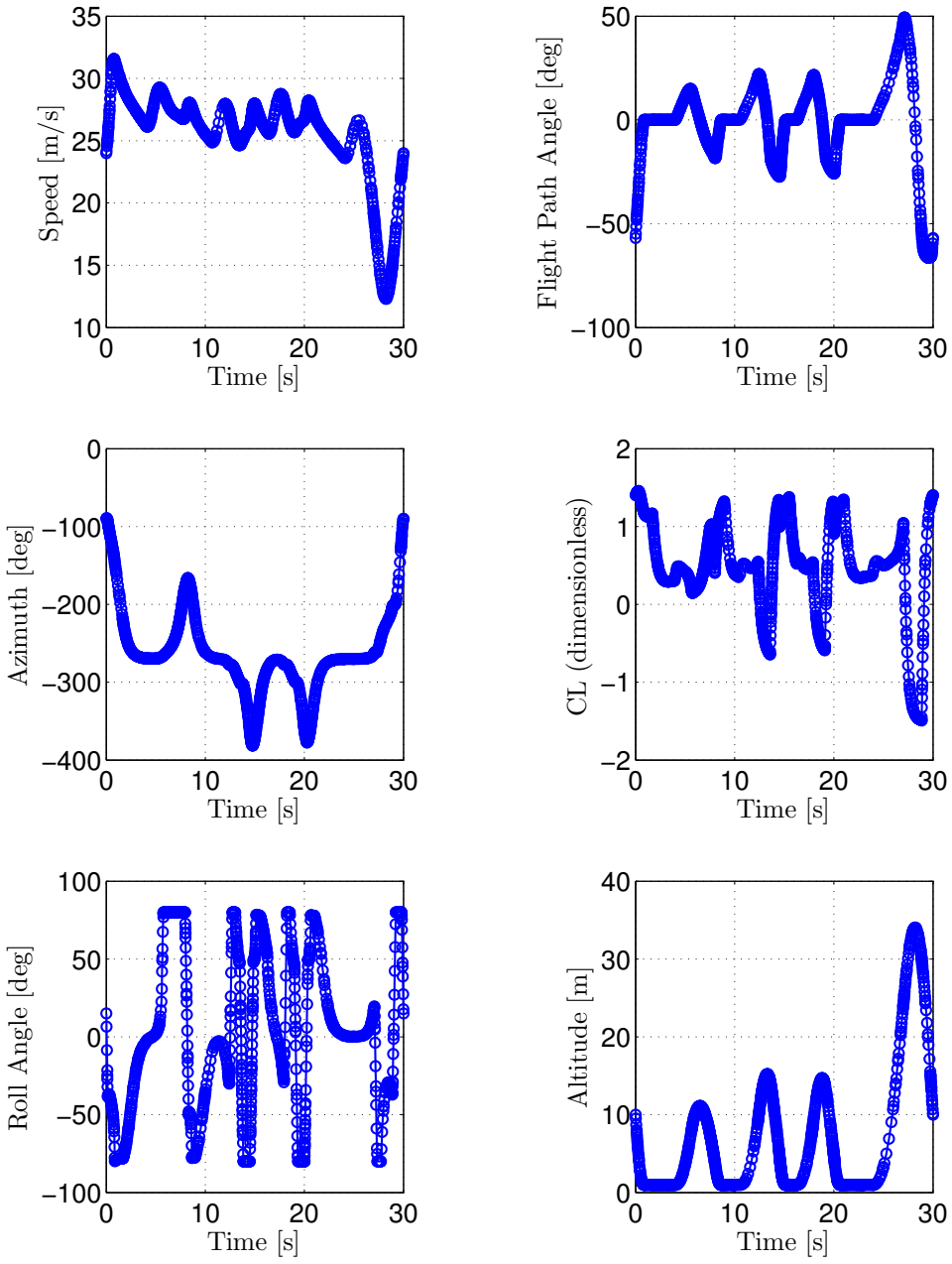


Figure 10: optimal dynamic soaring trajectory

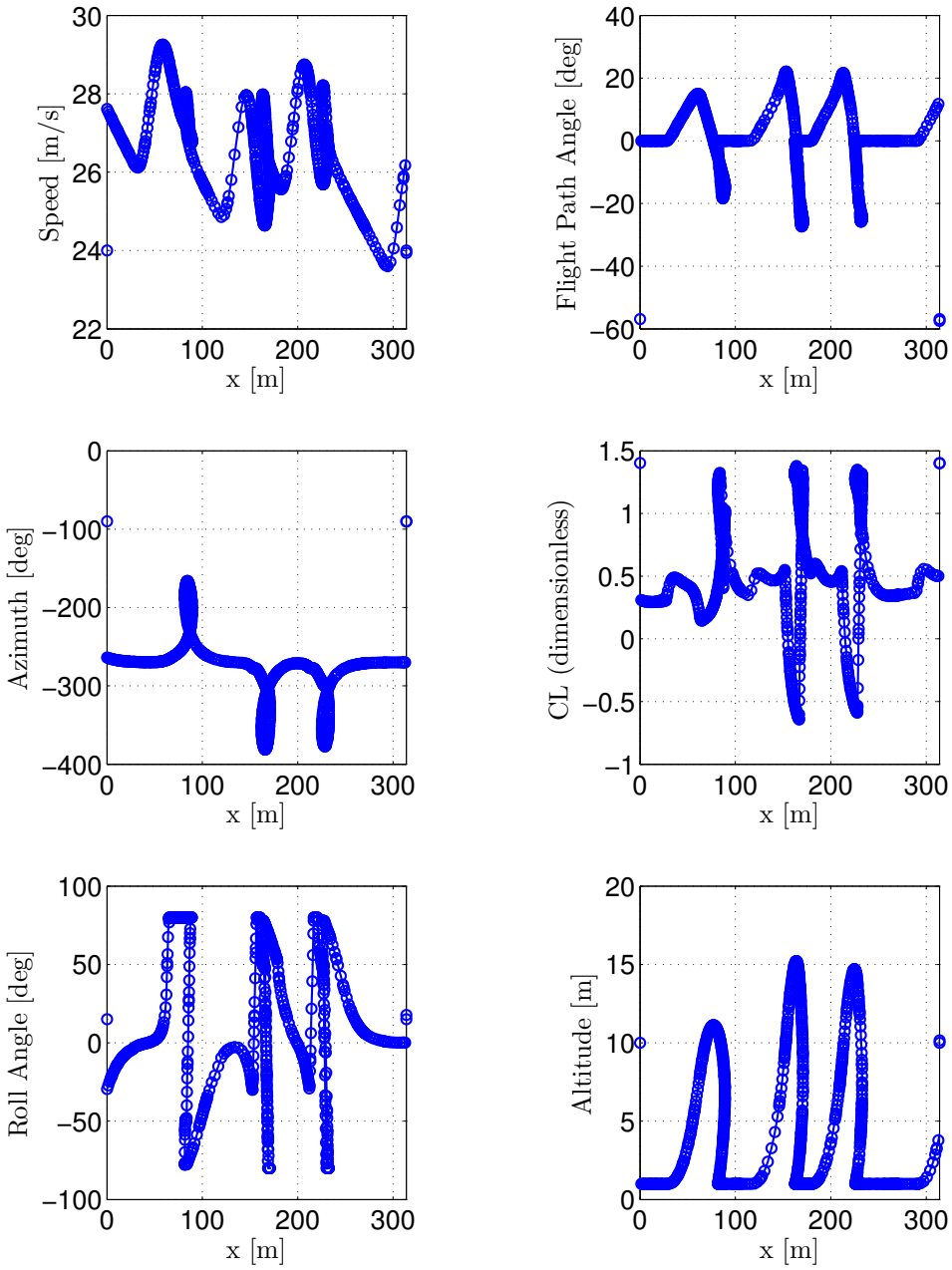


Figure 11: optimal dynamic soaring trajectory

8 Appendix: Code

```matlab
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Dan Wiese
% 16.323 - Term Project
% dS_Main.m
% Friday 09-May-2014
%------------------------------------------------------------------------------------
%DYNAMIC SOARING PROBLEM: MAIN FILE
% (1) Replace gpopsdir with the apporpriate directory on the local machine
% (2) Replace plotdir with the appropriate directory where the figures are to be
%     saved. Can simply set plotdir to homedir.
% (3) Go into dS_continuous.m to comment/uncomment different wind profiles
% (4) Run dS_Main.m. Results will be plotted.
%------------------------------------------------------------------------------------
%  -  Can generate a movie using dS_MakeMovie.m
%  -  Can plot saved data by uncommenting lines at the bottom of this script
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all;
close all;
clc;
restoredefaultpath;
thismfile=dbstack('-completenames');
homedir=fileparts(thismfile.file);
cd(homedir);
addpath(fullfile(fileparts(pwd)));
addpath('../Trajectory_Plotter');
addpath('../Saved_Solutions');
addpath('../Trajectory_Plotter/aircraft_models');
addpath('Z:\Dropbox\Dan\DOCUMENTS\School\MIT\06_Spring_2014\16323_Optimal_Control\Term_Project\Co
gpopsdir='Z:\Dropbox\Dan\DOCUMENTS\School\MIT\06_Spring_2014\16323_Optimal_Control\Term_Project\g
plotdir='Z:\Dropbox\Dan\DOCUMENTS\School\MIT\06_Spring_2014\16323_Optimal_Control\Term_Project\Fi
cd(gpopsdir)
gpopsMatlabPathSetup
cd(homedir)
% break
%Environment Data
rho         = 1.225;
beta        = 0.5;
Wxref       = -11;
zref        = 10;

%Albatross Data
CD0         = 0.033;
Emax        = 20;
g           = 9.8;
m           = 8.5;
S           = 0.65;
k           = 1/(4*Emax^2*CD0);

%Initial Condition
x0          = 0;
y0          = 0;
z0          = 10;
r0          = 0;
```

```
v0          = 24;
psi0        = 90*pi/180;
Phi0        = 0;
t0          = 0;

%Terminal Condition
xf          = 0;
yf          = 500;
zf          = 10;
rf          = 0;
vf          = 24;

%Constraints
xmin        = -100;
xmax        = 1000;
ymin        = -1000;
ymax        = 1000;
zmin        = 1.0;
zmax        = 40;
Vmin        = 12;
Vmax        = 47;
gammamin    = -75*pi/180;
gammamax    = 75*pi/180;
psimin      = -4*pi;
psimax      = 4*pi;
phimin      = -70/180*pi;
phimax      = 70/180*pi;
CLmin       = -1.0;
CLmax       = 1.5;
tfmin       = 20;
tfmax       = 30;
pmin        = -360*pi/180;
pmax        = 360*pi/180;
betamin     = beta;
betamax     = beta;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Build auxiliary data structure for GPOPS
auxdata.rho=rho;
auxdata.CD0=CD0;
auxdata.g=g;
auxdata.k=k;
auxdata.m=m;
auxdata.S=S;
auxdata.Emax=Emax;
auxdata.W0=0;
auxdata.lmin=-2;
auxdata.lmax=5;
auxdata.zref=zref;
auxdata.Wxref=Wxref;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Phase 1 Information
iphase = 1;
bounds.phase(iphase).initialtime.lower  = t0;
bounds.phase(iphase).initialtime.upper  = t0;
```

```
110 bounds.phase(iphase).finaltime.lower    = tfmin;
111 bounds.phase(iphase).finaltime.upper    = tfmax;
112 bounds.phase(iphase).initialstate.lower = [x0, y0, z0, v0, gammamin, -0.5*pi, ...
        phimin, CLmin];
113 bounds.phase(iphase).initialstate.upper = [x0, y0, z0, v0, gammamax, 1.5*pi, ...
        phimax, CLmax];
114 bounds.phase(iphase).state.lower        = [xmin, ymin, zmin, Vmin, gammamin, ...
        psimin, phimin, CLmin];
115 bounds.phase(iphase).state.upper        = [xmax, ymax, zmax, Vmax, gammamax, ...
        psimax, phimax, CLmax];
116 bounds.phase(iphase).finalstate.lower   = [xmin, -yf, zf, Vmin, gammamin, ...
        psimin, phimin, CLmin];
117 bounds.phase(iphase).finalstate.upper   = [xmax, yf, zmax, Vmax, gammamax, ...
        psimax, phimax, CLmax];
118 bounds.phase(iphase).control.lower      = [CLmin, pmin];
119 bounds.phase(iphase).control.upper      = [CLmax, pmax];
120 bounds.phase(iphase).path.lower         = auxdata.lmin;
121 bounds.phase(iphase).path.upper         = auxdata.lmax;
122
123 %Preallocate eventgroup to make trajectory periodic
124 bounds.eventgroup(1).lower              = zeros(1,6);
125 bounds.eventgroup(1).upper              = zeros(1,6);
126
127 bounds.parameter.lower                  = betamin;
128 bounds.parameter.upper                  = betamax;
129
130 tfguess    = 30;
131 N          = 100;
132 CL0        = CLmax;
133 basetime   = linspace(0,tfguess,N).';
134 xguess     = 0*basetime/tfguess;
135 yguess     = 0*basetime/tfguess;
136 zguess     = 0*basetime/tfguess;
137 vguess     = 0*basetime/tfguess;
138 gammaguess = 0*basetime/tfguess;
139 psiguess   = -1-basetime/4;
140 CLguess    = CL0*ones(N,1)/3;
141 phiguess   = -ones(N,1);
142 pguess     = -ones(N,1);
143 betaguess  = beta;
144
145 [xguess, yguess, zguess, vguess, gammaguess, psiguess, phiguess]=dS_LoadGuess(N);
146
147 %ASSEMBLE GUESS
148 guess.phase(iphase).time    = [basetime];
149 guess.phase(iphase).state   = [xguess, yguess, zguess, vguess, gammaguess, ...
        psiguess, phiguess, CLguess];
150 guess.phase(iphase).control = [CLguess, pguess];
151 guess.parameter             = [betaguess];
152
153 setup.name                              = 'Dynamic-Soaring-Problem';
154 setup.functions.continuous              = @dS_Continuous;
155 setup.functions.endpoint                = @dS_Endpoint;
156 setup.nlp.solver                        = 'ipopt';
157 % setup.nlp.solver                        = 'snopt';
158 setup.nlp.options.ipopt.linear_solver   = 'ma57';
```

```
% setup.nlp.options.ipopt.linear_solver = 'mumps';
setup.bounds                             = bounds;
setup.guess                              = guess;
setup.auxdata                            = auxdata;
setup.derivatives.supplier               = 'sparseCD';
setup.derivatives.derivativelevel        = 'second';
setup.scales.method                      = 'automatic-bounds';
setup.method                             = 'RPMintegration';
setup.mesh.method                        = 'hp1';
% setup.mesh.method                        = 'hpSliding';
setup.mesh.tolerance                     = 1e-5;
setup.mesh.colpointsmin                  = 4;
setup.mesh.colpointsmax                  = 10;

output = gpops2(setup);
solution = output.result.solution;
solname=dS_FileNamer;
save(solname,'solution')
windtype='log';

%% Plot Results

%UNCOMMENT TO PLOT SAVED SOLUTION DATA

% %Linear wind profile:
% load('linwind.mat','solution')
% windtype='lin';

% %Logarithmic wind profile:
% load('logwind.mat','solution')
% windtype='log';

% %Power law wind profile:
% load('expwind.mat','solution')
% windtype='exp';

dS_Plotter(solution,plotdir,homedir,windtype);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Dan Wiese
% 16.323 - Term Project
% dS_Continuous.m
% Friday 09-May-2014
%-----------------------------------------------------------------------------------
% DYNAMIC SOARING: EQUATIONS OF MOTION
% This function contains the equations of motion for the albatross to be used with
% GPOPS.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function phaseout = dS_Continuous(input)
%Take out the time, state, control, and parameter from input structure
t          = input.phase(1).time;
X          = input.phase(1).state;
```

```
u          = input.phase(1).control;
param      = input.phase(1).parameter;

%Take out the auxiliary data include atmospheric and albatross properties
rho        = input.auxdata.rho;
S          = input.auxdata.S;
CD0        = input.auxdata.CD0;
g          = input.auxdata.g;
k          = input.auxdata.k;
m          = input.auxdata.m;
Emax       = input.auxdata.Emax;

%Define the individual state variables from the state vector
x          = X(:,1);
y          = X(:,2);
z          = X(:,3);
V          = X(:,4);
gamma      = X(:,5);
psi        = X(:,6);
phi        = X(:,7);
CLa        = X(:,8);

CLcmd      = u(:,1);
p          = u(:,2);
beta       = param(:,1);

%Calculate some stuff
singamma   = sin(gamma);
cosgamma   = cos(gamma);
sinpsi     = sin(psi);
cospsi     = cos(psi);
sinphi     = sin(phi);
cosphi     = cos(phi);
vcosgamma  = V.*cosgamma;

% %BOUNDARY LAYER PROFILE (LINEAR)
% W0            = input.auxdata.W0;
% Wx            = -(beta.*z+W0);
% dWxdz         = -beta;
% dWxdt         = dWxdz.*V.*singamma;

%BOUNDARY LAYER PROFILE (LOGARITHMIC)
Wxref      = -11;
zref       = 10;
z0         = 0.15; %0.05-0.15 Sukumar, Selig, or 0.03sachs
Const      = (Wxref/log(zref/z0));
Wx         = Const*log((z+z0)/z0);
dWxdz      = Const*(1./(z+z0));
dWxdt      = dWxdz.*V.*singamma;

% %BOUNDARY LAYER PROFILE (POWER-LAW)
% Wxref      = -11;
% zref       = 10;
% a          = 3;
% Wx         = (Wxref/(1-exp(-a)))*(1-exp(-a*(z/zref)));
% dWxdz      = (a/zref)*(Wxref/(1-exp(-a)))*exp(-a*(z./zref));
```

```
% dWxdt       = dWxdz.*V.*singamma;

%Dynamic Soaring Thrust
dWxdtsinpsi    = dWxdt.*sinpsi;

%Calculate some stuff
CLsq        = CLa.^2;
vsq         = V.^2;
L           = 0.5*rho*S*CLa.*vsq;
D           = 0.5*(rho*S)*(CD0+k*CLsq).*vsq;

%Equations of Motion
xdot        = vcosgamma.*sinpsi+Wx;
ydot        = vcosgamma.*cospsi;
zdot        = V.*singamma;
Vdot        = -(1/m)*D-(g*singamma)-(dWxdtsinpsi.*cosgamma);
gammadot    = (cosphi./(m*V)).*L-(g*cosgamma./V)+(dWxdtsinpsi.*singamma./V);
psidot      = (sinphi./(m*V.*cosgamma)).*L-(dWxdt.*cospsi)./(V.*cosgamma);
phidot      = p;
CLadot      = -3*CLa+3*CLcmd;


%Calculate path
ngconstant     = (0.5*rho*S/m/g);
ng             = (0.5*rho*S/m/g).*CLcmd.*V.^2;

phaseout.dynamics  = [xdot, ydot, zdot, Vdot, gammadot, psidot, phidot, CLadot];
phaseout.path = ng;

end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Dan Wiese
% 16.323 - Term Project
% dS_Endpoint.m
% Friday 09-May-2014
%-----------------------------------------------------------------------------------
%DYNAMIC SOARING: BOUNDARY CONDITIONS AND OBJECTIVE FUNCTION
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function output = dS_Endpoint(input)

t0     = input.phase(1).initialtime;
tf     = input.phase(1).finaltime;
X0     = input.phase(1).initialstate;
Xf     = input.phase(1).finalstate;
beta   = input.parameter;

%Take apart initial state
x0=X0(1);
y0=X0(2);
h0=X0(3);
V0=X0(4);
gamma0=X0(5);
```

```
psi0=X0(6);
phi0=X0(7);
CL0=X0(8);

%Take apart final state
xf=Xf(1);
yf=Xf(2);
hf=Xf(3);
Vf=Xf(4);
gammaf=Xf(5);
psif=Xf(6);
phif=Xf(7);
CLf=Xf(8);

%Enforce periodicity of albatross flight
output.eventgroup(1).event = [phi0-phif hf-h0 Vf-V0 psi0-psif gamma0-gammaf ...
    CL0-CLf];

%Maximize distance traveled upwind
output.objective=-xf;

end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Dan Wiese
% 16.323 - Term Project
% dS_LoadGuess.m
% Friday 09-May-2014
%------------------------------------------------------------------------------------
%DYNAMIC SOARING: LOAD A SOLUTION GUESS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [xguess,yguess,zguess,vguess, gammaguess, psiguess, ...
    phiguess]=dS_LoadGuess(N)

% %Saved guess file, works for any trajectory
load('guess.mat','solution')

x=solution.phase(1).state(:,1);
y=solution.phase(1).state(:,2);
z=solution.phase(1).state(:,3);
V=solution.phase(1).state(:,4);
gamma=solution.phase(1).state(:,5);
psi=solution.phase(1).state(:,6);
phi=solution.phase(1).state(:,7);
CLa=solution.phase(1).state(:,8);

xguess=imresize(x, [N 1], 'nearest');
yguess=imresize(y, [N 1], 'nearest');
zguess=imresize(z, [N 1], 'nearest');
vguess=imresize(V, [N 1], 'nearest');
gammaguess=imresize(gamma, [N 1], 'nearest');
psiguess=imresize(psi, [N 1], 'nearest');
```

```
phiguess=imresize(phi, [N 1], 'nearest');

xguess=smooth(xguess);
yguess=smooth(yguess);
zguess=smooth(zguess);

end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Dan Wiese
% 16.323 - Term Project
% dS_PlotWind.m
% Friday 09-May-2014
%-------------------------------------------------------------------------------
%DYNAMIC SOARING: PLOT WIND VELOCITY PROFILE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Define altitude range over which to plot wind profile
z=0:1:40;

%Define wind profile
Wxref=-11;
zref=10;
z0=0.15; %0.05-0.15 Sukumar, Selig, or 0.03sachs
a=3;
beta=Wxref/zref;

Wxlog=Wxref*(log((z+z0)/z0))/(log(zref/z0));
Wxlin=beta*z;
Wxexp=(Wxref/(1-exp(-a)))*(1-exp(-a*(z/zref)));

%-------------------------------------------------------------------------------
axisfontsize=16;
plotlinewidth=3;
plotlinecolor=[0 0 0];

figure(1)
pp=plot(Wxlin,z,'linewidth',plotlinewidth,'color',plotlinecolor);
hold on
for ii=5:5:z(end);
    arrow([0 z(ii+1)],[Wxlin(ii+1) z(ii+1)])
end
xl=xlabel('Wind Velocity [m/s]','interpreter','latex','FontSize',axisfontsize);
yl=ylabel('Altitude [m]','interpreter','latex','FontSize',axisfontsize);
set(gca,'FontSize',16);
grid on;
ylim([0 z(end)]);
% xlim([-15 0]);
set(gcf,'Units','pixels');
set(gcf,'PaperUnits','inches','PaperPosition',[0 0 4.5 6]);
set(gcf,'PaperPositionMode','manual')
set(gcf,'InvertHardCopy','off');
set(gcf,'color',[1 1 1])
```

```
cd(plotdir)
print('-depsc','-r600','plot_wind_lin.eps');
cd(homedir)

figure(2)
pp=plot(Wxlog,z,'linewidth',plotlinewidth,'color',plotlinecolor);
hold on
for ii=5:5:z(end);
    arrow([0 z(ii+1)],[Wxlog(ii+1) z(ii+1)])
end
xl=xlabel('Wind Velocity [m/s]','interpreter','latex','FontSize',axisfontsize);
yl=ylabel('Altitude [m]','interpreter','latex','FontSize',axisfontsize);
set(gca,'FontSize',16);
grid on;
ylim([0 z(end)]);
% xlim([-15 0]);
set(gcf,'Units','pixels');
set(gcf,'PaperUnits','inches','PaperPosition',[0 0 4.5 6]);
set(gcf,'PaperPositionMode','manual')
set(gcf,'InvertHardCopy','off');
set(gcf,'color',[1 1 1])
cd(plotdir)
print('-depsc','-r600','plot_wind_log.eps');
cd(homedir)

figure(3)
pp=plot(Wxexp,z,'linewidth',plotlinewidth,'color',plotlinecolor);
hold on
for ii=5:5:z(end);
    arrow([0 z(ii+1)],[Wxexp(ii+1) z(ii+1)])
end
xl=xlabel('Wind Velocity [m/s]','interpreter','latex','FontSize',axisfontsize);
yl=ylabel('Altitude [m]','interpreter','latex','FontSize',axisfontsize);
set(gca,'FontSize',16);
grid on;
ylim([0 z(end)]);
% xlim([-15 0]);
set(gcf,'Units','pixels');
set(gcf,'PaperUnits','inches','PaperPosition',[0 0 4.5 6]);
set(gcf,'PaperPositionMode','manual')
set(gcf,'InvertHardCopy','off');
set(gcf,'color',[1 1 1])
cd(plotdir)
print('-depsc','-r600','plot_wind_exp.eps');
cd(homedir)

figure(4)
pp=plot(Wxlin,z,'linestyle','--','linewidth',plotlinewidth,'color',plotlinecolor);
hold on
pp=plot(Wxlog,z,'linestyle','-','linewidth',plotlinewidth,'color',plotlinecolor);
pp=plot(Wxexp,z,'linestyle',':','linewidth',plotlinewidth,'color',plotlinecolor);
xl=xlabel('Wind Velocity [m/s]','interpreter','latex','FontSize',axisfontsize);
yl=ylabel('Altitude [m]','interpreter','latex','FontSize',axisfontsize);
set(gca,'FontSize',16);
grid on;
legend('Linear','Logarithmic','Power-Law','location','SouthWest');
```

```
ylim([0 z(end)]);
% xlim([-15 0]);
set(gcf,'Units','pixels');
set(gcf,'PaperUnits','inches','PaperPosition',[0 0 4.5 6]);
set(gcf,'PaperPositionMode','manual')
set(gcf,'InvertHardCopy','off');
set(gcf,'color',[1 1 1])
cd(plotdir)
print('-depsc','-r600','plot_wind_linlog.eps');
cd(homedir)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Dan Wiese
% 16.323 - Term Project
% dS_Plotter.m
% Friday 09-May-2014
%-------------------------------------------------------------------------------------
% DYNAMIC SOARING: PLOTTER
% This function takes as its input the solution structure from GPOPS and plots the
% state and control histories.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function dS_Plotter(solution,plotdir,homedir,windtype)

%Take the state and input time histories from the solution structure
t=solution.phase(1).time;
x=solution.phase(1).state(:,1);
y=solution.phase(1).state(:,2);
z=solution.phase(1).state(:,3);
V=solution.phase(1).state(:,4);
gamma=solution.phase(1).state(:,5);
psi=solution.phase(1).state(:,6);
phi=solution.phase(1).state(:,7);
CL=solution.phase(1).state(:,8);
CLcmd=solution.phase(1).control(:,1);
p=solution.phase(1).control(:,2);

axisfontsize=16;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

cd(plotdir)

figure(1);
pp=plot3(x,y,z,'-o');
hold on
pp=plot3(x,y,0*z,'-','color',[1 0 0],'linewidth',4);
plot3(x(1),y(1),z(1),'o','markersize',16,'linewidth',4,'color',[0 0.7 0])
plot3(x(end),y(end),z(end),'x','markersize',20,'linewidth',4,'color',[1 0 0])
xl=xlabel('x [m]','interpreter','latex','FontSize',axisfontsize);
yl=ylabel('y [m]','interpreter','latex','FontSize',axisfontsize);
zl=zlabel('Altitude [m]','interpreter','latex','FontSize',axisfontsize);
```

```
grid on;
set(xl,'FontSize',18);
set(yl,'FontSize',18);
set(zl,'FontSize',18);
set(gca,'FontSize',16);
set(pp,'LineWidth',1.25);
set(gca,'FontSize',20);
set(gca,'FontName','Times');
%Format the figure
set(gcf,'Units','pixels');
set(gcf,'PaperUnits','inches','PaperPosition',[0 0 7.5 4]);
set(gcf,'PaperPositionMode','manual')
set(gcf,'InvertHardCopy','off');
set(gcf,'color',[1 1 1])
plotname1=strcat('plot_xyz_',windtype,'.eps');
print('-depsc','-r600',plotname1);

%-------------------------------------------------------------------------------------

figure(2);
pp=plot(x,y,'-o');
hold on
plot(x(1),y(1),'o','markersize',16,'linewidth',4,'color',[0 0.7 0])
plot(x(end),y(end),'x','markersize',20,'linewidth',4,'color',[1 0 0])
xl=xlabel('x [m]','interpreter','latex','FontSize',axisfontsize);
yl=ylabel('y [m]','interpreter','latex','FontSize',axisfontsize);
set(xl,'FontSize',18);
set(yl,'FontSize',18);
set(gca,'FontSize',16);
set(pp,'LineWidth',1.25);
daspect([1 1 1]);
%Format the figure
set(gcf,'Units','pixels');
set(gcf,'PaperUnits','inches','PaperPosition',[0 0 7.5 4]);
set(gcf,'PaperPositionMode','manual')
set(gcf,'InvertHardCopy','off');
set(gcf,'color',[1 1 1])
grid on;
plotname2=strcat('plot_xy_',windtype,'.eps');
print('-depsc','-r600',plotname2);

%-------------------------------------------------------------------------------------

figure(3);
subplot(3,2,1)
    pp=plot(t,V,'-o');
    xl=xlabel('Time [s]','interpreter','latex','FontSize',axisfontsize);
    yl=ylabel('Speed [m/s]','interpreter','latex','FontSize',axisfontsize);
%     set(gca,'FontSize',16,'YTick',40:40:240);
    set(gca,'FontSize',16);
    set(pp,'LineWidth',1.25);
    grid on;
    xlim([t(1) t(end)]);
subplot(3,2,2)
    pp=plot(t,gamma*180/pi,'-o');
    xl=xlabel('Time [s]','interpreter','latex','FontSize',axisfontsize);
```

```
    yl=ylabel('Flight Path Angle ...
        [deg]','interpreter','latex','FontSize',axisfontsize);
    set(gca,'FontSize',16);
    set(pp,'LineWidth',1.25);
    grid on;
    xlim([t(1) t(end)]);
subplot(3,2,3)
    pp=plot(t,psi*180/pi,'-o');
    xl=xlabel('Time [s]','interpreter','latex','FontSize',axisfontsize);
    yl=ylabel('Azimuth [deg]','interpreter','latex','FontSize',axisfontsize);
    set(gca,'FontSize',16);
    set(pp,'LineWidth',1.25);
    grid on;
    xlim([t(1) t(end)]);
subplot(3,2,4)
    pp=plot(t,CL,'-o');
    xl=xlabel('Time [s]','interpreter','latex','FontSize',axisfontsize);
    yl=ylabel('CL (dimensionless)','interpreter','latex','FontSize',axisfontsize);
    set(gca,'FontSize',16);
    set(pp,'LineWidth',1.25);
    grid on;
    xlim([t(1) t(end)]);
subplot(3,2,5)
    pp=plot(t,phi*180/pi,'-o');
    xl=xlabel('Time [s]','interpreter','latex','FontSize',axisfontsize);
    yl=ylabel('Roll Angle [deg]','interpreter','latex','FontSize',axisfontsize);
    set(gca,'FontSize',16);
    set(pp,'LineWidth',1.25);
    grid on;
    xlim([t(1) t(end)]);
subplot(3,2,6)
    pp=plot(t,z,'-o');
    xl=xlabel('Time [s]','interpreter','latex','FontSize',axisfontsize);
    yl=ylabel('Altitude [m]','interpreter','latex','FontSize',axisfontsize);
    set(gca,'FontSize',16);
    set(pp,'LineWidth',1.25);
    grid on;
    xlim([t(1) t(end)]);
%Format the figure
set(gcf,'Units','pixels');
set(gcf,'PaperUnits','inches','PaperPosition',[0 0 10 12]);
set(gcf,'PaperPositionMode','manual')
set(gcf,'InvertHardCopy','off');
set(gcf,'color',[1 1 1])
plotname3=strcat('plot_versus_t_',windtype,'.eps');
print('-depsc','-r600',plotname3);

%-------------------------------------------------------------------------------

figure(4);
subplot(3,2,1)
    pp=plot(x,V,'-o');
    xl=xlabel('x [m]','interpreter','latex','FontSize',axisfontsize);
    yl=ylabel('Speed [m/s]','interpreter','latex','FontSize',axisfontsize);
%     set(gca,'FontSize',16,'YTick',40:40:240);
    set(gca,'FontSize',16);
```

```
    set(pp,'LineWidth',1.25);
    grid on;
    xlim([x(1) x(end)]);
subplot(3,2,2)
    pp=plot(x,gamma*180/pi,'-o');
    xl=xlabel('x [m]','interpreter','latex','FontSize',axisfontsize);
    yl=ylabel('Flight Path Angle ...
        [deg]','interpreter','latex','FontSize',axisfontsize);
    set(gca,'FontSize',16);
    set(pp,'LineWidth',1.25);
    grid on;
    xlim([x(1) x(end)]);
subplot(3,2,3)
    pp=plot(x,psi*180/pi,'-o');
    xl=xlabel('x [m]','interpreter','latex','FontSize',axisfontsize);
    yl=ylabel('Azimuth [deg]','interpreter','latex','FontSize',axisfontsize);
    set(gca,'FontSize',16);
    set(pp,'LineWidth',1.25);
    grid on;
    xlim([x(1) x(end)]);
subplot(3,2,4)
    pp=plot(x,CL,'-o');
    xl=xlabel('x [m]','interpreter','latex','FontSize',axisfontsize);
    yl=ylabel('CL (dimensionless)','interpreter','latex','FontSize',axisfontsize);
    set(gca,'FontSize',16);
    set(pp,'LineWidth',1.25);
    grid on;
    xlim([x(1) x(end)]);
subplot(3,2,5)
    pp=plot(x,phi*180/pi,'-o');
    xl=xlabel('x [m]','interpreter','latex','FontSize',axisfontsize);
    yl=ylabel('Roll Angle [deg]','interpreter','latex','FontSize',axisfontsize);
    set(gca,'FontSize',16);
    set(pp,'LineWidth',1.25);
    grid on;
    xlim([x(1) x(end)]);
subplot(3,2,6)
    pp=plot(x,z,'-o');
    xl=xlabel('x [m]','interpreter','latex','FontSize',axisfontsize);
    yl=ylabel('Altitude [m]','interpreter','latex','FontSize',axisfontsize);
    set(gca,'FontSize',16);
    set(pp,'LineWidth',1.25);
    grid on;
    xlim([x(1) x(end)]);
%Format the figure
set(gcf,'Units','pixels');
set(gcf,'PaperUnits','inches','PaperPosition',[0 0 10 12]);
set(gcf,'PaperPositionMode','manual')
set(gcf,'InvertHardCopy','off');
set(gcf,'color',[1 1 1])
plotname4=strcat('plot_versus_x_',windtype,'.eps');
print('-depsc','-r600',plotname4);

%-------------------------------------------------------------------------------

figure(5);
```

```
subplot(2,3,1)
    pp=plot(t,V,'-o');
    xl=xlabel('Time [s]','interpreter','latex','FontSize',axisfontsize);
    yl=ylabel('Speed [m/s]','interpreter','latex','FontSize',axisfontsize);
%     set(gca,'FontSize',16,'YTick',40:40:240);
    set(gca,'FontSize',16);
    set(pp,'LineWidth',1.25);
    grid on;
    xlim([t(1) t(end)]);
subplot(2,3,2)
    pp=plot(t,gamma*180/pi,'-o');
    xl=xlabel('Time [s]','interpreter','latex','FontSize',axisfontsize);
    yl=ylabel('Flight Path Angle ...
        [deg]','interpreter','latex','FontSize',axisfontsize);
    set(gca,'FontSize',16);
    set(pp,'LineWidth',1.25);
    grid on;
    xlim([t(1) t(end)]);
subplot(2,3,3)
    pp=plot(t,psi*180/pi,'-o');
    xl=xlabel('Time [s]','interpreter','latex','FontSize',axisfontsize);
    yl=ylabel('Azimuth [deg]','interpreter','latex','FontSize',axisfontsize);
    set(gca,'FontSize',16);
    set(pp,'LineWidth',1.25);
    grid on;
    xlim([t(1) t(end)]);
subplot(2,3,4)
    pp=plot(t,CL,'-o');
    xl=xlabel('Time [s]','interpreter','latex','FontSize',axisfontsize);
    yl=ylabel('CL (dimensionless)','interpreter','latex','FontSize',axisfontsize);
    set(gca,'FontSize',16);
    set(pp,'LineWidth',1.25);
    grid on;
    xlim([t(1) t(end)]);
subplot(2,3,5)
    pp=plot(t,phi*180/pi,'-o');
    xl=xlabel('Time [s]','interpreter','latex','FontSize',axisfontsize);
    yl=ylabel('Roll Angle [deg]','interpreter','latex','FontSize',axisfontsize);
    set(gca,'FontSize',16);
    set(pp,'LineWidth',1.25);
    grid on;
    xlim([t(1) t(end)]);
subplot(2,3,6)
    pp=plot(t,z,'-o');
    xl=xlabel('Time [s]','interpreter','latex','FontSize',axisfontsize);
    yl=ylabel('Altitude [m]','interpreter','latex','FontSize',axisfontsize);
    set(gca,'FontSize',16);
    set(pp,'LineWidth',1.25);
    grid on;
    xlim([t(1) t(end)]);
%Format the figure
set(gcf,'Units','pixels');
set(gcf,'PaperUnits','inches','PaperPosition',[0 0 16 10]);
set(gcf,'PaperPositionMode','manual')
set(gcf,'InvertHardCopy','off');
set(gcf,'color',[1 1 1])
```

```
plotname3=strcat('plot_versus_t_',windtype,'_a.eps');
print('-depsc','-r600',plotname3);

%-------------------------------------------------------------------------------

cd(homedir)

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Dan Wiese
% 16.323 - Term Project
% dS_FileNamer.m
% Friday 09-May-2014
%-------------------------------------------------------------------------------
%DYNAMIC SOARING: CREATE THE NAME WITH TIMESTAMP TO SAVE DATA
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function solname=dS_FileNamer

thetime=clock;
year=thetime(1);
month=thetime(2);
date=thetime(3);
hour=thetime(4);
minute=thetime(5);
second=thetime(6);

year=sprintf('%04d',year);
month=sprintf('%02d',month);
date=sprintf('%02d',date);
hour=sprintf('%02d',hour);
minute=sprintf('%02d',minute);
second=sprintf('%02d',round(second));

solname=strcat('sol_',year,month,date,'_',hour,minute,second,'.mat');

end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Dan Wiese
% 16.323 - Term Project
% dS_Continuous.m
% Friday 09-May-2014
%-------------------------------------------------------------------------------
% DYNAMIC SOARING: EQUATIONS OF MOTION
% This function contains the equations of motion for the albatross to be used with
% GPOPS.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

load('linwind','solution')
```

```
13
14 %Take the state and input time histories from the solution structure
15 t=solution.phase(1).time;
16 x=solution.phase(1).state(:,1);
17 y=solution.phase(1).state(:,2);
18 z=solution.phase(1).state(:,3);
19 V=solution.phase(1).state(:,4);
20 gamma=solution.phase(1).state(:,5);
21 psi=solution.phase(1).state(:,6);
22 phi=solution.phase(1).state(:,7);
23 CL=solution.phase(1).state(:,8);
24 CLcmd=solution.phase(1).control(:,1);
25 p=solution.phase(1).control(:,2);
26
27 alpha=CL/2;
28 theta=alpha+gamma;
29 psi=psi-pi/2;
30
31 scale_factor=0.15;
32 step=0.005;
33 aircrafttype='cessna';
34
35 npathend=400;
36 N=200;
37
38 clearvars mov
39
40 xtraj=imresize(x(1:npathend), [N 1], 'nearest');
41 ytraj=imresize(y(1:npathend), [N 1], 'nearest');
42 ztraj=imresize(z(1:npathend), [N 1], 'nearest');
43 thetatraj=imresize(theta(1:npathend), [N 1], 'nearest');
44 phitraj=imresize(phi(1:npathend), [N 1], 'nearest');
45 psitraj=imresize(psi(1:npathend), [N 1], 'nearest');
46
47 for ii=1:N
48 jj=ii*1;
49 trajectory2(xtraj,ytraj,ztraj,thetatraj,phitraj,psitraj,10,10,aircrafttype);
50 hold on
51 %ground track
52 plot3(xtraj(1:jj),ytraj(1:jj),0*ztraj(1:jj),'color',[1 0 0],'linewidth',3);
53
54
55
56 trajectory2(xtraj(jj),ytraj(jj),ztraj(jj),thetatraj(jj),phitraj(jj),psitraj(jj),scale_factor,step
57 %Plot O and X at beginning and end
58 plot3(xtraj(1),ytraj(1),ztraj(1),'o','markersize',16,'linewidth',4,'color',[0 ...
       0.7 0])
59 plot3(xtraj(end),ytraj(end),ztraj(end),'x','markersize',20,'linewidth',4,'color',[1 ...
       0 0])
60
61
62 % image(k.*peaks,'parent',hax); %puts image in invisible axes
63
64
65
66
```

```
%Set axes limits
xlim([min(xtraj) max(xtraj)]);
ylim([min(ytraj) max(ytraj)]);
zlim([0 40]);

% set(gca,'CameraViewAngle',10)
set(gca,'CameraPosition',[-141 -500 200]*2);

set(gca,'OuterPosition',[0.2 0.2 0.8 0.8])
% set(gcf,'PaperUnits','inches','PaperPosition',[0 0 7.5 4]);

set(gca,'nextplot','replacechildren');

% %OVERHEAD VIEW
% set(gca,'CameraPosition',[25 -40 200]);
% set(gca,'CameraTarget',[25 -40 0])

mov(ii)=getframe(gca);

% writeVideo(vidObj, getframe(gca));

clf;
end

movie2avi(mov, 'lin_video_a.avi', 'compression','None', 'fps',20,'Quality',100);


% close all
%
%
%     vidObj = VideoWriter('peaks.avi');
%     open(vidObj);
%
%     % Create an animation.
%     trajectory2(xtraj,ytraj,ztraj,thetatraj,phitraj,psitraj,10,10,aircrafttype);
%     axis tight
%     set(gca,'nextplot','replacechildren');
%
%     for ii=1:N
%        jj=ii*1;
%             ...
    trajectory2(xtraj,ytraj,ztraj,thetatraj,phitraj,psitraj,10,10,aircrafttype);
%             hold on
%             ...
    trajectory2(xtraj(jj),ytraj(jj),ztraj(jj),thetatraj(jj),phitraj(jj),psitraj(jj),scale_factor,s
%             %Plot O and X at beginning and end
%             ...
    plot3(xtraj(1),ytraj(1),ztraj(1),'o','markersize',16,'linewidth',4,'color',[0 ...
    0.7 0])
%             ...
    plot3(xtraj(end),ytraj(end),ztraj(end),'x','markersize',20,'linewidth',4,'color',[1 ...
    0 0])
%             %Set axes limits
%             xlim([min(xtraj) max(xtraj)]);
%             ylim([min(ytraj) max(ytraj)]);
%             zlim([0 40]);
```

```
117 %
118 %             % set(gca,'CameraViewAngle',10)
119 %             set(gca,'CameraPosition',[-141 -500 200]*2);
120 %
121 %             set(gca,'OuterPosition',[0.2 0.2 0.8 0.8])
122 %             % set(gcf,'PaperUnits','inches','PaperPosition',[0 0 7.5 4]);
123 %
124 %             set(gca,'nextplot','replacechildren');
125 %
126 %         % Write each frame to the file.
127 %         currFrame = getframe;
128 %         writeVideo(vidObj,currFrame);
129 %     end
130 %
131 %     % Close the file.
132 %     close(vidObj);
```