Chaos, Solitons and Fractals 141 (2020) 110311

Contents lists available at ScienceDirect

Chaos, Solitons and Fractals

Nonlinear Science, and Nonequilibrium and Complex Phenomena

journal homepage: www.elsevier.com/locate/chaos

Letter to the editor

Corrigendum to “Mathematical modeling of COVID-19 transmission dynamics with a case study of Wuhan” [Chaos Solitons Fractals 135 (2020), 109846]

The following corrections need to be done to Ndaïrou et al. [1]:

1. In Section 3.1, the Jacobian matrix $J_{\mathcal{F}}$ has the entry in the first row and the third column interchanged with the entry in the first row and the fourth column. In other words, the correct matrix is

$$J_{\mathcal{F}} = \begin{bmatrix} 0 & \beta & \beta' & \beta l \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

2. In Table 1, where the values of the model parameters used in the numerical simulations are given, the values associated with δ_i, δ_p and δ_h are wrong. The correct values are:

$$\delta_i = \delta_p = \delta_h = \frac{1}{23}\ \text{day}^{-1}.$$

3. For the parameters used in the simulations, the basic reproduction number is not $R_0 = 0.945$, as wrongly given in Ndaïrou et al. [1], but should be corrected to $R_0 = 4.375$. Please see Appendix A for all details, where computations are carried out in the free and open-source computer algebra system SageMath.
4. In (7), the Jacobian matrix J_M has the entry in the first row and the third column interchanged with the entry in the first row and the fourth column. In other words, the correct matrix is

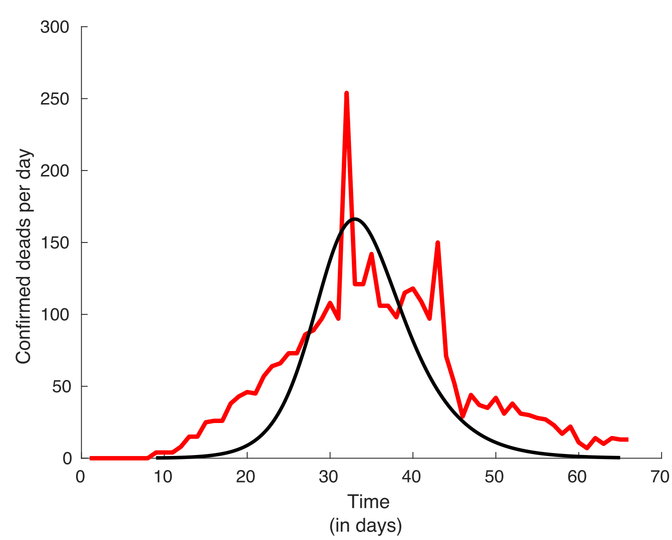
$$J_M = \begin{bmatrix} -\kappa & \beta & \beta' & l\beta \\ \kappa\rho_1 & -\varpi_i & 0 & 0 \\ \kappa\rho_2 & 0 & -\varpi_p & 0 \\ 0 & \gamma_a & \gamma_a & -\varpi_h \end{bmatrix}.$$

5. The values given in Table 2, of the sensitivity of R_0 evaluated for the parameter values used in the simulations, need to be corrected as follows:

Parameter	Sensitivity index
β	0.999
l	0.729
β'	0.00139
κ	0.000
ρ_1	0.997
ρ_2	0.00265
γ_a	−0.0210
γ_i	−0.215
γ_r	−0.671
δ_i	−0.0346
δ_p	−0.0000919
δ_h	−0.0583

Please see Appendix A for all details, where the computations to obtain such values are carried out in the free and open-source computer algebra system SageMath.

6. Figure 3 needs to be substituted by the following one:



This figure was obtained in Matlab, and the full code is given in Appendix B.

The authors of [1] would like to apologize for any inconvenience caused.

https://doi.org/10.1016/j.chaos.2020.110311

Appendix A. SageMath code to compute R_0 and its sensitivity indexes

```
sage:#the constant parameters values
sage: beta = 2.55; l = 1.56; betaprim = 7.65; kappa = 0.25; rho_1 = 0.580; rho_2 = 0.001;
gamma_a = 0.94;
sage: gamma_i = 0.27; gamma_r = 0.5; delta_i = 1/23; delta_p = 1/23; delta_h = 1/23; N = 11000000
sage: R_0 = ((beta*gamma_a*l*rho_2 +
sage: betaprim*rho_2*(gamma_r + delta_h))*(gamma_a + gamma_i + delta_i)
... + (beta*gamma_a*l*rho_1 +
...
sage: beta*rho_1*(gamma_r + delta_h))*(gamma_a + gamma_i + delta_p))/((gamma_r
sage: + delta_h)*(gamma_a + gamma_i + delta_i)*(gamma_a + gamma_i + delta_p))
sage: print R_0
4.37513184233091
sage: #Sentivity of beta
sage: S_beta = ((delta_i + gamma_a + gamma_i)*gamma_a*l*rho_2 + (gamma_a*l*rho_1 +
sage: (delta_h + gamma_r)*rho_1)*(delta_p + gamma_a +
sage: gamma_i))*beta/((beta*gamma_a*l*rho_2 + betaprim*(delta_h +
sage: gamma_r)*rho_2)*(delta_i + gamma_a + gamma_i) + (beta*gamma_a*l*rho_1 +
sage: beta*(delta_h + gamma_r)*rho_1)*(delta_p + gamma_a + gamma_i))
sage: print S_beta
0.998605066564884
sage: # sensitivity of l
sage: S_l = (beta*(delta_p + gamma_a + gamma_i)*gamma_a*rho_1 + beta*(delta_i +
sage: gamma_a + gamma_i)*gamma_a*rho_2)*l/((beta*gamma_a*l*rho_2 +
sage: betaprim*(delta_h + gamma_r)*rho_2)*(delta_i + gamma_a + gamma_i) +
sage: (beta*gamma_a*l*rho_1 + beta*(delta_h + gamma_r)*rho_1)*(delta_p +
sage: gamma_a + gamma_i))
sage: print S_l
0.728917935775866
sage: #sensitivity analysis of betaprim
sage: S_betaprim = betaprim*(delta_h + gamma_r)*(delta_i + gamma_a +
sage: gamma_i)*rho_2/((beta*gamma_a*l*rho_2 + betaprim*(delta_h +
sage: gamma_r)*rho_2)*(delta_i + gamma_a + gamma_i) + (beta*gamma_a*l*rho_1 +
sage: beta*(delta_h + gamma_r)*rho_1)*(delta_p + gamma_a + gamma_i))
sage: print S_betaprim
0.00139493343511561
sage: #sensitivity of rho_1
sage: S_rho1 = (beta*gamma_a*l + beta*(delta_h + gamma_r))*(delta_p + gamma_a +
sage: gamma_i)*rho_1/((beta*gamma_a*l*rho_2 + betaprim*(delta_h +
sage: gamma_r)*rho_2)*(delta_i + gamma_a + gamma_i) + (beta*gamma_a*l*rho_1 +
sage: beta*(delta_h + gamma_r)*rho_1)*(delta_p + gamma_a + gamma_i))
sage: print S_rho1
0.997350474592809
sage: #sensitivity of rho_2
sage: S_rho2 = (beta*gamma_a*l + betaprim*(delta_h + gamma_r))*(delta_i + gamma_a +
sage: gamma_i)*rho_2/((beta*gamma_a*l*rho_2 + betaprim*(delta_h +
sage: gamma_r)*rho_2)*(delta_i + gamma_a + gamma_i) + (beta*gamma_a*l*rho_1 +
sage: beta*(delta_h + gamma_r)*rho_1)*(delta_p + gamma_a + gamma_i))
sage: print S_rho2
0.00264952540719111
sage: #sensitivity of gamma_a
sage: S_gammaa = (delta_h + gamma_r)*(delta_i + gamma_a + gamma_i)*(delta_p + gamma_a +
sage: gamma_i)*gamma_a*((beta*(delta_p + gamma_a + gamma_i)*l*rho_1 +
sage: beta*gamma_a*l*rho_1 + beta*(delta_i + gamma_a + gamma_i)*l*rho_2 +
sage: beta*gamma_a*l*rho_2 + beta*(delta_h + gamma_r)*rho_1 +
sage: betaprim*(delta_h + gamma_r)*rho_2)/((delta_h + gamma_r)*(delta_i +
sage: gamma_a + gamma_i)*(delta_p + gamma_a + gamma_i)) -
sage: ((beta*gamma_a*l*rho_2 + betaprim*(delta_h + gamma_r)*rho_2)*(delta_i +
sage: gamma_a + gamma_i) + (beta*gamma_a*l*rho_1 + beta*(delta_h +
sage: gamma_r)*rho_1)*(delta_p + gamma_a + gamma_i))/((delta_h +
sage: gamma_r)*(delta_i + gamma_a + gamma_i)*(delta_p + gamma_a + gamma_i)^2)
sage: - ((beta*gamma_a*l*rho_2 + betaprim*(delta_h + gamma_r)*rho_2)*(delta_i
sage: + gamma_a + gamma_i) + (beta*gamma_a*l*rho_1 + beta*(delta_h +
sage: gamma_r)*rho_1)*(delta_p + gamma_a + gamma_i))/((delta_h +
sage: gamma_r)*(delta_i + gamma_a + gamma_i)^2*(delta_p + gamma_a +
```

```
sage: gamma_i)))/((beta*gamma_a*l*rho_2 + betaprim*(delta_h +
sage: gamma_r)*rho_2)*(delta_i + gamma_a + gamma_i) + (beta*gamma_a*l*rho_1 +
sage: beta*(delta_h + gamma_r)*rho_1)*(delta_p + gamma_a + gamma_i))
sage: print S_gammaa
-0.0209953489969400
sage: #sensitivity of Gamma_i
sage: S_gammai = (delta_h + gamma_r)*(delta_i + gamma_a + gamma_i)*(delta_p + gamma_a +
sage: gamma_i)*gamma_i*((beta*gamma_a*l*rho_1 + beta*gamma_a*l*rho_2 +
sage: beta*(delta_h + gamma_r)*rho_1 + betaprim*(delta_h +
sage: gamma_r)*rho_2)/((delta_h + gamma_r)*(delta_i + gamma_a +
sage: gamma_i)*(delta_p + gamma_a + gamma_i)) - ((beta*gamma_a*l*rho_2 +
sage: betaprim*(delta_h + gamma_r)*rho_2)*(delta_i + gamma_a + gamma_i) +
sage: (beta*gamma_a*l*rho_1 + beta*(delta_h + gamma_r)*rho_1)*(delta_p +
sage: gamma_a + gamma_i))/((delta_h + gamma_r)*(delta_i + gamma_a +
sage: gamma_i)*(delta_p + gamma_a + gamma_i)^2) - ((beta*gamma_a*l*rho_2 +
sage: betaprim*(delta_h + gamma_r)*rho_2)*(delta_i + gamma_a + gamma_i) +
sage: (beta*gamma_a*l*rho_1 + beta*(delta_h + gamma_r)*rho_1)*(delta_p +
sage: gamma_a + gamma_i))/((delta_h + gamma_r)*(delta_i + gamma_a +
sage: gamma_i)^2*(delta_p + gamma_a + gamma_i)))/((beta*gamma_a*l*rho_2 +
sage: betaprim*(delta_h + gamma_r)*rho_2)*(delta_i + gamma_a + gamma_i) +
sage: (beta*gamma_a*l*rho_1 + beta*(delta_h + gamma_r)*rho_1)*(delta_p +
sage: gamma_a + gamma_i))
sage: print S_gammai
-0.215400624349636
sage: #sensitivity of gamma_r
sage: S_gammar = (delta_h + gamma_r)*(delta_i + gamma_a + gamma_i)*(delta_p + gamma_a +
sage: gamma_i)*gamma_r*((beta*(delta_p + gamma_a + gamma_i)*rho_1 +
sage: betaprim*(delta_i + gamma_a + gamma_i)*rho_2)/((delta_h +
sage: gamma_r)*(delta_i + gamma_a + gamma_i)*(delta_p + gamma_a + gamma_i)) -
sage: ((beta*gamma_a*l*rho_2 + betaprim*(delta_h + gamma_r)*rho_2)*(delta_i +
sage: gamma_a + gamma_i) + (beta*gamma_a*l*rho_1 + beta*(delta_h +
sage: gamma_r)*rho_1)*(delta_p + gamma_a + gamma_i))/((delta_h +
sage: gamma_r)^2*(delta_i + gamma_a + gamma_i)*(delta_p + gamma_a +
sage: gamma_i)))/((beta*gamma_a*l*rho_2 + betaprim*(delta_h +
sage: gamma_r)*rho_2)*(delta_i + gamma_a + gamma_i) + (beta*gamma_a*l*rho_1 +
sage: beta*(delta_h + gamma_r)*rho_1)*(delta_p + gamma_a + gamma_i))
sage: print S_gammar
-0.670604500913797
sage: #sensitivity of delta_i
sage: S_deltai = (delta_h + gamma_r)*(delta_i + gamma_a + gamma_i)*delta_i*(delta_p +
sage: gamma_a + gamma_i)*((beta*gamma_a*l*rho_2 + betaprim*(delta_h +
sage: gamma_r)*rho_2)/((delta_h + gamma_r)*(delta_i + gamma_a +
sage: gamma_i)*(delta_p + gamma_a + gamma_i)) - ((beta*gamma_a*l*rho_2 +
sage: betaprim*(delta_h + gamma_r)*rho_2)*(delta_i + gamma_a + gamma_i) +
sage: (beta*gamma_a*l*rho_1 + beta*(delta_h + gamma_r)*rho_1)*(delta_p +
sage: gamma_a + gamma_i))/((delta_h + gamma_r)*(delta_i + gamma_a +
sage: gamma_i)^2*(delta_p + gamma_a + gamma_i)))/((beta*gamma_a*l*rho_2 +
sage: betaprim*(delta_h + gamma_r)*rho_2)*(delta_i + gamma_a + gamma_i) +
sage: (beta*gamma_a*l*rho_1 + beta*(delta_h + gamma_r)*rho_1)*(delta_p +
sage: gamma_a + gamma_i))
sage: print S_deltai
-0.0345941891985019
sage: #sensitivity of delta_p
sage: S_deltap = (delta_h + gamma_r)*(delta_i + gamma_a + gamma_i)*(delta_p + gamma_a +
sage: gamma_i)*delta_p*((beta*gamma_a*l*rho_1 + beta*(delta_h +
sage: gamma_r)*rho_1)/((delta_h + gamma_r)*(delta_i + gamma_a +
sage: gamma_i)*(delta_p + gamma_a + gamma_i)) - ((beta*gamma_a*l*rho_2 +
sage: betaprim*(delta_h + gamma_r)*rho_2)*(delta_i + gamma_a + gamma_i) +
sage: (beta*gamma_a*l*rho_1 + beta*(delta_h + gamma_r)*rho_1)*(delta_p +
sage: gamma_a + gamma_i))/((delta_h + gamma_r)*(delta_i + gamma_a +
sage: gamma_i)*(delta_p + gamma_a + gamma_i)^2))/((beta*gamma_a*l*rho_2 +
sage: betaprim*(delta_h + gamma_r)*rho_2)*(delta_i + gamma_a + gamma_i) +
sage: (beta*gamma_a*l*rho_1 + beta*(delta_h + gamma_r)*rho_1)*(delta_p +
sage: gamma_a + gamma_i))
sage: print S_deltap
-0.000919016790562303
```

```
sage: #sensitivity of delta_h
sage: S_deltah = (delta_h + gamma_r)*delta_h*(delta_i + gamma_a + gamma_i)*(delta_p +
sage: gamma_a + gamma_i)*((beta*(delta_p + gamma_a + gamma_i)*rho_1 +
sage: betaprim*(delta_i + gamma_a + gamma_i)*rho_2)/((delta_h +
sage: gamma_r)*(delta_i + gamma_a + gamma_i)*(delta_p + gamma_a + gamma_i)) -
sage: ((beta*gamma_a*l*rho_2 + betaprim*(delta_h + gamma_r)*rho_2)*(delta_i +
sage: gamma_a + gamma_i) + (beta*gamma_a*l*rho_1 + beta*(delta_h +
sage: gamma_r)*rho_1)*(delta_p + gamma_a + gamma_i))/((delta_h +
sage: gamma_r)^2*(delta_i + gamma_a + gamma_i)*(delta_p + gamma_a +
sage: gamma_i)))/((beta*gamma_a*l*rho_2 + betaprim*(delta_h +
sage: gamma_r)*rho_2)*(delta_i + gamma_a + gamma_i) + (beta*gamma_a*l*rho_1 +
sage: beta*(delta_h + gamma_r)*rho_1)*(delta_p + gamma_a + gamma_i))
sage: print S_deltah
-0.0583134348620693
```

Appendix B. Matlab code for Figures 2 and 3 of [1]

In our Matlab code we use the `fde12` routine freely available from MATLAB central [2].

```
clear all
realdata = [0 6; 1 12; 2 19; 3 25; 4 31; 5 38; 6 44; 7 60; 8 80;9 131;10 131;
11 259; 12 467; 13 688; 14 776; 15 1776; 16 1460; 17 1739; 18 1984; 19 2101; 20 2590;
21 2827; 22 3233; 23 3892; 24 3697; 25 3151; 26 3387; 27 2653; 28 2984; 29 2473; 30 2022;
31 1820; 32 1998; 33 1506; 34 1278; 35 2051; 36 1772; 37 1891; 38 399; 39 894; 40 397;
41 650; 42 415; 43 518; 44 412; 45 439; 46 441; 47 435; 48 579; 49 206; 50 130;
51 120; 52 143; 53 146; 54 102; 55 46; 56 45; 57 20; 58 31; 59 26; 60 11;
61 18; 62 27; 63 29; 64 39; 65 39];
aux = size(realdata);
deadpeople = [0 0; 1 0; 2 0; 3 0; 4 0; 5 0; 6 0; 7 0; 8 4; 9 4; 10 4;
11 8;12 15; 13 15; 14 25; 15 26; 16 26; 17 38; 18 43; 19 46; 20 45;
21 57; 22 64; 23 66; 24 73; 25 73; 26 86; 27 89; 28 97; 29 108;
30 97; 31 254; 32 121; 33 121; 34 142; 35 106; 36 106; 37 98;
38 115; 39 118; 40 109;41 97; 42 150; 43 71; 44 52; 45 29; 46 44;
47 37; 48 35; 49 42; 50 31;51 38; 52 31; 53 30; 54 28; 55 27; 56 23;
57 17; 58 22; 59 11; 60 07;61 14; 62 10; 63 14; 64 13; 65 13];
t0 = 0;
tend = realdata(aux(1),1);
time = t0:tend;
t0dead = deadpeople(1,1);
timedead = t0dead:tend;
totaldead = deadpeople(:,2);
deadpeople = [0 0; 1 0; 2 0; 3 0; 4 0; 5 0; 6 0; 7 0; 8 4; 9 4; 10 4;
11 8;12 15; 13 15; 14 25; 15 26; 16 26; 17 38; 18 43; 19 46; 20 45;
21 57; 22 64; 23 66; 24 73; 25 73; 26 86; 27 89; 28 97];
totalill = realdata(:,2);
beta = 2.55;
ell = 1.56;
betap = 3*beta;
kappa = 0.250;
rho1 = 0.580;
rho2 = 0.001;
gammaa = 0.94;
gammai = 0.27;
gammar = 0.500;
N = 11000000/(250);
initialvalue = realdata(1,2);
p0 = 5;
e0 = 0;
i0 = initialvalue-p0;
s0 = N-i0;
a0 = 0;
h0 = 0;
r0 = 0;
d0 = 0;
stepsize = 0.001;
delta = 1/(23);
system = @(t,X)[
```

```
-beta.*X(3).*X(1)./N-ell.*beta.*X(6).*X(1)./N-betap.*X(4).*X(1)./N;
beta.*X(3).*X(1)./N+ell.*beta.*X(6).*X(1)./N+betap.*X(4).*X(1)./N-kappa.*X(2);
kappa.*rho1.*X(2)-(gammaa+gammai).*X(3)-delta.*X(3);
kappa.*rho2.*X(2)-(gammaa+gammai).*X(4)-delta.*X(4);
kappa.*(1-rho1-rho2).*X(2);
gammaa.*(X(3)+X(4))-gammar.*X(6)-delta.*X(6);
gammai.*(X(3)+X(4))+gammar.*X(7);
];
[ts1,ys1] = fde12(1,system,t0,tend,[s0;e0;i0;p0;a0;h0;r0],stepsize);
figure
hold on
plot(time,totalill,'green-','LineWidth',2.5)
xlabel({'Time','(in days)'})
ylabel('Confirmed cases per day')
plot(ts1(1,1:end),ys1(3,:)+ys1(4,:)+ys1(6,:),'black','linewidth',2);
hold off
tau = 9;
aux2 = size(ys1);
aux3 = size(ts1);
sizetimes = aux3(2);
totaltime = tend-t0;
for k = 1:aux2(2)-tau*(sizetimes-1)/totaltime
shifted(k) = delta.*(ys1(3,k+tau)+ys1(4,k+tau)+ys1(6,k+tau));
end
newtime = tau:totaltime/(sizetimes-1):tend;
figure
hold on
plot(totaldead,'red-','LineWidth',2.5)
xlabel({'Time','(in days)'})
ylabel('Confirmed deads per day')
plot(newtime(:),shifted(:),'black','linewidth',2);
hold off
```

Funding

This research was funded by the Portuguese Foundation for Science and Technology (FCT) within project UIDB/04106/2020 (CIDMA). Ndaïrou is also grateful to the support of FCT through the Ph.D. fellowship PD/BD/150273/2019. The work of Area and Nieto has been partially supported by the Agencia Estatal de Investigación (AEI) of Spain, cofinanced by the European Fund for Regional Development (FEDER) corresponding to the 2014–2020 multiyear financial framework, project MTM2016-75140-P. Moreover, Nieto also thanks partial financial support by Xunta de Galicia under grant ED431C 2019/02.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

Typos found in [1] were first communicated by Georg Bader and Andreas Kleefeld. We are very grateful to both of them: to Georg for having accepted to join us in this corrigendum, as an author; and to Andreas, who kindly also accepted to review and validate this corrigendum.

References

[1] Ndaïrou F, Area I, Nieto JJ, Torres DFM. Mathematical modeling of COVID-19 transmission dynamics with a case study of Wuhan. Chaos Solitons Fractals 2020;135(109846):6.

[2] Garrappa R. Predictor-corrector PECE method for fractional differential equations MATLAB central file exchange. 2020. https://www.mathworks.com/matlabcentral/fileexchange/32918-predictor-corrector-pece-method-for-fractional-differential-equations.

Faïçal Ndaïrou
Center for Research and Development in Mathematics and Applications (CIDMA), Department of Mathematics, University of Aveiro, Aveiro 3810-193, Portugal

Iván Area
Departamento de Matemática Aplicada II, E. E. Aeronáutica e do Espazo, Campus de Ourense, Universidade de Vigo, Ourense 32004, Spain

Georg Bader
Institute for Applied Mathematics and Scientific Computing, Brandenburgische Technische Universtität Cottbus, Konrad-Wachsmann-Allee 1, Cottbus D-03046, Germany

Juan J. Nieto
Instituto de Matematicas, Universidade de Santiago de Compostela, Santiago de Compostela 15782, Spain

Delfim F.M. Torres*
Center for Research and Development in Mathematics and Applications (CIDMA), Department of Mathematics, University of Aveiro, Aveiro 3810-193, Portugal

*Corresponding author.
E-mail addresses: faical@ua.pt (F. Ndaïrou), area@uvigo.es (I. Area), georg.bader@b-tu.de (G. Bader), juanjose.nieto.roig@usc.es (J.J. Nieto), delfim@ua.pt (D.F.M. Torres)