CTA200 Assignment 2

Emily Crawley

05/05/2021

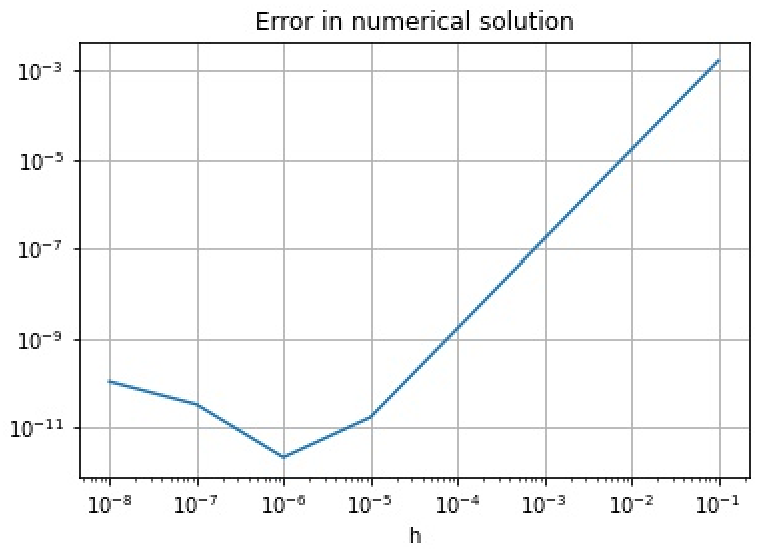
Question 1

Using the approximation given for evaluating a function $f(x)$ at x_0 for small finite h, we define `deriv(f, x0, h)` which computes

$$\frac{f(x_0+h)-f(x_0-h)}{2h}$$

We then define our function to be differentiated, `f(x) = sin(x)` and its analytical derivative `dfdx(x) = cos(x)`. Values of $h < 1$ were chosen ranging from 0.1 to 1e-8.

We compute the analytical derivative $cos(0.1)$, and the numerical derivatives at each value of h, plotting them each against the analytical solution. Since the h values vary by orders of 10 we plot the error on logarithmic axes using `pyplot.loglog`.



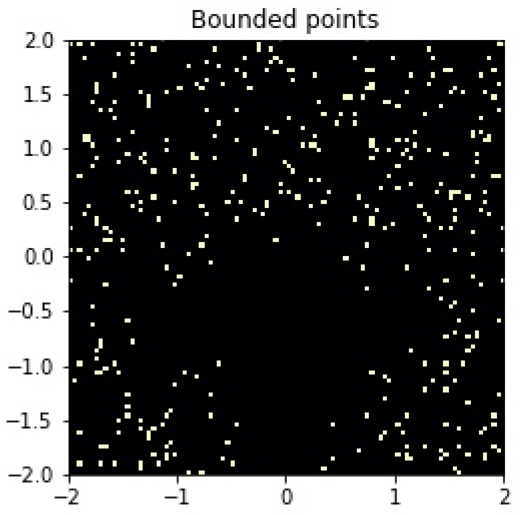
From the plot we can see that the error in the numerical solution decreases linearly for smaller values of h, until around $h < 10^{-6}$ when the larger relative error presumably comes from the precision limitations of the `float` type. The slope of the linear part of the graph represents how quickly the error decreases as h decreases, so in some sense represents its effectiveness in terms of accuracy. That is, we want to be able to greatly decrease error with only one or two order-of-magnitude decrements of h, so a steeper slope is preferable.

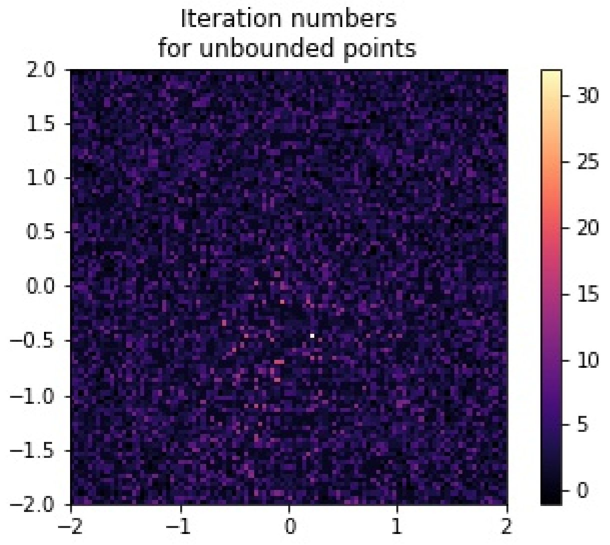
Question 2

We cover the complex plane in the square $|z|^2 = \Re(z)^2 + \Im(z)^2$ by taking $c = x + iy$ for numerous combinations of x- and y-values between -2 and 2.

For each point c we iterate the expression $z_{i+1} = z_i^2 + c$ starting from $z_0 = 0$ and test the absolute square of each z_i to see if it diverges from the boundary. If so, we assign it a value of 0 in a 2D boolean array representing whether each point is bounded or not. We also record the iteration number in a separate 2D array. The iteration number array contains values of -1 for points that stay bounded. We then plot the results using `pyplot.imshow`.

When tested running for 1000 iterations, none of the points in the sample set take more than 32 iterations to diverge, so to save time we will only iterate for 32 steps.





Question 3

Our initial value problem is given as:

$$\frac{dS}{dt} = -\frac{\beta S I}{N}, \tag{1}$$

$$\frac{dI}{dt} = \frac{\beta S I}{N} - \gamma I, \tag{2}$$

$$\frac{dR}{dt} = \gamma I \tag{3}$$

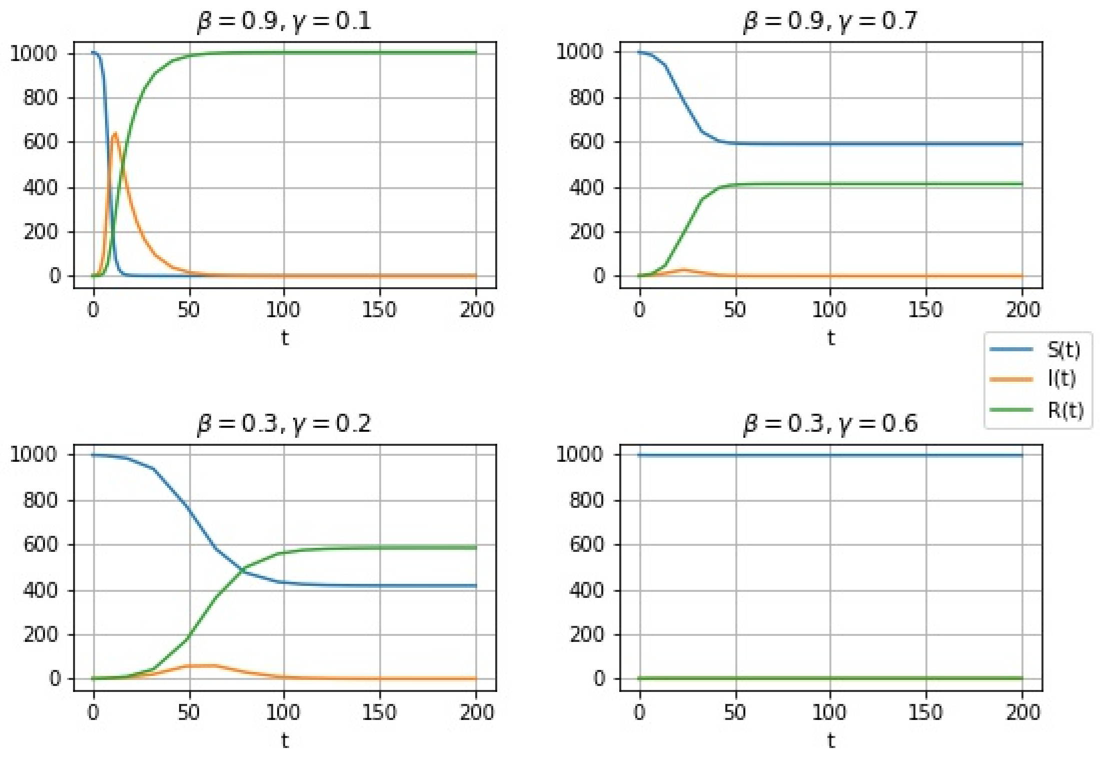
$I(0) = 1, S(0) = 999, R(0) = 0$

$N = 1000$, $0 \leq t \leq 200$

β represents an infection rate and γ represents a recovery rate, so we will select some meaningful variations on these parameters, i.e. a disease that spreads quickly but has a fast recovery rate, a disease that spreads quickly with a slower recovery rate, etc.

Using `scipy.integrate.solve_ivp` as the ODE integrator, which takes parameters `(fun, tspan, y0)`, we set up the time derivative functions as a vector $[\frac{dS}{dt}, \frac{dI}{dt}, \frac{dR}{dt}]$ for the `fun` parameter, and the initial conditions vector $[S(0), I(0), R(0)]$ as `y0`. We also specify the parameter `vectorized=True` so that the integrator handles the vectors properly. We will use its default Runge-Kutta method, `method='RK45'`.

The integrator outputs the three numerical solutions for S, I, R:



β = 0.9, γ = 0.1
β = 0.9, γ = 0.7
β = 0.3, γ = 0.2
β = 0.3, γ = 0.6
t
S(t)
I(t)
R(t)