

# A hybrid framework for compartmental models enabling simulation-based inference

Domenic P.J. Germano <sup>a,b,\*</sup>, Alexander E. Zarebski <sup>a,c,\*</sup>, Sophie Hautphenne <sup>d</sup>, Robert Moss <sup>d</sup>, Jennifer A. Flegg <sup>a</sup>, and Mark B. Flegg <sup>e</sup>

<sup>a</sup>*The School of Mathematics and Statistics, The University of Melbourne, Parkville, VIC, Australia*

<sup>b</sup>*The School of Mathematics and Statistics, The University of Sydney, Camperdown, NSW, Australia*

<sup>c</sup>*Pandemic Sciences Institute, University of Oxford, Oxford, United Kingdom*

<sup>d</sup>*Melbourne School of Population and Global Health, The University of Melbourne, Parkville, Vic, Australia*

<sup>e</sup>*School of Mathematics, Monash University, Clayton, VIC, Australia*

**Keywords**— Hybrid simulation — stochastic modelling — viral clearance — extinction — compartmental modelling

## Abstract

Multi-scale systems often exhibit a combination of stochastic and deterministic dynamics. In compartmental models, low occupancy compartments tend to exhibit stochastic dynamics while high occupancy compartments tend to follow deterministic dynamics. Representing both dynamics with existing methods is challenging. Failing to account for stochasticity in small populations can produce “atto-foxes”, for example in the Lotka-Volterra ordinary differential equation (ODE) model. This limitation becomes problematic when studying the extinction of species or the clearance of infection, but it can be overcome by using discrete stochastic models, such as continuous time Markov chains (CTMCs). Unfortunately, simulating CTMCs is impractical for many realistic models, where discrete events have very high frequencies.

In this work, we develop a novel mathematical framework to couple continuous ODEs and discrete CTMCs: “Jump-Switch-Flow” (JSF). In this framework, compartments can reach extinct states (“absorbing states”), thereby resolving atto-fox-type problems. JSF has the desired behaviours of exact CTMC simulation, but is substantially computationally faster than existing alternatives, by at least one order of magnitude, and can even obtain constant scaling, irrespective of compartment occupancy.

We demonstrate JSF’s utility for simulation-based inference, particularly multi-scale problems, with several case-studies. In a simulation study, we demonstrate how JSF can enable a more nuanced analysis of the efficacy of public health interventions. We also carry out a novel analysis of longitudinal within-host data from SARS-CoV-2 infections to quantify the timing of viral clearance. In this work, we show how JSF offers a novel approach to compartmental model simulation.

## Introduction

Dynamical systems are a powerful method for describing the world, and have successfully been applied in many fields. However, modelling populations which change in size across multiple scales remains a challenge Fowler, 2021. Population sizes in biological processes often change over orders of magnitude, e.g. the spread of infectious disease (Anderson et al., 1991); the boom-and-bust of insect populations (Ludwig et al., 1978); and the immune response to infections, such as HIV (Perelson, 2002), and influenza virus (Baccam et al., 2006).

The dynamics of small populations can be heavily influenced by stochastic effects, while for larger populations these fluctuations often average out, justifying the use of continuum models. When the population under consideration does not change in size over orders of magnitude, there is usually a natural (and obvious) choice between a stochastic or deterministic model. However, for multi-scale models, this decision remains a challenge, and developing modelling methods that can bridge these scales has been a long-standing goal of applied mathematics (Cotter et al., 2016; Flegg et al., 2014; Isaacson, 2013).

Compartmental models describe how quantities (e.g. the number of molecule, cells, or people) change in a dynamical system. Two popular ways to represent compartmental models are ordinary differential equations (ODEs) and continuous time Markov chains (CTMCs). Many ODEs one encounters are actually *ensemble averages* of CTMCs, although the mathematical justification of this is not always straightforward (Kurtz, 1970; Kurtz, 1971; Kurtz, 1972).

In ODEs, the state of the system changes continuously, which can lead to “atto-fox problems” where populations shrink to infeasibly small sizes, i.e. where a real population would likely have gone extinct (Fowler, 2021; Mollison, 1991; Lobry et al., 2015). In CTMCs, the discrete state changes stochastically and there may be absorbing states e.g. extinction of a population. As a result, CTMCs offer a more natural description of small populations sizes, however applying them to large populations can be computationally challenging. Moreover, a range of powerful techniques can be brought to bear on

---

\*These authors contributed equally to this work

ODEs, enabling more thorough analysis. This raises a natural question about what to do when compartment occupancy changes across orders of magnitude.

Exact and approximate algorithms to simulate CTMCs exist Gillespie, 1976; Simoni et al., 2019. Exact methods (e.g. Doob-Gillespie, first/next reaction, and rejection based methods) are computationally expensive when state transitions occur at a high rate (Sanft et al., 2015). Approximate methods (e.g. Tau-leaping (Gillespie, 2001; Cao et al., 2006) and the use of chemical Langevin equations (Rao et al., 2003; Gillespie, 2000; Gibson et al., 2000)) scale to high transition rates but can have unacceptable approximation error. To overcome the limitations of classical approximate methods, over the past two decades, there has been a concerted effort to develop *hybrid* stochastic-deterministic approaches (Simoni et al., 2019; Kreger et al., 2021; Bressloff et al., 2014; Rebuli et al., 2017).

Some hybrid modelling approaches involve partitioning the transitions in a CTMC into fast and slow transitions. Despite having significant implementation bookkeeping and overheads, these approaches can be sampled individually and subsequently synchronised in an efficient manner Simoni et al., 2019. Jump-diffusion differential equations partition the model compartments into *fluid* and *discrete* compartments, to construct *hybrid switching jump diffusion* processes Buckwar et al., 2011; Angius et al., 2015. Other hybrid simulation techniques have been developed in the context of within-host viral infection Kreger et al., 2021. However, this implementation is not generalised, but tailored to a single system, and does not allow switching between stochastic and deterministic regimes. In the context of determining the hybridisation between stochastic and deterministic couplings, path-integrals have been utilised, Bressloff et al., 2014, which permit analysis of hybrid processes. Recently, Kynaston et al. considered extended systems that represent each compartment with both a continuous and a discrete version and allow the for conversion between them Kynaston et al., 2023.

Essential for the real-world utility of many mathematical models is their ability to be calibrated to data. For deterministic compartment models, standard parameter inference methods, such as maximum likelihood and posterior sampling via Markov chain Monte Carlo (MCMC), can readily be applied (Ionides et al., 2006). For stochastic models these inference problems are usually much harder. Simulation-based inference has emerged as a way to handle these problems, e.g. the *particle filter* (Arulampalam et al., 2002; Kitagawa, 1996) and *approximate Bayesian computation* (Alahmadi et al., 2020).

Calibrated multi-scale simulations of ODE (compartment) models is an important milestone towards accurate predictive models that address stochastic behaviour. In a recent paper, Trindade and Zygalakis develop a hybrid scheme for chemical kinetics over different scales with the goal of demonstrating the utility for parameter estimation (Trindade et al., 2024). In this paper, a CTMC is used to model reaction events on small populations (below a threshold  $I_1 > 0$ ) and for better efficiency Tau-leaping is used for reactions that only impact large populations (above a threshold  $I_2 > I_1$ ). Between these two thresholds, a “blending function” is used to create a linear transition between an accurate CTMC and efficient Tau-leaping treatment of the reactions. The hybrid adoption of CTMC and Tau-leaping regimes has the property of maintaining appropriate levels of noise at large populations but struggles as populations grow much larger where a Chemical Langevin or deterministic ODE description may be more computationally appropriate.

In this paper, we present a simple and efficient hybrid simulation method which we call *Jump-Switch-Flow* (JSF). This method enables adaptive transitions between stochastic and deterministic regimes across compartments, while ensuring conservation principles are maintained. In this approach, compartments can individually switch regimes, allowing for the compartments to be split into stochastic and deterministic subsets, while remaining coupled. This regime switching presents unique challenges in matching at interfaces: ensuring that the deterministic regime represents the expected state under transition between regimes and that mass is conserved. Conservation is not always essential when it pertains to small population sizes, however in many cases sustained small populations have a significant impact on larger populations (e.g. enzymes in chemical processes and viruses within hosts.) We demonstrate the utility and properties of the JSF method through simulation studies, and a case study in which we reanalyse existing longitudinal SARS-CoV-2 viral load datasets.

Two simulation studies are presented to explore the properties of JSF and demonstrate its use in an inference setting. In the first simulation study, we compare the computational efficiency and accuracy of JSF with the exact Doob-Gillespie method and the Tau-leaping method. In the second simulation study, we demonstrate the types of insight available when using an approach that supports absorbing (extinction) states.

In the case study, we demonstrate The significance of being able to perform inference with models with absorbing states (i.e. models in which one of the compartments can go extinct.) In the case study we reanalyse longitudinal SARS-CoV-2 viral load data (Ke et al., 2022) using JSF to simulate a TEIRV (Target cells – Eclipsed cell – Infectious cells – Refractory cell – Virions) model and infer the state of host and virus across the infection. Understanding how a viral infection is cleared has important ramifications for both treatment and prevention. For example, the initial exposure may fail to initiate a systematic infection, and understanding the conditions under which this happens is important for infection prevention Pearson et al., 2011. Moreover, the infection may reach low levels, potentially escaping detection, but the virus may rebound and cause disease. Understanding when the virus has been cleared is important for optimising treatment duration. Inferring viral clearance can be computationally challenging (Yan et al., 2016), and this case study demonstrates how JSF enables the tractable estimation of viral clearance.

## Methods

### Jump-Switch-Flow mathematical framework

Consider a compartmental model with  $n$  compartments,  $\vec{V} = (v_1, \dots, v_n)$ , where the state variable,  $v_i := v_i(t)$ , represents the value of the  $i$ th compartment at time  $t$ . For example,  $v_i$  could be the number of people infected with a pathogen, or the copy number of a molecule in a cell. The state variables  $v_i$  may take values from different domains, depending upon the resolution needed, for example in an ODE,  $v_i$  will have real values and in a CTMC  $v_i$  might have integer values.

When the variables represent quantities, typically, discrete values are used to represent small populations, while larger populations are represented with a continuum. To accommodate both scales, let the domain of  $v_i$  be the set  $\{0, 1, \dots, \Omega_i\} \cup (\Omega_i, \infty)$ . The *switching threshold parameter*,  $\Omega_i \in \mathbb{Z}_{\geq 0}$ , is where the  $i$ th compartment transitions between discrete and continuous dynamics. If a compartment  $v_i \in \{0, 1, \dots, \Omega_i\}$ , we call it *discrete* (or *jumping*), and if  $v_i \in (\Omega_i, \infty)$ , we call it *continuous* (or *flowing*). While the switching threshold can be compartment specific, for ease of exposition, we will only consider a single threshold shared between all compartments, i.e.  $\Omega_i = \Omega$ .

At each time  $t$ , the components of  $\vec{V}$  can be partitioned into  $\vec{V}_F$  and  $\vec{V}_J$ , where  $\vec{V}_F$  contains the flowing variables  $v_i > \Omega$ , and  $\vec{V}_J$  contains the jumping variables  $v_i \leq \Omega$ . Therefore, at any moment in time  $\dim(\vec{V}_F)$  of the  $n$  compartments are flowing, and  $\dim(\vec{V}_J)$  of the  $n$  compartments are jumping, with  $\dim(\vec{V}_F) + \dim(\vec{V}_J) = n$ .

The dynamics of each compartment  $v_i$  are described by a set of  $m$  reactions  $\mathcal{R} = \{r_k\}_{k=1}^m$ . Each reaction  $r_k$  is defined by two properties: the rate (per unit time) at which it occurs,  $\lambda_k$ , which (usually) is a function of the state  $\vec{V}$ ; and the reaction's effect on the state, i.e. the change  $\eta_{ik}$  to the size of compartment  $v_i$  when reaction  $r_k$  occurs. As a vector,  $\vec{\lambda} \in \mathbb{R}^m$  is referred to as the *propensity vector*. As a matrix,  $\eta \in \mathbb{Z}^{n,m}$  is referred to as the *stoichiometric matrix*. For ODE models that only contain flowing variables, these reactions occur continuously and are written in the form:

$$\frac{d\vec{V}}{dt} = \eta \vec{\lambda}(t, \vec{V}). \quad (1)$$

For CTMC models, reactions in the system  $\mathcal{R}$  occur as discrete events. In the latter case, each reaction  $r_k$  has a separate propensity described by  $\lambda_k(t, \vec{V})$ . This propensity remains constant between reactions, but when a reaction  $r_k$  occurs, there is a change in  $\vec{V}$  (as specified by the elements of  $\eta_{ik}$ ), and therefore in  $\vec{\lambda}(t, \vec{V})$ .

As an example, consider the SIR model of epidemics, which, in ordinary differential equations, has the form  $\frac{dS}{dt} = -\frac{\beta SI}{S+I+R}$ ,  $\frac{dI}{dt} = \frac{\beta SI}{S+I+R} - \gamma I$ , and  $\frac{dR}{dt} = \gamma I$ . The state of this model is  $\vec{V} = (S, I, R)^T$ , and there are two “reactions”: infections ( $r_1$ ) and recoveries ( $r_2$ ). For infections, the rate of reaction may be modelled by  $\lambda_1 = \beta SI/(S+I+R)$ , and the entries of the associated column of the stoichiometric matrix are  $\eta_{1,1} = -1$ ,  $\eta_{2,1} = 1$  and  $\eta_{3,1} = 0$ . For recoveries, the rate of reaction may be modelled by  $\lambda_2 = \gamma I$ , and the entries of the associated column of the stoichiometric matrix are,  $\eta_{1,2} = 0$ ,  $\eta_{2,2} = -1$ , and  $\eta_{3,2} = 1$ .

We define  $\mathcal{R}_J \subseteq \mathcal{R}$  as the subset of reactions treated as stochastic *jump* events. The set  $\mathcal{R}_J$  can be defined more precisely in two ways. While we use only the second in this manuscript, both are described for clarity. The first way to define  $\mathcal{R}_J$  is as:  $\mathcal{R}_J = \{r_k : \exists i \text{ s.t. } v_i \in \vec{V}_J \text{ and } \eta_{ik} \neq 0\}$ . In this definition, a reaction  $r_k$  is included in  $\mathcal{R}_J$  if and only if the reaction has some material effect on a jumping (discrete) compartment  $v_i \in \vec{V}_J$ . This is a minimum requirement; a reaction should not be permitted to evoke a continuous change in a discrete compartment. However, where it makes sense to do so, it is possible to allow reactions to make discrete changes to flowing (continuous) compartments. The second way to define  $\mathcal{R}_J$ , which we use throughout this manuscript, captures a larger set of reactions:  $\mathcal{R}_J = \{r_k : \exists i \text{ s.t. } v_i \in \vec{V}_J \text{ and } (\eta_{ik} \neq 0 \text{ or } \partial_{v_i} \lambda_k \neq 0)\}$ . In this definition, a reaction is included in  $\mathcal{R}_J$  if either (1) it causes a material change in jumping (discrete) compartments or (2) it is influenced/caused by a discrete compartment (e.g. as catalyst of some reaction.)

Reactions in  $\mathcal{R}_J$  are simulated using stochastically sampled times, similar to CTMC models (and described below in the section on ‘Jump events’). It is important to note that unlike time homogeneous CTMC models, the propensities are *not* constant because the state  $\vec{V}$  (and therefore  $\vec{\lambda}$ ) are continuously varying. When any reaction  $r_k \in \mathcal{R}_J$  occurs, we say the system has *jumped* and an instantaneous change of  $\eta_{ik}$  for each compartment  $v_i$  occurs (irrespective of whether  $v_i \in \vec{V}_J$  or  $v_i \in \vec{V}_F$ , to ensure the conservation of mass). We therefore refer to reactions in  $\mathcal{R}_J$  as *jumps*. The reactions in  $\mathcal{R}_F = \mathcal{R} \setminus \mathcal{R}_J$  represent the continuous change of value of the relevant compartments, all of which are deterministic (see section on ‘Flow events’ above.) We therefore refer to reactions in  $\mathcal{R}_F$  as *flows*. At any moment in time  $|\mathcal{R}_F|$  is the number of reactions which are flowing, and  $|\mathcal{R}_J|$  is the number of reactions which are jumping, such that  $|\mathcal{R}_F| + |\mathcal{R}_J| = m$ .

Finally, the hybrid model that we propose is capable of *switching*. Switching events occur when the size of a compartment  $v_i$  crosses the threshold  $\Omega$  and therefore changes membership between the flowing  $\vec{V}_F$  and jumping  $\vec{V}_J$  sets (see sections on ‘Jump clock updates’ and ‘Switching events’). Importantly, switching events may also impact which reactions are jumps,  $\mathcal{R}_J$ . As such, switches are paradigm-defining events which ought to occur infrequently in comparison to jumps (which may occur frequently) and flows (which occur continuously).

Due to the way that  $\mathcal{R}$  is partitioned, it is possible to order the rows and columns of  $\eta$  into the upper-triangular block form:

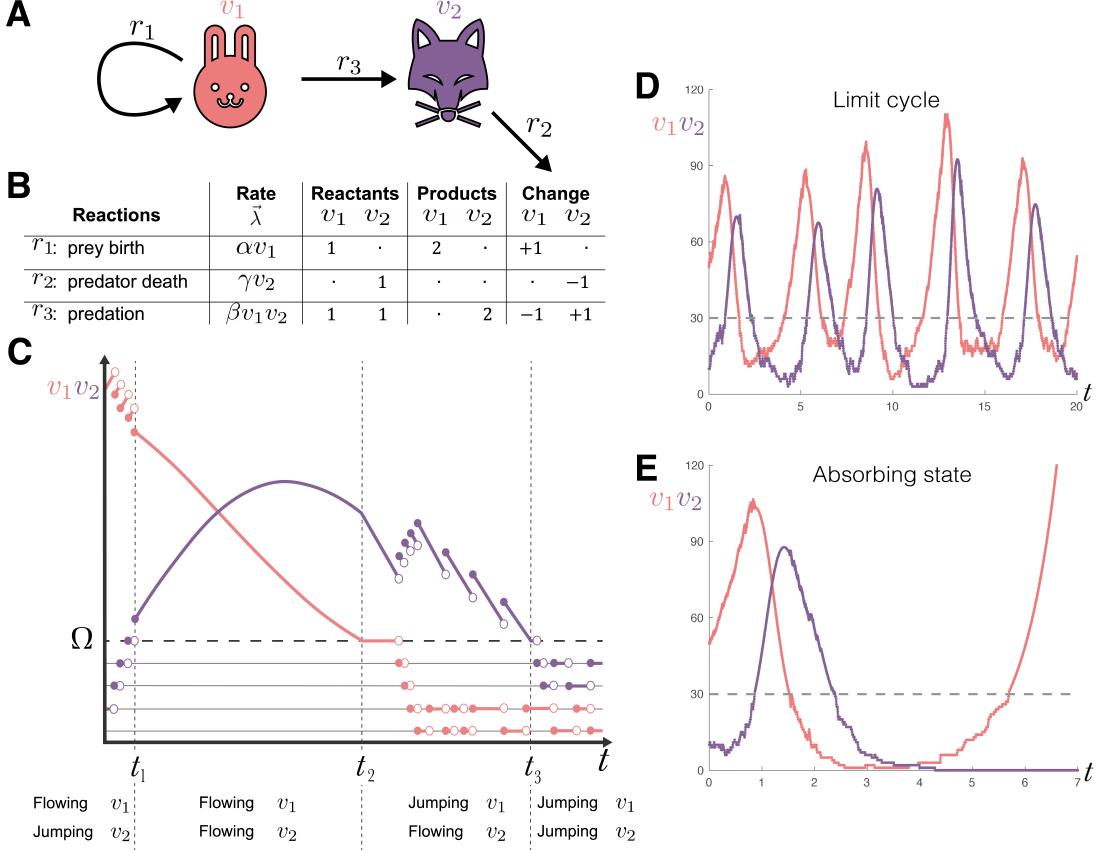
$$\eta = \left( \begin{array}{c|c} \eta_{FF} & \eta_{FJ} \\ \hline 0 & \eta_{JJ} \end{array} \right), \quad (2)$$

where  $\eta_{FF} \in \mathbb{Z}^{\dim(\vec{V}_F), |\mathcal{R}_F|}$ ,  $\eta_{JJ} \in \mathbb{Z}^{\dim(\vec{V}_J), |\mathcal{R}_J|}$  and  $\eta_{FJ} \in \mathbb{Z}^{\dim(\vec{V}_F), |\mathcal{R}_J|}$  refer to stoichiometric coefficients for changes in flowing compartments under flows, jumping compartments under jumps, and flowing compartments under jumps, respectively. Note that, the lower left corner is zeros because, by definition, flowing reactions cannot influence the state of jumping variables. Written as a system of equations analogous to (1), the hybrid JSF framework we propose formally takes the following form. For any time interval  $t_i < t < t_{i+1}$  between switching events  $i$  and  $i+1$ :

$$\frac{d\vec{V}_F}{dt} = \eta_{FF} \vec{\lambda}_F(t, \vec{V}) + \eta_{FJ} \vec{\lambda}_J(t, \vec{V}), \quad (3)$$

$$\vec{V}_J(t) = \vec{V}_J(t_i) + \eta_{JJ} \int_{t_i}^t \vec{\lambda}_J(s, \vec{V}) ds, \quad (4)$$

where  $\vec{\lambda}_F \in \mathbb{R}^{|\mathcal{R}_F|}$  are the reaction rates of flows and  $\vec{\Lambda}_J$  is a stochastic vector of  $|\mathcal{R}_J|$  delta-function spike trains that are derived from the realisations of  $|\mathcal{R}_J|$  different jumps sampled at rates which are dependent on the dynamic changes in the propensities  $\vec{\lambda}_J \in \mathbb{R}^{|\mathcal{R}_J|}$  for these jumps (see ‘Jump events’).



**Figure 1:** JSF provides a way to capture both the continuous deterministic dynamics of large populations and the discrete stochastic dynamics of small populations. **A** A compartmental model representation of the Lotka-Volterra system (5): prey ( $v_1$ , depicted as red rabbits which, in reaction  $r_1$ , reproduce at rate  $\alpha$ ), predators ( $v_2$ , depicted as purple foxes which, in reaction  $r_2$ , die at rate  $\gamma$ ), and their interaction (predation of rabbits by foxes, in reaction  $r_3$ , at a rate proportional to  $\beta$ ). **B** The compartmental model can be formalised as a reaction network via a stoichiometry matrix  $\eta$  which details each species role in the reactions. The reactants consumed  $\eta^-$  and the products produced  $\eta^+$  can be written explicitly with  $\eta = \eta^+ - \eta^-$ . **C** A hypothetical JSF trajectory demonstrating how the representation captures the continuous variation of some compartment-reaction pairs (flow) and the discrete stochastic changes (jumps) in others. Transitioning (switching) between a continuous and discrete state occurs at a threshold  $\Omega$  which is indicated with a horizontal dashed line. **D** An example trajectory from the Lotka-Volterra model exhibiting the characteristic cyclical behaviour with additional stochasticity influencing the dynamics at low population sizes. **E** An example trajectory showing the possibility for the predator species to go extinct and the subsequent growth of the prey species.

### Example: Lotka-Volterra model

The Lotka-Volterra model describes the populations of two species: prey,  $v_1$ , and predators,  $v_2$ . The prey reproduce at rate  $\alpha$ , the predators die at rate  $\gamma$  and new predators are produced through predation at rate  $\beta$ . When modelled with ODEs, this gives us the following system:

$$\begin{aligned} \frac{dv_1}{dt} &= \alpha v_1 - \beta v_1 v_2, \\ \frac{dv_2}{dt} &= \beta v_1 v_2 - \gamma v_2. \end{aligned} \tag{5}$$

The periodic solutions to (5) persist even when the predator species reaches infeasible population sizes: populations of order  $10^{-18}$  giving us “atto-foxes”.

Figure 1A shows a compartmental diagram of the Lotka-Volterra model with the reaction:  $r_1$ , birth of  $v_1$ ;  $r_2$ , death of  $v_2$ ; and  $r_3$ , conversion of  $v_1$  into  $v_2$ . Figure 1B shows a representation of the reactions and their corresponding stoichiometric matrix.

Figure 1C depicts a hypothetical trajectory of the Lotka-Volterra model when represented as a JSF process. There are

three switching events (at  $t_1$ ,  $t_2$  and  $t_3$ ), and each configuration of discrete (jumping) and continuous (flowing) states are featured:

1. Before  $t_1$ , the representation is  $\vec{V}_F = v_1$  and  $\vec{V}_J = v_2$  and the reactions are  $\mathcal{R}_J = \{r_2, r_3\}$  (hybrid regime).
2. During  $[t_1, t_2]$ , the representation is  $\vec{V}_F = (v_1, v_2)^\top$  and  $\vec{V}_J = \emptyset$  and the reactions are  $\mathcal{R}_J = \emptyset$  (ODE regime).
3. During  $[t_2, t_3]$ , the representation is  $\vec{V}_F = v_2$  and  $\vec{V}_J = v_1$  and the reactions are  $\mathcal{R}_J = \{r_1, r_3\}$  (hybrid regime).
4. From  $t_3$  onwards the representation is  $\vec{V}_F = \emptyset$  and  $\vec{V}_J = (v_1, v_2)^\top$  and the reactions are  $\mathcal{R}_J = \mathcal{R}$  (CTMC regime).

Figures 1D and E show two instances of random trajectories sampled from the Lotka-Volterra process as represented with JSF with a switching threshold of  $\Omega = 30$ . In the trajectory in Figure 1D the populations undergo several cycles while in Figure 1E, with the inclusion of discrete stochastic behaviour, the predator species goes extinct, allowing the prey to grow exponentially.

## Mathematical detail

Here we provide the mathematical details of JSF: the evolution and computation of values for flowing compartments (see ‘Flow events’); the evolution and computation of values for jumping compartments (see ‘Jump events’); the sampling of jump times (see ‘Jump clock updates’); and the switching of compartments between jumping and flowing regimes (see ‘Switching events’). Further details, including pseudo-code listings of the algorithms used to sample from the JSF process, are given in SI Section 1.

### Flow events

Equation (3) contains the differential equations describing the evolution of the flowing compartments. If we consider an interval of time between jumps,  $[t_0, t_1]$ , then  $\vec{V}_J(t)$  remains constant and therefore,  $\vec{\Lambda}_J(t, \vec{V}) = \vec{0}$  across this interval. Therefore, the differential equations for the flowing compartments are:

$$\frac{d\vec{V}_F}{dt} = \eta_{FF} \vec{\lambda}_F(t, \vec{V}). \quad (6)$$

This is a standard dynamical system of ODEs, which we numerically integrate forward in time over discrete time steps of size  $\Delta t$ . While an arbitrary order method may be used, we use a simple Forward Euler method:

$$\vec{V}_F(t + \Delta t) = \vec{V}_F(t) + \Delta \vec{V}_F(t) \quad (7)$$

$$= \vec{V}_F(t) + \Delta t \eta_{FF} \vec{\lambda}_F(t, \vec{V}) + O(\Delta t^2). \quad (8)$$

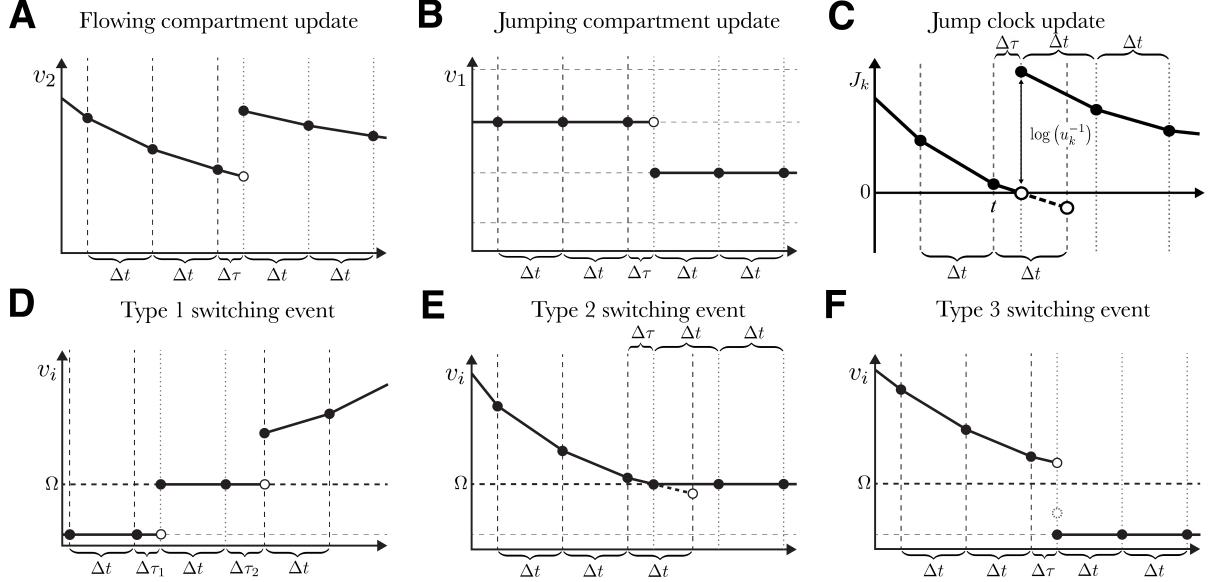
Figures 2A and 2B depict examples of a flowing compartment and a jumping compartment, respectively. Flow updates are given at intervals of discrete time step  $\Delta t$  as indicated. In the Jump-Switch-Flow algorithm, flowing time steps are bounded above by a finite, but small,  $\Delta t$ . Flowing time steps have this maximum duration if no jumps are found on this time interval (that is, if (6) remains valid). If a jump occurs at  $\Delta\tau < \Delta t$  from any given time (see Section ‘Jump events’) then a flowing time step of  $\Delta\tau$  is used instead of  $\Delta t$  in Equation (7). In this case, a jump is instantaneously applied after the flow event and the next interval of time is associated with new flow rates  $\vec{\lambda}_F$  (which are therefore updated at the exact moment of time of the jump event). We note therefore that error associated with the ODE solver is determined by the size of the finite capped time interval  $\Delta t$  according to classical ODE error analysis. In this case since we use Forward Euler, the error scale like  $O(\Delta t^2)$ . Using Equation (7), over each time step (flowing interval), flow events only directly affect flowing compartments as can be seen in Figure 2A. Jumping compartments like that shown in Figure 2B are unaffected directly by flowing events because the moments of a jumping event are pre-determined according to approximations in the trajectories of the flowing compartments which have the same accuracy as the ODE solver (for example, we assume a linear changes in the flowing compartments as consistent with Forward Euler method used in Equation (7) – see Section ‘Jump events’ for details. As indicated by discontinuities in both Figure 2A and Figure 2B, jump events may be applied to flowing and jumping compartments alike in order to conserve mass in a manner which we shall now describe.

### Jump events

The reactions in  $\mathcal{R}_J$  produce discontinuous jumps in the state vector  $\vec{V}$ . In Equations (3) and (4), each element in the vector  $\vec{\Lambda}_J$  corresponds to a reaction in  $\mathcal{R}_J$ . Consider, for example,  $r_k \in \mathcal{R}_J$ . The  $k^{\text{th}}$  element of  $\vec{\Lambda}_J$  is:

$$\Lambda_J^{(k)} = \sum_i \delta(t - t_i^{(k)}), \quad (9)$$

where  $\delta$  is the Dirac delta function and  $t_i^{(k)}$  is the time at which the reaction  $r_k$  occurs for the  $i^{\text{th}}$  time. In this way, the term  $\vec{\Lambda}_J$  in Equations (3) and (4) manifests as discrete jumps in both  $\vec{V}_F$  and  $\vec{V}_J$  at each jump. For the sake of the simulation, the computation of the jump times  $t_i^{(k)}$  for each instance  $i$  of each reaction  $r_k \in \mathcal{R}_J$  is all that is required. The stoichiometric matrix describes the change to each compartment when that jump event occurs.



**Figure 2:** The JSF process can be numerically integrated. **A** and **B**: Dynamic changes in flowing compartments (**A**) and jumping compartments (**B**). Flows are computed using the time step  $\Delta t$ . If a jump occurs within a flow update, the flows are adjusted up until the corresponding time  $\Delta\tau$  as a time step instead of  $\Delta t$  (indicated by a discontinuity in both **A** and **B**). Jumps occur when a jump clock reaches zero. **C**: The jump clock  $J_k$  for reaction  $k$  counts down until event  $k$  occurs when  $J_k = 0$ , i.e. between  $t$  when  $J_k > 0$  and  $t + \Delta t$  when  $J_k < 0$ . The time of the jump is found by interpolating for  $t + \Delta\tau$  using Equation (16), where  $\Delta\tau < \Delta t$ . Once the jump has occurred and jumps implemented according to the respective stoichiometric updates,  $J_k$  is reinitialised by sampling  $J_k = -\log(u_k)$  where  $u_k \sim \text{Unif}(0, 1)$ , and then counts down with successive time steps  $\Delta t$  until the next jump  $k$  is realised. **D** A switching event of type 1 occurs when a discrete compartment jumps above the switching threshold,  $\Omega$ . This is shown here after the jump time step  $\Delta\tau_2$ . **E** A switching event of type 2 occurs when a flowing compartment reaches the switching threshold,  $\Omega$ . Here, the time at which the switch occurs is found using continuation. **F** A switching event of type 3 occurs when a flowing compartment is caused to go below the switching threshold,  $\Omega$ , due to a jump in another compartment. Here the jump has caused the flowing compartment to end at the dotted circle (which we denote as  $\hat{v}_i$ ). This compartment is then re-labelled as jumping. Since jumping compartments require integer states, the state is re-initialised to take the value  $\lceil \hat{v}_i \rceil$  with probability  $v_i - \lfloor \hat{v}_i \rfloor$ , and otherwise  $\lfloor \hat{v}_i \rfloor$  such that the expected value of the state is at the dotted circle (in the example presented here, the variable was rounded down). For all switching events, membership of  $V_F$  and  $V_J$  is recomputed after switching.

The rate of reaction  $r_k$ ,  $\lambda_k(t, \vec{V})$ , may be state dependent so can change due to the continuous change caused by flows. Figures 2A and 2B depict an example of flowing and jumping compartment (respectively) experiencing a jump event. In this illustration, we note that if  $\lambda_k$  depends on  $v_2$  the event time  $t_i^{(k)}$  is not sampled from an exponential distribution which requires a constant propensity/rate between events. Formally, the jump times form an inhomogeneous Poisson process. There are multiple ways to sample the jump times  $t_i^{(k)}$ , see Klein et al., 1984 for a detailed discussion. We use a variant of the Next Reaction Method Gibson et al., 2000 (which is an optimised variant of the Doob-Gillespie method Gillespie, 1976) to sample jump times. We first note that the propensity for a jump is dependent only on the instantaneous state  $\vec{V}$ , and therefore at time  $t_0$  if there has been  $i-1$  jumps associated with reaction  $r_k$ , it has no bearing on the distribution of the time  $t_i^{(k)}$ . Therefore, we shall simply denote  $t_i^{(k)} = t_k$  as the *next* jump time for reaction  $r_k$ . The cumulative probability function from which  $t_k$  is sampled depends on the current time,  $t_0$ , and the evolution of the state variables in time,  $\vec{V}(t)$ . The cumulative distribution for the time of the next event associated with  $r_k$  is:

$$\text{CDF}(t; k) = 1 - \exp \left\{ - \int_{t_0}^t \lambda_k(s, \vec{V}(s)) ds \right\}. \quad (10)$$

To sample  $t_k$ , inverse transform sampling is used Klein et al., 1984, i.e. sample  $u_k \sim \text{Unif}(0,1)$  and then solve  $\text{CDF}(t_k; k) = u_k$  for  $t_k$ . To account for the varying state, we define a new function,  $J_k(t)$  the *jump clock* for reaction  $r_k$ , using Equation (10), and solving for when  $\text{CDF}(t_k; k) = u_k$  (inverse transform sampling). The jump clock acts as a timer, identifying the time  $t_k$  for when  $r_k$  next occurs, when  $J_k(t_k) = 0$ . The function  $J_k$  is defined by:

$$J_k(t_k) = -\log(u_k) - \int_{t_0}^{t_k} \lambda_k(s, \vec{V}(s)) ds, \quad (11)$$

noting that  $u_k$  and  $1-u_k$  have the same distribution. In general, we cannot solve directly for  $t_k$ , so we solve for it numerically by tracking the value of  $J_k(t)$  as  $\vec{V}$  evolves through flows, jumps and switches.

For each reaction  $r_k$ , at some initial time, for example  $t_{i-1}^{(k)}$ , being the continuous time of the  $(i-1)^{\text{th}}$  jump associated with reaction  $r_k$ , we sample  $u_k$  and initialise  $t_0 = t_{i-1}^{(k)}$ . The initial value of  $J_k(t)$  is therefore equal to the positive number  $\log(u_k^{-1})$  (since  $u_k \sim \text{Unif}(0,1)$ ). As time progresses,  $J_k(t)$  decreases according to (11) since  $\lambda_k \geq 0$ . The value of  $J_k(t)$

decreases to zero over time and when  $J_k(t) = 0$ , a jump associated with  $r_k$  is triggered (hence the name ‘‘jump clock’’). Once a jump clock reaches 0 and a jump is triggered, the clock is reset by sampling a new random number,  $u_k \sim \text{Unif}(0, 1)$ . See Figure 2C for a schematic illustration of how the jump clock is updated.

To update the jump clock, we require numerical integration of  $\lambda_k(t, \vec{V}(t))$  forward in time. Fortunately, we also have piece-wise polynomial approximations for  $\vec{V}_F(t)$  as a result of our numerical treatment of the continuous flows (see Subsection ‘‘Flow events’’), combined with piece-wise constant values for  $\vec{V}_J(t)$  which only change when jumps occur. We discuss the numerical integration of the jump clock below.

### Jump clock updates

For a given jump reaction  $r_k$ , a jump clock is initialised at time  $t_0$  with  $u_k \sim \text{Unif}(0, 1)$  giving  $J_k = \log(u_k^{-1})$ . From time  $t$  to  $t + \Delta t$ , with  $t > t_0$ , Equation (11) tells us the clock ticks down from  $J_k$  to  $J_k - \Delta J_k$ , where  $\Delta J_k$  is

$$\Delta J_k = \int_0^{\Delta t} \lambda_k(t+s, \vec{V}(t+s)) \, ds. \quad (12)$$

In general this integral is intractable, but we can approximate it as follows: consider the Taylor expansion of  $\lambda_k(t+s, \vec{V}(t+s))$  about  $s = 0$ , which gives

$$\Delta J_k = \int_0^{\Delta t} \lambda_k(t, \vec{V}) + \frac{\partial \lambda_k}{\partial \vec{V}} \frac{d\vec{V}^\top}{dt} s + O(s^2) \, ds = \int_0^{\Delta t} \alpha + \beta s + O(s^2) \, ds, \quad (13)$$

where we write  $\alpha = \lambda_k(t, \vec{V})$ , and  $\beta = \frac{\partial \lambda_k}{\partial \vec{V}} \frac{d\vec{V}^\top}{dt}$ . Since (i) our numerical evaluation of  $\vec{V}$  is piece-wise linear across the time interval, by virtue of the fact that we use a Forward Euler approximation to solving Equation (6), and (ii)  $\Delta t$  is small,  $\Delta J_k$  is approximated to the precision of our algorithm by taking

$$\Delta J_k \approx \frac{\Delta t}{2} (2\alpha + \beta \Delta t), \quad (14)$$

noting that since  $\vec{V}$  is piece-wise linear across the time interval, the convergence of the integration of the Jump-clock also behaves as  $\Delta t$ . Importantly, we know that between jumps  $\frac{d\vec{V}_F}{dt}$  is given by Equation (6) whilst  $\frac{d\vec{V}_J}{dt} = 0$ . Thus,  $\beta = \frac{\partial \lambda_k}{\partial \vec{V}_F} \left( \eta_{FF} \vec{\lambda}_F(t, \vec{V}) \right)^\top$  (evaluated at  $t$ ). To calculate the updated jump clock, we compute  $J_k(t + \Delta t) = J_k(t) - \Delta J_k$  as the provisional value. We then have two distinct cases: (i)  $J_k(t) - \Delta J_k > 0$ , then no jump occurred during the interval  $(t, t + \Delta t)$  and we have the jump clock  $J_k(t + \Delta t) := J_k(t) - \Delta J_k$ ; (ii)  $J_k(t) - \Delta J_k < 0$ , then a jump occurred (i.e. a  $r_k$  reaction) during the interval  $(t, t + \Delta t)$ , that we need to account for in the updated jump clock. In the case (ii) where a jump occurs within the interval  $(t, t + \Delta t)$ , let  $t + \Delta\tau$ , where  $0 < \Delta\tau < \Delta t$ , denote the time at which this jump occurs. We can find  $\Delta\tau$  by interpolation and solving the following for  $\Delta\tau$ :

$$2\Delta J_k - \Delta\tau(2\alpha + \beta\Delta\tau) = 0, \quad (15)$$

where  $\Delta J_k$  is the residual of the jump clock from  $t$  to  $t + \Delta\tau$ .  $\Delta\tau$  is then given by

$$\Delta\tau = \begin{cases} \frac{\sqrt{\alpha^2 + 2\beta\Delta J_k} - \alpha}{\beta}, & \beta \neq 0, \\ \frac{\alpha}{\Delta J_k}, & \beta = 0. \end{cases} \quad (16)$$

If there is a jump, rather than using  $\Delta t$  to forward compute the flowing compartments in Equation (7) we instead use  $\Delta\tau$  to take part of a flow step and then implement the jump after the flow to get the state at time  $t + \Delta\tau$ . After this, we reinitialise the jump clock  $J_k$  as described above. This procedure is illustrated in Figure 2C.

### Switching events

The way in which we handle switching is one of the main contributions of this manuscript. Switching events occur when variables transition between  $\vec{V}_J$  and  $\vec{V}_F$ , i.e. between discrete or continuous values. As a consequence, the contents of  $\mathcal{R}_J$  and  $\mathcal{R}_F$  may change. There are three types of switching events. The first involves a compartment moving from  $\vec{V}_J$  to  $\vec{V}_F$ . This transition is straightforward as a new equation is added to Equation (6), and the state  $\vec{V}_F$  is initialised at the switching time, with the new flowing compartment at  $\Omega$ . Figure 2D depicts an example of a type 1 switching event. The second type involves a flowing compartment moving from  $\vec{V}_F$  to  $\vec{V}_J$ . Here, the flowing compartment reaches the switching threshold,  $\Omega$ , and becomes discrete and no longer follows the flowing dynamics (of Equation (6)). The time at which this occurs is found by continuation, and computation is resumed. Figure 2E depicts an example of a type 2 switching event.

The third type of switching event involves a compartment jumping from  $\vec{V}_F$  to  $\vec{V}_J$ . In this case, a flowing compartment jumps down to the switching threshold  $\Omega$ , and becomes discrete. To highlight how the third type of switch event is necessary, we propose considering a simple model where two species,  $X$  and  $Y$ , interact to produce a third species,  $Z$ , with  $X$  also undergoing a death dominated birth-death process. Suppose that the switching threshold is set as  $\Omega = 1000$ , and that  $X(0) = 1001$ ,  $Y(0) = 10$  and  $Z(0) = 0$ . Due to the death reaction of  $X$ , suppose that within the first (Flow) step,  $X$  experiences some decay  $\varepsilon < 1$ . Therefore,  $X(t_1) = 1001 - \varepsilon > \Omega = 1000$  is in a flowing (continuous) regime. Now let’s suppose that in the next instance,  $X$  and  $Y$  combine to produce one  $Z$ . At this instance,  $X$  must experience some integer loss (which we can assume to be 1, but in general may be any positive value). In this case, producing  $Z$  will result in  $X(t_2) < \Omega$  but non integer. Therefore, to account for type three switch events, let  $v_i$  be the compartment switching from

$\vec{V}_F$  to  $\vec{V}_J$  due to a jump. In general, these types of jumps result in  $v_i$  being non-integer, i.e.  $v_i \notin \{0, 1, \dots, \Omega\} \cup (\Omega, \infty)$ . To ensure the values of  $v_i$  stay in  $\{0, 1, \dots, \Omega\} \cup (\Omega, \infty)$ , we add another constraint to the process, initially proposed by Rebuli et al. Rebuli et al., 2017. The idea is to randomly round the compartment value up or down to an integer in a way that conserves the average behaviour of the process. Let  $\hat{v}_i$  be the value of the flowing compartment  $v_i$  after jumping down across the threshold  $\Omega$  but before being rounded into  $\{0, 1, \dots, \Omega\} \cup (\Omega, \infty)$ . We apply the following rule to round  $v_i$  after the switch. We take  $v_i = \lceil \hat{v}_i \rceil$  with probability  $\hat{v}_i - \lfloor \hat{v}_i \rfloor$ , and otherwise we round down by setting  $v_i = \lfloor \hat{v}_i \rfloor$ . This ensures the expected value of the variable after the switch is  $\hat{v}_i$  as described under the flowing paradigm from which this compartment has come and that the variable remains in the domain  $\{0, 1, \dots, \Omega\} \cup (\Omega, \infty)$ . Figure 2F depicts an example of a type 3 switching event.

## Results

### Effects of interventions to reduce transmission

Simulation based inference (e.g. the particle filter) relies on being able to efficiently simulate from the generative process. To demonstrate how JSF can be used in this setting, we first present a simulation study of forecasting the elimination of a infectious pathogen.

### SIRS model with demography

The SIRS model with demography is an extension of the classic susceptible-infectious-removed (SIR) model. In the SIR model *susceptible* individuals may be infected by contact with *infectious* individuals; infected individuals eventually cease to be infectious and transition to the *removed* compartment. The SIRS model with demography extends the SIR model by allowing removed individuals to transition back to being susceptible to infection, for individuals to give birth to new (susceptible) individuals, and to allow for death of individuals (at equal rates). The SIRS model with demography can be written as the following ODE system, with susceptible (S), infectious (I), and recovered (R) individuals:

$$\begin{aligned} \frac{dS}{dt} &= -\beta \frac{IS}{N} + \omega R + \kappa N - \mu S, \\ \frac{dI}{dt} &= \beta \frac{IS}{N} - \gamma I - \mu I, \\ \frac{dR}{dt} &= \gamma I - \omega R - \mu R, \end{aligned} \tag{17}$$

where  $N(t) = S(t) + I(t) + R(t)$  is the total population size,  $\beta$  the infection rate,  $\gamma$  the recovery rate,  $\omega$  the immunity waning rate,  $\kappa$  the birth rate, and  $\mu$  the death rate. Figure 3A shows a compartmental diagram of the SIRS model with demography.

Figure 3B shows our simulated (via the Doob-Gillespie algorithm) time series of daily noisy measurements of the prevalence of infection. The true parameters used in the simulation are shown in Figure 3C; these values are broadly consistent with existing estimates for influenza or SARS-CoV-2. We assume that these measurements are drawn from a negative binomial distribution where the expected value is equal to the true prevalence of infection and there is a constant (known) dispersion parameter,  $k = 100$ .

### Estimating elimination probabilities

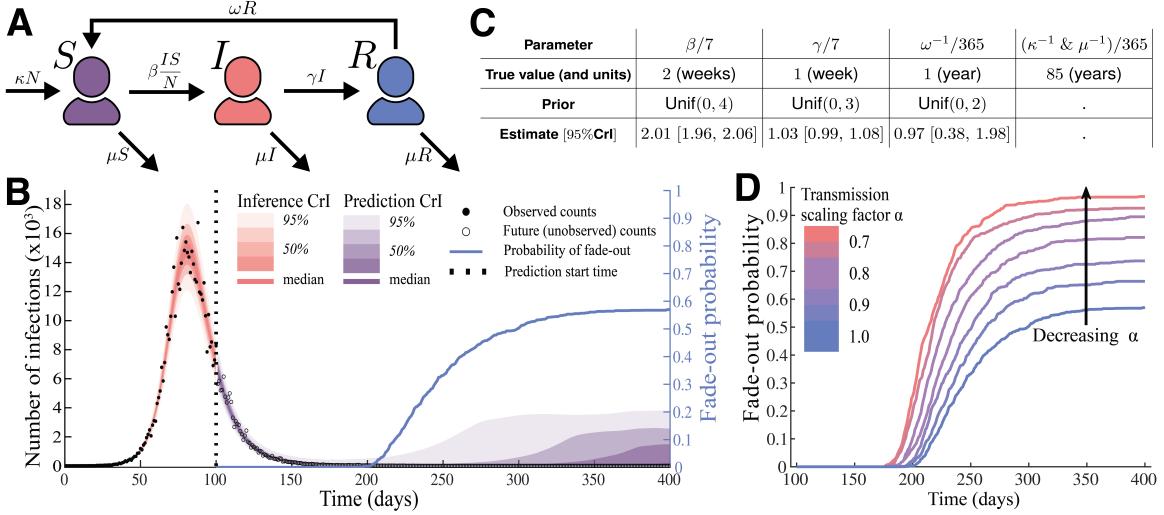
Combining the SIRS model with a particle filter (Moss, 2024; Kitagawa, 1996) we used the first 100 days of the time series to forecast the remaining 150 days. Figure 3B shows the estimated true prevalence across the first 100 days and forecasts the prevalence for the subsequent 150 days. Included in the forecast is a daily estimate of the probability that the pathogen has been eliminated (in the simulation the pathogen was eliminated on day 250). It is important to note that there is an endemic equilibrium in this model, so it is reasonable for the probability to plateau as it does.

To demonstrate the utility of the posterior distribution (i.e. the capacity of the particle filter to forecast the epidemic after the 100 days of observed data), we considered the impact of a possible intervention. This intervention, introduced on day 100, reduces the force of infection (and hence the effective reproduction number) by a factor of  $\alpha$ . Figure 3D shows how the probability of eliminating the pathogen increases substantially as we decrease  $\alpha$  from 1 (no intervention) to 0.7 (a 30% reduction in transmission). The fade-out probability is calculated as the proportion of sampled (hybrid stochastic-deterministic) trajectories that go extinct (via  $I = 0$ ) of the total number of sampled trajectories.

The full details of the configuration of the particle filter, and the marginal posterior distributions are given in SI Section 2E.

### Computational properties

To further investigate the computational properties of JSF and to compare it with exact (Doob-Gillespie) simulation and (approximate) tau-leaping simulation we used repeated simulations from the SIRS model with demography. We partitioned



**Figure 3:** JSF enables us to forecast when a pathogen will be eliminated from a population, i.e. when we can claim that an epidemic has “ended”, and quantify the likely impact of varying levels of intervention. **A** A compartmental model representation of the SIRS with demography. **B** A simulated time series of noisy measurements of prevalence along with inferred prevalence trajectories and forecasts of future prevalence with associated uncertainty bands. The dashed vertical line indicates the date up until which we have data. The solid blue line indicates an estimate of the probability that the pathogen has been eliminated from the population (as opposed to persisting at very low levels). **C** The true parameters used in the simulation along with their posterior estimates and the associated priors used. Each of the marginal posterior distributions peaks tightly about the true value. **D** The probability of pathogen elimination increases with the strength of the intervention against transmission. Near certainty of elimination is achieved with a 30% reduction of transmission.

the simulations into three classes: early stochastic extinction, fade-out after a single epidemic, and sustained transmission (SI Section 2.) As seen in Table SI 3, JSF and exact simulation generate very similar proportions of samples in each of these classes. Two aspects of this system of practical importance are the timing and magnitude of peak infections, and the total number of infections. As can be seen in Figure SI 7 the distribution of peak timing and total number of infections are in good agreement between JSF and exact simulation. For the magnitude of peak infections, there is substantially less variability in the JSF samples than the exact samples, but the proportional difference is small.

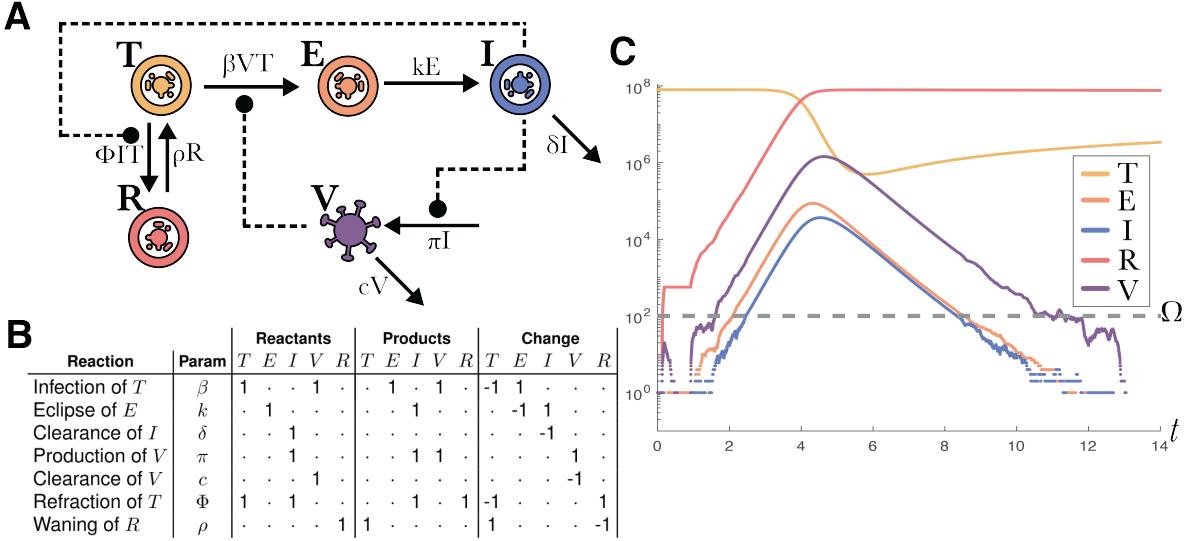
With regards to computational efficiency, Figure SI 10B shows the average time required per simulation using JSF, exact simulation (Doob-Gillespie), tau-leaping, and the efficient Tau-hybrid method Matthew et al., 2023 for the SIRS model for populations of different sizes. For populations of size above about  $10^5$  (i.e. approximately the size of a small city), JSF is substantially computationally faster than either exact simulation or tau-leaping, there being a difference of at least one order of magnitude in the amount of time needed, and (if a suitable Switching threshold is specified) a constant scaling, irrespective of compartment occupancy, may be obtained. To simplify the interpretation of the results in Figure SI 10B, each of the methods were implemented in the same interpreted language so that the observed differences can be attributed to the algorithm rather than the platform on which it is implemented. We have also included a comprehensive simulation case study of the simple birth-death process (see Section SI 2 of the supplementary information), which exhibits JSF computational efficiency for both birth dominated and death dominated regimes, with comparable results to the exact stochastic process. We also demonstrate JSF’s accuracy for particularly difficult to solve stochastic simulations which exhibit multiple time scales (see Section SI 3 of the supplementary information), where we see that JSF produces summary statistics and trajectories comparable to the exact stochastic process, out performing current hybrid methods.

### SARS-CoV-2 virus clearance informed by longitudinal data

Understanding viral clearance is important when deciding how long treatment must be maintained. This case study demonstrates how JSF enables inference of virus clearance, which has been computationally challenging to date Yan et al., 2016; Farrukee et al., 2018. In particular, we use a mathematical model — the TEIRV model described below — to study viral clearance using longitudinal data from 6 individuals infected with SARS-CoV-2 Ke et al., 2022.

### TEIRV model of within-host viral dynamics

The TEIRV model is an extension of the classic target-infected-virus (TIV) model Perelson, 2002. In the TIV model *target* cells may be *infected* by the virus; before dying, infected cells produce *virus*; and the virus can degrade or infect remaining target cells. The TEIRV extends the TIV model through the inclusion of an *eclipsed* compartment, to model the delay between infection of a target cell and the subsequent production of virus, and a *refractory* compartment, to model heightened antiviral defences of target cells, e.g. through the effects of interferons Ke et al., 2022; Blanco-Melo et al., 2020. To simplify



**Figure 4:** The TEIRV model describes the within-host dynamics of host cells and virus during infection. The model accounts for delays in virus production post cellular infection and the capability of target cells to enter a refractory state as a defence against infection. **A** A compartmental diagram of the TEIRV model showing the interactions between: *Target cells (T)*, *Refractory cells (R)*, cells in the *Eclipsed* phase of infection (*E*) and *Infectious* cells (*I*); and the (*Total*) *Virions* (*V*). Total Virion particles are utilised as to not distinguish between intra- and extra-cellular Virions. Solid arrows represent the exchange of mass (cells or virions) through compartments, while dashed lines represent the influence of a compartment on a reaction's rate of occurrence. **B** The representation of this model as a reaction network with a stoichiometric matrix. This formalises the dependency on different variables indicated by the dashed arrows in the compartmental diagram. **C** An example trajectory showing the exponential growth and decline in the amount of free virus across the duration of the infection, this process terminates in the virus, eclipse and infected cell populations reaching zero (shortly before time 12). Reaching zero which is possible in this hybrid continuous/discrete stochastic model but does not occur in a purely ODE based representation.

the use of this model a quasi-steady-state approximation for interferon production is employed. This approximation allows us to avoid the need to explicitly model the amount of interferon present.

Figure 4A shows a compartmental diagram of the TEIRV model. Target cells becoming infected by virions at rate  $\beta$ , which then enter the eclipsed phase. These eclipsed cells then become infectious at rate  $k$ . Infectious cells are cleared from the population at rate  $\delta$ , and produce virions at rate  $\pi$ . Virions are themselves cleared at rate  $c$ . We note that  $V$  is the total number of virion particles, and therefore this model does not distinguish between intra- and extra-cellular virions. As a result, following the infection of a target cell (*T*), no virion particles are lost, in this model. The infectious cells recruit interferon, which cause the target cells to become refractory (and hence protected against infection) at rate  $\Phi$ . The refractory cells return to a naive state as target cells at rate  $\rho$ . These assumptions are represented with the following ODE system:

$$\begin{aligned} \frac{dT}{dt} &= -\beta VT - \Phi IT + \rho R, \\ \frac{dE}{dt} &= \beta VT - kE, \\ \frac{dI}{dt} &= kE - \delta I, \\ \frac{dV}{dt} &= \pi I - cV, \\ \frac{dR}{dt} &= \Phi IT - \rho R. \end{aligned} \tag{18}$$

The stoichiometric matrix corresponding to these assumptions is shown in Figure 4B. Figure 4C, which shows a trajectory sampled from the TEIRV model when represented with JSF; the classic boom-bust dynamics of the viral populations can be seen in the exponential growth and decline of the *V* compartment. Unlike solutions of the ODE model in (18), we see that by time  $t = 14$ , the populations of eclipsed and infected cells, and the virus have gone extinct, indicating a definitive end to the infection.

The basic reproduction number,  $\mathcal{R}_0$  is a fundamental quantity of epidemiological models. For the TEIRV model,  $\mathcal{R}_0$  can be calculated from the next-generation matrix Diekmann et al., 2010:

$$\mathcal{R}_0 = \frac{\pi \beta T(0)}{\delta c}. \tag{19}$$

## Observation model linking virus to time series

The longitudinal data contains cycle number (CN) values from nasal samples. The CN values are inversely proportional to the number of virions present. In our analysis we used an existing (empirical) model to link the cycle numbers to the (logarithmic) viral genome load Ke et al., 2022:  $\log_{10} V = 11.35 - 0.25\text{CN}$ . We assume the observed values are drawn from a normal distribution with mean  $\log_{10} V$  and unit standard deviation, and that the values are truncated at the detection limit of  $-0.65$ .

## Estimating virus reproduction and clearance

Recall that the particle filter is capable of combining a mechanistic model of the within-host dynamics, such as the TEIRV, and an observation model, such as the viral load measurements, and will return a (Bayesian) posterior sample of both the parameters of the process and the trajectory of virus and cell populations through time. We assume that the infection begins with a single exposed target cell (in a population of  $8 \times 10^7$  target cells) Ke et al., 2022. This gives us an initial condition for the process:  $T(0) = 8 \times 10^7$ ,  $E(0) = 1$ ,  $I(0) = 0$ ,  $R(0) = 0$ . We leave the initial viral load,  $V(0)$ , as a parameter to be fit to the data.

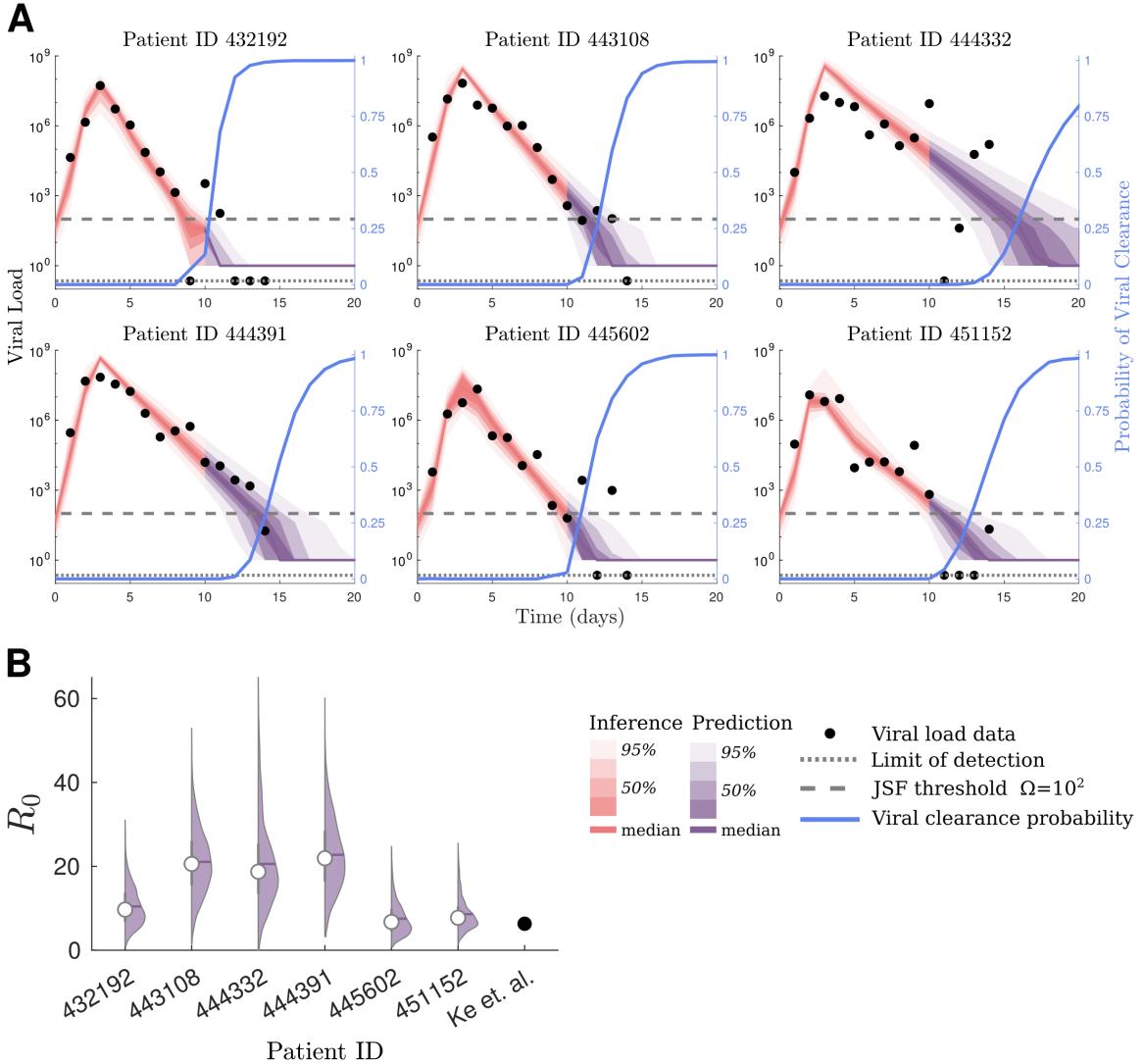
We selected six patient time series Ke et al., 2022, choosing ones that contained a full 14 data points, for both consistency and simplicity. Two of the model parameters were fixed as in previous analysis Ke et al., 2022:  $c = 10$ ,  $k = 4$ . See SI Section 3 for full details of the particle filter and JSF configuration.

Figure 5A shows our model fits to the first 10 days of viral load data for the six selected patients. After day 10, using the estimated parameters, we generate and predict the distribution of subsequent viral load until day 20. The estimated viral peak coincides with the data for each of the patients and the predicted viral trajectories closely match subsequent observations.

We estimate the probability of viral clearance over time (right blue axis of in the panels of Figure 5A). Viral clearance is defined by the point at which the virion, eclipsed, and infected cell populations all reach extinction. We note that this is a conservative measure if one were only interested in when a patient's *infectiousness* becomes negligible. However, for proof of concept of inferring such quantities, we have chosen to estimate the probability of complete viral clearance.

There is clear heterogeneity in the amount of time required to clear the virus (Figure 5A). We estimate that with high certainty patients 423192, 443108 and 445602 clear the virus within 20 days. In contrast, for patients 444332, 444391 and 451152 we infer that they have not cleared the virus within that time frame. Within those who did clear the virus, we observe that different patients require different amounts of time to clear the virus. Specifically, patient 432192 (5A) is inferred to have cleared the virus first. Patients 443108 and 445602 both require an estimated 16 days to obtain viral clearance.

The posterior distributions of the parameters (Figure SI 11) and priors (Table SI 5), are given in SI Section 5. Because of potential identifiability issues with the rate parameters, we instead compare the estimated reproduction number,  $R_0$ , to results from previous analysis (Ke et al., 2022). Figure 5B shows the posterior distributions of  $R_0$ , as well as the point estimates from Ke et al., 2022. Since the previous analysis, (Ke et al., 2022), reports identical estimates for  $\pi$ ,  $\beta$  and  $\delta$  across these patients, we only include a single point estimate for their work. Our estimate is consistent with previous results (Ke et al., 2022) for patients 432192, 445602 and 451152. However, patients 443108, 444332 and 444391 all have higher  $R_0$  estimates in our results. Our  $R_0$  estimates are consistent with similar within-host viral infection analyses of other respiratory pathogens (Baccam et al., 2006; Hernandez-Vargas et al., 2020; Gubbins, 2024).



**Figure 5:** JSF enables the inference of the viral load through time and the probability that the virus has been cleared from longitudinal data of SARS-CoV-2 viral load measurements. **A** Model fits to viral load data based on nasal swabs of six patients from Ke et al., 2022 using a refractory cell within-host model, using our Jump-Switch-Flow method. Nasal viral load data is shown as solid dots, and the limit of detection is shown as a dotted line. We show the switching threshold,  $\Omega = 10^2$  by a dashed line. The red and purple bands show mean (0%), 25%, 50%, 75% and 95% credible intervals (from dark to light) for the inference and prediction, respectively. We also estimate the probability of viral clearance at a given point in time, based on the previous data to that point, in blue (light). **B** The posterior distribution of viral  $R_0$  for each patient, along with the median (white circle), These patient specific estimates are contrasted with the single point estimate from previous analysis Ke et al., 2022.

## Discussion

We have presented a simple hybrid simulation method for compartmental models: *Jump-Switch-Flow* (JSF). This method facilitates efficient simulation of multi-scale models. Via simulation studies and an analysis of longitudinal data from SARS-CoV-2 infections we demonstrated the desirable computational properties of this method and the types of novel analyses which it enables. JSF allows compartments to dynamically change between stochastic and deterministic behaviour. Combining JSF with a simulation based inference method, e.g. a particle filter, enables inference for multi-scale models, and in situations where absorbing states (such as a population going extinct) are important.

Simulations show JSF produces trajectories largely consistent with gold standard exact simulation techniques, e.g. Doob-Gillespie algorithm (see for example Figures SI 1–SI 4 and SI 6 – SI 9). We demonstrate how JSF is highly accurate at producing summary statistics comparable to gold standard exact simulation techniques (see Figures SI 1 – SI 4), with a significant improvement in accuracy for problems with multiple time scales when compared to currently available hybrid methods (see Figure SI 3 and SI 4). However, JSF is also much faster for realistic population sizes (as can be seen in the comparison of computational speed in Figure SI 10B). We demonstrate how our approach can be incorporated into a particle filter (Moss, 2024), to perform parameter and state estimation and prediction (see Figure 3).

A strength of JSF is the capacity to infer when a process has reached an absorbing state, for example, an epidemic fading-out, or a virus being cleared. The ability to infer when an absorbing state has been reached is important for calibrating intervention measures (Parag et al., 2020) and understanding immune response (Yan et al., 2016). ODE based models rarely have absorbing states that can be reached in finite time, which fundamentally limits their capacity to describe extinction and clearance processes.

We analysed viral load for SARS-CoV-2 infections using the JSF within a particle filter with a TEIRV, refractory cell model (Ke et al., 2022). In the subset of the data considered, we find consistent parameter values between patients (SI Section 3). In a novel analysis, we estimated the probability of viral clearance through time, finding substantial heterogeneity in the time until viral clearance (see Figure 5A). This is important as an accurate quantification of when the virus has been cleared is crucial for determining appropriate treatment regimes.

A key advantage of JSF is the ability to combine the stochasticity of discrete small populations with convenient deterministic models for large populations. The point at which a compartment will transition between these descriptions is specified by the threshold parameters  $\Omega_i$  (one for each compartment). Setting these parameters to low values speeds up computation, while setting them higher captures more of the stochasticity in the process. As demonstrated in the simulation studies, an appropriate value can be determined through some preliminary simulations (see SI 4 for an example of preliminary simulation analysis for the SIRS model example). An important consideration in selecting this parameter is ensuring that the absorbing states can still be reached (e.g. compartment extinction), and that stable and steady states can be perturbed by random fluctuations. Potential considerations when specifying the values  $\Omega_i$  may also include preliminary simulations, increasing  $\Omega_i$  and observing how the computational complexity scales (see Figure SI 10). Furthermore, developing theory to better understand when a CTMC is well approximated by an ODE is at the core to determine how the switching threshold should be chosen. Understanding into the numerical properties of the Jump-Switch-Flow process, and how convergence of the method behaves for arbitrary choices of Flow event solvers will also prove to be an exciting area of further research. However, these lies outside the scope of this work.

Extensions include the incorporation of an intermittent Stochastic Differential Equation approximation between the CTMC and ODE regimes, and extension to spatial processes which exhibit both stochastic and deterministic behaviours.

The modelling framework presented in this paper has the potential to change the way compartmental models are developed and calibrated, moving us towards more accurate and more efficient hybrid methods. These models will help to form the basis of informed decision making, based on realistic and accurate descriptions of the system. This has broad applicability, from ecological models, chemical systems and single cell models, to infectious diseases and within-host models.

## Acknowledgement

We thank Ada Yan for helpful discussions regarding within-host modelling. J.A.F.’s research is supported by the Australian Research Council (FT210100034, CE230100001) and the National Health and Medical Research Council (APP2019093). S.H.’s research is supported by the Australian Research Council (DP200101281).

## Data Availability

The manuscript has associated data in a data repository available at <https://github.com/DGermano8/JSFGermano2024>. Original data related to SARS-CoV-2 viral load measurements can also be found at Ke et al., 2022.

## Code Availability

All code to reproduce our results is available at <https://github.com/DGermano8/JSFGermano2024>. A Python package implementing the Jump-Switch-Flow method is available at <https://DGermano8.github.io/JSF>.

## Author Contributions

J.A.F and M.B.F conceptualised research; D.P.J.G, A.E.Z, S.H., R.M, J.A.F and M.B.F designed research; D.P.J.G and A.E.Z performed research; D.P.J.G, A.E.Z and R.M contributed software tools; D.P.J.G and A.E.Z analyzed data; J.A.F and M.B.F supervised research; D.P.J.G and A.E.Z wrote the paper; D.P.J.G, A.E.Z, S.H., R.M, J.A.F and M.B.F reviewed the paper; J.A.F acquired funding.

## References

- Alahmadi, A.A., J.A. Flegg, D.G. Cochrane, C.C. Drovandi, and J.M. Keith (2020). “A comparison of approximate versus exact techniques for Bayesian parameter inference in nonlinear ordinary differential equation models”. In: *Royal Society Open Science* 7.3, p. 191315. doi: 10.1098/rsos.191315.

- Anderson, R.M. and R.M. May (1991). *Infectious diseases of humans: dynamics and control*. Oxford University Press.
- Angius, A., G. Balbo, M. Beccuti, E. Bibbona, A. Horvath, and R. Sirovich (2015). “Approximate analysis of biological systems by hybrid switching jump diffusion”. In: *Theoretical Computer Science* 587, pp. 49–72. DOI: 10.1016/j.tcs.2015.03.015.
- Arulampalam, M.S., S. Maskell, N. Gordon, and T. Clapp (2002). “A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking”. In: *IEEE Transactions on Signal Processing* 50.2, pp. 174–188. DOI: 10.1109/78.978374.
- Baccam, P., C. Beauchemin, C.A. Macken, F.G. Hayden, and A.S. Perelson (2006). “Kinetics of Influenza A Virus Infection in Humans”. In: *Journal of Virology* 80.15, pp. 7590–7599. DOI: 10.1128/jvi.01623-05.
- Blanco-Melo, D., B.E. Nilsson-Payant, W.C. Liu, S. Uhl, D. Hoagland, R. Møller, T.X. Jordan, K. Oishi, M. Panis, D. Sachs, T.T. Wang, R.E. Schwartz, J.K. Lim, R.A. Albrecht, and B.R. tenOever (2020). “Imbalanced Host Response to SARS-CoV-2 Drives Development of COVID-19”. In: *Cell* 181.5, 1036–1045.e9. DOI: 10.1016/j.cell.2020.04.026.
- Bressloff, P.C. and J.M. Newby (2014). “Path integrals and large deviations in stochastic hybrid systems”. In: *Physical Review E* 89.4, p. 042701.
- Buckwar, E. and M.G. Riedler (2011). “Runge–Kutta methods for jump-diffusion differential equations”. In: *Journal of Computational and Applied Mathematics* 236.6, pp. 1155–1182. DOI: 10.1016/j.cam.2011.08.001.
- Cao, Y., D.T. Gillespie, and L.R. Petzold (2006). “Efficient step size selection for the tau-leaping simulation method”. In: *The Journal of Chemical Physics* 124.4, p. 044109. DOI: 10.1063/1.2159468.
- Cotter, S.L. and R. Erban (2016). “Error Analysis of Diffusion Approximation Methods for Multiscale Systems in Reaction Kinetics”. In: *SIAM Journal on Scientific Computing* 38.1, B144–B163. DOI: 10.1137/14100052X.
- Diekmann, O., J.A.P. Heesterbeek, and M.G. Roberts (2010). “The construction of next-generation matrices for compartmental epidemic models”. In: *Journal of The Royal Society Interface* 7.47, pp. 873–885. DOI: 10.1098/rsif.2009.0386.
- Farrukee, R., A.E. Zarebski, J.M. McCaw, J.D. Bloom, P.C. Reading, and A.C. Hurt (2018). “Characterization of Influenza B Virus Variants with Reduced Neuraminidase Inhibitor Susceptibility”. In: *Antimicrobial Agents and Chemotherapy* 62.11, 10.1128/aac.01081–18. DOI: 10.1128/aac.01081–18.
- Flegg, M.B., S.J. Chapman, L. Zheng, and R. Erban (2014). “Analysis of the Two-Regime Method on Square Meshes”. In: *SIAM Journal on Scientific Computing* 36.3, B561–B588. DOI: 10.1137/130915844.
- Fowler, A.C. (2021). “Atto-Foxes and Other Minutiae”. In: *Bulletin of Mathematical Biology* 83.10, p. 104. DOI: 10.1007/s11538-021-00936-x.
- Gibson, M.A. and J. Bruck (2000). “Efficient Exact Stochastic Simulation of Chemical Systems with Many Species and Many Channels”. In: *The Journal of Physical Chemistry A* 104.9, pp. 1876–1889. DOI: 10.1021/jp993732q.
- Gillespie, D.T. (1976). “A general method for numerically simulating the stochastic time evolution of coupled chemical reactions”. In: *Journal of Computational Physics* 22.4, pp. 403–434. DOI: 10.1016/0021-9991(76)90041-3.
- (2000). “The chemical Langevin equation”. In: *The Journal of Chemical Physics* 113.1, pp. 297–306. DOI: 10.1063/1.481811.
- (2001). “Approximate accelerated stochastic simulation of chemically reacting systems”. In: *The Journal of chemical physics* 115.4, pp. 1716–1733.
- Gubbins, S. (2024). “Quantifying the relationship between within-host dynamics and transmission for viral diseases of livestock”. In: *Journal of the Royal Society Interface* 21.211, p. 20230445. DOI: 10.1098/rsif.2023.0445.
- Hernandez-Vargas, E.A. and J.X. Velasco-Hernandez (2020). “In-host Mathematical Modelling of COVID-19 in Humans”. In: *Annual Reviews in Control* 50, pp. 448–456. DOI: 10.1016/j.arcontrol.2020.09.006.
- Ionides, E.L., C. Bretó, and A.A. King (2006). “Inference for nonlinear dynamical systems”. In: *Proceedings of the National Academy of Sciences* 103.49, pp. 18438–18443. DOI: 10.1073/pnas.0603181103.
- Isaacson, S.A. (2013). “A convergent reaction-diffusion master equation”. In: *The Journal of Chemical Physics* 139.5, p. 054101. DOI: 10.1063/1.4816377.
- Ke, R., P.P. Martinez, R.L. Smith, L.L. Gibson, A. Mirza, M. Conte, N. Gallagher, CH. Luo, J. Jarrett, R. Zhou, A. Conte, T. Liu, M. Farjo, K.K.O. Walden, G. Rendon, C.J. Fields, L. Wang, R. Fredrickson, D.C. Edmonson, M.E. Baughman, K.K. Chiu, H. Choi, K.R. Scardina, S. Bradley, S.L. Gloss, C. Reinhart, J. Yedetore, J. Quicksall, A.N. Owens, J. Broach, B. Barton, P. Lazar, W.J. Heetderks, M.L. Robinson, H.H. Mostafa, Y.C. Manabe, A. Pekosz, D.D. McManus, and C.B. Brooke (2022). “Daily longitudinal sampling of SARS-CoV-2 infection reveals substantial heterogeneity in infectiousness”. In: *Nature Microbiology* 7.5, pp. 640–652. DOI: 10.1038/s41564-022-01105-z.
- Kitagawa, G. (1996). “Monte Carlo Filter and Smoother for Non-Gaussian Nonlinear State Space Models”. In: *Journal of Computational and Graphical Statistics* 5.1, pp. 1–25. DOI: 10.1080/10618600.1996.10474692.
- Klein, R.W. and S.D. Roberts (1984). “A time-varying Poisson arrival process generator”. In: *Simulation* 43.4, pp. 193–195. DOI: 10.1177/003754978404300406.
- Kreger, J., N.L. Komarova, and D. Wodarz (2021). “A hybrid stochastic-deterministic approach to explore multiple infection and evolution in HIV”. In: *PLOS Computational Biology* 17.12, e1009713.
- Kurtz, T.G. (1970). “Solutions of ordinary differential equations as limits of pure jump Markov processes”. In: *Journal of Applied Probability* 7.1, pp. 49–58. DOI: 10.2307/3212147.

- Kurtz, T.G. (1971). "Limit theorems for sequences of jump Markov processes approximating ordinary differential processes". In: *Journal of Applied Probability* 8.2, pp. 344–356. doi: 10.2307/3211904.
- (1972). "The Relationship between Stochastic and Deterministic Models for Chemical Reactions". In: *The Journal of Chemical Physics* 57.7, pp. 2976–2978. doi: 10.1063/1.1678692.
- Kynaston, J.C., C.A. Yates, A.V.F. Hekkink, and C. Guiver (2023). "The regime-conversion method: a hybrid technique for simulating well-mixed chemical reaction networks". In: *Frontiers in Applied Mathematics and Statistics* 9. doi: 10.3389/fams.2023.1107441.
- Lobry, C. and T. Sari (2015). "Migrations in the Rosenzweig-MacArthur model and the “atto-fox” problem". In: *Revue Africaine de la Recherche en Informatique et Mathématiques Appliquées* 20, pp. 95–125.
- Ludwig, D., D.D. Jones, and C.S. Holling (1978). "Qualitative analysis of insect outbreak systems: the spruce budworm and forest". In: *Journal of Animal Ecology* 47.1, pp. 315–332.
- Matthew, S., F. Carter, J. Cooper, M. Dippel, E. Green, S. Hodges, M. Kidwell, D. Nickerson, B. Rumsey, J. Reeve, L.R. Petzold, K.R. Sanft, and B. Drawert (2023). "GillesPy2: a biochemical modeling framework for simulation driven biological discovery". In: *Letters in biomathematics* 10.1, p. 87.
- Mollison, D. (1991). "Dependence of epidemic and population velocities on basic parameters". In: *Mathematical Biosciences* 107.2, pp. 255–287. doi: 10.1016/0025-5564(91)90009-8.
- Moss, R. (2024). "pypfilt: a particle filter for Python". In: *Journal of Open Source Software* 9.96, p. 6276. doi: 10.21105/joss.06276. URL: 10.21105/joss.06276.
- Parag, Kris V., Christl A. Donnelly, Rahul Jha, and Robin N. Thompson (Nov. 2020). "An exact method for quantifying the reliability of end-of-epidemic declarations in real time". In: *PLOS Computational Biology* 16.11, pp. 1–21. doi: 10.1371/journal.pcbi.1008478.
- Pearson, J.E., P. Krapivsky, and A.S. Perelson (2011). "Stochastic Theory of Early Viral Infection: Continuous versus Burst Production of Virions". In: *PLOS Computational Biology* 7.2, pp. 1–17. doi: 10.1371/journal.pcbi.1001058.
- Perelson, A.S. (2002). "Modelling viral and immune system dynamics". In: *Nature Reviews Immunology* 2.1, pp. 28–36. doi: 10.1038/nri700.
- Rao, C.V. and A.P. Arkin (2003). "Stochastic chemical kinetics and the quasi-steady-state assumption: Application to the Gillespie algorithm". In: *The Journal of Chemical Physics* 118.11, pp. 4999–5010. doi: 10.1063/1.1545446.
- Rebuli, N.P., N.G. Bean, and J.V. Ross (2017). "Hybrid Markov chain models of S-I-R disease dynamics". In: *Journal of Mathematical Biology* 75, pp. 521–541.
- Sanft, K.R. and H.G. Othmer (2015). "Constant-complexity stochastic simulation algorithm with optimal binning". In: *The Journal of Chemical Physics* 143.7, p. 074108. doi: 10.1063/1.4928635.
- Simoni, G., F. Reali, C. Priami, and L. Marchetti (2019). "Stochastic simulation algorithms for computational systems biology: Exact, approximate, and hybrid methods". In: *WIREs Systems Biology and Medicine* 11.6, e1459. doi: 10.1002/wsbm.1459.
- Trindade, T.T. and K.C. Zygalakis (2024). "A hybrid tau-leap for simulating chemical kinetics with applications to parameter estimation". In: *Royal Society Open Science* 11.12, p. 240157. doi: 10.1098/rsos.240157.
- Yan, A.W.C., P. Cao, and J.M. McCaw (2016). "On the extinction probability in models of within-host infection: the role of latency and immunity". In: *Journal of Mathematical Biology* 73.4, pp. 787–813. doi: 10.1007/s00285-015-0961-5.

# Supplementary Information: A hybrid framework for compartmental models enabling simulation-based inference

Domenic P.J. Germano <sup>a,b,\*</sup>, Alexander E. Zarebski <sup>b,c,\*</sup>, Sophie Hautphenne <sup>a</sup>, Robert Moss <sup>d</sup>, Jennifer A. Flegg <sup>a</sup>, and Mark B. Flegg <sup>e</sup>

<sup>a</sup> The School of Mathematics and Statistics, The University of Melbourne, Parkville, VIC, Australia

<sup>b</sup> The School of Mathematics and Statistics, The University of Sydney, Camperdown, NSW, Australia

<sup>c</sup> Pandemic Sciences Institute, University of Oxford, Oxford, United Kingdom

<sup>d</sup> Melbourne School of Population and Global Health, The University of Melbourne, Parkville, Vic, Australia

<sup>e</sup> School of Mathematics, Monash University, Clayton, VIC, Australia

## SI1 Jump-Switch-Flow: implementation details

To draw exact samples from the JSF process requires solving the differential equations for the flowing variables  $\vec{V}_F$ . For nonlinear systems these differential equations are usually intractable, so they are numerically integrated. Here we describe an algorithm for approximately sampling trajectories, assuming we have a numerical solver for the differential equations (when viewed as an initial value problem (IVP)), to approximate the solutions.

### SI1.1 Jump-clock

We implement the JSF algorithm with a slightly different jump clock as defined by  $J_k$  in Section . In particular, equate  $u_k = \text{CDF}(t_k; k)$  but do not take the log of this expression before defining  $J_k$  in Equation (11). As a result, in our codes we consider the jump clock  $\tilde{J}_k$ :

$$\tilde{J}_k(t) = u_k - \left(1 - \exp\left\{-\int_{t_0}^t \lambda_k(\vec{V}(s)) \, ds\right\}\right), \quad (\text{SI1})$$

where  $u_k \sim \text{Unif}(0, 1)$  and the time of the previous jump event  $t_0$ . Computing the times for the jump events follows a similar process as above. We compute the jump clock on the regular ODE mesh used to solve  $\vec{V}_F$  in Equation (6). From Equation (SI1) we can write  $\tilde{J}_k(t + \Delta t)$  in terms of  $\tilde{J}_k(t)$  and an integral of the reaction rate:

$$\tilde{J}_k(t + \Delta t) = \tilde{J}_k(t) + \left(\exp\left\{-\int_t^{t+\Delta t} \lambda_k(\vec{V}(s)) \, ds\right\} - 1\right) (\tilde{J}_k(t) + 1 - u_k). \quad (\text{SI2})$$

For steps where  $\tilde{J}_k(t + \Delta t) > 0$  we can do a single step of this process to get the updated jump clock values. However, if  $\tilde{J}_k(t + \Delta t) < 0$  a jump has occurred at some time  $t + \Delta\tau \in (t, t + \Delta t)$ , and we need to solve  $\tilde{J}_k(t + \Delta\tau) = 0$  to find when the jump occurred. Therefore, we require a method to find  $\Delta\tau$ , where  $0 < \Delta\tau < \Delta t$ . To do this, we start by letting the right hand side of Equation (SI2) equal zero and rearrange to get:

$$\int_t^{t+\Delta\tau} \lambda_k(\vec{V}(s)) \, ds = \ln\left\{\frac{\tilde{J}_k(t) + 1 - u_k}{1 - u_k}\right\}. \quad (\text{SI3})$$

If  $\lambda_k$  is a positive constant then:

$$\Delta\tau = \frac{1}{\lambda_k} \ln\left\{\frac{\tilde{J}_k(t) + 1 - u_k}{1 - u_k}\right\}. \quad (\text{SI4})$$

If  $\lambda_k$  is not constant, we approximate  $\lambda_k$ , which we call  $\hat{\lambda}_k$ , using the integration of  $\vec{V}(t)$ , as per Equation (6), since we know  $\vec{V}(t)$  and  $\vec{V}(t + \Delta t)$ . Therefore, we also know  $\lambda_k$  at  $t$  and  $t + \Delta t$ . We then write  $\hat{\lambda}_k$ , at  $t + \Delta\tau$  between times  $t$  and  $t + \Delta t$  via linear interpolation:

$$\hat{\lambda}_k(t + \Delta\tau) \approx \left(1 - \frac{\Delta\tau}{\Delta t}\right) \lambda_k(\vec{V}(t)) + \frac{\Delta\tau}{\Delta t} \lambda_k(\vec{V}(t + \Delta t)). \quad (\text{SI5})$$

---

\*These authors contributed equally to this work

We can now substitute this approximation into Equation (SI3), solve it, and rearrange for  $\Delta\tau$ . This gives a quadratic in  $\Delta\tau$  with the following solution:

$$\Delta\tau = \frac{\sqrt{\left(\Delta t \hat{\lambda}_k(t)\right)^2 + 2\alpha\Delta t \Delta\lambda_k} - \Delta t \hat{\lambda}_k(t)}{\Delta\lambda_k}, \quad (\text{SI6})$$

where we have introduced the shorthand  $\Delta\lambda_k = \lambda_k(\vec{V}(t + \Delta t)) - \lambda_k(\vec{V}(t))$ .

## SI1.2 Pseudocode details

As with a typical IVP, we require the following data:

- the initial condition of the system,  $\vec{V}(0)$ ;
- the final time  $T_{\max} > 0$ ;
- the stoichiometric Reactant,  $\eta^-$ , and Product,  $\eta^+$ , matrices, from which we derive the Exchange matrix,  $\eta = \eta^+ - \eta^-$ ;
- a set of the associated propensities of the reactions,  $\mathcal{R}$ ;
- a time step size for the numerical integrator,  $\Delta t$ ;
- a vector of switching thresholds,  $\Omega$ ;

To sample the Jump-Switch-Flow process, we have identified both an *exact* sampler, that samples the process exactly, and an *operator-splitting* sampler, that uses some simplifying approximations that makes sampling substantially faster.

### Exact Jump-Switch-Flow sampler

Here, we describe an *exact* algorithm for sampling trajectories from Jump-Switch-Flow. The algorithm to sample *exactly* from Jump-Switch-Flow is given in Algorithm SI1. First, initialise the jump clocks (line 3). On each iteration through the ODE loop, we first partition the reactions based on inclusion in  $\mathcal{S}$  (line 6). We use the IVP solver to compute  $\Delta\vec{V}_F(t)$  based on the flowing reactions (line 7). For each step of the ODE loop, we update the jump clocks (line 9). On lines 10–12, we determine if a jump event has occurred based on the sign of the the Jump-clock. If a jump event has occurred, compute the corresponding event time. On line 13 we then rewind the state (and jump clocks) to when the jump occurred. Resample the appropriate jump clock (line 14), and update the state to account for the jump (line 15). Repeat until the time has reached the maximum desired time.

---

**Algorithm SI1** An algorithmic description of the Exact Jump-Switch-Flow sampler.

---

**Require:**  $\vec{V}(0)$ ,  $\Delta t > 0$  and  $T_{\max} > 0$

- 1: Initialise model state
- 2:  $t \leftarrow 0$
- 3: Initialise jump clocks  $u_i \sim \text{Unif}(0, 1)$  for  $i = 1, \dots, N$
- 4: ( $\triangleright$  Start ODE loop)
- 5: **while**  $t < T_{\max}$  **do**
- 6:   Compute Jumping reactions  $\mathcal{S}(t)$
- 7:   Compute flow event,  $\Delta\vec{V}_F(t)$
- 8:   ( $\triangleright$  Perform jump loop for ODE mesh step)
- 9:   Update jump clocks,  $\tilde{J}_k$ , to  $t + \Delta t$  ( $\triangleright$  As per Equation (14))
- 10:   **if** any  $\tilde{J}_k \leq 0$  ( $\triangleright$  Reaction Occurred) **then**
- 11:     Identify first fired Reaction,  $j$
- 12:     Compute time of jump event,  $\Delta\tau$  ( $\triangleright$  As per Equation (16))
- 13:     Reverse jump clocks to time  $t + \Delta\tau$
- 14:     Resample  $u_j \sim \text{Unif}(0, 1)$
- 15:     Update both:  $\vec{V}_J(t + \Delta\tau) = \vec{V}_J(t) + \eta_j$  **and**  $\vec{V}_F(t + \Delta\tau) = \vec{V}_F(t) + \Delta\tau \Delta\vec{V}_F(t) + \eta_j$
- 16:      $t \leftarrow t + \Delta\tau$
- 17:   **else if**  $\tilde{J}_k > 0, \forall k$  **then**
- 18:     Update:  $\vec{V}_F(t + \Delta t) = \vec{V}_F(t) + \Delta t \Delta\vec{V}_F(t)$
- 19:      $t \leftarrow t + \Delta t$
- 20:   **end if**
- 21: **end while**

---

## Operator Splitting Jump-Switch-Flow sampler

Here, we describe the algorithm for the *operator-splitting* algorithm, for sampling trajectories from Jump-Switch-Flow, which utilises some simplifying approximations to improve sampling performance.

The Operator Splitting Jump-Switch-Flow sampler is described in Algorithm SI2. Unlike the Exact JSF sampler, when a jump event occurs, we continue to integrate the system with the same flow event,  $\Delta\vec{V}_F(t)$ , instead of resampling it as before. This, in turn, ensures that the ODE is not continually being resampled. While this adds complexity at the implementation stage, significant computational time saving is achieved.

We first initialise the Jump-Switch-Flow process by initialising all the jump clocks (line 3). On each iteration through the ODE loop, we first partition the reactions based on inclusion in  $\mathcal{S}$  (line 6). We use the IVP solver to compute  $\Delta\vec{V}_F(t)$  based on the flowing reactions (line 7). We now iterate through this current time-step,  $\Delta t$ , checking if any jump events occur (lines 10 – 24), named the *jump loop*. First, we track how much “relative time” ( $\delta t$ ) passes through this jump loop (line 9). We then update the jump clocks to the end of the jump loop, accounting for any relative time that has occurred (line 11). As with the Exact JSF sampler, on lines 12 – 14, we determine if a jump event has occurred using the Jump-clock. This gives us which Reaction has occurred, and at what time,  $t + \Delta\tau$ . On line 15 we then reverse the system to the time the jump event occurs, noting that we must reverse the excess between when the jump event occurred and the remainder of the time-step. We then resample the associated jump clock that just fired (line 16), and update the whole state (i.e. both flowing and Jumping variables) to the current time (line 17). We update the current time to time  $t + \Delta\tau$  (line 18) and track the relative time in the jump loop (line 19). We then continue through the jump look. If no jump events have occurred, we leave the loop (line 21). We then simply update the whole state according to the flow event calculated, accounting for any time spent inside the jump loop (line 24).

---

### Algorithm SI2 An algorithmic description of the Operator Splitting Jump-Switch-Flow sampler .

---

**Require:**  $\vec{V}(0)$ ,  $\Delta t > 0$  and  $T_{\max} > 0$

- 1: Initialise model state
- 2:  $t \leftarrow 0$
- 3: Initialise jump clocks  $u_i \sim \text{Unif}(0, 1)$  for  $i = 1, \dots, N$
- 4: ( $\triangleright$  Start ODE loop)
- 5: **while**  $t < T_{\max}$  **do**
- 6:   Compute jumping reactions  $\mathcal{S}(t)$
- 7:   Compute flow event,  $\Delta\vec{V}_F(t)$
- 8:   ( $\triangleright$  Perform jump loop for ODE mesh step)
- 9:   Set  $\delta t \leftarrow 0$
- 10:   **while**  $\Delta t > \delta t$  **do**
- 11:     Update jump clocks,  $\tilde{J}_k$ , to  $t + (\Delta t - \delta t)$
- 12:     **if** any  $\tilde{J}_k \leq 0$  ( $\triangleright$  Reaction Occurred) **then**
- 13:       Identify first fired Reaction,  $j$
- 14:       Compute time of jump event,  $t + \Delta\tau$
- 15:       Reverse jump clocks to time  $t + \Delta\tau$
- 16:       Resample  $u_j \sim \text{Unif}(0, 1)$
- 17:       Update both:  $\vec{V}_J(t + \Delta\tau) = \vec{V}_J(t) + \eta_j$  **and**  $\vec{V}_F(t + \Delta\tau) = \vec{V}_F(t) + \Delta\tau \Delta\vec{V}_F(t) + \eta_j$
- 18:        $t \leftarrow t + \Delta\tau$
- 19:        $\delta t \leftarrow \delta t + \Delta\tau$
- 20:     **else if**  $\tilde{J}_k > 0, \forall k$  **then**
- 21:       Leave while loop
- 22:     **end if**
- 23:   **end while**
- 24:   Update:  $\vec{V}_F(t + (\Delta t - \delta t)) = \vec{V}_F(t) + (\Delta t - \delta t) \Delta\vec{V}_F(t)$
- 25:    $t \leftarrow t + (\Delta t - \delta t)$
- 26: **end while**

---

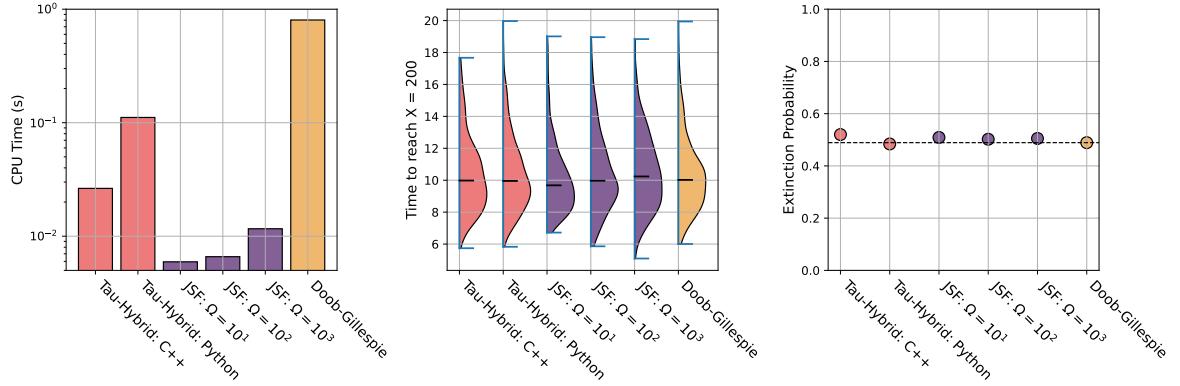
## SI2 Simulation Study: Birth-death processes

Fundamentally, one can conceptualise any compartmental model as a collection of interacting birth-death processes. For this reason, we will consider two very simple such processes: 1) a birth dominated birth-death process, and 2) a death dominated birth-death process. In this section, we will perform a simulation study to highlight the advantages of our Jump-Switch-Flow method, and compare the results to the gold standard, Doob-Gillespie Exact method, and the highly computationally efficient Tau-hybrid method, which is provided by GillesPy2, written in both C++ and Python.

The exact model we will utilise is a single species,  $X$ , experiencing birth with rate  $\alpha$ , which results in  $X \rightarrow 2X$ , and death with rate  $\beta$ , which results in  $X \rightarrow \emptyset$ .

## A birth dominated birth-death process

For the birth dominated process, we require  $\alpha > \beta$ . Therefore, we specify  $\alpha = 1$ , and  $\beta = 0.5$ . We also choose the initial species population  $X(0) = 1$ , and a simulation time of  $t = 20$ . The quantities of interest to test the accuracy and efficiency of the methods are (i) the CPU time for a particular simulation instance, (ii) the simulation time required to reach a particular value,  $X = 200$ , and (iii) the accuracy of the method to experience initial population extinction. Figure SI1 shows results for each of the three quantities of interest for the Tau-hybrid method (red), Doo-Gillespie (yellow) and three different Jump-Switch-Flow examples (purple) with varying switching thresholds of  $\Omega = 10^1$ ,  $\Omega = 10^2$ , and  $\Omega = 10^3$ , respectively. Here, 1000 unique trajectories are considered for each method.



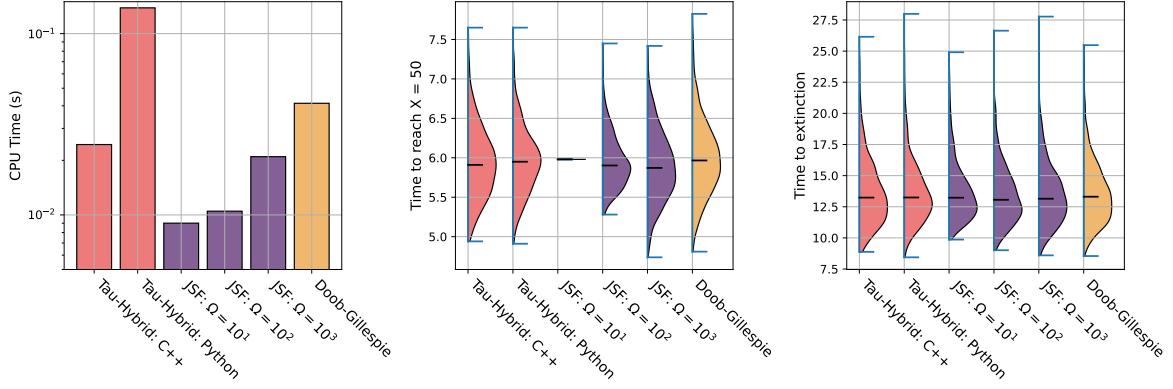
**Figure SI1:** Summary statistics for the birth dominated birth-death process for the Tau-hybrid (red) JSF for various switching thresholds,  $\Omega$ , (purple) and the Doob-Gillespie (yellow) methods. Left: CPU time required. We observe that as  $\Omega$  increases, the computational time increases, but remains the computationally least expensive. Middle: Distributions in time to reach  $X = 200$ . We observe that JSF results in comparable distributions to the exact process, with diminishing returns for  $\Omega > 10^2$ . Right: Extinction probability. We observe that all point estimates are comparable to that of the exact method.

Considering the CPU time, we can observe as we increase the switching threshold, JSF requires more CPU time, since more of the simulation is in a discrete, Jumping regime. We also observe that the CPU time is, at worst, around two orders of magnitude less, compared to that of Doob-Gillespie. In comparison, we observe that the C++ implementation of Tau-hybrid method requires at best three times the computational time to simulate the equivalent process as JSF, while the Python implementation is at least one order of magnitude slower, indicating that JSF is much more efficient. To assess the accuracy of each method, we now compare the (simulation) time for the species to reach 200. Here, both Tau-hybrid and JSF with switching thresholds of  $\Omega = 10^2$ , and  $\Omega = 10^3$  result in distributions and median value comparable to that of Doob-Gillespie. The only exception is that of JSF with a switching threshold of  $\Omega = 10^1$ . Lastly, we observe that all methods result in an initial extinction probability comparable to that of Doob-Gillespie.

These results suggest that JSF is capable of recovering the key quantities of the exact stochastic method, while being computationally cheaper than current hybrid methods and the exact method, for the birth dominated birth-death process.

## A death dominated birth-death process

For the birth dominated process, we require  $\alpha < \beta$ . Therefore, we specify  $\alpha = 0.5$ , and  $\beta = 1$ . We choose the initial species population  $X(0) = 100$ , and a simulation time of  $t = 30$ . As before, the quantities of interest to test the accuracy and efficiency of the methods are (i) the CPU time for a particular simulation instance, (ii) the simulation time required to reach a particular value,  $X = 50$ , and (iii) the simulation time require for the species to go extinct. Figure SI2 shows results for each of the three quantities of interest for the Tau-hybrid method (red), Doo-Gillespie (yellow) and three different Jump-Switch-Flow examples (purple) with varying switching thresholds of  $\Omega = 10^1$ ,  $\Omega = 10^2$ , and  $\Omega = 10^3$ , respectively. Again, 1000 unique trajectories are considered for each method. Unlike in the previous example, here we observe that all JSF examples require more CPU time to complete the equivalent computation, despite this being a decaying death process. However, we still observe that all JSF methods require less CPU time when compared to the C++ implementation of the Tau-hybrid method, this time consuming at least 1.5 times less computation time, and still at least one order of magnitude quicker when compared to the Python implementation. We also note that JSF requires at most half the computational time of that of the exact Doob-Gillespie method. We now consider the time for the species to reach a population of size  $X = 50$ . In this example, we see that JSF with  $\Omega = 10^1$  does not capture the distribution of the exact method, but only reports a single value. This is due to trajectories from this example all being deterministic at this value. If we consider JSF with  $\Omega = 10^2$ , we see that while the median is comparable to the exact method, the distribution is not as broad. This is because the method only becomes stochastic at  $X = 100$ , resulting in variance in the trajectory prior to this value being lost. As we increase the switching threshold to  $\Omega = 10^3$ , we observe that distribution in the Time to reach  $X = 50$  become comparable to that of the exact method. This provides a clear guide for how the switching threshold may be chosen, given some prior knowledge of the system. Lastly, we observe that extinction time for the JSF with  $\Omega = 10^2$  and  $\Omega = 10^3$  resulting in a



**Figure SI2:** Summary statistics for the death dominated birth-death process for the Tau-hybrid (red) JSF for various switching thresholds,  $\Omega$ , (purple) and the Doob-Gillespie (yellow) methods. Left: CPU time required. We again observe that as  $\Omega$  increases, the computational time increases, but still remains the computationally least expensive. Middle: Distributions in time to reach  $X = 50$ . We observe that a switching threshold of  $\Omega = 10^3$  is required for JSF to result in comparable distributions to the exact process, Right: Distribution in the time to extinction. We observe that for  $\Omega > 10^1$ , JSF results in comparable distributions to that of the exact method.

comparable distribution to that of the Doob-Gillespie method. Again, we also observe that JSF with  $\Omega = 10^1$  results in a truncated distribution to that of the exact method.

These results also suggest that JSF is capable of recovering the key quantities of the exact stochastic method, while being computationally cheaper than current hybrid methods, for the death dominated birth-death process. Through this example, we also show how JSF may exhibit sensitivity in the value of the switching threshold chosen, and how to circumvent these sensitivities.

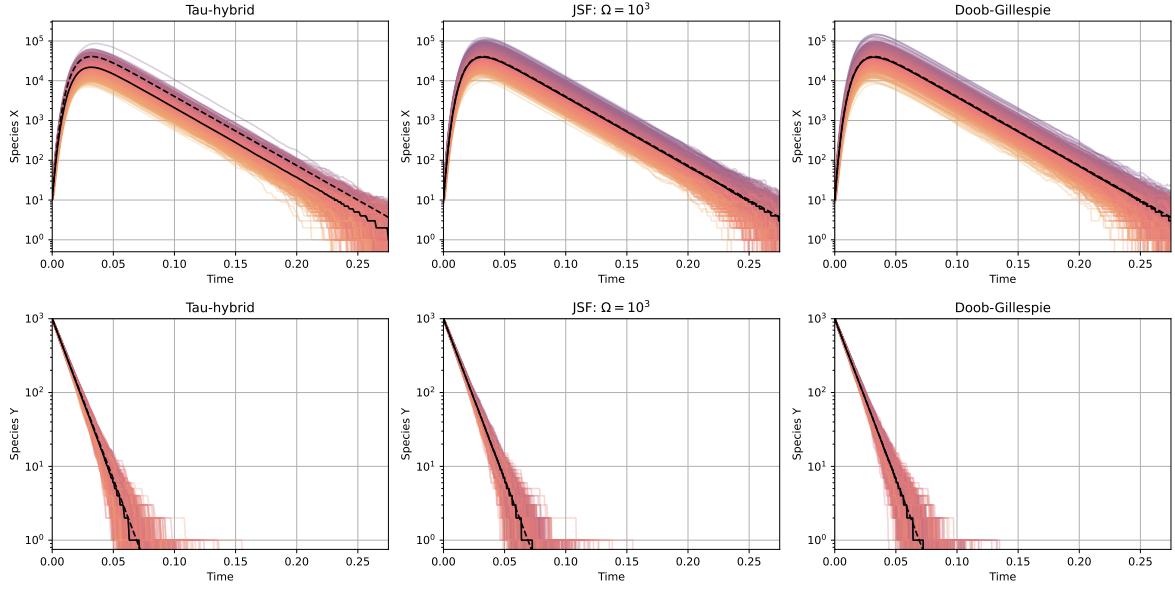
### S13 Simulation study: Multiple time-scales in autocatalysis

To emphasise the utility and accuracy of our Jump-Switch-Flow method, we will consider a simple autocatalysis example, with multiple time-scales. Consider two species,  $X$ , and  $Y$ , which react together via the following reactions:  $X + Y \rightarrow 2X + Y$ , with rate  $\alpha$ ,  $X \rightarrow 2X$ , with rate  $\beta$ ,  $X \rightarrow \emptyset$ , with rate  $\gamma$ , and  $Y \rightarrow \emptyset$  with rate  $\delta$ . To demonstrate the ability of JSF to simulate the trajectories with multiple time-scales, but importantly, simulate them *accurately*, we choose the following rate parameters:

1. a moderate growth rate of  $X$ :  $\beta = 10$ ,
2. a large death rate of  $Y$ , to ensure we cannot assume a constant  $Y$ :  $\delta = 100$ ,
3. a large death rate of  $X$ , meaning  $X$  quickly vanishes:  $\gamma = 50$ ,
4. and lastly a small autocatalysis rate, resulting in  $X$  varying drastically in scale:  $\alpha = 1$ .

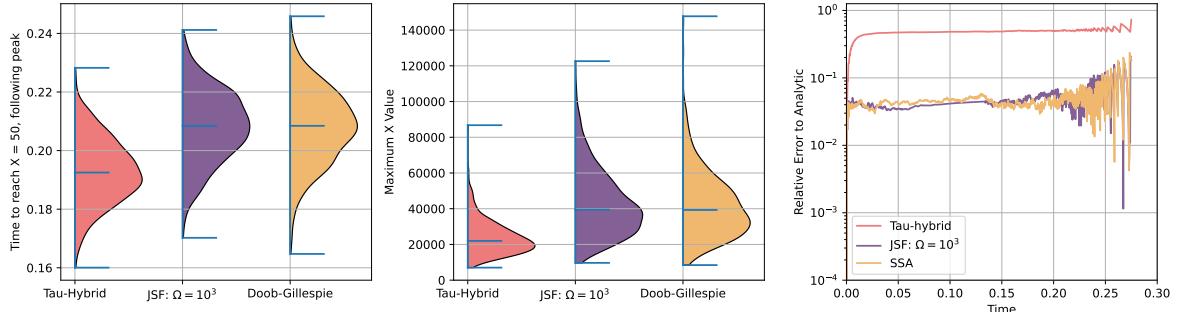
Despite being a relatively simple model, with an analytic deterministic description, simulating stochastic trajectories requires careful consideration of the partitioning between the stochastic reactions to ensure an accurate representation, as there are reactions that are occurring at multiple different timescales. For example, autocatalysis occurs initially relatively frequently, due to the large number of species  $Y$ , despite the small rate  $\alpha$ . However, following the rapid decline of  $Y$ ,  $X$  will experience moderate decay. Therefore, we expect to observe two different timescales in the species  $X$ : a rapid increase, followed by gradual decline.

We assign the initial species sizes  $X(0) = 10$ , and  $Y(0) = 10^2$ , and sample 1000 trajectories using the exact Doob-Gillespie method, GillesPy2's C++ hybrid method (Tau-hybrid) and JSF with a switching threshold of  $\Omega = 10^2$ . Figure SI3 shows the 1000 trajectories for Tau-hybrid, JSF (with  $\Omega = 10^2$ ), and Doob-Gillespie. We also present the analytic ODE solutions (dashed black) and the median value at a point (solid black).



**Figure SI3:** 1000 trajectories for the multiscale autocatalyst example for the Tau-hybrid method (left), the JSF method (middle) and the exact Doob-Gillespie method (right). Top shows species  $X$ , and bottom shows species  $Y$ . We also show the median species value (solid black line) and the analytic ODE solution (dashed black line). We see that the median trajectory of Tau-hybrid is less than the analytic solution, while both Doob-Gillespie and JSF match the analytic solution.

Here, we can see that all  $Y$  species solutions match the analytic solution, since species  $Y$  undergoes constant decay only. However, we clearly see that, while the median trajectories of JSF and Doob-Gillespie match that of the analytic description, the Tau-hybrid method median is less than the analytic solution. We can further quantify these discrepancies by looking at the distributions of the time for species  $X$  to reach a particular value, the distributions of the maximum  $X$  value, and also the relative point error between the median of the particular approach, and the analytic solution. We present these quantities in Figure SI4.



**Figure SI4:** Summary statistics for the multiscale autocatalyst example for the Tau-hybrid (red), JSF with  $\Omega = 10^3$  (purple) and Doob-Gillespie (yellow). Left: time to reach  $X = 50$  following initial peak. We see that the distributions of Doob-Gillespie and JSF are comparable, while tau-hybrid is skewed downward with a different median value. Middle: Maximum  $X$  value. Again, we observe JSF and Doob-Gillespie resulting in matching distributions, while Tau-hybrid is skewed downwards. Right: relative error to analytic solution. JSF and Doob-Gillespie display similarly small error, an order of magnitude less than Tau-hybrid.

Figure SI4 shows the distributions for time to reach  $X = 50$  following the peak, where we see that JSF and Doob-Gillespie produce comparable results. However we also observe that the Tau-hybrid method under-performs, and results in earlier hitting times. We observe a similar trend in the maximum  $X$  value, with Tau-hybrid producing a skewed downwards distribution, and JSF and Doob-Gillespie producing comparable distributions. Lastly, the relative point error of JSF and Doob-Gillespie are comparable, with Tau-hybrid an order of magnitude greater.

From the results presented here, we see that our Jump-Switch-Flow method is capable of producing highly accurate hybrid trajectories, even with highly multiscale systems, as the one presented here. We also observe the summary statistics

produced by JSF being comparable to those of the exact Doob-Gillespie method, and outperforming the highly efficient and sophisticated Tau-hybrid method.

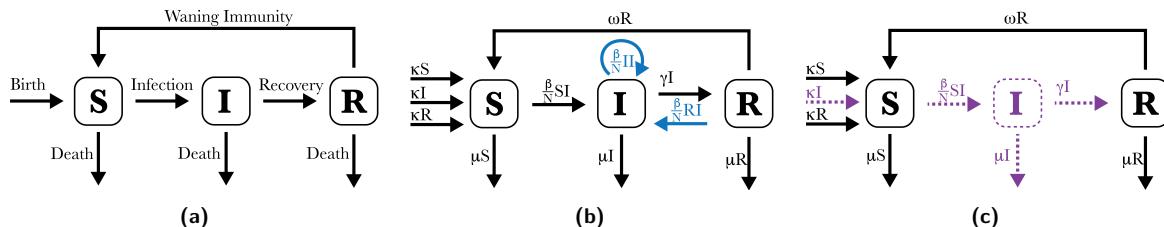
## SI4 Simulation study: SIRS with demography

We first present how the SIRS model with demography can be expressed with the Jump-Switch-Flow method. For the Jump-Switch-Flow method to be a useful method, it must reproduce behaviour that is representative of the exact process (i.e. the CTMC). That is, given some summary statistic, distributions produced via sampling the JSF process should be comparable to those obtained via sampling the CTMC. Moreover, we require that it does so with acceptable computational efficiency. We show, through simulation experiments, how the Jump-Switch-Flow method compares to the Doob-Gillespie method (Gillespie, 1976) for accuracy, and how our approach can exhibit sensitivity with respect to model inputs. We present how our Jump-Switch-Flow method compares to both the Doob-Gillespie and Tau-Leaping methods.

Figure SI5a shows the main reactions of the SIRS model with demography: individuals fall into one of three compartments: susceptible ( $S$ ), infectious ( $I$ ) and recovered ( $R$ ). Here, arrows represent how individuals move and progress through compartments. Specifically, individuals are born into the susceptible compartment. Births occur from individuals in the  $S$ ,  $I$  and  $R$  compartments, and therefore depend explicitly on the populations (discrete or continuous) in each of these compartments. Individuals can die within each of the compartments. Finally, individuals may progress through compartments via susceptible individuals experiencing infection, infected individuals experiencing recovery, and recovered individuals experiencing waning immunity.

In order to represent the SIRS model with demography as a Jump-Switch-Flow process, we need to associate rates, reactants and products to each of the arrows (see Figure SI5b). Note that the births arrow is split into three arrows depending upon the compartment of the parent. Moreover, one of the key assumptions of any SIR-type model is that the population is homogeneously mixed, which enables us to describe how individuals interact with one another. This enables us to say that susceptible individuals interact with infectious individuals with a rate inversely proportional to the whole population. However, there are likewise infectious individuals interacting with other infectious individuals, and also with recovered individuals. However, these last two interactions (as shown in blue in Figure SI5b) do not result in any exchange of individuals between compartments.

Figure SI5c shows the SIRS model with demography as a Jump-Switch-Flow process. In this example figure, the  $S$  and  $R$  compartments are continuous (flowing), while the  $I$  compartment is discrete (jumping). To decide which of the arrows/reactions are jumping, i.e. in  $S$ , and which are flowing, i.e. in  $S'$ , we consider the reactants and products of the arrows: if an arrow has any discrete reactant or product, then that arrow is also jumping, otherwise it is flowing. Table SI1 shows the reactants and products of the SIRS model with demography.



**Figure SI5:** (a) The labelled compartmental SIRS model with demography. (b) The SIRS model with demography where the arrows describe the interactions between the reactants and products. Here, blue arrows do not result in any flow through the system, and can be ignored. (c) The model as a Jump-Switch-Flow system with continuous (flowing)  $S$  and  $R$  (black), and discrete (jumping)  $I$  (purple). Since  $I$  is jumping, the reactions involving  $I$  are modelled as discrete, jumping reactions.

For the remainder of this section, we suppose parameter values for the SIRS model with demography that would be typical for a newly introduced, yearly seasonal communicable disease, as presented in Table SI2. With these parameter values, on average, we expect an infected individual to infect 2 other people per week (during the initial outbreak), and be infectious for 1 week, with immunity to the disease lasting for 1 year. For the population turnover, we assume individuals live for, on average, 85 years. Initially there are two infectious individuals (i.e.  $I(0) = 2$ ), no recovered individuals (i.e.  $R(0) = 0$ ), and the remainder of the population is susceptible (i.e.  $S(0) = N(0) - I(0)$ ).

As well as being computationally efficient, our Jump-Switch-Flow method is capable of capturing the inherent stochastic nature of the sampled process. For the SIRS model with demography, this is realised by three key different scenarios:

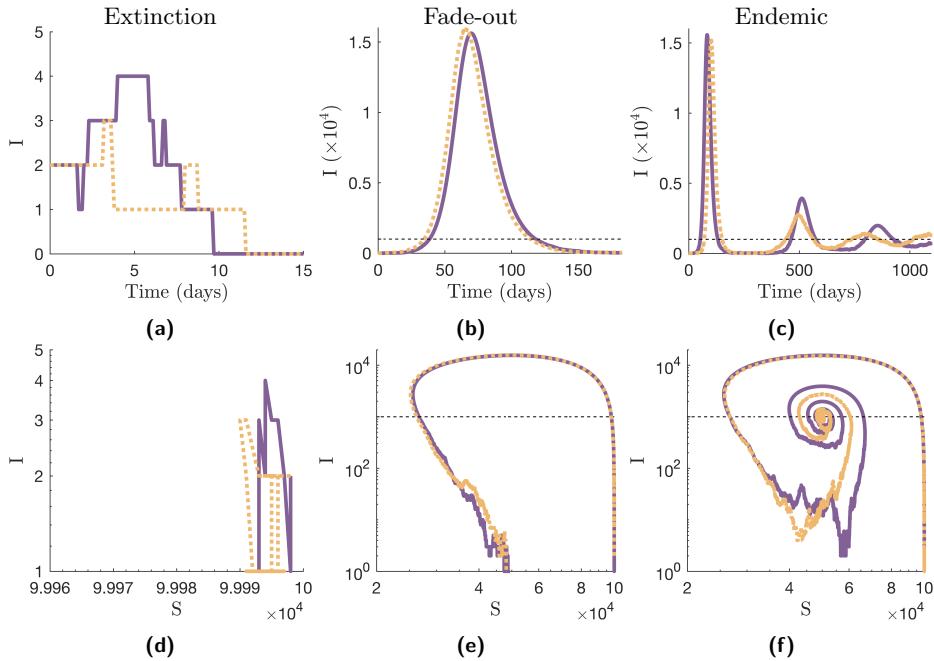
1. *Extinction*: all of the individuals recover before the disease manages to spread significantly throughout the population (see Figure SI6a and SI6d);
2. *Fade-out*: the disease spreads throughout the population, however it fades-out in the trough proceeding the first wave (see Figure SI6b and SI6e);
3. *Endemic*: the disease persists indefinitely within the population (see Figure SI6c and SI6f).

Reaction	Rate	Reactants, $\eta^-$			Products, $\eta^+$			$\eta$		
		S	I	R	S	I	R	S	I	R
Birth by $S$	$\kappa$	1	.	.	2	.	.	1	.	.
*Birth by $I$	$\kappa$	.	1	.	1	1	.	1	.	.
Birth by $R$	$\kappa$	.	.	1	1	.	1	1	.	.
*Infection of $S$	$\beta_S/N$	1	1	.	.	2	.	-1	1	.
*Infection of $I$	$\beta_I/N$	.	2	.	.	2	.	.	.	.
*Infection of $R$	$\beta_R/N$	.	1	1	.	1	1	.	.	.
*Recovery of $I$	$\gamma$	.	1	.	.	.	1	.	-1	1
Waning of $R$	$\omega$	.	.	1	1	.	.	1	.	-1
Death of $S$	$\mu$	1	.	.	.	.	.	-1	.	.
*Death of $I$	$\mu$	.	1	.	.	.	.	.	-1	.
Death of $R$	$\mu$	.	.	1	.	.	.	.	.	-1

**Table SI1:** The stoichiometric matrices for the SIRS model with demography. The (\*) indicates reactions that are jumping when  $I$  is discrete and  $S$  and  $R$  are continuous. The rows with blue coloured entries do not alter the exchange through the system, and so need not be represented.

Parameter	$\beta$	$\gamma$	$\omega$	$\kappa$	$\mu$
Rate description	Infection	Recovery	Immunity Waning	Birth	Death
Value	2/7	1/7	1/365	1/(85 × 365)	1/(85 × 365)

**Table SI2:** Parameter values used for SIRS model with demography.



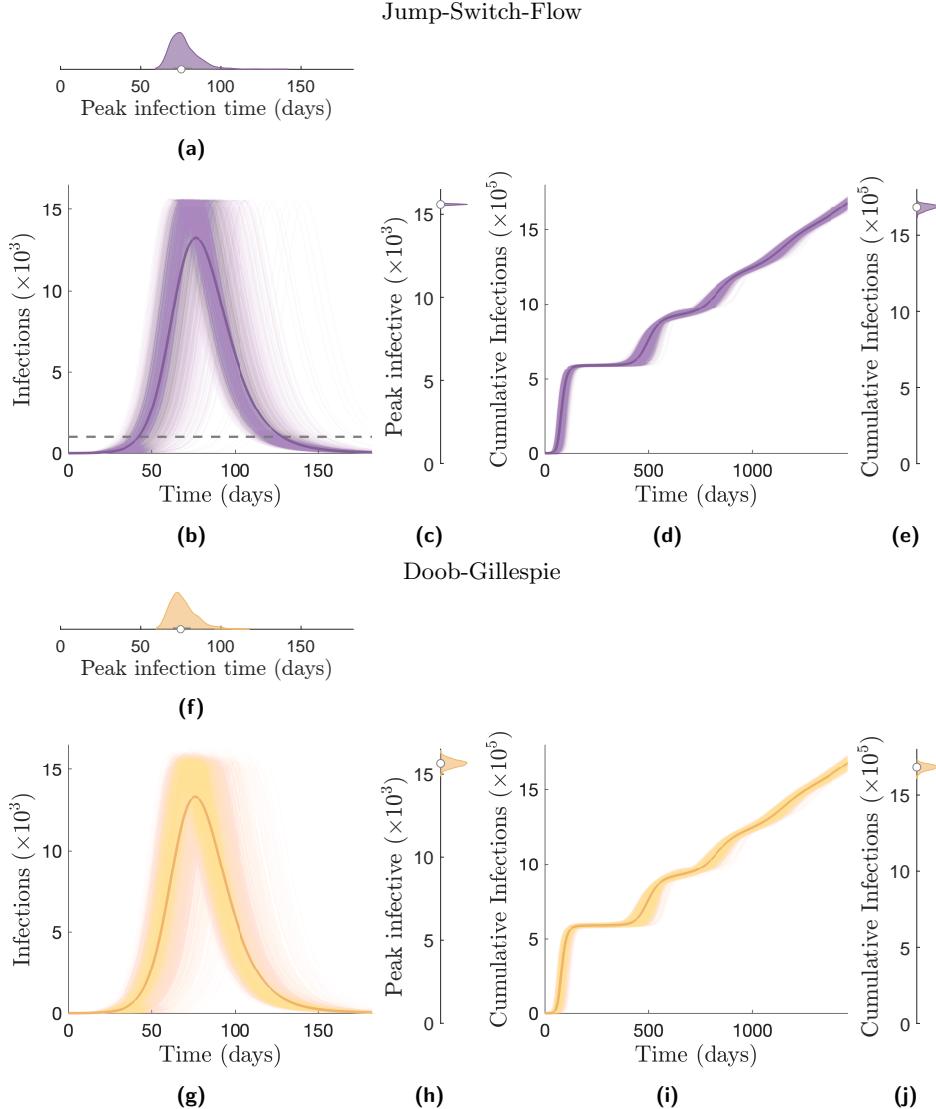
**Figure SI6:** Possible simulated trajectories of SIRS model with demography for an initial population size  $N(0) = 10^5$ , from Doob-Gillespie (yellow) and Jump-Switch-Flow (purple,  $\Omega = 10^3$ ) methods: (a) two examples of extinction trajectories of the infectious compartment, with associated  $S - I$  phase plane in (d); (b) two examples of fade-out trajectories of the infectious compartment, with associated  $S - I$  phase plane in (e); (c) two examples of endemic trajectories of the infectious compartment, with associated  $S - I$  phase plane in (f).

#### SI4.1 Simulation experiments: Comparing Jump-Switch-Flow and Doob-Gillespie

We compare our Jump-Switch-Flow method to the gold standard approach for simulating exact solutions to CTMCs, the Doob-Gillespie method. To do so, we first specified an initial population size of  $N(0) = 10^5$ , and a switching threshold of each compartment of  $\Omega = 10^3$ . We then generated 5,000 simulations using the parameters in Table SI2.

To capture the dynamics of the model, we track the infectious compartment as a key compartment of interest. In doing so, we then obtain distributions for the peak number of infectious individuals and also the time for the infection to peak, for the fade-out scenario. We also track the cumulative number of infected individuals after a certain amount of time for the

endemic scenario. These results are presented in Figure SI8.



**Figure SI7:** We compare the infectious compartment from realisations for Jump-Switch-Flow (b) and Doob-Gillespie (g) solutions, for a population size of  $N(0) = 10^5$ . We use a switching threshold of  $\Omega = 10^3$ . We also show the distributions of the time to infectious peak ((a) and (f)), and infectious peak values ((c) and (h)). We compare the cumulative infections over the simulations for Jump-Switch-Flow (d) and Doob-Gillespie (i) solutions and the final distributions ((e) and (j)). Point averages are also presented as a solid line.

Comparing Figures SI7a and SI7f, we can immediately observe that the distributions for time to peak for Jump-Switch-Flow method appears to be representative of that for the Doob-Gillespie method. This is because the delay in the epidemic take off at low population is fully captured with this sufficiently large choice of switching threshold,  $\Omega = 10^3$ , enabling the full stochastic effects to be accurately represented.

However, if we compare the distributions for the peak number of infectious individuals (SI7c and SI7h), we can see that they differ significantly, for this chosen switching threshold. This difference can be understood if we consider how the process is behaving. Once the infectious compartment is sufficiently large for the Jump-Switch-Flow method, the method switches from a stochastic representation to a deterministic representation. Therefore, once the infectious compartment becomes deterministic, given that the susceptible population is also deterministic, its peak value is determined by the number of both infectious and susceptible individuals at this time. Since the number of infectious people at this time will always be the same,  $I = \Omega = 10^3$  people, any variation in peak value is due to the number of susceptible individuals,  $S$ , at the time of switching.

If the summary statistic we instead compare is the distribution of the cumulative number of infected individuals after four years of epidemic circulation, we observe that, even for a relatively low switching threshold, the Jump-Switch-Flow method performs well comparatively to the Doob-Gillespie method, see Figures SI7e and SI7j. We also note that for these parameter choices and switching threshold, the infectious compartment switches between jumping and flowing states multiple times, yet the Jump-Switch-Flow method produces a cumulative distribution comparable to that of Doob-Gillespie. This indicates that the Jump-Switch-Flow method is robust for long term simulations.

The final summary statistics we are interested in are the probabilities of extinction, fade-out and endemic scenarios. Table SI3 shows the probability of each scenario computed from 5,000 simulated samples. Here, we see that both methods produce comparatively similar results, which indicates that the Jump-Switch-Flow method is capable of capturing the inherent stochasticity in the exact simulation of the system via the Doob-Gillespie method.

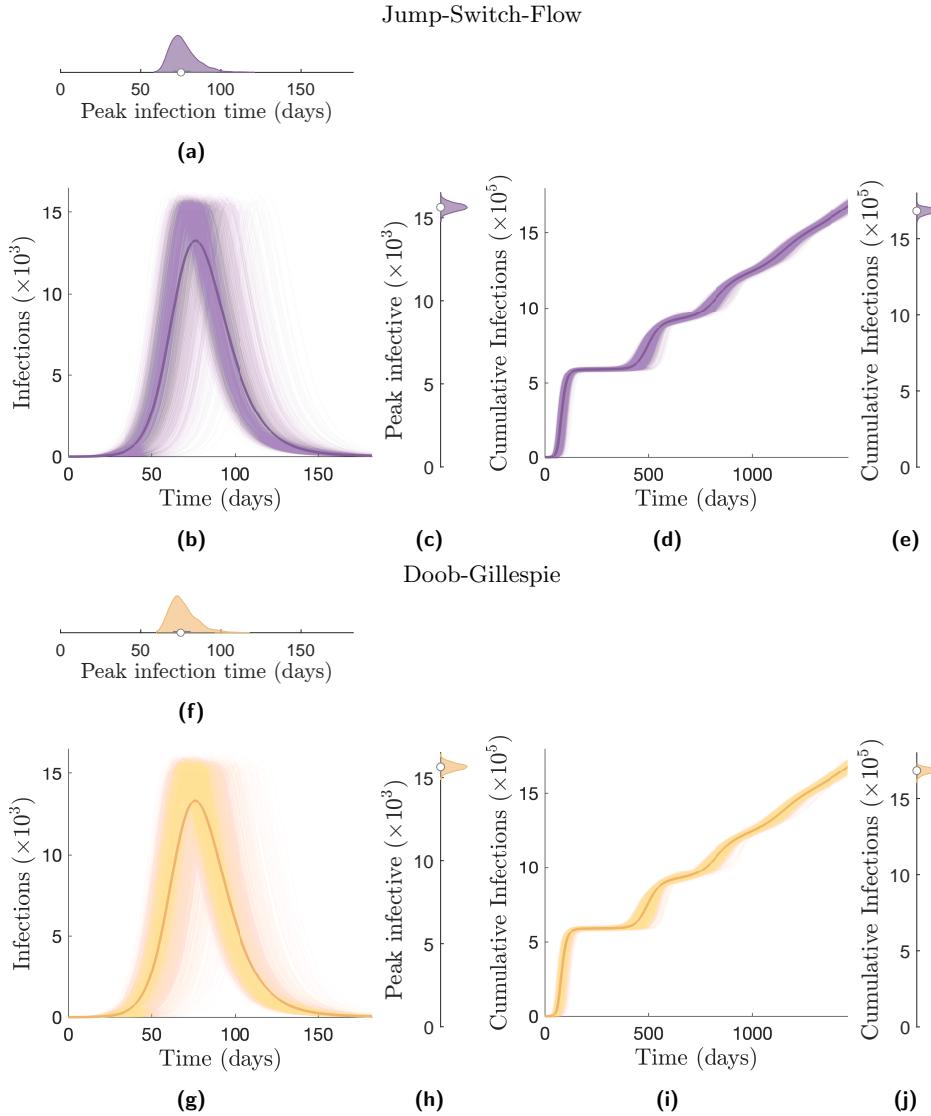
Method	Extinction	Fade-out	Endemic
Jump-Switch-Flow	$0.2476 \pm 0.0120$	$0.4774 \pm 0.0138$	$0.2750 \pm 0.0124$
Doob-Gillespie	$0.2520 \pm 0.0120$	$0.4668 \pm 0.0138$	$0.2812 \pm 0.0125$

**Table SI3:** Probabilities (with associated 95% confidence interval) of scenario outcomes for Jump-Switch-Flow and Doob-Gillespie, from 5,000 simulations.

We also compare the Jump-Switch-Flow and Doob-Gillespie methods where the switching threshold has been set to the population size, resulting in no compartments switching into the flowing state. These results are further discussed in SI4.2, where we show the two methods are indistinguishable.

#### SI4.2 Simulation experiments: Comparing Jump-Switch-Flow (with no switching) and Doob-Gillespie

We compare the Jump-Switch-Flow method to the Doob-Gillespie method. To implement the latter, we set the switching threshold to be the total population size, to ensure no compartments switch into a flowing regime, and therefore all compartments are completely stochastic. We can therefore investigate how our Jumping process compares to exact solutions of the Continuous Time Markov Chains.



**Figure SI8:** We compare the infectious compartment from realisations for Jump-Switch-Flow (b) and Doob-Gillespie (g) solutions, for a population size of  $N_0 = 10^5$ . We use a switching threshold of  $\Omega = 10^3$ . We also show the distributions of the time to infectious peak ((a) and (f)), and infectious peak values ((c) and (h)). We compare the cumulative infections over the simulations for Jump-Switch-Flow (d) and Doob-Gillespie (i) solutions and the final distributions ((e) and (j)). Point averages are also presented as a solid line.

Using parameter values in Table 3 of the main text, we generated 5,000 simulations, tracking the same quantities of interest. As with the case with compartment switching, the peak infection time distribution of JSF (Figure SI8a) and Doob-Gillespie (Figure SI8f) are comparable. However, this time, as well as a similar median peak infective individuals, the distributions of both JSF (Figure SI8c) and Doob-Gillespie (Figure SI8h) are also comparable. Lastly, as we saw with compartment switching, the distributions of cumulative infections after four years are comparable for both JSF (Figure SI8e) and Doob-Gillespie (Figure SI8j).

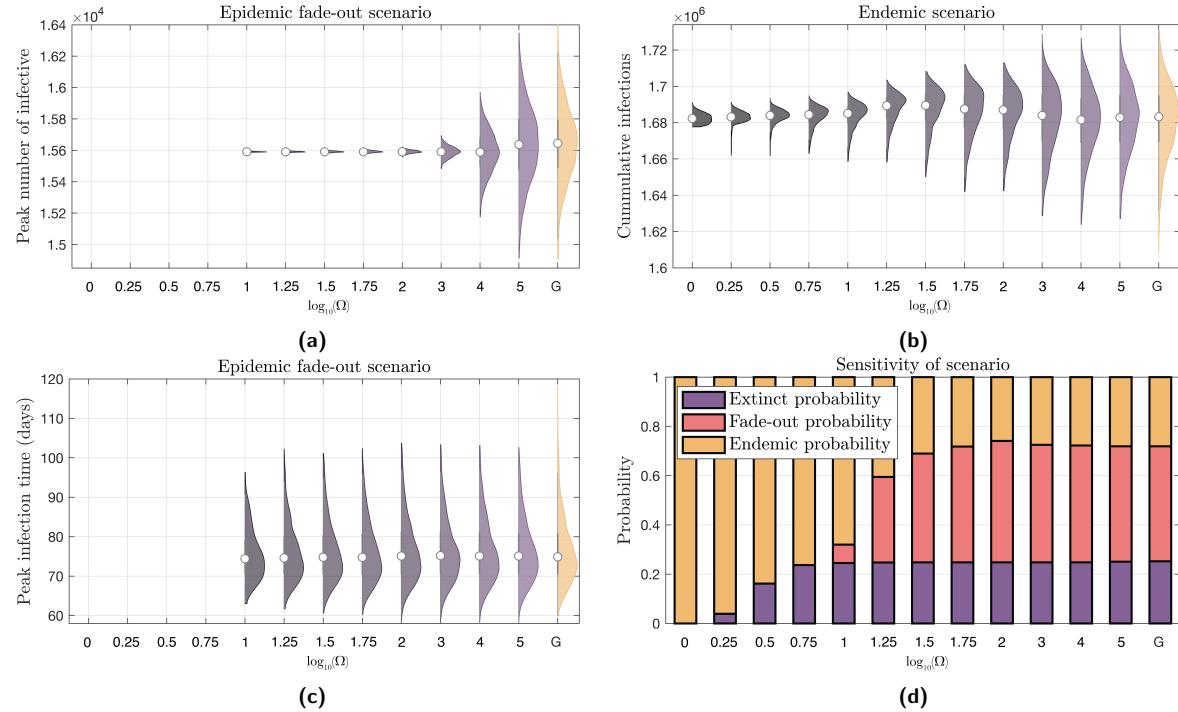
Table SI4 shows that the probabilities of scenario outcomes are also comparable for both Jump-Switch-Flow (with no switching) and Doob-Gillespie.

Method	Extinction	Fade-out	Endemic
Jump-Switch-Flow	$0.2512 \pm 0.0120$	$0.4700 \pm 0.0139$	$0.2788 \pm 0.0123$
Doob-Gillespie	$0.2520 \pm 0.0120$	$0.4668 \pm 0.0139$	$0.2812 \pm 0.0125$

**Table SI4:** Probabilities (with associated 95% confidence interval) of scenario outcomes for Jump-Switch-Flow (with no switching) and Doob-Gillespie, from 5,000 simulations.

### SI4.3 Simulation experiments: Sensitivity of Jump-Switch-Flow to switching threshold

In this section, we demonstrate how the accuracy of the Jump-Switch-Flow method can exhibit sensitivity with respect to the switching threshold (Figure SI9). In Figure SI9a, we can see that the distribution of peak number of infective individuals for the fade-out scenario is highly sensitive to the switching threshold. For a small switching threshold ( $\Omega = 10^1$ ), we see that the distribution exhibits low variance, is representative of the deterministic expectation, and is not capturing the distribution obtained with the Doob-Gillespie method. As the switching threshold is increased, we see that the distribution of the peak number of infective individuals increases in variance. However, the Jump-Switch-Flow method is not representative of the Doob-Gillespie method until a large switching threshold ( $\Omega = N(0) = 10^5$ ), where all compartments are in the jumping state.



**Figure SI9:** Summary statistics of interest to observe the sensitivity of the Jump-Switch-Flow method with respect to the switching threshold. (a) Sensitivity to the distribution of peak number of infective individuals for the fade-out scenario. (c) Sensitivity to the distribution of peak infection time for the fade-out scenario. (b) Sensitivity to the distribution of the cumulative number of infected individuals after four years for the endemic scenario. (d) Sensitivity to the probability of each scenario occurring. In each of the plots (a)-(c) grey to purple shaded distributions represent results obtained with the Jump-Switch-Flow method whilst the yellow distributions are the gold standard Doob-Gillespie algorithm (for which the placeholder 'G' is used to signify) which exhibit exact stochasticity but at the cost of high computational requirements.

Figure SI9c presents the distribution of peak infection time for the fade-out scenario for various switching thresholds, and comparing to the Doob-Gillespie method. Here, we see that the Jump-Switch-Flow method with a relatively small switching threshold ( $\Omega = 10^2$ ) is capable of capturing a representative distribution of the Doob-Gillespie method.

Similarly, Figure SI9b presents the distribution of the cumulative number of infected individuals after four years for the endemic scenario. Here, we again see that the Jump-Switch-Flow method with a relatively small switching threshold ( $\Omega = 10^3$ ) is also representative of the Doob-Gillespie method.

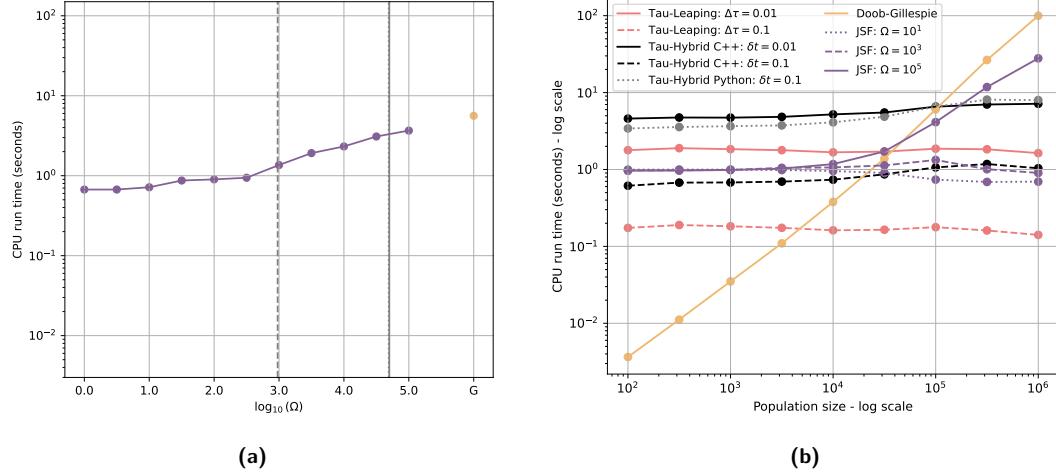
Lastly, Figure SI9d shows the probability of each scenario occurring (extinction, fade-out, endemic). Here, we see that the Jump-Switch-Flow method with a switching threshold of  $\Omega \leq 10^1$  over-represents the endemic scenario. However, increasing the switching threshold, we observe that the probability of each scenario occurring quickly becomes representative of those found by the Doob-Gillespie method.

### SI4.4 Simulation experiments: Computational efficiency of Jump-Switch-Flow

We now present the computational efficiency of the Jump-Switch-Flow method, and compare it to the Doob-Gillespie method and Tau-Leaping method (which is regarded as a computationally efficient, approximate, method), and the Tau-hybrid method, provided by GillesPy2 Matthew et al., 2023. We implement two variants of Tau-leaping with a fixed step size of  $\Delta\tau = 0.01$  days, and  $\Delta\tau = 0.1$  days respectively, two variants of the C++ implementation of the Tau-hybrid method, with time steps of size  $\Delta t = 0.01$  days and  $\Delta t = 0.1$  days respectively, one implementation of the Python Tau-hybrid

method, with time steps of size  $\Delta t = 0.1$  days, and the Jump-Switch-Flow method with a time step size of  $\Delta t = 0.01$  days.

Figure SI10a shows how the computational efficiency varies with the switching thresholds,  $\Omega$ , for the SIRS model with demography presented in the previous section, and compares it to the Doob-Gillespie method. We see that as we increase the switching threshold, the Jump-Switch-Flow method requires more computational time to simulate the scenario for four years, but is always more efficient than the Doob-Gillespie method. This increased efficiency is because the Jump-Switch-Flow method utilises the Next Reaction Method to compute the stochastic events, a method known to be more computationally more efficient than the direct, first reaction, Doob-Gillespie method.



**Figure SI10:** (a) The running time to simulate the scenario for four years for an initial population size of  $N(0) = 10^5$ , with varying switching threshold,  $\Omega$  (grey to purple), compared with the Doob-Gillespie ('G') method (yellow). The endemic steady state number of infectious individuals (vertical dashed line), susceptible individuals (vertical solid line), and recovered individuals (vertical dotted line) are presented. (b) The running time for the Jump-Switch-Flow method for an appropriate threshold choice is approximately constant as the population size varies (for example  $\Omega = 10^3$ ). If a poor switching threshold is chosen, the running time begins to scale exponentially (for example  $\Omega = 10^5$ ). For the Tau-Leaping method, the running time is constant, as we use fixed step size of  $\Delta\tau = 0.01$  (solid line) and  $\Delta\tau = 0.1$  (dashed line). We also present the Tau-hybrid method, in both Python and C++, with various different time meshes, C++ with a fine mesh ( $\delta t = 0.01$ , black solid line) and a coarse mesh ( $\delta t = 0.1$ , black dashed line), and Python with a coarse mesh ( $\delta t = 0.1$ , grey dotted line).

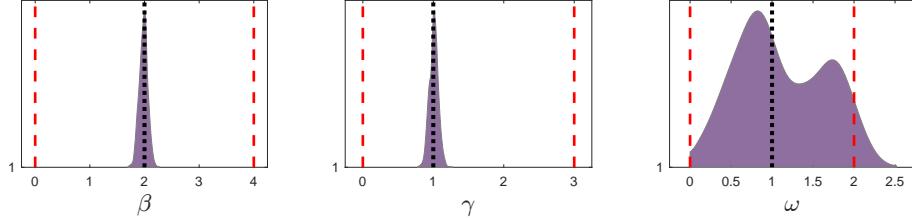
To observe how the computational efficiency of our Jump-Switch-Flow method varies with the total (population) size of the system, we consider three switching thresholds:  $\Omega = 10^1$  (dotted purple),  $\Omega = 10^3$  (dashed purple), and  $\Omega = 10^5$  (solid purple). Figure SI10b compares the Jump-Switch-Flow method (purple) to the Doob-Gillespie method (yellow), to the computationally more efficient approximation Tau-Leaping method ( $\Delta\tau = 0.01$ , dashed red, and  $\Delta\tau = 0.1$ , solid red), and to the Tau-hybrid method ( $\Delta t = 0.01$  solid black, and  $\Delta t = 0.1$  dashed black) provided by GillesPy2, noting the log-log scale. We simulate the SIRS system described above, for a total time of  $4 \times 365$  days. We can see that the computational efficiency of the Doob-Gillespie method scales exponentially with system size, as expected. The Tau-Leaping method initially has constant computational time scaling. In contrast, our Jump-Switch-Flow method is constant in time up to an initial system size of  $N(0) = 10^5$ , after which it begins to slowly decrease (for  $\Omega = 10^1$  and  $\Omega = 10^3$ ). This decrease is caused by more of the scenario being simulated in the flowing state, as the system becomes highly deterministic with a larger population. We also observe our Jump-Switch-Flow method increasing exponentially between  $N(0) = 10^4$ , for the largest switching threshold of  $\Omega = 10^5$  (solid purple). We observe our Jump-Switch-Flow method performing better, in comparison to Doob-Gillespie exact method, as we use the Next Reaction Method to sample the jump times. This is despite our implementation being required to perform extra computation to check the regime of the Jump-Switch-Flow process, and implement any relevant regime changes (even though they are not required in this particular instance). We also observe that for large, realistic population sizes (i.e.  $N(0) \geq 10^5$ ), the Tau-hybrid method with the coarse time mesh (black dashed line) performs at best comparable to, or worse than the Jump-Switch-Flow method, if a suitable  $\Omega$  is chosen (such as  $\Omega = 10^3$ ). Moreover, as we specify a finer time mesh (solid black line) we observe that the Tau-hybrid method performs worse across all population sizes. We anticipate this is due to the Tau-hybrid method utilising an arbitrary ODE solver, an adaptive  $\Delta\tau$  that is calculated at each step, and also due to the dynamic switching behaviour between stochastic and deterministic regimes that requires extra computation of the propensities to determine if a switching event should occur.

#### SI4.5 Simulation experiments: Inference with simulated data, parameter estimates

To test how the Jump-Switch-Flow method behaves when used to perform parameter estimation, we first generated synthetic data of a known SIRS model with demography. We simulate data for 400 days using the parameter values given in Figure 2C, with the Doob-Gillespie algorithm, and select a trajectory which experiences epidemic fade-out (at day 250), and an initial condition of two infected individuals ( $I(0) = 2$ ), no recovered individuals ( $R(0) = 0$ ), and a total population of

$N(0) = 10^5$  individuals. We track the number of infectious individuals within the population each day, and use a truncated binomial distribution to add noise into the data set, reflecting a realistic measurement process. We then couple our Jump-Switch-Flow method, using a switching threshold of  $\Omega = 10^3$ , with the open-source particle filter package pypfilt (Moss, 2024), using 2,000 particles. We first fix the demographic parameters to  $\kappa = \mu = 1/(85 \times 365)$  (since these are assumed to be standard within a given population). We perform the inference on the data set for the first 100 days of the epidemic to estimate  $\beta$ ,  $\gamma$  and  $\omega$ , and then use our parameter estimates to predict forward in time the possible trajectories based on the obtained predictions. Using these predictions, we also estimate the probability of epidemic fade-out occurring at times  $t = 100$  to  $t = 400$  days.

Figure SI11 shows the posterior distributions of parameters. The dotted black lines show the true value used to produce the simulated data, and the dashed red lines indicate the initial bounds on the prior estimates. We see that both  $\beta$  and  $\gamma$  are well estimated, with the posteriors centered around the true values. However  $\omega$  is not well estimated. This is because  $\omega$  acts on the order of 1 year, while the window of data used to fit the parameters to the data is 100 days ( $\approx 0.3$  years).



**Figure SI11:** Posterior distributions of parameters for the SIRS model with demography, obtained via a particle filter with 2,000 particles, and the Jump-Switch-Flow sampler. The red dashed lines indicate the initial bounds on prior samples. The vertical dotted black lines indicate the true value used to produce the simulated data.

## SI5 Inference Study: Estimating TEIRV model parameters using a particle filter

We implemented the Refractory Cell Model described by the authors for the viral load data obtained from nasal swabs, using our Jump-Switch-Flow method with a switching threshold on all compartments  $\Omega = 10^2$ . We implemented this in particle filter pypfilt Moss, 2024 to estimate the model parameters, using 6,000 particles per simulation. In previous analysis Ke et al., 2022, the decay of virion was fixed at  $c = 10$  virions/day, and the eclipse of cells to become infectious as  $k = 4$  cells/day, leaving the remaining parameters to be estimated. Therefore, we similarly fix  $c = 10$  virions/day and  $k = 4$  cells/day. The prior distributions used for the analysis with the TEIRV model are given in Table SI5.

### SI5.1 Parameter Estimation and Viral Clearance Prediction

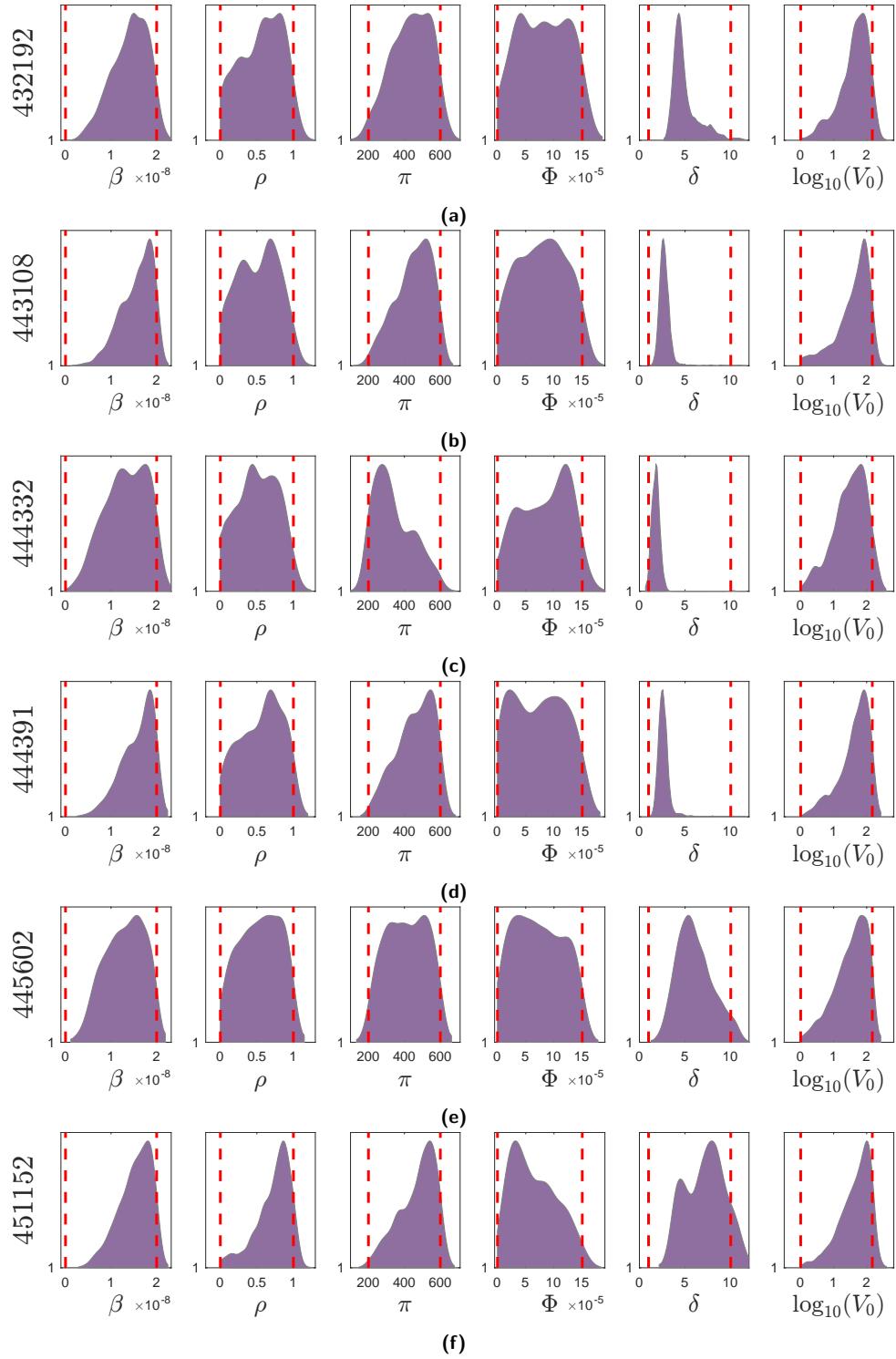
Figure 9 of the main text highlights how the estimated viral loads closely resemble the data collected from the 6 patients. We also show how the estimated  $R_0$  values for the patients vary. These  $R_0$  estimates are consistent with the original study by Ke et al., 2022. However, understanding how the parameter estimates vary between patients assists in understanding how the fitting process behaves.

The posterior distributions for each of the 6 patient's parameter estimates are provided in Figure SI12. These posterior distributions are found by fitting a violin envelope from the 6,000 estimates obtained from the particle filtering process. We also show the prior distributions, as red dashed lines, which show the lower and upper bounds for the uniform initial guesses.

We can see that, often, parameters  $\rho$  and  $\Phi$  are not identified here, as the posterior distributions closely resemble the priors (excluding patient 451152). We also observed that for all patients,  $\beta$  appears to be well estimated. The parameter  $\pi$  is also well estimated for patients 443108, 444332, 444391 and 451152, however patients 432192 and 445602 provide less insight. The parameter  $\delta$  is very well estimated by all patients, excluding patient 451152. Lastly, the initial viral loads,  $\log_{10}(V_0)$ , are all estimated well, and also estimated to take a large value.

Parameter	Prior
$\log_e(V_0)$	Uniform(0, 5)
$c$	Fixed at 10.0
$k$	Fixed at 4.0
$\beta \times 10^7$	Uniform(0, 20)
$\Phi \times 10^5$	Uniform(0, 15)
$\rho$	Uniform(0, 1)
$\delta$	Uniform(1, 10)
$\pi$	Uniform(200, 400)

**Table SI5:** The prior distributions used for the analysis with the TEIRV model.



**Figure SI12:** Posterior distributions of the model parameters and initial viral load for each of the selected SARS-CoV-2 infections. The red dashed lines indicate the initial (uniform) bounds on prior samples.