University of Maryland at College Park
Dept. of Aerospace Engineering

ENAE 432: Aerospace Control Systems

Final Examination: Design Project

Part I

This project is not a homework, but rather the **approved, formal equivalent of a final examination in ENAE 432**. This is the first of two parts of the project, aimed to get you started thinking about the dynamics of the system and the requirements for your design. More detailed specifications for the final write-up will be contained in the second part of the project, which will follow in about one week.

As this is an examination, **no collaboration of any form is permitted. You must only work individually, and may not discuss your work with anyone else – whether in the class or not**. This means current and former TAs, current and former students, tutors, internet chat groups or message lists, etc are *unauthorized* sources of information for this project. Similarly, you may not reference any materials beyond those explicitly distributed to you in this semester's class or those available in the suggested texts listed in the syllabus. Any materials beyond this approved set, including any past year course materials or materials downloaded from the internet, are *unauthorized and prohibited* sources of information for this project.

Any evidence of collaboration, discussion, or use of unauthorized sources, for any aspect of this project will be immediately forwarded for judicial review with the attendent potential consequences. Note as well that "apathy or acquiescence in the presence of academic dishonesty is not a neutral act" (quoting from the UMD code of academic integrity). To witness academic dishonesty and remain silent is to be complicit and culpable in that dishonesty. I remind you again that this project is your *final examination* – while you will be working on it outside of class, it is in no way similar to a homework that you are free to discuss with others. You are in every way subject to the same rules governing an in-class examination, save that you may use computers and course materials to assist your effort.

The *only* permissible source of discussions about this project is with me personally, and the contents of those discussions may not be shared with others. However, **as this is a final exam**, I will answer only factual questions about the theory discussed in the class, clarifying questions about the subject or general objectives of the project, and practical details about using any provided project code. In particular, I will not give specific guidance on the details of your modeling, data interpretation, controller design, controller implementation, controller performance, etc. These are entirely up to you, and indeed are the primary basis for your final grade on the project.

"Sky Crane" Control System

For your project this semester, you will design part of an EDL ("Entry, Descent, and Landing") control system similar to that which was used for the "Sky Crane" terminal descent phase of the NASA missions which landed the rovers "Curiosity" and "Perserverance" on Mars. Let's see if you can be as successful as JPL's engineers were at this task!

To accomplish this, you will have to weigh a number of factors for your final design. In addition to the fundamental constraints and tradeoffs inherent in any feedback control system, you will also have to consider important details of the Martian environment, as well as the cost and complexity of the system needed to implement your proposed controller on the vehicle.

System Dynamic Model

We will use a very simplified model of the coupled lander-rover dynamics in the terminal descent stage after the tether has been deployed. The model assumes that the descent rate is being maintained as constant by a separate control system, and only models the horizontal translation of the lander and deflection angle of the tether. The tether is assumed to be constantly in tension.

So long as the tether angle and its rate of change are kept sufficiently small (no more than about ± 0.5 rads and ± 0.1 rad/sec respectively), a linear model for the system is

$$\begin{aligned} (M+m)\ddot{y}(t) + mL\ddot{\theta}(t) &= Ku(t) \\ mL\ddot{y}(t) + (I+mL^2)\ddot{\theta}(t) &= -mg_m L\theta(t) - b\dot{\theta}(t) \end{aligned}$$

In this model, $y(t)$ is the horizontal position of the center of mass of the lander (in meters), $\theta(t)$ is the angle (in rad) of the tether with respect to the vertical ($\theta = 0$ means the rover center of mass is directly below the lander CM), and $u(t)$ are the commands from the onboard computer to the rocket thrusters which provide translational forces. The constants M and m are the masses (in kg) of the lander and rover, respectively; I is the rotational inertia (kg-m^2) of the rover about its own cg; L is the length (in meters) of the tether; g_m is the acceleration (m/sec^2) due to gravity *at the surface of Mars*; b is a damping constant due to friction in the tether attachment points; and finally K is a composite constant reflecting the effective gain between the digital output of the computer (which throttles the engine) and the actual effective lateral thrust (in N) applied to the vehicle (which is $K * u(t)$). This constant depends on the engine efficiency, D/A conversion electronics, throttle design, etc.

For the project, the lander is descending at a constant rate of 1/3 m/sec, and the terminal descent phase is assumed to start when the rover wheels are 30m above the Martian surface. We suppose that at this moment, the onboard guidance system has determined that the system needs to move a certain distance to the right in order to avoid newly detected obstacles and set the lander down on flat terrain. The objective is to accomplish this maneuver within the 90 seconds remaining to touchdown, preventing the rover from swinging too violently, and ensuring that the rover touches down with its wheels nearly level and almost no horizontal velocity.

The guidance and control system on the lander measures the horizontal displacement $y(t)$, and nominally will update the control inputs $u(t)$, 10 times per second (10 Hz). If needed for your design faster electronics can "purchased", at the cost of penalizing your score, up to a maximum of a 50 Hz update rate. Details of the cost and scoring will be provided in Part II.

As the lander approaches the surface, the descent thrusters will begin to stir up the dust on the Martian surface. This will have the effect of making the lateral displacement measurements progressively noisier, with up to ± 20 cm inaccuracy. You do not have a lateral velocity sensor, so $\dot{y}(t)$ cannot be measured directly. Moreover, although control of the tether angle $\theta(t)$ will be of crucial importance for mission success, this angle also cannot be measured directly. The only indication that the rover might be swinging by an undesireable amount would be through its indirect effect on the horizontal motion $y(t)$ of the lander.

The dusty environment of the Martian surface doesn't just affect the sensors on the lander – it is in fact the entire reason for attempting to land the rover in this tethered fashion. If that dust is too violently swirling from the descent thrusters, the dust can be driven into the rover itself, damaging its mechanisms and science instruments. The tether length is nominally 7m, which should give sufficient clearance, but you can change that if desired between 2-10 m. Longer tethers will give a higher probability of the rover being unaffected by the dust, while shorter tethers may be easier to stabilize, but will have a greatly increased chance of inducing rover failure due to dust exposure. The probability of failure as a function of the tether length L will be given in Part II, and will act as a scaling factor on your overall final score.

Note from the dynamics above that the lander acceleration $\ddot{y}(t)$ drives the tether angle dynamics (this acceleration acts like an "input" to the $\theta(t)$ dynamics). **The strategy therefore is to control $y(t)$ in a steady, smooth fashion which minimizes its acceleration, thereby minimizing the impact of horizontal maneuvers on the tether angle $\theta(t)$. Sudden jerks in position, arising from high gain, high bandwidth feedback, will excite oscillations of the tethered rover and potentially destabilize the entire system. On the other hand, the controller must have sufficient bandwidth to actual move the rover clear of the obstacles before touchdown. Finding the right balance is the essential engineering challenge of this design.**

Although the general form of the Sky Crane dynamics are known, precise numerical values for the many of the physical constants are not. Nominally the lander will have mass $M = 900$ kg although there may be a $\pm 5\%$ variation based on the amount of fuel expended in the landing phases leading up to the tether deployment. The Martian gravitational acceleration is accurately known to at least one decimal place (look it up!). The remaining physical parameters – m and I which describe the mass properties of the rover, b which describes the frictional damping in the tether, and the constant K governing the applied forces – are all much more variable, and need to be determined via experiments with your personal Sky Crane model, as described below.

Finally, note that the lateral thrusters on the lander can provide no more than ± 600 N of thrust. That is, the force $Ku(t)$ applied to the lander will not exceed ± 600 regardless of the value of K or of $u(t)$. The thrusters will instead saturate at these limiting values whenever the computer commands would try to produce forces outside this range. This limitation may have an impact on the design of your control logic.

Gantry Crane Data

The Sky Crane dynamics above actually describe a *family* of possible models, one for every possible value of the uncertain numerical constants. Nominal ranges for these parameters are known to be $m = 900 \pm 150$, $b = 200 \pm 100$, $I = 750 \pm 100$. The parameter K lies either in the interval [0.5 100] or the interval [-100 -0.5]; values of K in the latter range would indicate that something in the electrical connections between the computer and thrusters is wired backwards. Rather than go back into the system and fix the connections (which might risk damage to delicate adjacent electronics), the problem can be simply corrected in software (i.e. your control algorithm!) for such cases, assuming it has been correctly identified prelaunch.

Every student in the class has her/his *own individual dynamic model* for the project. There is quite a range of variation in the different models, and no two are identical. Only by careful measurement and experimentation can you get a good approximate idea of the actual constants which govern the dynamics of your system. Parameter values for all models are in the ranges identified above.

In order to help identify the physical constants for your system, you have access to a gantry crane simulation of the Sky Crane configuration set up in a laboratory. In this system, a gantry crane moving (nearly) frictionlessly on a rail simulates the horizontal motion of the lander, and an exact replica of the rover is hung on a tether below it. The gantry has a mass identical to the lander ($M = 900$), and uses exactly the same control and electrical systems to generate the forces which move it. The resulting laboratory model then has exactly the same dynamics as the Sky Crane model above, although, since the lab is on Earth rather than Mars, you would use the usual g for Earth's gravitational acceleration in the model instead of the Martian acceleration g_m which appears in the Sky Crane model.

Since the crane is in the lab, you can experiment with it in various ways that go beyond the nominal Sky Crane configuration. For example, you may operate the system with the rover stowed securely within the lander. In this case the tether degree of freedom is eliminated from the dynamics, so that the above equations simplify to

$$(M + m)\ddot{y}(t) = Ku(t)$$

You may also do experiments with the gantry crane locked in place. In this case the translational degree of freedom is eliminated, and the hanging rover/tether acts as a simple pendulum:

$$(I + mL^2)\ddot{\theta}(t) = -mgL\theta(t) - b\dot{\theta}(t)$$

In this case, no input acts on the system, but you may release the rover from different initial angles and study the resulting oscillations.

In all three modes (Sky Crane mode, stowed lander mode, locked gantry mode) you may experiment with the tether played out to different lengths, thus changing the value of L used in each of the governing models. Any value of L between 2m and 7m can be used (note that you can use up to $L = 10$ in the actual mission, but the laboratory test equipment only allows for L to be at most 7).

By studying the responses of this system in its various modes, looking at the effects of different inputs $u(t)$, different initial tether angles $\theta(0)$, and/or different tether lengths L you will be able to uniquely identify all four uncertain parameters. Note that there are lots of possible experiments that could be done! However, only a few, carefully chosen, experiments are required to get the information needed.

Experiments with the gantry crane model are accomplished with the `gantrysim` Matlab function. This function will simulate the crane/rover dynamics in one of the three modes above, with values of L, $\theta(0)$, and $u(t)$ that you specify, and provide arrays with time histories of the resulting displacements $y(t)$ and angles $\theta(t)$ which result from your experiment; the angle measurements are reported in degrees. Note that **there is a small amount of measurement noise in the recorded position and angle values of each experiment: about ± 2 cm in positions and $\pm 2^\circ$ in the angles.** Note also that the crane is physically limited to $\pm$10m of total horizontal travel, and will stop (in translation) when it reaches these limits. (And if it repeatedly hits these stops with appreciable velocity the crane may break, so don't abuse this function!)

More complete documentation about the use of the `gantrysim` function can be found at the end of this document. An important point to remember is that the first input to this function *must* be the last four digits of your UID number. This causes the simulation to execute with the physical parameters specific to your personal Sky Crane model.

Finally, be aware that `gantrysim` (as well as the `landsim` function distributed later with Part II) is provided as a "p-code" file, with a .p extension. These are binary, compiled m-files which cannot be modified or examined. You cannot open them in Matlab's editor (or any other text editor); you can only execute these files as commands. They can be run from the Matlab command window or used in your own Matlab scripts just like ordinary m-files, assuming they are stored in your current working directory (i.e. the folder shown at the top of the Matlab command window).

Important: You *must* use for your design the Sky Crane model corresponding to your UID number. A large part of your grade on this project will be determined by running the controller you design with your specific model and examining the results. Using someone else's data (and/or control strategy) will almost certainly produce poor performance or instability in your vehicle, resulting in a very low grade for the project. If for any reason you cannot download the required files, or run them in Matlab, tell me as soon as possible so that I can resolve the situation quickly.

Part I: Getting Started

1. Find generically the transfer function $G(s)$ which relates $Y(s)$ to $U(s)$ symbolically in terms of the physical parameters K, L, etc. $G(s)$ should be a transfer function directly relating $Y(s)$ to $U(s)$; there should be no θ terms in this transfer function!
2. Use the `gantrysim` function to identify as specifically as possible the numerical form of your $G(s)$. Note that this data may not allow you to identify all parameters uniquely, but it will give you sufficient information to approximately identify the most significant features of your $G(s)$. Use the nominal parameter values as needed where the test data does not allow determination of unique values.
3. Generate a Bode plot for your estimated $G(s)$ model. Based on this, start to think about how you will need to design your controller to satisfy all the above requirements. Plan initially for 10Hz loop rates, and consider tracking accuracy, disturbance and noise rejection, etc constraints on the design of $H(s)$.
4. Consider also the effect of tether length L in your analysis, in particular the relative advantages or disadvantages of shorter ($L \approx 2-3$) vs longer ($L \approx 7-8$) tethers.

Using gantrysim

```
[tt,y,theta]=gantrysim(uid,mode,L,th0,tmax)

Runs the gantry simulation of Sky Crane system.

Inputs:
uid -- last 4 digits of your UID
mode -- 1 for a full sim, 2 for gantry locked, 3 for rover stowed
L -- desired tether length
th0 -- desired initial tether angle, in degrees
tmax -- desired duration of experiment

Outputs:
tt -- array of sample times at which data is collected
y -- array with values of gantry position at each time in array tt
theta -- array with values of tether angle at each time in array tt

To run this simulation you must also have an m-file called control.m
in your current directory (although this file is ignored in mode 2).
Gantrysim will use this file to define the input that is applied to
the system at each time step.  The control.m function must have an
input/output format so that u=control(t,y) will assign the desired
value for input u given the current values of t and y.  This is similar
to the code you will need to write for your controller in Part II,
except there the first input to your control function will be yd
instead of t.

Two simple example control.m files:

--------------------------
Example 1:  Apply a constant input u=10
--------------------------
function u=control(t,y)
u = 10;
end
--------------------------

--------------------------
Example 2: Apply a sinusoidal input, freq 2, amplitude 3
--------------------------
function u=control(t,y)
u = 3*sin(2*t);
end
--------------------------

**********************************************************************
WARNING:  It is a *really* bad idea to do experiments with gantrysim
that result in the crane repeatedly banging into the stops. . .
**********************************************************************
```