Gateway Computing: MATLAB

Project D: Traffic Simulation

David R. B. Kraemer

Planning Due Friday, 15 Apr 9pm; Project Due Friday 29 Apr by 9pm

The Problem

Despite advances in self-driving cars, humans still control most of the vehicles on our roads. Those roads have too many cars. In the USA people spend an average of about 100 hours per year stuck in traffic. In Bogota Colombia the average is about 300 hours per year!
https://inrix.com/press-releases/scorecard-2018-us/

"If we are to avoid traffic congestion becoming a further drain on our economy, we must invest in intelligent transportation systems to tackle our mobility challenges."

Trevor Reed, INRIX
https://inrix.com/press-releases/scorecard-2018-us/

One interesting facet of traffic dynamics is that once traffic speed and density reach a certain threshold, the traffic becomes unstable. Any small disturbance leads to "rolling backups" that seem to form spontaneously. The flow of traffic acts like a compressible fluid, viewed from a distance. The backup travels as a wave. This is directly analogous to compression waves in air ("sound"). The speed of the wave depends on factors like traffic speed, traffic density, and braking intensity. If drivers are distracted or fail to scan ahead of the car immediately in front of them, rolling backups are more easily initiated.

Traffic backups can be mitigated by traffic density management (control lights on on-ramps), speed management, and improved driving control algorithms (both human control and autonomous control).

Background 1: Simulation

A. Initial-Value Problem
The problem is essentially an Initial-Value Problem, consisting of coupled differential equations (the accelerations of each car, dependent on the interactions with the other cars), and the initial conditions.

We can solve these equations using the familiar Euler's method for each car:

- Calculate the acceleration of a car;
- Estimate the new velocity from the old velocity plus the acceleration times the time step;
- Estimate the new position from the old position plus the acceleration times the time step;

B. Object-Oriented Programming in MATLAB

We will implement the simulation using objects to model the behavior of the cars, as well as an object for the track.

When an object modifies another object, it is usually best to use the “pass by reference” method (handle objects) instead of the default “pass by value”.

C. Animation

One way to create an animation is to draw each frame of the animation on a figure.

It is important to note that the time step of the simulation should be independent of the frame rate of the animation. It is common that the simulation time step required for an accurate simulation is far smaller than the time between frames required for a smooth animation. Set the simulation time step and the animation frame rate independent of each other. Check that they are multiples of each other. Use the modulo operator to only draw a new frame when necessary.

MATLAB’s `VideoWriter` function can be used to save the animation to a file that can be played back later. The process is:

- Create a video object with the `VideoWriter` function;
- `set` the `frameRate` in frames per second (20fps is a good compromise between smoothness and file size);
- `open` the movie file;
- When the simulation has reached a time step when a frame should be drawn,
  - Update the position of the graphics objects (the cars);
  - Use the `getframe` command to get a frame of the animation, and use the `writeVideo` function to add that to the movie;
- At the end of the simulation, `close` the movie file.

The Project Assignment

Part I: Planning

You are required to plan your approach to this project and submit this plan via Blackboard. This may be done as a group of up to 4 people. If you are working on the plan as a group make sure you communicate your group members to the course assistant during your planning class.

An example plan for a generic coding problem is provided on the Blackboard website to give you an idea of what we are looking for. The idea of submitting your plan is to take the time to think through the different parts of the project, so that you have a roadmap for your work. Note that there are four parts of the work that should be addressed in your plan:

- Designing: What is the design for the solution? What are the critical parts of the problem? What are the inputs and outputs? If there is graphical output, what will this output look like? How will parts of your solution utilize or connect to other parts?
- Coding: What control-structures, iteration techniques, built-in functions or other programming techniques will you need to use? Diagram flow charts of some of your subroutines.
- Testing: What are some example cases you would want to test to make sure the subroutines work along the way? What are some example cases you could think of to test the final program?
- Debugging: How will you check for bugs, particularly those that do not result in syntax errors? Are their parts that you anticipate will create problems due to their complexity? Could these be broken down into simpler pieces that could be debugged separately?

Make sure your plan references the project description and identifies the most relevant aspects of the project. Your plan will be judged according to how well thought-out it is. We will provide feedback on this plan for you to use. To communicate your plan, feel free to use regular paper, a whiteboard, sticky-notes or any medium of choice. Submit these as a clear photograph or scan and assemble them to produce a good quality and reasonably sized single PDF file that you can submit via Blackboard.

Part II: Coding and Testing

Your assignment is to write a computer program using object-oriented programming techniques that will simulate the flow of traffic around a circular track.

A. Single Car on a Track

First, make a simple simulation of a single car that runs around the track. Make a `Car` class, a `Track` class, and a `trafficSimulation` script.

Let the simulation run until it seems to reach a steady state.

Suggestions:

Let's assume that the car is a sphere. No, seriously! This enables us to draw the car as a circle. Collision detection will use a circular boundary as well.

The `Car` class should have these properties:

- `theta`, the angle measured from the positive horizontal axis, in radians
- `vel`, current speed of the car in m/s
- `D`, the diameter of the car in m
- `maxAccel`, the maximum acceleration of the car in m/s^2; use 9 m/s^2
- `maxDecel`, the maximum deceleration of the car in m/s^2; use 1.5 times the max acceleration here;
- `shape`, the MATLAB graphics object used to draw the "car"

The `Car` class should have these methods:

- Its constructor method, of course;
- `move`, to calculate the updated acceleration, velocity, and position of the car at the next time step;
- `collisionCheck`, to check for a collision with the car in front

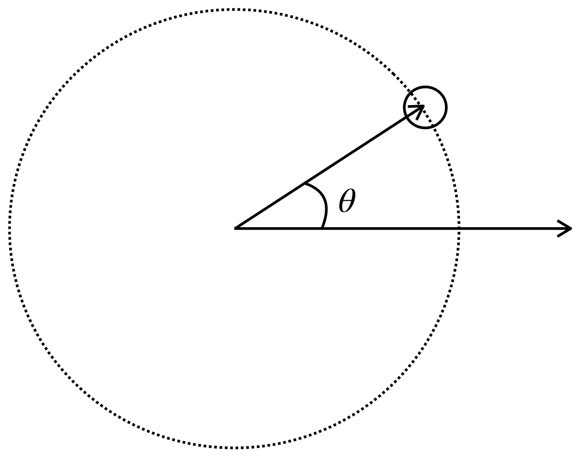

Figure 1. Car position on the track.

The car's shape should be a circle. This can be done by making a `rectangle` with 100% `Curvature`.

The car should accelerate at its maximum rate until it gets to the speed limit, then stay at that speed.

The `Track` class should have these properties:

- `diameter`, the circular track diameter in m (measured across the middle of the road)
- `speedLimit` in m/s (enter it in mph or km/h and convert in your code)
- `shape`, the MATLAB graphics object used to draw the track

The `Track` class should have these methods:

- Its constructor method, of course.

(Note that a Class without methods could be implemented more simply as a structure. For this simulation, keep the `Track` as a Class. This would enable further extensions using methods, such as creating obstructions on the road.)

The track's shape should be a circle. This can be done by making a `rectangle` with 100% `Curvature`. Actually, make the track have an inner and outer boundary circle. The inner circle should be 4 car diameters smaller than the track diameter, and the outer circle should be 4 car diameters larger than the track diameter. Note that this implies that the `Track` object needs to know the size of a car, so pass a `Car` object to the Track's constructor.

B. Multiple Cars on a Track

Now extend your simulation to allow multiple cars on the track. Try out five, and let the simulation run until it seems to reach a steady state.

Modify your `Car` class (suggestion: save different versions!) so that the cars interact with each other:

- If the car is at least 20 car diameters away from the car in front of it, then it should accelerate;
- If the car's speed is less than the car in front of it, and if it is at least 4 car diameters away, then it should accelerate;
- If the car's speed is greater than the car in front of it, and if it is less than 14 car diameters away, then it should brake;

Modify the car's shape so that its `FaceColor` is green when accelerating, yellow when coasting, and red when it is braking.

For this simulation, let's assume that the drivers are lead-footed; when they accelerate, they floor it. When the brake, they panic-brake.

C. Create a Movie File

Now add the instructions to create a movie file from the animation. See Background 1 C above.

Part III: Application

Test out your simulation using the following conditions:

- Track diameter is 200m
- Speed limit is 65mph
- 15 cars, starting from rest

Then try the simulation with 16 cars.

Finally, find a condition that allows 16 cars to travel more smoothly. Modify your car's control algorithm; change the traffic speed; get creative!

Your final submission should include

- Your Class m-files
- Your simulation script m-file
- Three tests of your Car class, from the command line, with given results compared to expected results. At least two of the tests should examine the move method;
- Three videos, uploaded to YouTube or a similar service. (Use "pass by reference": include links in your report; do not submit the whole video files.
  - 15 cars on the track
  - 16 cars on the track
  - An improved 16-car simulation.
- A brief discussion of the Applications, and how the three scenarios differ.

All submitted via the Blackboard site.