Supplementary material for "An epidemiological compartmental model with automated parameter estimation for the spread of COVID-19 with analysis of data from Germany and Brazil"

Adriano A. Batista [1*] and Severino Horácio da Silva[2†]

[1] *Departamento de Física, Universidade Federal de Campina Grande, 58051-900 Campina Grande PB, Brazil.*

[2] *Departamento de Matemática, Universidade Federal de Campina Grande, 58051-900 Campina Grande PB, Brazil.*

(Dated: February 24, 2022)

Here we describe the numerical integration method used, the data, and the code. The latest version of the code used in the article is publicly available in the gitHub repository: https://github.com/aabatista/covid19_parameter_estimation. We also present one alternative proof (non-negativeness).

* adriano@df.ufcg.edu.br

† horacio@mat.ufcg.edu.br

I. NUMERICAL INTEGRATION

Here is a simplified version of our model:

$$\frac{dS}{dt} = -\kappa SI, \tag{1}$$
$$\frac{dI}{dt} = -\gamma I + \kappa SI, \tag{2}$$
$$\frac{dR}{dt} = \rho I, \tag{3}$$
$$\frac{dM}{dt} = \lambda I, \tag{4}$$

where $\gamma = \frac{1}{\tau} = \lambda + \rho$. We consider that the susceptible population does not vary appreciably in the time span of one day, so that we can assume that it is basically constant in this approximation of the integration between t_n and t_{n+1}. Furthermore, we also assume initially, for the sake of simplicity, that all three independent parameters of the model are constant in time. Hence, when $t_n < t < t_{n+1}$, after integrating Eq. (2) from t_n to t, we find

$$I(t) = I_n e^{(-\gamma+\kappa S_n)(t-t_n)}, \tag{5}$$

in which $S_n = S(t_n)$. When $t = t_{n+1}$, we obtain

$$I_{n+1} = I_n e^{(-\gamma+\kappa S_n)\Delta t},$$

where $\Delta t = t_{n+1} - t_n$. From the integration of Eq. (1), we find

$$\begin{aligned} \ln \frac{S_{n+1}}{S_n} &= -\kappa \int_{t_n}^{t_{n+1}} I(t)dt = -\kappa I_n \int_{t_n}^{t_{n+1}} e^{(-\gamma+\kappa S_n)(t-t_n)}dt = \frac{\kappa I_n}{(\gamma - \kappa S_n)}\left[e^{(-\gamma+\kappa S_n)\Delta t} - 1\right] \\ &= \kappa \frac{I_{n+1} - I_n}{\gamma - \kappa S_n}. \end{aligned} \tag{6}$$

Thus, we obtain

$$S_{n+1} = S_n e^{\frac{\kappa \Delta I}{\gamma - \kappa S_n}}, \tag{7}$$

where $\Delta I = I_{n+1} - I_n$. From Eqs. (3) and (5), we find

$$R_{n+1} = R_n + \rho I_n \frac{e^{(-\gamma+\kappa S_n)\Delta t} - 1}{-\gamma + \kappa S_n} = R_n + \frac{\rho \Delta I}{-\gamma + \kappa S_n} \tag{8}$$

Likewise, from Eqs. (4) and (5), we find

$$M_{n+1} = M_n + \lambda I_n \frac{e^{(-\gamma+\kappa S_n)\Delta t} - 1}{-\gamma + \kappa S_n} = M_n + \frac{\lambda \Delta I}{-\gamma + \kappa S_n} \tag{9}$$

In summary, we replace the system of ordinary differential equations, Eqs. (1-4) by the system of difference equations

$$\begin{aligned} S_{n+1} &= S_n e^{\frac{\kappa \Delta I}{\gamma - \kappa S_n}}, \\ I_{n+1} &= I_n e^{(-\gamma + \kappa S_n)\Delta t}, \\ R_{n+1} &= R_n + \frac{\rho \Delta I}{-\gamma + \kappa S_n}, \\ M_{n+1} &= M_n + \frac{\lambda \Delta I}{-\gamma + \kappa S_n}. \end{aligned} \tag{10}$$

When the parameters vary in time, we have

$$\begin{aligned} S_{n+1} &= S_n e^{\frac{\kappa_n \Delta I}{\gamma - \kappa_n S_n}}, \\ I_{n+1} &= I_n e^{(-\gamma + \kappa_n S_n)\Delta t}, \\ R_{n+1} &= R_n + \frac{\rho_n \Delta I}{-\gamma + \kappa_n S_n}, \\ M_{n+1} &= M_n + \frac{\lambda_n \Delta I}{-\gamma + \kappa_n S_n}, \end{aligned} \tag{11}$$

where $\kappa_n = \kappa(t_n)$, $\lambda_n = \lambda(t_n)$, and $\kappa_n = \kappa(t_n)$. In the linear limit we have

$$\begin{aligned} S_{n+1} &= S_n(1 - \kappa_n I_n \Delta t), \\ I_{n+1} &= I_n[1 + (-\gamma + \kappa_n S_n)\Delta t] = I_n \left[1 + (R_{0n} - 1)\frac{\Delta t}{\tau}\right], \\ R_{n+1} &= R_n + \rho I_n \Delta t, \\ M_{n+1} &= M_n + \lambda I_n \Delta t, \end{aligned} \tag{12}$$

where R_{0n} is the basic reproduction number at time t_n.

II. NON-NEGATIVENESS PROOF

This is an alternative proof of the positivity proof presented in the section 3 of the Appendix. We show below that if the initial values of the state variables $S_0, I_0, R_0, M_0 \geq 0$ of our ODE system given by Eq. (1), then $S(t)$, $I(t)$, $R(t)$, $M(t)$ will always remain ≥ 0. Since the system flow of our model is continuous, any variable has to cross the zero value before becoming negative. If $I = 0$, then $\frac{dI}{dt} = 0$. Hence, $I(t) \geq 0$ for all $t > 0$ since $I_0 > 0$. Consequently, $dM/dt \geq 0$ implying that $M(t) \geq 0$, since $I \geq 0$ and $\lambda(t) \geq 0$. If $R = 0$, then $dR/dt \geq 0$ since $I \geq 0$ and $\rho \geq 0$. Hence, $R(t) \geq 0$ for all t if $R_0 > 0$. Finally, if $S = 0$, $dS/dt = \nu(I + R) \geq 0$.

A. Boundedness proof

If $\mu = \nu = 0$, then, from Eq. (1) of the paper,

$$\frac{d}{dt}(S + I + R + M) = 0. \tag{13}$$

Hence, $S(t) + I(t) + R(t) + M(t) = S_0 + I_0 + R_0 + M_0 = 1$. As the initial condition is $S_0 = 1 - 1/P_0$ and $I_0 = 1/P_0$, $R_0 = M_0 = 0$ and state variables are all non-negative, as shown above, we obtain that $0 \leq S(t), I(t), R(t), M(t) \leq 1$. In the case of Eq. (1) with μ and ν positive, we can write

$$\frac{d}{dt}(S + I + R + M) = (\nu - \mu)(S + I + R) \leq (\nu - \mu)(S + I + R + M), \tag{14}$$

assuming $\nu > \mu$. Hence, we obtain $S(t) + I(t) + R(t) + M(t) \leq e^{(\nu-\mu)t}$. Therefore, as $S(t)$, $I(t)$, $R(t)$, and $M(t)$ are positive, they are all bounded in any given finite time interval.

B. Positiveness of $I(t)$

The rate in which $I(t)$ decreases the fastest occurs when $\kappa(t) = 0$. Hence, at most $I(t)$ decreases exponentially with rate $\mu + 1/\tau$. Therefore, in a finite time interval, $I(t) > 0$ if $I_0 > 0$.

III. DATA

All data in the manuscript are contained in the following databases. The data from Germany and Brazil, was obtained from the site https://data.humdata.org/dataset/novel-coronavirus-2019-ncov-cases which contains the data compiled by the Johns Hopkins University Center for Systems Science and Engineering (JHU CSSE). The specific CSV files for the confirmed, recovered, and deceased are:

- time_series_covid19_confirmed_global_narrow.csv
- time_series_covid19_deaths_global_narrow.csv
- time_series_covid19_recovered_global_narrow.csv

The header of all three files is:

```
Province/State,Country/Region,Lat,Long,Date,Value,ISO 3166-1 Alpha 3-Codes
Region Code,Sub-region Code,Intermediate Region Code
```

The data from Brazilian states and cities are obtained from the url: https://data.brasil.io/dataset/covid19/caso.csv.gz

The header file is

```
date,state,city,place_type,confirmed,deaths,order_for_place,is_last,
estimated_population_2019,city_ibge_code,confirmed_per_100k_inhabitants,
death_rate
```

IV. CODE DESCRIPTION AND USAGE

The programs, written in Python, used in the manuscript are: activeWorld.py, covidBR.py, and sensibilityAnalysis.py. Fig. 1 is generated by sensibilityAnalysis.py. The results for Germany (Figs.2-3) and Brazil (Figs. 4-5) are generated by activeWorld.py. The results for Paraíba (Figs. 6-7) and Campina Grande (Figs. 8-9) are generated by covidBR.py

A. Description of activeWorld.py

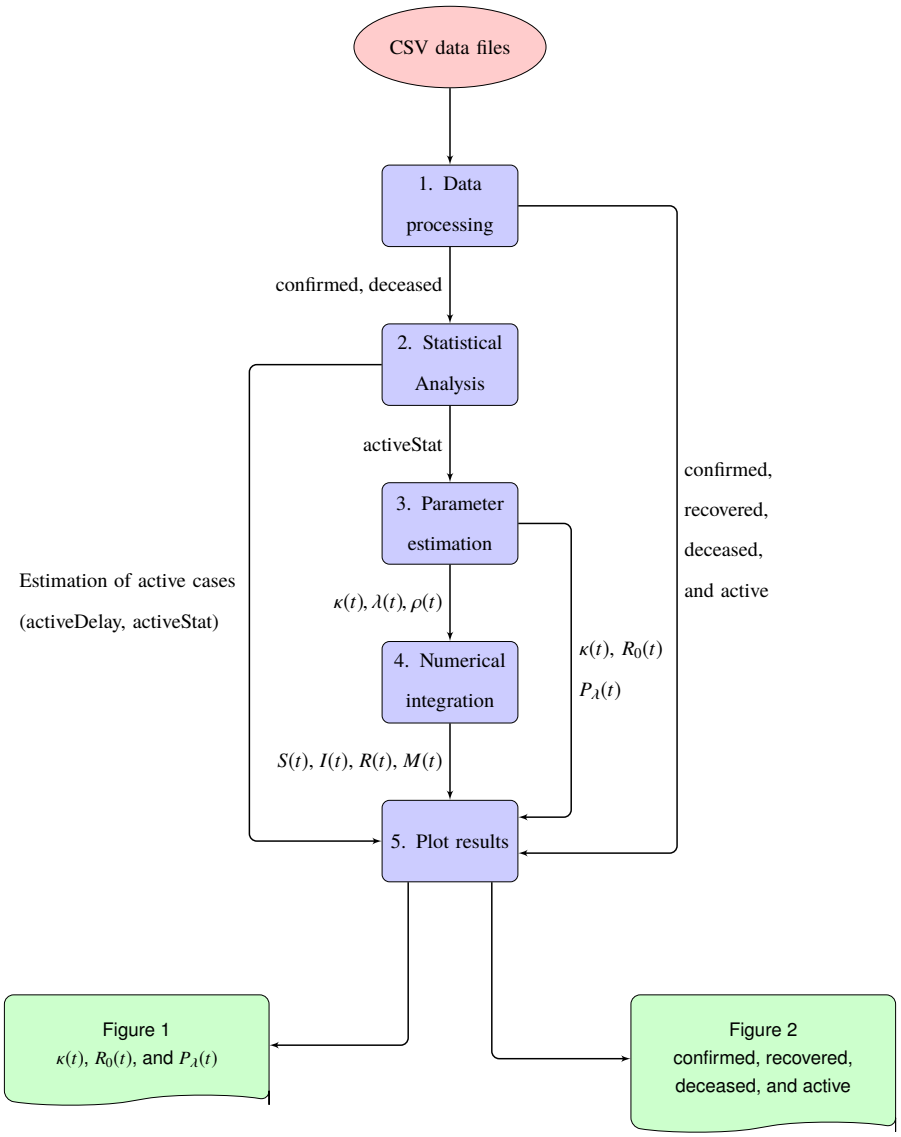

FIG. 1. Flowchart of activeWorld.py

1. Data processing

```
# 1. Data processing
csvFiles = ['time_series_covid19_confirmed_global_narrow.csv',
    'time_series_covid19_deaths_global_narrow.csv',
    'time_series_covid19_recovered_global_narrow.csv']
# a. Open confirmed datafile create dataframe
df = pd.read_csv(csvFiles[0], sep =',', skiprows=0, header=0, names=["Prov
# Filter country data
df = df[df['Country/Region'].str.contains(country)]
df = df.sort_values('Date', ascending=True)
confirmed = df.Value.to_numpy().astype(int)
ind0 = (confirmed!=0).argmax() # first nonzero index of confirmed
ind0 += offset
confirmed = confirmed[ind0:] # cut off the leading zeros
print('ind0', ind0, 'offset', offset)
dates = pd.to_datetime(df.Date) # covert string dates to datetime format
dates = dates[ind0:]
# b. Open deaths datafile, create dataframe.
df = pd.read_csv(csvFiles[1], sep =',', skiprows=0, header=0, names=["Prov
df = df[df['Country/Region'].str.contains(country)]
deaths = df.Value.to_numpy().astype(int)
deaths = deaths[::-1]
deaths = deaths[ind0:]
indFirstDeath = (deaths!=0).argmax()
print(indFirstDeath, 'First death on', dates.iloc[indFirstDeath], deaths[i
# c. Open recovered datafile, create dataframe.
df = pd.read_csv(csvFiles[2], sep =',', skiprows=0, header=0, names=["Prov
df = df[df['Country/Region'].str.contains(country)]
recovered = df.Value.to_numpy().astype(int)
recovered = recovered[::-1]
recovered = recovered[ind0:]
```

```
# d. generate active cases from data
activeCases = confirmed-deaths-recovered

# e. Obtain country population (World Bank data)
df = pd.read_csv('worldPopulation.csv', sep =',', skiprows=0, header=0,
    names=['Country_Code','Ranking','Country','Population'])
df = df[df['Country'].str.contains(country)]
population = df['Population'].iloc[0]
population = population.replace(' ','')
population = population.replace(',','')
print(df)
P_0 = int(population)*1000
print(P_0)
```

2. *Statistical Analysis*

```
# 2. Statistical analysis
# Active cases delay estimate
activeDelay = confirmed[delay:]-confirmed[:-delay]
n_avg=delay
day = 24 # day in hours
dt = 1.0/day  # integration time-step
tau = (1+n_avg*day)*dt # average time duration of infection
nu = mu= 0.0
# Active cases statistical estimate
q = n_avg/(1+n_avg) # probability to remain sick after 1 day
newConf = np.diff(confirmed) # casos novos confirmados ao dia
newDead = np.diff(deaths) # casos novos confirmados ao dia
N = len(newConf)
activeStat = np.ones(N)
n_pow = np.arange(N)
n_pow_des = n_pow[::-1]
q_to_n = q**n_pow_des
```

```
for i in n_pow:
  activeStat[i] = np.sum(newConf[:i]*q_to_n[N-i:])
```

3. *Parameter estimation*

```
# 3. Parameter estimation
# a. Contagion rate function
contagionRate = []
N_as = len(activeStat)
for i in np.arange(0, N_as):
  if i+7<=N_as:
    XX = np.arange(i, i+7)
    ZZ = activeStat[i:i+7]
  else:
    XX = np.arange(i-7, i)
    ZZ = activeStat[i-7:i]
  if len(XX)==len(ZZ):
    slope, intercept, r_value, p_value, std_err = stats.linregress(XX, ZZ)
    I_avg = np.sum(ZZ)/7
    if I_avg==0:
      ka_t=0
    else:
      ka_t = slope/I_avg+1.0/tau
    #print(i, ka_t)
    if ka_t<0:
      ka_t = 0
    elif ka_t>cutoff:
      ka_t = cutoff
    contagionRate.append(ka_t)
print(len(dates[1:-7]), len(contagionRate))

############################################################
def kappa(t):
```

```
  ind = int(1.0*t)
  N=len(contagionRate)
  if ind>=N:
    ind = N-1
  ka = contagionRate[ind]
  if ind>0 and ind<N and ka==0:
    ka = (contagionRate[ind-1]+contagionRate[ind+1])/2
  return ka
############################################################
# vectorize contagion rate
kappa_t_vec = []
n_max = len(confirmed)-1 # in days
tt = np.arange(0.0, n_max, dt)
for ind, t in enumerate(tt[::day]):
  ka = kappa(t)
  kappa_t_vec.append(ka)
kappa_t_vec = np.array(kappa_t_vec)
# generate Rt
R0_t = tau*kappa_t_vec

# b. Lethality probability
P_let = np.zeros(N)
for i in n_pow[:-1]:
    denom = np.sum(newConf[:i]*q_to_n[N-i:])
    if denom>0:
      s = newDead[i]/denom
    else:
      s=0
    P_let[i] = s/(1-q)
print('P_let')
# c. Lethality and recovery rates
la = P_let/tau
rho = 1.0/tau-la
```

```
P_rho=1-P_let
```

4. Numerical integration

```
# 4. Numerical model epidemic evolution (modified SIR model)
#############################################################
def get_l_r(t):
  ind = int(1.0*t)
  N=len(P_let)
  if ind>=N:
    ind = N-1
  la = P_let[ind]/tau
  rho = 1.0/tau - la
  return la, rho
#############################################################
def derivs (x, t): # return derivatives of the array x
   S = x[0] # susceptíveis
   I = x[1] # infectados
   R = x[2] # Recuperados
   M = x[3] # Mortos
   T = S+I+R
   l, r = get_l_r(t)
   dSdt = nu*T-mu*S-kappa(t)*S*I
   dIdt = -mu*I+kappa(t)*S*I-r*I-l*I
   dRdt = r*I-mu*R
   dMdt = l*I
   return [dSdt, dIdt, dRdt, dMdt]
#############################################################
C_0 = confirmed[0]
R_0 = recovered[0]
D_0 = deaths[0]
A_0 = C_0-R_0-D_0
print('C_0', C_0, 'A_0', A_0, 'R_0', R_0, 'D_0', D_0)
```

```
yinit = [1.0-C_0/P_0, A_0/P_0, R_0/P_0, D_0/P_0] # initial values
y = scipy.integrate.odeint(derivs, yinit, tt)
S = y[:, 0]
I = y[:, 1]
R = y[:, 2]
M = y[:, 3]
```

B. Description of the code covidBR.py

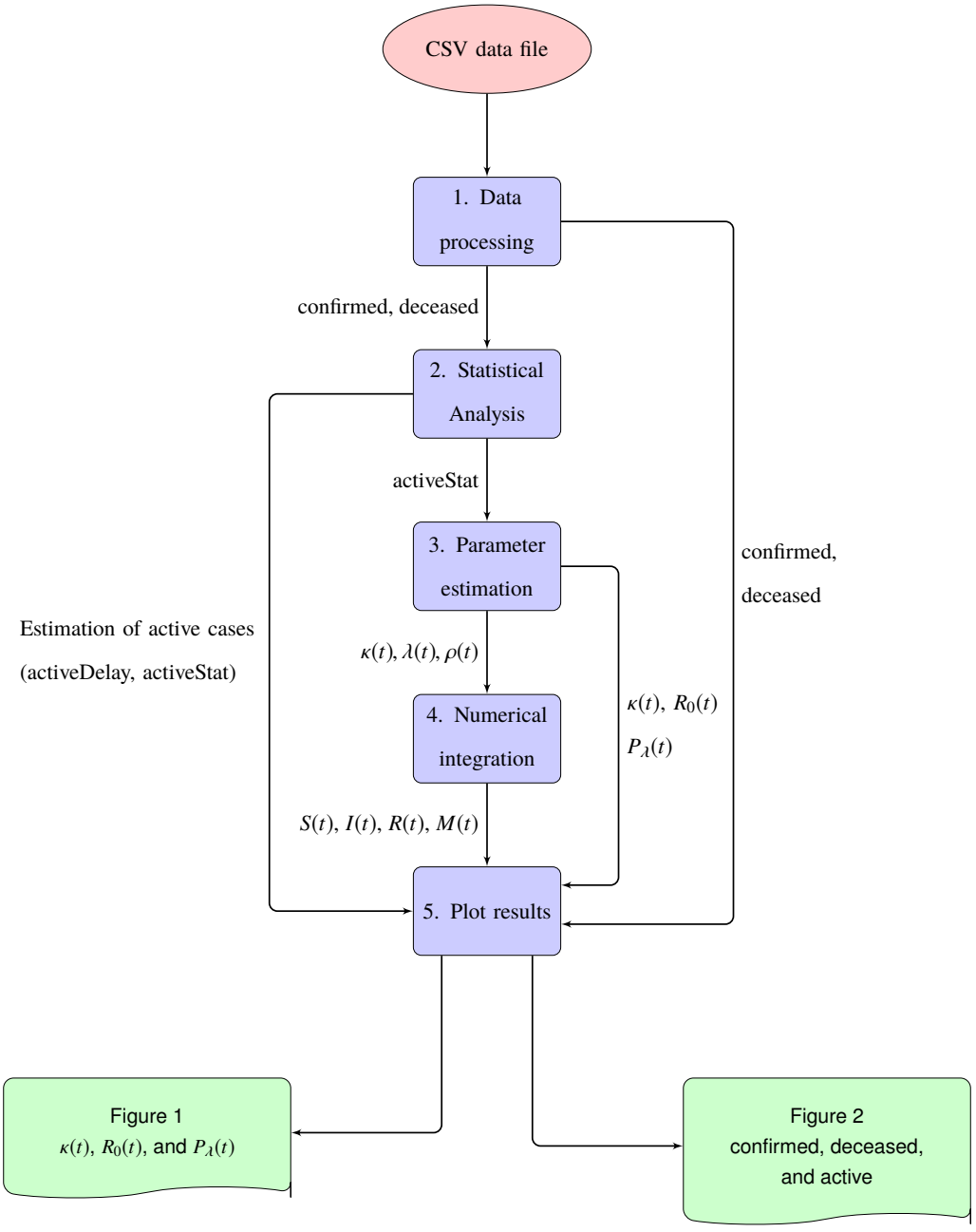

FIG. 2. Flowchart of covidBR.py. The code is very similar to activeWorld.py

V. FORECASTING

A. Markov Chain

The list with the last 3 weeks of $\kappa(t)$ data is

```
[0.08240635 0.06846006 0.06489878 0.06836595 0.07755655 0.08288899
 0.07865233 0.06396189 0.05664613 0.05661004 0.05664208 0.06474608
 0.06957361 0.06598644 0.05260842 0.05664613 0.05661004 0.05664208
 0.06474608 0.06957361 0.06598644]
```

The list of differences $\Delta\kappa_i = \kappa(t_i) - \kappa(t_{i-1})$ is

```
[-1.39462919e-02 -3.56128297e-03  3.46716808e-03  9.19060214e-03
  5.33243964e-03 -4.23665992e-03 -1.46904361e-02 -7.31575917e-03
 -3.60917017e-05  3.20364767e-05  8.10400363e-03  4.82752846e-03
 -3.58716748e-03 -1.33780257e-02  4.03771634e-03 -3.60917017e-05
  3.20364767e-05  8.10400363e-03  4.82752846e-03 -3.58716748e-03]
```

If $\Delta\kappa_i < 0$, in the above list, replace it with 0, otherwise replace it with 1. We then obtain the list

```
[0 0 1 1 1 0 0 0 0 1 1 1 0 0 1 0 1 1 1 0]
```

The list of lengths of the continuous sequences of 1's are

```
[3, 3, 1, 3].
```

The list of lengths of the continuous sequences of 0's are

```
[2, 4, 2, 1, 1].
```

Based on this, the average length of positive increments of the contagion rate is $\bar{n}_+ = 2.5$ days and of negative increments is $\bar{n}_- = 2.0$ days. Based on this we find the transition probabilities

$$\begin{aligned} q_{++} &= \frac{\bar{n}_+}{1+\bar{n}_+} \approx 0.714 \\ p_{+-} &= 1 - q_{++} \approx 0.286 \\ q_{--} &= \frac{\bar{n}_-}{1+\bar{n}_-} \approx 0.667 \\ p_{-+} &= 1 - q_{--} \approx 0.333 \end{aligned} \tag{15}$$

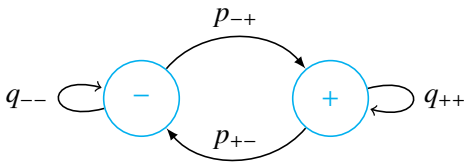

FIG. 3. Markov chain diagram. The transition probabilities are given by Eq. (15) and were obtained according to the algorithm shown above.

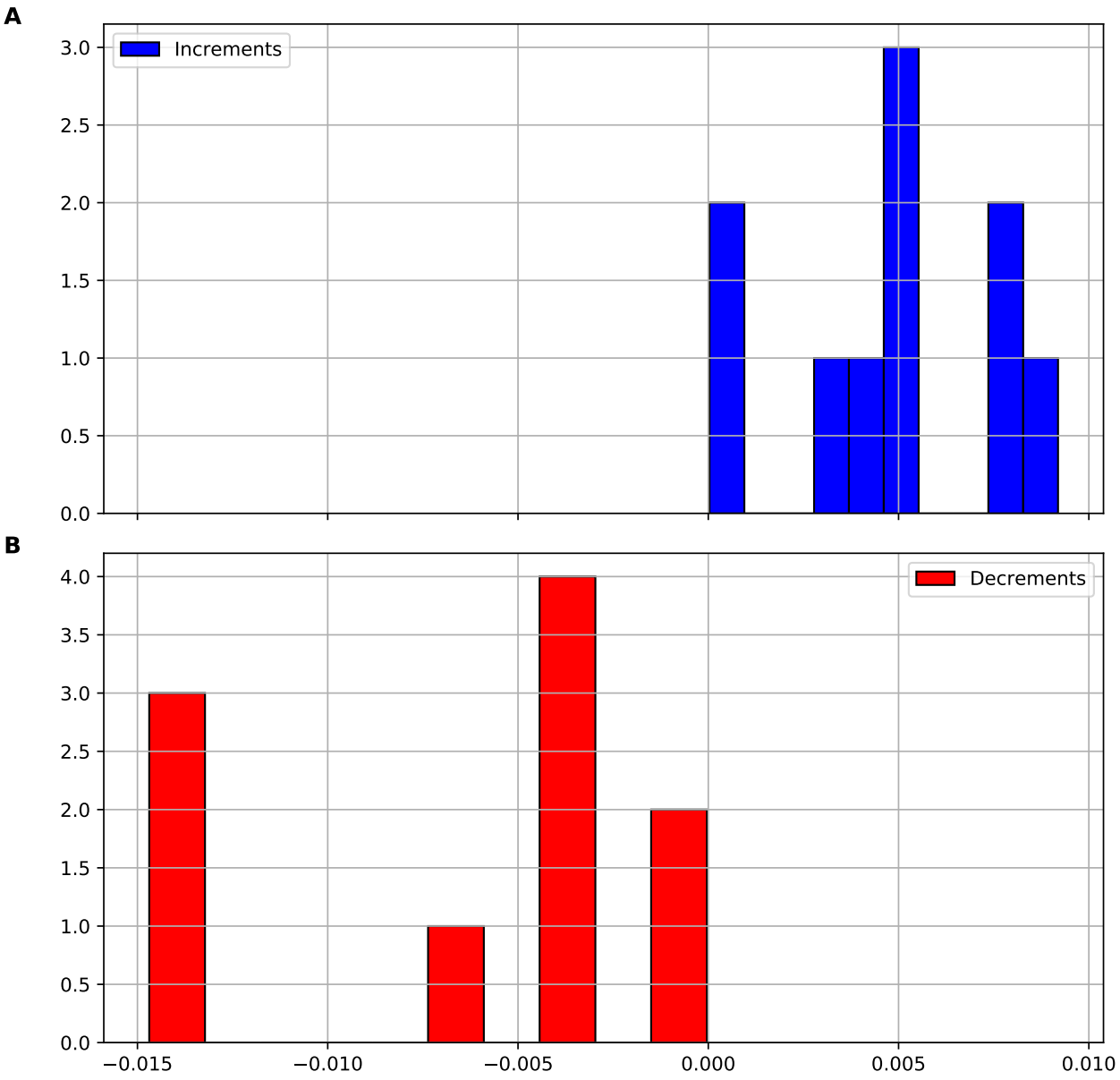

FIG. 4. **A** Probability distribution for the increments of $\kappa(t)$ in the next two weeks based on the time series of the past 20 days of data obtained from $\Delta\kappa_i$. **B** Probability distribution for the decrements of $\kappa(t)$.

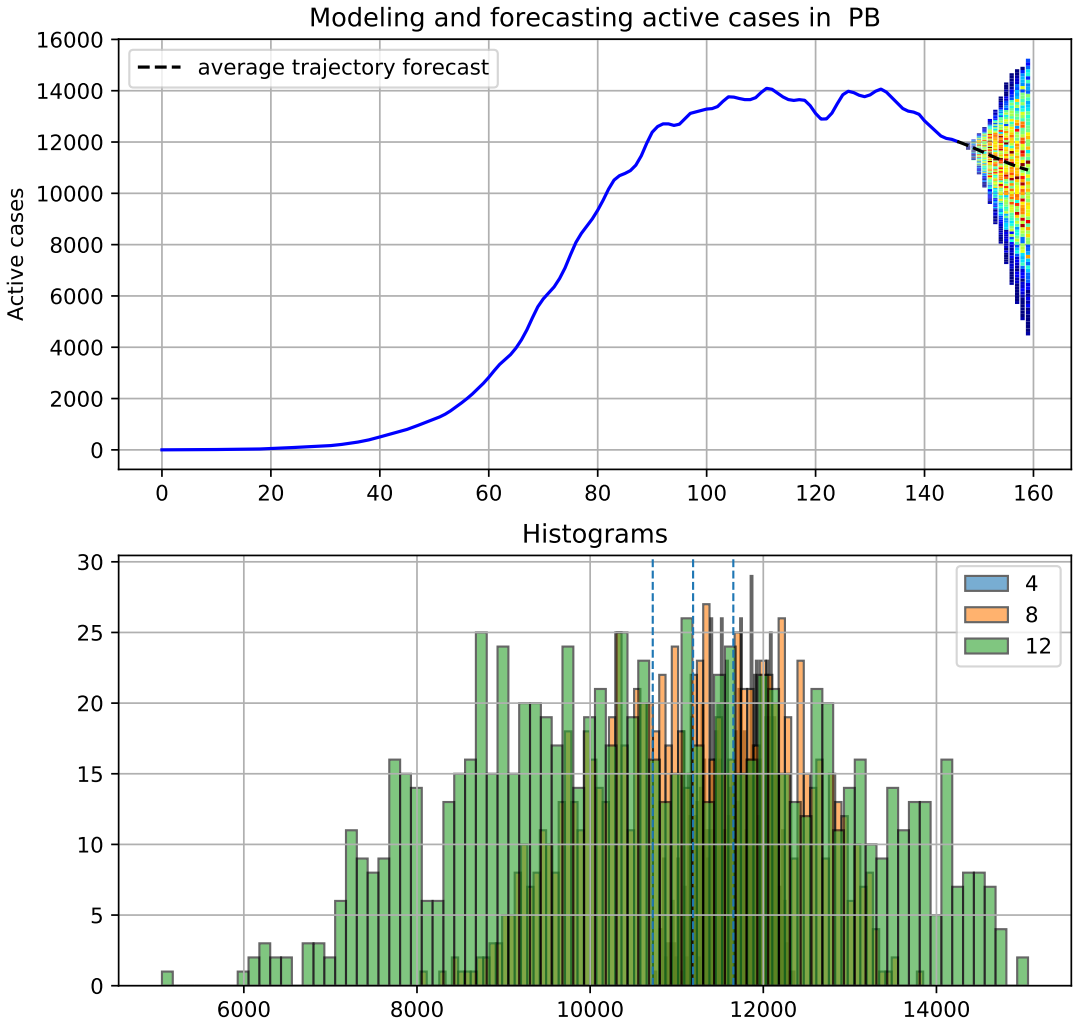

FIG. 5. **A** Time series of active cases and the forecast for the next two weeks. **B** Histograms of slices of trajectories for the 4th, 8th, and 12th day from the date of the last active data point. The vertical dashed lines indicate the mean values.

VI. HOW TO RUN THE CODE

The command line to run activeWorld.py from a Linux terminal is

```
python3 activeWorld.py config.txt
```

The file config.txt has the following lines in the case of Germany:

```
country,Germany
delay,13
offset,41
cutoff,0.41
```

The file config.txt has the following lines in the case of Brazil:

```
country,Brazil
delay,14
offset,22
cutoff,0.43
```

The command line to run covidBR.py from a Linux terminal is

```
python3 covidBR.py config.txt
```

The file config_PB.txt has the following lines in the case of Paraíba:

```
state,PB
city,_
delay,14
offset,0
cutoff,0.38
```

The file configCampinaGrandePB.txt has the following lines in the case of Campina Grande:

```
state,PB
city,Campina Grande
delay,14
offset,21
cutoff,0.35
```