Case Study I: Initial-Value Problems

MTH 3150: Numerical Methods and Scientific Computing
Franklin W. Olin College of Engineering

Figure 1: Leonhard Euler has a lot of methods, theorems, and results named in his honor — did he actually invent Euler's method?

Overview

IN THIS CASE STUDY we examine methods for solving initial-value problems [1]. You should already be familiar with Euler's method from earlier Olin courses, and you should have used some of the built-in MATLAB solvers like `ode45` at some point in the past. We want you to develop and implement algorithms for Euler's method and an extension known as Modified Euler's method and determine the accuracy of these methods by solving a given problem. In addition, we want you to use both methods in combination in order to select a variable time-step in order to compute a solution to a given error tolerance, and then compare your method with the performance of `ode45`.

[1] For our purposes an initial-value problem consists of a system of ordinary differential equations $\dot{\mathbf{y}} = \mathbf{f}(t, \mathbf{y})$ along with a set of initial conditions $\mathbf{y}(0)$ — more general initial-value problems can be formulated.

Baseball Model

THE FLIGHT OF A BASEBALL is governed by three main forces: gravity, drag due to air resistance, and Magnus force due to spin. If we ignore the Magnus force[2], the path of the baseball stays in the plane, so we can model it as a projectile in two dimensions. Under these conditions, a reasonable model for the baseball based on Newton's Laws is

[2] How justified are we in ignoring this force?

$$\begin{aligned} \frac{d\mathbf{r}}{dt} &= \mathbf{v}, \\ m\frac{d\mathbf{v}}{dt} &= m\mathbf{g} - \frac{1}{2}\rho A C_d |\mathbf{v}|\mathbf{v}, \end{aligned}$$

Figure 2: Isn't cricket the same thing as baseball, but just slightly more interesting?

where $\mathbf{r} = [x(t) \;\; y(t)] \;\; m$ is the position of the baseball at time t and $\mathbf{v} = [\dot{x}(t) \;\; \dot{y}(t)] \;\; m/s$ is its velocity. $\mathbf{g} = [0 \;\; -9.8] \;\; m/s^2$ is the acceleration due to gravity, C_d is the drag coefficient (0.3), ρ is the density of air ($1.3Kg/m^3$ at sea level), A is the cross-sectional area of the baseball ($0.0042m^2$), and m is the mass of the baseball ($0.145Kg$). For the purpose of this case study, let's assume that the baseball leaves the bat at an initial height of 1 meter with a velocity of roughly 45 m/s

at an angle of 35° to the horizontal. The initial conditions are therefore

$$\begin{aligned} x(0) &= 0, \\ y(0) &= 1, \\ \dot{x}(0) &= 45\cos(35^\circ), \\ \dot{y}(0) &= 45\sin(35^\circ). \end{aligned}$$

Objectives

THIS CASE STUDY has several goals: first, we want you to examine explicit one-step methods for solving initial-value problems; second, we want to implement Euler's methods and Modified Euler's method in a general setting, validate them on a problem with a known solution[3], and then use them to solve a problem without a known solution; third, we want to re-acquaint you with MATLAB in order for you to implement your methods in a reliable, robust, maintainable, and efficient manner[4]; and fourth, we want you to write a report in which you put these various pieces together in a professionally-written document and which demonstrates your understanding of the methods and their context. In particular, we want you to

- Develop a set of functions in MATLAB in order to simulate the motion of the baseball using Euler's method. Use your functions to compute the range (horizontal distance in flight) for a variety of fixed time-steps. Repeat for Modified Euler.
- Extend and modify your algorithms for Euler and Modified Euler in order to control error in the solution by using variable time-steps. Use your code to compute the range for a variety of error tolerances and examine the distribution of time-steps. Compare to `ode45` in MATLAB.

[3] By known solution we mean a problem whose solution can be written down on paper using functions like *exp*, *sin*, *cos*, etc.

[4] By reliable we mean the code is error free and can be trusted to compute what it is supposed to compute. By robust we mean the code can be applied generally and can handle "bad" situations. By maintainable we mean that the code can be changed and updated with minimal effort. By efficient we mean the code does not waste computing resources without good reason

Validation and Work

VERY FEW PIECES OF CODE work at the first time of asking. Most code of more than a few lines needs to be incrementally built, with each section being tested, before being packaged away for later use. Code written for the purpose of scientific computing often has an additional burden in the sense that the code is being used to solve a previously unsolved problem — how do we build the confidence to trust the results of our simulations? Normally we achieve this through careful code development and validation on known problems. In this case study, we don't have a solution on paper to the range of a baseball under the influence of drag, but we can solve for the range in the absence of drag. After validation, we then use our code to do some work - in this case to compute the range with drag, and to examine the performance of the methods being used. In particular, for this case study we want you to

- Validate your algorithm in the absence of drag where an exact solution exists for the time of flight and range achieved by the baseball[5]. Run your algorithms with fixed time-steps for the predicted flight time and record the position of the baseball at this time.
- Define the error in the numerical approximation as the difference between the predicted position of the baseball and the position determined computationally. Examine this error as a function of time-step for both Euler's and Modified Euler's method.
- Combine both methods in order to choose time-steps which result in approximations within a given error tolerance, and examine the distribution of time-steps as a function of the error tolerance.

[5] You will need to work out the solution on paper

Report

Prepare a brief typewritten report of roughly 4 pages in length in which you review the methods used, your implementation using code snippets as evidence, and your results. At a minimum your report should include:

- a section which briefly discusses the methods used, and places them in the broader context of methods for solving initial-value problems[6]
- a section which briefly discusses your implementation of the methods on this problem, using small pieces of code as evidence.
- a section which briefly discusses the results of your simulations, including (but not limited to)
  - a graph which compares the error in solving the problem **without** drag as a function of fixed step-size Δt for each method[7].
  - a graph which compares the range **with** drag as a function of fixed step-size Δt for each method.
  - a graph which compares the computation time as a function of step-size Δt for each method.
  - a graph which compares the distribution of step-sizes as a function of the error tolerance for both your method and `ode45`.
- a section which briefly summarizes your findings, including questions you have about this case study and any reflection you care to provide.

[6] This means you need to **read** about methods for solving initial-value problems from a variety of sources, i.e. not just wikipedia

[7] Don't forget that log-log plots are your friend!