

Communication and usage of genAI

The **only** communication allowed is with the lecturer, who is available to clarify questions if necessary. You should promise academic honesty, and that this notebook is your own. You are allowed to inspect only the course materials of FP2P 2025 and your own notes.

The Mathematica Notebook Assistant requires a paid subscription, which is not available.

If you get stuck, it is allowed to consult *gemini.google.com*

But if you have used an AI tool in any cell (with code or a conclusion text), please indicate that clearly in a separate text cell below that cell, and highlight that text cell.

As an ultimate resort, only after having tried the genAI, you can ask for help via the mail (from your VU mail address, not from canvas) with your notebook attached to **ivo@few.vu.nl**.

As a proof of that promise please insert here a scan or photo of your signature via the **Insert** menu, with **Picture/From File**:

Kushnava Singha, 28.01.2003,2872230

If all else fails type your name, date of birth, and student number in the above cell.

Grading

The maximum number of points is indicated before each question, total 100. If you have used an AI tool in a cell, points will be subtracted, depending upon the importance, to be judged by the lecturer. If your answer is completely AI generated *and* the answer is correct, you will get 60% of the maximum number of points.

Most common mistakes

always note the colors of your variables and symbols, blue is unknown, black is a known global variable, green is a dummy or local variable (see menu Help / Why the Coloring?)

take care of matching { } or [] or () magenta color of {, [or (indicates incomplete matching.

a space between two variables is a multiplication, better use a * for multiplication, instead of a space

too many or too little spaces or primes

mistakes in the syntax of the arguments of a function call. the _ is mandatory for a dummy name, the : is needed before the default value. avoid usage of the _ in names of variables.

mistakes in the arguments when calling a function. even when default values are present, when you call the function with e.g. a different fourth argument, the first three arguments must be supplied.

interchanging the letter o and the digit 0, or the letter l and the digit 1

functions start with a capital (Sin, NDSolve, PlotStyle etc)

function arguments between square brackets, separated by commas

derivative notation: $x'[t]$ and $x''[t]$ etc (first a letter, then 1 or 2 primes, then between $[]$ the independent variable, also with $x[t]$)

omitting the time argument, or wrong time argument as initial condition

omitting the parentheses for a numerator

interchanging of plus and minus signs,

note the difference between $=$ (Set) and $==$ (Equal) and $:=$ (SetDelayed, used to define a function with arguments)

note the difference between $/.$ and $//.$ (ReplaceRepeated)

sloppy use of global variables

omission of $;$ (CompoundExpression) in Module

names of objects must be exactly identical (because Mathematica is case sensitive)

names of objects must be unique, at least **2** characters, which can not be confused with names of functions or variables (avoid name botches). the first character must *not* be a digit.

click between two cells to get a new input cell (NB never try to type input in an output cell)

When your computation does not end, select from the Evaluation menu Abort Evaluation (**Alt+.**).

When necessary, use list indexing $[[]]$ (Part) or functions like Flatten, First, Last to extract expressions without curly brackets $\{ \}$

use Copy and Paste (from parts of output cells) only in case of emergency.

in case of emergency: select from the Evaluation menu Quit Kernel
or start with an empty slate (remove all global variables):

In[303]:=

```
Remove["Global`*"]
```

When this statement is on top of your notebook you can also select Evaluate Notebook from the Evaluation menu

When Manipulate crashes ...

Evaluation / Quit Kernel

Evaluation / Dynamic Updating Enabled

The SIR model of two countries

The simplest SIR model of two countries consists of three compartments for each country: susceptible, $s(t)$; infected, $i(t)$; and recovered, $r(t)$ (note the lowercase r). The six compartments are described by six coupled nonlinear ordinary differential equations (ODEs): $s_1'(t) = -R01 \cdot \gamma_1 \cdot i_1(t) \cdot s_1(t)$, $i_1'(t) = R01 \cdot \gamma_1 \cdot i_1(t) \cdot s_1(t) - (\gamma_1 + m_{21}) \cdot i_1(t) + m_{12} \cdot i_2(t)$ and $r_1'(t) = \gamma_1 \cdot i_1(t)$ and $s_2'(t) = -R02 \cdot \gamma_2 \cdot i_2(t) \cdot s_2(t)$, $i_2'(t) = R02 \cdot \gamma_2 \cdot i_2(t) \cdot s_2(t) - (\gamma_2 + m_{12}) \cdot i_2(t) + m_{21} \cdot i_1(t)$ and $r_2'(t) = \gamma_2 \cdot i_2(t)$. The model contains six parameters: the reproduction numbers $R01$ and $R02$ (note the uppercase R in $R0$), the recovery rates γ_1 and γ_2 (which will be assumed to be equal to γ for both countries) and the interaction rates m_{21} and m_{12} . The initial conditions are $s_1(0) = s_{10}$, $r_1(0) = r_{10}$, $i_1(0) = i_{10}$, $s_2(0) = s_{20}$, $r_2(0) = r_{20}$, $i_2(0) = i_{20}$. Because of the nonlinearity we will solve the coupled nonlinear ordinary differential equations numerically use `NDSolveValue`. Because we have to solve numerically all the values of parameters, initial conditions and the time interval have to be known beforehand.

(a,4) Make a list (using `{` and `}` or `List`) called **values** with substitution rules for these initial conditions

```
values = {s10 → 1, i10 → (1 / 90), r10 → 0, tend → 200, γ → 1 / 4, R01 → 4 / 3, s20 → 1, i20 → 0, r20 → 0, R02 → 4 / 3, m12 → 1 / 300, m21 → 1 / 300}
```

Retype the above input line yourself in the input cell below and check that it works.

In[304]:=

```
values = {s10 → 1, i10 → 1 / 90, r10 → 0, tend → 200, γ → 1 / 4, R01 → 4 / 3, s20 → 1, i20 → 0, r20 → 0, R02 → 4 / 3, m12 → 1 / 300, m21 → 1 / 300}
```

Out[304]=

$$\left\{ s_{10} \rightarrow 1, i_{10} \rightarrow \frac{1}{90}, r_{10} \rightarrow 0, \text{tend} \rightarrow 200, \gamma \rightarrow \frac{1}{4}, R_{01} \rightarrow \frac{4}{3}, s_{20} \rightarrow 1, i_{20} \rightarrow 0, r_{20} \rightarrow 0, R_{02} \rightarrow \frac{4}{3}, m_{12} \rightarrow \frac{1}{300}, m_{21} \rightarrow \frac{1}{300} \right\}$$

(b,15) Now solve the coupled nonlinear ordinary differential equations numerically for time from 0 until the end time **tend**

```







{s1t, i1t, r1t, s2t, i2t, r2t} = NDSolveValue[{s1'[t] == -R01*γ*i1[t]*s1[t],
      i1'[t] == R01*γ*i1[t]*s1[t] - (γ+m21)*i1[t] + m12*i2[t],
      r1'[t] == γ*i1[t],
      s2'[t] == -R02*γ*i2[t]*s2[t],
      i2'[t] == R02*γ*i2[t]*s2[t] - (γ-m12)*i2[t] + m12*i1[t],
      r2'[t] == γ*i2[t],
      i1[0] == i10, s1[0] == s10, r1[0] == r10,
      s2[0] == s20, r2[0] == r20, i2[0] == i20} /. values,

      {s1[t], i1[t], r1[t], s2[t], i2[t], r2[t]}, {t, 0, tend /. values}]

```

Out[308]=

```

{InterpolatingFunction[ Domain: {{0., 200.}} Output: scalar] [t],
InterpolatingFunction[ Domain: {{0., 200.}} Output: scalar] [t],
InterpolatingFunction[ Domain: {{0., 200.}} Output: scalar] [t],
InterpolatingFunction[ Domain: {{0., 200.}} Output: scalar] [t],
InterpolatingFunction[ Domain: {{0., 200.}} Output: scalar] [t],
InterpolatingFunction[ Domain: {{0., 200.}} Output: scalar] [t]}

```







Call your output objects {s1t,i1t,r1,s2t,i2t,r2}

In[311]:=

```
{s1t, i1t, r1t, s2t, i2t, r2t}
```

Out[311]=

```

{InterpolatingFunction[ Domain: {{0., 200.}} Output: scalar] [t],
InterpolatingFunction[ Domain: {{0., 200.}} Output: scalar] [t],
InterpolatingFunction[ Domain: {{0., 200.}} Output: scalar] [t],
InterpolatingFunction[ Domain: {{0., 200.}} Output: scalar] [t],
InterpolatingFunction[ Domain: {{0., 200.}} Output: scalar] [t],
InterpolatingFunction[ Domain: {{0., 200.}} Output: scalar] [t]}

```

(c,4) Now plot this list from 0 until the end time **tend**, with PlotOption PlotLegends→{"Susceptible1", "Infected1", "Recovered1", "Susceptible2", "Infected2", "Recovered2"}

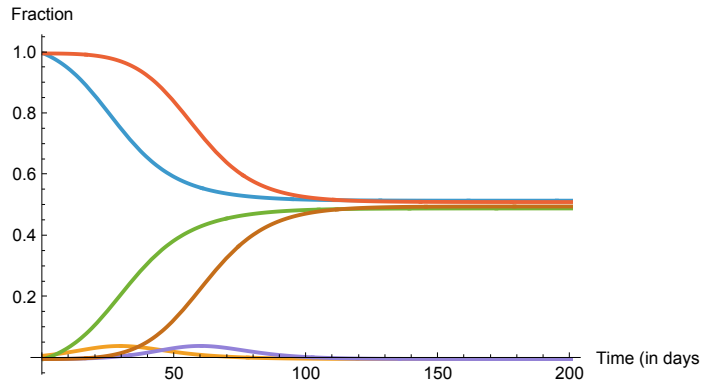
In[323]:=

```
s1t[5]
Plot[{s1t, i1t, r1t, s2t, i2t, r2t}, {t, 0, tend /. values},
  AxesLabel → {"Time (in days", "Fraction"}]
```

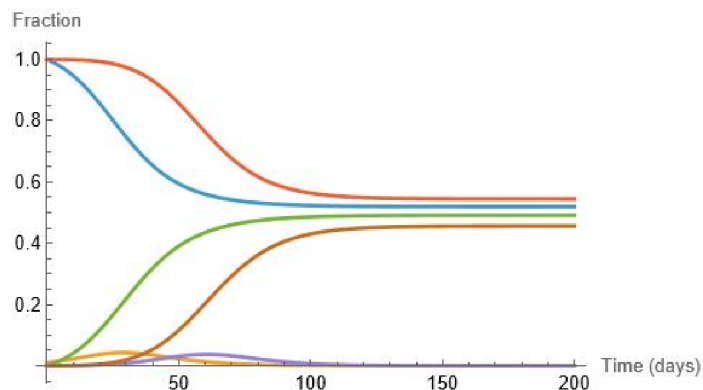
Out[323]=

InterpolatingFunction[ Domain: {{0., 200.}} Output: scalar] [t] [5]

Out[324]=



Your plot should look like this



(d,20) Write a function called **modSIR** that uses Module, with as arguments R01, R02, tend, i10, i20, m12, m21, γ , s10, r10, s20, r20.

Use as the default values those from the list **values** above. Check that you can reproduce the previous plot.

```

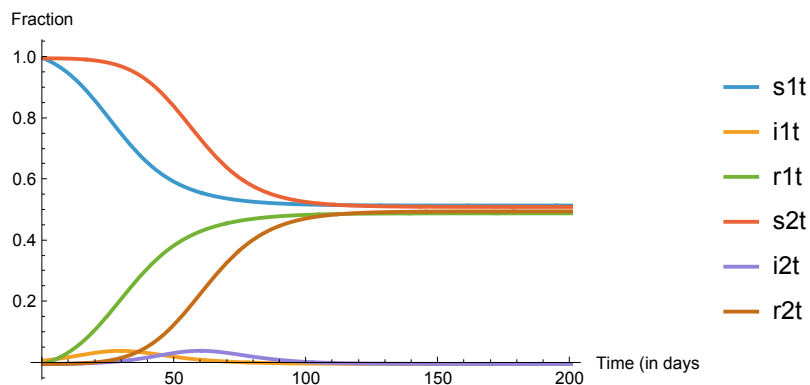
modSIR[R01_ : 4 / 3, R02_ : 4 / 3, tend_ : 200,
  i10_ : 1 / 90, i20_ : 0, m12_ : 1 / 300, m21_ : 1 / 300,  $\gamma$ _ : 1 / 4,
  s10_ : 1, r10_ : 0, s20_ : 1, r20_ : 0] := Module[{} ,
  {s1t, i1t, r1t, s2t, i2t, r2t} = NDSolveValue[{s1'[t] == -R01 *  $\gamma$  * i1[t] * s1[t],
    i1'[t] == R01 *  $\gamma$  * i1[t] * s1[t] - ( $\gamma$  + m21) * i1[t] + m12 * i2[t],
    r1'[t] ==  $\gamma$  * i1[t],
    s2'[t] == -R02 *  $\gamma$  * i2[t] * s2[t],
    i2'[t] == R02 *  $\gamma$  * i2[t] * s2[t] - ( $\gamma$  - m12) * i2[t] + m21 * i1[t],
    r2'[t] ==  $\gamma$  * i2[t],
    i1[0] == i10, s1[0] == s10, r1[0] == r10,
    s2[0] == s20, r2[0] == r20, i2[0] == i20},
    {s1[t], i1[t], r1[t], s2[t], i2[t], r2[t]}, {t, 0, tend}];
  Plot[{s1t, i1t, r1t, s2t, i2t, r2t}, {t, 0, tend /. values},
    PlotLegends -> {"s1t", "i1t", "r1t", "s2t", "i2t", "r2t"},
    AxesLabel -> {"Time (in days", "Fraction"}]]

```

In[332]:=

modSIR[]

Out[332]=

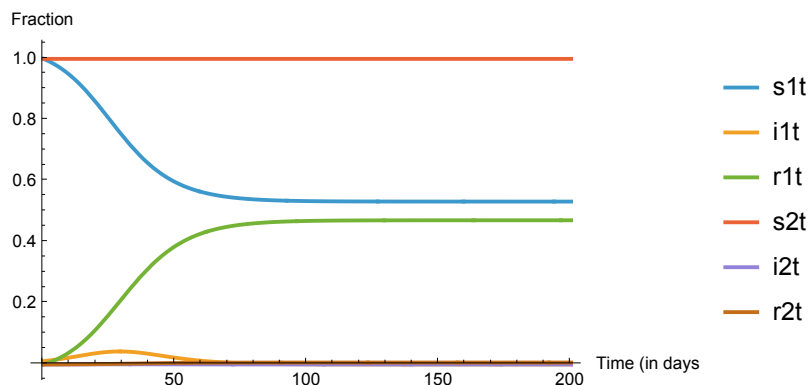


(e,2) Call modSIR with R01=4/3 and R02=0

In[333]:=

modSIR[4 / 3, 0]

Out[333]=

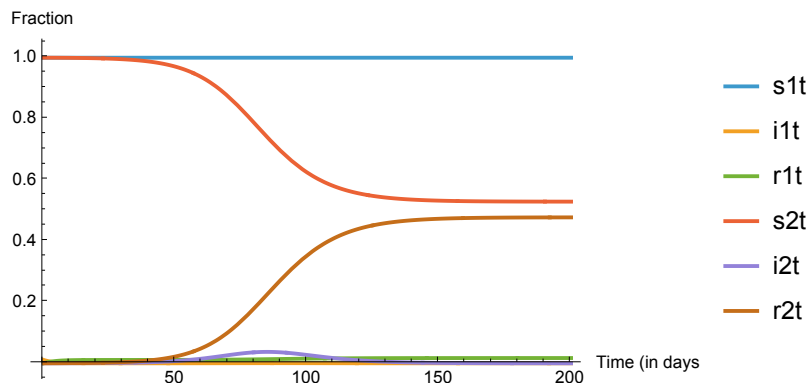


(f,2) Call modSIR with R02=4/3 and R01=0

In[334]:=

`modSIR[0, 4 / 3]`

Out[334]:=



(g,5) Describe your conclusion from the 3 previous plots (about 50 words)
(conclusion:)

From the first plot we can assume that there are two groups of people namely group1 and group2 the disease is introduced only in 1. Due to migration between the two groups the infection spreads from 1 to 2, as can be seen with the staggered rise of $i1t$ which peaks earlier than $i2t$, both eventually subside. From the next two plots we observe the reproduction number as the major factor in determining whether a disease persists in a population, in the second plot $R02$ is considered to be zero hence there is no infection as any infection which may enter group 2 due to migration from group 1 is too slow to cause infections. For the third plot the infected person migrates to group2 before the disease can spread in group1 hence we only find a peak in infections in group 2 instead of group 1.

(h,10) Write a function called `modSIRend` which is a modification of `modSIR` that returns a list with the values of $r1$ and $r2$ at time $tend$, and call these $r1end$, $r2end$, i.e. the fraction of the populations that has been infected and either recovered or deceased.

In[335]:=

```
modSIRend[R01_ : 4 / 3, R02_ : 4 / 3, tend_ : 200,
  i10_ : 1 / 90, i20_ : 0, m12_ : 1 / 300, m21_ : 1 / 300,  $\gamma$ _ : 1 / 4,
  s10_ : 1, r10_ : 0, s20_ : 1, r20_ : 0] := Module[{},
  {s1t, i1t, r1t, s2t, i2t, r2t} = NDSolveValue[{s1'[t] == -R01 *  $\gamma$  * i1[t] * s1[t],
    i1'[t] == R01 *  $\gamma$  * i1[t] * s1[t] - ( $\gamma$  + m21) * i1[t] + m12 * i2[t],
    r1'[t] ==  $\gamma$  * i1[t],
    s2'[t] == -R02 *  $\gamma$  * i2[t] * s2[t],
    i2'[t] == R02 *  $\gamma$  * i2[t] * s2[t] - ( $\gamma$  - m12) * i2[t] + m12 * i1[t],
    r2'[t] ==  $\gamma$  * i2[t],
    i1[0] == i10, s1[0] == s10, r1[0] == r10,
    s2[0] == s20, r2[0] == r20, i2[0] == i20},
    {s1[t], i1[t], r1[t], s2[t], i2[t], r2[t]}, {t, 0, tend}];
  r1end = r1t /. {t -> tend};
  r2end = r2t /. {t -> tend};
  {r1end, r2end}]

modSIRend[]
```

Out[336]:=

`{0.492887, 0.499395}`

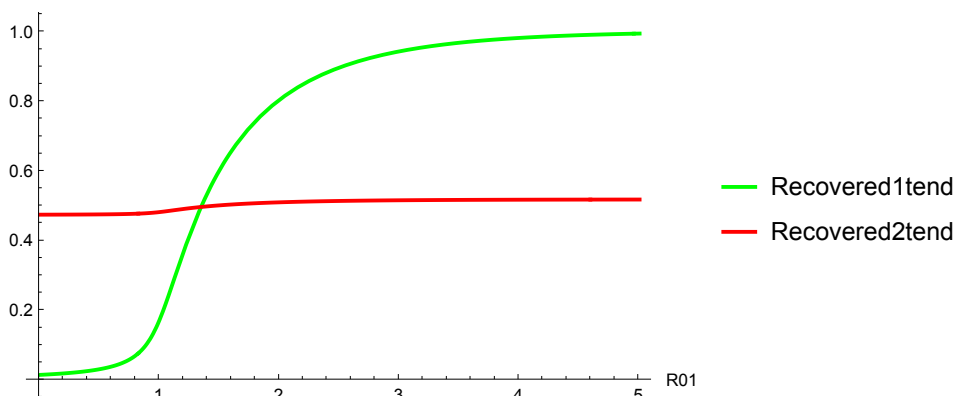
(i,5) Now make a plot of $r1end$ and $r2end$ with $R01$ from 0 to 5, using **different** colors. All other arguments are equal to their default values.

Here and in the plots below use the PlotOption `PlotLegends→{"Recovered1tend", "Recovered2tend"}` and label your axes automatically.

In[350]:=

```
Plot[{modSIRend[R01][[1]], modSIRend[R01][[2]]}, {R01, 0, 5}, AxesLabel→Automatic,
PlotLegends → {"Recovered1tend", "Recovered2tend"}, PlotStyle → {Green, Red}]
```

Out[350]=

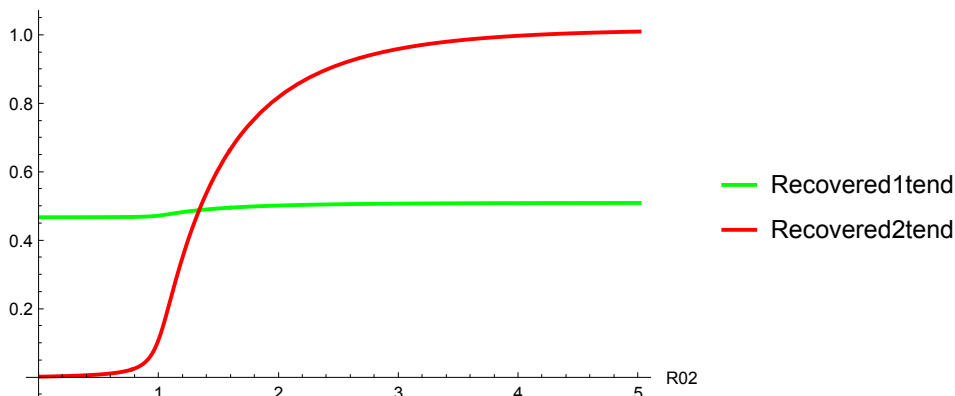


(j,2) Now make a plot of $r1end$ and $r2end$ with $R02$ from 0 to 5, using **different** colors. All other arguments are equal to their default values.

In[352]:=

```
Plot[{modSIRend[4 / 3, R02][[1]], modSIRend[4 / 3, R02][[2]]},
{R02, 0, 5}, AxesLabel → Automatic,
PlotLegends → {"Recovered1tend", "Recovered2tend"}, PlotStyle → {Green, Red}]
```

Out[352]=

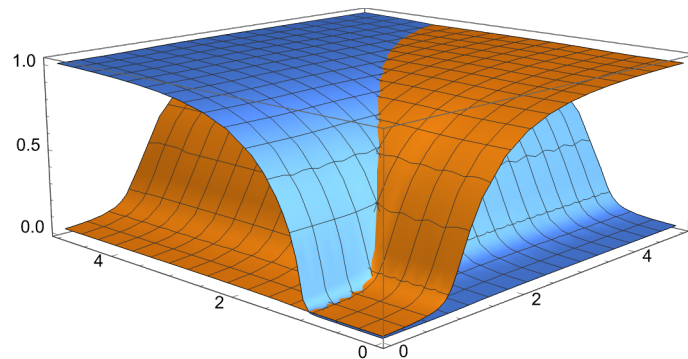


(k,3) Now make a 3D plot of $r1end$ and $r2end$ with both $R01$ and $R02$ from 0 to 5, using **different** colors automatically (so you will get the default colours). All other arguments are equal to their default values.

In[353]:=

```
Plot3D[{modSIRend[R01, R02][[1]], modSIRend[R01, R02][[2]]}, {R01, 0, 5}, {R02, 0, 5}]
```

Out[353]=



(l, 5) Describe your conclusion from the 3 previous plots (about 50 words)
(conclusion:)

According to the three plots above it is evident that the value of R influences how fast people recover from the disease, in the first example where R_2 was kept the same, we saw minor differences in the number of recovered people at the end of the simulation in group 2 however for group 1 as the R_1 value increase the fraction of recovered population was very low which dramatically increased around an R value of 1 which can be considered as the critical point. We can apply the same hypothesis and it matches up with figure 2. And in figure 3 after both R_1 and R_2 are bigger than 1 we see fraction of recovered individuals for both groups tend to 1.

(m,10) Define a function called *R0lockdown* (with five arguments: t , $t_{\text{lockdown1}}$, $t_{\text{lockdown2}}$, R_0 , $R_0\text{lockdown}$) using Piecewise to model a temporary lockdown. if $t_{\text{lockdown1}} < t < t_{\text{lockdown2}}$ then R_0 equals $R_0\text{lockdown}$, else it equals R_0 . The default values for the arguments $t_{\text{lockdown1}}$, $t_{\text{lockdown2}}$, R_0 , $R_0\text{lockdown}$ are 50, 100, 4/3, 2/3

In[362]:=

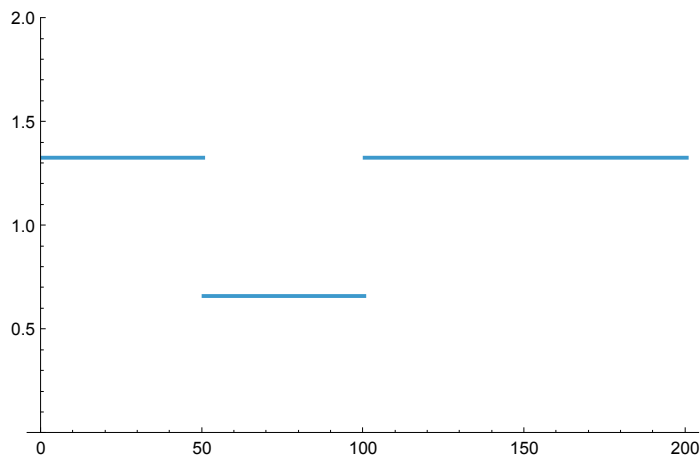
```
R0lockdown[t_, tlockdown1_ : 50, tlockdown2_ : 100, R0_ : 4 / 3,
  R0lockdown_ : 2 / 3] := Piecewise[{{R0lockdown, tlockdown1 < t < tlockdown2},
  {R0, tlockdown1 > t}, {R0, tlockdown2 < t}}]
```

Make a plot of *R0lockdown* until 200

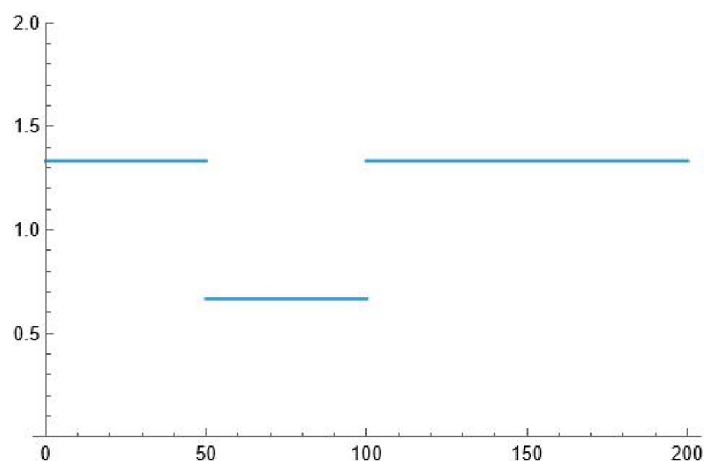
In[364]:=

```
Plot[R0lockdown[t], {t, 0, 200}, PlotRange -> {0, 2}]
```

Out[364]=



Your plot should look like this

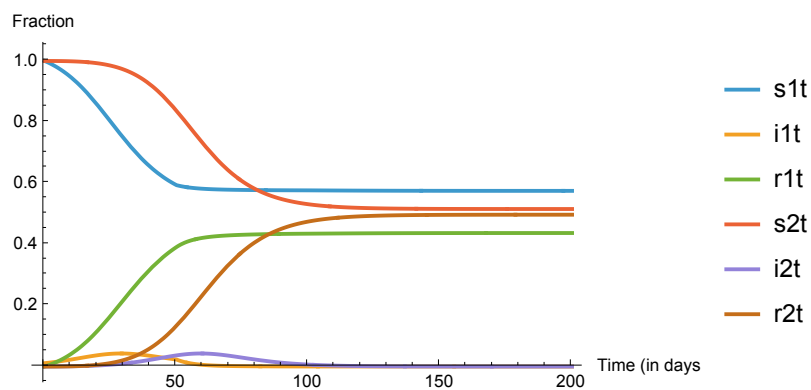


(n,2) Call modSIR with $R01=R0lockdown[t]$ and $R02=4/3$

In[365]:=

```
modSIR[R0lockdown[t], 4 / 3]
```

Out[365]=



(o,2) Call modSIR with $R01=R0lockdown[t,10,60]$ and $R02=4/3$

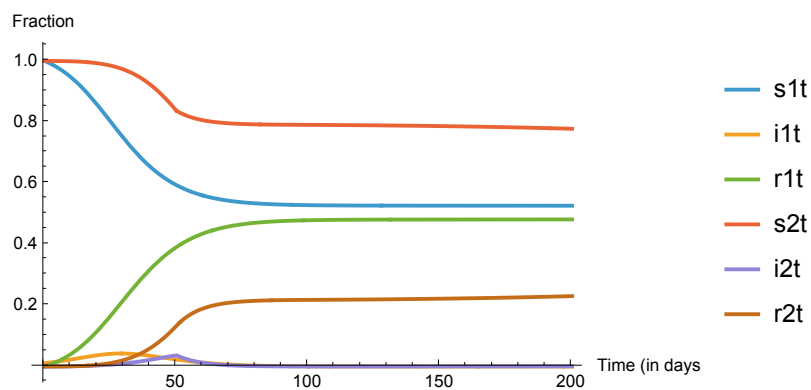
In[370]:=

(p,2) Call modSIR with $R02=R0lockdown[t]$ and $R01=4/3$

In[366]:=

```
modSIR[4 / 3, R0lockdown[t]]
```

Out[366]=



(q,2) Call modSIR with $R02=R0lockdown[t,10,60]$ and $R01=4/3$

In[369]:=

(r,5) Describe your conclusion from the 4 previous plots (about 50 words)
(conclusion:)

From the previous plots we can infer that implementing a lockdown helps reduce the spread from the initial infected group (group1) to the non infected group(group2).as the peak in figure 3 is lower and sharper than in figure 1 indicating less infections among group 2 . Implementing lockdown in group 1 doesnot help reduce the number of infections in group 2 however implementing lockdown in group 2 also seems to have an inconsequential effect on group1. The recovery rate tends to be better when lockdown is not implemented in the groups. When we implement lockdown in group 2 the number of susceptible population increases highlighting that most of the population hasnt come in contact with the disease.