PATH PREDICTION FOR AN EARTH BASED DEMONSTRATION FLIGHT

Anuj Arora
Mentor – Prof. Jerrold Marsden, Co-mentor – Ms. Claire Newman, Philip Du Toit

Abstract - For navigation on distant planets/moons, control from the Earth is too slow for practical purposes. Titan, Saturn's moon, presents one such example. Thus, autonomous navigation is preferred. An Earth based demonstration flight will be held in the Mojave Desert to test autonomous navigation. The aim of this project is to investigate the balloon's optimal flight path (minimizing fuel consumption by exploiting wind fields) ahead of time by assuming a perfect wind model, and then to explore ways of optimally modifying this path during the flight itself. The DMOC (Discrete Mechanics and Optimal Control) algorithm is used as a tool to generate the optimal path to be followed, which minimizes a cost function (the control force required from the vehicle was minimized). The code, using DMOC, is setup to take in a start point, final point and initial velocity and output a trajectory that minimizes aerobot control force/fuel consumption by exploiting the wind fields to help the aerobot reach its destination. The wind fields are generated using the Weather Research and Forecasting (WRF) Model. Future work will involve perfecting this method for autonomous navigation (to minimize fuel consumption), updating the aerobot model to include drag forces experienced (currently not implemented) and possibly implementing a method that uses pre-computed primitives (using the DMOC algorithm) to reduce on-board computation time. This would have implications not only for the Titan mission, but also for future navigation missions and possibly Earth based navigations as well.

Introduction

In this paper, a method for autonomous navigation, exploiting local wind fields to minimize fuel expenditure, is proposed for Montgolfier balloons/aerobots that are used for navigation on other planets/moons. It is hoped that the methods proposed can someday be utilized on Titan, Saturn's largest moon.

The Cassini-Huygens mission has shown Saturn's moon, Titan, as an exotic world, unlike any other, but bearing some uncanny resemblances to the Earth[1]. The Cassini orbiter radar has revealed vast expanses of equatorial sand dunes and high latitude seas believed to contain liquid methane and ethane[2][3]. The view at one place obtained by the Huygens probe has shown a complex valley drainage network believed to be carved by flash floods of liquid hydrocarbons. This is one of the reasons why Titan is of such great interest amongst scientists[4].

NASA has, thus, recently announced its intention of going back to Titan in collaboration with European and Japanese Space agencies. One possible mode of exploration that can be used on Titan is a lighter –than-air vehicle that could operate for years in Titan's cold dense atmosphere and descend frequently to the surface for close up observations and sampling. A hot air (Montgolfiere)

balloon (with vertical control provided by heating air to produce buoyancy) is one option[5]. Another option is an aerobot (with control along the x and y as well as z axis)[6][7].

To navigate on Titan, autonomous control will be required. For this purpose, an Earth based demonstration flight has been proposed in the Mojave Desert in the summer of 2009, where path prediction for the aerobot/balloon to be used will be provided using knowledge of local wind fields only[8]. The idea of this test flight is to demonstrate that a balloon can autonomously navigate in the Earth's atmosphere (where we are familiar with the wind patterns) and thus hopefully show that it might be possible to do the same on Titan (for which we currently possess very limited knowledge about atmospheric conditions).

In order to be able to generate an optimal trajectory to be followed, an initial point (the starting point of the aerobot/balloon), a final point (the desired final destination of the aerobot/balloon) and an initial velocity are provided. A final velocity can also be incorporated, if desired. Using this and knowledge of local wind fields, an optimal trajectory needs to be calculated that minimizes the control effort required (and thus, indirectly minimizing the fuel consumption of the aerobot/balloon). The trajectory is discretized, i.e. points along the optimal trajectory are found and the resulting trajectory is pieced together using the points generated (during test runs, 50 points were used for trajectory generation). To generate the optimal path, the DMOC (Discrete Mechanics and Optimal Control) algorithm is used (see Methods section). DMOC gives the optimal points along the trajectory, along with the velocity and control forces at each of these points and the time between them. In order to use the DMOC algorithm, an initial guess of the trajectory needs to be provided. The constraint equations need to be set up as well (to ensure that the resulting trajectory obeys the laws of motion). An objective function needs to be provided (that shall be minimized). The objective function minimized is the fuel expenditure/control force needed from the balloon. These parameters are then fed into fmincon, a MATLAB minimization routine, that finds a local minimum for the given objective function and outputs the optimal parameters.

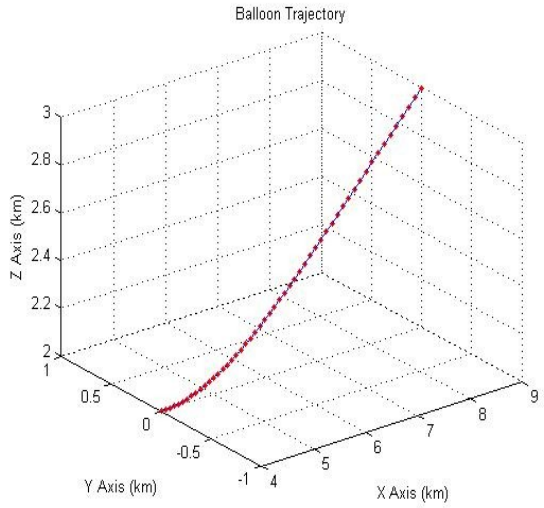
The wind fields used as input are obtained from the Weather Research and Forecasting (WRF) Model (see Methods section). The WRF Model outputs wind fields every hour on constant sigma levels (where sigma pressure = surface pressure) on an irregular, curvilinear grid. The WRF output is then interpolated onto constant altitude levels and a regular Cartesian grid, which is much more convenient for subsequent calculations during which WRF wind data are interpolated both spatially and temporally to provide predicted/sample wind fields at the locations and times required. Since the test flight will, tentatively, be held in the Mojave Desert, the wind fields used are taken from a 5 day simulation of the Death Valley region in the Mojave Desert, starting at 12 pm on July 5, 2005.

Results using synthetic wind fields

The DMOC algorithm was first tested using synthetic wind fields for which the optimal trajectory was either known or easy to intuit (see methods section). Initially, to test whether the DMOC algorithm had been implemented correctly, the code was run using synthetic wind fields, where the correct solution is known. The initial guess for the optimal trajectory provided was a straight line

trajectory. Only vertical control was provided to the balloon. The solution in such cases was as expected, showing that the DMOC algorithm was correctly implemented.

As an example, without any wind at all points, the balloon was able to move only in the vertical direction, which is correct since it has control force in the vertical direction only. Another, slightly more complicated example used for testing the implementation of DMOC involved considering wind fields in the x-direction only. The winds used were (0.2z,0,0), i.e. winds are blowing in the x-direction only, which depend on the z-coordinate of that position (winds are constant in the x-y plane). An initial position of (4,0,2), a final position of (9,0,3) and an an initial velocity of (0,0,0) for the balloon were input. The number of discretization points along the trajectory given was 50. With an initial guess of 0.16 for the time step 'h', the solution for the optimal trajectory and the optimal control forces needed from the balloon are shown in Figure 1 and 2 respectively. The optimal 'h' output in this case was 0.207.



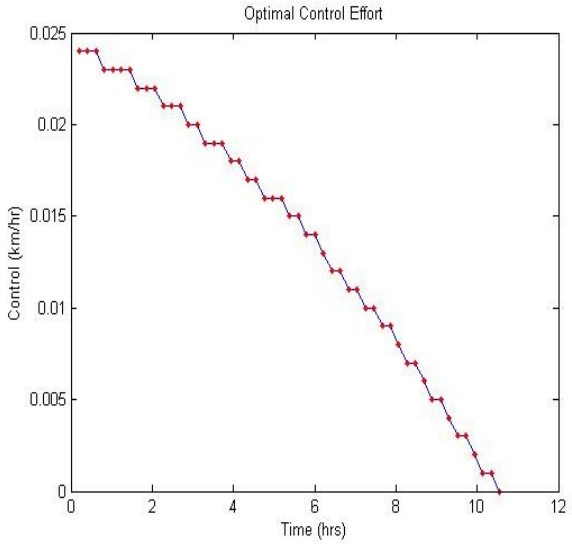
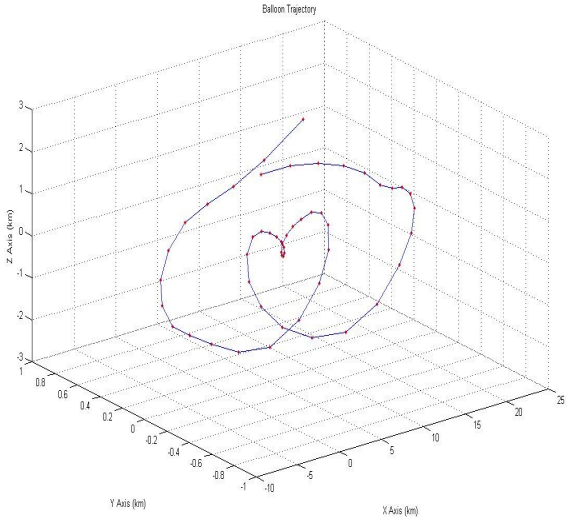


Figure 1: Balloon trajectory **Figure 2: Optimal control forces in the z direction**

However, with everything else remaining the same, except for the initial guess of the time step 'h', the results produced were quite different. In this case, the initial guess provided for 'h' was 1.4. The results of the optimization are shown in Figure 3 (trajectory) and Figure 4 (control forces). The optimal 'h' output was now 8.75 – considerably higher than the previous output of 'h'-0.207!



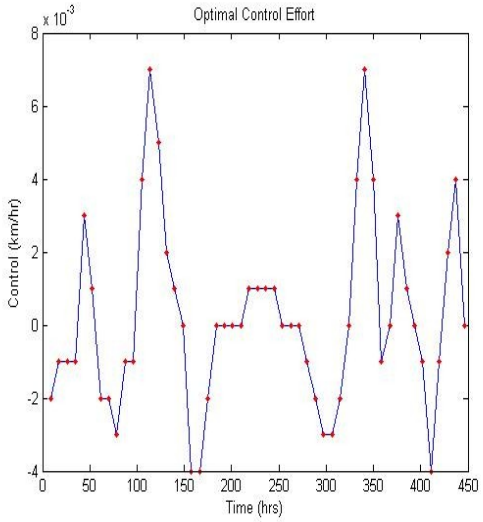


Figure 3: Balloon trajectory **Figure 4: Optimal control forces in the z direction**

The sum of forces(which is the function being minimized) is much smaller in the first case as opposed to the second. Thus, the first case is a more optimal solution as compared to the second one.

The difference in solutions occurs due to the fact that fmincon finds a local minimum to output the solution, and this local minimum is found around the initial guess provided. These experiments demonstrate the importance of the initial guess.

Results using real wind fields

When using real wind fields the importance of the initial guess became even more apparent. With control in the z-direction only, DMOC largely found it hard to converge and reach an optimal solution with a straight line initial guess. With control along all three axes, the solution output was almost always a straight line, similar to the initial guess, and thus required large horizontal control forces.

To test the performance of the algorithm, a Lagrangian advection scheme (the 'Newman' code – see description in methods section) was used to find the trajectory of a passive tracer being blown about by the WRF wind field for five hours, representing a frictionless balloon with no control forces. This trajectory is shown in Figure 5. By choosing the end point as the desired end point of a DMOC run begun at the start point, it is known that the optimal result *should* be zero control forces at all times, with the optimal trajectory matching that produced by the Newman code (to within numerical differences between the solvers used in the two schemes). However, the results from DMOC with a **straight line initial guess** did not reflect such a solution. The solution output in this case was almost a straight line as in our other test cases. Considerable control forces were required according to the solution output by DMOC. To correct this, the initial guess was changed. The initial guess now given was the passive path that would be followed by the balloon without applying any control force. With

this initial guess, DMOC converged to a solution which required no control forces and with a trajectory very similar to the one output by the Newman code (see Figure 6).

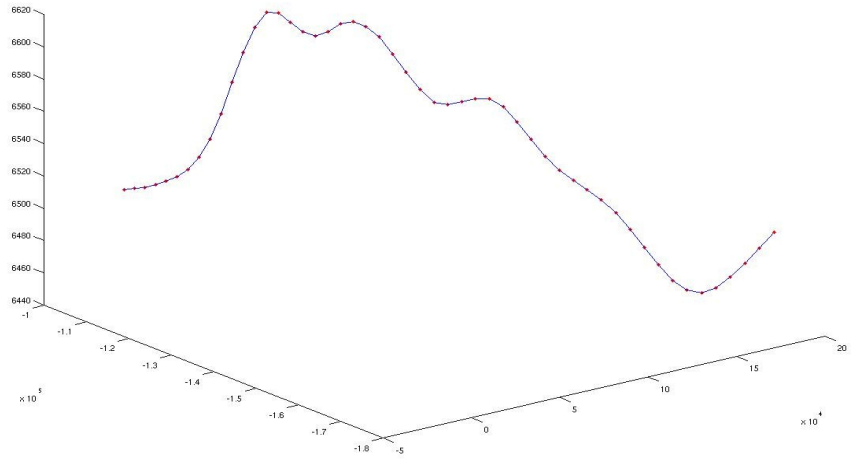


Figure 5: Trajectory followed by a passive tracer in 5 hours as produced by the Newman code.

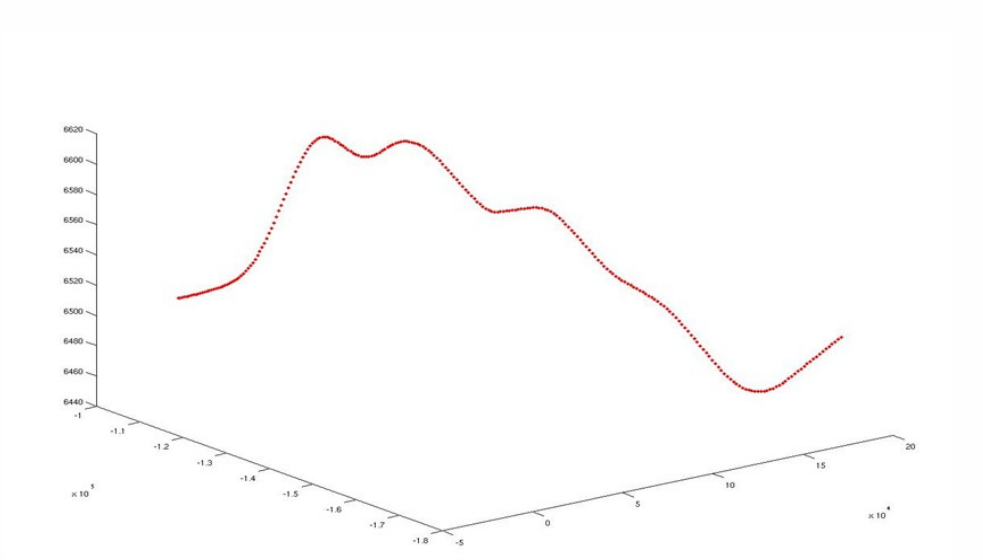


Figure 6: Optimal trajectory produced using the DMOC algorithm

As was found when using synthetic wind fields (see methods section) the results are thus very sensitive to the initial guess. But, using the Newman code to provide this guess would be useful only in cases where no/mild control forces are required and when the end point reached passively by the

balloon is sufficiently close to the desired end point. For most other situations, this method of initial guessing, by itself, might not be too useful.

Conclusions

Providing a good initial guess is thus the major problem in successfully using DMOC to find the optimal trajectory of a balloon or aerobot. An initial guess that is close to the actual solution is needed in order for DMOC to be able to converge to a sufficiently optimal solution. Along with a good method to generate the initial guess of the trajectory, a reliable method to generate an initial guess of the time step 'h' is also needed, since different guesses of 'h' can lead to drastically different solutions. Using Lagrangian advection code, which provides the trajectory of a frictionless, passive balloon, may provide a means of improving the initial guess by accounting for transport by the wind field. However, further work is required to adapt this to cases where the destination point is very far from that reached by a passive balloon (i.e., where considerable control forces must be required). With the development of suitable methods for initial guessing, this method using the DMOC algorithm seems to be promising for autonomous path prediction.

Future Work

To overcome problems arising due to a poor initial guess, an approach combining the capabilities of the Newman code and the DMOC algorithm can be used[10]. As mentioned earlier, Newman code has the ability to place passive tracers at user specified points and track the path followed by them in the wind fields input. This can be used to generate an initial guess for DMOC.

The time step 'h' can be fixed at about 5 – 10 minutes (or some other practical time step can be chosen). First, in MATLAB, the point (say, point A) reached passively by the balloon in 'h' time is noted. Now, a multitude of about 25 - 50 tracers can be started at different altitudes, on either side of point A (a total of 50 – 100 tracers), representing vertical control. These will represent the various points reached if quantized control forces (in the z direction) were applied by the balloon at the start point.

Using Newman code, it is seen where these tracers land up after a long period of time and which tracer got closest to the final destination. The start point of that particular tracer is then used as the second point in the trajectory. This process is repeated to build up the trajectory point by point, until the final destination (or a nearby region around it) is reached. This is how the initial guess can be setup. This would be much more accurate than a straight line initial guess and much closer to the actual solution. This would also avoid the need to provide an initial guess for 'h'.

Once the current model is successful with the DMOC algorithm, the model including drag forces can be incorporated to have a more accurate model of the balloon. If the balloon's specifications (such as its drag constant, terminal velocity, maximum control force that can be exerted, etc.) are known, they can be implemented as well. In the coming weeks, I shall work in these directions in an attempt to successfully predict optimal trajectories using the DMOC algorithm.

Methods

DMOC

As mentioned before, the DMOC algorithm is implemented using MATLAB's fmincon function (which solves for a local minimum around the initial guess)[9]. In the balloon model currently implemented, the balloon is considered as a point mass. Drag forces have been excluded in the model being tested with - it can be added once the model is perfected.

Before using fmincon, three things need to be setup –

1) An initial guess for the trajectory
2) The constraint equations
3) An objective function

Initial Guess - For the initial guess, a straight line was provided starting from the given initial point at the given initial velocity and ending at the given final position. The discretization points for the initial guess were placed at uniform distances along this straight line. The velocity along each of the points was calculated using finite differences of the location points, i.e. velocity(k) = (position(k+1)-position(k))/h, where the velocity at point k is found and h is the time step needed for moving from one point of the trajectory to the next. As an initial guess for control forces, finite differencing of the velocity at trajectory points was used. As mentioned earlier, the initial guess needs to be improved. As an improvement for trajectories requiring minimal control forces and where the end point reached passively is close to the desired end point, an initial guess that involves the balloon going passively with the wind can be used. But for most cases, a better initial guess needs to be implemented.

Constraint Equations –

Linear Constraints - The linear constraint equations were setup to ensure that the given optimized trajectory satisfies the equations of motion. Constraints on the discrete positions (along the trajectory), velocities and forces were setup. Drag forces were not included in the model used for testing (A model including drag forces was setup, but hasn't been robustly tested yet). The constraints were setup as follows –

- At a point along the trajectory, the balloon velocity (V) was the velocity provided by the control force (v) plus the wind velocity (uvw), i.e $V = v + uvw$
- The acceleration provided by the control force (f) is the simple finite differencing of the velocity provided by the control force (v), i.e. $f = \Delta v / \Delta time$
- The control forces in the x and y directions are zero, i.e. control force in only the z direction was given, i.e. $f(x) = f(y) = 0$

Non Linear Constraints – The total time taken to get from the initial point to the final point is also optimized by optimizing the time step ('h') between two consecutive points on the optimal trajectory. The constraints setup on the time step are that it must be between 0 and 10 hours(10 was chosen for practical purposes, since the number of discretization points along the trajectory chosen was 50, so for medium scale travel distances of about 1000 – 1500 km, with a time step of 10 (hours), the total time taken would be 500 hours. For practical purposes, we wouldn't want such distances to be traversed in more than 3 weeks). Later on, practical constraints on the maximum velocity that the balloon can travel at, the maximum control force that can be applied, etc. can be incorporated once specifications of the balloon's abilities are known.

The Objective Function – The objective function is the function that fmincon (the MATLAB function), shall minimize. While navigating on Titan, the fuel supply is limited. Thus, the primary objective is to minimize fuel consumption by the balloon. This can be done indirectly by minimizing the control force that the balloon needs to exert. The least squares method (applied to the control forces exerted) was used to provide the objective function. The control force is directly minimized in the first (preliminary) model of the aerobot, whereas in a later version (which can be implemented once the first model is perfected), the objective function involves the change in the density of gas inside the balloon (which is actually the source of control force in the z direction).

If the time of travel needs to be minimized, the time step 'h' can be minimized in the objective function. Alternatively, a combination of the time step 'h' and the total control forces can be incorporated into the objective function, depending on the factors most important to minimize.

The Weather Research Forecasting (WRF) Model

The WRF Model was used to generate the wind fields in the desired area (an area around the Mojave Desert was used for testing) at a desired time (wind fields from July 5, 2005 were used). The WRF Model is a mesocale numerical weather prediction system designed for operational forecasting and atmospheric research[11]. It features multiple dynamical cores, a 3-dimensional variational (3DVAR) data assimilation system, and a software architecture allowing for parallel computation and system extensibility. WRF is suitable for applications across scales ranging from meters to thousands of kilometers. The WRF model provides an operational forecasting model that is flexible and efficient computationally, while offering advances in physics, numerics, and data assimilation[12].

The WRF Model (compiled for real data cases) is run via the WRF Pre-processing System (WPS) first and then WRF is run.

WPS - This program is used for 1) defining simulation domains; 2) interpolating terrestrial data (such as terrain, landuse, and soil types) to the simulation domain; and 3) degribbing and interpolating meteorological data from another model to this simulation domain.

Each of the WPS programs reads parameters from a common namelist file. This namelist file has separate namelist records for each of the programs and a shared namelist record, which defines

parameters that are used by more than one WPS program. The WPS consists of three independent programs: *geogrid***,** *ungrib*, and *metgrid*[13]

Geogrid

Geogrid defines the simulation domains, and interpolate various terrestrial data sets to the model grids. The simulation domains are defined using information specified by the user in the "geogrid" namelist record of the WPS namelist file, namelist.wps. By default, and in addition to computing latitude and longitudes for every grid point, geogrid will interpolate soil categories, land use category, terrain height and other parameters to the model grids. The 10' resolution for the data sets was used. Variables in the geogrid section of the namelist file primarily define the size and location of all model domains, and where the static geographical data are found. The output files are written in netCDF format.

Ungrib

The ungrib program reads GRIB files, "degribs" the data, and writes the data in a simple format, called the intermediate format. The GRIB files contain time-varying meteorological fields and are typically from another regional or global model, such as NCEP's NAM or GFS models. The ungrib section of the namelist file contains two variables, which determine the output format written by ungrib and the name of the output files.

Metgrid

The metgrid program interpolates horizontally the intermediate-format meteorological data extracted by the ungrib program onto the simulation domains defined by the geogrid program. The interpolated metgrid output can then be ingested by the *real* program (which is run after WPS). The range of dates that will be interpolated by metgrid are defined in the "share" namelist record of the WPS namelist file, and date ranges are specified individually in the namelist for each simulation domain. Output from metgrid is written in the netCDF format. In the metgrid section of the namelist, the path to the ungribbed files needs to be provided, along with the path for the output files.

Now, after WPS is run, the **WRF** model is run. The WRF model uses the namelist.input file, in which the parameters (such as start time, date, end time, date, etc.) must match the namelist file used for WPS. The WRF model has 2 steps:

real.exe

This program interpolates the met_em* files *(generated by metgrid.exe)* vertically, creates boundary and initial condition files and does some consistency checks.

wrf.exe

Generates the model forecast**.**

Newman Code

Newman is research code for computing FTLE and extracting LCS. The Newman code can be used for parallel computations of Finite Time Liapunov Exponent (FTLE), LCS (Lagrangian Coherent Structures) ridge extractions, computations with N dimensions as well as for outputting drifter trajectories and velocity fields at any given time and position[14].

Newman code was used, in this project, to compute trajectories followed by passive tracers using the 'Trace Compute' feature of Newman[15]. The velocity fields were provided in numerical data files to Newman code to allow it to compute passive trajectories followed by tracers in the wind fields input.

Acknowledgements

I would like to thank Prof. Jerrold Marsden, Ms. Claire Newman, Philip Du Toit, Ms. Ashley Moore, Nick Heavens and Marin K for their valuable inputs and suggestions, without which, this project wouldn't have been possible.

References

[1] Detailed exploration of Titan with a Montgolfiere Aerobot, http://www.cosis.net/abstracts/EPSC2006/00200/EPSC2006-A-00200.pdf?PHPSESSID=ef23731eb16cddf728620a0b4c166033

[2] The sand seas of Titan : Discovery and implications for methane climatology and wind patterns, http://www.lpi.usra.edu/meetings/dunes2008/pdf/7034.pdf

[3] A look at Titan surface from the Cassini radar SAR and Radiometry data, http://www.lpi.usra.edu/meetings/lpsc2006/pdf/1497.pdf

[4]Titan SURF, http://gc.caltech.edu/public/Titan_SURF

[5] Autonomous Aerobots for Exploration of Titan and Venus, http://marstech.jpl.nasa.gov/publications/Jones_MontgolfiereBalloons_Inte.pdf

[6] Dynamics and Control Study of a Titan Aerobot from a System Design Perspective, http://pdf.aiaa.org/preview/CDReadyMGNC05_1089/PV2005_5975.pdf

[7]Autonomous Aerobots for Exploration of Titan and Venus, http://www-robotics.jpl.nasa.gov/tasks/showTask.cfm?FuseAction=ShowTask&TaskID=96&tdaID=3239

[8]Path Prediction for an Earth based Demonstration Flight, http://gc.caltech.edu/public/Titan_SURF/Path_Prediction_for_an_Earth-based_Demonstration_Balloon_Flight

[9] Discrete Mechanics and Optimal Control, http://www.cds.caltech.edu/~marsden/wiki/uploads/dmoc/home/JuMaOb04-IFAC.pdf

[10] Optimal Trajectory Generation in Ocean Flows, http://www.cds.caltech.edu/~marsden/bib/2005/12-InShMa2005/InShMa2005.pdf

[11]The Weather Research and Forecasting Model, http://www.wrf-model.org/index.php

[12] User's Guide for Advanced Research WRF (ARW) Modeling System Version 2, http://www.mmm.ucar.edu/wrf/users/docs/user_guide/users_guide_chap3.htm

[13]WRF/ARW Online Tutorial, http://www.mmm.ucar.edu/wrf/OnLineTutorial/index.htm

[14] Software – Newman, http://www.cds.caltech.edu/~pdutoit/software.php

[15]Newman User guide, http://www.cds.caltech.edu/~pdutoit/software/newman.pdf