# A C++ code for predicting COVID-19 cases by least-squares fitting of the Logistic model

**Preprint** · March 2020

DOI: 10.13140/RG.2.2.28264.01283

Ademir Xavier
Brazilian Space Agency
**68** PUBLICATIONS **278** CITATIONS

# A C++ code for predicting COVID-19 cases by least-squares fitting of the Logistic model

Ademir Xavier Jr.[*]

Brazilian Space Agency, Brasilia/DF, Brazil.

March, 28th 2020

**Abstract**

This report describes a C++ code to fit the Logistic model to the daily total cases reported by data sources of the COVID-19 outbreak using least-squares. Some examples for China, Brazil, India and Italy are tested numerically and the code listing is given. The method can be used to "track" the daily evolution of new cases and to make predictions about the time of maximum daily cases as well as the total number of cases at the end of the pandemic.

Keywords: covid-19, epidemic, least-squares, logistic model, pandemic, steepest descent.

## 1 Introduction

Given the outbreak of the Covid-19 pandemic in the beginning of 2020, a search for numerical models to forecast the epidemic evolution are under way. A "first principles" approach would try to model all possible variables responsible for the final number of cases and deaths in a given region. Given the observed contamination rate, the use of quick models [1, 2, 3] entirely based on data seems indicated, while more sophisticated or complex treatments are developed. More than that, such empirical models can result in

[*]E-mail: ademir.junior@aeb.gov.br

important parameter descriptors that can help to characterize the epidemic in a given region or country and compare it to other regions.

However, these models should be applied with care since their validity is restricted to a single outbreak at a time and closed populations. In this note I present a simple code to forecast the total number of cases by fitting past accumulated values of the total number of cases to the so-called "Logistic model" following M. Batista [2]. The approach uses as initial guess a sequence of three distinct values directly provided by data and the fitting is improved by minimizing the variance using the steepest descent method.

This work is organized as follows: in Section 1 a review of the Logistic model is presented; Section 2 describes how to minimize the variance of the model applied to a time series containing the number of accumulated cases; Section 4 presents some examples and numerical tests (for Brazil, China, India and Italy), Section 5 brings a succinct description of the code and an example of the function `main.cpp` to run the code. Section 7 contains the code listing (the declaration `covid.h` and the functions in `functions.cpp`). The conclusion is presented in Section 6.

## 2 Applying the logistic model to the COVID-19 outbreak

Forecasts for the time evolution of a population growth (including viral contamination) is approximated by the well-known logistic model [3]

$$\frac{dN}{dt} = rN\left(1 - \frac{N}{K}\right), \tag{1}$$

with $N$ the total number of cases at a given time $t$, $r$ the infection growth constant and $K$ another constant representing the total number of cases in the final state of the outbreak. When $N$ is small, the growth of the number of cases is approximately proportional to $rN$. As $N \to K$, the growth rate goes to zero and the epidemic stops. Such equation has the well-known solution

$$N(t) = \frac{K}{1 + Ae^{-rt}}. \tag{2}$$

with $A$ a new parameter associated with the initial population size. A "time constant" $\tau$ can be inferred from this model as $\tau = 1/r$. Given such time evolution, the rate of contamination is simply

$$\frac{dN}{dt} = \frac{rAKe^{-rt}}{(1 + Ae^{-rt})^2}. \tag{3}$$

For small $t$, the Logistic model behaves as an exponential growth

$$N = \frac{Ke^{rt}}{A + e^{rt}} \approx \frac{K}{A}e^{rt}, \tag{4}$$

from which we extract the relation of $A$ and $K$ with the initial population size $N(0)$ in the epidemic initial phase $\approx N(0)e^{rt}$. On the other hand, for large $t$ it is easy to see that $N(t) \to K$, which represents the final number of cases. The limitation of the model is fairly obvious. First, it admits a homogeneous "mixing" among individuals in a "closed population". If the system is not closed (as in the real case), the existence of susceptible individuals in the original group in contact with newly infected cases might give rise to a new outbreak. During a tight "lockdown", the model is supposed to work within a given accuracy which tends to worsen if the social distancing is called off eventually and new cases are observed.

The "maximum epidemic time" can be estimated as the time when the maximal daily rate is achieved. Given that

$$\frac{d^2N}{dt^2} = \frac{r^2AKe^{-rt}}{(1 + Ae^{-rt})^2}\left[\frac{2Ae^{-rt}}{(1 + Ae^{-rt})} - 1\right], \tag{5}$$

the condition $d^2N/dt^2 = 0$ leads to

$$\bar{t} = \frac{\ln A}{r}, \tag{6}$$

as the maximum epidemic time $\bar{t}$. Substituting this value into Eq. (2) we find

$$N(\bar{t}) = \frac{K}{2}. \tag{7}$$

The maximum epidemic time is the inflection point of the total of accumulated cases. After this time, the epidemic evolution cannot be approximated by a simple exponential growth curve and it is meaningless to speak about a "doubling time" for the time evolution. As time goes by in this phase, the accumulated number of cases slows down.

## 3 Finding the parameters for a data set

We assume that $n$ daily accumulated case counts $M_i$ (with $n \leq 3$) is given as a series $\{d_i, M_i\}$ with $d_i$ the day index (an integer here represented by $i$, see Table1). We search for values of $(K^\star, A^\star, r^\star)$ for which the function

$$s^2 = \frac{1}{n}\sum_{i=1}^{n}\left[M_i - N(t_i)\right]^2 \tag{8}$$

achieves a minimum. The search should return the set of values $\vec{p^\star} = (K_c, A_c, r_c)$ for which $s^2(\vec{p})$ is a minimum. Define a parameter $0 \leq \alpha \leq 1$ and a vector of steps $\delta\vec{p} = (\delta K, \delta A, \delta r)$ with $|\delta p| \ll 1$. Given the parameter value at an iteration step $j$, the new parameter vector $\vec{p_l}(j+1)$ is found by using the steepest-descent rule [4]:

$$p_l(j+1) = p_l(j) - \alpha\frac{\partial s^2}{\partial p_l}\delta p_l(j), \tag{9}$$

with $l$ running indices over $k$, $A$ or $r$, and where the partial derivative is evaluated at the point $p_l(j)$. The value of $\alpha$ controls the convergence speed.

The derivatives guiding the descent for each parameter are given by

$$\begin{aligned}
\frac{\partial s^2}{\partial K} &= -\frac{2}{n}\sum_{i=1}^{n}\left[M_i - N(t_i)\right]\frac{\partial N_i}{\partial K},\\
\frac{\partial s^2}{\partial A} &= -\frac{2}{n}\sum_{i=1}^{n}\left[M_i - N(t_i)\right]\frac{\partial N_i}{\partial A},\\
\frac{\partial s^2}{\partial r} &= -\frac{2}{n}\sum_{i=1}^{n}\left[M_i - N(t_i)\right]\frac{\partial N_i}{\partial r}.
\end{aligned} \tag{10}$$

with

$$\begin{aligned}
\frac{\partial N}{\partial K} &= \frac{1}{1+Ae^{-rt}},\\
\frac{\partial N}{\partial A} &= -\frac{Ke^{-rt}}{(1+Ae^{-rt})^2},\\
\frac{\partial N}{\partial r} &= \frac{rKAe^{-rt}}{(1+Ae^{-rt})^2}.
\end{aligned} \tag{11}$$

with $N_i = N(t_i)$.

In order to apply the method an initial guess $(K(0), A(0), r(0))$ is necessary. According to [2], suitable initial values for these parameters can be obtained by choosing three initial data points $(M_{k-2m}, M_{k-m}, M_k)$ with $k \approx n$

and $m \approx n/2$. They correspond to three widely separated case counts so that $k-2m \approx 1$ or the first day in the series. Then, it is possible to show [2] that

$$
\begin{aligned}
K(0) &\approx \frac{M_{k-m}(M_{k-2m}M_{k-m} - 2M_{k-2m}M_k + M_{k-m}M_k)}{M_{k-m}^2 - M_k M_{k-2m}}, \\
A(0) &\approx \frac{(M_k - M_{k-m})(M_{k-m} - M_{k-2m})}{M_{k-m}^2 - M_k M_{k-2m}} \left[\frac{M_k(M_{k-m} - M_{k-2m})}{M_{k-2m}(M_k - M_{k-m})}\right]^{(k-m)/m} \\
r(0) &\approx \frac{1}{m} \ln \left[\frac{M_k(M_{k-m} - M_{k-2m})}{M_{k-2m}(M_k - M_{k-m})}\right],
\end{aligned}
\tag{12}
$$

The criteria of acceptance of this guess demand that all parameters be positive and that $K > M_k$. Sometimes the initial guess is so good that no refined values are necessary. In order to minimize the variance given by Eq. (8), the steepest descent procedure is applied as a way to minimize the variance Eq. (8).

## 4 Finding solutions

The code was tested initially for Brazil using updated data by the Brazilian Ministry of Health available at Wikipedia [5]. The data was adjusted so as to start from February 27th. The forecast was applied successively and new daily forecasts were generated for a short advanced period (usually no longer than 3 days) after the last available date. On March 26th ($n = 28$), the best guess was obtained for $k = 11$ and $m = 27$ as shown in Table 1. The result is depicted in Figure 1. The left part of this figure contains the data (black) and the fitted model for the total number of official cases. On the right, the output for the daily number of cases or $dN/dt$ (red curve) and $M_k - M_{k-1}$ (bar plot) for $k$ is shown. The initial input for the best convergence were $\delta K$=0.1, $\delta A$=0.1 and $\delta r$=$1\times10^{-9}$ with $\alpha$=0.5 (see Table1). Convergence was attained after $\approx 470$ iterations.

The evolution for three other countries were tested using the data source available on the Humanitarian Data Exchange (HDE) website [6]. The numbers for China were generated by summing the original values for all Chinese provinces contained in the MS Excel file provided by HDE. The results are all shown in Table 1. Graphical results for China, India and Italy are shown in Figs. 2, 3 and 4.

| Case | China | Brazil | Italy | India |
|---|---|---|---|---|
| $i = 0$ | 01/22/20 | 02/27/20 | 01/31/20 | 03/03/20 |
| $n$ | 66 | 28 | 37 | 25 |
| $m$ | 30 | 11 | 14 | 12 |
| $k$ | 65 | 27 | 35 | 24 |
| $\alpha$ | 0.001 | 0.5 | 0.0005 | 0.5 |
| $dK$ | 0.1 | 0.1 | 0.1 | 0.1 |
| $dA$ | 0.001 | 0.1 | 0.001 | 0.1 |
| $dr$ | $1 \times 10^{-9}$ | $1 \times 10^{-9}$ | $1 \times 10^{-9}$ | $1 \times 10^{-9}$ |
| $K_c$ | 81367 | 4215 | 110245 | 1406 |
| $A_c$ | 43.85 | 11800 | 767.85 | 403.9 |
| $r_c$ | 0.212 | 0.365 | 0.219 | 0.26 |
| $s$ | 2033 | 52 | 1184 | 25 |

Table 1: Fitting parameters and final estimated values for the logistic model applied to China, Brazil, Italy and India. Here $i = 0$ is the initial date of the data series, $n$ is the total number of days in the data, $m$ and $k$ are two suitable days used to calculate the initial guess, $(dK, dA, dr)$ is the parameter increment vector, $\alpha$ is the acceleration parameter, and $(K_c, A_c, r_s)$ is the final solution for the Logistic model. Also $s$ is the final data dispersion as given by Eq. 8.

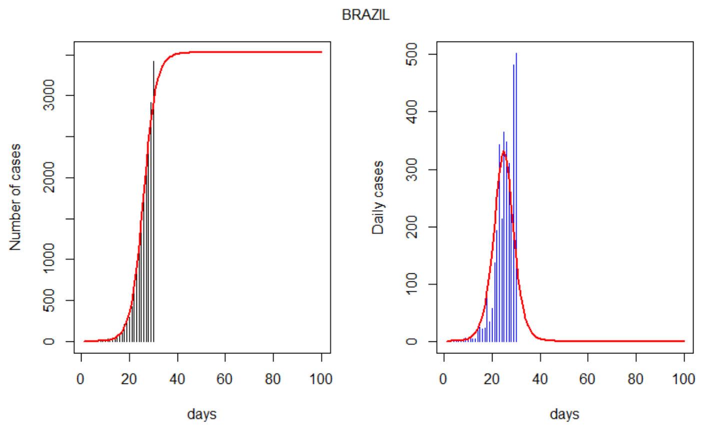

Figure 1: (Left) The number of cases (black bars) in Brazil and fitted logistic model (red) for 60 days after the initial date (February 27th). (Right) The daily number of cases (blue) and fitted logistic model for the same time interval.

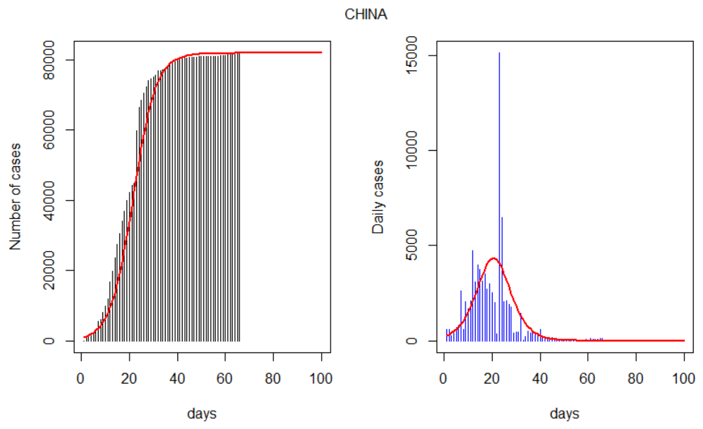

Figure 2: (Left) The number of cases (black bars) in China and fitted logistic model (red) for 100 days after the initial date. (Right) The daily number of cases (blue) and fitted logistic model for the same time interval. The model parameters are seen in Table 1.

To find a solution, the input series is chosen so that the initial dates do not contain many repeated values (as is often the case in the beginning of the outbreak maybe due to the under-reporting). In the present stage of the

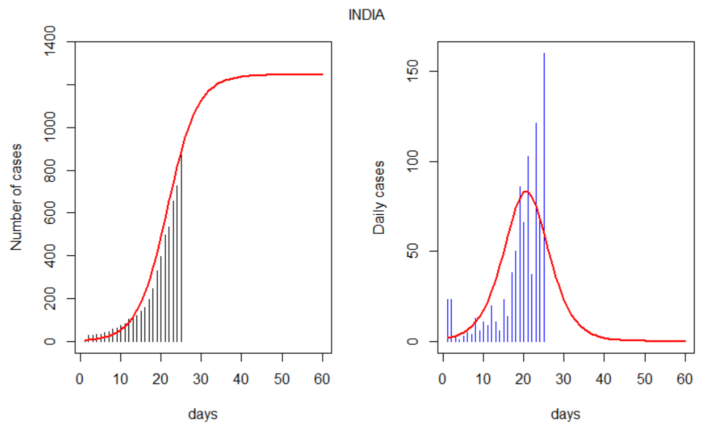

Figure 3: (Left) The number of cases (black bars) in India and fitted logistic model (red) for 60 days after the initial date. (Right) The daily number of cases (blue) and fitted logistic model for the same time interval. The model parameters are seen in Table 1.

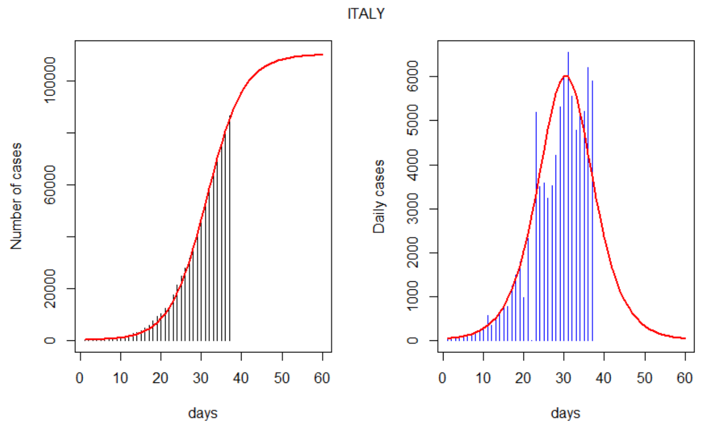

Figure 4: (Left) The number of cases (black bars) in Italy and fitted logistic model (red) for 60 days after the initial date. (Right) The daily number of cases (blue) and fitted logistic model for the same time interval. The model parameters are seen in Table 1.

presented code in Section7, $(m, k)$ are chosen manually so that the largest possible $K(0)$ is obtained through the first relation in Eqs. (12). Then, the search for the minimum of $s^2$ is performed by a suitable choice of $\alpha$ and the increment vector $(dK, dA, dr)$. For all cases treated, the increment of $r$ was observed to be $dr = 10^{-9}$. In the beginning of the outbreak, typical $K$ values are much smaller than the "converged" one after the outbreak passes the inflection point of the daily case curve (which is the case of India by the time this report was uploaded). As it is easy to see, the fitting is severely affected by under-reporting which seems to occur in the initial phase and is more evident when the daily count is compared to the forecast $dN/dt$.

## 5 The Algorithm

The code implements two classes called `LSSD` and its set of `tools` as described in the Section "Code Listing". The subroutines are declared in the file **Covid.h** and implemented in a separate file called **functions.ccp**. Input data are read in `LSSD::readinputfile(string filename)` with `filename` the name of the input file containing two columns separated by ";". The first column is simply a counter and the second one contains the total counts observed. It is possible to use the first column to enter the date in a user-specified format and write it to external output files. These outputs are written by two functions that are essentially the same code but differ in the time range of the presented results. The parameter `EndTime` controls the number of days past the last data day to calculate the forecasts.

The daily change is stored in the `vector` container `dDATA`, which has the same size of `DATA`, and whose first and second elements are made equal. All formulas given here are implemented separately in the class `tools` as described by comments in its listing of Section 7.

An example of main function to run the code is given below. In this example `covid` of the class `LSSD` is first tested for an initial guess inside an `if-then` structure. Only if a solution satisfying the criteria $K > M_k$ is found, the Steepest Descent procedure is invoked. The vectors `NcLog` and `dNcLog` are filled with the result of the optimization and are save to the external files. Examples of search parameters as given in Table 1 are copied in the list of the `main.cpp` below.

**Example of main.cpp**

```
#include "covid.h"
int main() {
LSSD covid;
```

```
string ok;
int m, k;
float alpha, dK, dA, dr;

cout << "Estimating the COVID19 evolution
using the Logistic Model" << endl;
cout << "A Xavier Jr, March 2020" << endl;
cout << "=============================================" << endl;
covid.EndTime = 60;
covid.readinputfile("brazil_data.dat");

/*
china
m=30;
k=65;
alpha = 1.e-2; //acceleration parameter
dK = 0.1;
dA = 0.001;
dr = 1e-9;
*/

//brazil
m = 12;
k = 30;
alpha = 0.5; //acceleration parameter
dK = 0.1;
dA = 0.1;
dr = 1.e-9;

if (covid.FindInitialGuess(m, k)) {

/*
Calculate the forecast for the total number of cases
and daily variation
*/
covid.Logistic();
covid.dLogistic();
covid.varN = covid.VarNcases();
cout << "Estimated VAR::" << covid.varN << endl;
cout << "Final dispersion s(0)::" << sqrt(covid.varN) << endl;

// Minimize s2
covid.steepestdescent(alpha, dK, dA, dr);
/*
Calculate new forecast for the optimized solution, total number of cases and
daily variation
*/
covid.Logistic();
covid.dLogistic();
covid.varN = covid.VarNcases();
cout << "Final VAR::" << covid.varN << endl;
cout << "Final dispersion::" << sqrt(covid.varN) << endl;

covid.writeoutput("logistic_output.csv");
covid.writeprediction("logistic_pred.csv");
}
else
cout << "No solution could be found" << endl;
cout << "It seems I've done all..." << endl;
cin >> ok;
return 0;
}
```

In the proposed main function above, the parameters of the Steepest Descent calculation are defined as variables in the function body but can be read from an external file. The number of total iterations is declared as constant in `covid.h`, but can also be entered as an external parameter. After minimizing `varN`, new values for `tp` and `Cp` are calculated. In the present implementation, since the derivatives are given by simple relations, the gradient for updating the parameters are calculated at every step but does not need to be so. The user can modify the update equations to calculate the derivatives only when the value of $s^2$ stops decreasing and a new gradient direction becomes necessary which can improve the code performance. The presented code was successfully tested using the C++ compiler of the Microsoft Visual Studio Community 2015 [7] environment, Version 14, Update

3 (2016). Other details are presented in the comments of the code listing.

## 6 Conclusions

This note presents a “hands-on” approach to forecast daily values for the total of new Covid-19 cases based on available data and under certain strong assumptions. It is important to emphasize the limits of application of the model in a scenario where the susceptible population is not closed, the number of cases is dependent on the number of available test kits at a given time, and the existence of under-reporting of cases contribute to make the available input data very noisy. It is naive to expect the model can be applied with any accuracy if more than one outbreak is observed.

The procedure seeks for an initial guess of three parameters of the Logistic model function $K$, $A$ and $r$ and minimizes an amount proportional to the error of the model applied to the data. Using daily updates, it is possible to track the behavior of the epidemic in its several phases as defined in [1, 2] as new number of cases are updated. In the first stages of the pandemic, it is extremely difficult to forecast any final $K = N(\infty)$ while the evolution follows a remarkable exponential growth (that is, it is easy to forecast new values) and the evolution is highly sensitive to the initial number of cases. As the epidemic progresses, new values of $K$ tend to increase as the model attempts to forecast the approach of a final phase. There is an error associated with each parameter $K$, $A$ and $r$, but they are not considered here. The model can also be used to determine the time of “maximum daily cases” as the inflection point of the growth curve.

## 7 Code Listing

### Covid.h file

```
#include<iostream>
#include <stdio.h>
#include <stdlib.h>
#include<string>
#include<time.h>
#include<vector>
#include <sstream>
#include <iomanip>
```

```
#include <fstream>
#define _USE_MATH_DEFINES
#include<math.h>
#include<complex>
#include <valarray>
using namespace std;

/*
Estimating the COVID-19 evolution
using the Logistic Model, A Xavier Jr, March 2020
*/

//maximum number of iterations
const int max_iter = 10000;
class tools {
;
public:
float Ncases(int jx, float Kx, float Ax, float rx);
float dNcases(int jx, float Kx, float Ax, float rx);
float K(float C1, float C2, float C3);
float A(float m, float jx, float C1, float C2, float C3);
float r(float m, float C1, float C2, float C3);
float dNdK(int tx, float Kx, float Ax, float rx);
float dNdA(int tx, float Kx, float Ax, float rx);
float dNdr(int tx, float Kx, float Ax, float rx);
};

/*
Class Least Squares Steepest Descent
DATA contains the series of input data
dDATA containg the daily number of cases
C1, C2 and C3 are the three
numbers M(k-2m), M(k-m) and M(k) for calculating
the input guess of the Logistic model
tp - time of maximum number of daily cases (Eq. 6)
Cp - number of cases at tp, equal to K/2
NcLog, dNcLog - contain the forecast of the model
for the total and the
```

```
daily number of cases
Logistic() and dLogistic() updates
NcLog and dNcLog with the model forecast
*/
class LSSD{
;
public:
int EndTime;
float C1, C2, C3, Kg, Ag, rg, kg, mg;
float tp, Cp;
float varN;
vector<float> DATA, dDATA;
vector<float> NcLog, dNcLog;
tools FF;
float VarNcases();
void Logistic();
void dLogistic();
float dRdK();
float dRdA();
float dRdr();
bool FindInitialGuess(int m, int k);
void steepestdescent(float alpha, float dKx, float dAx, float drx);
// Input and output
void readinputfile(string filenamex);
void writeoutput(string filenamex);
void writeprediction(string filenamex);
};
```

*functions.cpp file*

```
#include "COVID.h"
/*
Read input file filenamex and save into DATA[] and dDATA[]
*/
void LSSD::readinputfile(string filenamex) {
string line;
vector<string> lines;
int linecounter = 0;
string field;

ifstream myfile(filenamex); //to open the file (convert to *char)
if (myfile.is_open()) {
cout << "->>>> Open input file::" << filenamex << endl;
while (getline(myfile, line)) {
if (line.substr(0, 1) != "#") { //skip lines starting with #
lines.push_back(line); //saves valid line in lines
}
}
myfile.close();
}
for (int i = 0; i < lines.size(); i++) {
field = lines[i].substr(lines[i].find(";") + 1, lines[i].length());
DATA.push_back((float)stof(field));
}

//Find the daily difference keeping the same size of the original data
dDATA.push_back(DATA[1] - DATA[0]); //value 0 = value 1
for (int i = 1; i < lines.size(); i++) {
dDATA.push_back(DATA[i] - DATA[i - 1]);
}
}
/*
writing model and input data (including daily change)
for the data time span
*/
void LSSD::writeoutput(string filenamex) {
ofstream myfile(filenamex); //opens file
cout << "Saving output::" << filenamex << endl;
myfile << "day, Ncsimul, Ncases, dNcsimul, dNcases "<< endl;
for (int i = 0; i < DATA.size(); i++) {
myfile << i+1 << "," << NcLog[i] << "," << DATA[i] << ","
<< dNcLog[i] << "," << dDATA[i] << endl;
}
myfile.close(); //closes file
}
/*
```

```
writing model and input data (including daily change)
up to EndTime
*/
void LSSD::writeprediction(string filenamex) {
ofstream myfile(filenamex); //opens file
cout << "Saving prediction file::" << filenamex << endl;
myfile << "day, Ncsimul, dNcsimul" << endl;
for (int i = 0; i < EndTime; i++) {
myfile << i+1 << "," << NcLog[i] << "," << dNcLog[i] << endl;
}
myfile.close(); //closes file
}

/*
NcLog vector contains the Logistic model (vector of EndTime-1 elements)
*/
void LSSD::Logistic() {
for (int i = 0; i <= EndTime; i++) {
NcLog.push_back(FF.Ncases(i, Kg, Ag, rg));
}
}

/*
dNcLog vector contains the derivative of the
Logistic model (vector of EndTime-1 elements)
*/
void LSSD::dLogistic() {
for (int i = 1; i <= EndTime; i++) {
dNcLog.push_back(FF.dNcases(i, Kg, Ag, rg));
}
}

/*
Eq. (2)
*/
float tools::Ncases(int jx, float Kx, float Ax, float rx) {
return Kx / (1 + Ax*exp(-rx*jx));
}
```

```
/*
Eq. (3)
*/
float tools::dNcases(int jx, float Kx, float Ax, float rx) {
return Kx*Ax*rx*exp(-rx*jx) / ((1 + Ax*exp(-rx*jx))*(1 + Ax*exp(-rx*jx)));
}

/*
First relation in Eq. (12)
*/
float tools::K(float C1, float C2, float C3) {
return C2*(C1*C2 - 2 * C1*C3 + C2*C3) / (C2*C2 - C3*C1);
}

/*
Second relation in Eq. (12)
*/
float tools::A(float m, float jx, float C1, float C2, float C3) {
float partial, exponent;

exponent = (jx - m) / m;
partial = (C3 - C2)*(C2 - C1) / (C2*C2 - C3*C1);
partial *= pow((C3*(C2 - C1) / (C1*(C3 - C2))), exponent);
return partial;
}

/*
Third relation in Eq. (12)
*/
float tools::r(float m, float C1, float C2, float C3) {
return log(C3*(C2 - C1) / (C1*(C3 - C2))) / m;
}

/*
First relation in Eq. (11)
*/
float tools::dNdK(int tx, float Kx, float Ax, float rx) {
```

```
return 1 / (1 + Ax*exp(-rx*tx));
}

/*
Second relation in Eq. (11)
*/
float tools::dNdA(int tx, float Kx, float Ax, float rx) {
return -Kx*exp(-rx*tx) / ((1 + Ax*exp(-rx*tx))*(1 + Ax*exp(-rx*tx)));
}

/*
Third relation in Eq. (11)
*/
float tools::dNdr(int tx, float Kx, float Ax, float rx) {
return rx*Kx*Ax*exp(-rx*tx) / ((1 + Ax*exp(-rx*tx))*(1 + Ax*exp(-rx*tx)));
}

/*
To find the initial parameter before optimization
The initial guess is (Kg, Ag,, rg) using mg and kg externally given
*/
bool LSSD::FindInitialGuess(int mx, int kx) {
bool retval;
mg = (float) mx;
kg = (float) kx;
C1 = DATA[kg - 2 * mg];
C2 = DATA[kg - mg];
C3 = DATA[kg];
cout << "An initial guess for the Logistic model was found..." << endl;
//Initial guess
Kg = FF.K(C1, C2, C3);
Ag = FF.A(mg, kg, C1, C2, C3);
rg = FF.r(mg, C1, C2, C3);

//Kg has to be larger than the last data entered
if ((Ag < 0) || (Kg < DATA[DATA.size()-1]) ) {
cout << "It wasnt possible to find a solution..." << endl;
retval = false;
```

```
}
else {
cout << "K(0)=" << Kg << endl;
cout << "A(0)=" << Ag << endl;
cout << "r(0)=" << rg << endl;

//Find the time of maximum daily cases
tp = log(Ag) / rg;
Cp = Kg / 2;

cout << "Estimated time constant::" << 1 / rg << " days" << endl;
cout << "Estimated time of maximum::" << tp
<< " days after the initial day" << endl;
cout << "Estimated maximum cases at end time::" << Kg << endl;
retval = true;
}
return retval;
}

//calculating the s2 (Eq. 8) for the model and data
float LSSD::VarNcases() {
float temp=0;
float N;
for (int i = 0; i < DATA.size(); i++) {
N = FF.Ncases(i, Kg, Ag, rg);
temp += (DATA[i] - N)*(DATA[i] - N);
}
return temp / DATA.size();
}

/*
Derivative of s2 with respect to K, Eq. 10
*/
float LSSD::dRdK() {
float temp = 0;
float N, dN;
for (int i = 0; i < DATA.size(); i++) {
N = FF.Ncases(i, Kg, Ag, rg);
```

```
dN = FF.dNdK(i, Kg, Ag, rg);
temp += (DATA[i] - N)*dN;
}
return -2.*temp / DATA.size();
}

/*
Derivative of s2 with respect to A, Eq. 10
*/
float LSSD::dRdA() {
float temp = 0;
float N, dN;
for (int i = 0; i < DATA.size(); i++) {
N = FF.Ncases(i, Kg, Ag, rg);
dN = FF.dNdA(i, Kg, Ag, rg);
temp += (DATA[i] - N)*dN;
}
return -2.*temp / DATA.size();
}

/*
Derivative of s2 with respect to r, Eq. 10
*/
float LSSD::dRdr() {
float temp = 0;
float N, dN;
for (int i = 0; i < DATA.size(); i++) {
N = FF.Ncases(i, Kg, Ag, rg);
dN = FF.dNdr(i, Kg, Ag, rg);
temp += (DATA[i] - N)*dN;
}
return -2.*temp / DATA.size();
}

/*
minimize the residue s2 and find a better estimate for K, A and r
using the Steepest Descent method
```

```
The iteration parameters are given externaly by alphax, dKx, dAx and drx
*/
void LSSD::steepestdescent(float alphax, float dKx, float dAx, float drx) {
float R0,R;
float dK, dA, dr, alpha;
int iter = 0;

alpha = alphax; //acceleration parameter
dK = dKx;
dA = dAx;
dr = drx;
//Find the initial dispersion
R0 = VarNcases();

for (int j = 0; j < max_iter; j++) {
//calculating the gradients
Kg -= dRdK()*dK*alpha;
Ag -= dRdA()*dA*alpha;
rg -= dRdr()*dr*alpha;
R = VarNcases();
//if the actual dispersion is smaller than the previous
//update previous value
if (R < R0) {
R0 = R;
cout << '\r' << "iter " << j << " K=" << Kg <<
" A=" << Ag << " r=" << rg << " Residue= " << R;
iter = j;
}
else
break;
}
cout << endl << "Converged values"  << endl;
cout << "=========================" << endl;
cout << "K(" << iter <<  ")=" << Kg << endl;
cout << "A(" << iter << ")=" << Ag << endl;
cout << "r(" << iter << ")=" << rg << endl;
tp = log(Ag) / rg;
Cp = Kg / 2;
cout << "Time constant::" << 1 / rg << " days" << endl;
cout << "Final time of maximum daily cases::" << tp
<< " days after the initial day" << endl;
cout << "Final expected maximum number of cases::" << Kg << endl;
}
```

## References

[1] Batista M. (2020), *Estimation of the final size of the COVID-19 epidemic*, medRxiv, 2020.2002.2016.20023606

[2] Batista M. (2020). *Estimation of the final size of the second phase of the coronavirus COVID 19 epidemic by the logistic model.* February 2020.[ResearhGate Link]. (Last accessed: March 2020)

[3] Brauer, F., Castillo-Chavez, C., & Castillo-Chavez, C. (2012).*Mathematical models in population biology and epidemiology* (Vol. 2). New York, Springer.

[4] Bevington, P. R. & Robinson, D. K. (2003). *Data reduction and error analysis for the physical sciences.* 3rd Edition. McGraw Hill, Inc.

[5] Wikipedia (2020). *2020 coronavirus pandemic in Brazil.* Available here. (Last accessed: March 2020)

[6] Humanitarian Data Exchange (2020). https://data.humdata.org/ (Last accessed: March 2020)

[7] Ritchie, P. (2016). *Introduction to Visual Studio 2015.* In Practical Microsoft Visual Studio 2015 (pp. 1-25). Apress, Berkeley, CA.