-2021 年度卒業研究レポート-

感染症モデル

明治大学　総合数理学部　現象数理学科
4 年 2 組 77 番 平田謙志朗
2022 年 2 月 24 日

目 次

1　はじめに

地球全体で流行しているCOVID-19は今なお多く見られ、その原因となるコロナウイルスが変異を続け、三年目に突入した。それに伴って日本政府にも特別なコロナウイルス対策班が設置され、多くの専門家が様々なまん延防止重点措置や緊急事態宣言などの対策を取ってきた。専門家は対策を行う理由として様々な数値を提示しそれを根拠にしている。それらは、コンピュータによる数値解析により出されたものである。日本で最初に流行が起きた時、当時北海道大学教授であった西浦博先生によって、何も対策を取らなければ、日本の全人口の8割感染、42万人死亡するという説がコンピュータの数値解析により提唱された。マスクなどを始めとする感染症対策が行われたため、被害は抑えられたが、その結果、西浦博先生による解析結果が大げさすぎたのではないかと批判をする人が出てきた。専門家たちはそれなりの根拠を持って、ある程度妥当な結果を出しているが、それは一般の人にはなかなか伝わりにくい。今回この著者は、西浦博先生の研究データや資料を参考にして再現し、どのような根拠からこの説が出されたのかを確認し、実際に起こっている現象の一部を再現することにより、簡単ではあるが解説を行い、知ろうとするものにとってわかりやすい資料となれば良いと考えている。

2　西浦博先生によるコロナウイルス感染症モデルの数値解析再現

2.1　SIRモデルについて

西浦博先生によるコロナウイルス感染症モデルには、1927年に先行研究[1]されたSIRモデルという感染症モデルが用いられている。

$$\begin{pmatrix} \frac{dS}{dt} \\ \frac{dI}{dt} \\ \frac{dR}{dt} \end{pmatrix} = \begin{pmatrix} -\beta SI \\ \beta SI - \gamma I \\ \gamma I \end{pmatrix} \tag{1}$$

Sは感受性者数（感染する可能性のある人$Susceptible$)、Iは感染者数（感染中の人$Infectious$)、Rは回復者数（感染状態から回復もしくは死に至った人$Recovered$）

である。β は定数であり、感染系数や感染率と呼ばれ、[2] の説明では「単位時間内に人口一人当たりで接触が生じる率」を意味し、

$$\beta = \frac{\gamma R_0}{N} \tag{2}$$

で定義され、感染のしやすさを表している。

γ は回復率と呼ばれ、[2] の説明では「1 人の I が (自然) 治癒する単位時間当たりの率」を意味する。R_0 は基本再生産数といい、(2.3) で詳しく説明されている。N は総人口である。

この SIR モデルでは、すべての人が感受性者 (S) で、その中に一定数の感染者 (I) が存在するところからはじまる。感染者は感染率 β にしたがって増えて行き、従って、感受性者は減り、回復率 γ にしたがって回復者が増えて行く。感染者はある時点までは増えて行くがその時点を過ぎると減っていく。その様子が図 1 により確認することができる。

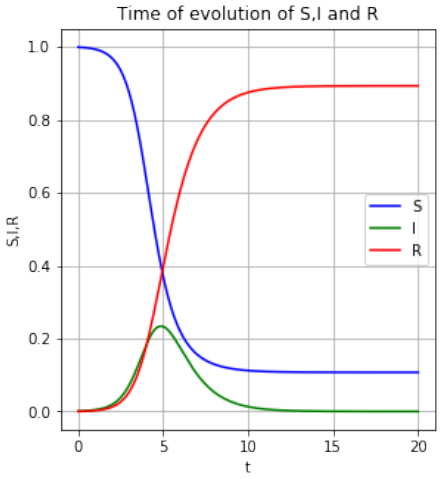

図 1: S, I, R の時間の流れによる変化 (簡易版) ※付録

この SIR モデルを用いることにより、図 1 のように、ある感染症が起こった場合の S、I、R の変化を時間経過とともに見ることができる。西浦博先生はこの SIR モデルを基礎にして解析を行っている。

2.2 年齢構造化モデル

2020 年、4 月 15 日西浦博先生らによって提唱された 8 割感染 42 万人死亡説 [3] が、どのようなシミュレーションから導かれたかを見ていくことにする。

西浦博先生によるプログラムの内容 [4] を Python に置き換えてシミュレーションを行った。その中で使われていた微分方程式は SIR モデルに加え、年齢構造化モデルを用いていた。それが以下である。

$$
\begin{pmatrix}
\frac{dS_c}{dt} \\
\frac{dI_c}{dt} \\
\frac{dR_c}{dt} \\
\frac{dS_a}{dt} \\
\frac{dI_a}{dt} \\
\frac{dR_a}{dt} \\
\frac{dS_e}{dt} \\
\frac{dI_e}{dt} \\
\frac{dR_e}{dt}
\end{pmatrix}
=
\begin{pmatrix}
-\beta_c S_c I \\
\beta_c S_c I - \gamma I_c \\
\gamma I_c \\
-\beta_a S_a I \\
\beta_a S_a I - \gamma I_a \\
\gamma I_a \\
-\beta_e S_e I \\
\beta_e S_e I - \gamma I_e \\
\gamma I_e
\end{pmatrix}
\tag{3}
$$

(※ $I = I_c + I_a + I_e$)

この年齢構造化モデルで西浦先生は人口を、15 歳未満の子ども ($childs$)、15 〜 65 歳未満の生産年齢人口 ($adults$)、65 歳以上の高齢者 ($elderly\ people$) の三つで分けている。一般的に、子供は人口が少なく感染しにくく重症化もしにくい、生産年齢人口は多く感染しやすい、高齢者は人数が少なく感染しやすく重症化もしやすい。そういった年齢による違いを反映させるためこの構造化モデルが採用されている。

2.3 初期値とパラメータ

はじめに、下付き文字についての説明をしておく。S_*, I_*, R_* などの $* = c, a, e$ は $childs, adults, elderly\ people$ の頭文字で、例えば S_c は子供の感受性者数を表して

いる。

変数の初期値は西浦博先生によるプログラムの内容 [4] に従っている。初期値を示すと、$t = 0$ のとき、$S_c = 15758424, S_a = 76499818, S_e = 35185241$、$I_c = 0, I_a = 10$(初めに生産年齢人口から 10 人の感染者が現れたとする。$), I_e = 0$、$R_c = 0, R_a = 0, R_e = 0$ となっている。パラメータについて示すと、β は、

$$\beta_c = \frac{\gamma R_0 \alpha_c}{N_c}\ , \quad \beta_a = \frac{\gamma R_0 \alpha_a}{N_a}\ , \quad \beta_e = \frac{\gamma R_0 \alpha_e}{N_e} \tag{4}$$

で定義される。

ここで、γ は、

$$\frac{1}{\gamma} = (\text{平均世代時間}) \tag{5}$$

で定義される。

ここで平均世代時間とは、一人の感染者が感染してからその二次感染者が感染するまでの期間 (日) であり、プログラムでは 4.8(日) となって いる。従って、γ は平均世代時間の逆数となる。平均世代時間の値については、[3] で西浦博先生自身の研究をもとに定められていることが説明されている 。

R_0 は基本再生産数といい、1 人の感染者が生み出す 2 次感染者の平均値である。プログラムでは $R_0 = 2.5$ となっており、その値については、[3] で説明されている。西浦博先生によって研究が行われた当時は、日本で流行が起こり始めていた頃であったため、日本における十分なデータが得られず基本再生産数が定量化できていなかった。しかし、その時にはすでに欧州諸国の基本再生産数の推定値が得られており、それが 2〜3 の間にあることに基づいて、便宜的にドイツにおける基本再生産数の推定値である 2.5 を採用している ([3])。

α は早期にデータが集まっていた武漢の 2 月初旬までの重篤患者数と同程度の比 (子どもはほぼゼロ、生産年齢人口：高齢者で概ね 1：3 から 1：4 程度）になるように導入されている。$(\alpha_c = 0.009, \alpha_a = 0.630, \alpha_e = 0.361, \alpha_c + \alpha_a + \alpha_e = 1)$, N は総人口 ($N = 127443493, N_c = 15758424, N_a = 76499828, N_e = 35185241$。プログラムより。) となっている。

以上を元に (4) に代入して計算すると、

$$\beta_c \approx 2.97 \times 10^{-10}, \quad \beta_a \approx 4.29 \times 10^{-9}, \quad \beta_e \approx 5.34 \times 10^{-9} \tag{6}$$

が得られる。

2.4　プログラムとその結果

年齢構造化モデルの Python によるプログラムの再現内容は以下となっている。(※ここに示すのは簡易的なプログラムである。プログラム全体は付録に記してある。)

(Python プログラム ※付録)

```
def sir(x, t, beta_c,beta_a,beta_e,gamma): #微分方程式の定義
  I=x[1]+x[4]+x[7]
  return [-beta_c*x[0]*I,beta_c*x[0]*I-gamma*x[1], gamma*x[1],
       -beta_a*x[3]*I,beta_a*x[3]*I-gamma*x[4], gamma*x[4],
       -beta_e*x[6]*I,beta_e*x[6]*I-gamma*x[7], gamma*x[7]]
beta_c = (gamma*R0/Nc)*  alpha_c #$\beta$の計算
beta_a = (gamma*R0/Na)*  alpha_a
beta_e = (gamma*R0/Ne)*  alpha_e
x0=[S_c,I_c,R_c,S_a,I_a,R_a,S_e,I_e,R_e] #初期値
n=1000
t=np.linspace(0.0, 100.0, n+1)
x=odeint(sir, x0, t, args=(beta_c,beta_a,beta_e,gamma)) #微分方程式の計算
```

プログラムの結果、時間による S,I,R の値の変化は図 2 のようになった。

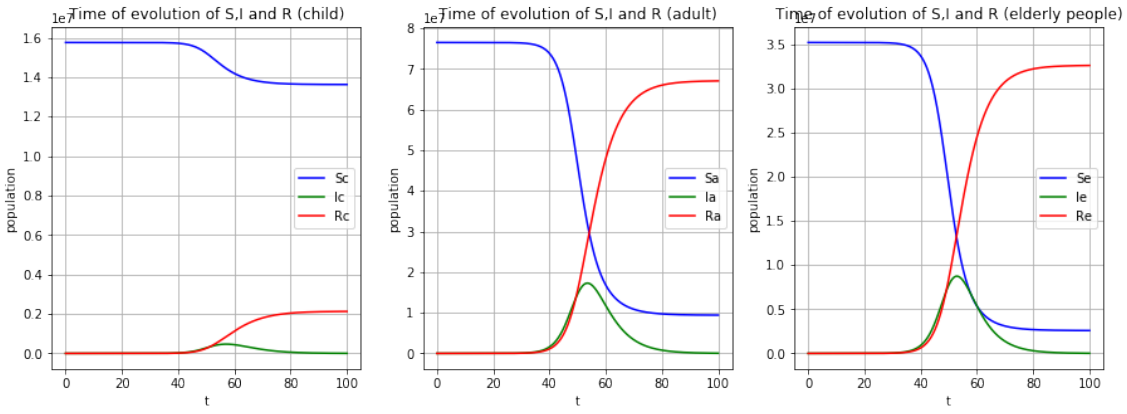

図 2: 各人口の時間の流れ ※付録

また、$t = 100$ の時の総感染者数は、子供の総感染者数 $\approx 2,127,305$、大人の総感染者数 $\approx 67,045,211$、高齢者の総感染者数 $\approx 32,583,993$、総感染者数 $\approx 101,756,510$

となっている。総人口のうち総感染者数の割合が約 7.9 割なので、総人口の約 8 割が感染することがわかる。

2.5　死者数計算

致死率はIFR(*Infection Fatality Risk*)で表し、$IFR_c = 0$、$IFR_a = 0.0015$、$IFR_c = 0.01$ となる。

(Python プログラム続き ※付録)

```
Death_c=x[n,2]*IFR_c ＃年齢別の死者数の計算
Death_a=x[n,5]*IFR_a
Death_e=x[n,8]*IFR_e
Death=Death_c+Death_a+Death_e ＃総死者数の計算
```

上記プログラムの結果、子供、大人、高齢者の死者数は 0 人、約 100568 人、約 325840 人。総死者数は約 426408 人となり、何も流行対策を取らなければ日本国民の約 42 万人が死亡することになるということがわかる。

2.6　流行曲線

年代別の流行曲線を以下に示す。流行曲線は、縦軸が、単位時間に新規感染者がどれだけ増えるかを表しており、横軸が時間 (日) を表しているため、その日の新規感染者数を表している。

図 3 の右のグラフは、西浦 [3] に提示されていた流行曲線である。その再現を著者がしたものが左の図である。ここでは総人口を 10 万人としていてそれに合わせて、各人口の値も変化させている。こうしてみてみると、西浦博先生の解析したデータと、著者が再現したデータがほぼ一致していることがわかる。このことから、著者による再現はほぼ正しく行われているように思える。

3　流行規模の決定

感染症の流行が起こると毎回感染者数の最大値があり、その最大値までは感染者数が増加するが、そこに達すると、感染者数は減少し、ほぼ 0 人になる。その最大値や、流行が起こってからおさまるまでの期間は何の値で決まっているのかを調べていく。変化を簡易的に見るために年齢構造化モデルを採用しない SIR モ

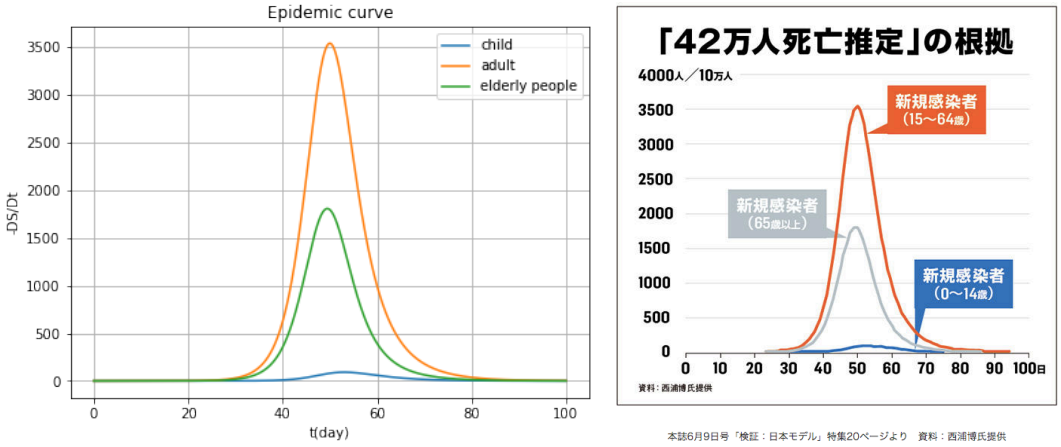

図 3: 流行曲線 (左, 著者による再現、右, 西浦博先生によるデータ ※付録)

デルを用いている。図 4 に示したのは、感染者の初期値 10 人、$\gamma = \frac{1}{4.8}$、$R_0 = 2.5$ の時の結果である。(ここで覚えておいてほしいところは横軸が 1〜100 となって

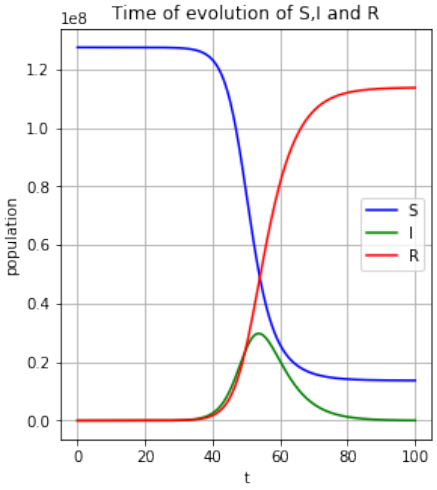

図 4: 感染者の初期値 10 人、$\gamma = \frac{1}{4.8}$、$R_0 = 2.5$ ※付録

いるところである。)

3.1　初期値の変化による違い

ここでは感染者の初期値を変化させてその結果を見ていく。感染者の初期値を $10^3, 10^5, 10^7$ と変えていき、図 4 とともに比べる。($\gamma = \frac{1}{4.8}$、$R_0 = 2.5$)

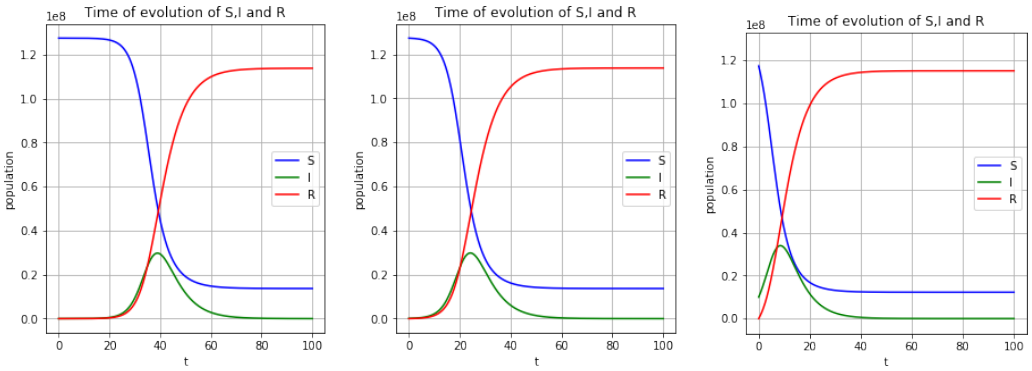

図 5: 左から $10^3, 10^5, 10^7$(人) ※付録

図 4 と図 5 を比べてみると、感染者の初期値が増えると、流行が起こるのが早くなることがわかる。実際、感染者の初期値が $10^3, 10^5, 10^7$ の時、総感染者数が 113,752,978 人、113,775,496 人、115,181,596 人となり、特に感染者の初期値が 1000 万人の時に総感染者がさらに増えていることがわかる。このことから、一度に多くの感染者が現れると、流行の波も大きくなることがわかる。

3.2 γ の変化による違い

ここでは γ の値を変化させてその結果を見ていく。γ の値を $\frac{1}{2}$ 倍 $(\gamma = \frac{1}{9.6})$ した時と、2 倍 $(\gamma = \frac{1}{2.4})$ した時とを図 4 とともに比べる。(感染者の初期値 10、$R_0 = 2.5$)

図 4 と図 6 を比べてみると、γ の値を $\frac{1}{2}$ 倍すると、回復するのが遅くなるため、全体的に流行する期間が 2 倍となっている。逆に、γ の値を 2 倍すると、回復するのが早くなるため、全体的に流行する期間が $\frac{1}{2}$ 倍となっている。また、この結果だけを見ると、γ の値の変化だけでは総感染者数に変化は見られないため、早期回復に力を入れても、感染する人を減らすことはできないことがわかる。しかし、流行する期間を縮めることができるため、早く感染症から脱却するためには、早期回復に力を入れることは意味があることではある。

3.3 R_0 の変化による違い

ここでは R_0 の値を変化させてその結果を見ていく。R_0 の値を 1.5 にした時と、R_0 の値を 3.5 にした時とを図 4 とともに比べる。(感染者の初期値 10、$\gamma = \frac{1}{4.8}$)

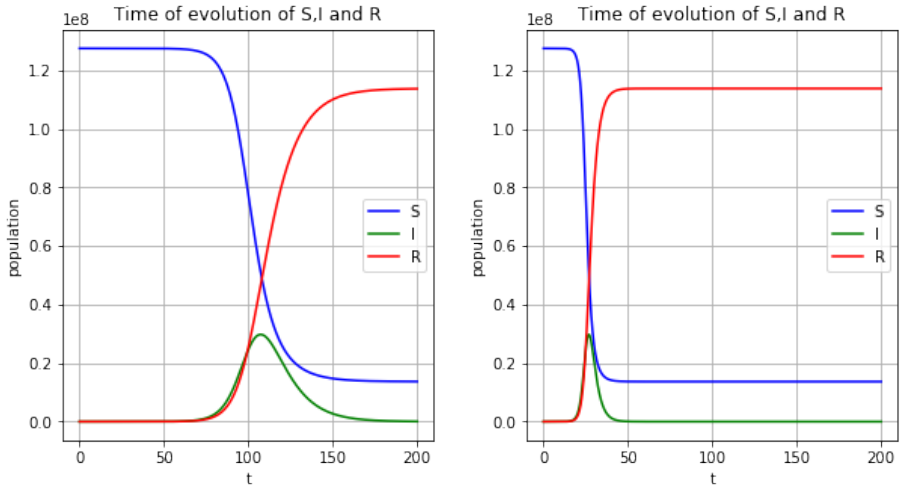

図 6: 左から $\gamma = \frac{1}{9.6}$、$\gamma = \frac{1}{2.4}$ ※付録

図 4 と図 7 を比べてみると、R_0 の値が大きくなれば流行する期間が短くなり、感染者も増えることがわかる。基本再生産数 R_0 の値が増すということは、一般的に人から人への感染が多くなってくるということを表している。R_0 が 1.5 の時と R_0 が 2.5 の時を比べると感染者数の増え方に大きな差があるが、R_0 が 2.5 の時と R_0 が 3.5 の時を比べると感染者数の増え方にあまり差がないように見える。実際に総感染者数は R_0 が 1.5 の時、約 73,266,179 人、R_0 が 2.5 の時、約 113,761,767 人、R_0 が 3.5 の時、約 123,108,473 人となり、R_0 が 1.5 から 2.5 に変化する時よりも、2.5 から 3.5 に変化する時の方が総感染者数の増え幅が狭くなってることがわかる。したがって、流行期間全体にわたって基本再生産数を下げれば下げる程、総感染者数を減らすことに意味が出てくる。

4　出入国者の影響

流行がおさまったと見えても、実際には感染者数が 0 人になることはほとんど無い。しかし、SIR モデルの解析では流行がおさまると感染者数がある時点から必ず 0 人になる。この差は何によるものなのかを考えた時に、外国からの侵入が原因ではないのかと考えた。外国からの侵入、つまり、入国者の存在である。入国者数はコロナウイルスが流行っていた期間中、著者の計算 [5][6][7] によると、毎日約 2190 人いることがわかっている。これらをもとにし、仮に $SIR-a$ モデルとなずけて微分方程式を更新してみた。$SIR-a$ モデルでは、簡易的に入国者と出国者の数を同数にし、毎日 a 人の出入りがあることにする。さらに、よく日本に入

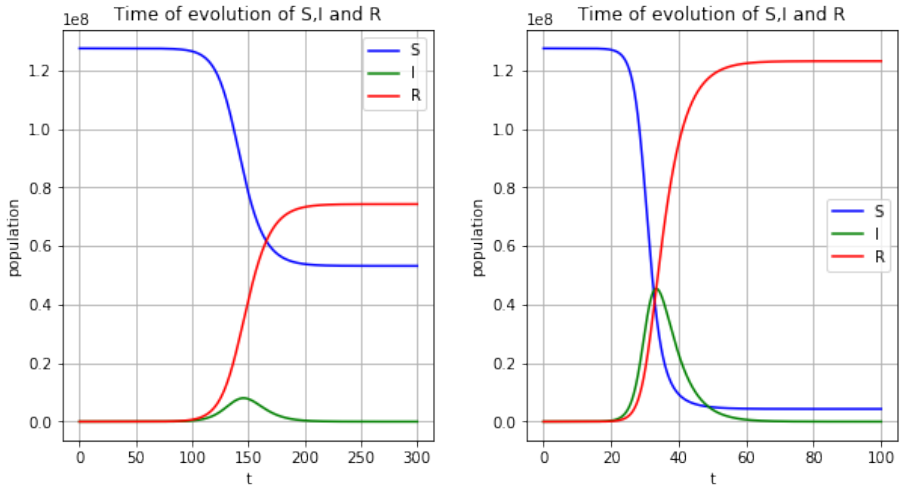

図 7: 左から $R_0 = 1.5, R_0 = 3.5$ ※付録

国するする国の人々と日本の人々とでは感染者の割合がかなり異なっているため、そこに差をつけるため f と j というパラメータを用意した。$f(foreigner)$ は入国者のうちの感受性者の割合で、fa 人が感受性者で $(1-f)a$ 人が感染者となっており、$j(japan)$ は出国者のうちの感受性者の割合で、ja 人が感受性者で $(1-j)a$ 人が感染者となっている。一般的によく日本に入国する国の人々の感染者の割合は、日本の人々の感染者の割合よりもかなり高い。また、$SIR-a$ モデルでは、出入国者に回復者がいないことを前提とし、一度入国したものは日本の感染者の割合に順応することにする。以上を考慮して $SIR-a$ モデルを更新し (7) に定めた。

$$(7) \qquad \begin{pmatrix} \frac{dS}{dt} \\ \frac{dI}{dt} \\ \frac{dR}{dt} \end{pmatrix} = \begin{pmatrix} -\beta SI + fa - ja \\ \beta SI - \gamma I + (1-f)a - (1-j)a \\ \gamma I \end{pmatrix}$$

(7) にしたがって書き換えたプログラムを以下に示す。プログラムでは、$a = 2190, f = 0.9999, j = 0.99$ としている。プログラム実行の結果は図 8 の左に示す。

(Python プログラム ※付録)

```
def sir(x, t, beta,gamma,a):
```

```
    f=0.99
    j=0.9999
    return [-beta*x[0]*x[1]+a*f-a*j,
          beta*x[0]*x[1]-gamma*x[1]+a*(1-f)-a*(1-j),
          gamma*x[1]]
a=2190
```

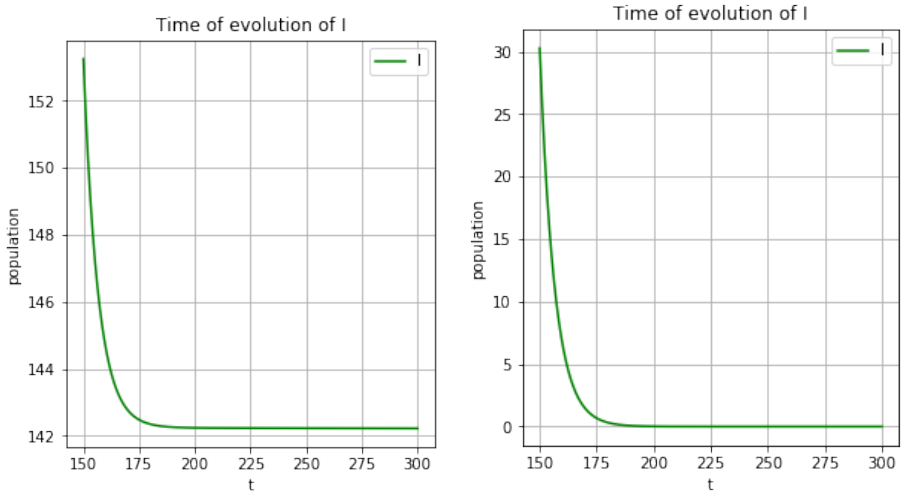

図 8: 出入国者あり (左) なし (右) ※付録

$I(t) < 1$ となれば、実質的に $I(t) = 0$ と考えることにする。図 8 の左右を比べると、感染者数が 0 人になる場合とならない場合とが再現されていることがわかる。図 8 の左が日本における流行の第 1 波、第 2 波などの波の間に起こる現象を再現している。流行がおさまったと見えても、実際は少しずつではあるが感染者が毎日出ているのである。これらのことから、感染症の流行が起き、完全に流行を抑えたい場合は入国者を 0 人にすることで感染者を 0 人にする可能性を見出すことができることがわかった。

5 時間による R_0 の変化の違い

緊急事態宣言、まん延防止重点措置などの対策をとると、本来、基本再生産数 R_0 の値は、下がっていく。このような対策は感染症の流行が起こり始めてから行われ、流行がおさまると解除される。したがって、今までのプログラムを書き換え、$t = 45$ から $t = 70$ まで $R_0 = 1.5$ にして計算した。書き換えたプログラムは以

下のようになっている。

(Python プログラム ※付録)

```
def R0(t):
        if 0.0<=t<45.0:
            return 2.5
        elif 45.0<=t<70.0:
            return 1.5
        else:
            return 2.5
def sir(x, t, beta,gamma):
    beta = (gamma*R0(t)/N)
    return [-beta*x[0]*x[1],beta*x[0]*x[1]-gamma*x[1], gamma*x[1]]
```

R_0 を関数として、プログラムに組み込むことにした。一般的に緊急事態宣言、まん延防止重点措置などの対策は基本的に流行が起こってから発出される。したがって、$t = 45$ の時点から R_0 の値を下げている。また、それらの対策は、流行がおさまると解除される。ここについてはあまり厳密にはしていないが、$t = 70$ の時点から R_0 の値を 2.5 に戻している。上記プログラムの結果は図 9 に示す。

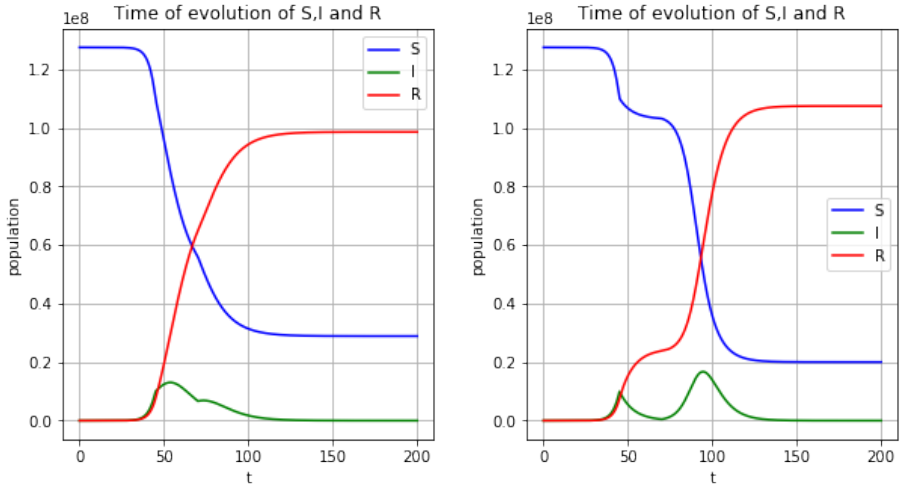

図 9: $t = 45$ から $t = 70$ まで $R_0 = 1.5$(左)、$R_0 = 0.5$(右)、それ以外 $R_0 = 2.5$ ※付録

図 4 と図 9 を比べてみると、R_0 の値を下げると、そのままの時より感染者数が減っていることがわかる。特に図 9 の右を見ると、R_0 の値を 0.5 に下げた時点で、

感染者数が急激に下がり、R_0 の値をもとに戻すと、感染者数が急激に上がり、いわゆる、流行の第 1 波、第 2 波と呼ばれるような結果を確認することができる。総感染者数にも変化があり、$R_0 = 1.5$ に下げた時は約 98,584,161 人、$R_0 = 0.5$ に下げた時は約 107,442,835 人であった。何の対策を取らなかった場合の総感染者数の約 113,761,767 人と比べると数は減っているが、$R_0 = 1.5$ に下げた時の方が $R_0 = 0.5$ に下げた時よりも少ないことを考えると、ただただ R_0 を下げると言う考え方では、総感染者数を減らすと言う意味においては、あまり良くないということがわかる。これらの結果を見る限り、できるだけ継続して対策を行うことが必要であることがわかる。

6　まとめ

西浦博先生の記事から日本国内 8 割感染 42 万人死亡説がどのように出されたのかを Python の再現により確認することができた。また、様々な変数の値を変化させることによって得られた結果から感染症の流行の対策をする意味を確認することができた。今回の研究のように、一つ一つ確認していくことができれば日本の専門家が何を根拠にして数値を提示しているのかがわかる。こういった専門家の資料を噛み砕いて説明するような資料は以外と少ないのかもしれない。労力は少しかかるが感染症の流行を知らない後の世代のためにもこの資料を残しておきたい。

謝辞

本来 2 年間で済んだところ個人的事情で 4 年間もお世話になり、途中体調を崩した時にも忙しい中、個人的サポートを行ってくださった桂田祐史先生に感謝申し上げます。体調を崩した時に精神的サポートを行ってくれた友人 N.S に感謝申し上げます。ギリギリではありましたが無事に卒業論文を完成させることができました。お二人のますますのご発展をお祈り申し上げます。

参考文献

[1] Kermack, W. O. and McKendrick, A. G.: A Contributions to the Mathematical Theory of Epidemics, *Proceedings of the Royal Society of London. Series A*, Vol. 115, No. 772, pp. 700 – 721 (1927).

[2] 西浦博　編著：感染症疫学のためのデータ分析入門, 金芳堂 (2021).

[3] Newsweek 日本版, 特別寄稿「8 割おじさん」の数理モデルとその根拠—西浦博・北大教授,
https://www.newsweekjapan.jp/stories/world/2020/06/8-39_1.php(2020/6/11).

[4] GitHub,contactmodel,COVID19−Japan−Reff,BerkleyMadonna_May2020.txt,
https://github.com/contactmodel/COVID19-Japan-Reff/blob/master/BerkleyMadonna_May2020.txt.

[5] 出入国在留管理庁, 公表情報, 令和 2 年のプレリリース, 令和 2 年上半期における外国人入国者数及び日本人出国者数等について, 最終閲覧日 2022/2/24,
https://www.moj.go.jp/isa/publications/press/nyuukokukanri04_00010.html.

[6] 出入国在留管理庁, 公表情報, 令和 3 年のプレリリース, 令和 2 年における外国人入国者数及び日本人出国者数等について, 最終閲覧日 2022/2/24,
https://www.moj.go.jp/isa/publications/press/nyuukokukanri13_00015.html.

[7] 出入国在留管理庁, 公表情報, 令和 3 年のプレリリース, 令和 3 年上半期における外国人入国者数及び日本人出国者数等について, 最終閲覧日 2022/2/24,
https://www.moj.go.jp/isa/publications/press/13_00020.html#:~:text=%EF%BC%93%20%E5%A4%96%E5%9B%BD%E4%BA%BA%E5%85%A5%E5%9B%BD%E8%80%85,%E3%81%BE%E3%81%97%E3%81%9F%EF%BC%88%E8%A1%A8%EF%BC%91%EF%BC%89%E3%80%82.

付録

(図 2 の結果、全人口の感染者の割合、各 β, 総感染者数, 総死者数の値、を得るためのプログラム)

```
# nishiurasir.py --- SIR model
# coding: utf-8
#https://github.com/contactmodel/COVID19-Japan-Reff/blob/master/BerkleyMadonna_May2
#https://www.newsweekjapan.jp/stories/world/2020/06/8-39_1.php
import numpy as np
from scipy.integrate import odeint
```

```
import matplotlib.pyplot as plt
import random

# x=(S,I,R)
def sir(x, t, beta_c,beta_a,beta_e,gamma):
    I = x[1]+x[4]+x[7]
    return [-beta_c*x[0]*I,beta_c*x[0]*I-gamma*x[1], gamma*x[1],
           -beta_a*x[3]*I,beta_a*x[3]*I-gamma*x[4], gamma*x[4],
           -beta_e*x[6]*I,beta_e*x[6]*I-gamma*x[7], gamma*x[7]]

Nc = 15758424
Na = 76499828
Ne = 35185241

alpha_c = 0.009
alpha_a = 0.630
alpha_e = (1-alpha_c-alpha_a)
gamma = 1/4.8
R0 = 2.5

IFR_c = 0
IFR_a = 0.0015
IFR_e = 0.0100

beta_c = (gamma*R0/Nc)*  alpha_c
beta_a = (gamma*R0/Na)*  alpha_a
beta_e = (gamma*R0/Ne)*  alpha_e

S_c = 15758424
I_c = 0
R_c = 0
S_a = 76499818
I_a = 10
R_a = 0
S_e = 35185241
I_e = 0
```

```
R_e = 0
Population = S_c+S_a+S_e

ab = beta_c
b = beta_a
c = beta_e

print('beta_c=',ab,'beta_a=',b,'beta_e=',c)

x0 = [S_c,I_c,R_c,S_a,I_a,R_a,S_e,I_e,R_e]
n = 1000
t = np.linspace(0.0, 100.0, n+1)
x = odeint(sir, x0, t, args=(beta_c,beta_a,beta_e,gamma))

Infection_c = x[n,2]
Infection_a = x[n,5]
Infection_e = x[n,8]
Infection = x[n,2]+x[n,5]+x[n,8]

a = Infection/Population*100
print('全人口の中の感染者の割合=',a)
print('子供感染者数＝',Infection_c,'大人感染者数＝',Infection_a,'高齢
者感染者数＝',Infection_e,'総感染者数＝',Infection)

Death_c = x[n,2]*IFR_c
Death_a = x[n,5]*IFR_a
Death_e = x[n,8]*IFR_e

plt.figure(figsize=(15,5))

plt.subplot(131)
plt.plot(t,x[:,0],'b', label='Sc')
plt.plot(t,x[:,1],'g', label='Ic')
plt.plot(t,x[:,2],'r', label='Rc')
plt.legend(loc='best')
plt.xlabel('t')
```

```
plt.ylabel('population')
plt.title("Time of evolution of S,I and R (child)")
plt.grid()

plt.subplot(132)
plt.plot(t,x[:,3],'b', label='Sa')
plt.plot(t,x[:,4],'g', label='Ia')
plt.plot(t,x[:,5],'r', label='Ra')
plt.legend(loc='best')
plt.xlabel('t')
plt.ylabel('population')
plt.title("Time of evolution of S,I and R (adult)")
plt.grid()

plt.subplot(133)
plt.plot(t,x[:,6],'b', label='Se')
plt.plot(t,x[:,7],'g', label='Ie')
plt.plot(t,x[:,8],'r', label='Re')
plt.legend(loc='best')
plt.xlabel('t')
plt.ylabel('population')
plt.title("Time of evolution of S,I and R (elderly people)")
plt.grid()

Death = Death_c+Death_a+Death_e

print(' 子供総死者数＝',Death_c,' 大人総死者数 ',Death_a,' 高齢者総死者
数＝ ',Death_e,' 総死者数= ',Death,'  ',)

plt.show()
```

(図 3 の結果を得るためのプログラム)

```
# nishiurasir.py --- SIR model
# coding: utf-8
#https://github.com/contactmodel/COVID19-Japan-Reff/blob/master/
BerkleyMadonna_May2020.txt
```

```
#https://www.newsweekjapan.jp/stories/world/2020/06/8-39_1.php
import numpy as np
from scipy.integrate import odeint
import matplotlib.pyplot as plt

# x=(S,I,R)
def sir(x, t, beta_c,beta_a,beta_e,gamma):
    I = x[1]+x[4]+x[7]
    return [-beta_c*x[0]*I,beta_c*x[0]*I-gamma*x[1], gamma*x[1],
           -beta_a*x[3]*I,beta_a*x[3]*I-gamma*x[4], gamma*x[4],
           -beta_e*x[6]*I,beta_e*x[6]*I-gamma*x[7], gamma*x[7]]

N = 15758424+76499828+35185241
Nc = 15758424/N*100000
Na = 76499828/N*100000
Ne = 35185241/N*100000

alpha_c = 0.009
alpha_a = 0.630
alpha_e = (1-alpha_c-alpha_a)
gamma = 1/4.8
R0 = 2.5

beta_c = (gamma*R0/Nc)*  alpha_c
beta_a = (gamma*R0/Na)*  alpha_a
beta_e = (gamma*R0/Ne)*  alpha_e

S_c = 15758424/N*100000
I_c = 0
R_c = 0
S_a = 76499818/N*100000
I_a = 10/N*100000
R_a = 0
S_e = 35185241/N*100000
I_e = 0
R_e = 0
```

```
x0 = [S_c,I_c,R_c,S_a,I_a,R_a,S_e,I_e,R_e]
n = 1000
t = np.linspace(0.0, 100.0, n+1)
x = odeint(sir, x0, t, args=(beta_c,beta_a,beta_e,gamma))

plt.figure(figsize=(15,5))

plt.subplot(121)
plt.plot(t,beta_c*x[:,0]*(x[:,1]+x[:,4]+x[:,7]),label='child')
plt.plot(t,beta_a*x[:,3]*(x[:,1]+x[:,4]+x[:,7]),label='adult')
plt.plot(t,beta_e*x[:,6]*(x[:,1]+x[:,4]+x[:,7]),label='elderly people')
plt.legend(loc='best')
plt.xlabel('t(day)')
plt.ylabel('-DS/Dt')
plt.title("Epidemic curve")
plt.grid()

plt.show()
```

(図 4,5,6,7 の結果を得るためのプログラム)

```
# testsir3.py --- SIR model
# coding: utf-8
import numpy as np
from scipy.integrate import odeint
import matplotlib.pyplot as plt

# x=(S,I,R)
def sir(x, t, beta,gamma):
    return [-beta*x[0]*x[1], beta*x[0]*x[1]-gamma*x[1], gamma*x[1]]

Iini=10
N=127443493
S = N-Iini
I = Iini
R = 0
```

```
gamma=1/4.8
R0=2.5
beta = (gamma*R0/N)

x0=[S,I,R]
n=1000
t=np.linspace(0.0, 100.0, n+1)
x=odeint(sir, x0, t, args=(beta,gamma))

plt.figure(figsize=(10,5))

plt.subplot(121)
plt.plot(t,x[:,0],'b', label='S')
plt.plot(t,x[:,1],'g', label='I')
plt.plot(t,x[:,2],'r', label='R')
plt.legend(loc='best')
plt.xlabel('t')
plt.grid()
plt.show()
```

(図 8 の結果を得るためのプログラム)

```
# nishiurasir.py --- SIR model
# coding: utf-8
import numpy as np
from scipy.integrate import odeint
import matplotlib.pyplot as plt
import random
# x=(S,I,R)
def sir(x, t, beta,gamma,a):
    f=0.99
    j=0.9999
    return [-beta*x[0]*x[1]+a*f-a*j,beta*x[0]*x[1]-gamma*x[1]+a*(1-f)-a*(1-j),
     gamma*x[1]]

a=2190
N=127443493
```

```
gamma=1/4.8
R0 =2.5
beta = (gamma*R0/N)
ini=10
S = N-ini
I = ini
R = 0

x0=[S,I,R]
n=2000
tmax=300
tini=150
t=np.linspace(0, tmax, n+1)
x=odeint(sir, x0, t, args=(beta,gamma,a))

Infection=x[n,2]
a=Infection/S*100
print(' 全人口の中の感染者の割合=',a)
print(' 総感染者数＝',Infection)

nini=int(tini/tmax*n)
plt.figure(figsize=(15,5))

plt.subplot(131)
plt.plot(t[nini:],x[nini:,1],'g', label='I')
plt.legend(loc='best')
plt.xlabel('t')
plt.ylabel('population')
plt.title("Time of evolution of I")
plt.grid()

print(x[n])

print(' 総死者数= ',Death)

plt.show()
```

(図 9 の結果を得るためのプログラム)

```
# nishiurasir.py --- SIR model
# coding: utf-8
import numpy as np
from scipy.integrate import odeint
import matplotlib.pyplot as plt

def R0(t):
        if 0.0<=t<45.0:
            return 2.5
        elif 45.0<=t<70.0:
            return 2.5
        else:
            return 2.5

# x=(S,I,R)
def sir(x, t,alpha,gamma):
    beta = (gamma*R0(t)/N)*alpha
    return [-beta*x[0]*x[1],beta*x[0]*x[1]-gamma*x[1], gamma*x[1]]

N = 127443493
alpha = 1
print('総人口=',N)
gamma = 1/4.8

Iini = 10
S = N-Iini
I = Iini
R = 0

x0 = [S,I,R]
n = 2000
tmax = 200
tini = 0
t = np.linspace(0, tmax, n+1)
```

```
x = odeint(sir, x0, t, args=(alpha,gamma))

Infection = x[n,2]
a = Infection/S*100
print(' 全人口の中の感染者の割合=',a)

print(' 総感染者数＝',Infection)

nini = int(tini/tmax*n)
plt.figure(figsize=(15,5))

plt.subplot(131)
plt.plot(t,x[:,0],'b', label='S')
plt.plot(t[nini:],x[nini:,1],'g', label='I')
plt.plot(t,x[:,2],'r', label='R')
plt.legend(loc='best')
plt.xlabel('t')
plt.ylabel('population')
plt.title("Time of evolution of S,I and R")
plt.grid()

plt.show()
```