

Computational Modeling & Simulation I (Interdisciplinary Science)
CS 346

Spring 2020

Optional, Ungraded HW—RK4 & Second-Order Systems
(Please see the notes below!)

- The primary learning objectives for this assignment are:
 - Additional experience with the Runge-Kutta 4 simulation technique, applying it to second-order dynamical systems. This assignment will help you prepare for HW3 (the Hodgkin-Huxley model), coming soon!
 - Additional experience using anonymous functions in clean, readable, and well-organized code.
 - Developing with a programming style that is of particular importance to our community: Code that can be *very* easily modified and re-used for running multiple experiments.
- This assignment is completely optional and ungraded—**It will not count toward your grade for CS346 this semester, whether or not you submit it to me for feedback.** You are encouraged, however, to work on this exercise in the same team that will be doing HW3 (or individually, if you're doing HW3 individually). This assignment is intended to precede that HW.

Optional Exercises: Runge-Kutta 4!

Some introductory notes, which may be helpful for these exercises and as preparation for HW3:

- These exercises are about implementing Runge-Kutta 4 simulations. **They are intended to be done after completing our lab exercises on RK4 simulations** but before—and as preparation for—HW3, in which you will simulate of a classic model, the Hodgkin-Huxley model of action potentials in neurons!
- In these exercises, it is intentional that the code for one exercise may be *very similar* to the code for another in the exercises below. If you submit work, however, please *do not combine responses*—submit different, individual code files for each coding exercise. (It's fine to submit a script that runs multiple other scripts, i.e., running multiple simulations with a single script, but each coding exercise should be in its own file, not combined with others.)
- Your write-ups for these exercises should explain how the intended model is implemented in your simulations. Some decisions may be subtle and require some explanation, so please explain as needed! For simulations, *unless values are given to you for an exercise* (which is frequently the case on some assignments!), your write-up should include the values of constants / parameters employed for each run of the simulation and a very brief explanation of why you chose to run those particular values for simulations. Descriptions of results should be concise and information-heavy; feel free to include figures (e.g., Matlab plots) in write-ups to illustrate your observations.

If there are any questions about what might be good to include in a write-up, please let me know!

- As always, readability is an essential part of the assignment: Make sure both your code and your write-up are easy to read and understand. As part of this, be sure to follow the style guidelines presented in lecture¹ for implementing a Runge-Kutta 4 simulation: Make sure it is easy to tell exactly what function is used to compute the derivative and exactly what arguments are provided to that function for each estimate $\partial_1, \partial_2, \partial_3, \partial_4$.

For example, if you are modeling the dynamics (change in value) of some variable P , then you should have a function `dPdt(...)` somewhere in your code (perhaps it could be defined using anonymous function syntax `dPdt = @(...) ...?`), and in your simulation loop, you would have lines that call that function, such as

```
dp1 = dPdt( ... ) * dt;  
dp2 = dPdt( ... ) * dt;  
dp3 = dPdt( ... ) * dt;  
dp4 = dPdt( ... ) * dt;
```

to calculate the estimates $\partial_1, \partial_2, \partial_3, \partial_4$.

Structure like this makes it easy to read and understand an RK4 simulation loop. Code that is not *at least* this easily readable may not merit full credit. Please feel free to ask me any questions about code style and readability for these exercises!

Exercises: Have A Ball!

These exercises are based on examples from Chapter 3.1: the example of a ball being thrown straight up from the side of a bridge, from the section titled “Acceleration, Velocity, and Position”; and the example of a ball being dropped from a height of 400m, from the section titled “Friction during Fall.”

1. Implement an Euler simulation of the example of a ball being thrown straight up from the side of a bridge, from the section titled “Acceleration, Velocity, and Position.” You should use exactly the parameters, variables, and initial values given in the textbook. Simulate it for the length of time used to get Figure 3.1.2 in the text, using the same timestep used for that figure ($\Delta t = 0.25s$), and graph the values of position and velocity.
2. The simulation resulting in Figure 3.1.2 was not an Euler method, however—it was Runge-Kutta 4. So, implement an RK4 simulation of the same system, and simulate it to generate the graph of position and velocity in Figure 3.1.2.

(When you implement this simulation, please keep in mind exercise 3 below—ideally, your code for this exercise and exercise 3 would be very similar!)

¹They will be included in lecture notes following our RK4 lab, but I’m happy to discuss them with you as part of your work on the lab, if you’d like!

3. Now imagine this simulation occurred in a strange world in which gravity (represented by gravitational constant g) was not the only factor in acceleration of the ball. Instead, acceleration varied with velocity and time: for time t , height $h(t)$, and velocity $v(t)$, $a(t) = g + 0.01 * (v(t) + h(t)) + 0.3 * t^2$. (Note: This will also affect the equation for velocity—it will no longer be $v(t) = 15 - 9.8t$!)

Model and implement an RK4 simulation of this new environment, with the same simulation parameters as above (i.e., simulated for 4 seconds, $\Delta t = 0.25s$).

If your implementation for exercise 2 is implemented in a way that fits the general form of RK4 simulations, modifying your simulation for exercise 2 to make it work for this exercise will be very straightforward!

4. Implement an RK4 simulation of the system given in the section “Friction during Fall,” described in Equation Set 3.1.1. As described in the text, run the simulation for 15s; use a timestep of $\Delta t = 0.01s$. Graph the values of position and speed in the same figure—the result will not look exactly like Figure 3.1.5, because you will not have to represent two different scales on the same y-axis, but it will contain the same information as Figure 3.1.5.

As noted in exercise 5 below, your code for this exercise should be as similar to exercises 2 and 3 as possible.

5. **(For Write-up Only)** It is often important for simulation code to be easily modified and used for different simulations, so in this case (which I admit is somewhat contrived!), if these exercises were graded, maximal credit answers for the above three RK4 simulations would be as similar to each other as possible. That is, if the code is more different from exercise to exercise, it would earn less credit; if the code is more similar from exercise to exercise, it would earn more credit. (This is not always a good criterion for programming style, it’s just being used for these exercises!)

In your write-up, list all changes made to transform your code from exercise 2 to exercise 3, and from exercise 3 to exercise 4. Also, explain why your code answers for the three exercises are *maximally similar to each other*, so code couldn’t be structured to be correct for those simulations with fewer changes.

This may be a bit of an unusual exercise, so please feel free to ask for clarification on any of it! As always, questions are welcome!

(IMPORTANT NOTE: If you feel you need to make slight sacrifices to readability or style to ensure that your three RK4 simulation code answers are as similar as possible, then for this somewhat contrived exercise to practice this coding style, please make those slight sacrifices. Be sure to document them in comments in your code, however, so readers understand why you made the choices you did!)