Mathematical Biosciences and Engineering

http://www.aimspress.com/journal/MBE

MBE, 18(6): 7979–7998
DOI: 10.3934/mbe.2021396
Received: 06 July 2021
Accepted: 24 August 2021
Published: 14 September 2021

Research article

Analysis of a COVID-19 compartmental model: a mathematical and computational approach

Zita Abreu[1], **Guillaume Cantin**[2] **and Cristiana J. Silva**[1,*]

[1] Center for Research and Development in Mathematics and Applications (CIDMA), Department of Mathematics, University of Aveiro, 3810–193 Aveiro, Portugal

[2] Laboratoire de Mathématiques Appliquées, FR-CNRS-3335, 25, Rue Philippe Lebon, Le Havre Normandie 76063, France

* **Correspondence:** Email: cjoaosilva@ua.pt.

Abstract: In this note, we consider a compartmental epidemic mathematical model given by a system of differential equations. We provide a complete toolkit for performing both a symbolic and numerical analysis of the spreading of COVID-19. By using the free and open-source programming language Python and the mathematical software SageMath, we contribute for the reproducibility of the mathematical analysis of the stability of the equilibrium points of epidemic models and their fitting to real data. The mathematical tools and codes can be adapted to a wide range of mathematical epidemic models.

Keywords: SAIRP epidemic model; COVID-19; stability analysis; free and open-source software; reproducibility of scientific method

1. Introduction

Mathematical models for the transmission dynamics of infectious diseases have been a powerful tool to understand and control epidemics. An important amount of these mathematical models are given by systems of ordinary differential equations (ODE's), considering a continuous-time framework. In the last decades, a great number of compartmental models has been proposed and many of them share some of the assumptions of the well known SIR model, first proposed by Kermack-McKendrick in 1927 [5] and also given by a system of ODE's.

The importance of compartmental models given by systems of ODE's has been even more highlighted since the beginning of the COVID-19 pandemic. In fact, SIR, SEIR, SEIRD-type models, among many others, have been used to analyze, predict and control the spread of SARS-CoV-2 virus worldwide, in what follows we refer to some of this models applied to COVID-19 given by systems

of ODE's. The effect of the lockdown on the spread of COVID-19 was analyzed in [4] by considering a mathematical model that assesses the imposition of the lockdown in Nigeria. Besides lockdown measures, in [6] quarantine, and hospitalization of COVID-19 infected individuals is analyzed. A SIDARTHE compartmental model, is proposed in [10] which discriminates between infected individuals depending on whether they have been diagnosed and on the severity of their symptoms and is fitted to the COVID-19 epidemic in Italy. The authors claim that restrictive social-distancing measures will need to be combined with widespread testing and contact tracing to end the ongoing COVID-19 pandemic. The spread of COVID-19 in Portugal is modeled in [14], fitting the model to real data of simultaneously the number of active infected and hospitalized individuals, SEIR types models applied to COVID-19 are proposed in, for example, [12, 16, 21], were the models are fitted to real data. Quarantine and lockdown measures are considered in [24] with SEIQR type models. Fractional SEIR type models are also important in modeling and predicting the spread of infectious diseases. Their advantages in comparison to other type of models are highlighted in, for example, [1] were COVID-19 epidemic in Pakistan is analyzed through a fractional SEIR type model using the operator of Atangana-Baleanu. The Atangana-Baleanu derivative is also considered in [15]. A nabla discrete ABC-fractional order COVID-19 model is analyzed in [13]. In [2] a fractional model is proposed to study the first COVID-19 outbreak in Wuhan, China. The outbreak in Wuhan is also considered in [7, 19]. A Caputo fractional model is proposed in [18] and fitted to COVID-19 data of Galicia, Spain and Portugal. A Bats-Hosts-Reservoir-People transmission fractional-order COVID-19 model for simulating the potential transmission with the thought of individual response and control measures by the government is analyzed in [23]. Stochastic epidemic models for COVID-19 spread and control are proposed in, e.g., [8, 11]. A new class of distributions are applied to the generalized log-exponential transformation of Gumbel Type-II and implemented using real data of COVID-19, in [27]. The exponential transformation of Gumbel Type-II distribution for modeling COVID-19 data is used to analyze the number of deaths due to COVID-19 for Europe and China, in [28].

Among other important issues, in the mathematical analysis of compartmental models given by systems of ODE's, we may emphasize the stability analysis of the equilibrium points, the basic reproduction number and the model fitting to real data. Although the difficulty of this analysis depends on the complexity of the model under study, part of it is common to the majority of the models and may become simplified if we use adequate mathematical software. In this paper, we show how to use free and open-source software for the mathematical analysis of the models. We focus on the mathematical software SageMath (version 9.2) [22] and `Python` programming language [20], version 3.8, and the Python Libraries: Numpy (version 1.18.5), Pandas (version 0.25.3), Scipy (version 1.4.1) and Matplotlib (version 2.0.0).

In this note, we consider a SAIRP model, given by a system of five ODE's, for the transmission of SARS-CoV-2, first proposed in [26] and after generalized to piecewise constant parameters and complex networks model in [25]. We provide all the codes that allow us to compute the equilibrium points, basic reproduction number, visualize the global stability of the equilibrium points and by considering piecewise constant parameters. Moreover, we provide the Python code that allows to estimate some of the piecewise constant parameters and fit the model to the real data of COVID-19 transmission in Portugal, from March 2, 2020 until April 15, 2021. It is important to note, that all the codes are elementary and thus can be easily adapted to other compartmental models, see e.g., [3, 6, 9, 17, 24, 30].

The paper is organized as follows. In Section 2, the SAIRP model, given by a system of ODE's,

is described. The equilibrium points of the SAIRP model are computed in Section 3 and their global stability is illustrated using SageMath (version 9.2) [22] and Python (version 3.8) [20]. The generalized SAIRP model with piecewise constant parameters is presented and the Python code for the model fitting to COVID-19 Portuguese real data is given, in Section 4.

2. Mathematical model

In this section, we consider the compartmental SAIRP mathematical model proposed in [25, 26] for the transmission dynamics of COVID-19. We start by recalling the assumptions of the model. The total population $N(t)$, with $t \in [0, T]$ (in days) and $T > 0$, is subdivided into five classes: susceptible individuals (S); asymptomatic infected individuals (A); active infected individuals (I); removed (including recovered individuals and COVID-19 induced deaths) (R); and protected individuals (P). Therefore, $N(t) = S(t) + A(t) + I(t) + R(t) + P(t)$, considering a continuous time framework, with $t \in [0, T]$. The total population is homogeneous and has a variable size, with constant recruitment rate, Λ, and natural death rate, $\mu > 0$. The susceptible individuals S become infected by contact with active infected I and asymptomatic infected A individuals, at a rate of infection $\beta\frac{(\theta A+I)}{N}$, where θ represents a modification parameter for the infectiousness of the asymptomatic infected individuals A and β represents the transmission rate. Only a fraction q of asymptomatic infected individuals A develop symptoms and are detected, at a rate ν. Active infected individuals I are transferred to the recovered/removed individuals R, at a rate δ, by recovery from the disease or by COVID-19 induced death. A fraction p, with $0 < p < 1$, is protected (without permanent immunity) from infection, and is transferred to the class of protected individuals P, at a rate ϕ. A fraction m of protected individuals P returns to the susceptible class S, at a rate w. Let $\nu = \nu q$ and $\omega = wm$. The previous assumptions are described by the following system of ordinary differential equations:

$$\begin{cases} \dot{S}(t) = \Lambda - \beta(1-p)\frac{\theta A(t)+I(t)}{N(t)}S(t) - (\phi p + \mu)S(t) + \omega P(t), \\ \dot{A}(t) = \beta(1-p)\frac{\theta A(t)+I(t)}{N(t)}S(t) - (\nu + \mu)A(t), \\ \dot{I}(t) = \nu A(t) - (\delta + \mu)I(t), \\ \dot{R}(t) = \delta I(t) - \mu R(t), \\ \dot{P}(t) = \phi p S(t) - (\omega + \mu)P(t). \end{cases} \tag{2.1}$$

The compartments and parameters descriptions and notations of system (2.1) are resumed in Table 1.

Table 1. Description and notation of the compartments and parameters of model (2.1).

Compartment	Description
S	Susceptible individuals
A	Asymptomatic individuals
I	Infected individuals
R	Recovered individuals
P	Protected individuals

Parameter	Description
Λ	Recruitment rate
μ	Natural death rate
β	Transmission rate
θ	Modification parameter
v	Transfer rate from A to I
q	Transfer fraction from A to I
ϕ	Transfer rate from S to P
p	Transfer fraction from S to P
w	Transfer rate from P to S
m	Transfer fraction from P to S
δ	Recovery rate

Consider the compact invariant region

$$\Omega = \left\{ x = (S,\, A,\, I,\, R,\, P)^T \in (\mathbb{R}^+)^5 \;;\; 0 < S + A + I + R + P \leq \frac{\Lambda}{\mu} \right\}. \tag{2.2}$$

The model (2.1) is biologically and mathematically well-posed, that is for any initial condition $x_0 = (S_0,\, A_0,\, I_0,\, R_0,\, P_0)^T \in \Omega$, the system (2.1) admits a unique solution defined on $[0,\, \infty)$, whose components are non-negative. Furthermore, the region Ω defined by (2.2) is positively invariant [25].

3. Equilibrium points and stability analysis

In this section, we use the free and *open-source* mathematical software SageMath (version 9.2) [22] to help us compute the equilibrium points and the basic reproduction number of model (2.1). The global stability analysis of the equilibrium points is illustrated through numerical simulations developed using Python (version 3.8) [20].

3.1. Computing equilibrium points and basic reproduction number in SageMath

The model (2.1) has two equilibrium points:

- disease-free equilibrium, denoted by Σ_0, given by

$$\Sigma_0 = (S_0, A_0, I_0, R_0, P_0) = \left(\frac{\Lambda\ (\omega + \mu)}{\mu\ (p\phi + \mu + \omega)}, 0, 0, 0, \frac{\phi\, p\Lambda}{\mu\ (p\phi + \mu + \omega)} \right); \tag{3.1}$$

- endemic equilibrium, Σ_+, whenever $R_0 > 1$, given by

$$\Sigma_+ = (S_+, A_+, I_+, R_+, P_+)$$

with

$$\begin{aligned} S_+ &= \frac{\Lambda(\omega+\mu)}{(p\phi+\mu+\omega)\mu}R_0^{-1}\,, \\ A_+ &= \frac{\Lambda}{\nu+\mu}R_0^{-1}(R_0-1)\,, \\ I_+ &= \frac{\Lambda\nu}{(\nu+\mu)(\delta+\mu)}R_0^{-1}(R_0-1)\,, \\ R_+ &= \frac{\delta\Lambda\nu}{(\nu+\mu)(\delta+\mu)\mu}R_0^{-1}(R_0-1)\,, \\ P_+ &= \frac{\Lambda\phi p}{(p\phi+\mu+\omega)\mu}R_0^{-1}\,, \end{aligned} \tag{3.2}$$

where the basic reproduction number, R_0, is given by

$$R_0 = \frac{\beta\,(1-p)\,(\delta\,\theta+\mu\,\theta+\nu)\,(\omega+\mu)}{(\delta+\mu)\,(\nu+\mu)\,(p\phi+\mu+\omega)}\,. \tag{3.3}$$

The SageMath code reads as follows:

```
var('S','A','I','R','P','phi','mu','nu','delta',\
'omega','Lambda','theta','beta','p','lambdat');
a0 = p*phi+mu; a1 = nu+mu; a2 = delta+mu; a3 = omega+mu;
lambdat = beta*(A*theta+I)*(1-p);
N = S+A+I+R+P;
eqS = Lambda + omega*P-lambdat*S/N-a0*S;
eqA = lambdat*S/N-a1*A;
eqI = A*nu-I*a2;
eqR = I*delta-R*mu;
eqP = S*p*phi-P*a3;
pretty_print((eqS+eqA+eqI+eqP+eqR).full_simplify())
sistema = [eqR, eqA == 0, eqI == 0, eqP == 0, eqS == 0];
sol = solve(sistema, A, I, P, R, S); pretty_print(sol)
```

The output `sol` gives the two biological meaningful equilibrium points Σ_0 and Σ_+ given by models (3.1) and (3.2), respectively.

```
DFE = sol[2]; pretty_print(DFE)
EE = sol[0]; pretty_print(EE)
```

To compute the basic reproduction number R_0, (3.3), we follow the approach presented in [29].

```
F1 = 0; F2 = lambdat*S/N; F3 = 0; F4 = 0; F5 = 0;
V1 = -eqS; V2 = F2-eqA; V3 = -eqI; V4 = -eqR; V5 = -eqP;
```

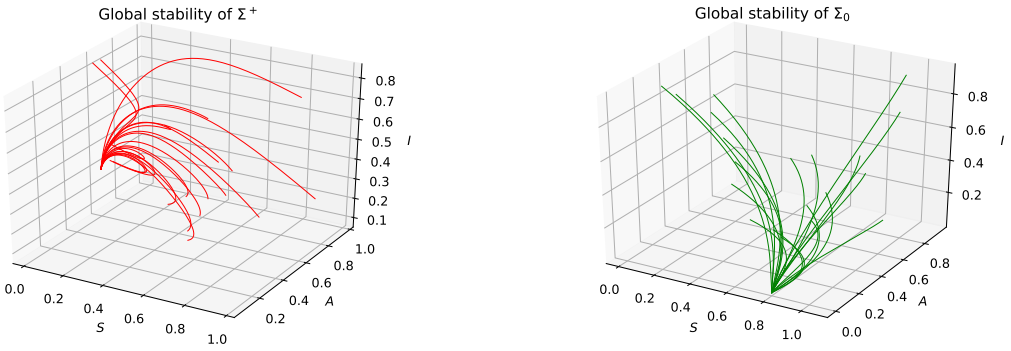

Figure 1. Left: global stability of the endemic equilibirum Σ^+ ($R_0 \simeq 6.87$). Right: global stability of the disease-free equilibrium Σ_0 ($R_0 \simeq 0.0687$).

```
JF = jacobian((F1, F2, F3, F4, F5),(S, A, I, R, P));
JV = jacobian([V1, V2, V3, V4, V5],(S, A, I, R, P));
IV = (JV).inverse();
M = JF * IV;
M1 = M.subs(DFE);
vp = M1.eigenvalues();
R0 = (vp[0].factor());
```

3.2. Stability analysis

The following theorems hold and are numerically illustrated in Figure 1.

Theorem 1 (Local stability of the DFE, [25])**.** *The disease-free equilibrium, Σ_0, is locally asymptotically stable whenever $R_0 < 1$.*

Theorem 2 (Global stability of the DFE, [25])**.** *If $R_0 < 1$, then the disease-free equilibrium, Σ_0, is globally asymptotically stable in Ω.*

Theorem 3 (Global stability of the EE, [25])**.** *The compact region Γ defined by*

$$\Gamma = \left\{ x = (S,\, A,\, I,\, R,\, P)^T \in (\mathbb{R}^+)^5 \,;\, S + A + I + R + P = \frac{\Lambda}{\mu} \right\}$$

is positively invariant under the flow induced by system (2.1)*. It contains the disease-free equilibrium, Σ_0, and the endemic equilibrium, Σ_+, if $R_0 > 1$. Furthermore, if $R_0 > 1$, then the endemic equilibrium Σ_+ is globally asymptotically stable in Γ.*

The Python code to generate Figure 1 is given below. We first import several Python scientific libraries: `numpy` contains standard routines for numerical computations; `scipy` contains the function `odeint`, which implements the Runge-Kutta method for integrating ODE's systems; `matplotlib` contains useful functions for producing figures.

```
#!/usr/bin/env python3

import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from scipy.integrate import odeint
from random import random
```

Next, we define the parameters for the SAIRP model (2.1).

```
# SAIRP parameters
mu = 1
Lambda = 1*mu
v = 1
q = 1
nu = 1
delta = 1
theta = 1
phi = 5
delta = 0.1
w = 1
p = 0.1
omega = 1
beta = 10 # beta = 10 for EE or beta = 0.1 for DFE
```

Using expression (3.3), we easily compute the basic reproduction number R_0.

```
N = beta*(1-p)*(delta*theta+mu*theta+nu)*(omega+mu)
D = (delta + mu)*(nu+mu)*(p*phi+mu+omega)
r0 = N/D
print('R0 =', r0)
```

Afterwards, we define the SAIRP model given by system (2.1).

```
def SAIRP(X, t):
    S, A, I, R, P = X
    N = S+A+I+R+P
    C = beta*(1-p)*(theta*A+I)/N
    dS = Lambda - C*S - phi*p*S + omega*P - mu*S
    dA = C*S - nu*A - mu*A
    dI = nu*A - delta*I - mu*I
    dR = delta*I - mu*R
    dP = phi*p*S - omega*P - mu*P
    return [dS, dA, dI, dR, dP]
```

Finally, we integrate the SAIRP model (2.1) with several randomly chosen initial conditions and we produce the 3D Figure 1.

```
time = np.arange(0, 10, 0.01)
fig = plt.figure()
ax = fig.gca(projection='3d')
ax.set_title(r"Global stability of $\Sigma^+$")
ax.set_xlabel(r"$S$")
ax.set_ylabel(r"$A$")
ax.set_zlabel(r"$I$")
for j in range(20):
    S0 = random()
    A0 = random()
    I0 = random()
    R0 = 1
    P0 = 1

    X0 = [S0, A0, I0, R0, P0]

    orbit = odeint(SAIRP, X0, time)
    S, A, I, R, P = orbit.T
    ax.plot(S, A, I, 'r', lw=0.5)
plt.savefig('EE-GAS.pdf')
plt.show()
```

4. Mathematical model with piecewise constant parameters

In this section, we consider the SAIRP model with piecewise constant parameters, proposed in [25], which allows to model the impact of public health policies and the human behavior in the dynamics of the COVID-19 epidemic.

For the sake of simplicity, the equations of the SAIRP model (2.1) can be rewritten as

$$\dot{x}(t) = f\left(x(t),\, \alpha\right), \quad t > 0, \tag{4.1}$$

with $x = (S,\, A,\, I,\, R,\, P)^T \in \mathbb{R}^5$ and $\alpha = (\Lambda,\, \mu,\, \beta,\, p,\, \theta,\, \phi,\, \omega,\, \nu,\, \delta)^T \in \mathbb{R}^9$, where the non-linear operator f is defined in $\mathbb{R}^5 \times \mathbb{R}^9$ by

$$f(x,\, \alpha) = \begin{pmatrix} \Lambda - \beta(1-p)\frac{\theta A + I}{N} - \phi p S + \omega P - \mu S \\ \beta(1-p)\frac{\theta A + I}{N} S - \nu A - \mu A \\ \nu A - \delta I - \mu I \\ \delta I - \mu R \\ \phi p S - \omega P - \mu P \end{pmatrix}. \tag{4.2}$$

The time line $[0,\ T_{\text{end}}]$ is subdivided into a finite number of $n+1$ intervals

$$[T_0,\ T_1) \cup [T_1,\ T_2) \cup \cdots \cup [T_n,\ T_{\text{end}}],$$

with disjoint unions, and we introduce a piecewise constant function α defined on each time interval as

$$\alpha(t) = \alpha_i, \quad t \in [T_i,\ T_{i+1}), \quad 0 \le i \le n,$$

with $T_0 = 0$, $T_{n+1} = T_{\text{end}}$ and $\alpha_i \in \mathbb{R}^9$. Next, consider the sequence of Cauchy problems defined for each initial condition $x_0 \in \Omega$ by

$$\begin{cases} x(0) = x_0, & \dot{x}(t) = f(x(t),\ \alpha_0), \quad T_0 < t < T_1, \\ x(T_i) = \lim\limits_{\substack{t \to T_i \\ t \in (T_{i-1},\ T_i)}} x(t), & \dot{x}(t) = f(x(t),\ \alpha_i), \quad T_i < t < T_{i+1}, \quad 1 \le i \le n\,. \end{cases} \tag{4.3}$$

The following theorem establishes the well-posedness of system (4.3).

Theorem 4. *For any initial condition $x_0 \in \Omega$, the sequence of Cauchy problems given by model* (4.3) *admits a unique global solution, denoted again by $x(t,\ x_0)$, whose components are non-negative. Furthermore, the region Ω is positively invariant* [25].

We recall that the solutions of problem (4.3) are continuous on the time interval $[T_0,\ T_{\text{end}}]$, but may not be of class $\mathscr{C}^1$ at $t = T_i$, $0 \le i \le n-1$. From the modeling point of view, each change of parameters occurring at time $t = T_i$ $(1 \le i \le n-1)$ corresponds, for example, to a public announcement of confinement/lift of confinement or prohibition of displacement [25].

In what follows, we sub-divide the time interval $[0, 410]$ days into 9 sub-intervals and consider a set of piecewise parameters which are estimated in order to fit the real data of COVID-19 spread in Portugal, since the first confirmed case on March 2, 2020, until April 15, 2021.

Model fitting to real data with Python In order to fit the real data of active infected individuals by SARS-CoV-2 (detected by test) daily provided by the health authorities in Portugal [31], we use the model (4.3) with piecewise constant parameters and use the programming language Python (version 3.8) [20], and the Python Libraries: Numpy (version 1.18.5), Pandas (version 0.25.3), Scipy (version 1.4.1) and Matplotlib (version 2.0.0). The real data are available in [31] or, for example, in the data repository site, see e.g. [32]. In what follows we explain the goal of each code block.

First we need as previously to import several Python scientific libraries.

```
#!/usr/bin/env python3

import numpy as np
from scipy.integrate import odeint
import pandas as pd
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
```

We consider the real data from [31, 32], here denominated *data-pt-covid19* with extension *.xlsx*. To read the real data one can use the following code.

```
excel_file = 'data-pt-covid19.xlsx'
M = pd.read_excel(excel_file)
M_New = M[['INF.ATIVOS']]
```

Next, we subdivide the time window of the total 410 days into 9 subintervals. This specific subdivision is related to the increase/decrease of the transmission of the virus in the community, the public health measures implemented by the Portuguese authorities and the human behavior, that changes over time.

```
M_New1 = np.squeeze(np.asarray(M_New[0:73]))
M_New2 = np.squeeze(np.asarray(M_New[73:90]))
M_New3 = np.squeeze(np.asarray(M_New[90:130]))
M_New4 = np.squeeze(np.asarray(M_New[130:163]))
M_New5 = np.squeeze(np.asarray(M_New[163:200]))
M_New6 = np.squeeze(np.asarray(M_New[200:253]))
M_New7 = np.squeeze(np.asarray(M_New[253:304]))
M_New8 = np.squeeze(np.asarray(M_New[304:329]))
M_New9 = np.squeeze(np.asarray(M_New[329:410]))

maxi = len(M)

tf = np.linspace(0,maxi)

tf1 =np.linspace(0,73,73)
tf2 =np.linspace(73,90,17)
tf3 =np.linspace(90,130,40)
tf4 =np.linspace(130,163,33)
tf5 =np.linspace(163,200,37)
tf6 =np.linspace(200,253,53)
tf7 =np.linspace(253,304,51)
tf8 =np.linspace(304,329,25)
tf9 =np.linspace(329,410,81)
```

Next, we draw the curve of active infected individuals with COVID-19 in Portugal (real data), from the first confirmed case day, March 2, 2020, until April 15, 2021.

```
fig1 = plt.figure(facecolor='w',
                  num=None,
                  figsize=(8, 6),
                  dpi=100,
                  edgecolor='k')
fig1.suptitle('Active infected - Portugal',
```

```
                fontsize=14,
                fontweight='bold')
plt = fig1.add_subplot(111,
                       facecolor='white',
                       axisbelow=True)
```

In order to solve the model (4.1) we need to define the initial conditions, for $t = 0$, that correspond to the number of individuals in each class on March 2, 2020.

```
# Total population, N
N = 10295894 + 2 + 2/0.15

# Initial number of infected and recovered individuals, I0 and R0.
S = 10295894
I = 2
A = (2/0.15)
R = 0
P = 0
```

Then, we define the values of the parameters that take constant values for all time $t \in [0, 410]$.

```
theta = 1
phi = 1/12
w = 1/45
nu = 0.15
q = 0.15
delta =1/(27)
Lambda = ((0.19/100)*N)/365
mu = 1/(81*365)
```

The parameters β, p and m are assumed to be piecewise constant. Moreover, we estimate the parameter β for all $t \in [0, 410]$ days and the parameters m and p in some of the sub-intervals, using `scipy.optimize.curve_fit` from Python, see [33] for more details, that uses non-linear least squares method to fit a function to data.

```
p1 = 0.675
p2 = 0.650
p3 = 0.580
p4 = 0.610
p5 = 0.580
p8 = 0.370
p9 = 0.550

m1 = 0.09
m2 = 0.09
```

```
m3 = 0.18
m4 = 0.16
m5 = 0.17

# The SAIRP model
def sairp_model(y,tf,N,beta,theta,p,phi,w,nu,q,delta,m,Lambda,mu):
        S,A,I,R,P = y
        C = beta * (theta*A+I)/(S+A+I+R+P)
        dS = Lambda-C*(1-p)*S-phi*p*S+w*m*P-mu*S
        dA = C*(1-p)*S-nu*A-mu*A
        dI = nu*A-delta*I-mu*I
        dR = delta*I-mu*R
        dP = phi*p*S-w*m*P-mu*P
        return dS,dA,dI,dR,dP

# Initial conditions vector
y0 = S, A, I, R, P

#1 sub-interval of time
def fit_odeint1(tf1,beta):
    return odeint(sairp_model, y0, tf1, args=ARGS1)[:,2]

popt1, pcov = curve_fit(fit_odeint1,tf1,M_New1,bounds=[(0.25),(2)])
fitted1 = fit_odeint1(tf1, *popt1)
beta1 = popt1[0]
print("\n beta1: ",beta1)

ret = odeint(sairp_model, y0, tf1, args=(N,\
        beta1,theta,p1,phi,w,nu,q,delta,m1,Lambda,mu))
S, A, I, R, P = ret.T

#2 sub-interval of time
y2 = ret.T[0][72],ret.T[1][72],ret.T[2][72],ret.T[3][72],ret.T[4][72]
def fit_odeint2(tf2,beta):
        return odeint(sairp_model, y2, tf2,\
        args=(N,beta,theta,p2,phi,w,nu,q,delta,m2,Lambda,mu))[:,2]

popt2, pcov = curve_fit(fit_odeint2,tf2,M_New2,bounds=[(0.25),(2)])
fitted2 = fit_odeint2(tf2, *popt2)
beta2 = popt2[0]
print("\n beta2: ",beta2)

ret = odeint(sairp_model, y2, tf2,\
```

```
        args=(N, beta2,theta,p2,phi,w,nu,q,delta,m2,Lambda,mu))
S2, A2, I2, R2, P2 = ret.T

#3 sub-interval of time
y3 = ret.T[0][16],ret.T[1][16],ret.T[2][16],ret.T[3][16],ret.T[4][16]
def fit_odeint3(tf3,beta):
        return odeint(sairp_model, y3, tf3,\
        args=(N, beta,theta,p3,phi,w,nu,q,delta,m3,Lambda,mu))[:,2]

popt3, pcov = curve_fit(fit_odeint3,tf3,M_New3,bounds=[(0.25),(2)])
fitted3 = fit_odeint3(tf3, *popt3)
beta3 = popt3[0]
print("\n beta3: ",beta3)

ret = odeint(sairp_model, y3, tf3,\
        args=(N, beta3,theta,p3,phi,w,nu,q,delta,m3,Lambda,mu))
S3, A3, I3, R3, P3 = ret.T

#4 sub-interval of time
y4 = ret.T[0][39],ret.T[1][39],ret.T[2][39],ret.T[3][39],ret.T[4][39]
def fit_odeint4(tf4,beta):
        return odeint(sairp_model, y4, tf4,\
        args=(N, beta,theta,p4,phi,w,nu,q,delta,m4,Lambda,mu))[:,2]

popt4, pcov = curve_fit(fit_odeint4,tf4,M_New4,bounds=[(0.25),(2)])
fitted4 = fit_odeint4(tf4, *popt4)
beta4 = popt4[0]
print("\n beta4: ",beta4)

ret = odeint(sairp_model, y4, tf4,\
        args=(N, beta4,theta,p4,phi,w,nu,q,delta,m4,Lambda,mu))
S4, A4, I4, R4, P4 = ret.T

#5 sub-interval of time
y5 = ret.T[0][32],ret.T[1][32],ret.T[2][32],ret.T[3][32],ret.T[4][32]
def fit_odeint5(tf5,beta):
        return odeint(sairp_model, y5, tf5,\
        args=(N, beta,theta,p5,phi,w,nu,q,delta,m5,Lambda,mu))[:,2]

popt5, pcov = curve_fit(fit_odeint5,tf5,M_New5,bounds=[(0.25),(2)])
fitted5 = fit_odeint5(tf5, *popt5)
beta5 = popt5[0]
print("\n beta5: ",beta5)
```

```
ret = odeint(sairp_model, y5, tf5,\
        args=(N, beta5,theta,p5,phi,w,nu,q,delta,m5,Lambda,mu))
S5, A5, I5, R5, P5 = ret.T

#6 sub-interval of time
y6 = ret.T[0][36],ret.T[1][36],ret.T[2][36],ret.T[3][36],ret.T[4][36]
def fit_odeint6(tf6,beta,m,p):
        return odeint(sairp_model, y6, tf6,\
        args=(N, beta,theta,p,phi,w,nu,q,delta,m,Lambda,mu))[:,2]

popt6, pcov = curve_fit(fit_odeint6,tf6,M_New6,\
        bounds=[(0.25,0.09,0.29),(2,1,0.58)])
fitted6 = fit_odeint6(tf6, *popt6)
beta6 = popt6[0]
m6 = popt6[1]
p6 = popt6[2]
print("\n beta6: ",beta6)
print("\n m6: ",m6)
print("\n p6: ",p6)

ret = odeint(sairp_model, y6, tf6,\
        args=(N, beta6,theta,p6,phi,w,nu,q,delta,m6,Lambda,mu))
S6, A6, I6, R6, P6 = ret.T

#7 sub-interval of time
y7 = ret.T[0][52],ret.T[1][52],ret.T[2][52],ret.T[3][52],ret.T[4][52]
def fit_odeint7(tf7,beta,m,p):
        return odeint(sairp_model, y7, tf7,\
        args=(N, beta,theta,p,phi,w,nu,q,delta,m,Lambda,mu))[:,2]

popt7, pcov = curve_fit(fit_odeint7,tf7,M_New7,\
        bounds=[(0.25,0.09,0.29),(2,1,0.58)])
fitted7 = fit_odeint7(tf7, *popt7)
beta7 = popt7[0]
m7 = popt7[1]
p7 = popt7[2]
print("\n beta7: ",beta7)
print("\n m7: ",m7)
print("\n p7: ",p7)

ret = odeint(sairp_model, y7, tf7,\
        args=(N, beta7,theta,p7,phi,w,nu,q,delta,m7,Lambda,mu))
```

```
S7, A7, I7, R7, P7 = ret.T

#8 sub-interval of time
y8 = ret.T[0][50],ret.T[1][50],ret.T[2][50],ret.T[3][50],ret.T[4][50]
def fit_odeint8(tf8,beta,m):
        return odeint(sairp_model, y8, tf8,\
        args=(N, beta,theta,p8,phi,w,nu,q,delta,m,Lambda,mu))[:,2]

popt8, pcov = curve_fit(fit_odeint8,tf8,M_New8,\
        bounds=[(0,0.09),(100,1)])
fitted8 = fit_odeint8(tf8, *popt8)
beta8 = popt8[0]
m8 = popt8[1]
print("\n beta8: ",beta8)
print("\n m8: ",m8)

ret = odeint(sairp_model, y8, tf8,\
        args=(N, beta8,theta,p8,phi,w,nu,q,delta,m8,Lambda,mu))
S8, A8, I8, R8, P8 = ret.T

#9 sub-interval of time
y9 = ret.T[0][24],ret.T[1][24],ret.T[2][24],ret.T[3][24],ret.T[4][24]
def fit_odeint9(tf9,beta,m):
        return odeint(sairp_model, y9, tf9,\
        args=(N, beta,theta,p9,phi,w,nu,q,delta,m,Lambda,mu))[:,2]

popt9, pcov = curve_fit(fit_odeint9,tf9,M_New9,\
        bounds=[(0,0.09),(0.78,1)])
fitted9 = fit_odeint9(tf9, *popt9)
beta9 = popt9[0]
m9 = popt9[1]
print("\n beta9: ",beta9)
print("\n m9: ",m9)

ret = odeint(sairp_model, y9, tf9,\
        args=(N, beta9,theta,p9,phi,w,nu,q,delta,m9,Lambda,mu))
S9, A9, I9, R9, P9 = ret.T
```

Finally, we make the plot with the real data and the model solution $I(t)$, for $t \in [0, 410]$.

```
# Plot the data on 9 separate curves for I(t)
plt.plot(tf1, fitted1, color="purple", alpha=5, lw=2.5,\
        label='Model -> 2 March - 13 May')
plt.plot(tf2, fitted2 , color="magenta", alpha=5, lw=2.5,\
```

```
        label='Model -> 13 May - 30 May')
plt.plot(tf3, fitted3 , color="blue", alpha=5, lw=2.5,\
        label='Model -> 30 May - 9 July')
plt.plot(tf4, fitted4 , color="aqua", alpha=5, lw=2.5,\
        label='Model -> 9 July - 11 August')
plt.plot(tf5, fitted5 , color="lawngreen", alpha=5, lw=2.5,\
        label='Model -> 11 August - 17 September')
plt.plot(tf6, fitted6 , color="darkorange", alpha=5, lw=2.5,\
        label='Model -> 17 September - 9 November')
plt.plot(tf7, fitted7 , color="chocolate", alpha=5, lw=2.5,\
        label='Model -> 9 November - 30 December')
plt.plot(tf8, fitted8 , color="red", alpha=5, lw=2.5,\
        label='Model -> 30 December - 24 January')
plt.plot(tf9, fitted9 , color="maroon", alpha=5, lw=2.5,\
        label='Model -> 24 January - 15 April')

plt.plot(M_New,color="black",alpha=5,\
        lw=2,linestyle='dotted',label='Real data')

xcoords = [73, 90, 130, 163, 200, 253, 304, 329]

for xc in zip(xcoords):
plt.axvline(x=xc, color="black", linestyle='dotted', alpha=0.5, lw=2)

plt.set_xlabel('Time (days)',fontweight="bold")
plt.set_ylabel('Active infected',fontweight="bold")
plt.yaxis.set_tick_params(length=0)
plt.xaxis.set_tick_params(length=0)
plt.grid(b=True, which='major', c='w', lw=2, ls='-')
legend = plt.legend(title="Population: ",loc=6,bbox_to_anchor=(1.05,0.2))
```

The graphic that results from the previous Python code is given in Figure 2.

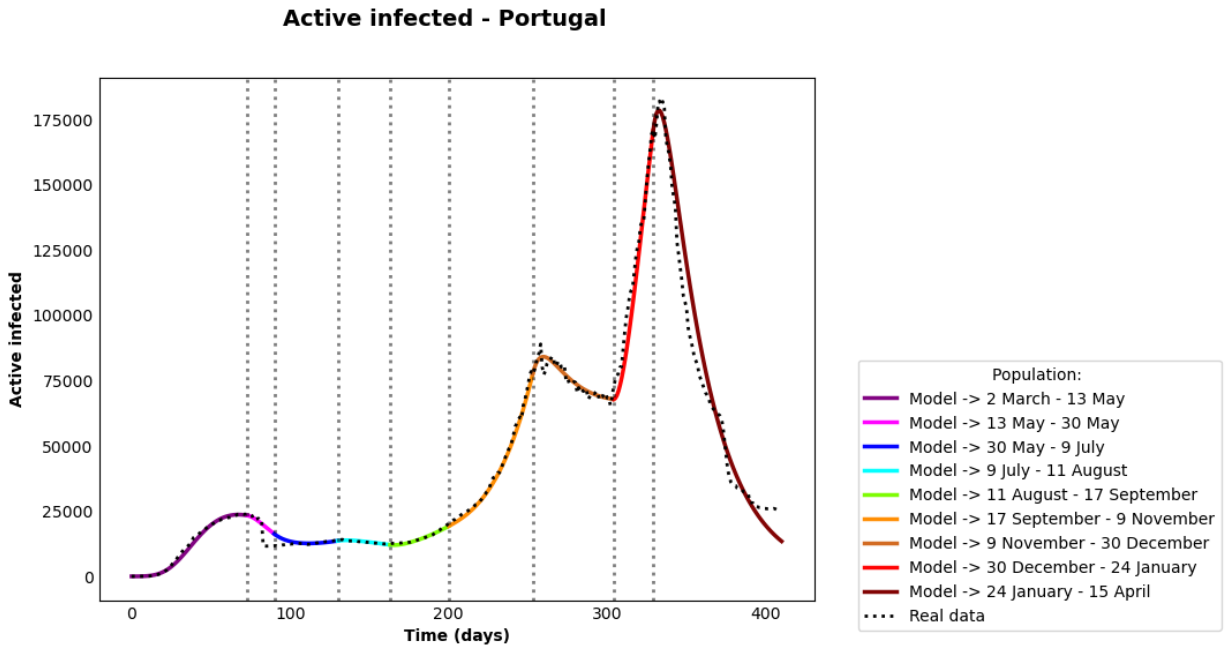

Figure 2. Output of the Python code: fitting the SAIRP model (4.3) to COVID-19 active infected individuals in Portugal, between March 02, 2020 and April 15, 2021.

We recall, that the mathematical method and numerical code can be applied to other models given by systems of ODE's that describe the transmission dynamics of different virus or bacteria in human, animals or cells, for example.

5. Conclusions

In this note, we provided a complete toolkit for performing both a symbolic and numerical analysis of a recent epidemic model, presented in [25], introduced in order to study the spreading of the COVID-19 pandemic. This innovative compartmental model takes into account the possible transmission of the virus by asymptomatic individuals, as well as the possibility to protect a fraction of the affected population by public health strategies, such as confinement or quarantine. Since the public health policies have a strong influence on the spreading of the epidemic, we have also considered a piecewise constant parameters extension of the initial autonomous system, which have proved its ability to fit with real data. Once the mathematical analysis of such a complex compartmental model can be tedious, we have presented a computational approach, which is intended to support the theoretical analysis:

- using the free and open-source software Sagemath, we have first computed the symbolic expressions of the basic reproduction number R_0 (Eq (3.3)), of the disease-free equilibrium Σ_0 (Eq (3.1)) and of the endemic equilibrium Σ_+ (Eq (3.2)), which highlight the role of each parameter of the model;
- using the scientific libraries numpy, `scipy` and `matplotlib` of the free and open-source language Python, we have integrated the epidemic model, in order to illustrate the theoretical stability statements (see Figure 1);

- finally, we fitted the piecewise constant parameters extension of the initial model with recent real-world data (see Figure 2).

Overall, the programs presented in this note have been written in a sufficiently general manner, so that they can easily be adapted to a great number of other epidemic models. In a near future, we aim to apply our computational approach to an improved version of our complex compartmental model, so as to consider the effects of the mutations of the virus and the benefits of vaccination.

Acknowledgments

This research is partially supported by the Portuguese Foundation for Science and Technology (FCT) by the project UIDB/04106/2020 (CIDMA). Cristiana J. Silva is also supported by FCT via the FCT Researcher Program CEEC Individual 2018 with reference CEECIND/00564/2018.

Conflict of interest

The authors declare that they have no conflict of interest.

References

1. Z. Ahmad, M. Arif, F. Ali, I. Khan, K. S. Nisar, A report on COVID-19 epidemic in Pakistan using SEIR fractional model, *Sci. Rep.*, **10** (2020), 1–14.
2. S. Ahmad, A. Ullah, Q. M. Al-Mdallal, H. Khan, K. Shah, A. Khan, Fractional order mathematical modeling of COVID-19 transmission, *Chaos Solitons Fractals*, **139** (2020), 110256.
3. M. Amouch, N. Karim, Modeling the dynamic of COVID-19 with different types of transmissions, *Chaos Solitons Fractals*, **150** (2021), 111188.
4. I. A. Baba, A. Yusuf, K. S. Nisar, A. Abdel-Aty, T. A. Nofal, Mathematical model to assess the imposition of lockdown during COVID-19 pandemic, *Results Phy.*, **20** (2021), 103716.
5. N. Bacar, McKendrick and Kermack on epidemic modelling (1926–1927), in *A Short History of Mathematical Population Dynamics*, Springer, (2011), 89–96.
6. S. Bugalia, V. P. Bajiya, J. P. Tripathi, M. T. Li, G. Q. Sun, Mathematical modeling of COVID-19 transmission: the roles of intervention strategies and lockdown, *Math. Biosci. Eng.*, **17** (2020), 5961–5986.
7. S. A. Cheema, T. Kifayat, A. R. Rahman, U. Khan, A. Zaib, et al., Is social distancing, and quarantine effective in restricting covid-19 outbreak? Statistical evidences from Wuhan, China, *Comput. Mater. Con.*, **66** (2021), 1977–1985.
8. J. Danane, K. Allali, Z. Hammouch, K. S. Nisar, Mathematical analysis and simulation of a stochastic COVID-19 L vy jump model with isolation strategy, *Results Phys.*, **23** (2021), 103994.
9. Z. B. Dieudonné, Mathematical model for the mitigation of the economic effects of the Covid-19 in the Democratic Republic of the Congo, *Plos One*, **16** (2021), e0250775.
10. G. Giordano, F. Blanchini, R. Bruno, P. Colaneri, A. Di Filippo, A. Di Matteo, et al., Modelling the COVID-19 epidemic and implementation of population-wide interventions in Italy, *Nat. Med.*, **26** (2020), 855–860.

11. G. Hussain, T. Khan, A. Khan, M. Inc, G. Zaman, K. S. Nisar, A. Akgul, Modeling the dynamics of novel coronavirus (COVID-19) via stochastic epidemic model, *Alex. Eng. J.*, **60** (2021), 4121–4130.
12. S. Khajanchi, K. Sarkar, J. Mondal, K. S. Nisar, S. F. Abdelwahab, Mathematical modeling of the COVID-19 pandemic with intervention strategies, *Results Phys.*, **25** (2021), 104285.
13. A. Khan, H. M. Alshehri, T. Abdeljawad, Q. M. Al-Mdallal, H. Khan, Stability analysis of fractional nabla difference COVID-19 model, *Results Phys.*, **22** (2021), 103888.
14. A. P. Lemos-Pai£o, C. J. Silva, D. F. Torres, A new compartmental epidemiological model for COVID-19 with a case study of portugal, *Ecol. Complex.*, **44** (2020), 100885.
15. K. Logeswari, C. Ravichandran, K. S. Nisar, Mathematical model for spreading of COVID-19 virus with the Mittag Leffler kernel, *Numer. Meth. Part. Differ. Equations*, (2020), 1–16.
16. L. López, X. Rodo, A modified SEIR model to predict the COVID-19 outbreak in Spain and Italy: Simulating control scenarios and multi-scale epidemics, *Results Phys.*, **21** (2021), 103746.
17. J. Y. Mugisha, J. Ssebuliba, J. N. Nakakawa, C. R. Kikawa, A. Ssematimba, Mathematical modeling of COVID-19 transmission dynamics in Uganda: Implications of complacency and early easing of lockdown, *Plos One*, **16** (2021), e0247456.
18. F. Ndairou, I. Area, J. J. Nieto, C. J. Silva, D. F. M. Torres, Fractional model of COVID-19 applied to Galicia, Spain and Portugal, *Chaos Solitons Fractals*, **144** (2021), 110652.
19. K. S. Nisar, S. Ahmad, A. Ullah, K. Shah, H. Alrabaiah, M. Arfan, Mathematical analysis of SIRD model of COVID-19 with Caputo fractional derivative based on real data, *Results Phys.*, **21** (2021), 103772.
20. Python Software Foundation, *Python Language Reference*, (2019). Available from: http://www.python.org.
21. A. Radulescu, C. Williams, K. Cavanagh, Management strategies in a SEIR-type model of COVID 19 community spread, *Sci. Rep.*, **10** (2020), 1–16.
22. The Sage Developers, *The Sage Mathematics Software System*, (2020). Available from: https://www.sagemath.org.
23. A. S. Shaikh, I. N. Shaikh, K. S. Nisar, A mathematical model of COVID-19 using fractional derivative: outbreak in India with dynamics of transmission and control, *Adv. Differ. Equ.*, (2020), 1–19.
24. S. Sharma, V. Volpert, M. Banerjee, Extended SEIQR type model for COVID-19 epidemic and data analysis, *Math. Biosic. Eng.*, **17** (2020), 7562–7604.
25. C. J. Silva, G. Cantin, C. Cruz, R. Fonseca-Pinto, R. Fonseca, E. S. Santos, et al., Complex network model for COVID-19: human behavior, pseudo-periodic solutions and multiple epidemic waves, *J. Math. Anal. Appl.*, Forthcoming 2006.
26. C. J. Silva, C. Cruz, D. F. M. Torres, A. P. Muñuzuri, A. Carballosa, I. Area, et al., Optimal control of the COVID-19 pandemic: controlled sanitary deconfinement in Portugal, *Sci. Rep.*, **11** (2021), 1–15.
27. T. N. Sindhu, A. Shafiq, Q. M. Al-Mdallal, On the analysis of number of deaths due to Covid-19 outbreak data using a new class of distributions, *Results Phys.*, **21** (2021), 103747.

28. T. N. Sindhu, A. Shafiq, Q. M. Al-Mdallal, Exponentiated transformation of Gumbel Type-II distribution for modeling COVID-19 data, *Alex. Eng. J.*, **60** (2021), 671–689.

29. P. van den Driessche, J. Watmough, Reproduction numbers and sub-threshold endemic equilibria for compartmental models of disease transmission, *Math. Biosci.*, **180** (2002), 29–48.

30. C. Y. Yang, J. Wang, A mathematical model for the novel coronavirus epidemic in Wuhan, China, *Math. Biosci. Eng.*, **17** (2020), 2708–2724.

31. Direção Geral da Saúde – COVID-19, *Ponto de Situação Atual em Portugal*, (2021). Available from: https://covid19.min-saude.pt/ponto-de-situacao-atual-em-portugal.

32. GitHub, *Dados Relativos a Pandemia COVID-19 em Portugal*, (2021). Available from: https://github.com/dssg-pt/covid19pt-data.

33. SciPy.org, *scipy.optimize.curve fit*, (2021). Available from: https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.curve_fit.html.

AIMS Press