Example: COVID-2019 data for an entire Country

Table of Contents

I am taking some data from John Hopkins university [1]

[1] https://github.com/CSSEGISandData/COVID-19

Initialisation

The parameters are here taken as constant except the death rate and the cure rate.

```
clearvars;close all;clc;
% Download the data from ref [1] and read them with the function getDataCOVID
[tableConfirmed,tableDeaths,tableRecovered,time] = getDataCOVID();
% time = time(1:end-1);
fprintf(['Most recent update: ',datestr(time(end)),'\n'])
```

```
Most recent update: 27-Jun-2020
```

```
Location = 'France';
% Version 4.8 and above have an additional table conditions (thank to Aleks Czernicki)
% For more details, see: https://github.com/ECheynet/SEIR/pull/12
try
    indR = find(contains(tableRecovered.CountryRegion,Location)==1 & (tableRecovered.Pr
    indC = find(contains(tableConfirmed.CountryRegion,Location)==1 & (tableConfirmed.Pr
    indD = find(contains(tableDeaths.CountryRegion,Location)==1 & (tableDeaths.Province
catch exception
    searchLoc = strfind(tableRecovered.CountryRegion,Location);
    indR = find(~cellfun(@isempty,searchLoc)) ;

    searchLoc = strfind(tableConfirmed.CountryRegion,Location);
    indC = find(~cellfun(@isempty,searchLoc)) ;

    searchLoc = strfind(tableDeaths.CountryRegion,Location);
    indD = find(~cellfun(@isempty,searchLoc));
end
```

```
indR = 110
indC = 118
indD = 118
```

```
disp(tableRecovered(indR,1:2))
```

ProvinceState	CountryRegion
<missing>	"France"

```
disp(tableConfirmed(indC,1:2))
```

ProvinceState	CountryRegion
<missing>	"France"

```
disp(tableDeaths(indD,1:2))
```

ProvinceState	CountryRegion
<missing>	"France"

```
indR = indR(1);
indD = indD(1);
indC = indC(1);

Recovered = table2array(tableRecovered(indR,5:end));
Deaths = table2array(tableDeaths(indD,5:end));
Confirmed = table2array(tableConfirmed(indC,5:end));
% If the number of confirmed cases is small, it is difficult to know whether
% the quarantine has been rigorously applied or not. In addition, this
% suggests that the number of infectious is much larger than the number of
% confirmed cases
minNum= round(0.25*max(Confirmed));
indRemoved = unique([find(Confirmed<=minNum),find(isnan(Confirmed))]);
Recovered(indRemoved)=[];
Deaths(indRemoved)=[];
time(indRemoved)= [];
Confirmed(indRemoved)=[];

if isempty(Confirmed)
    warning('"Confirmed" is an empty array. Check the value of "minNum". Computation ab
    return
end


Npop= 80e6; % population
```

Fitting of the generalized SEIR model to the real data

```
% Definition of the first estimates for the parameters
alpha_guess = 0.06; % protection rate
beta_guess = 1.0; % Infection rate
LT_guess = 5; % latent time in days
Q_guess = 0.1; % rate at which infectious people enter in quarantine
lambda_guess = [0.01,0.001,0]; % recovery rate
kappa_guess = [0.001,0.001,10]; % death rate

guess = [alpha_guess,...
    beta_guess,...
    1/LT_guess,...
    Q_guess,...
```

```
    lambda_guess,...
    kappa_guess];


% Initial conditions
Q0 = Confirmed(1)-Recovered(1)-Deaths(1);
I0 = 0.1*Q0; % Initial number of infectious cases. Unknown but unlikely to be zero.
E0 = 0.5*Q0; % Initial number of exposed cases. Unknown but unlikely to be zero.
R0 = Recovered(1);
D0 = Deaths(1);

Active = Confirmed-Recovered-Deaths;
Active(Active<0) = 0; % No negative number possible
[alpha1,beta1,gamma1,delta1,Lambda1,Kappa1,lambdaFun,kappaFun] = ...
    fit_SEIQRDP(Active,Recovered,Deaths,Npop,E0,I0,time,guess);
```

```
                                    Norm of      First-order
 Iteration  Func-count     f(x)          step          optimality
     0         11      3.7295e+14                      2.26e+16
     1         22     6.04875e+13      0.0298265       2.97e+15
     2         33     1.02475e+13      0.0403765       3.98e+14
     3         44      1.6144e+12       0.075889       5.41e+13
     4         55     2.01831e+11       0.103887       7.51e+12
     5         66     2.14032e+10      0.0557155       9.83e+11
     6         77     6.30412e+09      0.0944753       8.73e+10
     7         88     3.31579e+09      0.0592876       5.57e+10
     8         99     3.19691e+09     0.00255643       1.83e+10
     9        110     3.11869e+09     0.00907839       1.68e+10
    10        121     3.10057e+09    0.000514891       1.51e+10
    11        132     2.77443e+09      0.0688203       1.07e+10
    12        143     2.52768e+09       0.101353       5.54e+09
    13        154     2.43743e+09       0.100036       6.53e+09
    14        165     2.40164e+09      0.0563771       1.04e+10
    15        176     2.39521e+09     0.00031437       1.65e+09
    16        187     2.39521e+09       0.536465       1.65e+09
    17        198     2.33207e+09       0.134116       1.71e+10
    18        209     2.31009e+09    0.000594356       2.66e+09
    19        220     2.20515e+09       0.134116       8.19e+09
    20        231     2.16176e+09      0.0352855       6.91e+09
    21        242     2.15391e+09    0.000374983       4.61e+09
    22        253     2.04383e+09       0.181268       5.44e+09
    23        264     1.95515e+09       0.187721       7.52e+09
    24        275     1.87113e+09       0.238857       1.09e+10
    25        286     1.83536e+09       0.100052       3.99e+09
    26        297      1.8342e+09     0.00437334       3.89e+09
    27        308     1.83388e+09     4.9635e-05          3e+09
    28        319     1.83306e+09    0.000311786       7.09e+08
    29        330     1.79692e+09       0.268233       7.39e+09
    30        341     1.79579e+09     0.00010121       5.99e+09
    31        352     1.79106e+09    0.000729706       4.51e+09
    32        363      1.7824e+09      0.0886897       6.72e+08
    33        374     1.78032e+09      0.0817541       3.64e+08
    34        385     1.78018e+09      0.0207643       1.37e+08
    35        396     1.78015e+09      0.0128217       1.11e+08
    36        407     1.78014e+09     0.00272898       8.42e+06

Local minimum possible.

lsqcurvefit stopped because the final change in the sum of squares relative to
its initial value is less than the value of the function tolerance.
```

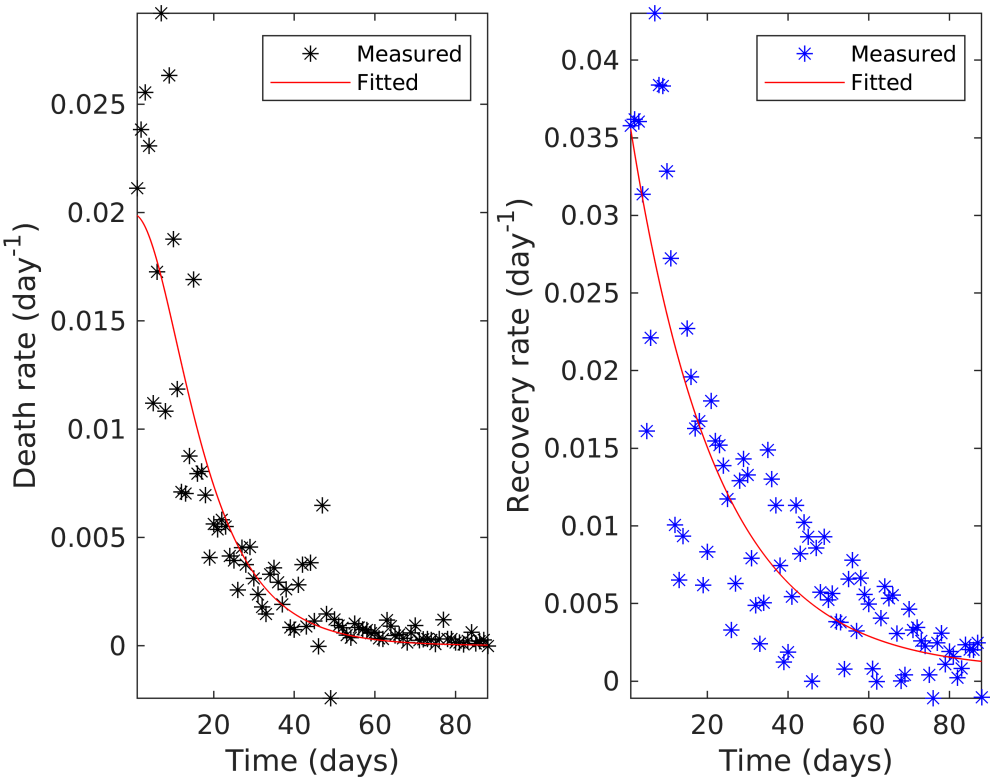
```
<stopping criteria details>
```

Simulate the epidemy outbreak based on the fitted parameters

```
dt = 1/24; % time step
time1 = datetime(time(1), 'Locale', 'en_US'):dt:datetime(datestr(floor(datenum(now)+dat
N = numel(time1);
t = [0:N-1].*dt;
[S,E,I,Q,R,D,P] = SEIQRDP(alpha1,beta1,gamma1,delta1,Lambda1,Kappa1,...
    Npop,E0,I0,Q0,R0,D0,t,lambdaFun,kappaFun);
```

Display the fitted and measured death and recovery rates

```
checkRates(time,Active,Recovered,Deaths,kappaFun,lambdaFun,Kappa1,Lambda1);
```



Comparison of the fitted and real data

Active cases = Confirmed-Deaths-Recovered (database) = Quarantined (SEIQRDP model)

```
clf;close all;
figure
semilogy(time1,Q,'r',time1,R,'b',time1,D,'k');
hold on
semilogy(time,Active,'ro',time,Recovered,'bo',time,Deaths,'ko');
% ylim([0,1.1*Npop])
```

```
ylabel('Number of cases')
xlabel('time (days)')
leg = {'Active (fitted)',...
        'Recovered (fitted)','Deceased (fitted)',...
        'Active (reported)','Recovered (reported)','Deceased  (reported)'};
legend(leg{:},'location','southoutside')
set(gcf,'color','w')
grid on
axis tight
 set(gca,'yscale','lin')
```