Numerical solution of deterministic epidemiological models*

John M. Drake & Pejman Rohani

1 Introduction

Many of the core theories of epidemic propagation are expressed is a system of *ordinary differential equations* known as a *compartmental model*. This session introduces techniques for numerically solve systems of nonlinear differential equations with an adaptive step size solver.

2 The SIR model

As we introduced in the lecture, the classical SIR compartmental model tracks the fraction of the population in each of three classes (susceptible, infected, recovered). In a *demographically closed* system, flow out of one class must enter another class, giving rise to a *conservation property*. The state variables change according to the following system of differential equations:

$$\begin{aligned}\frac{dS}{dt} &= -\beta SI \\ \frac{dI}{dt} &= \beta SI - \gamma I \\ \frac{dR}{dt} &= \gamma I\end{aligned}$$

where S, I, and R are the proportion of susceptible, infected, and recovered individuals and γ is the recovery rate.

In a *demographically open* system, the number of individuals in the population may change due to births and deaths that happen at *per capita* rates α and μ. The we have

$$\begin{aligned}\frac{dS}{dt} &= \alpha - \beta SI - \mu S \\ \frac{dI}{dt} &= \beta SI - \gamma I - \mu I \\ \frac{dR}{dt} &= \gamma I - \mu R\end{aligned}$$

If we set $\alpha = \mu$ then births exactly balance deaths and the population remains at a constant size, yielding

$$\frac{dS}{dt} = \mu - \beta SI - \mu S$$
$$\frac{dI}{dt} = \beta SI - \gamma I - \mu I$$
$$\frac{dR}{dt} = \gamma I - \mu R$$
.

Like many epidemiological models, one can't solve the SIR equations analytically. Rather, to find the *trajectory* of a continuous-time model such as the SIR, we integrate these equations numerically. What we mean by this is that we use a computer algorithm to approximate the solution. In general, this can be a tricky business. Fortunately, this is a well studied problem in numerical analysis and (when the equations are smooth, well-behaved functions of a relatively small number of variables) standard numerical integration schemes are available to approximate the integral with arbitrary precision. Particularly, R has a very sophisticated ODE solver, which (for many problems) will give highly accurate solutions. To use the numerical integration package, we must load the package

```
> require(deSolve)                          #deSolve library needed for this computing session
```

The ODE solver needs to know the right-hand sides of the ODE. We give it this information as a function (sub-routine). Note that the form of the arguments and output of this function must exactly match what is expected by the `ode` routine. Thus, for instance, the time variable `t` must be the first argument even if the function is *autonomous* or *time-invariant* so that `t` is neglected in the calculation. Here we right a function to return the derivatives of the closed SIR model.

```
> sir.model.closed <- function (t, x, params) {    #here we begin a function with three arguments
+   S <- x[1]                                      #create local variable S, the first element of x
+   I <- x[2]                                      #create local variable I
+   R <- x[3]                                      #create local variable R
+   with(                                          #we can simplify code using "with"
+        as.list(params),                          #this argument to "with" lets us use the variable names
+        {                                         #the system of rate equations
+          dS <- -beta*S*I
+          dI <- beta*S*I-gamma*I
+          dR <- gamma*I
+          dx <- c(dS,dI,dR)                       #combine results into a single vector dx
+          list(dx)                                #return result as a list
+        }
+        )
+ }
```

Notice that here, we've assumed β is constant.

[Note: In case the `with` function is unfamiliar, it serves here to make the parameters `params` available to the expressions in the brackets, *as if they were variables*. One could achieve the same effect by, for example, `dS <- params["mu"]*(1-S)-params["beta"]*S*I` and so on.]

We now state the times at which we want solutions, assign some values to the parameters, and specify the *initial conditions*, *i.e.*, the values of the state variables S, I, and R at the beginning of the simulation:

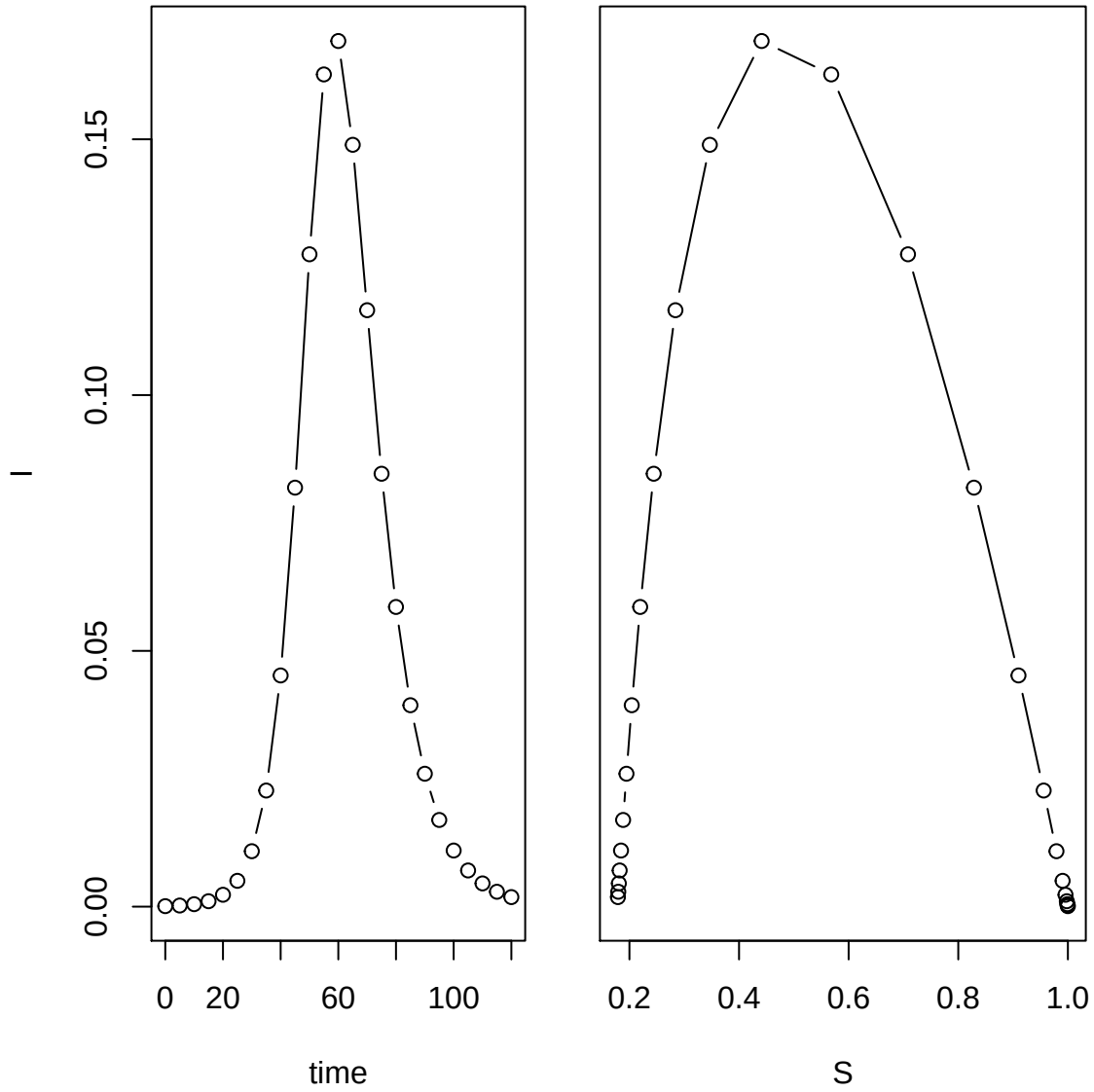
```
> times <- seq(0,120,by=5)                    #function seq returns a sequence
> params <- c(beta=0.3,gamma=1/7)             #function c "c"ombines values into a vector
> xstart <- c(S=9999/10000,I=1/10000,R=0)     #initial conditions
```

Next, we simulate a model trajectory with the `ode` command:

```
> out <- as.data.frame(ode(xstart,times,sir.model.closed,params))  #result stored in dataframe
```

and plot the results

```
> op <- par(fig=c(0,0.5,0,1),mar=c(4,4,1,1))                    #set graphical parameters
> plot(I~time,data=out,type='b')                                 #plot the I variable against time
> par(fig=c(0.5,1,0,1),mar=c(4,1,1,1),new=T)                     #re-set graphical parameters
> plot(I~S,data=out,type='b',yaxt='n',xlab='S')                  #plot phase portrait
> par(op)                                                       #re-set graphical parameters
```



Exercise 1. Explore the dynamics of the system for different values of the β and γ parameters by simulating and plotting trajectories as time series and in phase space (e.g., I vs. S).

Exercise 2. Explore the dynamics of the system for one set of β and γ at different initial conditions. What happens if there is pre-existing immunity in the population?

Exercise 3. Modify the codes given to study the dynamics of a demographically open SIR model.

***Exercise 4.** Modify the codes given to study the dynamics of an $SEIR$ model.