Efficient computation of the probability generating function in a general branching process model for infectious diseases

Frederik Mølkjær Andersen
Supervisors: Samir Bhatt & Susanne Ditlevsen

April 13, 2023

1 BRANCHING PROCESSES IN EPIDEMIOLOGY

Mathematical epidemiology is key to gaining knowledge about the spread of contagious diseases. Whether this entails inference and forecasting of current ongoing epidemics or simulations of possible future ones, mathematical models for infectious diseases play a large role in controlling the impact of these diseases on society and how to handle them. On the most basic level what produces an epidemic is infections happening as discrete events between two individuals and these events are essential in understanding the spread of a disease, however for big-picture epidemiological decision-making, what is mainly interesting is the population aggregate of these individual infections defining the epidemic trajectory. Naturally, the same contrast is found in the mathematical modeling of infectious diseases, where there exist two broad categories of modeling strategies, individual-based models and models based on governing equations.

Individual-based models are most prominently encountered as agent-based models [1], which simulate an epidemic directly from the individual infection events between individuals. Individual-based models are usually easy to understand and it is simple to make them quite complex and to model under realistic assumptions, however, deeper mathematical analysis is not tractable, inference is challenging and reliable estimates from these model requires a high number of simulations. On the other hand governing equations simply model the population-level quantities directly, and due to their simpler mathematical description they are easier to analyze and to perform inference over, at the cost of making strong simplifying assumptions. Popular governing equation approaches include the *susceptible-infectious-recovered* (SIR) models [2] and renewal equation models [3].

Combining the low-level descriptions of individual-based models and the population overview of governing equations is important

to understand the effects of the individual infection events on the population level, in a stochastic setting in particular we would want to understand the effect of randomness in infection events on the trajectory of the epidemic [4]. In this area lies stochastic branching process models, which are based on a stochastic process describing the individual infection events according to simple rules, but from which we can derive governing equations for their distribution with analytical tools.

An interesting starting point for branching process models for infectious diseases is the Bellman-Harris process [5], which assumes that each infected individual has a random period of infection after which the individual infects a random number of new individuals, which then again follows the same process independently of each other. This simple setup of a model turns out to be analytically tractable and interestingly, the expectation of the process is governed by a renewal equation [6]. We thus see a clear path from the individual-based models to the governing equations. Extensions of the Bellman-Harris process are many, but in particular, the Crump-Mode-Jagers process [7, 8, 9] allows for a more epidemiologically relevant process distributing the time of infections from a given individual over the whole period of infection, according to some counting process.

The Crump-Mode-Jagers process used in epidemic modeling usually assumes a constant rate of infection, but [10] introduced a time-varying version of the Crump-Mode-Jagers process where the offspring distribution is allowed to change over time. Furthermore, the process is analytically tractable and it is shown in [10] that the expectation of the process follows a renewal-type governing equation. Modeling the expectation of a process is in itself not that fruitful without knowledge of the uncertainty around this mean, in particular for stochastic processes where uncertainty from individual infection events might have a large impact on the uncertainty of the epidemic trajectory. [4] extended the results on this process to obtain a governing equation for its probability-generating function, allowing a full description of its distribution.

However, for epidemiological practitioners, the general form integral equation for the probability-generating function is not easy to work with. Simple implementations in interpreted languages such as python exist but are not very efficient, requiring advanced computation techniques, such as using a just-in-time compiler to speed up the code, in order to be usable in practice. Hence there is a need for a simple and fast implementation of the probability-generating function which is easy to use for epidemiologists. We present a vectorized version of the current implementation giving a considerable increase in efficiency, which is coded directly in python using only standard libraries.

2 PROBABILITY GENERATING FUNCTIONS

The distribution of a stochastic process is often a difficult object to work with directly, hence we could map the distribution into another object which in turn should be easier to analyze, and from which we can obtain the original distribution when needed. A well-known example of such a map is a probability-generating function, and in particular when working with branching processes, these turn up everywhere [11].

The probability generating function (or just generating function) of a random variable X with values in $\mathbb{N}_0$ is the function $F : \overline{D}(0; R) := \{z \in \mathbb{C} \mid |z| \leqslant R\} \to \mathbb{C}$ given by

Probability generating functions are most often defined as functions from $[0,1]$ to $[0,1]$, however, there's no trouble in our complex extension as long as we stay inside the disk of convergence

$$F_X(s) := \mathbb{E}\left[s^X\right] = \sum_{k=0}^{\infty} s^k \mathbb{P}(X = k)$$

where R is the radius of absolute convergence of the power series. Note that $R \geqslant 1$ as $\sum_{k=0}^{\infty} \mathbb{P}(X = k) = 1 < \infty$, and as we shall see we are mostly interested in the generating function evaluated at $z \in \overline{D}(0, 1)$, so we will without trouble ignore eventual convergence problems for $R > 1$.

2.1 Properties of generating functions

The paramount property of the generating function is that it generates the probability mass function (i.e. all the point probabilities of X) through repeated differentiation. In particular, we have

$$\mathbb{P}(X = k) = \frac{1}{k!} \frac{d^k}{ds^k} F(s)\bigg|_{s=0} \tag{1}$$

thus, the generating function determines the distribution of the random variable completely, and we can gain insight into the distribution just by working with the generating function.

Of particular usefulness to us is the fact that the generating function of a random sum of i.i.d. variables has a particularly nice form. If we let $X_1, X_2, \ldots$ be a sequence of i.i.d. $\mathbb{N}_0$ random variables with common generating function $F_X(s)$ and Y be a $\mathbb{N}_0$ random variable independent of $X_1, X_2, \ldots$ with generating function $F_Y(s)$, then the generating function of $S := \sum_{i=1}^{Y} X_i$ is given by

$$F_S(s) = F_Y(F_X(s)) \tag{2}$$

This is hence giving us a way to investigate the distribution of S.

2.2 Examples

We illustrate the use of generating functions via the Poisson and negative binomial distributions, these examples will continue throughout the report as illustrations of our methods.

Example 2.2.1 (Generating function of Poisson distribution)**.** Let $Y \sim \text{Pois}(\lambda)$ for $\lambda > 0$, i.e. $\mathbb{P}(Y = k) = e^{-\lambda}\lambda^k/k!$ for $k \in \mathbb{N}_0$. We will now see that the pmf can be found from the generating function. The generating function is then

$$F_Y(s) = \sum_{k=0}^{\infty} s^k \frac{e^{-\lambda}\lambda^k}{k!} = e^{-\lambda} \underbrace{\sum_{k=0}^{\infty} \frac{(s\lambda)^k}{k!}}_{=: e^{s\lambda}} = e^{\lambda(s-1)}$$

Now we use eq. (1) to reconstruct the pmf,

$$\begin{aligned} P(Y = k) &= \frac{1}{k!}\frac{d^k}{ds^k} F_Y(s)\Big|_{s=0} \\ &= \frac{e^{-\lambda}}{k!}\frac{d^k}{ds^k} e^{s\lambda}\Big|_{s=0} \\ &= \frac{e^{-\lambda}\lambda^k}{k!} e^{s\lambda}\Big|_{s=0} = \frac{e^{-\lambda}\lambda^k}{k!} \end{aligned}$$

and we see that we gain the same familiar probability mass function as we know for the Poisson distribution.

Example 2.2.2 (Negative binomial is compound Poisson)**.** Let $Z \sim \text{NegBin}(\phi, p)$ for $\phi > 0$ and $p \in (0, 1)$ for $k = 1, 2, ..., i$. Z can be shown to have generating function $F_Z(s) = (p/(1-(1-p)s))^{\phi}$ directly using the probability mass function. We will now see that the negative binomial distribution can be constructed as a compound Poisson distribution, that is $Z \overset{\mathcal{D}}{=} \sum_{i=1}^{\mathcal{M}} Y_i$ where $Y_1, Y_2, ...$ is a sequence of i.i.d. $\mathbb{N}_0$ random variables and $\mathcal{M}$ is Poisson and independent of $Y_1, Y_2, ...$

Consider such a setup with $Y_i \sim \text{Logarithmic}(1-p)$ and $\mathcal{M} \sim \text{Pois}(-\phi\log(p))$. We have just seen in Example 2.2.1 that $F_{\mathcal{M}}(s) = \exp(-\phi\log(p)(s-1))$ and one can find that $F_{\mathcal{M}}(s) = \log(1-(1-p)s)/\log(p)$. Now we can use eq. (2) to find the generating function of $S := \sum_{i=1}^{\mathcal{M}} Y_i$,

$$\begin{aligned} F_S(s) &= F_{\mathcal{M}}(F_Y(s)) \\ &= \exp\left(-\phi\log(p)\left(\frac{\log(1-(1-p)s)}{\log(p)} - 1\right)\right) \\ &= \exp\left(\log\left(\left(\frac{1-(1-p)s}{p}\right)^{-\phi}\right)\right) \\ &= \left(\frac{p}{1-(1-p)s}\right)^{\phi} = F_Z(s) \end{aligned}$$

and since the generating function determines the distribution completely we have $Z \overset{\mathcal{D}}{=} \sum_{i=1}^{\mathcal{M}} Y_i$.

2.3 A trick to compute derivatives

For simple distributions like the Poisson distribution we can explicitly go from the generating function to the probability mass function without trouble, however, for more complicated generating functions this becomes quite much harder, if not impossible. The central generating function of this paper doesn't even have a closed-form expression, and we would have to resort to numerical derivatives.

Alternatively, we can use Cauchy's integral formula for derivatives [12]. Let D be a bounded domain with smooth boundary ∂D. If $f : D \to \mathbb{C}$ is analytic and can be smoothly extended onto the closure $\overline{D}$ of the domain, we can compute all derivatives of f on D by the formula,

$$f^{(n)}(z) = \frac{n!}{2\pi i} \oint_{\partial D} \frac{f(\zeta)}{(\zeta - z)^{n+1}} \, d\zeta \tag{3}$$

for $z \in D$ and $n \in \mathbb{N}_0$.

Generating functions are power series, hence they are by definition analytic on their disk of convergence $D(0; R) := \{ z \in \mathbb{C} \mid |z| < R \}$ and continuous on $\overline{D}(0; R)$, moreover the same is then true for any $r \in (0, R]$, f is analytic on $D(0; r)$ and continuous on $\overline{D}(0; r)$. The boundary $\partial D(0; r)$ is the circle of radius r, which is certainly smooth. It is then clear that Cauchy's integral formula can be used to reformulate eq. (1) as $0 \in D(0; r)$,

$$\mathbb{P}(X = k) = \frac{1}{k!} F^{(k)}(0) = \frac{1}{2\pi i} \oint_{\partial D(0;r)} \frac{F(s)}{s^{n+1}} \, ds, \quad k \in \mathbb{N}_0. \tag{4}$$

To integrate over $\partial D(0; r)$ we can use the parametrization $z = re^{2\pi i u} \in \partial D(0; r)$ for $u \in [0, 1)$ and thus with $dz = 2\pi i \, re^{2\pi i \theta} \, du$ we get

$$\mathbb{P}(X = k) = \frac{1}{2\pi i} \int_0^1 \frac{F(re^{2\pi i u})}{(re^{2\pi i u})^{n+1}} 2\pi i \, re^{2\pi i u} \, du \tag{5}$$

$$= \frac{1}{r^n} \int_0^1 \frac{F(re^{2\pi i u})}{e^{2\pi i u n}} \, du, \quad k \in \mathbb{N}_0 \tag{6}$$

A simple approximation to this integral is to consider the left Riemann sum for large M. Although seemingly a crude approximation as [13] notes, for a periodic function, which the integrand in eq. (6) is, the simple sum converges geometrically.

$$\mathbb{P}(X = k) \approx \frac{1}{Mr^n} \sum_{m=0}^{M-1} F(re^{2\pi i m/M}) e^{-2\pi i n m/M} \tag{7}$$

but note that the sum here is simply a discrete Fourier transform of the sequence $\left(F(re^{2\pi i m/M})\right)_{m \leqslant M-1}$, making computations fast using a fast Fourier transform. [14] suggests using $r = 1$, which we will follow.

3 A TIME-VARYING CMJ PROCESS

In the following, we develop an outbreak model using a time-varying version of the Crump-Mode-Jagers branching process [7, 8, 9]. The process is extended such that the properties of the infection process are allowed to vary over time for each individual. We use the notion of random characteristics [15] to model both prevalence and cumulative incidence under the same model framework. The model description loosely follows [10] and [4] in which the model was first proposed.

An index case, i_0, is infected at a deterministic time $\tau_{i_0} \geqslant 0$. All following infections in the process will be k'th generation offspring of this index case for some $k \in \mathbb{N}$, we denote the set of all individuals infected by $\mathcal{I}$ and let $\mathcal{I}^* = \mathcal{I} \setminus \{i_0\}$. From the index case, infections happen according to the random duration of infection for the individual and a counting process allowing for infections to happen randomly over the period of infection. The individuals infected by the index case can be thought of as index cases in new independent epidemics started at their respective infection times. We will now formalize the random elements describing the infection process and the main branching process.

In practice $\tau_{i_0} = 0$, but we will see that leaving it as a free parameter is essential for the coming derivations of the generating function.

Each individual $i \in \mathcal{I}$ is associated with an i.i.d. collection of random triples indexed by the infection time $\tau \geqslant 0$, which determines the subsequent infection dynamics of the individual,

$$\left\{ \left(L_i^\tau, \left\{ N_i^\tau(t) \right\}_{t \geqslant \tau}, \left\{ \chi_i^\tau(t) \right\}_{t \geqslant \tau} \right) \right\}_{\tau \geqslant 0} =: \left\{ (L_i^\tau, N_i^\tau, \chi_i^\tau) \right\}_{\tau \geqslant 0}$$

each $\left\{ (L_i^\tau, N_i^\tau, \chi_i^\tau) \right\}_{\tau \geqslant 0}$ being an i.i.d. copy of $\left\{ (L^\tau, N^\tau, \chi^\tau) \right\}_{\tau \geqslant 0}$ for each $i \in \mathcal{I}$. For given infection time $\tau \geqslant 0$ we let

This self-similarity is a fundamental property that allows us to characterize the whole process only from the first generation of infected individuals.

- L^τ: Infectious period. A strictly positive real random variable with distribution function G^τ, determining for how long an individual is infected.
- $N^\tau := \{N^\tau(t)\}_{t \geqslant \tau}$: Infection counting process. A counting process on $[\tau, \infty)$ such that $N^\tau(t)$ is the number of offspring at time $t \leqslant \tau + L^\tau$. More on this process in section 3.1.
- $\chi^\tau := \{\chi^\tau(t)\}_{t \geqslant \tau}$ Random characteristic. A stochastic process on $[\tau, \infty)$, this is simply a bookkeeping device for what to count in the process.

If we were not in an epidemic context, the random elements L^τ and $\{N^\tau(t)\}_{t \geqslant 0}$ might more appropriately be called respectively the lifetime and reproduction counting process

For completeness, we extend the stochastic processes onto the whole real line by letting $N^\tau(t) = \chi^\tau(t) = 0$ for $t < \tau$. We now define the time-varying Crump-Mode-Jagers process counted with the random characteristic χ^τ as

$$Z(t, \tau_{i_0}) = \sum_{i \in \mathcal{I}} \chi_i^{\tau_i}(t), \quad t \geqslant \tau_{i_0} \geqslant 0 \tag{8}$$

We understand this process as counting the individuals in the population having the characteristic χ^τ at the calendar time t. I.e. it is natural, but not actually necessary for our exposition, that $\chi^\tau(t) \in \{0, 1\}$. In our epidemic context there are two innate random characteristics to study, namely to count everyone who have been infected up to and including the current time, that is the cumulative incidence,

$$\chi^\tau_{CI}(u) = \mathbb{1}_{[\tau,\infty)}(u), \quad u \in \mathbb{R} \tag{9}$$

and counting everyone infected at the moment, that is prevalence,

$$\chi^\tau_{PR}(u) = \mathbb{1}_{[\tau,\tau+L^\tau)}(u), \quad u \in \mathbb{R} \tag{10}$$

Note the difference in time indexing from [10], here the infection counting process and the random characteristic are processes on $[\tau, \infty)$ not $[0, \infty)$, thus (8) is formulated slightly differently.

Our goal is to study $Z(t, \tau_{i_0})$ through its mean and higher moments, as to establish an expression for the mean cumulative incidence and prevalence as well as their uncertainty, under one general framework. The seeming simplicity of eq. (8) is deceiving as it hides the complexities of the infectious period and infectious counting process, but these are of course essential in the investigation of the process.

3.1 The infection counting process

For a given individual $i \in \mathcal{I}$ born at time τ_i with infectious period $L_i^{\tau_i}$ (still a random variable) will now infect new individuals according to the counting process $N_i^{\tau_i}$, such that the individual i has infected $N_i^{\tau_i}(t)$ new individuals at time $t \leqslant \tau_i + L_i^{\tau_i}$, after that, no new offspring will arise from the individual. The process itself can be either dependent or independent of the infectious period. If the process is independent of $L_i^{\tau_i}$ we will stop counting infections after time $\tau_i + L_i^{\tau_i}$. For simplicity of notation, we will just consider a given triplet of random elements $\left\{(L_i^\tau, N_i^\tau, \chi_i^\tau)\right\}_{\tau \geqslant 0}$ for a given infection time $\tau \geqslant 0$ from now on.

The time-varying Crump-Mode-Jagers process, as outlined above is a generalization of several known branching processes including Markov branching processes and the Bellman-Harris process [7],

Example 3.1.1 (Bellman-Harris process)**.** In an epidemic context, we characterize the (time-varying) Bellman-Harris process through individuals infecting all their offspring at the moment they stop being infectious. That is at time $\tau + L^\tau$ individual i infects a random number of individuals, let's say $\xi(\tau + L^\tau)$, where $\{\xi(t)\}_{t \geqslant 0}$ is a stochastic process taking values in $\mathbb{N}_0$ independent of $\{L^\tau\}_{\tau \geqslant 0}$. We then have our infection counting process given by

$$N^\tau(t) = \mathbb{1}_{[L^\tau,\infty)}(t)\xi(\tau + L^\tau) \tag{11}$$

If we were to let the distribution of L^τ and $\{\xi(t)\}_{t \geqslant 0}$ be independent of the time-parameters τ and t respectively, we would recover the classical non-time-varying Bellman-Harris process [5].

As we wish to model an infection process, the assumption of Example 3.1.1 that an individual infects all its offspring at a single time is not realistic. Rather we will stick with a more general counting process driven by an underlying Lévy process. The derivation of the probability generating function which follows in section 4 relies on the following assumptions on the infection counting process

Assumption 3.1.1 (Independence). $N^\tau(t)$ *is independent of* L^τ.

Assumption 3.1.2 (Continuous in probability). N^τ *is continuous in probability, that is for all* $\varepsilon > 0$

$$\lim_{\delta \to 0} \mathbb{P}\Big(\big|N^\tau(t+\delta) - N^\tau(t)\big| \geqslant \varepsilon\Big) = 0, \qquad \forall t \geqslant \tau \geqslant 0$$

An important consequence of this assumption is that any discrete formulation of the epidemic does not fit in the model.

Assumption 3.1.3 (Underlying Lévy process). N^τ *can be constructed from a Lévy process* $\mathcal{N} := \{\mathcal{N}(t)\}_{t \geqslant 0}$ *and some non-negative rate function* $r^\tau(t)$ *as*

$$N^\tau(t) = \mathcal{N}\left(\int_\tau^t r^\tau(t)\,ds\right)$$

The counting process associated with each individual is independent and thus can be constructed from different independent copies of $\mathcal{N}$. *Remark that not any Lévy process would define a suitable counting process,* $\mathcal{N}$ *has to be a counting process itself.*

We will consider two cases of N^τ satisfying assumptions 3.1.1 to 3.1.3.

Example 3.1.2 (Inhomogenous Poisson process). If $\mathcal{N}$ is a unit-rate homogeneous Poisson process on $[0,\infty)$ independent of $\{L^\tau\}_{\tau \geqslant 0}$, such that $\mathcal{N}(t) \sim \text{Pois}(t)$. In particular, $\mathcal{N}$ is a Lévy counting process, and the infectious counting process N^τ defined by assumption 3.1.3 is then an inhomogeneous Poisson process with rate function r^τ. A defining feature of the Poisson process is that it only jumps with size $+1$ [16]. In an epidemic context, this corresponds to an individual only being able to infect one person at a time, if we want to consider infection events with more than one infected individual we could consider the infection process as a compound Poisson process.

Example 3.1.3 (Negative binomial process). Let $Y_1, Y_2, \ldots$ be a sequence of i.i.d. variables such that $Y_i \sim \text{Logarithmic}(1-p)$, and let $\mathcal{M} = \{\mathcal{M}(t)\}_{t \geqslant 0}$ be a homogeneous Poisson process with rate $-\phi \log(p)$ for $p \in (0,1), \phi > 0$, that is $\mathcal{M}(t) \sim \text{Pois}(-\phi \log(p) t)$. Assume further that $\mathcal{M}$ is independent of $Y_1, Y_2, \ldots$, we then consider the compound Poisson process $\mathcal{N} := (\mathcal{N}(t))_{t \geqslant 0}$ given by

$\log(p) < 0$ *as* $p \in (0,1)$, *hence the rate is positive as it should be.*

$$\mathcal{N}(t) = \sum_{i=1}^{\mathcal{M}(t)} Y_i. \tag{12}$$

Then by Example 2.2.2 we get $\mathcal{N}(t) \sim \text{NegBin}(\phi t, p)$, hence we will call this process a homogeneous negative binomial process. As $\mathcal{N}$ is a compound Poisson process, it is furthermore a Lévy process [16, Prop. 1.3.11] and since $Y_i \geqslant 1$ and $\mathcal{M}(t)$ is increasing $\mathcal{N}$ defines a counting process. The infectious counting process N^τ defined by assumption 3.1.3 is then an inhomogeneous negative binomial process.

3.2 Derived processes

Our derivation of the generating function of $Z(t,\tau)$ is aided by assumption 3.1.3 as the derived counting process of infection events, the process counting the times at which $N^\tau(t)$ jumps disregarding the size of each jump, is an inhomogeneous Poisson process. This can be seen as follows.

Consider the process of jump events $J_\mathcal{N}$ of our Lévy process $\mathcal{N}$, given by

$$J_\mathcal{N}(t) := \left|\{u \in [0,t] \mid \mathcal{N}(u) - \mathcal{N}(u-) > 0\}\right| \tag{13}$$
$$= \left|\{u \in [0,t] \mid \mathcal{N}(u) - \mathcal{N}(u-) \geqslant 1\}\right| \tag{14}$$

where $\mathcal{N}(u-) = \lim_{t \to u^-} \mathcal{N}(t)$ denotes the limit as approaching u from the left, and $|\cdot|$ is the cardinality of a set. The two sets above are the same as $\mathcal{N}$ has positive integer-valued increments. $\mathcal{N}$ is Lévy so by [16, thm. 2.3.5] $J_\mathcal{N}$ is a Poisson process with some rate κ.

Remember that counting processes are always right-continuous, $\mathcal{N}(u+) = \mathcal{N}(u)$.

Now let J^τ be the process defined by

$$J^\tau(t) := J_\mathcal{N}\left(\int_\tau^t r^\tau(s)\,ds\right) \tag{15}$$

i.e. J^τ is the counting process of infection events in N^τ. Since $J_\mathcal{N}$ is a Poisson process, J^τ is an inhomogeneous Poisson process with rate $\kappa r^\tau(t)$. In particular, we have that $J^\tau(t) \sim \text{Pois}(\kappa\lambda^\tau(t))$ where $\lambda^\tau(t) = \int_\tau^t r^\tau(s)\,ds$, so by Example 2.2.1 $J^\tau(t)$ must have generating function

$$\mathcal{J}^\tau(t;s) = \exp\left(\kappa\lambda^\tau(t)(s-1)\right) \tag{16}$$

for $t \geqslant \tau$.

J^τ alone will not characterize the whole infection counting process N^τ as multiple infections at the same infection event are allowed in the framework, hence we introduce the process of jump sizes Y^τ for each infection event time $\tau \in \{u \in [0,\infty) \mid \mathcal{N}(u) - \mathcal{N}(u-) > 0\}$, given by

$$Y^\tau(t) := N^\tau(t) - N^\tau(t-). \tag{17}$$

We will assume that $t \mapsto Y^\tau(t)$ is constant, hence each Y^τ is just considered a random variable, not a stochastic process. This interprets

as the number of infections at a given infection event is not dependent on how long an individual has been infected, only at which time it was infected. We denote the generating function of Y^τ by $y^\tau(s)$.

Example 3.2.1 (Inhomogeneous Poisson process, cont'd)**.** With $\mathcal{N}$ and N^τ as defined in Example 3.1.2, we consider the derived processes. As $\mathcal{N}$ is a Poisson process, it only has jumps of size one, i.e. $\mathcal{N}(u) - \mathcal{N}(u-) = 1$ when $u \geqslant 0$ is a jump time. But the process of jump events $J_\mathcal{N}$ is then given by

$$J_\mathcal{N}(t) = \left|\{\, u \in [0,t] \mid \mathcal{N}(u) - \mathcal{N}(u-) = 1 \,\}\right| = \mathcal{N}(t) \tag{18}$$

and the rate of $J_\mathcal{N}$ is thus $\kappa = 1$ as $\mathcal{N}$ was unit-rate. Consequently, we have $J^\tau = N^\tau$ is an inhomogeneous Poisson process with rate function r^τ. Again from the fact that Poisson processes (homogeneous or not) only have jumps of size one, we have $Y^\tau = 1$ for an infection time τ, we thus have generating function

$$y^\tau(s) = \mathbb{E}s^{Y^\tau} = \mathbb{E}s^1 = s. \tag{19}$$

Example 3.2.2 (Negative binomial process, cont'd)**.** With $\mathcal{N}$ and N^τ as defined in Example 3.1.3, we consider the derived processes. We wish to see that $J_\mathcal{N}$ is a Poisson process with rate $\kappa = -\phi\log(p)$. A Poisson process, $\mathcal{M}$ with rate κ can be characterized by being Lévy and having the properties [17, Thm. 2.1.1]

- $\mathbb{P}(\mathcal{M}(t+h) - \mathcal{M}(t) = 0) = 1 - \kappa h + o(h)$
- $\mathbb{P}(\mathcal{M}(t+h) - \mathcal{M}(t) = 1) = \kappa h + o(h)$
- $\mathbb{P}(\mathcal{M}(t+h) - \mathcal{M}(t) > 1) = o(h)$

hence if we can write out these for $J_\mathcal{N}$ we can determine κ.

We have $\mathcal{N}(h) \sim \mathrm{NegBin}(\phi h, p)$, and as $\mathcal{N}$ is Lévy and thus stationary we have

$$\mathbb{P}(\mathcal{N}(t+h) - \mathcal{N}(t) = 0) = \mathbb{P}(\mathcal{N}(h) = 0) = p^{\phi h} \tag{20}$$
$$\mathbb{P}(\mathcal{N}(t+h) - \mathcal{N}(t) > 0) = \mathbb{P}(\mathcal{N}(h) > 0) = 1 - p^{\phi h} \tag{21}$$

now using the Taylor series of exp we have

$$p^{\phi h} = \exp\left(\phi h \log(p)\right) = 1 + \phi\log(p)h + o(h) \tag{22}$$
$$1 - p^{\phi h} = 1 - (1 + \phi\log(p)h) + o(h) = -\phi\log(p)h + o(h) \tag{23}$$

But the events $(\mathcal{N}(h) = 0)$ and $(J_\mathcal{N}(h) = 0)$ must be the same as they both mean that no jumps have happened in $\mathcal{N}$ over the interval $[0,h]$, likewise their complements must also be equal, hence

$$\mathbb{P}(J_\mathcal{N}(t+h) - J_\mathcal{N}(t) = 0) = \mathbb{P}(J_\mathcal{N}(h) = 0) = 1 - (-\phi\log(p))h + o(h) \tag{24}$$
$$\mathbb{P}(J_\mathcal{N}(t+h) - J_\mathcal{N}(t) = 1) = \mathbb{P}(J_\mathcal{N}(h) = 1) = -\phi\log(p)h + o(h) \tag{25}$$

From this we conclude that J_N is a Poisson process with rate $\kappa = -\phi \log(p)$ and that J^τ is an inhomogeneous Poisson process with rate function $-\phi \log(p) r^\tau$. The jumps of N^τ must then be of size $Y^\tau \sim \text{Logarithmic}(1-p)$ for any infection time τ by the compound Poisson construction, and as noted in Example 2.2.2 we have generating function

$$y^\tau(s) = \frac{\log(1-(1-p)s)}{\log(p)}. \tag{26}$$

Note that if we parametrize p through ϕ as $p = \phi/(1+\phi)$ we can recover the Poisson case of Example 3.2.1 by letting $\phi \to \infty$.

3.3 The rate function

In an epidemiological context we can factorize the rate function $r^\tau(t)$, i.e. the rate of infection events at time t from an individual infected at time τ, into a population level rate of infection events at time t, call it $\rho(t)$, and an individual infectiousness at time t for an individual infected at time τ, call it $\nu(t-\tau)$ for $t \geqslant \tau$. Now we will assume that

$$r^\tau(t) = \rho(t)\nu(t-\tau) \tag{27}$$

The interpretation above entails that ν is a density, $\int_0^\infty \nu(t)\,dt = 1$, otherwise, ρ cannot be interpreted as a rate. With ν being a density, we introduce its distribution function $\mathcal{V}(t) = \int_0^t \nu(s)\,ds$.

4 PGF OF THE TIME-VARYING CMJ PROCESS

We will now derive an integral equation for the probability generating function of the process $Z(t, \tau_{i_0})$ developed in section 3. Thus let $F(t, \tau_{i_0}; s) = F(t, \tau_{i_0}) := \mathbb{E}\left[s^{Z(t,\tau_{i_0})}\right]$, where we will supress dependence on s from the notation and let $\tau = \tau_{i_0}$ denote the infection time of the index case.

We start the derivation by conditioning on the infectious period of the index case $L^\tau \sim G^\tau$ and using the *law of total expectation (lote)*, and split up the integral according to whether the index case is still infected at time t

$$\begin{aligned}
F(t,\tau) &= \mathbb{E}\left[s^{Z(t,\tau)}\right] = \mathbb{E}\left[\mathbb{E}\left[s^{Z(t,\tau)} \,\middle|\, L^\tau\right]\right] \\
&= \int_0^\infty \mathbb{E}\left[s^{Z(t,\tau)} \,\middle|\, L^\tau = u\right] dG^\tau(u) \\
&= \int_{t-\tau}^\infty \mathbb{E}\left[s^{Z(t,\tau)} \,\middle|\, L^\tau = u\right] dG^\tau(u) && (28) \\
&\quad + \int_0^{t-\tau} \mathbb{E}\left[s^{Z(t,\tau)} \,\middle|\, L^\tau = u\right] dG^\tau(u) && (29)
\end{aligned}$$

We will now consider each integral (28) and (29) separately.

4.1 Integral (28)

For integral (28) we have $L^\tau \geqslant t-\tau$, i.e. the index case is still infectious and able to infect others at time t. So the number of offspring from the index case at time t is independent of the future value of L^τ since all infections from $\{N^\tau(s)\}_{\tau \leqslant s \leqslant t}$ are counted irrespectively of the value of L^τ. That is, on the event $(L^\tau \geqslant t-\tau)$ we have $\mathbb{E}\left[s^{Z(t,\tau)} \,\middle|\, L^\tau\right] = \mathbb{E}\left[s^{Z(t,\tau)} \,\middle|\, L^\tau \geqslant t-\tau\right]$ is constant, and thus

$$\begin{aligned}(28) &= \int_0^{t-\tau} \mathbb{E}\left[s^{Z(t,\tau)} \,\middle|\, L^\tau \geqslant t-\tau\right] \mathrm{d}G^\tau(u) \\ &= \left(1-G^\tau(t-\tau)\right)\mathbb{E}\left[s^{Z(t,\tau)} \,\middle|\, L^\tau \geqslant t-\tau\right] \\ &= \overline{G}^\tau(t-\tau)\mathbb{E}\left[s^{Z(t,\tau)} \,\middle|\, L^\tau \geqslant t-\tau\right] \qquad (30)\end{aligned}$$

letting $\overline{G}^\tau$ be the survival function of L^τ.

Working on the conditional expectation $\mathbb{E}\left[s^{Z(t,\tau)} \,\middle|\, L^\tau \geqslant t-\tau\right]$, we assume in the following that $L^\tau \geqslant t-\tau$. The number of infection events caused by the index case at time t is then $J^\tau(t)$. We can then denote the times at which the infection events occurred by $K_1, \ldots, K_{J^\tau(t)} \in [\tau, t]$, the indexing here is assumed to be random such that we don't necessarily have any ordering of the K_i's based on their index. From section 3.2 we know that J^τ is an inhomogeneous Poisson process, and by assumption 3.1.2 N^τ is continuous in probability (see remark 4.1.1), so the event-times of J^τ must be iid. with common distribution $h_K^\tau(\cdot\,; t) \cdot m$, where,

Here m denotes the Lebesgue measure on $\mathbb{R}$ and $h \cdot m$ denotes a measure with density h.

$$h_K^\tau(k; t) = \frac{r^\tau(k)}{\int_\tau^t r^\tau(k)\,\mathrm{d}k}\,\mathbb{1}_{[\tau,t]}(k) = \frac{r^\tau(k)}{\lambda^\tau(t)}\,\mathbb{1}_{[\tau,t]}(k). \qquad (31)$$

Note that at each infection event time K_i the index case infects Y^{K_i} individuals, which is possibly more than one.

The self-similarity of $Z(t,\tau)$ says that each newly infected individual of the index case behaves as an index case in a new independent epidemic started at their infection time. This allows us to write the branching process as

$$Z(t,\tau) = \chi^\tau(t) + \sum_{i=1}^{J^\tau(t)} \sum_{j=1}^{Y^{K_i}} Z_{ij}(t, K_i) \qquad (32)$$

here each Z_{ij} is an independent copy of Z, such that for each of the infected individuals $j \leqslant Y^{K_i}$ at infection event time K_i follows Z_{ij}. The summation holds as each infected individual behaves independently once infected. Note that $\chi^\tau(t) = 1$ for both $\chi^\tau = \chi^\tau_{CI}$ and $\chi^\tau = \chi^\tau_{PR}$ (eqs. (9) and (10)) since $L^\tau \geqslant t-\tau$, which are the two primary cases of interest.

Now for each $i \leqslant J^\tau(t)$ the Z_{ij}'s are iid. with $Z_{ij} \overset{\mathcal{D}}{=} Z_j$ for appropriate $Z_j \overset{\mathcal{D}}{=} Z$ which are independent and likewise K_i's are iid. with $K_i \overset{\mathcal{D}}{=} K$ for some $K \sim h_K \cdot m$, for $i \leqslant J^\tau(t)$

$$\sum_{j=1}^{Y^{K_i}} Z_{ij}(t, K_i) \overset{\mathcal{D}}{=} \sum_{j=1}^{Y^K} Z_j(t, K) \tag{33}$$

is iid. as Y^K, the number of infected individuals at event time K is independent of the further trajectory of the process Z_j. We can therefore use eq. (2) to write the conditional generating function as

$$\mathbb{E}\left[s^{Z(t,\tau)} \,\middle|\, L \geqslant t-\tau\right] = s^{\chi^\tau(t)} g^\tau\left(t; \mathbb{E}\left[s^{\sum_{j=1}^{Y^K} Z_j(t,K)}\right]\right) \tag{34}$$

Then using *lote*, conditioning on the value of K we have for the inner expectation,

$$\mathbb{E}\left[s^{\sum_{j=1}^{Y^K} Z_j(t,K)}\right] = \int_\tau^t \mathbb{E}\left[s^{\sum_{j=1}^{Y^k} Z_j(t,k)}\right] h_K^\tau(k;t)\,dk \tag{35}$$

$$= \int_\tau^t y^\tau(F(t,k)) h_K^\tau(k;t)\,dk \tag{36}$$

again using eq. (2) since the Z_j's are iid. and independent of Y^k. Now plugging this into eq. (34) and using that J^τ is an inhomogeneous Poisson process with generating function given by eq. (16), we have for χ^τ counting either cumulative incidence or prevalence, such that $\chi^\tau(t) = 1$, that

$$\mathbb{E}\left[s^{Z(t,\tau)} \,\middle|\, L \geqslant t-\tau\right] = s \exp\left[\kappa\lambda^\tau(t)\left(\int_\tau^t y^\tau(F(t,k)) h_K^\tau(k;t)\,dk - 1\right)\right] \tag{37}$$

and finally, we have an expression for integral (28)

$$(28) = s\overline{G}^\tau(t-\tau) \exp\left[\kappa\lambda^\tau(t)\left(\int_\tau^t y^\tau(F(t,k)) h_K^\tau(k;t)\,dk - 1\right)\right] \tag{38}$$

Remark 4.1.1. Assumption 3.1.2 is necessary for $K_1, \ldots, K_{J^\tau(t)}$ to be iid. Consider N^τ with positive probability of jumping at time $k \geqslant \tau$, then $\mathbb{P}(K_2 = k \mid K_1 = k) = 0 \neq \mathbb{P}(K_2 = k)$ as the K_i's must be distinct, but this is contradictory to the K_i's being iid. So N^τ has to be continuous in probability, where we have $\mathbb{P}(K_i = k) = 0$ anyway for all $k \geqslant \tau$.

4.2 Integral (29)

A very similar argument can be done for integral (29), we will only sketch out the differences from what's done above, a thourough derivation can be found in [4, app. B]. Importantly for integral (29) we have

$L^\tau < t - \tau$, that is, the index case is no longer able to infect others at time t. We cannot just do something equivalent to eq. (30) since the number of offspring from the index individual is now dependent on the value of L^τ, as only the jumps of N^τ that happened before time $\tau + L^\tau$ are counted. Instead, the number of infection events from the index case is $J^\tau(\tau + L^\tau)$ for all $t > \tau + L^\tau$. Working on the conditional expectation $\mathbb{E}\left[s^{Z(t,\tau)} \,\middle|\, L^\tau = u\right]$ for some $u < t - \tau$, we assume in the following that $L^\tau = u$.

Now the derivation is identical to that of integral (28) changing out t for $\tau + L^\tau = \tau + u$ whenever t was used as the last possible infection time for the index case, and noting that $\chi^\tau_{CI}(t) = 1$ and $\chi^\tau_{PR}(t) = 0$ for $t > \tau + L^\tau$, so for counting prevalence, we have

$$\mathbb{E}\left[s^{Z(t,\tau)} \,\middle|\, L^\tau = u\right] = \exp\left[\kappa\lambda^\tau(\tau+u)\left(\int_\tau^{\tau+u} y^\tau(F(t,k))h^\tau_K(k;u)\,dk - 1\right)\right] \tag{39}$$

We then have an expression for integral (29) when counting prevalence,

$$(29) = \int_0^{t-\tau} \exp\left[\kappa\lambda^\tau(\tau+u)\left(\int_\tau^{\tau+u} y^\tau(F(t,k))h^\tau_K(k;u)\,dk - 1\right)\right] dG^\tau(u) \tag{40}$$

and multiplying by s if counting cumulative incidence.

4.3 Collecting the integrals

We can now use the expressions for integrals (28) and (29) to get an expression for our generating function for counting prevalence,

$$\begin{aligned} F(t,\tau) &= s\overline{G}^\tau(t-\tau)\exp\left[\kappa\lambda^\tau(t)\left(\int_\tau^t y^\tau(F(t,k))h^\tau_K(k;t)\,dk - 1\right)\right] \\ &\quad + \int_0^{t-\tau} \exp\left[\kappa\lambda^\tau(\tau+u)\left(\int_\tau^{\tau+u} y^\tau(F(t,k))h^\tau_K(k;u)\,dk - 1\right)\right] dG^\tau(u) \end{aligned} \tag{41}$$

Now using that $h^\tau_K(k;t) = r^\tau(k)/\lambda^\tau(t) = r^\tau(k)/\int_\tau^t r^\tau(k)\,dk$ for $k \in [\tau, t]$, we can write

$$\begin{aligned} F(t,\tau) &= s\overline{G}^\tau(t-\tau)\exp\left[\kappa\left(\int_\tau^t y^\tau(F(t,k))r^\tau(k)\,dk - \lambda^\tau(t)\right)\right] \\ &\quad + \int_0^{t-\tau} \exp\left[\kappa\left(\int_\tau^{\tau+u} y^\tau(F(t,k))r^\tau(k)\,dk - \lambda^\tau(\tau+u)\right)\right] dG^\tau(u) \\ &= s\overline{G}^\tau(t-\tau)\exp\left[\kappa\left(\int_\tau^t y^\tau(F(t,k))r^\tau(k)\,dk - \int_\tau^t r^\tau(k)\,dk\right)\right] \\ &\quad + \int_0^{t-\tau} \exp\left[\kappa\left(\int_\tau^{\tau+u} y^\tau(F(t,k))r^\tau(k)\,dk - \int_\tau^{\tau+u} r^\tau(k)\,dk\right)\right] dG^\tau(u) \\ &= s\overline{G}^\tau(t-\tau)\exp\left[\kappa\int_\tau^t \left(y^\tau(F(t,k)) - 1\right)r^\tau(k)\,dk\right] \\ &\quad + \int_0^{t-\tau} \exp\left[\kappa\int_\tau^{\tau+u} \left(y^\tau(F(t,k)) - 1\right)r^\tau(k)\,dk\right] dG^\tau(u) \end{aligned} \tag{42}$$

We could alternatively translate the dk integrals by $-\tau$ such that all lower integral bounds are 0, calling the variable of integration ν instead to avoid confusion of where K lives,

$$F(t,\tau) = s\overline{G}^{\tau}(t-\tau)\exp\left[\kappa\int_0^{t-\tau}\left(y^{\tau}(F(t,\tau+\nu))-1\right)r^{\tau}(\tau+\nu)\,d\nu\right] + \int_0^{t-\tau}\exp\left[\kappa\int_0^{u}\left(y^{\tau}(F(t,\tau+\nu))-1\right)r^{\tau}(\tau+\nu)\,d\nu\right]\,dG^{\tau}(u) \tag{43}$$

Furthermore, to streamline the integrals, and help us in the coming discretization of the integrals, we can consider the integrals inside the exponential as Riemann-Stieltjes integrals like the outer integrals. If we use the factorization of $r^{\tau}(\tau+\nu) = \rho(\tau+\nu)\nu(\nu)$ from section 3.3 where ν is a density with distribution function $\mathcal{V}$, we could write

$$F(t,\tau) = s\overline{G}^{\tau}(t-\tau)\exp\left[\kappa\int_0^{t-\tau}\left(y^{\tau}(F(t,\tau+\nu))-1\right)\rho(\tau+\nu)\,d\mathcal{V}(\nu)\right] + \int_0^{t-\tau}\exp\left[\kappa\int_0^{u}\left(y^{\tau}(F(t,\tau+\nu))-1\right)\rho(\tau+\nu)\,d\mathcal{V}(\nu)\right]\,dG^{\tau}(u) \tag{44}$$

Remark 4.3.1. Using Riemann-Stieltjes integrals turns out to give some numerical stability when performing the numerical integration presented in section 5.1, but it also allows for the distributions of the infectious period and individual infectiousness over time to not be absolutely continuous. This makes it straightforward to work with distribution with point masses and jumps, as well as distributions that we might only be able to simulate from, e.g. complex mixtures or alike, where we can simply integrate with respect to the empirical distribution function [18].

As noted in example 3.1.2 and 3.1.3 we devote our interest mostly to the two cases of N^{τ} being an inhomogeneous Poisson process or a negative binomial process. In both cases, we can simplify eq. (43) slightly by writing out the form of y^{τ}.

Example 4.3.1 (Inhomogeneous Poisson process, cont'd.)**.** We continue from Example 3.2.1, where we found that $\kappa = 1$ and $y^{\tau}(s) = s$, hence we can write eq. (44) as

$$F(t,\tau) = s\overline{G}^{\tau}(t-\tau)\exp\left[\int_0^{t-\tau}\left(F(t,\tau+\nu)-1\right)\rho(\tau+\nu)\,d\mathcal{V}(\nu)\right] + \int_0^{t-\tau}\exp\left[\int_0^{u}\left(F(t,\tau+\nu)-1\right)\rho(\tau+\nu)\,d\mathcal{V}(\nu)\right]\,dG^{\tau}(u) \tag{45}$$

Example 4.3.2 (Negative binomial process, cont'd.)**.** We continue from Example 3.2.1, where we found that $\kappa = -\phi \log(p)$ and $\mathcal{Y}^\tau(s) = s$, hence we can write eq. (44) as

$$\begin{aligned} F(t,\tau) = \\ & s\overline{G}^\tau(t-\tau) \exp\left[-\phi \log(p) \int_0^{t-\tau} \left(\tfrac{\log(1-(1-p)F(t,\tau+\nu))}{\log(p)} - 1\right) \rho(\tau+\nu)\, d\mathcal{V}(\nu)\right] \\ & + \int_0^{t-\tau} \exp\left[-\phi \log(p) \int_0^{u} \left(\tfrac{\log(1-(1-p)F(t,\tau+\nu))}{\log(p)} - 1\right) \rho(\tau+\nu)\, d\mathcal{V}(\nu)\right] dG^\tau(u) \end{aligned}$$

or taking $\log(p)$ inside the integral,

$$\begin{aligned} F(t,\tau) = \\ & s\overline{G}^\tau(t-\tau) \exp\left[-\phi \int_0^{t-\tau} \Big(\log\big(1-(1-p)F(t,\tau+\nu)\big) - \log(p)\Big) \rho(\tau+\nu)\, d\mathcal{V}(\nu)\right] \\ & + \int_0^{t-\tau} \exp\left[-\phi \int_0^{u} \Big(\log\big(1-(1-p)F(t,\tau+\nu)\big) - \log(p)\Big) \rho(\tau+\nu)\, d\mathcal{V}(\nu)\right] dG^\tau(u) \end{aligned} \tag{46}$$

Remark 4.3.2. Recall that all the generating function equations above are for counting prevalence, if we were to count cumulative incidence instead, we have to multiply the second integral by s to account for the initial individual being counted at all times.

The generating functions eqs. (45) and (46) as well as their corresponding equations for counting cumulative incidence can be viewed as specific instances of the integral equation

The integral equation could be made more general, by substituting $\overline{G}^\tau$, ρ, $\mathcal{V}$ and G^τ by appropriate generic functions.

$$\begin{aligned} F(t,\tau;s) = \overline{G}^\tau(t-\tau) q_1\left(\int_0^{t-\tau} \psi\big(F(t,\tau+\nu;s)\big)\rho(\tau+\nu)\, d\mathcal{V}(\nu);\, s\right) \\ + \int_0^{t-\tau} q_2\left(\int_0^{u} \psi\big(F(t,\tau+\nu;s)\big)\rho(\tau+\nu)\, d\mathcal{V}(\nu);\, s\right) dG^\tau(u) \end{aligned} \tag{47}$$

again we might suppress the dependence on s. We recover the generating function equation with

$$q_1(z;\, s) := se^z \tag{48}$$

$$q_2(z;\, s) := \begin{cases} e^z, & \text{if } \chi^\tau = \chi^\tau_{PR} \\ se^z, & \text{if } \chi^\tau = \chi^\tau_{CI} \end{cases} \tag{49}$$

$$\psi(z) := \begin{cases} z-1, & \text{for } N^\tau \text{ Poisson proc.} \\ -\phi\big(\log(1-(1-p)z) - \log(p)\big), & \text{for } N^\tau \text{ NegBin proc.} \end{cases} \tag{50}$$

5 SOLVING THE PGF EQUATION

As mentioned we are in practice only interested in the process $Z(t,0)$ and its generating function $F(t,0)$ as this corresponds to an epidemic started at time $\tau = 0$. However, due to the self-similarity property

used in deriving the generating function equation, and as visible in eq. (47), to compute $F(t,0)$ we need to evaluate $F(t,\tau)$ over the whole characteristic curve being the line from the data curve $t=\tau$, on which we know $F(t,\tau)=F(t,t)=s$, to $(t,0)$.

The terms characteristic curve and data curve are borrowed from the theory of PDEs and the method of characteristics [19].

Note in particular that to evaluate $F(t',0)$ at a given time $t'>0$ we only need to evaluate the function $\tau\mapsto F(t',\tau)$ for all $\tau\leqslant t'$, hence we can for each $t\geqslant 0$ consider $\tau\mapsto F(t,\tau)$ as a single-variable function. Instead of doing this directly, we simplify the integral bounds, such that they only depend on one variable, and will therefore consider the auxiliary single-variable function for given time $t\geqslant 0$, $F_t(\sigma)=F(t,t-\sigma)$, i.e. we can consider $\sigma=t-\tau$ the time since the index case was infected at time t. By eq. (47) we have the corresponding integral equation for $F_t(\sigma)$ for $t\geqslant\sigma\geqslant 0$,

$$\begin{aligned}F_t(\sigma)=\overline{G}^{t-\sigma}(\sigma)q_1\left(\int_0^\sigma\psi\big(F_t(\sigma-\nu;s)\big)\rho(t-\sigma+\nu)\,d\mathcal{V}(\nu);\,s\right)\\+\int_0^\sigma q_2\left(\int_0^u\psi\big(F_t(\sigma-\nu;s)\big)\rho(t-\sigma+\nu)\,d\mathcal{V}(\nu);\,s\right)\,dG^{t-\sigma}(u).\end{aligned}\tag{51}$$

For given $t\geqslant 0$, evaluating $F(t,0)$ now corresponds to solving $F_t(t)=F(t,t-t)=F(t,0)$. We know the values of $F_t(0)=F(t,t-0)=F(t,t)=s$, so we need to compute $F_t(\sigma)$ over the characteristic line from $\sigma=0$ to t (from $(t,\sigma)=(t,0)$ to (t,t) if we let t vary freely).

5.1 Discretization

In practice we will approximate $F_t(t)$ over a discrete time-grid $t=0,\Delta,\ldots,N\Delta$ with $N\in\mathbb{N}$ and $\Delta>0$. By the discussion above, to approximate $F_{n\Delta}(n\Delta)$ for given $n\leqslant N$ we need to evaluate $F_{n\Delta}(\sigma)$ over the characteristic line from $\sigma=0$ to $n\Delta$, which again is done over a discrete time-grid $\sigma=0,\Delta,\ldots,n\Delta$. We approximate $F_{n\Delta}(i\Delta)$ recursively, using Riemann-Stieltjes sums to approximate the integrals, for $i\leqslant n\leqslant N$ by,

$F_0(0)=s$ is known.

$$\begin{aligned}\widetilde{F}_{n\Delta}(i\Delta):=\overline{G}^{(n-i)\Delta}(i\Delta)q_1\left(\sum_{k=1}^{i}\psi\Big(\widetilde{F}_{n\Delta}\big((i-k)\Delta\big)\Big)\rho\big((n-i+k)\Delta\big)\Delta\mathcal{V}_k\right)\\+\sum_{j=1}^{i}q_2\left(\sum_{k=1}^{j-1}\psi\Big(\widetilde{F}_{n\Delta}\big((i-k)\Delta\big)\Big)\rho\big((n-i+k)\Delta\big)\Delta\mathcal{V}_k\right)\Delta G_j^{(n-i)\Delta}\end{aligned}\tag{52}$$

Recall that an empty sum $\sum_{k=1}^{0}$ is defined to be zero.

where

$$\Delta\mathcal{V}_k:=\mathcal{V}(\nu_k)-\mathcal{V}(\nu_{k-1})\tag{53}$$

$$\Delta G_j^{(n-i)\Delta}:=G^{(n-i)\Delta}(\nu_j)-G^{(n-i)\Delta}(\nu_{j-1})\tag{54}$$

where ν_k is the k'th element of a partition $\mathcal{P}$ of the appropriate integration domain satisfying that $k\Delta\in[\nu_{k-1},\nu_k]$ for all k. A natural

partition of the domain $[0, i\Delta]$ is the equidistant $\mathcal{P} = (0, 1, ..., i\Delta)$, which results in eq. (52) using simple right Riemann-Stieltjes sums for numerical integration.

Remark 5.1.1 (Infections at infection time). We remark that in eq. (52) the inner integral of the last term in the integral equation is not discretized in a usual manner, like in the first term, although in the continuous integral version eq. (51) the integrals seem equivalent. But unlike the continuous version, we have to be explicit about inequalities and at which time points individuals start infecting others. Consider in particular the situation where the $k = i$ in eq. (52), this term corresponds to the number of infections caused by the index case at its time of infection $t = \tau$. But under this integral, we assume that $L^\tau < t - \tau = 0$, but as L^τ is a positive variable, this cannot happen, thus no infections can happen at this time, and we have to exclude that term from the summation. This is no problem in the continuous case as the event that infections happen at infection time is an event of measure zero.

6 EFFICIENT IMPLEMENTATION

We will now present two implementations of eq. (52), one direct implementation using several nested for-loops to evaluate the integrals and one vectorized version which evaluates the integrals efficiently using matrix computations.

For both implementations we note that in eq. (52) the indexation of $\widetilde{F}_{n\Delta}(\cdot)$ goes backwards $F_{n\Delta}\big((i-1)\Delta\big), F_{n\Delta}\big((i-2)\Delta\big), ..., F_{n\Delta}(0)$ for $k = 1, 2, ..., i$. This will be cumbersome when we want to store the values of $F_{n\Delta}(i\Delta)$ in a matrix. Thus we reparametrize the summations, for given $i \leqslant n$ letting $k' := i - k$ for the first summation,

$$\begin{aligned} &\sum_{k=1}^{i} \psi\Big(\widetilde{F}_{n\Delta}\big((i-k)\Delta\big)\Big)\rho\big((n-i+k)\Delta\big)\Delta\mathcal{V}_k \\ &\quad = \sum_{k'=0}^{i-1} \psi\Big(\widetilde{F}_{n\Delta}\big(k'\Delta\big)\Big)\rho\big((n-k')\Delta\big)\Delta\mathcal{V}_{i-k'} \end{aligned} \tag{55}$$

and likewise for the second summation first letting $k' := i - k$ and thereafter $j' = i - j$,

$$\begin{aligned}
&\sum_{j=1}^{i} q_2 \left(\sum_{k=1}^{j-1} \psi\left(\widetilde{F}_{n\Delta}((i-k)\Delta)\right) \rho((n-i+k)\Delta) \Delta \mathcal{V}_k \right) \Delta G_j^{(n-i)\Delta} \\
&\quad = \sum_{j=1}^{i} q_2 \left(\sum_{k'=i-j+1}^{i-1} \psi\left(\widetilde{F}_{n\Delta}(k'\Delta)\right) \rho((n-k')\Delta) \Delta \mathcal{V}_{i-k'} \right) \Delta G_j^{(n-i)\Delta} \\
&\quad = \sum_{j'=0}^{i-1} q_2 \left(\sum_{k'=j'+1}^{i-1} \psi\left(\widetilde{F}_{n\Delta}(k'\Delta)\right) \rho((n-k')\Delta) \Delta \mathcal{V}_{i-k'} \right) \Delta G_{i-j'}^{(n-i)\Delta}
\end{aligned} \tag{56}$$

combining eqs. (55) and (56) we get (deleting the primes from the summation variables)

$$\begin{aligned}
\widetilde{F}_{n\Delta}(i\Delta) = \overline{G}^{(n-i)\Delta}(i\Delta) q_1 \left(\sum_{k=0}^{i-1} \psi\left(\widetilde{F}_{n\Delta}(k\Delta)\right) \rho((n-k)\Delta) \Delta \mathcal{V}_{i-k} \right) \\
+ \sum_{j=0}^{i-1} q_2 \left(\sum_{k=j+1}^{i-1} \psi\left(\widetilde{F}_{n\Delta}(k\Delta)\right) \rho((n-k)\Delta) \Delta \mathcal{V}_{i-k} \right) \Delta G_{i-j}^{(n-i)\Delta}
\end{aligned} \tag{57}$$

6.1 Matrix setup

We will store the results and intermediary computations in matrices which will turn out fruitful, thus we start with some notational conventions. We use zero-indexing for all matrices and denote the submatrix with rows n to m and columns i to j by $A[n : m, i : j]$, if e.g. $i = j$ we simply write $A[n : m, i]$.

Using the new indexations of eq. (57) we store the values of $\widetilde{F}_{n\Delta}(i\Delta)$ in a $(N+1) \times (N+1)$ lower diagonal matrix $\widetilde{F}$, such that $\widetilde{F}[n, i] = \widetilde{F}_{n\Delta}(i\Delta)$,

We use zero-indexed matrices for notational simplicity and to make implementation in python *easy.*

$$\widetilde{F} := \begin{bmatrix}
\widetilde{F}_0(0) & 0 & 0 & \cdots & 0 \\
\widetilde{F}_\Delta(0) & \widetilde{F}_\Delta(\Delta) & 0 & \cdots & 0 \\
\widetilde{F}_{2\Delta}(0) & \widetilde{F}_{2\Delta}(\Delta) & \widetilde{F}_{2\Delta}(2\Delta) & \cdots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\widetilde{F}_{N\Delta}(0) & \widetilde{F}_{N\Delta}(\Delta) & \widetilde{F}_{N\Delta}(2\Delta) & \cdots & \widetilde{F}_{N\Delta}(N\Delta)
\end{bmatrix}$$

The first column of the matrix is populated with the known value $F_t(0) = s$ in all rows. We then follow the now discretized "characteristic curve" from left to right in each row until we hit the diagonal, that is, we can compute the value of $\widetilde{F}[n, i]$ from $\widetilde{F}[n, i-1], \widetilde{F}[n, i-2], ..., \widetilde{F}[n, 0]$ for $i \leqslant n$. Ultimately we are interested in the values $\widetilde{F}_{n\Delta}(n\Delta)$, i.e. the diagonal of $\widetilde{F}$.

All quantities but those dependent on earlier values of $\widetilde{F}_{n\Delta}$ entering into eq. (57) can be precomputed and stored in appropriate matrices according to their indexation. For the survival function of the

infectious period, we use a $(N+1)\times(N+1)$ lower diagonal matrix $\overline{G}$, such that $\overline{G}[n,i] = \overline{G}^{(n-i)\Delta}(i\Delta)$ reflecting the reverse indexation of i and $n-i$ in each row,

$$\overline{G} := \begin{bmatrix} \overline{G}^{0}(0) & 0 & 0 & \cdots & 0 \\ \overline{G}^{\Delta}(0) & \overline{G}^{0}(\Delta) & 0 & \cdots & 0 \\ \overline{G}^{2\Delta}(0) & \overline{G}^{\Delta}(\Delta) & \overline{G}^{0}(2\Delta) & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \overline{G}^{N\Delta}(0) & \overline{G}^{(N-1)\Delta}(\Delta) & \overline{G}^{(N-2)\Delta}(2\Delta) & \cdots & \overline{G}^{0}(N\Delta) \end{bmatrix}$$

Notice the Toeplitz structure of the $(n-i)\Delta$ index.

In the Riemann-Stieltjes sums ρ and $\Delta\mathcal{V}$ are always multiplied, as they are derived from the rate function r^{τ}, hence we will store these values premultiplied in a $N\times N$ upper diagonal matrix $D_{\mathcal{V}}$, such that $D_{\mathcal{V}}[n,i] = \rho\big((N-i+n)\Delta\big)\Delta\mathcal{V}_{N-i}$

$$D_{\mathcal{V}} := \begin{bmatrix} \rho(N\Delta)\Delta\mathcal{V}_{N} & \rho\big((N-1)\Delta\big)\Delta\mathcal{V}_{N-1} & \cdots & \rho(\Delta)\Delta\mathcal{V}_{1} \\ 0 & \rho(N\Delta)\Delta\mathcal{V}_{N-1} & \cdots & \rho(2\Delta)\Delta\mathcal{V}_{1} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \rho(N\Delta)\Delta\mathcal{V}_{1} \end{bmatrix}$$

And finally, we store the values of ΔG in a $N\times N$ upper diagonal matrix D_G like $D_{\mathcal{V}}$ such that $D_G[n,i] := \Delta G^{n\Delta}_{N-i}$,

$$D_G := \begin{bmatrix} \Delta G^{0}_{N} & \Delta G^{0}_{N-1} & \Delta G^{0}_{N-2} & \cdots & \Delta G^{0}_{1} \\ 0 & \Delta G^{1}_{N-1} & \Delta G^{1}_{N-2} & \cdots & \Delta G^{1}_{1} \\ 0 & 0 & \Delta G^{2}_{N-2} & \cdots & \Delta G^{2}_{1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \Delta G^{N-1}_{1} \end{bmatrix}$$

Remark 6.1.1. At each iteration in the algorithms below, we only access parts of the matrices $\overline{G}$, $D_{\mathcal{V}}$ and D_G. In particular, we never access the upper, respectively lower triangular part of the matrices, so whether they are actually set to zero does not matter.

6.2 Algorithms

We now present the direct and vectorized algorithm for computing eq. (57). As visible from algorithm 1 we need at least two nested for loops to implement the discretization directly, depending on whether the sums are vectorized or not.

For the vectorized algorithm we need to make elementwise computations on our matrices, so for a real function f and matrix A we let $f(A)$ be the function f applied elementwise on A. Elementwise (Hadamard) multiplication is used as well, we denote this $A\odot B$ for A and B of same dimensions. For two matrices A and B with the same number of rows, we let $[A;B]$ be the column concatenation of the matrices. Furthermore, we need a few special matrix transformations

Algorithm 1 Direct implementation

Require: matrices $\overline{G}, D_{\mathcal{V}}, D_G$
Require: functions q_1, q_2 and ψ
Require: number of time steps N
Require: step size Δ
$\widetilde{F} \leftarrow 0_{(N+1)\times(N+1)}$
$\widetilde{F}[0:N, 0] = s$
for $n = 1, ..., N$ **do**
  for $i = 1, ..., n$ **do**
    $\widetilde{F}^{n,i} \leftarrow \widetilde{F}[n, 0:(i-1)]$
    $D_{\mathcal{V}}^{n,i} \leftarrow D_{\mathcal{V}}[n-i, (N-i):(N-1)]$
    $D_G^{n,i} \leftarrow D_G[n-i, (N-i):(N-1)]$
    $\widetilde{F}[n,i] \leftarrow \overline{G}[n,i] q_1\left(\sum_{k=0}^{i-1} \psi\left(\widetilde{F}^{n,i}[k]\right) D_{\mathcal{V}}^{n,i}[k]\right)$
      $+ \sum_{j=0}^{i-1} q_2\left(\sum_{k=j+1}^{i-1} \psi\left(\widetilde{F}^{n,i}[k]\right) D_{\mathcal{V}}^{n,i}[k]\right) D_G^{n,i}[j]$
  end for
end for
return $\mathrm{diag}(\widetilde{F})$

ColFlip, RowSum, RowCumSum which denote reversing the order of the columns in the matrix, summing over the elements of each row and making a cumulative sum over each row. Note that ColFlip and RowCumSum returns matrices of the same dimension as their input, while RowSum returns a vector with length equal to the number of rows in the input matrix.

The algorithms are fairly easy to implement in an interpreted language, and with the vectorization approach, we can get efficient computation without the need for compilation. Implementations and an example use case coded in python are supplied in the companion jupyter notebook which can be found as an appendix or at https://github.com/molkjar/pgf_disc.

Remark 6.2.1. Note that we in practice want to compute the probability mass function. From section 2.3 we can do this by computing the fast Fourier transform of the generating function at some appropriate points, this entails many evaluations of the generating function. Note however that our setup in algorithms 1 and 2 allows us to only compute the matrices $\overline{G}, D_{\mathcal{V}}, D_G$ once and use them in each evaluation.

Remark 6.2.2. To improve efficiency even further, especially when having to evaluate the generating function many times as mentioned above, one could take advantage of the typically light-tailed distributions G^τ and $\mathcal{V}$, resulting in the values of the matrices $D_{\mathcal{V}}$ and D_G begin very small in the upper left corner, that is for high values of t. Thus one could with only a small error in the computation of the generating function truncate these matrices.

Algorithm 2 Vectorized implementation

Require: matrices $\overline{G}, D_V, D_G$
Require: functions q_1, q_2 and ψ
Require: number of time steps N
Require: step size Δ
$\widetilde{F} \leftarrow 0_{(N+1)\times(N+1)}$
$\widetilde{F}[0:N, 0] = s$
for i=1,...,N **do**
    $B \leftarrow \psi\big(\widetilde{F}[i:N, 0:(i-1)]\big) \odot D_V[0:(N-i), (N-i):(N-1)]$
    $B_1 \leftarrow \mathrm{ColFlip}\big(\mathrm{RowCumSum}\big(\mathrm{ColFlip}(B)\big)\big)$
    $I_1 \leftarrow \overline{G}[i:N, i] \odot q_1\big(B_1[0:(N-i), 0]\big)$
    $B_2 < -\big[B_1[0:(N-i), 1:(i-1)]\,;\, 0_{(N-i+1)\times 1}\big]$
    $I_2 \leftarrow \mathrm{RowSum}\big(q_2(B_2) \odot D_G[0:(N-i), (N-i):(N-1)]\big)$
    $F[i:(N+1), i] \leftarrow I_1 + I_2$
end for
return $\mathrm{diag}(\widetilde{F})$

6.3 Discussion

In this report, we have sought to clearly outline and unify some of the results from [10, 4] and to write a simple and efficient implementation of the generating function in an interpreted language, such that practitioners easily can apply the methods. The approach uses a generalized branching process framework, which can model both cumulative incidence and prevalence, under a time-varying infection period and offspring distribution, allowing for quite realistic assumptions in the modeling process while being analytically tractable.

Importantly we have found a governing equation, not only for the expectation but for the whole distribution of our time-varying Crump-Mode-Jagers branching process through its probability-generating function. This allows us to investigate uncertainty generated in the stochastic process itself, which can have a large impact on the possible trajectories of the epidemic.

We showed how the generating function of the Crump-Mode-Jagers branching process is following a renewal-like integral equation for which we have provided a discretization scheme allowing for numerical implementation. However, implementing the discretized generating function requires several nested for-loops, which from an efficiency point of view is quite bad, at least in an interpreted language. And while it is possible to implement the discretization in a compiled language such as C++, and provide an interface for running it through a more commonly used language that more practitioners

use, extensibility would be hard for people without any knowledge of such languages.

Instead, we have provided a vectorized algorithm that can efficiently compute the probability-generating function even though implemented in an interpreted language, and we have further investigated a useful way of numerically constructing the probability mass function from this generating function. Both methods are implemented, as well as a non-vectorized version of the generating function, in the companion jupyter notebook, where there is also a showcase of the method under a given branching process model.

Future directions for practitioners to start using methods like the present one is making further examples of real-world use cases, showing how different models and model parameters lead to different distributions and performing inference over the model. One particular use case could be exploiting previously estimated time-varying transmissibility rates from known diseases, and using these in our model to assess the uncertainty of the epidemic trajectory.

REFERENCES

[1] Lander Willem et al. "Lessons from a Decade of Individual-Based Models for Infectious Disease Transmission: A Systematic Review (2006-2015)". In: *BMC Infectious Diseases* .17.1 (Dec. 2017), p. 612.

[2] William Ogilvy Kermack and A. G. McKendrick. "A Contribution to the Mathematical Theory of Epidemics". In: *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character* .115.772 (Aug. 1927), pp. 700–721.

[3] Christophe Fraser. "Estimating Individual and Household Reproduction Numbers in an Emerging Epidemic". In: *PLoS ONE* 2.8 (Aug. 22, 2007). Ed. by Alison Galvani, e758.

[4] Matthew J. Penn et al. *The Uncertainty of Infectious Disease Outbreaks Is Underestimated*. Oct. 25, 2022. URL: http://arxiv.org/abs/2210.14221 (visited on 01/19/2023). preprint.

[5] Richard Bellman and Theodore E. Harris. "On the Theory of Age-Dependent Stochastic Branching Processes". In: *Proceedings of the National Academy of Sciences* .34.12 (Dec. 1948), pp. 601–604.

[6] A. T. Bharucha-Reid. "ON THE STOCHASTIC THEORY OF EPIDEMICS". In: *Contributions to Biology and Problems of Health*. Ed. by Jerzy Neymann. University of California Press, Dec. 31, 1956, pp. 111–120.

[7] Kenny S Crump and Charles J Mode. "A General Age-Dependent Branching Process. I". In: *Journal of Mathematical Analysis and Applications* .24.3 (Dec. 1968), pp. 494–508.

[8] Kenny Crump and Charles J Mode. "A General Age-Dependent Branching Process. II". In: *Journal of Mathematical Analysis and Applications* .25.1 (Jan. 1969), pp. 8–17.

[9] Peter Jagers. "A General Stochastic Model for Population Development". In: *Scandinavian Actuarial Journal* .1969.1-2 (Jan. 1969), pp. 84–103.

[10] Mikko S. Pakkanen et al. *Unifying Incidence and Prevalence under a Time-Varying General Branching Process*. Dec. 21, 2022. URL: http://arxiv.org/abs/2107.05579 (visited on 01/19/2023). preprint.

[11] Peter Jagers. *Branching Processes with Biological Applications*. London ; New York: Wiley, 1975. 268 pp.

[12] Theodore W. Gamelin. *Complex Analysis*. Undergraduate Texts in Mathematics. New York, NY: Springer New York, 2001.

[13] Folkmar Bornemann. "Accuracy and Stability of Computing High-order Derivatives of Analytic Functions by Cauchy Integrals". In: *Foundations of Computational Mathematics* .11.1 (Feb. 2011), pp. 1–63.

[14] Joel C. Miller. "A Primer on the Use of Probability Generating Functions in Infectious Disease Modeling". In: *Infectious Disease Modelling* 3 (2018), pp. 192–248.

[15] V. A. Vatutin and A. M. Zubkov. "Branching Processes. II". In: *Journal of Soviet Mathematics* .67.6 (Dec. 1993), pp. 3407–3485.

[16] David Applebaum. *Lévy Processes and Stochastic Calculus*.

[17] Sheldon Ross. *Stochastic Processes*.

[18] Albert N. Shiryaev. *Probability 1*. Graduate Texts in Mathematics. New York, NY: Springer New York, 2016.

[19] Lawrence C. Evans. *Partial Differential Equations*. 2nd ed. Graduate Studies in Mathematics v. 19. Providence, R.I: American Mathematical Society, 2010. 749 pp.

Discretization of time-varying CMJ generating function

Author: *Frederik Mølkjær Andersen*

Companion notebook for the report *Efficient computation of the probability generating function in a general branching process model for infectious diseases*

This notebook should be read in conjunction with the report.

A `.ipynb` file constructing this pdf can be found on my GitHub (molkjar).

This notebook showcases the two implementations of the generating function for the time-varying presented in Section 6 in the report and the computation of the probability mass function from it as presented in Section 2.3. We will focus on the branching process counting prevalence under a negative binomial infection process.

In our example assume that $L^\tau \sim \Gamma(5, \theta(\tau))$ where $\theta(\tau)$ is a τ-dependent *scale*-parameter

$$\theta(\tau) := \frac{1}{1 + 0.01\tau}$$

and that individual infectiousness follows a $\Gamma(3, 1)$ distribution. We assume that the time-varying transmission rate ρ is assumed to be given as

$$\rho(t) := 1.5 + \sin(0.15t).$$

Despite the choice of counting characteristic, infection process etc. above the functions presented here can handle all cases mentioned in the report, in particular it can compute the generating function for cumulative incidence and use the Poisson infection process.

Most of our implementation is done through storing in and computing on matrices, here implemented as two-dimensional `numpy` arrays. Distribution related functions and a bit of linear algebra magic is drawn from `scipy` and plotting is done using `matplotlib`.

```
import numpy as np
from scipy.stats import gamma
from scipy.linalg import toeplitz
import matplotlib.pyplot as plt
```

```
def G(t, tau):
  return gamma.cdf(t, a = 6, scale = 1 / (1 + 0.01 * tau))

def V(t):
  return gamma.cdf(t, a = 3, scale = 1)

def rho(t):
```

```
  return 1.5 + np.sin(0.175 * t)
```

We now define a helper function which returns all the matrices, functions and alike used in the computation of the generating function, the definitions follow from Section 6 of the report. The result is saved in a list. In particular this lets us precompute all of these quantities which are invariant of s, the argument of the generating function.

```
def pgf_prep(N, Delta, G, V, rho, phi, count="prev", inf_proc="negbin"):
  if inf_proc == "negbin":
    def psi(z):
      return - phi * (np.log(phi + 1 - z) - np.log(phi))
  if inf_proc == "pois":
    def psi(z):
      return z - 1

  def q_1(z, s):
    return s * np.exp(z)

  if count == "prev":
    def q_2(z, s):
      return np.exp(z)
  if count == "ci":
    def q_2(z, s):
      return s * np.exp(z)

  Gbar_t = np.tril(np.tile(np.arange(0, N+1) * Delta, (N+1, 1)).T)
  Gbar_tau = np.tril(toeplitz(np.arange(0, N+1) * Delta))
  Gbar = 1 - G(Gbar_t - Gbar_tau, Gbar_tau)

  D_t = np.triu(np.tile(np.arange(N, -1, -1) * Delta, (N, 1)))
  D_tau = np.triu(np.tile(np.arange(0, N) * Delta, (N+1, 1)).T)

  DV = -rho(D_t[:, :-1] + D_tau[:, :-1]) * np.diff(V(D_t))
  DG = -np.diff(G(D_t, D_tau))

  return [N, psi, q_1, q_2, Gbar, DV, DG]
```

Algorithm 1

We are ready to implement Algorithm 1 from the report. The algorithm involves three nested for-loops so will probably take a while to finish, at least for high N. The function takes two

arguments, the generating function argument s and the resulting list from `pgf_prep` above.

```
def pgf_direct(s, prep_list):
  N, psi, q_1, q_2, Gbar, DV, DG = prep_list

  F = np.zeros((N+1, N+1), dtype=np.complex_)
  F[:, 0] = s

  for n in range(1, N+1):
    for i in range(1, n+1):
      int_1 = Gbar[n, i] * q_1(np.sum(psi(F[n, 0:i]) * DV[n-i, (N-i):N]), s)

      int_2 = 0 + 0j
      for j in range(i):
        int_2_in = np.sum(psi(F[n, (j+1):i]) * DV[n-i, (N-i+j+1):N])
        int_2 += q_2(int_2_in, s) * DG[n-i, N-i+j]

      F[n, i] = int_1 + int_2

  return np.diag(F)
```

Algorithm 2

Next we will use the vectorization scheme outlined in the report, `numpy` has a lot of nice array transformation features which we use an abundance of. This implementation contains only one for-loop (which we can't get rid of), so should be much faster than `pgf_direct`. Further vectorization over `s` is possible using three-dimensional arrays, however this takes up a lot of unnecessary memory and without much gain in speed, however the problem is embarrassingly parallelizable as computation for each s is completely independent from each other.

Have in mind that both implementations do the exact same computations (with diregard to what happens in the parts of the matrices which are never acessed), so their results should be identical, up to some floating point error.

```
def pgf_vec(s, prep_list):
  N, psi, q_1, q_2, Gbar, DV, DG = prep_list

  F = np.zeros((N+1, N+1), dtype=np.complex_)
  F[:, 0] = s

  for i in range(1, N+1):
```

```
    B = psi(F[i:(N + 1), 0:i]) * DV[0:(N - i + 1), (N - i):N]
    B_1 = np.cumsum(B[:, ::-1], axis = 1)[:, ::-1]

    int_1 = Gbar[i:(N + 1), i] * q_1(B_1[:, 0], s)

    B_2 = np.concatenate((B_1[:, 1:i], np.zeros((B_1.shape[0], 1))), axis = 1)
    int_2 = np.sum(q_2(B_2, s) * DG[0:(N - i + 1), (N - i):N], axis = 1)

    F[i:(N + 1), i] = int_1 + int_2

  return np.diag(F)
```

Computing the probability mass function

As described in Section 2.3 of the report we can take advantadge of the Cauchy integral and fast Fourier transform to compute the probability mass function from the generating function. Below is a function that takes the `pgf_prep` list of object and computes the derivatives of the generating function and thus the mass function. To take care numerical errors, the mass function is normalized before it is returned.

As mentioned above, computing of the generating function at many values of s is embarrassingly parallelizable, however parallelization is not trivial in a Jupyter Notebook, so we will let it be as is.

```
def pmfft(M, prep_list):
  pgf = np.array(list(map(lambda m: pgf_vec(np.exp(2.0j*np.pi*m/M), prep_list), range(M))))
  fft = np.real(np.fft.fft(pgf, axis=0))
  pmf = fft * (fft >= 0)
  return pmf / np.sum(pmf, axis=0)
```

In practice we want to compute moments of the process for now we will investigate the mean of the process, however higher moments are easily found from the probability mass function.

```
def mean(pmf):
  M = np.shape(pmf)[0]
  loc = np.linspace(0, M-1, M)
  mn = np.sum(pmf.T * loc, axis = 1)
  return mn
```

A test for correctness

A sanity check that our algorithms above are actually computing the generating function we are looking for. We use the previously developed algorithm for the mean of the process (Pakkanen, 2022) which is implemented below, and compare the mean computed from the generating function.

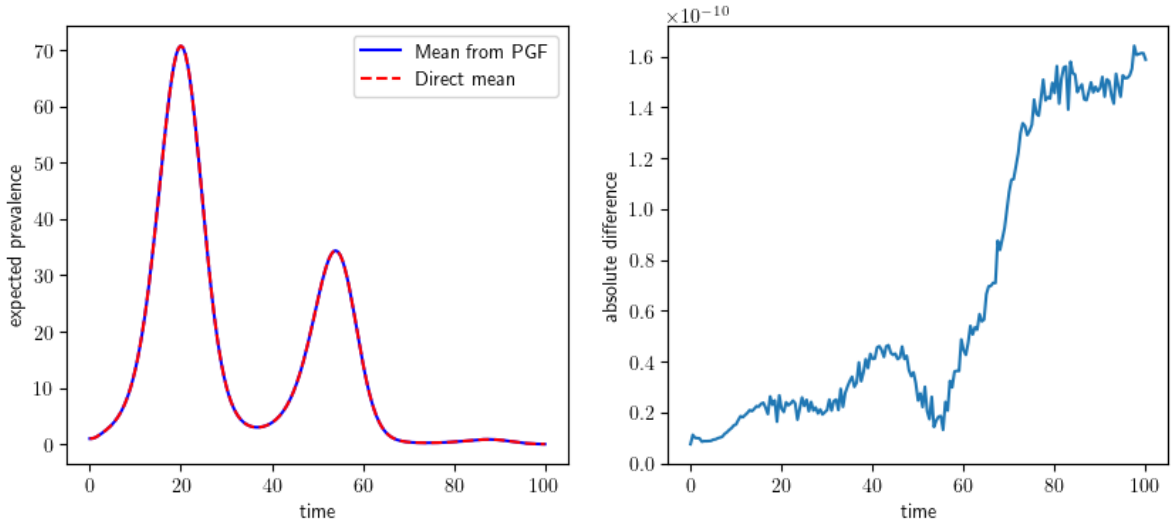
```
def quick_mean(N, Delta, G, V, rho):
  def h(t, tau):
    return 1 - G(t - tau, tau)

  def lam(t, tau):
    return rho(t + tau) * (V(t) - V(t-Delta)) * (1 - G(t, tau))

  fhat = np.zeros(shape = (N + 1, N + 1))

  for n in range(N + 1):
      for i in range(n + 1):
          if i == 0:
              fhat[n, i] = h(n * Delta, n * Delta)
          else:
              fhat[n, i] = h(n * Delta, (n - i) * Delta) + np.sum(fhat[n, i - np.arange(1,

  return np.diagonal(fhat)
```

We will generate the discretization over $N = 200$ time steps with step size $\Delta = 0.5$, i.e. simulating the epidemic until time $t = 100$. The fast Fourier transform need a quite high number of terms, we use $M = 5000$, but remember that this task is easy to parallelise.

```
N = 200
Delta = 0.5

mn = quick_mean(N, Delta, G, V, rho)

p_list = pgf_prep(N, Delta, G, V, rho, 1, inf_proc = "pois")
pmf = pmfft(5000, p_list)
```

```
grid = Delta * np.arange(0, N+1)

fig, (ax1, ax2) = plt.subplots(1, 2)
fig.set_size_inches(10, 4, forward=True)
```

```
ax1.plot(grid, mean(pmf), "b", label = "Mean from PGF")
ax1.plot(grid, mn, "r--", label = "Direct mean")
ax1.set(
  xlabel = "time",
  ylabel = "expected prevalence"
)
ax1.legend()

ax2.plot(grid, abs(mean(pmf) - mn))
ax2.set(
  xlabel = "time",
  ylabel = "absolute difference"
)
```



We see that the two approaches only differ by some floating point error.