Jihye Lee
Spring 2024

(a) Change the initial velocities of vehicles to v = (1:n)'. Explain what this change represents.

- Initially, v = rand(size(q)) generates a vector of the same size as q, where q specifies the positions of the vehicles. Here, v becomes a column vector containing velocities corresponding to each vehicle on the road. In this instance, with 22 vehicles, each vehicle's velocity is set randomly between 0 and 1 m/s. Consequently, at the first time step, all vehicles have velocities less than 1 m/s, rapidly accelerating to an average of approximately 9 m/s. However, altering it to v = (1:n)', while still producing a column vector of 22 vehicle velocities, assigns each vehicle an initial velocity based on its order on the road. Hence, the first vehicle begins with a velocity of 1 m/s, the second with 2 m/s, and so forth, until the 22nd vehicle starts with a velocity of 22 m/s. This arrangement results in faster vehicles rapidly closing in on the vehicle ahead, as they all travel in a circular path, with the slowest vehicle positioned in front of the faster ones. Consequently, faster vehicles must decelerate swiftly to avoid collisions. As depicted in the figure on the right, the dots quickly decrease to low velocities before accelerating again as they exit the traffic congestion. On the left figure, we observe cars accumulating into a line of sluggish traffic in a section of the circle before resuming acceleration.

(b) Now change the time step dt to 0.1, and the number of computer steps per plotting event np to 1. (…) Running the code should produce a weird behavior. Explain what happens and why.

- Changing the time step size, dt, from 1 s to 0.1 s, and adjusting the number of compute steps per plotting event, np, from 10 to 1, results in the same amount of time passing between plotting events (as 1s * 10 = 10 s and 0.1 s * 1 = 0.1 s). However, the number of time steps occurring decreases from 10 to 1. This reduction in time steps leads to decreased accuracy due to error, as the method employed for evaluating the forward step is the Euler Method, known to be a first-order method. The weird behavior observed includes cars appearing to collide or pass through one another, accelerating beyond the plotted area, or coming to a complete stop. Such behavior is abnormal and shouldn't occur. However, since we are utilizing a first-order method (Euler's Method) to compute positions, reducing the number of time steps increases error. Consequently, the behavior witnessed in the plots does not accurately reflect the true behavior of the function f at these time points.

(c) For the previous code, change the time stepping from Euler’s method to Range-Kutta 4. (…) explain why the new time stepping fixes the problem encounter in (b).

- This adjustment successfully rectified the anomalous behavior observed in part (b), which stemmed from the utilization of Euler's Method rather than a higher-order method. Euler's Method introduced significant error, leading to plotting outcomes diverging markedly from the true behavior dictated by the function. The global error of Euler's method scales as O(dt), implying that with insufficient time steps, accuracy diminishes, impeding the resolution of system dynamics.

Contrastingly, Runge-Kutta 4, being a fourth-order method with a global error of O(dt^4), exhibits substantially enhanced accuracy with fewer steps. This superiority stems from its approach of determining slopes at four points within each time step interval and averaging them to compute subsequent points. In contrast, Euler's Method solely considers the tangent slope information at the left side of the interval, approximating behavior at the next point based on this singular piece of data.

Furthermore, it's crucial to recognize that over numerous steps, error accumulates more rapidly with Euler's Method, leading to a greater divergence from the actual function value compared to Runge-Kutta 4. Consequently, Runge-Kutta 4 yields a significantly closer approximation to the true function, f, compared to Euler's Method, especially given the substantial variations across the function's domain. Euler's Method struggles to accurately capture this behavior unless the time step size is drastically reduced.

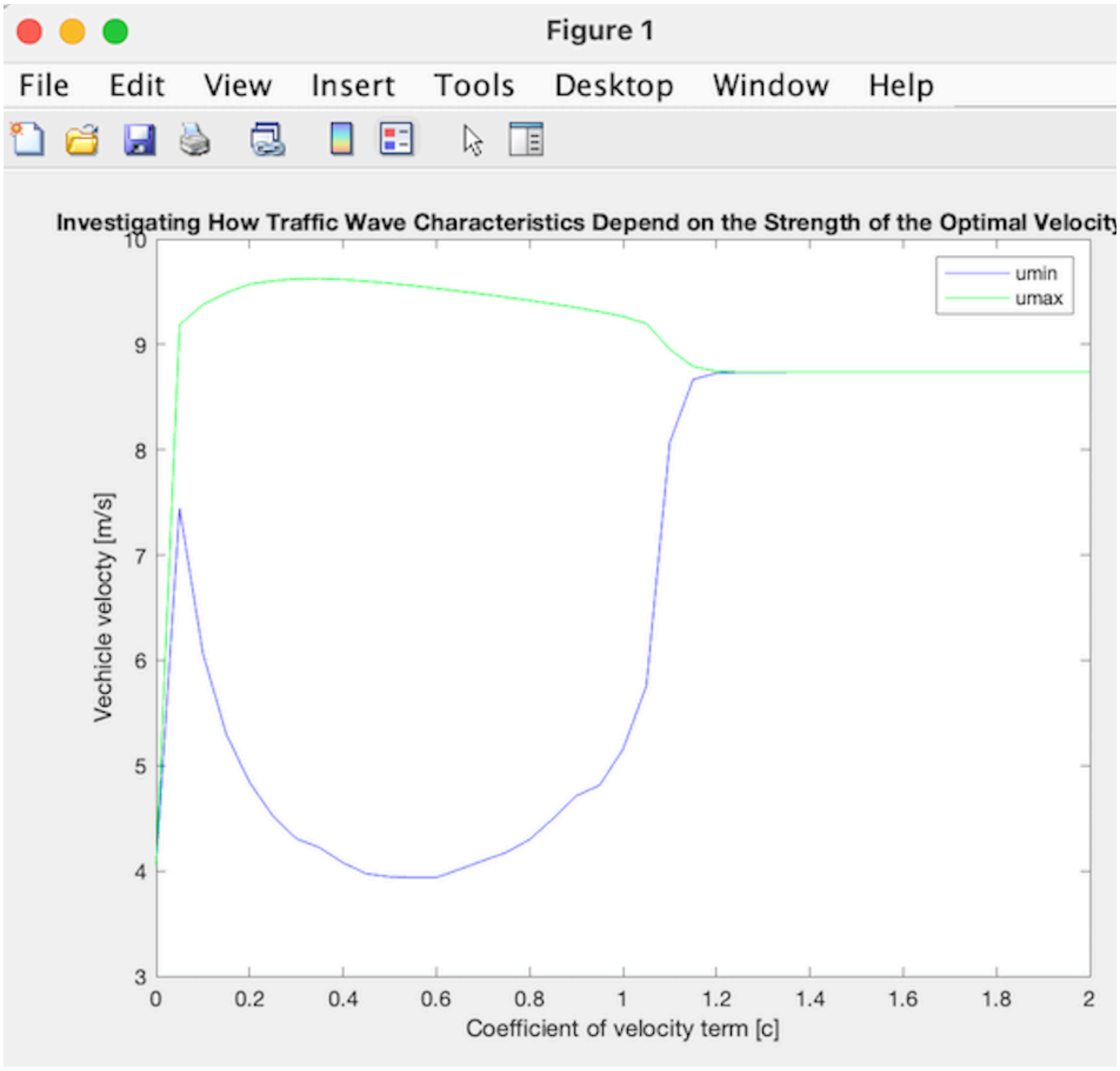

Figure 1: Problem d

(d) Change your code from part (c) to have dt = 1e-2 and np = 10 again, but keep the Runge-Kutta 4 time stepping. With this code, conduct a parameter study to investigate how the characteristics of the traffic wave depend on the strength of the optimal velocity term. Specifically, vary the coefficient in front of the term from 0 to 2 in steps of 0.05. Describe your observations and explain the results.

- When ‘c’, the coefficient, equals 0, the optimal velocity term (the second term in equation 'f') holds no influence, aligning with the behavior observed in the follow-the-leader model, which lacked an optimal velocity term entirely. Consequently, both the maximum and minimum final velocities ('umax' and 'umin' respectively) coincide, as depicted in the plot where 'umin' and 'umax' are approximately 4 m/s. This suggests stability when ‘c’ equals 0, mirroring the behavior of the follow-the-leader model with identical parameters of 'dt' and 'np'. In this scenario, cars disregard optimal velocity, basing their behavior solely on that of the leading car.

Upon increasing ‘c’ from 0 to 0.05, both 'umin' and 'umax' experience a significant increase to approximately 7.5 m/s and 9.2 m/s respectively. This surge is attributed to the introduction of ‘c’ as a nonzero coefficient, thereby influencing the vehicles' velocities through the optimal velocity term. Subsequently, between ‘c’ values of approximately 0.05 and 1.25, 'umin' exhibits a pronounced concave-up shape, indicative of a mix of very fast and very slow vehicles concurrently occupying the road. This discrepancy in final values for 'umin' and 'umax' at 'tf' signals the presence of a traffic wave, rendering the model unstable.

Around ‘c’ = 1.25, 'umin' and 'umax' values begin to converge, settling at approximately 8.7 m/s, a trend that persists for 'a' values greater than 1.25. Consequently, the model stabilizes again, demonstrating that for ‘c’ values exceeding 1.25, further increments in ‘c’ do not alter the model's behavior.

When ‘c’ exceeds 1.25, we notice that the minimum and maximum velocities ('umin' and 'umax') start to converge and stabilize around a certain value, approximately 8.7 m/s in this case. This stabilization occurs because, at larger values of ‘c’, the influence of the optimal velocity term becomes dominant. As ‘c’ increases, vehicles increasingly adjust their velocities to approach the optimal velocity dictated by their headway. Consequently, the discrepancies in velocities among vehicles decrease, leading to a more uniform flow of traffic. This equilibrium state persists for ‘c’ values greater than 1.25, indicating that further increments in ‘c’ do not significantly alter the behavior of the model.

In the original simulation, the plots for one of these larger values of ‘c’ demonstrate this stabilized behavior. For instance, if we consider a value of ‘c’ greater than 1.25, such as ‘c’ = 2, the plot would show 'umin' and 'umax' converging to around 8.7 m/s after an initial period of instability. The traffic flow would appear more uniform, with fewer fluctuations in vehicle velocities compared to the plots corresponding to smaller ‘c’ values where traffic exhibited wave-like behavior. This stabilized state reflects the model's ability to reach a dynamic equilibrium, where vehicles adjust their speeds to approach an optimal velocity while maintaining consistent spacing between them, resulting in smoother traffic flow.