GeorgiaInstitute
of Technology

Homework Assignment 7
ODE Solving – Simulating a Car
Due: Friday, April 16, 2010 at 12noon

Description and Outcomes

In this assignment, you will develop a simulation of a car. Compared to the set of differential equations that we solved in class, this model is an order of magnitude more complex. Yet, you will see that the solution mechanism remains essentially the same.

The learning objectives of this assignment are:

- to improve your understanding of solving ODEs numerically
- to learn about simulating physical systems in Matlab
- to learn about the different ODE solvers in Matlab
- to learn how to debug your code
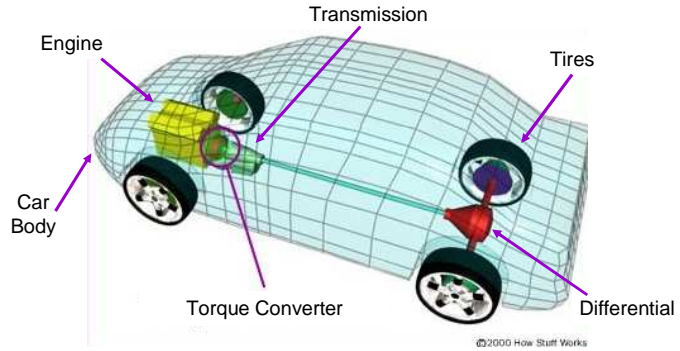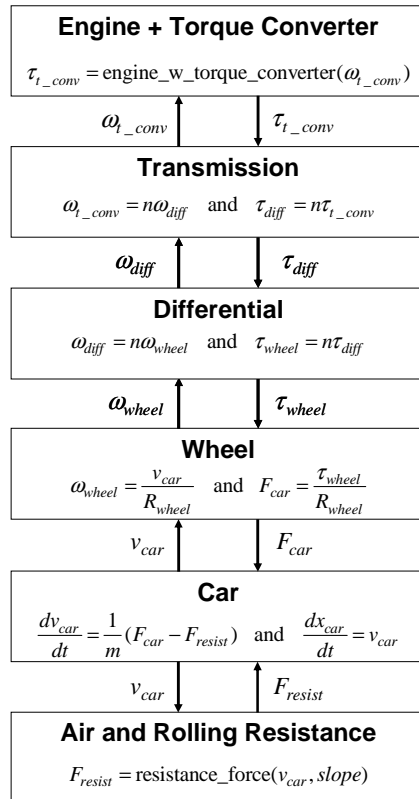
Background

The last two homework assignments of this semester (HW7 and HW8) are integrative in nature. That is, you will build on several previous homework assignments and combine portions of the previous solutions into the solutions of HW7 and HW8. So, what may seem like a lot of work is really more cutting and pasting than writing new code. You will also notice that many of the numerical methods that we have studied throughout the semester are combined in the last two assignments. You will be asked to identify these methods in Task 3. In HW8, the simulation developed in this assignment will be further extended and optimization will be applied to the result. This will bring together most of the numerical algorithms that we have studied this semester!

In this assignment, the focus is on simulating the dynamic behavior of a car: How does the position and velocity of the car change as a function of time? Specifically, you will evaluate the car's performance in a drag race – how fast can you finish a quarter-mile race? We rely on a second order ordinary differential equation that defines the car's motion; there does not exist a closed-form analytical solution for the differential equation. Instead, you need to compute the derivatives of the position and velocity numerically by combining all the models for the individual drivetrain components as is shown in the figure on the next page.

One should consider the following points when developing the model:

- The main block in the graph below is the "car" block — it defines the differential equations as a function of the car and resistance forces. All the other blocks are needed to compute these car and resistance forces. Follow the arrows in the graph to implement the model in a systematic fashion.

- When simulating the car, you need to consider when to shift gears. This is done based on a shift table. For our purposes, the shift table can be defined simply as a vector with three elements: *v12*, *v23*, and *v34* — the velocities in m/s at which the car shifts from first to second, second to third and third to fourth gear, respectively. To determine which gear to use, you should compare the current velocity of the car to these shift velocities. To achieve maximum performance, you should pick the shift velocities computed in HW2. Actually, due to the small loss of power in the torque converter, it is better to shift slightly earlier, namely, at 17, 30 and 46 m/s.

- In HW4, you implemented a function called *resistance_force*. This function can be reused here.

Engine + Torque Converter
$\tau_{t\_conv} = \text{engine\_w\_torque\_converter}(\omega_{t\_conv})$

ω_{t_conv} τ_{t_conv}

Transmission
$\omega_{t\_conv} = n\omega_{diff}$ and $\tau_{diff} = n\tau_{t\_conv}$

ω_{diff} τ_{diff}

Differential
$\omega_{diff} = n\omega_{wheel}$ and $\tau_{wheel} = n\tau_{diff}$

ω_{wheel} τ_{wheel}

Wheel
$\omega_{wheel} = \dfrac{v_{car}}{R_{wheel}}$ and $F_{car} = \dfrac{\tau_{wheel}}{R_{wheel}}$

v_{car} F_{car}

Car
$\dfrac{dv_{car}}{dt} = \dfrac{1}{m}(F_{car} - F_{resist})$ and $\dfrac{dx_{car}}{dt} = v_{car}$

v_{car} F_{resist}

Air and Rolling Resistance
$F_{resist} = \text{resistance\_force}(v_{car}, slope)$

- To avoid stalling the engine, a torque converter must be considered in the simulation. We have provided a function called *engine_w_torque_converter* that builds on the engine model used in HW2. The model requires the following three Matlab functions that are provided on the course web-site: *engine_w_torque_converter.m*, *torque_converter.m*, and *engine_model_CJP.m*.

- The car model in this assignment requires several parameters. We use the same values as in previous assignments:

```
mass = 1650;                            % [kg]
wheel_radius = 0.332;                   % [m]
differential_ratio = 4.266;             % []
gear_ratios = [2.77 1.54 1.00 0.69];    % []
shift_table = [17 30 46];               % [m/s]
drag_coefficient = 0.32;                % []
frontal_area = 2.2;                     % [m2]
rolling_coefficient = 0.009;            % []
air_density = 1.29;                     % [kg/m3]
slope = 0;                              % [%]
```

Tasks
Task 1: Write the function car_model_XYZ (where XYZ are your initials).

- The function car_model_XYZ should implement the differential equation describing the motion of a car as out-lined in the figure above. Note that it will call several other functions that are necessary to describe the dynamics of the entire drivetrain. Three other functions that you will need are provided on the course web-site: *engine_w_torque_converter*,

torque_converter, and *engine_model_CJP*. Remember: try to reuse as much of the code that you have written for the previous assignments, and don't hesitate to use the solutions that are available on the course web-site.

- The function car_model_XYZ should use the appropriate inputs and output so that it can be solved using the Matlab solver *ode45*. Please, read about *ode45* in Matlab Help to understand how this function works.
  http://www.mathworks.com/access/helpdesk/help/techdoc/ref/ode45.shtml

- The state vector, *y*, should include the car's position and velocity as follows:
  y = [velocity; position].

Task 2: Write a script car_simulation_XYZ (where XYZ are your initials).

- Write a script that performs the car simulation and produces two plots: one of the velocity (in km/h) as a function of time and one of the position (in m) as a function of time (Make sure to include plot labels, title, etc.). The script should also report on the following metrics (make sure to include this output in your report):

  o How long does it take to reach the finish line?
  o How fast does the car go at the finish line?
  o What is the car's average speed?

- The simulation should reflect the car's motion while participating in a quarter-mile (402.3m) drag race, starting the car at a position and velocity of zero and running at full throttle.

- To solve the set of differential equations defined in *car_model,* you should use the Matlab function *ode45*:

```
% simulate the car
options = odeset('Events',@end_of_race_event);
[t,y] = ode45(@car_model,[0,50],[0;0],options,car_data,race_data);
```

- The *end_of_race_event* function is provided on the course web-site. It stops the simulation when the car reaches the end of the quarter-mile race trajectory.

Task 3: Analyze the structure of your simulation code

Include in your report an analysis of the structure of the simulation. Look through all the functions you wrote and downloaded from the course web-site. Identify which functions call which other functions and also which numerical methods are being applied. You will notice how in this simple car model almost all the numerical methods that we have studied this semester are applied. To make your analysis easy to read (and grade), write it as bulleted list, each bullet stating a simple fact about your model. For instance:

- The car_simulation script uses the ODE solvers *ode45* to solve the system of ODEs defined by car_model.

Report
Turn in a brief report in MS Word containing the following:
1. A copy of all your Matlab code.
2. A copy of all your figures you generated.
3. Your answer to task 3.
4. A statement of collaboration (see below) – include this even if you did not collaborate with anybody

Evaluation Criteria
A detailed grade sheet for this assignment is provided as a separate file on the course web-site. Print out this sheet, fill in your name, and staple it to the front of your submission.

Submission

1. **Hardcopy in class:** At the *beginning* of class, hand in a hardcopy that includes the grade sheet and your report. Make sure you staple everything together so that we don't lose anything. (no staple = incorrect submission)
2. **T-square:** save all your files in a zip-file called HW7.FamilyName.FirstName.zip (replace FamilyName and FirstName with YOUR names) and submit this file in the Assignments section of T-square. The zip-file should contain: your Matlab functions and script, and your report. The deadline for electronic submission is *at 12noon*.

Late Submission
Remember that the deadline for this assignment will be strictly enforced. After the deadline has passed you will receive a late-penalty. Don't wait until the last minute to get started! We repeat here the policy that is included in the syllabus:

Late Submission
T-square is set up such that late submissions are accepted until 48 hours after the due date. We will accept late homework but with the following penalties. Submit before 24 hours late and receive 20% off of your grade. Submit before 48 hours late and receive 40% off. For submission more than 48 hours late, no more partial credit will be awarded. If you are submitting late, bring a hardcopy to the office of your instructor. If you plan on submitting late, a quick e-mail by the deadline would be appreciated so that we can make appropriate plans for grading your assignment.

Collaboration
We would like to re-emphasize the policy on collaboration. Collaboration is encouraged. Discussing the assignments with your peers will help you to develop a deeper understanding of the material. However, "discussing the assignment" does not mean solving it together; it does not mean asking your friend to debug your code for you. I encourage you to discuss how to approach the problem, which Matlab functions to use, or how to interpret the results, but I do expect each student to turn in a report and Matlab functions that reflect the student's individual work. Do not copy code from another student. Do not copy parts of other electronic documents. In general, an activity is acceptable if it promotes learning by you and your peers. For example, you learn from discussing alternate solution approaches with your friend, but you don't learn from blindly copying your friend's code. To avoid any confusion, each homework solution should explicitly identify the students with whom you collaborated and what the extent of the collaboration was. Any copying on homework and/or exams will be dealt with severely and reported to the Dean of Students – No exceptions. If you have questions about this collaboration policy, do not hesitate to ask your instructor.