SIR Program Suite Description and Instructions

Program Overview

The SIR suite contains four Matlab programs, one that implements the mathematical model, one that runs a test on that implementation, and two that use it to set up virtual experiments.

sir_sim.m is a function program that accepts input data called from a keyboard command or a script file and produces a table of results for the class counts in each day. The input parameters include β and γ along with parameters that indicate the initial fractions of the infected and removed classes.

SIR_simtest.m is a script that runs one simulation using sir_sim.m. A single plot shows the susceptible, infectious, and removed fractions of the population for a given scenario. Scenarios are specified using the basic reproductive number, the average duration of the infectious phase, and the initial infectious and removed fractions.

SIR_simplot.m is a script that uses sir_sim.m to prepare a set of plots that show the time history of the scenario. Two plots show the susceptible and infected populations over time, while the third plot shows the solution as a curve in the SI phase plane.

SIR_paramstudy.m is a script that uses sir_sim.m to prepare a set of plots that show how certain key quantities change when one of the parameters is varied:

1. The maximum fraction of infected individuals;
2. The time when the maximum number infected is reached;
3. The final percentage still at risk at the end of the outbreak;
4. The number of days before the total of infected people is below a target threshold.

Getting Started

We assume here that the reader already knows how to use the Matlab editor to create and execute a script file, which is a set of instructions to be executed according to a prescribed flow of control. No significant programming experience is required. Users should start with SIR_simtest.m, which is designed to ensure that the main program is functioning correctly and to provide an easy entry into Matlab programming.

SIR_simtest.m is a very simple program, with no programming structures such as loops or conditionals. It contains only 24 lines of code, along with section dividers (lines that start with %%), comments (lines that start with %), and blank lines.

The lines in the SCENARIO DATA, COMMON DATA, and COMPUTATION sections are simple assignment statements. These are statements of the form "name=value" that assign either a specific value or a value computed from a formula or function to the name. The scenario is defined in the SCENARIO DATA section. This is the only section in the program that the user needs to change. This structuring is good programming practice, as it means that all of the lines that need to be

specified for a scenario are grouped together, while other blocks of code contain only lines that do not need to be changed when the scenario changes.

Line 40 is a call to the sir_sim function. This function resides in the separate file sir_sim.m, which must be in the same directory as SIR_simtest.m. The function call uses the name of the function followed by a list of arguments in parentheses. The function file uses these arguments in place of simple assignment statements. Its output consists of lists giving the time histories of the three state variables. Lines 42-43 compute a vector of new infections by subtracting the following day's susceptible count from the current day's.

Lines 32-34, and 49-54 are graphics statements. It is always a good idea to start a graph with "clf" and "hold on", which clear out any old graph and tell Matlab not to erase any subsequent plots without special instructions to do so. The "opengl" statement avoids some graphics bugs that occur on some computers. The plot statements require the horizontal and vertical components of the data, along with any options. These options consist of a pair of specifications, one for the plotting parameter to be controlled and the other for the value to be used. The default plot lines are too thin for most presentation purposes, so it is often good to set the line width to be some value between 1.4 and 1.7, inclusive. Some graph properties, such as axis labels, are best set in subsequent lines, as is done in lines 52-53. Where desired, one can use the "legend" function to specify which curve is which. The arguments to be given are a list of character strings followed by any optional properties. Usually Matlab makes a good choice of location without the programmer's help, but it is often best to specify where you want the legend using direction names such as "Northeast".

Lines 56-58 produce the text output for the program. The single quotation mark appended to "times" converts that variable from a row vector to a column vector; the definition of "results" makes a row of column vectors. The last two program lines give the maximum infectious fraction and the final susceptible fraction.

SIR_simtest.m can be used to run any scenario for the SIR epidemic model simply by changing the four lines in the scenario data section. Sometimes you may want to tinker with graphics statements to improve the appearance and/or design of a plot.

Using SIR_simplot and SIR_paramstudy

These scripts are constructed so that they can be applied to different experiments with a minimum of changes. The changes are identified in the comments of the script files, and are described here in more detail.

It is important to understand the logical structure used to organize these programs. Each is subdivided into sections with a particular name and purpose.

1. The scripts begin with some brief documentation that describes the program and how to use it and, most importantly, identifies the particular version of sir_sim that it has been tested on. The script may work with newer or older versions, depending on what changes were made, but one should not expect this.

2. The first programming section contains default scenario data, which are the values that will be assumed for the parameters to be used as input values for sir_sim if that particular parameter is not the experiment parameter. These default values may need to be changed, depending on the experiment.

3. The next section contains the data for the experiment parameter, described in the program as the "Independent Variable". The program is designed to require minimal changes for different experiments; most of those changes are in this section.

   (a) First, the set of parameter values to be used for the independent variable must be specified with a generic name. The actual parameter you have chosen will be identified elsewhere.

      i. The values are listed in SIR_simplot under the generic name "xvals" and are specified individually using Matlab syntax for a row vector or list. Each of the parameter values in this list will generate its own set of curves in the three-panel plot. Generally a set of 3 to 8 values is best.

      ii. In SIR_paramstudy, the independent variable values are used as the horizontal coordinate on each plot, so a much larger set is necessary. The user specifies the first value, the last value, and the count of values, keeping in mind that the end point values are both counted. For example, if you want the values 0, 0.05, 0.1, and so on up to 1.0, you choose first=0, last=1, and N=21, dividing the interval from 0 to 1 into 20 equal parts.

   (b) The set of values is the only thing that needs to be defined in the INDEPENDENT VARIABLE DATA section of SIR_simplot. SIR_paramstudy also requires the user to specify the name of the parameter to be used as the horizontal axis label. This specification only applies to the graph; so far the program does not know which of the parameters is to take on the values predefined using first, last, and N.

4. Next comes the INITIALIZATION section, which does basic housekeeping tasks such as setting up the plot windows and defining any needed data structures.

5. The programs conclude with sections for computation and output, with the graphs being created in the computation section if they occur inside a loop or in an output section if they are only produced after all the data is collected.

6. The key to the program structure is that each parameter has a default value that will be used in all scenarios, except that one of the parameters will instead use the values specified either as xvals or using first, last, and N. The function call requires one value to be passed in for each of the parameters. The first line inside the main program loop identifies the name of the parameter whose values were defined in the Independent Variable section. This line overwrites the previous value each time through the loop; meanwhile, the other parameters have only been given default values, so these are also used. The default value for the independent variable parameter is never used, but having it specified in the data section allows for a minimum of changes as the program is repurposed for a new experiment.

7. Under some circumstances, the user might want to tinker with the details of the plot specifications, perhaps by overriding Matlab's choice of axis limits, modifying a legend, or adding text. Generally, the graphs produced by Matlab's automatic routine will be reasonably well designed.

SPUR modifications

Addressing questions about the impact of isolation of individuals with symptoms requires a more sophisticated model, such as the SPUR model described in the Student Notes. These changes are needed to convert the SIR programs to SPUR. Start by saving copies of SIR_simtest.m and SIR_paramstudy.m as SPUR_simtest.m and SPUR_paramstudy.m. Then make the indicated changes.

For spur_sim:

1. Change the function argument list to `(beta,sigma,gamma,I0,q,V,target)`.
2. In INITIALIZATION, change the definition of results to have four columns instead of three. After the definition of `results`, add these lines to calculate `P0` and `U0`:

   ```
   b = beta-sigma+gamma;
   c = -(1-q);
   rho = (sqrt(b.\^2-4*beta*c)-b)/(2*beta);
   P0 = I0/(1+rho);
   U0 = rho*P0;
   ```

   Also change the formula for `Y` to `Y = [S0,P0,U0];`
3. In COMPUTATION, change both instances of `Y(2)` to `Y(2)+Y(3)`. Also change the formula for `I` to `results(:,2)+results(:,3)` and the formula for `R` to `results(4)`.
4. Change the FUNCTION FOR THE DIFFERENTIAL EQUATION to match the model given in P1-2, Student Notes. This will require you to change the splitting of components to reflect the variable order S-P-U, add a line `I=P+U`, replace the formula for `Ip` with two formulas for `Pp` and `Up`, and change the assembly line to use the derivatives `Sp`, `Pp`, and `Up` instead of `Sp` and `Ip`.

For SPUR_simtest:

1. In SCENARIO DATA, set `R0=5` and `T=10` and add assignments `Tp=2` and `q=0.5`.
2. In COMPUTATION, add `sigma=1/Tp` and `gamma=1/(T-Tp)` before the formula for `beta`. Also change the function call from `sir_sim` to `spur_sim` and change the function call to match the function name and argument list of spur_sim.
3. Run SPUR_simtest to make sure everything works. Error messages referring to the computation line indicate errors in spur_sim. After you fix these, rerun SPUR_simtest until the program works.
4. It is a good idea to update the comments in the programs. This makes it easier to understand programs you haven't looked at in a long time.

For SPUR_paramstudy:

1. Modify DEFAULT SCENARIO DATA and COMPUTATION in the same way as in SPUR_simtest.
2. Set the experiment to vary the parameter `q` from 0 to 1.