CTA200H Assignment 2

Sacha Lévy

McGill University, sacha.levy@mail.mcgill.ca

Abstract

In this paper we present our results for the second assignment of the CTA200H summer course.

1 Numerical approximate derivative

1.1 Method

We first define the python method used to approximate value of the function derivative at a given point. We then compute the relative error between the numerical and analytical derivative.

1.2 Analysis

We observe on 1 that the relative approximation error grows linearly in log scale with respect to the step size used in the numerical derivation process. This indicates that to gain an order of magnitude in relative precision, we would need to decrease our step size by an order of magnitude too.

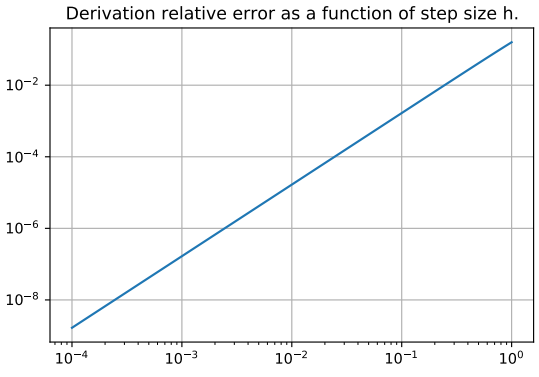

Figure 1. Loglog plot for derivation error as a function of step size.

2 Study of boundary on recursive complex function

2.1 Method

We define the recursive complex function as follows:

$$zi + 1 = zi + c$$

. We compute the values of this function up to 9 iterations (as we are limited by integer overflow), and check the triangular identity binding the square of the modulus to the sum of the squares of the imaginary and real parts of the resulting numbers. We then produce two plots first showing the distribution in the complex space of the unbounded numbers not respecting the identity, and another plot showing from which iterations did a number become unbounded. We compute the color matrix for the unbounding starting with an array full of zeros. On each iteration we check if the any number has become unbounded, if so we check if this number was already unbounded on the previous iteration. If not, we set the value of the unbounded iteration to the current iteration counter.

2.2 Analysis

Based on the plot shown in 2, the distribution of the unbounded number seems random. Similarly, the color map shows numbers becoming gradually unbounded. We see that these numbers seem to also follow a random distribution as shown in figure 3.

3 Disease modelling using SIR model and scipy integrator

3.1 Method

We employ the scipy *solve_ivp* module to solve our SIR differential equation system. We use parameters (0.05) for beta, and (5, 25) for gamma. We use matplotlib to represent the different levels of population disease on 4 different subplots (depending on the parameter values for gamma and beta).

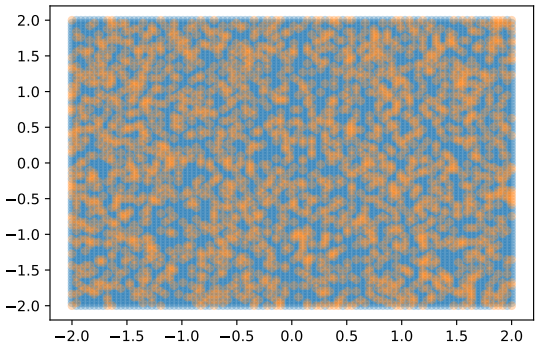

Figure 2. Scatter plots showing bounded and unbounded points distribution.

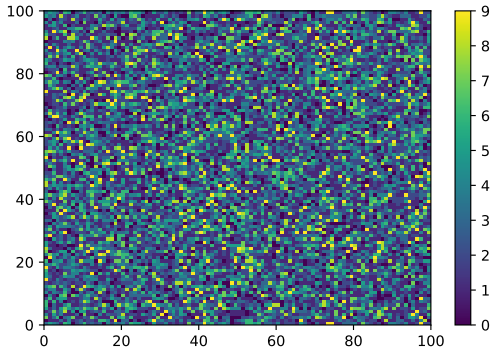

Figure 3. Colormap displaying de-bounding iteration gradient.

3.2 Analysis

Beta directly correlates with the rate at which people become infected and are susceptible of infection. Gamma determines the rate of recovery. If gamma is too low then the population is not able to recover, and if beta is to low then the number of people susceptible of contracting the disease drops too quickly to be realistic. Note that the parameter chosen here are not ideal to simulate the spread of the disease as we notice that the infected curve remains at 0 throughout the time-span of convergence towards a solution.

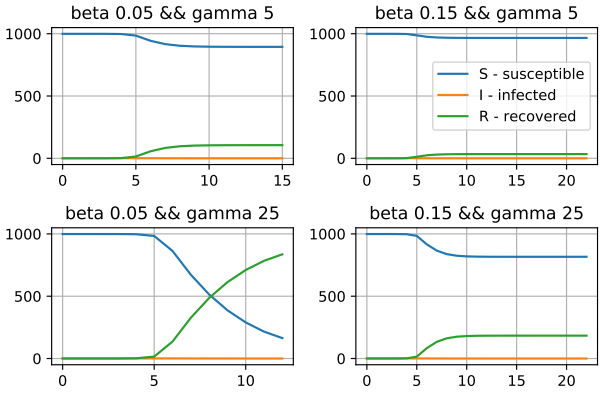

Figure 4. Parametric curves for solved IVP system.