Received: 2 July 2023 | Revised: 17 January 2024 | Accepted: 31 January 2024

DOI: 10.1002/sim.10036

TUTORIAL IN BIOSTATISTICS

Statistics in Medicine WILEY

Parameter estimation and forecasting with quantified uncertainty for ordinary differential equation models using *QuantDiffForecast*: A MATLAB toolbox and tutorial

Gerardo Chowell | **Amanda Bleichrodt** | **Ruiyan Luo**

Department of Population Health Sciences, School of Public Health, Georgia State University, Atlanta, Georgia, USA

Correspondence
Gerardo Chowell, Department of Population Health Sciences, School of Public Health, Georgia State University, Atlanta, GA, USA.
Email: gchowell@gsu.edu

Funding information
National Science Foundation, Grant/Award Numbers: 2026797, 2125246; National Institutes of Health, Grant/Award Number: R01 GM 130900

Mathematical models based on systems of ordinary differential equations (ODEs) are frequently applied in various scientific fields to assess hypotheses, estimate key model parameters, and generate predictions about the system's state. To support their application, we present a comprehensive, easy-to-use, and flexible MATLAB toolbox, *QuantDiffForecast*, and associated tutorial to estimate parameters and generate short-term forecasts with quantified uncertainty from dynamical models based on systems of ODEs. We provide software (https://github.com/gchowell/paramEstimation_forecasting_ODEmodels/) and detailed guidance on estimating parameters and forecasting time-series trajectories that are characterized using ODEs with quantified uncertainty through a parametric bootstrapping approach. It includes functions that allow the user to infer model parameters and assess forecasting performance for different ODE models specified by the user, using different estimation methods and error structures in the data. The tutorial is intended for a diverse audience, including students training in dynamic systems, and will be broadly applicable to estimate parameters and generate forecasts from models based on ODEs. The functions included in the toolbox are illustrated using epidemic models with varying levels of complexity applied to data from the 1918 influenza pandemic in San Francisco. A tutorial video that demonstrates the functionality of the toolbox is included.

KEYWORDS
ODE model toolbox, ordinary differential equations, parameter estimation, real-time forecasting and performance, tutorial

1 | BACKGROUND

Mathematical models based on systems of ordinary differential equations (ODEs) are frequently applied in many scientific disciplines, including the biological and social sciences (eg, References [1-3]). These dynamic models, specified by one or more ODEs and their parameters, are key to investigating the collective dynamics of the system that arise in

different regions of the parameter space.[4] Estimating one or more parameters of the ODE model with quantified uncertainty from observed time-series data tracking one or more states of the system of interest (ie, the inverse problem) is a key step to calibrating a model for specific applications. Once the ODE model is calibrated to data, this can be used to test hypotheses and derive predictions about the future states of the system. Here we introduce an easy-to-use and flexible MATLAB toolbox, *QuantDiffForecast*, and associated tutorial to estimate parameters and generate short-term forecasts with quantified uncertainty from dynamical models based on ODEs.[5]

This tutorial paper introduces the user-friendly MATLAB toolbox, *QuantDiffForecast*, and tutorial for estimating parameters and forecasting time-series trajectories with quantified uncertainty using ODEs and a parametric bootstrapping approach.[5,6] This toolbox is written for a diverse audience, including graduate students training in applied mathematical and statistical sciences. The toolbox's functions allow the user to infer model parameters and assess forecasting performance for different ODE models specified by the user, using different estimation methods and error structures in the data. The functionality of the toolbox is illustrated using models with varying levels of complexity, including phenomenological growth models and mechanistic models comprising multiple differential equations, which are calibrated using simulated and actual data. A tutorial video that demonstrates the toolbox functionality is available on YouTube (https://www.youtube.com/watch?v=eyyX63H12sY).

We start by providing a theoretical overview of ODE models (Section 2), parameter estimation methodologies and model fitting (Section 3), forecasting (Section 4), and performance metrics (Section 5) available as part of the *QuantDiffForecast* toolbox. We then illustrate the application of the toolbox to specify the user parameters and functions needed to estimate parameters, display model fits, and generate and evaluate forecasts using, for illustration, time-series data of the 1918 influenza pandemic in San Francisco (Sections 6 and 7).

2 | ORDINARY DIFFERENTIAL EQUATION MODELS (ODES)

Mathematical models based on ODEs are important tools to address scientific questions that involve dynamic processes and require the estimation of parameters and predictive analysis. ODE models vary in complexity in terms of the number of variables and parameters that characterize the dynamic states of the ODE system. These dynamic models are specified by a set of equations and their parameters that together quantify the temporal states of the system via a set of interrelated dynamic quantities. Broadly, ODE models can be classified as phenomenological and mechanistic. Phenomenological ODE models provide an empirical approach to investigating patterns in the observed data. In contrast, mechanistic models aim to capture mechanisms involved in the dynamics of the problem under study to explain patterns in the observed data. An ODE model comprised of a system of h ordinary-differential equations is: given by:

$$\dot{x_1}(t) = g_1(x_1, x_2, \ldots, x_h, \Theta)$$

$$\dot{x_2}(t) = g_2(x_1, x_2, \ldots, x_h, \Theta)$$

$$\vdots$$

$$\dot{x_h}(t) = g_h(x_1, x_2, \ldots, x_h, \Theta)$$

Above, $\dot{x_i}$ denotes the rate of change of the system state x_i where $i = 1, 2, \ldots, h$ and $\Theta = (\theta_1, \theta_2, \ldots, \theta_m)$ is the set of model parameters. Let $f(t, \Theta)$ denote the expected temporal trajectory of the observed state of the system. Here, *observed state* refers to the specific state variable of the ODE system that has been observed or measured in a study or experiment. On the other hand, the latent states correspond to the ODE states that are not directly observed but are inferred from the mathematical modeling of the observed variables. In the context of epidemics, the observed state often corresponds to the number of new cases over time.

Employing the toolbox, the user can fully specify ODE models comprised of one or more differential equations, which will be used for model simulation, parameter estimation by fitting the model to data, and using the calibrated model to conduct forecasts with quantified uncertainty. For this purpose, the user will need to write a function that specifies the ODE model consisting of the system of equations describing how the state variables change over time as well as indicating the characteristics of the model parameters (eg, names, ranges, initial guesses, whether parameters are estimated or fixed according to prior information) and the state variables (eg, names, initial conditions).

3 | MODEL CALIBRATION AND PARAMETER INFERENCE

In this toolbox, we assume that there is a single observed state. Let $y_{t_1}, y_{t_2}, \ldots, y_{t_n}$ denote the time series of the observed state of the system used to calibrate the model. Here, $t_j, j = 1, 2, \ldots, n$, are the time points for the time series data, and n is the number of observations. Let $f(t, \Theta)$ denote the expected temporal trajectory of the observed state of the system. We can estimate the set of model parameters, denoted by Θ, by fitting the model solution to the observed data via nonlinear least squares or maximum likelihood estimation.[7] This is the model calibration step that consists of searching for a match between observed and simulated model solutions via statistical inference. Detailed information regarding parameter estimation methodology, uncertainty quantification, and model assessment implemented in this toolbox can be found in the following sections.

3.1 | Parameter estimation

The toolbox provides two options of estimation methods, nonlinear least squares (NLS) or maximum likelihood estimation (MLE), which can be specified by setting the `<method1>` user parameter to the appropriate value within the `options_fit.m` and `options_forecast.m` files. The structure of both files, along with available parameter options for `<method1>` can be found in Supplementary file S1.

3.1.1 | Nonlinear least square estimation (NLS)

Nonlinear least squares estimation (`<method1>=0`) is achieved by searching for the set of parameters $\widehat{\Theta}$that minimizes the sum of squared differences between the observed data $y_{t_1}, y_{t_2}, \ldots, y_{t_n}$and the best fit of the model (model mean) which corresponds to $f(t, \Theta)$. That is, $\Theta$is estimated by $\widehat{\Theta} = \arg\min \sum_{j=1}^{n} \left(f(t_j, \Theta) - y_{t_j} \right)^2$. This parameter estimation method weights each data point equally and does not require a specific distributional assumption for y_t, except for the first moment $E[y_t] = f(t; \Theta)$. That is, the mean of the observed data at time t is equivalent to the expected count denoted by $f(t, \Theta)$at time t.[8] This method can produce asymptotically unbiased estimates under the right conditions (such as homoscedasticity),[9] and the estimated model mean, $f\left(t_j, \widehat{\Theta}\right)$, yields the best fit to the observed data y_{t_j} in terms of the squared L2 norm. But when the errors have non-constant variance (heteroscedasticity), it can lead to inefficiency in the NLS estimates, and we will use the MLE method to incorporate the error structure (Section 3.1.2).

We can solve the nonlinear least squares optimization problem using the *fmincon* function in MATLAB.[10] In this function, to specify $f(t, \Theta) = \dot{C}(t)$ given Θ, the states of the ODE system are solved numerically using the internal MATLAB function *ode15s.m*,[11] which is especially suited for stiff ODE systems[12] that frequently arise in models related to epidemic transmission due to varied timescales, rapid changes, or interventions. For instance, if the transition from exposed to infectious happens much faster than the transition from infectious to recovered during an epidemic, such as that described in Section 7, it leads to stiffness. We employ MATLAB's *MultiStart* feature[13] to specify the number of random initial guesses of the model parameters using the parameter `<numstartpoints>` in the `options.m` and `options_forecast.m` files to thoroughly search for the best-fit parameter estimates and check that the solution is unique and the parameters are identifiable.

3.1.2 | Maximum likelihood estimation (MLE)

In addition to nonlinear least squares fitting, we can also estimate model parameters via maximum likelihood estimation (MLE)[14] with specific assumptions about the error structure in the data (eg, Poisson, Negative Binomial).

For a Poisson error structure, the full log-likelihood of Poisson is given by:

$$\sum_{j=1}^{n} \left\{ y_i ln(\mu_i) - ln\left(y_{t_j}!\right) - \mu_j \right\},$$

where $\mu_j = f\left(t_j, \Theta\right)$ denotes the mean of y_{t_j} and $f(t, \Theta)$is the mean curve to be estimated from the differential equations. The Poisson error structure can be specified by setting `<method1>=1` and `<dist1>=1` in the `options.m` and `options_forecast.m` files.

To account for the possible overdispersion in the data, we also consider the negative binomial distribution, which models the number of successes y before the rth ($r > 0$) failure occurs. Its mass function is

$$f(y|r,p) = \left(\frac{r+y-1}{y}\right)p^y(1-p)^r = \frac{1}{y!}\prod_{j=0}^{y-1}(j+r)\cdot p^y(1-p)^r$$

with mean $= \mu = \frac{rp}{(1-p)}$, variance$=\sigma^2 = \frac{rp}{(1-p)^2} > \mu$, where $p \in [0,1]$ denotes the success probability in each experiment. For n observations $y_{t_1}, y_{t_2}, ..., y_{t_n}$, the full log-likelihood is

$$l(r,p) = \sum_{j=1}^{n}\left\{\left\{\sum_{i=0}^{y_{t_j}-1} ln(i+r)\right\} + y_{t_j} ln(p) + r ln(1-p) - ln\left(y_{t_j}!\right)\right\},$$

which can be expressed with μ and σ^2 by plugging in $p = 1 - \frac{\mu}{\sigma^2}$ and $r = \frac{\mu^2}{\sigma^2-\mu}$, where $\mu = f\ (t, \Theta)$ is the mean curve to be estimated from the differential equation. There are different types of variances commonly used in a negative binomial distribution. In this toolbox, we include the options that variance is linear in the mean $\sigma^2 = \mu + \alpha\mu$ (`<method1>=3`, `<dist1>=3`), quadratic in the mean $\sigma^2 = \mu + \alpha\mu^2$ (`<method1>=4`, `<dist1>=4`), and more generally $\sigma^2 = \mu + \alpha\mu^d$ with any $-\infty < d < \infty$ (`<method1>=5`, `<dist1>=5`).

3.2 | Uncertainty quantification using parametric bootstrapping

To quantify parameter uncertainty, we follow a parametric bootstrapping approach, which is a frequentist method that allows for the computation of standard errors and related statistics without closed-form formulas.[15] The bootstrapping approach generates new data sets by resampling the original data, and then parameter values are estimated from each of these new bootstrap realizations.[6,15] We generate B bootstrap samples from the best-fit model $f(t, \widehat{\Theta})$ with an assumed error structure to quantify the uncertainty of the parameter estimates and construct confidence intervals.

Typically, the error structure in the data is modeled using a probability model such as the normal, Poisson or negative binomial distributions. Using nonlinear least squares methods, in addition to a normally distributed error structure, we can also assume a Poisson or a negative binomial distribution, whereby the variance-to-mean ratio is empirically estimated from the time series. To estimate this constant ratio, we group a fixed number of observations (eg, 7 observations for daily data into a bin across time), calculate the mean and variance for each bin, and then estimate a constant variance-to-mean ratio by calculating the average of the variance-to-mean ratios over these bins. When employing nonlinear least squares methods, the desired error structure is specified using parameter `<dist1>` in the `options_fit.m` or `options_forecast.m` files. When using maximum likelihood estimation, users can estimate parameter uncertainty for Poisson and negative binomial error structures without specifying the desired error structure (`<dist1>`), as the toolbox automatically sets `<method1>=<dist1>`.

Using the best-fit model $f(t, \widehat{\Theta})$, we generate B-times replicated simulated datasets of size n, where the observation at time $t_j$is sampled from the corresponding distribution. Next, we refit the model to each of the B simulated datasets to re-estimate the parameters using the same estimation method for the bootstrap sample as for the original data. This allows us to quantify the uncertainty of the estimate using that method. The new parameter estimates for each realization are denoted by $\widehat{\Theta}_b$,where $b = 1, 2, \ldots, B$. Using the sets of re-estimated parameters $\left(\widehat{\Theta}_b\right)$,it is possible to characterize the empirical distribution of each parameter estimate, calculate the variance, and construct confidence intervals for each parameter. The resulting uncertainty around the model fit can similarly be obtained from $f\left(t, \widehat{\Theta}_1\right)$, $f\left(t, \widehat{\Theta}_2\right)$, $\ldots$, $f\left(t, \widehat{\Theta}_B\right)$. We also similarly quantify the uncertainty of composite parameters whose values depend on several existing model parameters and are often useful to gauge the behavior of the modeled system. We characterize the

uncertainty using a few hundred bootstrap realizations (~300), which the user can specify using parameter `<B>` in the options files.

3.3 | Model selection and quality of model fit

To compare the quality of model fit, we can compare the AIC_c (corrected Akaike Information Criterion) values of the models. The AIC_c is given by[16,17]:

$$\mathrm{AIC}_c = -2\log(\text{likelihood}) + 2m + \frac{2m(m+1)}{n_d - m - 1},$$

where m is the number of model parameters and n_d is the number of data points. Specifically for normal distribution, the AIC_c is:

$$\mathrm{AIC}_c = n_d \log(\mathrm{SSE}) + 2m + \frac{2m(m+1)}{n_d - m - 1},$$

where $\mathrm{SSE} = \sum_{j=1}^{n_d}\left(f\left(t_j, \widehat{\Theta}\right) - y_{t_j}\right)^2$. This metric accounts for model complexity regarding the number of model parameters and is used for model selection.

4 | MODEL-BASED FORECASTS WITH QUANTIFIED UNCERTAINTY

Based on the best-fit model $f(t, \widehat{\Theta})$,we can make h ahead forecasts using the estimate.

$f(t+h, \widehat{\Theta})$. The uncertainty of the forecasted value can be obtained using the previously described parametric bootstrap method (Section 3.2). Let

$$f\left(t+h, \widehat{\Theta}_1\right), f\left(t+h, \widehat{\Theta}_2\right), \ldots, f\left(t+h, \widehat{\Theta}_B\right)$$

denote the forecasted value of the current state of the system propagated by a horizon of h time units, where $\widehat{\Theta}_b$ denotes the estimation of parameter set Θ from the b_{th} bootstrap sample. We can use these values to calculate the bootstrap variance to measure the uncertainty of the forecasts and use the 2.5% and 97.5% percentiles to construct the 95% prediction intervals (PI), with the assumed error structure.

We can set the forecasting horizon (h) using the parameter `<forecastingperiod1>` in the `options_forecast.m` file. Moreover, the parameter `<getperformance>` is a Boolean variable (`0/1`) to indicate whether the user wishes to compute the performance metrics of the forecasts when sufficient data is available to do so. The structure of the `options_forecast.m` file is described in Supplementary Text S2 (Supplementary file S1).

5 | PERFORMANCE METRICS

To assess calibration and forecasting performance, we used four performance metrics: the mean absolute error (MAE), the mean squared error (MSE), the coverage of the 95% prediction intervals, and the weighted interval score (WIS).[18] While it is possible to generate h-time units ahead forecasts of an evolving process, those forecasts looking into the future cannot be evaluated until sufficient data for the h-time units ahead has been collected.

The *mean absolute error* (MAE) is given by:

$$\mathrm{MAE} = \frac{1}{N}\sum_{h=1}^{N}\left|f\left(t_h, \widehat{\Theta}\right) - y_{t_h}\right|,$$

where t_h are the time points of the time series data,[19] and N is the length of the calibration period or forecasting period. Similarly, the *mean squared error* (MSE) is given by:

$$\text{MSE} = \frac{1}{N}\sum_{h=1}^{N}\left(f\left(t_h, \widehat{\Theta}\right) - y_{t_h}\right)^2.$$

The coverage of the 95% *prediction interval* (PI) corresponds to the fraction of data points that fall within the 95% PI, calculated as:

$$95\%\text{PI coverage} = \frac{1}{N}\sum_{t=1}^{n}\mathbf{1}\{Y_t > L_t \cap Y_t < U_t\},$$

where L_t and U_t are the lower and upper bounds of the 95% PIs, respectively, Y_t are the data and $\mathbf{1}$ is an indicator variable that equals 1 if Y_t is in the specified interval and 0 otherwise.

The *weighted interval score* (WIS)[18,20] is a proper score that provides quantiles of predictive forecast distribution by combining a set of Interval Scores (IS) for probabilistic forecasts. An IS is a simple proper score that requires only a central $(1-\alpha)\times 100\%$PI[18] and is described as:

$$\text{IS}_\alpha(F,y) = (u-l) + \frac{2}{\alpha}\times(l-y)\times\mathbf{1}(y<l) + \frac{2}{\alpha}\times(y-u)\times\mathbf{1}(y>u).$$

In this equation, $\mathbf{1}$ refers to the indicator function, meaning that $\mathbf{1}(y<l)=1$ if $y<l$ and 0 otherwise. The terms l and u represent the $\frac{\alpha}{2}$ and $1-\frac{\alpha}{2}$ quantiles of the forecast F. The IS consists of three distinct quantities:

The sharpness of F, given by the width $u-l$ of the central $(1-\alpha)\times 100\%$ PI.

1. A penalty term $\frac{2}{\alpha}\times(l-y)\times\mathbf{1}(y<l)$ for the observations that fall below the lower end point l of the $(1-\alpha)\times 100\%$PI. This penalty term is directly proportional to the distance between y and the lower end l of the PI. The strength of the penalty depends on the level α.
2. An analogous penalty term $\frac{2}{\alpha}\times(y-u)\times\mathbf{1}(y>u)$ for the observations falling above the upper limit u of the PI.

To provide more detailed and accurate information on the entire predictive distribution, we report several central PIs at different levels $(1-\alpha_1)<(1-\alpha_2)<\ldots<(1-\alpha_K)$ along with the predictive median, $\tilde{y}$, which can be seen as a central prediction interval at level $1-\alpha_0 \longrightarrow 0$. This is referred to as the WIS, and it can be evaluated as follows:

$$\text{WIS}_{\alpha_{0:K}}(F,y) = \frac{1}{K+\frac{1}{2}}.\left(w_0.|y-\tilde{y}| + \sum_{k=1}^{K} w_k.\text{IS}_{\alpha_k}(F,y)\right),$$

where $w_k = \frac{\alpha_k}{2}$ for $k=1,2,\ldots.K$ and $w_0=\frac{1}{2}$. Hence, WIS can be interpreted as a measure of how close the entire distribution is to the observation in units on the scale of the observed data.[21,22]

6 | OVERVIEW OF THE *QUANTDIFFFORECAST* TOOLBOX

Table 1 lists the names of user functions associated with the toolbox, along with a brief description of their role. The internal functions associated with the toolbox are given in Table S1 (Supplementary file S1). As described below, the user can specify the parameters related to model fitting and forecasting in the default user input files (`options_fit.m` and `options_forecast.m`). However, in this toolbox, the user can also pass specific input file names in the function call instead of using the default `options_fit.m` and `options_forecast.m` files to quickly apply input parameter specifications as illustrated below. Section 7 will provide further details regarding the specific applications, and illustrations of associated MATLAB scripts for the available features of the toolbox.

TABLE 1 Description of the user functions associated with the toolbox.

Function	Role
`options_fit.m`	Specifies the parameters related to model fitting, including the time series data characteristics, the model, parameter estimation method, error structure, and calibration period. The structure of the `options_fit.m` file is given in Supplementary Text S1 (Supplementary file S1).
`options_forecast.m`	Specifies the parameters related to model forecasting, including the forecasting period, the calibration period, the characteristics of the time series data, the model, parameter estimation method, and the error structure. The structure of the `options_forecast.m` file is given in Supplementary Text S2 (Supplementary file S1).
`plotODEModel.m`	Plots model solutions based on the ODE model, parameter ranges and initial conditions provided by the user in the `options_fit.m` file.
`Run_Fit_ODEModel.m`	Fits the ODE model specified by the user to data with quantified uncertainty.
`Run_Forecasting_ODEModel.m`	Fits the ODE model specified by the user to data with quantified uncertainty and generates a model-based forecast with quantified uncertainty.
`plotFit_ODEModel.m`	Display the ODE model fit and the empirical distribution of the parameters. It also saves output `.csv` files in the output folder with the model fit, the parameter estimates, Monte Carlo standard errors, 95% CIs, and the calibration performance metrics.
`plotForecast_ODEModel.m`	Display the model-based forecast and the performance metrics of the forecast. Moreover, the data associated with the forecasts, the parameter estimates, and the calibration and forecasting performance metrics are saved as `.csv` files in the output folder.

The workflow described in this tutorial is summarized in Figure 1. It is composed of six main sections: (1) specifying the ODE model, (2) model and code testing, (3) fitting the model to data through statistical inference, (4) plotting the resulting model fits and calibration performance metrics, (5) generating short-term forecasts with quantified uncertainty, and (6) plotting the resulting short-term forecasts and the associated performance metrics.

7 | THE 1918 INFLUENZA PANDEMIC IN SAN FRANCISCO: A HANDS-ON EXAMPLE

For this tutorial, we illustrate the application of the toolbox using growth and SEIR (susceptible-exposed-infectious-removed) models of the transmission dynamics and control of the spread of infectious diseases in the context of the 1918 influenza pandemic in San Francisco. We provide step-by-step instructions for both general application and tutorial-specific application of the toolbox's functions, along with providing brief descriptions of the data and SEIR model used in the tutorial.

7.1 | Installation and setup

1. Download the MATLAB code located in the folder ***forecasting_odemodels code*** from the GitHub repository: https://github.com/gchowell/paramEstimation_forecasting_ODEmodels/.
2. Create 'input' folder in your working directory where your input data will be stored.
3. Create 'output' folder in your working directory where the output files will be stored.
4. Open a MATLAB session.

7.2 | The input dataset

For this toolbox, the time-series data will be stored in the 'input' folder and needs to be a text file with the extension *.txt. The first column should correspond to the time index: 0, 1, 2, 3, … , and the second corresponds to the temporal incidence

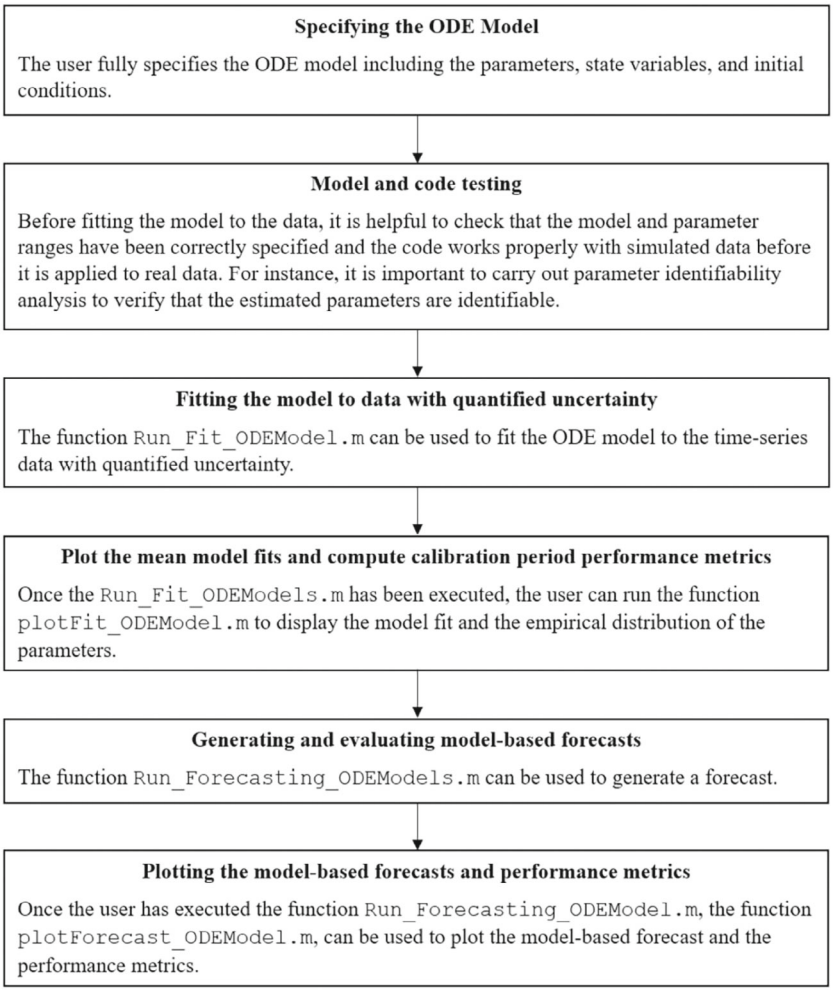


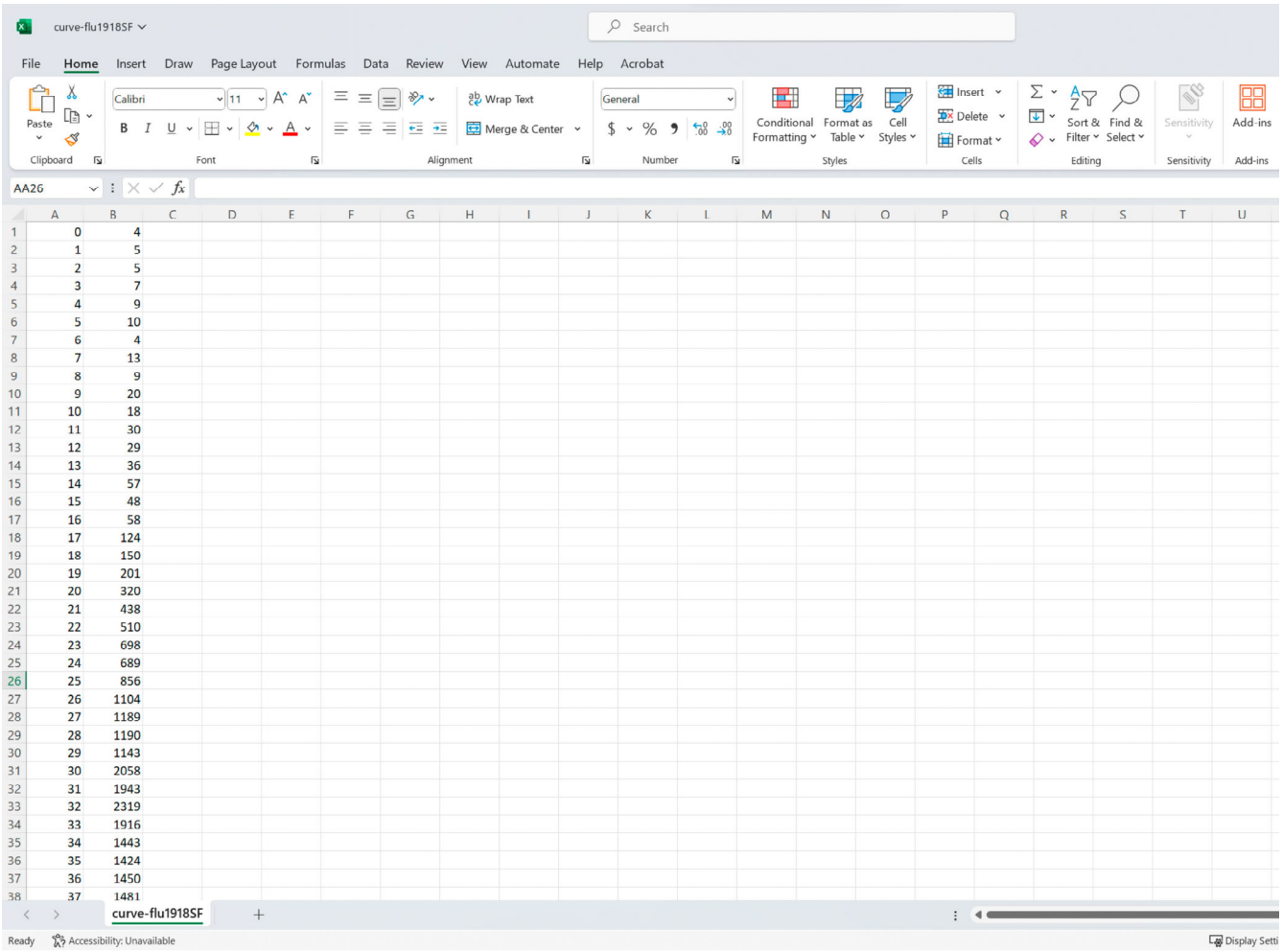
FIGURE 1 Overview of the workflow for parameter estimation and forecasting from dynamical models based on systems of ordinary differential equations.

data. If the time series file contains cumulative incidence count data, the name of the time series data file must start with "cumulative". Otherwise, there are no formal file naming conventions that must be followed.

For the purpose of the tutorial, we utilize the daily incident curve of the fall wave of the 1918 influenza pandemic in San Francisco.[23] The data file is located in the input folder within the working directory (file path: `./input/curve-flu1918SF.txt`). A snapshot in Excel of the contents of the file is shown below (Figure 2):

In the `options_fit.m` and `options_forecast.m` files, the user specifies the parameters related to model fitting and forecasting, respectively. For instance, parameter `<cadfilename1>` is a string used to indicate the name of the data file, parameter `<caddisease>` is a string used to indicate the name of the disease related to the time series data, whereas `<datatype>` is a string parameter indicating the nature of the data (eg, cases, deaths, hospitalizations). Below includes the code script for specifying the data set properties associated with the tutorial data.

```
<============================================================>
% <====================== Datasets properties  ==================>
% <============================================================>
% Located in the input folder, the time series data file is a text file with
extension *.txt.
% The time series data file contains the incidence curve of the epidemic of
interest.
% The first column corresponds to time index: 0,1,2, ... and the second
```



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	0	4																			
2	1	5																			
3	2	5																			
4	3	7																			
5	4	9																			
6	5	10																			
7	6	4																			
8	7	13																			
9	8	9																			
10	9	20																			
11	10	18																			
12	11	30																			
13	12	29																			
14	13	36																			
15	14	57																			
16	15	48																			
17	16	58																			
18	17	124																			
19	18	150																			
20	19	201																			
21	20	320																			
22	21	438																			
23	22	510																			
24	23	698																			
25	24	689																			
26	25	856																			
27	26	1104																			
28	27	1189																			
29	28	1190																			
30	29	1143																			
31	30	2058																			
32	31	1943																			
33	32	2319																			
34	33	1916																			
35	34	1443																			
36	35	1424																			
37	36	1450																			
38	37	1481																			

FIGURE 2 A snapshot in Excel of the contents of the `curve-flu1918SF.txt` data file used in the tutorial.

```
% column corresponds to the temporal incidence data. If the time series file
contains cumulative incidence count data,
% the name of the time series data file must start with "cumulative".

cadfilename1='curve-flu1918SF'

caddisease='1918 Flu'; % string indicating the name of the disease related to the
time series data

datatype='cases'; % string indicating the nature of the data (cases, deaths, hospi-
talizations, etc)
```

7.3 | Example model: The SEIR epidemic model

The simplest and most popular mechanistic ODE compartmental model for describing the spread of an infectious agent in a well-mixed population is the well-known SEIR (susceptible-exposed-infectious-removed) model.[24] This model requires 4 parameters (transmission rate, the latent period, the average infectious period, and the population size) and four state variables that keep track of the number of susceptible, exposed, infectious, and removed individuals over time. In addition, the models often include an additional state variable to keep track of the number of new infectious individuals over time, frequently used to link the model to time-series data. This model assumes no births or natural deaths in the population.

In this model, the infection rate or force of infection is often defined as the product of three quantities: a constant transmission rate (β), the number of susceptible individuals in the population ($S(t)$), and the probability that a susceptible individual encounters an infectious individual $\left(\frac{I(t)}{N}\right)$. Here, $I(t)$ represents the number of infected individuals in the population at time t, and N is the population size. Exposed individuals (E) become infectious (I) after an average latent period given by $\frac{1}{\kappa}$. Infectious individuals become recovered (R) after an average infectious period given by $\frac{1}{\gamma}$. The model is based on a system of ODEs that keep track of the temporal progression in the number of susceptible (S), exposed (E), infectious (I), and recovered (R) individuals as follows:

$$\begin{cases} \dot{S} = -\beta S(t)\frac{I(t)}{N} \\ \dot{E} = \beta S(t)\frac{I(t)}{N} - \kappa E(t) \\ \dot{I} = \kappa E(t) - \gamma I(t) \\ \dot{R} = \gamma I(t) \\ \dot{C} = \kappa E(t) \end{cases}$$

In the above system, the auxiliary variable C(t) keeps track of the cumulative number of infectious individuals, and $\dot{C}(t)$ keeps track of the curve of new cases (incidence), which is often used to link the model to time series data. If $f(t, \Theta)$denotes the temporal trajectory of the observed state of the system, then $f(t, \Theta)$will correspond to $\dot{C}$(t) in the SEIR model above.

In a completely susceptible population, for example, $S(0) \approx N$, the number of infectious individuals grows following an exponential function during the early epidemic growth phase, for example, $I(t) \approx I_0 e^{(\beta-\gamma)t}$. Moreover, the basic reproduction number (R_0) quantifies the average number of secondary cases generated per primary case during the initial transmission phase.[2] This parameter is a function of several parameters of the epidemic model, including the epidemiological classes' transmission rates and infectious periods that contribute to new infections. Moreover, R_0 often serves as a threshold parameter for the SEIR-type compartmental models. If $R_0 > 1$ then an epidemic is expected to occur whereas values of $R_0 < 1$ cannot sustain disease transmission. For this simple SEIR model, R_0 is simply given by the product of the mean transmission rate (β) and the mean infectious period ($\frac{1}{\gamma}$) as follows: $R_0 = \frac{\beta}{\gamma}$.

7.4 | Specifying the ODE model and characteristics of parameters and states

7.4.1 | Specifying the ODE model

Using a `.m` MATLAB function file as specified in `<model.fc>` in the `options_fit.m` or `options_forecast.m` files, the user can specify the model's state variables, time, and parameters as inputs and the corresponding state derivates as outputs. Similarly, the user can give a name to the model in variable `<model.name>`. MATLAB supports numerical solvers for ODEs of the form `dx = f(t,x,params0,extra0)`. Hence, the user needs to create a model function `f` that returns a column vector of state derivatives (`dx`), given as inputs the vectors of the model states (`x`), time (`t`), model parameters (`params0`), and the optional vector `extra0` with any additional parameters (ie, data streams or static variables) that are used in the model.

Tutorial: Specifying the ODE model

We use the SEIR model as an example to illustrate the specification of an ODE model. The file `SEIR1.m` contains the following function named `SEIR1` that specifies the standard SEIR epidemiological ODE model described in the previous section:

```
function dx=SEIR1(t,x,params0,extra0)

%  params0(1) = beta, params0(2)=k,  params0(3)=gamma, params0(4)=N

dx=zeros(5,1);  % define the vector of the state derivatives

dx(1,1)= -params0(1)*x(1,1).*x(3,1)./params0(4); %S
dx(2,1)= params0(1)*x(1,1).*x(3,1)/params0(4) -params0(2)*x(2,1); %E
```

```
dx(3,1)= params0(2)*x(2,1) - params0(3)*x(3,1); %I
dx(4,1)= params0(3)*x(3,1); %R
dx(5,1)= params0(2)*x(2,1); %cumulative infections
```

Five state variables are specified in the above SEIR model. Specifically, state variable `x(1)` corresponds to the number of susceptible individuals (S) at time t, `x(2)` corresponds to the number of exposed individuals (E) at time t, `x(3)` corresponds to the number of infectious individuals (I) at time t, `x(4)` corresponds to the number of recovered individuals (R) at time t, and `x(5)` tracks the cumulative number of newly infectious individuals (C) at time t. Moreover, four parameters are involved in the model. Specifically, the transmission rate (β), the rate of progression from infection to infectiousness (κ), the recovery rate (γ), and the population size (N) are the model parameters specified in this order using the vector `params0`. Thus, the first element of the parameter vector specified in `params0(1)` corresponds to β, `params0(2)` corresponds to κ, `params0(3)` corresponds to γ, and `params0(4)` corresponds to N. Finally, the ODE model's function returns the 5-element vector **dx** with the derivatives of the model's state variables. Thus, `dx(1)` corresponds to $dS(t)/dt$, `dx(2)` corresponds to $dE(t)/dt$, `dx(3)` corresponds to $dI(t)/dt$, `dx(4)` corresponds to $dR(t)/dt$, and `dx(5)` corresponds to $dC(t)/dt$. In the `options_fit.m` or `options_forecast.m` files, the `SEIR1.m` function associated with the SEIR model is specified as follows:

```
model.fc=@SEIR1; % name of the model function
model.name='SEIR model';   % string indicating the name of the ODE model
```

7.4.2 | The model parameters

The user will also need to specify the characteristics of the parameters within the `options_fit.m` or `options_forecast.m` files including the names or symbols used to refer to the model parameters in `<params.label>` as a vector of string values, the parameter ranges to be used when the parameters are estimated from the data are specified in two vectors indicating their lower bound `(<params.LB>)` and upper bound values `(<params.UB>)`, the initial parameter values in vector `<params.initial>`, and whether parameters should remain fixed (`1`) to the initial values indicated in `<params.initial>` or estimated from data (`0`) using the vector `<params.fixed>` of Boolean values. The user can also specify in the Boolean variable `<params.fixI0>` whether the initial value of the observed state variable fitted to data is fixed according to the first observation in the time series data (`1`) or estimated from data along the other parameters (`0`). Finally, the user can obtain an estimate of a composite parameter, which is a function of two or more individual model parameters, by specifying the name of the function that defines the composite parameter in `<params.composite>` and the name of the composite parameter is indicated in `<params.composite_name>`. For instance, the basic reproduction number R_0 in the SEIR model defined above is a composite parameter given by the ratio of the transmission rate and the recovery rate (β/γ). Finally, the user can use `<params.extra0>` to pass any extra parameters (eg, data, static variables) needed in the model function.

Tutorial: Specifying the model parameters

As applicable to the tutorial, in the `options_fit.m` or `options_forecast.m` files, the characteristics of the parameters for the SEIR model follow:

```
params.label={'\beta','\kappa','\gamma','N'};  % list of symbols to refer to the
model parameters
params.LB=[0.01 0.01 0.01 20]; % lower bound values of the parameter estimates
based on literature
params.UB=[10 2 2 1000000]; % upper bound values of the parameter estimates based
on literature
params.initial=[0.6 1/1.9 1/4.1 100000]; % initial parameter values/guesses
params.fixed=[0 1 1 1]; % Boolean vector to indicate any parameters that should
remain fixed (1) to initial values indicated in params.initial. Otherwise the
parameter is estimated (0).
params.fixI0=1; % Boolean variable indicating if the initial value of the fitting
variable is fixed according to the first observation in the time series (1).
Otherwise, it will be estimated along with other parameters.
```

```
params.composite=@R0s;  % Estimate a composite function of the individual model
parameter estimates otherwise it is left empty.
params.composite_name='R_0';  % Name of the composite parameter
params.extra0=[]; % used to pass any extra parameters (e.g., data, static
variables) to the model function.
```

Since <params.fixed>=[0 1 1 1], the transmission rate (β) will be estimated from the time-series data. In contrast, the other three parameters (κ, γ, N) are fixed based on prior information using the initial values specified in the vector <params.initial>. Moreover, the file R0s.m defines the following function of the composite parameter R0, a function of individual parameters β and γ, and <params.extra0> is empty, so no additional data or variables are used in the model.

```
function composite_val=R0s(params0)

% beta(1), k(2), gamma(3), N(4)
Composite_val=params0(:,1)./params0(:,3);
```

7.4.3 | The ODE model state variables

The user will also need to specify the characteristics of the state variables comprising the ODE model that describes the dynamical system of interest including the state variable names specified in the vector of strings <vars.label>, a vector containing the default initial values of the state variables (ie, initial conditions) in <vars.initial>, and the index of the observed state variable in <vars.fit_index>, which is used for parameter estimation. As mentioned above, the user specifies in the Boolean variable <params.fixI0> whether the initial value of the observed state variable is fixed according to the first observation in the time series data (1) or estimated from data along the other parameters (0). Finally, the user can also specify in Boolean variable <vars.fit_diff> whether the derivative of the model's state variable, specified by <vars.fit_index>, should be fit to data.

Tutorial: The ODE model state variables

As the time-series data used for the tutorial to estimate parameters corresponds to the daily number of new influenza cases (not cumulative cases), the fitting variable that links the SEIR model with the data is $dC(t)/dt$, the derivative of state variable $C(t)$ in the SEIR model. Hence, in this case, <vars.fit_index> should be set to 5 and <vars.fit_diff> should be set to 1 since $C(t)$ corresponds to the fifth state variable in the ordered vector of state variables: $S(t), E(t), I(t), R(t), C(t)$. In the options_fit.m or options_forecast.m files, the characteristics of the state variables of the SEIR model follow:

```
vars.label={'S','E','I','R','C'}; % list of symbols to refer to the variables
included in the model
vars.initial=[params.initial(4)-5 0 5 0 5];  % vector of initial conditions for the
model variables
vars.fit_index=5; % index of the model's variable that will be fit to the observed
time series data
vars.fit_diff=1; % boolean variable to indicate if the derivative of model's
fitting variable should be fit to data.
```

7.5 | Generating preliminary model solutions

Before fitting the model to the data, it is helpful to check that the user has correctly specified the model by checking that the model's solutions for the parameter ranges specified by the user correspond to a broad range of expected solutions. For instance, if the model solutions corresponding to ranges of parameter values (specified by the user using vectors <params.LB> and <params.UB>) show unexpected patterns that are not in line with the theory behind the model, this will suggest the presence of errors in the model specification. The user must correct those before the model can be

used for further analysis. For example, in the context of epidemic trajectories, the model solutions should not include negative values or correspond to epidemic sizes that exceed the size of the population. Moreover, comparing the range of model solutions with the data that the user intends to use for estimating parameters is useful. For instance, if the range of model solutions do not cover the range of the data, it would not be possible to obtain a good fit to the data, and the user will need to adjust the parameter ranges accordingly before the user can proceed to fit the model to the data. The function `plotODEModel.m` can be used to plot model solutions where user provides the model, parameter ranges and initial conditions in the `options_fit.m` file. The resulting plot also overlays the time-series data.

7.5.1 | Tutorial: Generating preliminary model solutions

In the following MATLAB call, the user passes the specific input options file `options_fit_SEIR_flu1918_dist1_1.m`, for our example using the SEIR model and the fall wave of the 1918 influenza pandemic in San Francisco. This way, the user can quickly retrieve specific parameter specifications using different input options files.

```
≫ plotODEModel(@options_fit_SEIR_flu1918_dist1_1)
```

This function will generate a figure (Figure 3) showing B model solutions (blue lines) of the observed state variable (specified in `<vars.fit_index>`) by generating a random sample of B parameter sets from the parameter ranges and time period specified by the user in the file `options_fit_SEIR_flu1918_dist1_1.m`. The plot also shows the model solution that follows the data most closely based on the sum of squared errors (green line) and the time-series data (red circles). The right subplot zooms in a region of the left subplot in the range of the data, showing that a subset of the model trajectories follows the data closely (Figure 3). If the model comprises more than one state variable, the model solutions for all state variables are also displayed in a different figure (Figure 4).

7.6 | Fitting the models to data with quantified uncertainty

The function `Run_Fit_ODEModel.m` can be used to fit the ODE model to the time-series data with quantified uncertainty. The function uses the input parameters provided by the user in the `options_fit.m` file. The function can also

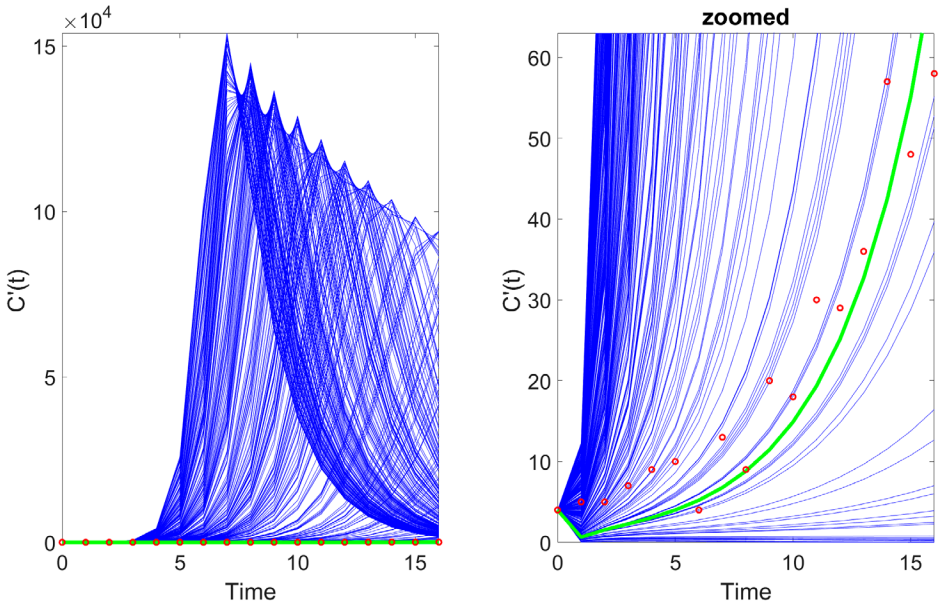


FIGURE 3 Plots of the model solutions for the SEIR model with parameter ranges and initial conditions provided by user in the `options_fit_SEIR_flu1918_dist1_1.m` file. It also shows the model solution that follows the data most closely based on the sum of squared errors (green line) and the time-series data (red circles) of the second wave of the 1918 influenza pandemic specified in the input file. The right subplot zooms in a region of the left subplot in the range of the data, showing that a subset of the model trajectories follows closely the data.

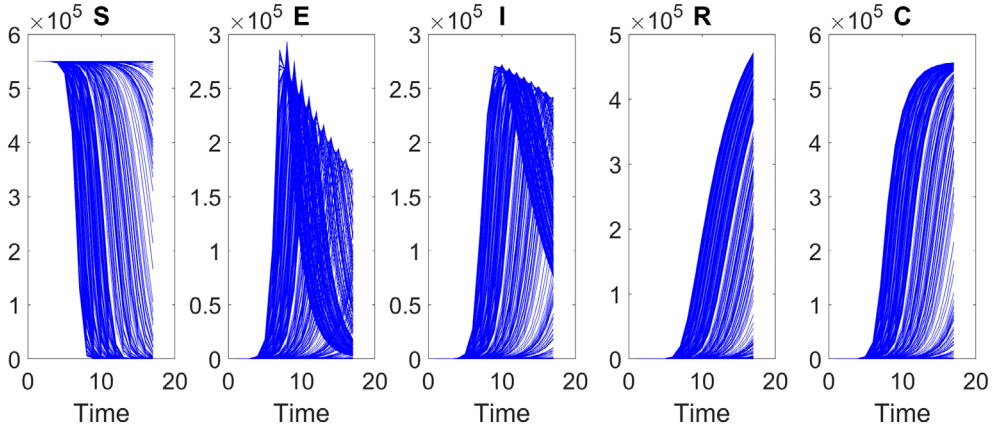


FIGURE 4 Plots of the model solutions for all state variables included in the SEIR model with parameter ranges and initial conditions provided by user in the `options_fit_SEIR_flu1918_dist1_1.m` file for the second wave of the 1918 influenza pandemic specified in the input file. This plot is outputted as part of the `plotODEModel.m` function.

receive the parameters related to the rolling window analysis (`<tstart1>`, `<tend1>`, and `<windowsize1>`) as passing input parameters with the remaining inputs accessed from the `options_fit.m` file.

7.6.1 | Rolling window analysis

A rolling window analysis can be useful to assess the stability of the model parameters and forecasts over time and requires the specification of three parameters in the input `options_fit.m` or `options_forecast.m` files: The start time (`<tstart1>`) of the first rolling window, the window size (`<windowsize1>`), and the end time (`<tend1>`) which corresponds to the start time of the last rolling window. Hence, the first rolling window contains observations for period `<start1>` to `<tstart1>` + `<windowsize1 - 1`, the second rolling window contains observations for period `<start1>` + 1 through `<windowsize1>`, and so on. Therefore, `<windowsize1>` corresponds to the length of the calibration period for each model fit. The outputs obtained from the rolling window analysis correspond to the parameter estimates and their uncertainty for each rolling window subsample. A plot of the parameter estimates over the rolling windows can help examine how the estimates change with time. The parameters can be specified in the `options_fit.m` and `options_forecast.m` files as shown below,

```
% <================================================================>
% <============== Parameters of the rolling window analysis =========>
% <================================================================>

windowsize1=17;  % moving window size

tstart1=1; % time point for the start of rolling window analysis

tend1=1; % time point for the end of the rolling window analysis
```

They can also be passed as input parameters to the fitting and forecasting functions as in the example below.

Tutorial: Fitting the models to data with quantified uncertainty

We can fit the SEIR model described above to the early phase of the fall wave of the 1918 influenza pandemic in San Francisco located in the input folder (file path: `./input/curve-flu1918SF.txt`). To that end, we use maximum likelihood estimation with a Poisson error structure (ie, `<method1>=1` and `<dist1>=1` in the `options_fit.m` file) to estimate the transmission rate (β) while fixing the latent period at 1.9 days ($\kappa = \frac{1}{1.9}$) and the infectious period at 4.1 days ($\gamma = \frac{1}{4.1}$) based on the epidemiology of influenza.[6] The corresponding input parameters are given in

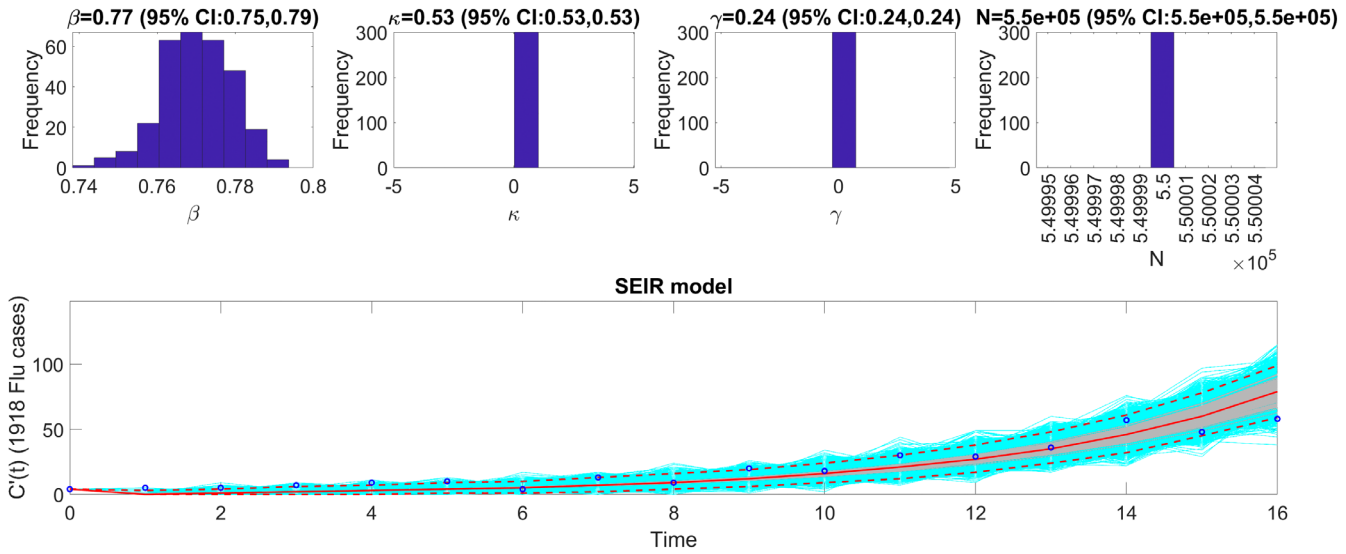


FIGURE 5 Fitting the simple SEIR model with Poisson error structure (`<method1>=1` and `<dist1>=1`) to the early phase of the fall wave of the 1918 influenza pandemic in San Francisco. The upper panel provides the empirical distributions of model parameters along with their 95% CIs that correspond with the model fit shown in the bottom panel. Overall, the model provides a good fit to data. However, the 95% prediction interval only covers about 65% of the data points. The transmission rate parameter was estimated at 0.77 (95% CI: 0.75, 0.79). The solid red line is the median model fit. The gray lines correspond to the model fits obtained from 300 bootstrap realizations. In contrast, the cyan lines indicate the predictive uncertainty around the model fit, which are used to derive the 95% prediction intervals (dashed lines), with Poisson error structure.

`options_fit_SEIR_flu1918_dist1_1.m`. Then, we can pass the specific input options file along with the rolling window parameters in the function call in MATLAB as follows:

```
≫ Run_Fit_ODEModel(@options_fit_SEIR_flu1918_dist1_1,1,1,17)
```

In the above call to the function, `<tstart1>=1`, `<tend1>=1`, and `<windowsize>=17`. Hence, this function will generate a single model fit to the first 17 days of data and store several output MATLAB files related to the model fit, parameter estimates, and the quality of model fit in the output folder. For each model fit, it will also generate a figure (Figure 5) with the model fit and the corresponding empirical distributions of the parameters along with their 95% CIs. In this case, the transmission rate parameter (β) was estimated at 0.77 (95% CI: 0.75, 0.79).

7.7 | Plotting the mean model fits and computing calibration performance metrics

Once the `Run_Fit_ODEModels.m` has been executed, the user can run the function `plotFit_ODEModel.m` to display the model fit and the empirical distribution of the parameters, including any composite parameter specified by the user using variable `<params.composite>` in the input file. If the model comprises more than one state variable, the model solutions for all state variables are also displayed in a different figure (Figure 6).

It also saves output `.csv` files in the 'output' folder with the model fit, the parameter estimates including 95% CIs, the Monte Carlo standard errors of the parameter estimates, the AIC_c values, and the calibration performance metrics.

7.7.1 | Tutorial: Plotting the mean model fits and computing calibration performance metrics

As applicable to the tutorial, the call for plotting mean model fits and computing calibration performance metrics follows:

```
≫ plotFit_ODEModel(@options_fit_SEIR_flu1918_dist1_1,1,1,17)
```

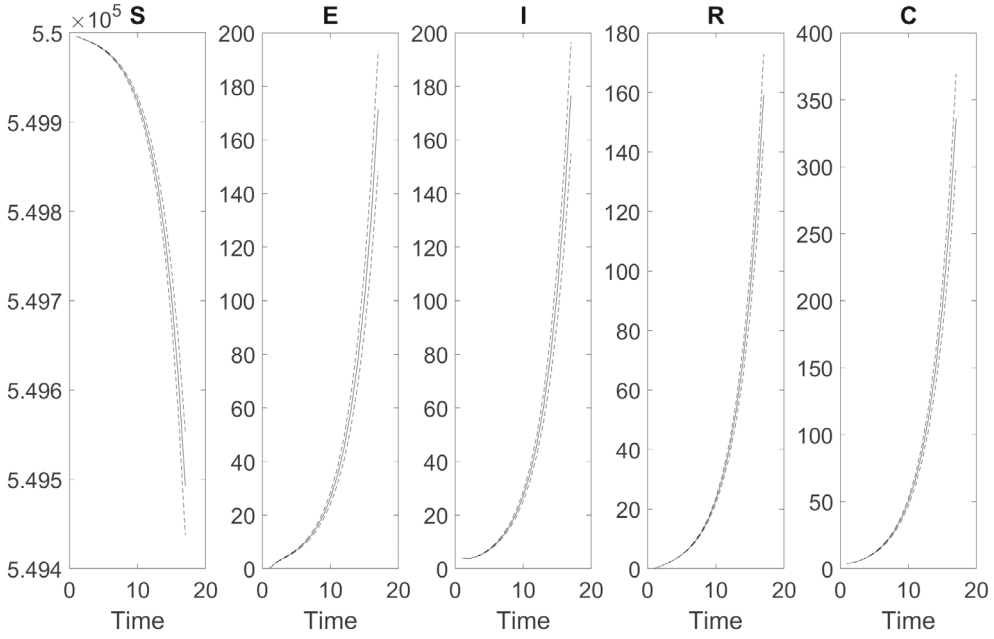


FIGURE 6 Plots of the model solutions for all state variables included in the SEIR model for the second wave of the 1918 influenza pandemic specified in the input file. This plot is outputted as part of the `plotFit_ODEModel.m` function.

This function will store the following `.csv` files in the output folder:

1. The model fit to the data:

```
Fit-model_name-SEIR model-tstart-1-fixI0-1-method-1-dist-1-tstart-1-tend-1-calibrationperiod-17-horizon-0-1918 Flu-cases.csv
```

2. Model parameter estimates:

```
parameters-rollingwindow-model_name-SEIR model-fixI0-1-method-1-dist-1-tstart-1-tend-1-calibrationperiod-17-horizon-0-1918 Flu-cases.csv
```

3. Monte Carlo standard errors:

```
MCSEs-rollingwindow-model_name-SEIR model-fixI0-1-method-1-dist-1-tstart-1-tend-1-calibrationepriod-17-horizon-0-1918-Flu-cases.csv
```

4. AIC_c values:

```
AICcs-rollingwindow-model_name-SEIR model-fixI0-1-method-1-dist-1-tstart-1-tend-1-calibrationepriod-17-horizon-0-1918-Flu-cases.csv
```

5. Calibration performance metrics:

```
performance-calibration-model_name-SEIR model-fixI0-1-method-1-dist-1-tstart-1-tend-1-calibrationperiod-17-horizon-0-1918 Flu-cases.csv
```

For this example, the model with a Poisson error structure provides a good fit to the data. However, the 95% PI only covers about 59% of the data points, suggesting that a different error structure may better capture the data. Repeating the fitting process of the SEIR model using a negative binomial error structure (`<method1>=3`, `<dist1>=3` in the `options_fit_SEIR_flu1918_dist1_3.m` file) yields an improvement as the coverage of the 95% prediction interval increases to 94.1% as shown in Figure 7 and Table 2. The distribution of R_0 (composite parameter specified in the input

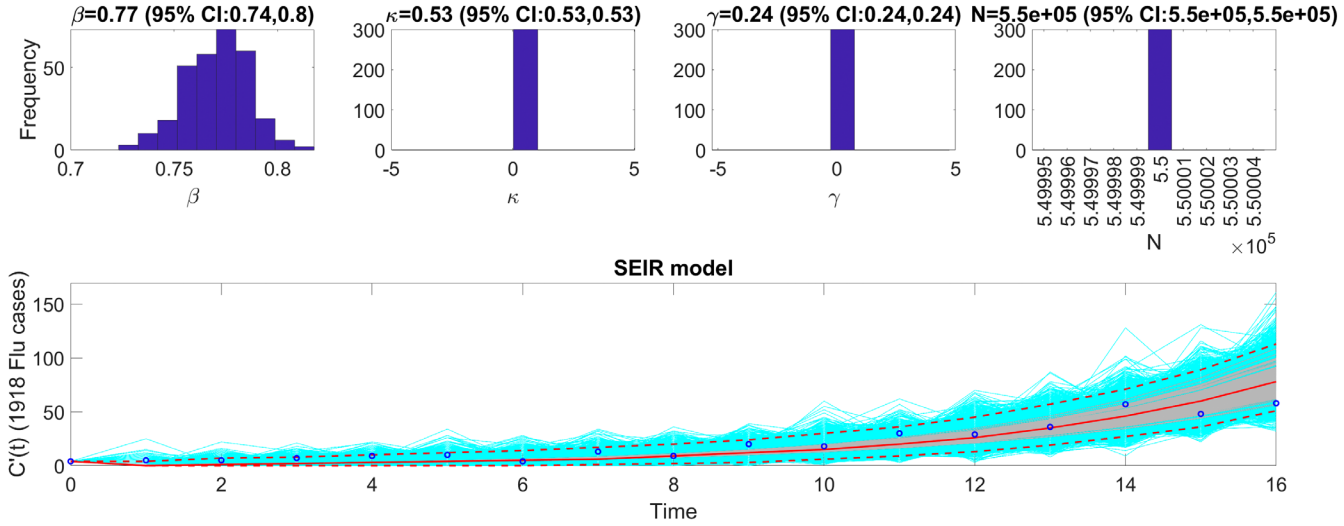


FIGURE 7 Fitting the simple SEIR model with a negative binomial error structure (`<method1>=3` and `<dist1>=3`) to the early phase of the fall wave of the 1918 influenza pandemic in San Francisco (17-day calibration period). The upper panel provides the empirical distributions of model parameters along with their 95% CIs that correspond with the model fit shown in the bottom panel. The model is well calibrated to the data with the resulting 95% prediction interval covering 94.1% of the data points. The transmission rate parameter was estimated at 0.77 (95% CI: 0.74, 0.80). The solid red line is the median model fit. The gray lines correspond to the model fits associated with the 300 bootstrap realizations. In contrast, the cyan lines indicate the predictive uncertainty around the model fit, which are used to derive the 95% prediction intervals (dashed lines).

TABLE 2 Calibration performance metrics for a 17-day calibration period quantifying how well the fits of the SEIR model with a Poisson error structure (`<dist1>=1`) and a negative binomial error structure (`<dist1>=3`) captured the early phase of the fall wave of the 1918 influenza pandemic in San Francisco. The metrics indicate that the negative binomial error structure better fits the data, especially regarding the 95% prediction interval coverage.

Model	Calibration period	MAE	MSE	Coverage 95% PI	WIS
SEIR model with Poisson error structure (`<dist1>=1`)	17	5.73	58.91	58.82	3.89
SEIR model with negative binomial error structure (`<dist1>=3`)	17	5.74	61.43	94.12	3.67

file) is also part of the output (Figure 8). It is worth noting that using the negative binomial error structure (`<dist1>=3`) requires estimating an additional parameter as explained in the parameter estimation section above.

7.8 | Generating, plotting and assessing model-based forecasts

7.8.1 | Generating model-based forecasts

To generate a forecast, we can use the function `Run_Forecasting_ODEModels.m`. This function uses the input parameters provided by the user in the input `options_forecast.m` file. However, the function can also receive `<tstart1>`, `<tend1>`, `<windowsize1>`, and `<forecastingperiod>` as passing input parameters with the remaining input parameters accessed from the `options_forecast.m` file.

Tutorial: Generating model-based forecasts

We can fit the SEIR model to the first 17 days of the fall wave of the 1918 influenza pandemic in San Francisco assuming a negative binomial error structure (ie, `<method1>=3` and `<dist1>=3` in `options_forecast.m file`) and generate a 10-day ahead prediction by running the function from MATLAB's command line as follows:

```
≫Run_Forecasting_ODEModel(@options_forecast_SEIR_flu1918_dist1_3,1,1,17,10)
```

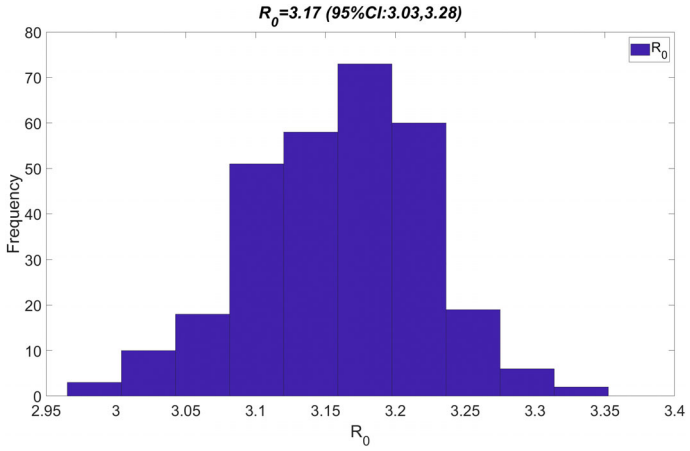


FIGURE 8 The empirical distribution of the basic reproduction number R0 (composite parameter) obtained by fitting the SEIR model to the initial 17-day growth phase of the 1918 influenza pandemic in San Francisco with a Poisson error structure (`<method1>=1` and `<dist1>=1`).

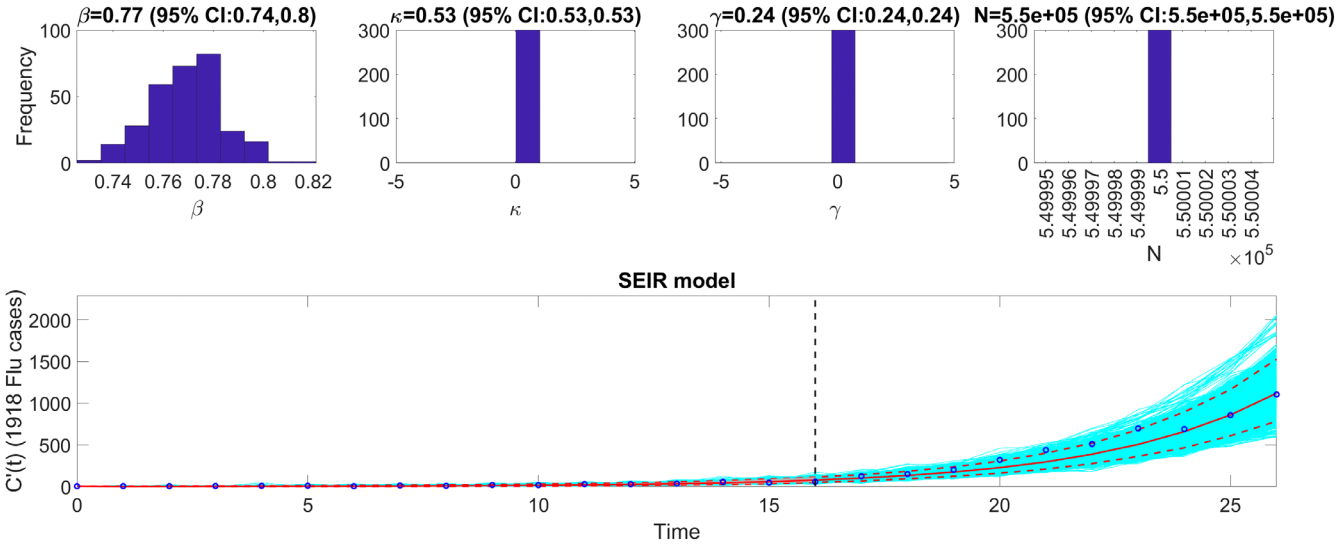


FIGURE 9 The SEIR model fit and 10-day forecast based on the first 17 days of fall wave of the 1918 influenza pandemic in San Francisco using a negative binomial error structure (`<dist1>=3`). The upper panel provides the empirical distributions of model parameters along with their 95% CIs. The solid red line is the median model fit. The blue dots indicate the data points. The gray lines, which wrap tightly around the median model fit (solid red line), correspond to the mean of the model fits obtained from the parametric bootstrapping with 300 bootstrap realizations. In contrast, the cyan lines indicate the predictive uncertainty around the model fit, which are used to derive the 95% prediction intervals (dashed lines). The vertical dashed line separates the 17-day calibration (left) and the 10-day ahead forecast (right), which performed well.

The above call passes the specific input options file `options_forecast_SEIR_flu1918_dist1_3.m` instead of using the default `options_forecast.m` file. This way the user can quickly retrieve specific parameter specifications using different input options files. The above call will generate a single model fit and forecast and store several output MATLAB files related to the model fit and forecast, parameter estimates, as well as calibration and forecasting performance metrics. It will also generate a figure (Figure 9) with the model fit and 10-day ahead forecast and the corresponding empirical distributions of the parameters. Overall, the 10-day ahead forecast shown in Figure 9 showed a good performance.

7.8.2 | Plotting and assessing model-based forecasts

Once the user has executed the function `Run_Forecasting_ODEModel.m`, the function `plotForecast_ODEModel.m` can be used to plot the model-based forecast and the performance metrics of the forecast (MSE, MAE, 95% PI, WIS) based on the inputs indicated in the input file. This function can also receive `<tstart1>`, `<tend1>`, `<windowsize1>`, and `<forecastingperiod>` as passing input parameters while the remaining inputs are retrieved from the `options_forecast.m` file. Moreover, the data associated with the forecasts, the parameter estimates, and the calibration and forecasting performance metrics, are saved as `.csv` files in the output folder.

Tutorial: Plotting and assessing model-based forecasts

For example, the following line illustrates the execution of the function from MATLAB's command window:

```
≫plotForecast_ODEModel(@options_forecast_SEIR_flu1918_dist1_3, 1,1,17,10)
```

This function plots the model fit based on a 17-day calibration period, a 10-day ahead forecast, and the empirical distribution of the estimated parameters (Figure 9). It also displays the associated forecasting performance metrics (see Figure 10). If the model comprises more than one state variable, the model solutions for all state variables are also displayed in a different figure (Figure 11).

For comparison, we can also generate the forecast using the exponential growth model (EXP) with a negative binomial error structure (`<method1>=3`, `<dist1>=3`), specified in the input file `options_forecast_EXP_flu1918_dist1_3.m`. We then compare the forecasting performance of this model with that obtained using the SEIR model using the performance metrics. Figure 12 shows the corresponding forecast obtained from the exponential growth model and the resulting empirical distribution of the model's growth rate.

The results indicate that the SEIR model is better calibrated to the data compared to the exponential growth model according to the calibration performance metrics shown in Table 3. On the other hand, the forecasting performance metrics of the exponential growth model and the SEIR model (shown in Table 3) indicate that the SEIR model yielded a better forecast in terms of the MAE and MSE metrics. In contrast, the exponential growth model yielded a better forecast in terms of the coverage of the 95% prediction interval and the WIS.

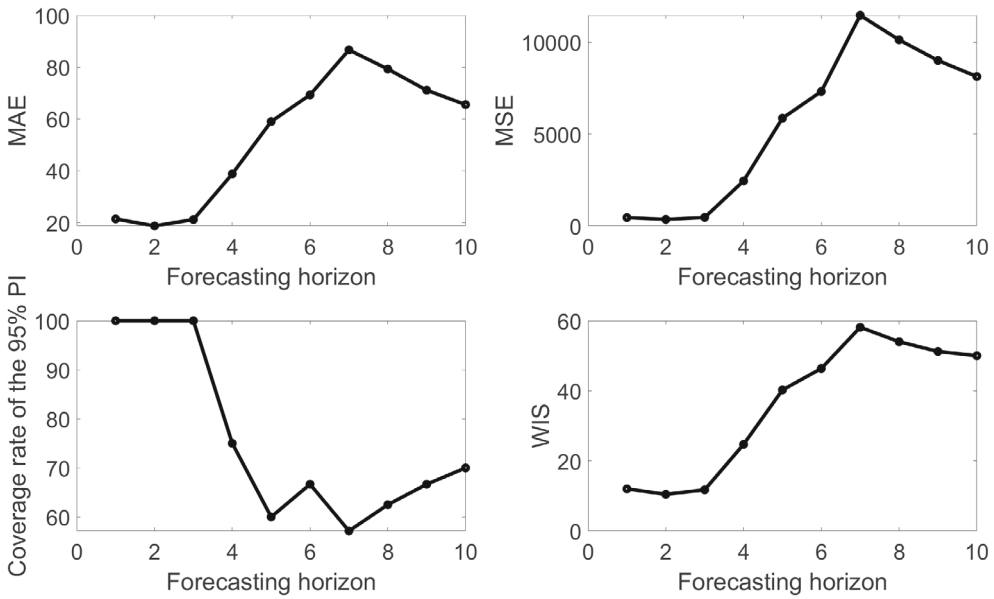


FIGURE 10 Forecasting performance metrics associated with the 10-day ahead forecast obtained from fitting the SEIR model to the initial 17 days of the fall wave of the 1918 influenza pandemic in San Francisco using the negative binomial error structure (`<dist1>=3`).

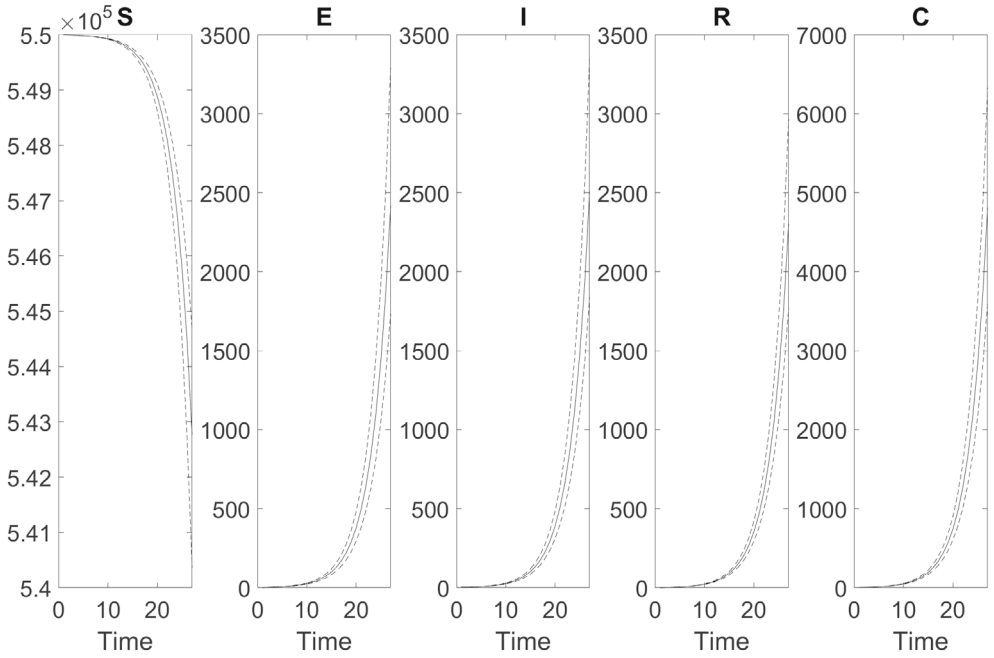


FIGURE 11 Plots of the model solutions for all state variables included in the SEIR model for the second wave of the 1918 influenza pandemic specified in the input file. This plot is outputted as part of the `plotForecast_ODEModel.m` function.

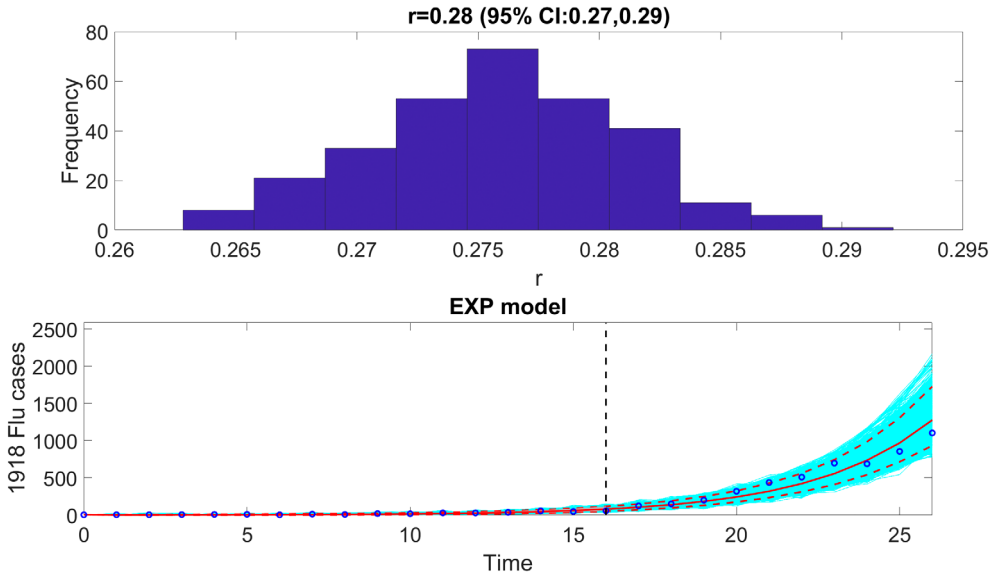


FIGURE 12 The exponential growth model fit and 10-day ahead forecast after the calibrating the model with the first 17 days of the fall wave of the 1918 influenza pandemic in San Francisco using a negative binomial error structure (`<dist1>=3`). The upper panel provides the empirical distributions of model parameters along with their 95% CIs. The solid red line is the median model fit. The blue dots indicate the data points. The gray lines, which wrap tightly around the median model fit (solid red line), correspond to the model fits obtained from the parametric bootstrapping with 300 bootstrap realizations whereas the cyan lines indicate the predictive uncertainty around the model fit and are used to derive the 95% prediction intervals (dashed lines). The vertical dashed line separates the 17-day calibration period (left) and the 10-day ahead forecast, which tracked the epidemic well.

TABLE 3 Calibration and forecasting performance metrics obtained from the fits of the SEIR model and the exponential growth model with a negative binomial error structure (ie, `<method1>=3`, `<dist1>=3`) based on the first 17 days of the fall wave of the 1918 influenza pandemic in San Francisco.

Calibration performance					
Model	**Calibration period**	**MAE**	**MSE**	**Coverage 95% PI**	**WIS**
SEIR model	17	5.74	61.43	94.12	3.67
Exponential growth	17	5.99	67.47	100.0	3.81
Forecast performance					
Model	**Forecasting period**	**MAE**	**MSE**	**Coverage 95% PI**	**WIS**
SEIR model	10	65.83	8430.51	70.0	49.77
Exponential growth	10	80.18	9384.90	90.0	46.93

Note: The SEIR model yielded a better forecast in terms of the MAE and MSE metrics. In contrast, the exponential growth model yielded a better forecast regarding the coverage of the 95% prediction interval and the WIS.

8 | DISCUSSION

In this tutorial-based primer, we have introduced a comprehensive toolbox that will be broadly applicable to fit and forecast time-series trajectories from ordinary differential equation models with quantified uncertainty using a parametric bootstrapping approach. The toolbox can be used as part of the curriculum of student training in mathematical biology, applied differential equations, infectious disease modeling, and specialty courses in epidemic modeling and time-series forecasting. We illustrate the toolbox functions using simple phenomenological and mechanistic models and different assumptions about the error structure in time series data of the 1918 influenza pandemic in San Francisco.

Our toolbox relies on bootstrapping methodology to quantify the uncertainty associated with observation error. This flexible and powerful computational method for estimating parameters and generating forecasts with quantified uncertainty continues to grow in popularity along with improved computational speed and resources. It is especially useful when it is challenging to derive theoretical formulas from complex mathematical models. Overall, our approach to parameter estimation is based on trajectory matching. In this vein, other methods include gradient matching, two-state least squares,[25,26] and profiled estimation.[27] However, we also note that Bayesian estimation methodologies offer alternative ODE estimation and forecasting approaches.[28,29]

It is worth noting some limitations and areas for future work. First, the toolbox is predominantly designed for deterministic rather than stochastic ODE systems due to the inherent differences in the nature and analysis between deterministic and stochastic systems.[30,31] Second, in our frequentist approach to parameter estimation and forecasting, we rely on established methods to search for the optimal set of parameters that yield the best fit to the time series data. Nevertheless, such optimization routines may have difficulties finding the global optimal set of parameters as model complexity increases. For this reason, the modeler may want to run the optimization algorithm using more starts of the initial parameter guesses to increase the likelihood of finding the global optimum and/or refine the search space based on domain-specific knowledge. Third, the employed parametric bootstrapping methodology quantifies the uncertainty only associated with observation error. At the same time, other factors also contribute to the uncertainty of estimates, such as the numerical error due to discretization, local minima or maxima, and even the identifiability of the ODE models.[27,32-35] Even though there have been numerous efforts to address these issues, such as MATLAB providing different ODE solvers with different levels of accuracy, the profiled method[27] or the regularized predictor-corrector algorithm to alleviate the local minima problem,[32] and the parameter identifiability studies,[33-35] we cannot distinguish these sources of variations and quantify them for arbitrarily user-specified ODEs. Furthermore, in many applications, the error due to the model is unknown, and the uncertainty due to observation error is likely much larger than that due to numerical/discretization error, and this is an ongoing research area.[36] Depending on the application, the modeler may need to conduct simulation studies to assess the influence of time discretization on ODE solutions. For example, some studies have evaluated explicit vs implicit methods,[37] adaptive time-stepping methods that adjust the time step size based on the solution behavior,[38,39] and the impact on computational complexity.[40] Fourth, the toolbox is currently intended for users with minimal programming skills. We plan to develop a web interface with intuitive navigation to enhance the toolbox's

usability and accessibility, making it more widely accessible to users with varying technical expertise. Finally, we plan to exploit parallel computing techniques in future software versions to speed up the running time.

AUTHOR CONTRIBUTIONS

G.C. conceived and developed the first version of the toolbox and wrote the first draft of the tutorial; G.C., A.B, R.L contributed to analysis and writing subsequent drafts of the tutorial. A.B. produced the tutorial video.

ACKNOWLEDGEMENTS

We acknowledge the contributions of students enrolled in the Infectious Disease Modeling Course at Georgia State University by testing the toolbox through various applied exercises.

CONFLICT OF INTEREST STATEMENT

The authors declare no conflict of interests.

DATA AVAILABILITY STATEMENT

The dataset analyzed during the current study includes the daily case notifications for the influenza pandemic in San Francisco (USA), 1918 [https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2358966/]. Please refer to the included data files for a .txt version of the influenza pandemic in San Francisco (USA), 1918 dataset, along with all MATLAB code utilized in the tutorial (Data Files). The code and .txt file can also be found at [https://github.com/gchowell/paramEstimation_forecasting_ODE].

ORCID

Gerardo Chowell https://orcid.org/0000-0003-2194-2251

REFERENCES

1. Anderson RM, May RM. *Infectious Diseases of Humans: Dynamics and Control*. Oxford university press; 1991.
2. Brauer F, Castillo-Chavez C, Castillo-Chavez C. *Mathematical Models in Population Biology and Epidemiology*. 2nd ed. New York, NY: Springer; 2012.
3. Yan P, Chowell G. *Quantitative Methods for Investigating Infectious Disease Outbreaks*. Cham, Switzerland: Springer; 2019.
4. Mondal P, Shit L, Goswami S. Study of effectiveness of time series modeling (ARIMA) in forecasting stock prices. *Int J Eng Res Appl*. 2014;4(2):13-29.
5. Chowell G. Fitting dynamic models to epidemic outbreaks with quantified uncertainty: a primer for parameter uncertainty, identifiability, and forecasts. *Infect Dis Model*. 2017;2(3):379-398. doi:10.1016/j.idm.2017.08.001. Accessed October 4, 2023.
6. Chowell G, Ammon C, Hengartner N, Hyman J. Transmission dynamics of the great influenza pandemic of 1918 in Geneva, Switzerland: assessing the effects of hypothetical interventions. *J Theor Biol*. 2006;241(2):193-204. doi:10.1016/j.jtbi.2005.11.026. Accessed October 4, 2023.
7. Banks HT, Hu S, Thompson WC. *Modeling and Inverse Problems in the Presence of Uncertainty*. Boca Raton, Fl: CRC Press; 2014.
8. Myung IJ. Tutorial on maximum likelihood estimation. *J Math Psychol*. 2003;47(1):90-100. doi:10.1016/S0022-2496(02)00028-7. Accessed October 4, 2023.
9. Wu C-F. Asymptotic theory of nonlinear least squares estimation. *Ann Stat*. 1981;9(3):501-513. doi:10.1214/aos/1176345455. Accessed October 11, 2023.
10. The MathWorks Inc. Fmincon—find minimum of constrained nonlinear multivariable function. *MathWorks*. https://www.mathworks.com/help/optim/ug/fmincon.html. Published 2006. Updated 2023. Accessed October 11, 2023.
11. The MathWorks Inc. ode15s—solve stiff differential equations and DAEs—variable order method. MathWorks. https://www.mathworks.com/help/matlab/ref/ode15s.html. Published 2006. Accessed October 11, 2023.
12. Ashino R, Nagase M, Vaillancourt R. Behind and beyond the MATLAB ODE suite. *Comput Math Appl*. 2000;40(4–5):491-512. doi:10.1016/S0898-1221(00)00175-9 Accessed October 11, 2023.
13. The MathWorks Inc. Multistart—Find Multiple Local Minima. MathWorks. https://www.mathworks.com/help/gads/multistart.html, Published 2010. Accessed October 11, 2023.
14. Roosa K, Luo R, Chowell G. Comparative assessment of parameter estimation methods in the presence of overdispersion: a simulation study. *Math Biosci Eng*. 2019;16(5):4299-4313. doi:10.3934/mbe.2019214. Accessed October 4, 2023.
15. Hastie T, Tibshirani R, Friedman J. The elements of statistical learning. *Springer Series in Statistics*. New York, NY: Springer; 2001.
16. Hurvich CM, Tsai C-L. Regression and time series model selection in small samples. *Biometrika*. 1989;76(2):297-307. doi:10.1093/biomet/76.2.297. Accessed October 4, 2023.

17. Sugiura N. Further analysis of the data by Akaike's information criterion and the finite corrections: further analysis of the data by Akaike's. *Commun Stat Theory Methods*. 1978;7(1):13-26. doi:10.1080/03610927808827599. Accessed October 4, 2023.
18. Gneiting T, Raftery AE. Strictly proper scoring rules, prediction, and estimation. *J Am Stat Assoc*. 2007;102(477):359-378. doi:10.1198/016214506000001437. Accessed October 4, 2023.
19. Kuhn M, Johnson K. *Applied Predictive Modeling*. New York, NY: Springer; 2013.
20. University of Nicosia. M4Competition Competitor's Guide: Prizes and Rules. 2018 http://www.unic.ac.cy/test/wp-content/uploads/sites/2/2018/09/M4-Competitors-Guide.pdf. Accessed October 4, 2023.
21. Bracher J, Ray EL, Gneiting T, Reich NG. Evaluating epidemic forecasts in an interval format. *PLoS Comput Biol*. 2021;17(2):e1008618. doi:10.1371/journal.pcbi.1008618. Accessed October 4, 2023.
22. Cramer EY, Ray EL, Lopez VK, et al. Evaluation of individual and ensemble probabilistic forecasts of COVID-19 mortality in the United States. *PNAS*. 2022;119(15):e2113561119. doi:10.1073/pnas.2113561119. Accessed October 4, 2023.
23. Chowell G, Nishiura H, Bettencourt LM. Comparative estimation of the reproduction number for pandemic influenza from daily case notification data. *J R Soc Interface*. 2007;4(12):155-166. doi:10.1098/rsif.2006.0161. Accessed October 4, 2023.
24. Z-l W, Wang D-f, H-Y S, Yan X. Comparison of a physical model and phenomenological model to forecast groundwater levels in a rainfall-induced deep-seated landslide. *J Hydrol (Amst)*. 2020;586:124894. doi:10.1016/j.jhydrol.2020.124894. Accessed October 4, 2023.
25. Gugushvili S, Klaassen CA. -consistent parameter estimation for systems of ordinary differential equations: bypassing numerical integration via smoothing. *Bernoulli*. 2012;18(3):1061-1098. doi:10.3150/11-BEJ362. Accessed October 4, 2023.
26. Liang H, Wu H. Parameter estimation for differential equation models using a framework of measurement error in regression models. *J Am Stat Assoc*. 2008;103(484):1570-1583. doi:10.1198/016214508000000797. Accessed October 4, 2023.
27. Ramsay JO, Hooker G, Campbell D, Cao J. Parameter estimation for differential equations: a generalized smoothing approach. *J R Stat Soc Series B Stat Methodol*. 2007;69(5):741-796. doi:10.1111/j.1467-9868.2007.00610.x. Accessed October 4, 2023.
28. Andrade J, Duggan J. A Bayesian approach to calibrate system dynamics models using Hamiltonian Monte Carlo. *Syst Dyn Rev*. 2021;37(4):283-309. doi:10.1002/sdr.1693. Accessed October 4, 2023.
29. Grinsztajn L, Semenova E, Margossian CC, Riou J. Bayesian workflow for disease transmission modeling in Stan. *Stat Med*. 2021;40(27):6209-6234. doi:10.1002/sim.9164. Accessed October 4, 2023.
30. Ionides EL, Bretó C, King AA. Inference for nonlinear dynamical systems. *PNAS*. 2006;103(49):18438-18443. doi:10.1073/pnas.0603181103 Accessed October 4, 2023.
31. Allen LJ. *An Introduction to Stochastic Processes with Applications to Biology*. 2nd ed. Boca Raton, Fl: CRC Press; 2010.
32. Smirnova A, Bakushinsky A. On iteratively regularized predictor–corrector algorithm for parameter identification. *Inverse Probl*. 2020;36(12):125015. doi:10.1088/1361-6420/abc530. Accessed October 15, 2023.
33. Hong H, Ovchinnikov A, Pogudin G, Yap C. SIAN: software for structural identifiability analysis of ODE models. *Bioinformatics*. 2019;35(16):2873-2874. doi:10.1093/bioinformatics/bty1069 Accessed October 15, 2023.
34. Meshkat N, Anderson C, DiStefano JJ III. Finding identifiable parameter combinations in nonlinear ODE models and the rational reparameterization of their input–output equations. *Math Biosci*. 2011;233(1):19-31. doi:10.1016/j.mbs.2011.06.001. Accessed October 15, 2023.
35. Miao H, Xia X, Perelson AS, Wu H. On identifiability of nonlinear ODE models and applications in viral dynamics. *SIAM Rev Soc Ind Appl Math*. 2011;53(1):3-39. doi:10.1137/090757009. Accessed October 15, 2023.
36. Smith RC. *Uncertainty Quantification: Theory, Implementation, and Applications*. Philadelphia, PA: SIAM; 2013.
37. Alexander R. Solving ordinary differential equations i: nonstiff problems (e. hairer, sp norsett, and g. wanner). *SIAM Rev Soc Ind Appl Math*. 1990;32(3):485. doi:10.1137/1032091. Accessed October 14, 2023.
38. Ascher UM, Petzold LR. *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*. Philadelphia, PA: SIAM; 1998.
39. Shampine LF, Reichelt MW. The matlab ode suite. *SIAM J Sci Comput*. 1997;18(1):1-22. doi:10.1137/S1064827594276424. Accessed October 14, 2023.
40. Butcher JC. *Numerical Methods for Ordinary Differential Equations*. 3rd ed. UK: John Wiley & Sons; 2016.

SUPPORTING INFORMATION

Additional supporting information can be found online in the Supporting Information section at the end of this article.

How to cite this article: Chowell G, Bleichrodt A, Luo R. Parameter estimation and forecasting with quantified uncertainty for ordinary differential equation models using *QuantDiffForecast*: A MATLAB toolbox and tutorial. *Statistics in Medicine*. 2024;43(9):1826-1848. doi: 10.1002/sim.10036