# Peculiarities of Counterfactual Point Process Generation

Gerrit Großmann
gerrit.grossmann@dfki.de
DFKI
Kaiserslautern, Germany

Sumantrak Mukherjee
sumantrak.mukherjee@dfki.de
DFKI
Kaiserslautern, Germany

Sebastian Vollmer
sebastian.vollmer@dfki.de
DFKI
Kaiserslautern, Germany

## Abstract

(Spatio-)Temporal point processes are the de facto standard for modeling events in (space and) time. This study addresses the generation of counterfactual sequences: Given a model and an observed sequence, we investigate how the sequence would have evolved under a hypothetical alternative model.

Our contribution is two-fold: Firstly, we demonstrate how to easily leverage established stochastic simulation algorithms to generate counterfactual sequences. Secondly, we reveal that different simulation methods—despite being statistically equivalent—correspond to distinct structural causal models of the point process, producing distinct counterfactual distributions.

Given these findings, we recommend exercising greater caution when applying counterfactual reasoning in this domain, particularly concerning the relationship between counterfactual generation and the underlying physical processes.

## CCS Concepts

• **Computing methodologies** → **Modeling methodologies**; *Planning under uncertainty*; • **Information systems** → *Geographic information systems*.

## Keywords

Point Process, Event Modeling, Causality, Counterfactual Generation

**ACM Reference Format:**
Gerrit Großmann, Sumantrak Mukherjee, and Sebastian Vollmer. 2024. Peculiarities of Counterfactual Point Process Generation. In *1st ACM SIGSPATIAL International Workshop on Spatiotemporal Causal Analysis (STCausal'24), October 29-November 1 2024, Atlanta, GA, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3681778.3698785

## 1 Introduction

Imagine being presented with an alarming record of traffic accidents in your town, leading you to wonder: *"Would the number of accidents be lower if the additional road had not been built?"* In this work, we seek to address questions like this within the framework of point processes, while carefully considering the underlying assumptions.

Temporal point processes (TPPs) are a powerful framework for modeling sequences of events that occur over time [Rasmussen 2018]. Marked temporal point processes (MTPPs) and spatio-temporal point processes (STPPs) extend this framework by annotating events with additional information such as marks and coordinates [Reinhart 2018]. These modeling paradigms are employed in diverse fields such as finance [Linderman and Adams 2014], healthcare and epidemiology [Kim et al. 2019], social networks, and seismology [Kwon et al. 2023]. The cornerstone of TPPs is the *conditional intensity function*, which represents the instantaneous rate at which events occur given the current time (and position) and the history of past events. This function underpins inference, analysis, and prediction tasks. An intriguing but difficult type of analysis concerns the *"What if"* questions [Noorbakhsh and Rodriguez 2022].

*"What if"* questions are fundamental to human reasoning and play a crucial role in the mathematical formalization of causality, where they are discussed under the term *counterfactuals*. Counterfactual analysis in the context of TPPs involves asking how an event sequence might have differed under alternative scenarios. This analysis allows researchers to formalize the effects of interventions and hypothetical changes in the system.

However, relating the intuition of human causal reasoning to the mathematical formalization of causality is complex and fraught with potential pitfalls [Halpern 2019; Hedden 2023; Markus 2021]. Nevertheless, it is widely accepted that counterfactual samples can be computed within the framework of structural causal models [Pearl et al. 2016].

Structural causal models (SCMs) use directed acyclic graphs (DAGs) to represent causal relationships between variables. This mechanistic relationship can be exploited to sample from a counterfactual distribution. The method involves re-evaluating the data-generating process under specific model modifications while fixing the exogenous noise variables. Maintaining the same background conditions, represented by the noise terms, aligns well with human counterfactual reasoning as it answers the question: *"What would have happened if I had changed one thing (model modification) but kept everything else the same (external or internal noise)?"*

In this work, we explore how to convert point process models to SCMs to perform counterfactual analysis and identify potential pitfalls in this process. Specifically, we demonstrate that there is no single unique SCM corresponding to a point process model. Instead, different simulation algorithms, while statistically equivalent, correspond to different SCMs and, therefore, different counterfactual distributions.

We formalize point processes and the counterfactual distribution in Section 2. We show how to represent simulation algorithms as SCMs in Section 3 and in Section 4 perform the counterfactual generation. Section 5 illustrate our findings. A discussion of related works in Section 6 provides context for our contributions. Finally, a conclusion in Section 7 completes the manuscript.


*STCausal'24, October 29-November 1 2024, Atlanta, GA, USA*

ACM ISBN 979-8-4007-1154-1/24/10
https://doi.org/10.1145/3681778.3698785

Our analysis and derivations are all done on (vanilla) TPPs, with an extensions to spatial TPPs provided in Appendix A.

## 2 Background

In this section, we formally introduce our point process based model for event data and explain how to simulate it. We also present our model for causality through SCMs and demonstrate how to use them for generating counterfactuals in a general setting. For a detailed introduction to TPPs, we refer the reader to [Rasmussen 2018; Reinhart 2018; Rodriguez and Valera 2018], and for SCMs and Pearl's do-calculus, to [Pearl 2009; Pearl et al. 2016].

### 2.1 Temporal Point Processes

A given event sequence is a finite set of non-negative, real-valued timestamps, which is represented as an increasing sequence $H = [t_0, t_1, \ldots, t_n]$, where $t_i \in \mathbb{R}_{\geq 0}$ and $t_i < t_{i+1}$. W.l.o.g. we assume that $t_0 = 0$.

A TPP model identifies a set of event sequences with a probability density. Typically, one considers the set of event sequences where $t_n$ (the maximal time point) is smaller than a given time horizon $T$. This is denoted as

$$\mathcal{H}_T = \{\text{all event sequences with } t_n \leq T\}.$$

The likelihood $f(\cdot)$ identifies each sequence $H \in \mathcal{H}_T$ with its probability density. In TPPs, the future can never influence the past, thus, the PDF can be decomposed into a factorization.

The likelihood function for the sequence $H$ going until time horizon $T$ is:

$$f(H) = \left(\prod_{i=1}^{n} f^{\text{pred}}(t_{i+1} \mid H_{t_i})\right)\left(1 - F^{\text{pred}}\left(T \mid H_{t_n}\right)\right) .$$

The term $f^{\text{pred}}(t_{i+1} \mid H_{t_i})$ is known as the (conditional) *predictive distribution*. It represents the probability density of the next event occurring at time $t_{i+1}$ given the history of all previous events up to time $t_i$. $F^{pred}$ is used to denote the cumulative probability of an event happening until time horizon T.

In practice, the (conditional) intensity function $\lambda(t \mid H_t)$ is used instead of the predictive distribution because it is easier to parameterize and has some other practical advantages.

The (conditional) intensity function, intuitively, represents the instantaneous rate of occurrence of events at time $t$ given the history $H_t$. It provides a measure of how likely an event is to happen in the next infinitesimal interval of time, given all the past events. The intensity function is defined as follows:

$$\lambda(t \mid H_t) = \lim_{\Delta \to 0} \frac{P(\text{Event happens in } [t, t+\Delta t) \mid H_t)}{\Delta}$$

where $\lambda(t \mid H_t)$ represents the conditional intensity function, $H_t$ is the history of events up to (but not including) time $t$.

The corresponding factorization the likelihood of an event sequence $H = [t_0, t_1, \ldots, t_n]$ with $t_n \leq T$ is:

$$f(H) = \left(\prod_{i=1}^{n} \lambda(t_i \mid H_{t_i})\right) \exp\left(-\int_0^T \lambda(t \mid H_t)\, dt\right) .$$

### 2.2 Stochastic Simulation of Point Processes

In this section, we document three well-known stochastic simulation algorithms for generating event sequences for a given time horizon $T$ and intensity $\lambda(t \mid H_t)$. All algorithms are statistically equivalent, meaning that they sample from the same distribution (the distribution over $\mathcal{H}_T$ induced by $\lambda(\cdot \mid \cdot)$).

The overall algorithm that provides the framework is outlined as follows:

**Algorithm 1** Stochastic Simulation of TPPs

**Input:**
  Time horizon $T \in \mathbb{R}_{>0}$
  Intensity function $\lambda : \mathbb{R}_{>0} \times \mathcal{H}_T \to \mathbb{R}_{>0}$
**Output:**
  Event sequence $H$
Initialize $t_0 = 0$, $H = [t_0]$, and $i = 0$
**while** `true` **do**
  $i = i + 1$
  $t_i$ = `sample_next_event_time`$(H, t_{i-1}, \lambda(\cdot \mid \cdot))$
  **if** $t_i > T$ **then**
    **Return** $H$
  **end if**
  Add $t_i$ to $H$
**end while**

The function `sample_next_event_time`$(H, t, \lambda(\cdot \mid \cdot))$ takes as input the history (up to the current time point), the timepoint of the last event (explicitly included for better readability), and the conditional intensity function. The three algorithms, we discuss next, implement this function in different ways.

*Method I - Naïve Method.* The rationale behind the first method is to take directly take the definition of the intensity function and decide if an event happens for each point in time. Therefore, we discretize the time using a very small constant $\Delta$ approximating an infinitesimal time interval.

**Algorithm 2** Method I (Naïve): `sample_next_event_time`$(H, t, \lambda)$

**Input:**
  List of events $H$
  Intensity function $\lambda(t \mid H)$
  Current timepoint $t$
**Output:**
  Next event time $t'$
Initialize time $t' = t$
Choose a small time interval $\Delta$
**while** `true` **do**
  Update time: $t' = t' + \Delta$
  Compute event probability: $p_{t'} = \lambda(t' \mid H) \cdot \Delta$
  Sample $u$ uniformly at random in $[0, 1]$
  **if** $u < p_{t'}$ **then**
    **Return** $t'$
  **end if**
**end while**

Since the events happen on the real number line, discretising time is an approximation. Technically, we need to note that this method converges to the correct algorithm if $\Delta$ converges to zero.

*Method II - Numerical Integration.* The idea behind the method employing numerical integration of the intensity function [Rasmussen 2018] is that instead of sampling a random variate to decide if an event happens at each time point, we can sample the number of times we would need to generate variates until we eventually reach a case where $u < p_t$ [Großmann et al. 2020]. However, since $p_t$ changes over time, it is difficult to directly predict the amount of time (or the number of steps) until this happens. Therefore, we sample a random variate from an exponential distribution with rate one and then integrate until it reaches $l$.

**Algorithm 3** Method II (Integration): `sample_next_event_time`($H$, $t$, $\lambda$)

**Input:**
  List of events $H$
  Intensity function $\lambda(t \mid H)$
  Current timepoint $t$
**Output:**
  Next event time $t'$
Choose a small time interval $\Delta$
Sample $u$ uniformly at random in $[0, 1]$
Define $l = -\ln(u)$ ▷ $l$ follows exp. distr. with rate 1
Initialize $A = 0$ ▷ Accumulated area under the curve
Initialize $t' = t$
**while** $A < l$ **do**
  $t' = t' + \Delta$
  Update $A = A + \lambda(t' \mid H) \cdot \Delta$
**end while**
**Return** $t'$

Again, this method is correct in the limit as $\Delta$ converges to zero. It nicely illustrates another meaning of the intensity function: the integral of the intensity function determines the expected number of events over a given time interval.

That is, we find the next event time, $t'$, such that:

$$\int_t^{t'} \lambda(t^* \mid H)\, dt^* = l.$$

*Method III - Thinning.* The thinning algorithm [Ogata 1981] as illustrated in 4 was invented to decrease the computational costs of event sequence generation. The idea is that one can generate an event sequence of some $\lambda(t \mid H)$ by first generating a homogeneous event sequence with a constant rate $\lambda_{\max}$ (assuming $\lambda_{\max} > \lambda(t \mid H)$ for all $t$ and $H$) and then removing some of the events. This approach is computationally efficient because generating event sequences with a fixed rate can be done by sampling inter-event times using the exponential distribution.

The downside of this method is that finding an upper bound for the intensity function is not always possible. For simplicity, and w.l.o.g., we assume that we have a fixed upper bound over the while time interval while the widely-used Ogata's thinning algorithm also works with piece-wise upper bounds [Ogata 1981; Rasmussen 2018].

**Algorithm 4** Method III (Thinning): `sample_next_event_time`($H$, $t$, $\lambda$)

**Input:**
  List of events $H$
  Intensity function $\lambda(t \mid H)$
  Current timepoint $t$
  (Additionally) $\lambda_{\max}$, an upper bound of $\lambda(t \mid H)$
**Output:**
  Next event time $t'$
Initialize $t' = t$
**while** `true` **do**
  Sample $u$ uniformly at random in $[0, 1]$
  Set $t^{\text{candidate}} = \frac{-\ln(u)}{\lambda_{\max}}$ ▷ follows exp. distr. with rate $\lambda_{\max}$
  Update time: $t' = t' + t^{\text{candidate}}$
  Sample $u'$ uniformly at random in $[0, 1]$
  **if** $u' \leq \frac{\lambda(t'|H)}{\lambda_{\max}}$ **then**
    **Return** $t'$
  **end if**
**end while**

## 2.3 SCMs and the Counterfactual Distribution

A Structural Causal Model (SCM) is a mathematical framework designed to represent causal relationships among a set of variables by specifying the underlying data-generating process. Formally, an SCM is defined as a tuple $(\mathbf{U}, \mathbf{V}, \mathbf{F}, P(\mathbf{U}), \mathbf{PA})$, where $\mathbf{U}$ denotes the set of exogenous (external) variables, $\mathbf{V}$ represents the set of endogenous (system) variables, $\mathbf{F}$ is the set of structural functions, $P(\mathbf{U})$ is the probability distribution over the exogenous variables, and $\mathbf{PA}$ denotes the set of parent-child relationships between the variables.

The set $\mathbf{PA}$ implicitly defines a DAG over the variables, such that each $V_i \in \mathbf{V}$ has a corresponding set of parent variables $PA_i \subseteq \mathbf{V} \cup \mathbf{U}$, where the exogenous variables $U_i \in \mathbf{U}$ are always root nodes. The values of the exogenous variables $U_i$ are governed by the probability distribution $P(\mathbf{U})$. The values of the endogenous variables $V_i \in \mathbf{V}$ are governed by the functions $f_i \in \mathbf{F}$ that take as input the values of the parent nodes of $V_i$. Specifically, $V_i = f_i(PA_i)$. For simplicity and to ensure better consistency with the data-generating process, we allow endogenous variables that have no exogenous variables as parents. We also allow $V_i$ to be a root node, in which case $PA_i = \emptyset$, and $V_i$ is a constant.

*Counterfactuals.* SCMs make it possible to study the effects of modifications, i.e., *interventions*, to the data generation process (e.g., by fixing the value of a specific $V_i$). Counterfactuals constitute the third rung of Pearl's Causal Hierarchy where based on an observed outcome one tries to *imagine* how changing a specific event or condition while keeping everything fixed changes the outcome. Hence, they are used to study "*What if*" scenarios.

When observational data, typically consisting of measurements of some endogenous variables, is available alongside a specified SCM, one can hypothesize about how the observational data would have looked like under a different hypothetical SCM. The computation of a this counterfactual observational data involves a three-step procedure [Pearl et al. 2016]:

- **Abduction:** Infer the values of the exogenous variables $\mathbf{U}$ from the observed evidence, effectively computing the posterior distribution of the noise variables.
- **Action:** Modify the SCM by intervening on the variables or relationships of interest. Note that the modified SCM needs to have the same number of noise variables as the original one.
- **Prediction:** Utilize the modified SCM, along with the inferred values of the exogenous variables (posterior noise), to predict the hypothetical outcome.

It is important to note that observations can either fix the exogenous variables to specific values or update the prior distribution $P(\mathbf{U})$ to a posterior distribution, depending on the information provided by the data.

*Monotonicity.* It is typically desired that counterfactual samples fulfil a property called *monotonicity* [Hızlı et al. 2023; Noorbakhsh and Rodriguez 2022; Oberst and Sontag 2019], which promotes intuitive behaviour in the counterfactual distribution. Specifically, monotonicity means that if an intervention on a variable causes an outcome in one scenario, then a stronger intervention in the same scenario should also cause the outcome B. Monotonicity prevents counterintuitive situations where, for example, increasing the infectivity rate of a disease (while keeping everything else fixed) leads to fewer cases than a scenario with a lower infectivity rate.

*Rethinking SCMs.* At a high level, we can think of an SCM as a mapping that takes a sample $\mathbf{U}$ from a noise distribution $P(\mathbf{U})$ and generates assignments of all endogenous variables $\mathbf{V}$. In counterfactual analysis, we examine scenarios where the same noise sample (or a specific distribution) is used to generate endogenous variables of a different SCM.

Thus, counterfactuals can be understood as a mapping between two distinct probability spaces: one representing the observations and their associated probabilities as determined by the original SCM, and the other representing the observations and probabilities under the counterfactual SCM. This mapping can be either deterministic (one-to-one) or stochastic (governed by the posterior distribution of the noise variables).

## 3 SCM Representation of Point Processes

We can directly convert the three sampling algorithms from Section 2.2 into SCMs. These SCMs will fulfil the monotonicity assumption in a reasonable sense. Note that we always use $U$ to denote external/noise variables. The corresponding DAGs of the SCMs are illustrated in Figure 1. Please note that, unlike typical SCMs with exogenous noise variables for all endogenous variables, we only add exogenous noise variables to nodes corresponding to steps in the simulation algorithm which sample from a distribution. All other relations are kept noise-free and deterministic to represent the stochastic generation accurately.

*Naïve Method.* We have the following variables:

- $H_{i\Delta}$ (note that the time index is always a multiple $\Delta$) is a sequence of event times and contains the events up to time point $t = i\Delta$.
- $\lambda_{i\Delta} \in \mathbb{R}_{\geq 0}$ indicates the instantaneous rate at time point $t = i\Delta$.
- The noise variable $U_i$, drawn uniformly in $[0, 1]$ for each time step $i$, introduces stochasticity to the model.
- $E_i \in \{\texttt{true}, \texttt{false}\}$ indicates if an event is happening within $[i\Delta, (i+1)\Delta)$ based on the current intensity and noise.

Next, we need to define the functional relations between the variables. Specifically, we define how the value of a node is determined by the value of its parental nodes.

$$
\begin{aligned}
&\lambda_{i\Delta} = \lambda(i\Delta \mid H_{i\Delta}),\\
&E_i = U_i < \lambda_{i\Delta} \cdot \Delta,\\
&H_0 = [],\\
&\text{and for } i > 0:\\
&H_{i\Delta} = \begin{cases} H_{(i-1)\Delta} & \text{if } E_{i-1} = \texttt{false},\\ H_{(i-1)\Delta} \cup \{(i-1)\Delta\} & \text{otherwise.} \end{cases}
\end{aligned}
$$

Note that, by convention (the first event $t_0$ happening at $t = 0$), we have that $E_0 = \texttt{true}$. The final event sequence is determined by considering the timepoints of all $E_i$ that are set to true.

The relationship to Algorithm 2 is that, in each iteration, the variable $u$ maps to the node $U_i$, and the *if* condition maps to computing the value of the node $E_i$.

*Numerical Integration.* We have the following variables:

- $U_i \in [0, 1]$, the noise that introduces stochasticity to the model.
- $H_{t_i}$, the event history which contain the event sequence that includes all events that happen before $t_i$.
- $t_i \in \mathbb{R}_{\geq 0}$, the time of the $i$-th event.
- $\lambda_i : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$, the intensity as a function over time that we can integrate.

The functional relationships are defined as follows:

$$
\begin{aligned}
H_{t_i} &= \begin{cases} [0] & \text{for } i = 0,\\ H_{t_{i-1}} \cup \{t_{i-1}\} & \text{for } i > 0, \end{cases}\\
\lambda_i(t) &= \begin{cases} \lambda(t \mid H_{t_i}) & \text{for } t > \max(H_{t_i}),\\ 0 & \text{otherwise,} \end{cases} \qquad (1)\\
t_i \quad &\text{is s.t.:} \int_0^{t_i} \lambda_i(t)\, dt = -\ln(U_i).
\end{aligned}
$$

Note that $-\ln(U_i)$ follows an exponential distribution with rate one. The final event sequence is determined by the list of all $t_i$.

The relationship to Algorithm 3 is that, for each function call, the variable $u$ maps to the node $U_i$, and the computation of the integral corresponds to the computation of node $t_i$ based on integrating $\lambda_i$.

*Thinning.* The novel aspect of the thinning algorithm is that it requires two random noise variates for each event candidate. The first variate, $U_i$, is used to determine the candidate for the $i$-th event. The second random variate, $U_i'$, is used to decide whether this candidate actually becomes an event ($E_i = \texttt{true}$) or is rejected ($E_i = \texttt{false}$).

The candidate event times can be computed as (assuming $t_0 = 0$):

$$
t_i = t_{i-1} + \frac{-\ln(u_i)}{\lambda_{\max}} \quad . \qquad (2)
$$

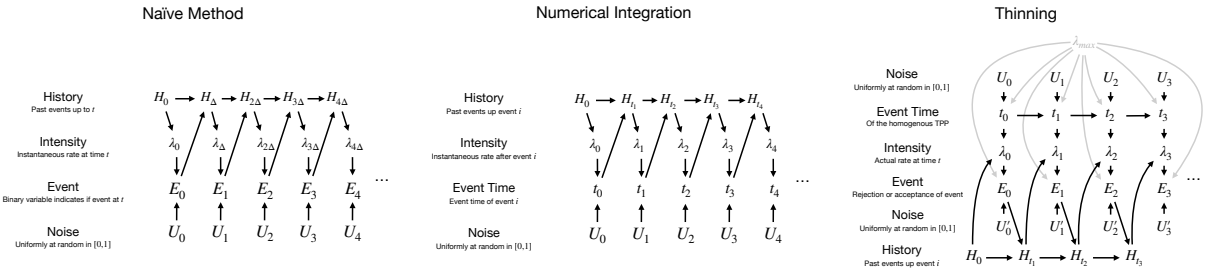


**Figure 1: SCM representations of three algorithms for realising counterfactual event sequences**

Note that $\frac{-\ln(U_i)}{\lambda_{\max}}$ follows an exponential distribution with rate $\lambda_{\max}$.

The intensity of the $i$-th event is $\lambda_i = \lambda(t_i \mid H_{t_i})$, and it is used to compute the rejection probability, leading to:

$$E_i = U_i' \leq \frac{\lambda_i}{\lambda_{\max}} \,. \tag{3}$$

The final event sequence is determined by filtering out all $t_i$ where the corresponding $E_i$ is `true`.

The relationship to Algorithm 4 is that, each function call adds multiple layers to the SCM until eventually one $E_i$ will be true. That is, the (rejected) candidate times are explicitly accounted for in the SCM. The noise variables in the algorithms directly correspond to the nodes $U_i$ and $U_i'$.

## 4 Sampling from the Counterfactual Distribution

This section describes the process of sampling from the counterfactual distribution by applying the abstract recipe from Section 2.3 to the constructed SCMs. From now on, we will use colors do indicate if data, parameters, or models belong to the original or the counterfactual setting. Our focus will be on converting an *original* event sequence into a counterfactual event sequence using a provided counterfactual intensity function, denoted as $\lambda^{\text{cf}}(t \mid H)$. These methods can be generalized to other counterfactual interventions.

Note that, if we compute counterfactuals using the *Abduction, Action, Prediction* method, the following automatically holds:

- When the counterfactual intensity function is identical to the original intensity function, i.e., $\lambda(\cdot \mid \cdot) = \lambda^{\text{cf}}(\cdot \mid \cdot)$, the resulting counterfactual event sequence matches the original event sequence exactly.
- Given a pair of intensity functions, $\lambda^{\text{cf}}(\cdot \mid \cdot)$ and $\lambda(\cdot \mid \cdot)$, the following procedure yields an unbiased sample from the distribution over event sequences induced by $\lambda^{\text{cf}}(\cdot \mid \cdot)$: (1) Sample a random event sequence, $H$, from $\lambda(\cdot \mid \cdot)$; (2) Compute the counterfactual event sequence using $\lambda^{\text{cf}}(\cdot \mid \cdot)$. In other words, by marginalizing over all possible sequences, the counterfactual sample provides a distributional sample as if the counterfactual intensity function had been employed through stochastic simulation.

The second point can also be interpreted to mean that the mapping induced by the counterfactual process does not distort the probability distributions. For example, in Figure 7 in Appendix B, sampling directly from a coin with $p_H = 0.5$ is equivalent to sampling from a coin where $p_H = 0.4$ and then transitioning to the counterfactual case of $p_H = 0.5$.

For plausible counterfactual sequences, akin to the monotonicity condition, we formulate the following properties we expect to hold:

- If an event occurs at time $t$ in the original sequence and the counterfactual intensity at this time point is higher, then the event must also occur in the counterfactual sequence.
- If an event does not occur in the original sequence at time $t$ and the counterfactual intensity at this time point is lower, then the event cannot occur in the counterfactual sequence.

Next, we discuss how to translate the three SCMs and the general method form the previous sections into concrete algorithms. A schematic illustration of the methods is provided in Figure 2.

*Naïve Method.* Computing the posterior noise distribution for a given sequence is straightforward and results in an interval for each time step $i$. Depending on whether an event occurs, this interval is either very small ($[0, P^i_{\text{event}}]$) or large ($[P^i_{\text{event}}, 1]$). Note that $P^i_{\text{event}}$ will typically be extremely small because $\Delta$ is considered to be extremely small. To generate the counterfactual event sequence, we can then resample from that interval for each time step. In this case, $u_i$ in Algorithm 5 directly corresponds to $U_i$ in the SCM in Figure 1a and the return value $H^{\text{cf}}$ corresponds to the timepoints for which $E_i = \texttt{true}$ in the counterfactual distribution of the SCM (i.e., given the counterfactual intensity function where the noise variables are instantiated based on the posterior noise distribution). The sampling algorithm is provided in Algorithm 5.

*Numerical Integration.* The interesting part about this method is that we do not have a posterior noise distribution that corresponds to a given event sequence. Instead, we have exact values for each noise variable. Thus, we identify the unique values of $U_i$ that generate the observed sequence and, swapping the intensity function then yields the counterfactual sequence. The variables form the SCM in Figure 1b directly map to Algorithm 6. Note that this method implicitly changes the time horizon of the sequence.

*Thinning.* A direct translation is more difficult than in the previous cases because we do not observe the sequence of event time candidates, only the sequence of successful events. This would make it necessary to integrate over all possible candidate sequences, which is infeasible. To circumvent this, we combine the numerical

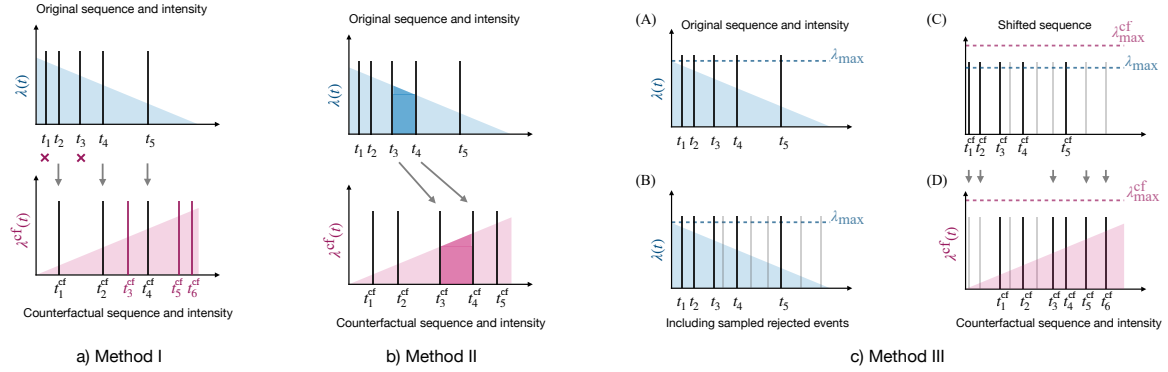


**Figure 2: Schematic overview of the counterfactual event sequence generation for the three methods. The original intensity (increases over time) is shown in blue, the counterfactual intensity is shown in purple (decreases over time).**
**(a) Method I: Where the original intensity is larger than the counterfactual intensity, events may disappear (indicated with purple crosses). Conversely, where the original intensity is lower than the counterfactual intensity, new events may emerge (dark purple lines). Since new events may emerge the indexing of events in the counterfactual sequence don't correspond to the indexing of the observed sequence.**
**b) Method II: Event times are shifted to the right to ensure that the integral from $t_i$ to $t_{i+1}$ of $\lambda(t)$ equals the integral from $t_i^{\mathbf{cf}}$ to $t_{i+1}^{\mathbf{cf}}$, which is demonstrated for $t_3$ and $t_4$ with darker shaded areas.**
**c) Method III: (A) The original sequence of five arbitrary events. (B) Sampling of rejected events (gray) given $\lambda(t)$. (C) Leftward shift of events, assuming $\lambda_{\mathbf{max}}^{\mathbf{cf}} > \lambda_{\mathbf{max}}$. (D) Application of counterfactual intensity $\lambda^{\mathbf{cf}}(t^{\mathbf{cf}})$ stochastically flips event types: rejected events can become real (black) where $\lambda^{\mathbf{cf}}(t_i^{\mathbf{cf}}) > \lambda(t_i)$, and real events can become rejected (gray) where $\lambda^{\mathbf{cf}}(t_i^{\mathbf{cf}}) < \lambda(t_i)$. The final counterfactual sequence are the black events in (D).**

methods to generate counterfactual sequences of the background process and combine it with the way counterfactual choices were made in the naïve method.

First, we need to generate samples from the background process, which is challenging because we only observe accepted samples in the trajectory $H$. We employ the same trick as [Noorbakhsh and Rodriguez 2022]. That is, we build a complementing or shadow process [Cota and Ferreira 2017; Großmann et al. 2020] that accounts exactly for the difference between the upper bound, $\lambda_{\max}$, and the actual intensity: $\lambda^{\text{shadow}}(t \mid H_t) = \lambda_{\max} - \lambda(t \mid H_t)$. We then conceptually consider the rejected events not rejections but as instantiations of the shadow process. This allows us to sample from the set of rejected events by fist sampling a homogeneous sequence using $\lambda_{\max}$ and then thinning this sequence based on the shadow process, yielding a sample from the posterior background sequence. The reason this works is that the sequence from the shadow process is independent of the accepted (observed) sequence, allowing us to generate it retrospectively, even after the accepted data has been fixed.

After this, we obtain a sequence of accepted events in the original process, denoted by $H$, and a sequence of rejected events, denoted by $R$. The effect of the counterfactual intensity function $\lambda^{\text{cf}}(t \mid H_t)$ is to swap events between $H$ and $R$. Specifically, an event can move from $R$ to $H$ if the counterfactual intensity is greater than the original intensity, and vice versa.

Adjusting the counterfactual upper bound $\lambda_{\max}^{\text{cf}}$ has the same effect as described in Algorithm 6, shifting the samples to the left if $\lambda_{\max}^{\text{cf}} > \lambda_{\max}$ or to the right if $\lambda_{\max}^{\text{cf}} < \lambda_{\max}$.

The sampling algorithm is detailed in Algorithm 7.

It may seem counter-intuitive that selecting a different upper bound $\lambda_{\max}^{\text{cf}}$ alters the counterfactual event sequence even if the actual intensity function remains unchanged. This would indeed be the case if the thinning algorithm were merely a computational trick to simplify stochastic generation. However, if we interpret the rejections as an intrinsic part of the physical process and assign physical meaning to the upper bound, then this adjustment is reasonable.

## 5 Case Study: A Mechanistic Model of Epidemic Spreading

In this case study, we examine a mechanistic model of epidemic spreading to analyze how counterfactual interventions on epidemic parameters alter a given sequence of infection events. Specifically, we demonstrate that intervening on the transmissibility aligns with the counterfactual distribution derived from Algorithm 6. Essentially, a higher infection rate results in infection events occurring more rapidly, as individuals become infectious earlier.

In contrast, modifying the connectivity of agents increases the frequency of infection events. For example, if people stop working from home and start meeting at work, new infection chains become possible. This scenario corresponds more closely to the counterfactual computations described in Algorithm 5.

The purpose of this example is not propose this as the most suitable model for counterfactual epidemic spreading. Rather, it aims to illustrate that this relationship between physical world and

**Algorithm 5** Generating Cf. TPP Sequences - Method I

**Input:**
  Event sequence $H = [t_0, t_1, \ldots, t_n]$
  Conditional intensity function $\lambda(t \mid H_t)$
  Time horizon $T$
  Counterfactual intensity function $\lambda^{\text{cf}}(t \mid H_t)$
  Step size $\Delta \in \mathbb{R}_{>0}$
**Output:**
  Event sequences, sampled from the counterfactual distribution $H^{\text{cf}}$
Split the time interval $[0, T]$ into steps of size $\Delta$. ▷ Assume $T/\Delta$ is an integer.
Define $\tau_i = i \cdot \Delta \quad \text{for} \quad i = 0, 1, 2, \ldots, \frac{T}{\Delta}$
**for** each interval $[\tau_i, \tau_{i+1})$ **do**
  Compute the event probability $P^i_{\text{event}} = \lambda(\tau_i \mid H_{\tau_i}) \cdot \Delta$
  **if** $\exists t' \in H$ such that $t' \in [\tau_i, \tau_{i+1}]$ **then** ▷ Event happened
    Sample $u_i$ uniformly at random from $[0, P^i_{\text{event}}]$
  **else** ▷ No event
    Sample $u_i$ uniformly at random from $[P^i_{\text{event}}, 1]$
  **end if**
**end for**
Initialize the counterfactual history $H^{\text{cf}} = []$
**for** each interval $[\tau_i, \tau_{i+1})$ **do**
  Compute the counterfactual intensity $\lambda^{\text{cf}}_i = \lambda^{\text{cf}}(\tau_i \mid H^{\text{cf}}_{\tau_i})$ for that interval.
  Compute the new threshold $\theta^{\text{cf}}_i = \lambda^{\text{cf}}_i \cdot \Delta$
  **if** $u_i \leq \theta^{\text{cf}}_i$ **then**
    Add $\tau_i$ to $H^{\text{cf}}$ ▷ Cf. event happens
  **end if**
**end for**
**Return** $H^{\text{cf}}$

**Algorithm 6** Generating Cf. TPP Sequences - Method II

**Input:**
  Event sequence $H = [t_0, t_1, \ldots, t_n]$
  Conditional intensity function $\lambda(t \mid H_t)$
  Counterfactual intensity function $\lambda^{\text{cf}}(t \mid H_t)$
  Step size $\Delta \in \mathbb{R}_{>0}$
**Output:**
  Event sequences, sampled from the counterfactual distribution $H^{\text{cf}}$
**for** $i \in [1, \ldots, n]$ **do**
  Compute $l_i = \int_{t_{i-1}}^{t_i} \lambda(t \mid H_t)\, dt$ ▷ Assuming $t_0 = 0$, use step size $\Delta$
**end for**
Initialize the counterfactual history $H^{\text{cf}} = []$
**for** $i \in [1, \ldots, n]$ **do**
  Compute $t^{\text{cf}}_i$ such that $l_i = \int_{t^{\text{cf}}_{i-1}}^{t^{\text{cf}}_i} \lambda^{\text{cf}}(t \mid H_t)\, dt$ ▷ Assuming $t^{\text{cf}}_0 = 0$
  Add $t^{\text{cf}}_i$ to $H^{\text{cf}}$
**end for**
**Return** $H^{\text{cf}}$

**Algorithm 7** Generating Cf. TPP Sequences - Method III

**Input:**
  Event sequence $H = [t_0, t_1, \ldots, t_n]$
  Upper bound $\lambda_{\max} \in \mathbb{R}_{>0}$
  Conditional intensity function $\lambda(t \mid H_t)$
  Counterfactual upper bound $\lambda^{\text{cf}}_{\max} \in \mathbb{R}_{>0}$
  Counterfactual intensity function $\lambda^{\text{cf}}(t \mid H_t)$
**Output:**
  Event sequences, sampled from the cf. distribution $H^{\text{cf}}$
Initialize the sequence of rejected $R = []$
Generate a homogeneous TPP sequence, $L$, with rate $\lambda_{\max}$.
  ▷ Sample a posterior sequence of rejected events
**for** each event $t_i$ in $L$ **do**
  Sample $u_i \in [0, 1]$
  **if** $u_i < \frac{\lambda_{\max} - \lambda(t_i \mid H_{t_{i-1}})}{\lambda_{\max}}$ **then** ▷ Part of shadow process
    Add $t_i$ to $R$
  **end if**
**end for**
  ▷ $H^*$ is the union of rejected and accepted events
Define $H^* = H \cup R$
Define $H^{\text{cf}} = []$ ▷ Cf. sequence of accepted events
Define $R^{\text{cf}} = []$ ▷ Cf. sequence of rejected events
**for** each $t_i$ in $H^*$ **do**
  Compute acceptance threshold: $\theta_i = \frac{\lambda(t_i \mid H_{t_i})}{\lambda_{\max}}$
  **if** $t_i \in H$ **then** ▷ Event accepted in original process
    Sample uniformly $u_i \in [0, \theta_i]$
  **else** ▷ Event was rejected in original process
    Sample $u_i \in [\theta_i, 1]$
  **end if**
  ▷ Shift event times according to the new upper bound
  Compute inter-event time $x_i = t_i - t_{i-1}$ (assuming $t_0 = 0$)
  Compute counterfactual inter-event time $x^{\text{cf}}_i = x_i \cdot \frac{\lambda_{\max}}{\lambda^{\text{cf}}_{\max}}$
  $t^{\text{cf}}_i = \max(R^{\text{cf}} \cup H^{\text{cf}}) + x^{\text{cf}}_i$ ▷ max of $\emptyset$ is 0.
  ▷ Decide acceptance in the cf. process.
  Compute acceptance probability $p'_i = \frac{\lambda^{\text{cf}}(t^{\text{cf}}_i \mid H^{\text{cf}})}{\lambda^{\text{cf}}_{\max}}$
  **if** $u_i < p'_i$ **then**
    Add $t^{\text{cf}}_i$ to $H^{\text{cf}}$
  **else**
    Add $t^{\text{cf}}_i$ to $R^{\text{cf}}$
  **end if**
**end for**
**Return** $H^{\text{cf}}$

counterfactual generation is complex and necessitates careful design choices. This is only one way of expressing epidemic spreading as a mechanistic model and that there are also more possibilities on how to model counterfactual distributions within this setting.

## 5.1 Model Formulation

We use a network-based stochastic model of epidemic spreading. That is, a given contact graph, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, specifies a set of agents (as nodes) and their connectivity (as edges). We use a simple continuous-time Susceptible-Infected model where nodes

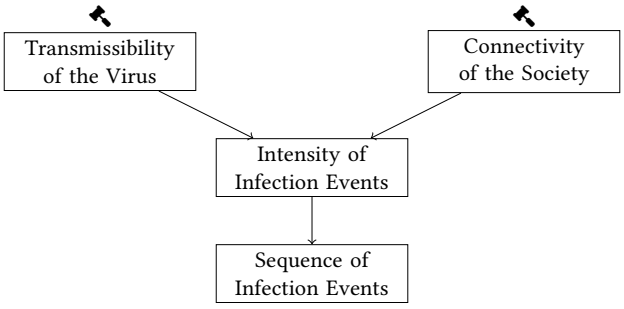


**Figure 3: The rate of infection events depends on both the properties of the virus and the characteristics of society. Counterfactual analysis of these properties (e.g., *"What if the virus was less transmissible?"* or *"What if society was less connected?"*) can lead to a different intensity of infection events, and consequently, to a different sequence of infection events in a non-obvious way.**

occupy one of two states: infected or susceptible. Infected nodes infect their susceptible neighbors after a random exponentially distributed waiting time (parametrized by the infection rate constant $\beta \in \mathbb{R}_{\geq 0}$). Infected nodes remain infected indefinitely. This leads to a continuous-time Markov chain (CTMC) semantics.

*5.1.1 Generating Trajectories.* We need a method to sample trajectories from this model that allows for the computation of counterfactuals. Specifically, the external noise should be fixed given a particular event sequence.

To stochastically generate a trajectory, we first sample a random matrix of waiting times $\mathbf{T} \in \mathbb{R}^{N \times N}$, where $t_{ij} \in \mathbb{R}_{\geq 0}$ follows an exponential distribution with the infection rate $\beta$. This value represents the time it takes for node $v_i$, after becoming infected, to infect node $v_j$ if they are adjacent. If node $v_j$ is already infected or the two nodes are not connected by an edge, the induced infection event has no influence. We start with a single randomly chosen infected node. In each step, we identify the next infection transmission based on the connectivity $\mathcal{E}$, the infection times $\mathbf{T}$, and the current time, and identify the next susceptible node to become infected. The simulation ends when all nodes are infected (which will eventually happen if the network is connected).

## 5.2 Counterfactuals

We assume that we observe which nodes get infected by which node at which point in time. Thus, the noise posterior is a distribution over $\mathbf{T}$. A given observed sequence fixes parts of the matrix $\mathbf{T}$ (the entries that correspond to actual new infections) but only provides bounds for the remaining entries.

We test two types of counterfactual manipulations (cf. Figure 3): changing the transmissibility of the virus (by changing $\beta$) and changing the connectivity of the contact graph (by changing $\mathcal{E}$).

## 5.3 Specification

We utilize a random graph where each pair of nodes is connected with a probability of $p = 0.2$. To test the counterfactual connectivity, we increase this probability to $p = 0.3$, resulting in five additional edges (cf. Figure 5). The infection rate is set to $\beta = 0.2$ for the original sequence and $\beta = 0.19$ for the counterfactual sequence with decreased transmissibility.

## 5.4 Results

The main results are reported in Figure 4 where the infection times of all nodes are reported as time-stamped events. Alternatively, the propagation of the infection without timestamps is also shown in Figure 6.

As expected, changing the infection rate merely decelerates the infection process without altering the infection chains. In contrast, changing the connectivity opens new potential pathways for the infection to spread. We observe that five nodes become infected through different pathways, leading to earlier infections.

The concept of monotonicity is maintained in this mechanistic model, as a decreased infection rate ensures that infections occur strictly later (i.e., a smaller $\beta$ can never cause a node to become infected earlier). Similarly, adding new edges cannot result in a node becoming infected later.

It is also important to note that for the intervention on the infection rate, the counterfactual is fixed, whereas changing the connectivity yields a distribution of counterfactuals. From this distribution, we present one arbitrary example.

# 6 Related Work

*Causality and TPPs.* Foundational work on the combination of point processes and causality was conducted by Gao et al. [Gao et al. 2021]. They demonstrated how to compute the average treatment effect on on marked TPPs.

However, for this work, the most relevant prior literature is the work by Noorbakhsh and Rodriguez [Noorbakhsh and Rodriguez 2022], who studied the problem of generating counterfactual sequences for TPPs. To the best of our knowledge, this is the only work that directly transforms a sampling technique into a method to sample counterfactual sequences. Their approach relies on certain assumptions, such as the ability to handle only linear Hawkes processes by leveraging its branching process interpretation. In their method, each event is attributed to a subsequent event. The algorithm processes all observed events sequentially, retaining only those events whose triggering event is preserved in the counterfactual distribution. For new events that did not occur in the original sequence, new processes are generated, and events are sampled from these processes using the superposition principle. This recursive process continues for all newly sampled events. Moreover, assigning events and generating new ones based on counterfactual events introduces additional stochasticity in an unprincipled way.

Their work is rooted in the transformation of a Markov decision process (MDP) into a Gumbel-Max SCM as proposed in [Oberst and Sontag 2019]. Using the Gumbel-Max distribution is beneficial because when the probability of an action taken in a counterfactual setting increases, it automatically means that this action is also taken. Likewise, if an action was not taken in the original sequence and its probability decreases in the counterfactual setting, it is also not taken in the counterfactual sequence.

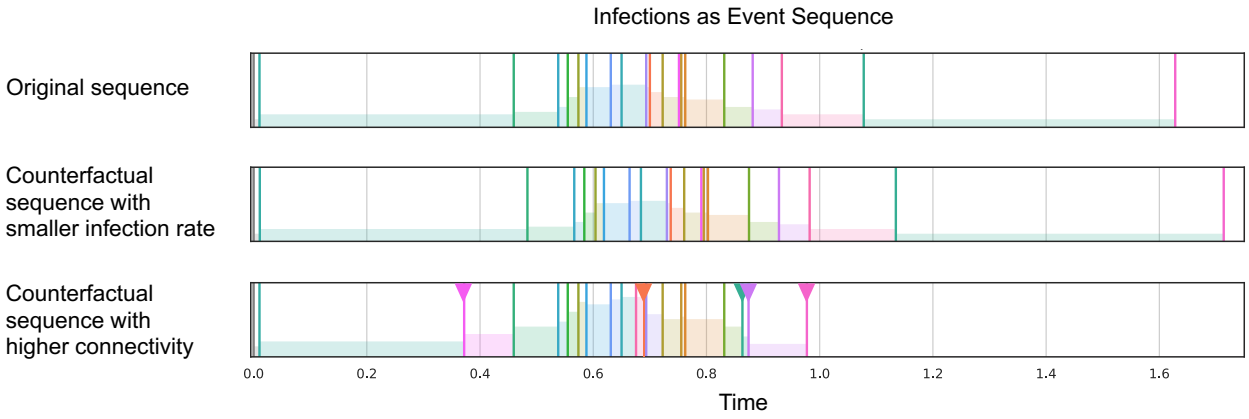


**Figure 4: Top row: Original sequence showing the infection times of all nodes. Middle row: Counterfactual sequence with decreased virus transmissibility. Bottom row: Counterfactual sequence with increased connectivity of the contact graph. Nodes that become infected at a different timepoint compared to the original sequence are annotated with a triangle. Colors identify nodes and correspond to Figure 5. The shaded area indicates the instantaneous rate of an event, which is determined by the number of SI edges and the infection rate.**

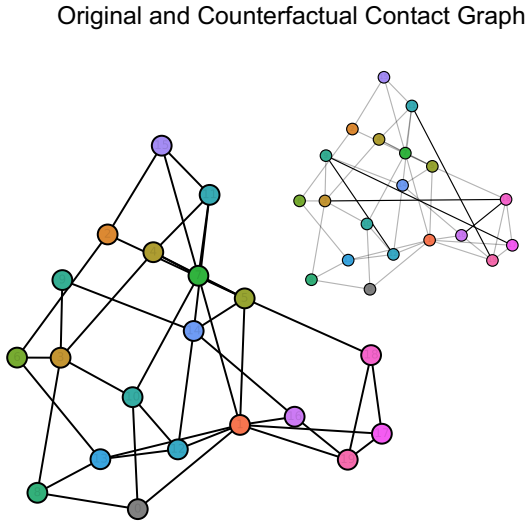


**Figure 5: The original contact graph is shown on the left. The contact graph (undirected as per standard practise) used for counterfactual generation is shown on the right. It was created by increasing the probability of two pairs of nodes being connected, resulting in five additional edges. Only the new edges are shown in black, while edges that are also present in the original graph are shown in gray.**

Computing counterfactuals in the context of Reinforcement Learning has since been a widely researched topic, with most work focusing on off-policy evaluation and optimizing hypothetical action sequences [Hızlı et al. 2022; Parbhoo et al. 2022; Tsirtsis and Rodriguez 2024].

Jing et al. [Jing et al. [n. d.]] explore counterfactuals in TPPs within healthcare, specifically optimizing treatment strategies by answering counterfactual questions. They build directly on the framework by Noorbakhsh et al. but focus on sampling and optimizing counterfactual treatment roll-outs using temporal logic rules to enforce plausibility.

Moreover, [Hızlı et al. 2023] use a combination of Gaussian Processes and marked TPPs to construct a causal model of the treatment–outcome relationship within patients, learned from observational data. They extend the method by Noorbakhsh et al. to generate counterfactual samples in a more flexible way. In their follow-up work [Hızlı et al. 2024], this framework is extended to accurately capture the direct and indirect effects of interventions.

*Counterfactuals in Neural TPPs.* Neural networks are also used to generate counterfactual trajectories, as an alternative approach to Pearl's do-calculus. Bica et al. [Bica et al. 2020] generate counterfactuals in point processes using RNNs. They employ data from a simulated tumor growth model and real-world data from the *Medical Information Mart for Intensive Care* database. Their focus is on adversarial training to create representations invariant to treatment assignments. Essentially, they learn a model that can simulate future states based on a learned representation where information predicting which treatment is taken is removed. This approach differs from traditional counterfactual methods that compute a posterior of the noise distribution, thereby using information from the observed trajectory to shape the counterfactual trajectory. Likewise, Zhang et al. construct a causal model to study the influence of fake news in social media as a point process, where they use domain adversarial training to account for the bias introduced by personalized recommendation systems [Zhang et al. 2022].

*Ambiguity of Counterfactuals.* A comprehensive introduction to the meaning of counterfactuals in different frameworks (Rubin's Causal Model, Lewis' Theory of Counterfactual Conditionals, Pearl's do-calculus) is provided by [de Lara 2024; Markus 2021]. For an introduction to causal models that incorporate space and time, we refer to [Kang et al. 2024]. Discussions on how well these notions

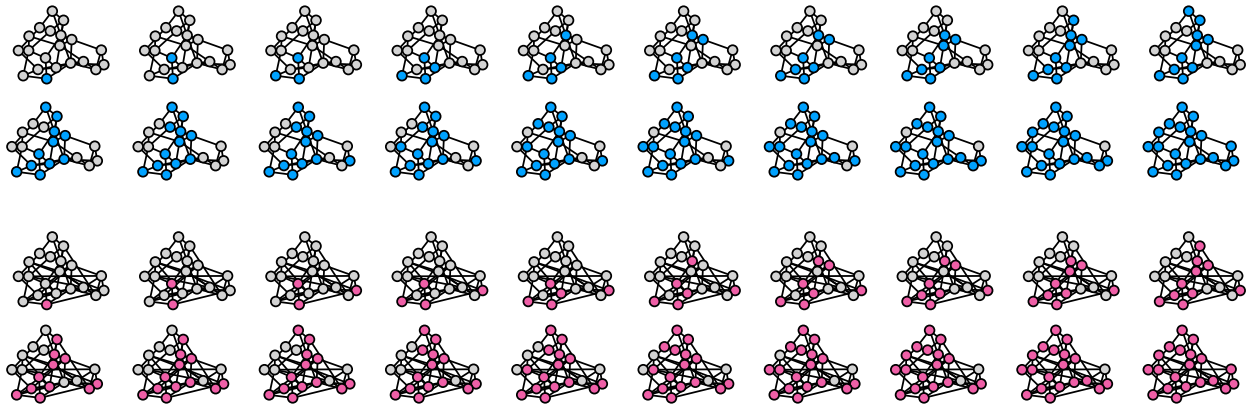

**Figure 6: Trajectory of disease propagation (I: blue/purple, S: gray) in the original contact graph (top) and in the counterfactual contact graph with five additional edges (bottom).**

correspond to our intuitive understanding of "*What if*" questions are rare. However, [Chvykov 2021; Gundersen and Kallestrup 2023] investigate paradoxes and discusses potential resolutions. We will not explore the philosophical literature in detail but refer interested readers to [Hájek 2014; Lebow 2000; Lewis 2016].

## 7 Conclusions and Future Work

In this work, we introduced three straightforward algorithms to generate counterfactual event sequences based on a hypothetical intensity function. These algorithms were derived directly from stochastic simulation algorithms and their corresponding structural causal models, thereby circumventing the need for a Gumbel-Max construction. Furthermore, we demonstrated that these three methods produce different types of counterfactuals. This diversity does not imply incorrectness but rather indicates significant flexibility in design choices. Using a mechanistic model of epidemic spreading, we investigated the conditions under which certain counterfactual algorithms might more accurately reflect the physical reality modeled by the intensity function.

Overall, this work illustrates that generating counterfactual sequences is computationally simple, but the implicit design choices must be carefully considered to ensure they reflect relevant aspects of reality.

## Acknowledgments

This work was funded by the Bundesministerium für Bildung und Forschung - (BMBF) under Grant 01IW23005. We thank Georgi Vitanov for helpful comments on the manuscript. We thank the anonymous reviewers of the STCausal workshop for their constructive feedback.

## References

Ioana Bica, Ahmed M Alaa, James Jordon, and Mihaela van der Schaar. 2020. Estimating counterfactual treatment outcomes over time through adversarially balanced representations. *arXiv preprint arXiv:2002.04083* (2020).

Pavel Chvykov. 2021. *Does Butterfly Affect.* https://www.lesswrong.com/posts/quQwHZTaRDJFoTYLd/does-butterfly-affect The article discusses the conventional understanding of the butterfly effect and its applicability in real-world complex systems..

Wesley Cota and Silvio C Ferreira. 2017. Optimized Gillespie algorithms for the simulation of Markovian epidemic processes on large and heterogeneous networks. *Computer Physics Communications* 219 (2017), 303–312.

Lucas de Lara. 2024. On the (in) compatibility between potential outcomes and structural causal models and its signification in counterfactual inference. (2024).

Tian Gao, Dharmashankar Subramanian, Debarun Bhattacharjya, Xiao Shou, Nicholas Mattei, and Kristin P Bennett. 2021. Causal inference for event pairs in multivariate point processes. *Advances in Neural Information Processing Systems* 34 (2021), 17311–17324.

Gerrit Großmann, Luca Bortolussi, and Verena Wolf. 2020. Efficient simulation of non-Markovian dynamics on complex networks. *Plos one* 15, 10 (2020), e0241394.

Lars Bo Gundersen and Jesper Kallestrup. 2023. Counterfactuals, irrelevant semifactuals and the 1.000. 000 bet. *Inquiry* (2023), 1–17.

Alan Hájek. 2014. Most counterfactuals are false. (2014).

Joseph Y. Halpern. 2019. *Actual Causality* (2 ed.). The MIT Press, Cambridge, MA.

Brian Hedden. 2023. Counterfactual decision theory. *Mind* 132, 527 (2023), 730–761.

Çağlar Hızlı, ST John, Anne Juuti, Tuure Saarinen, Kirsi Pietiläinen, and Pekka Marttinen. 2024. Temporal causal mediation through a point process: direct and indirect effects of healthcare interventions. *Advances in Neural Information Processing Systems* 36 (2024).

Çağlar Hızlı, ST John, Anne Tuulikki Juuti, Tuure Tapani Saarinen, Kirsi Hannele Pietiläinen, and Pekka Marttinen. 2022. Joint point process model for counterfactual treatment-outcome trajectories under policy interventions. In *NeurIPS 2022 Workshop on Learning from Time Series for Health.*

Çağlar Hızlı, ST John, Anne Tuulikki Juuti, Tuure Tapani Saarinen, Kirsi Hannele Pietiläinen, and Pekka Marttinen. 2023. Causal modeling of policy interventions from treatment-outcome sequences. In *International Conference on Machine Learning*. PMLR, 13050–13084.

Zilin Jing, Chao Yang, and Shuang Li. [n. d.]. Counterfactual Optimization of Treatment Policies Based on Temporal Point Process. In *ICML 3rd Workshop on Interpretable Machine Learning in Healthcare (IMLH).*

Mingyu Kang, Duxin Chen, Ziyuan Pu, Jianxi Gao, and Wenwu Yu. 2024. Spatio-Temporal Graphical Counterfactuals: An Overview. *arXiv preprint arXiv:2407.01875* (2024).

Minkyoung Kim, Dean Paini, and Raja Jurdak. 2019. Modeling stochastic processes in disease spread across a heterogeneous social system. *Proceedings of the National Academy of Sciences* 116, 2 (2019), 401–406.

Junhyeon Kwon, Yingcai Zheng, and Mikyoung Jun. 2023. Flexible spatio-temporal Hawkes process models for earthquake occurrences. *Spatial Statistics* 54 (2023), 100728.

Richard Ned Lebow. 2000. What's so different about a counterfactual? *World politics* 52, 4 (2000), 550–585.

Karen S Lewis. 2016. Elusive counterfactuals. *Noûs* 50, 2 (2016), 286–313.

Scott Linderman and Ryan Adams. 2014. Discovering latent network structure in point process data. In *International conference on machine learning*. PMLR, 1413–1421.

Keith A Markus. 2021. Causal effects and counterfactual conditionals: contrasting Rubin, Lewis and Pearl. *Economics & Philosophy* 37, 3 (2021), 441–461.

Kimia Noorbakhsh and Manuel Rodriguez. 2022. Counterfactual temporal point processes. *Advances in Neural Information Processing Systems* 35 (2022), 24810–24823.

Michael Oberst and David Sontag. 2019. Counterfactual off-policy evaluation with gumbel-max structural causal models. In *International Conference on Machine Learning*. PMLR, 4881–4890.
Yosihiko Ogata. 1981. On Lewis' simulation method for point processes. *IEEE transactions on information theory* 27, 1 (1981), 23–31.
Sonali Parbhoo, Shalmali Joshi, and Finale Doshi-Velez. 2022. Generalizing off-policy evaluation from a causal perspective for sequential decision-making. *arXiv preprint arXiv:2201.08262* (2022).
Judea Pearl. 2009. *Causality*. Cambridge university press.
Judea Pearl, Madelyn Glymour, and Nicholas P Jewell. 2016. *Causal inference in statistics: A primer*. John Wiley & Sons.
Jakob Gulddahl Rasmussen. 2018. Lecture notes: Temporal point processes and the conditional intensity function. *arXiv preprint arXiv:1806.00221* (2018).
Alex Reinhart. 2018. A review of self-exciting spatio-temporal point processes and their applications. *Statist. Sci.* 33, 3 (2018), 299–318.
M Gomez Rodriguez and Isabel Valera. 2018. Learning with temporal point processes. *Tutorial at ICML* (2018).
Stratis Tsirtsis and Manuel Rodriguez. 2024. Finding counterfactually optimal action sequences in continuous state spaces. *Advances in Neural Information Processing Systems* 36 (2024).
Yizhou Zhang, Defu Cao, and Yan Liu. 2022. Counterfactual neural temporal point process for estimating causal influence of misinformation on social media. *Advances in Neural Information Processing Systems* 35 (2022), 10643–10655.

# A Extensions to Spatio-Temporal Point Processes

A spatio-temporal point process (STPP) is a type of point process where events are associated with locations in space (typically 2-dimensional Euclidean space) as well as time. The data is represented as a sequence of tuples:

$$H = [(t_1, \mathbf{s}_1), (t_2, \mathbf{s}_2), (t_3, \mathbf{s}_3), \ldots],$$

where $t_i$ denotes the time of the event, and $\mathbf{s}_i \in \mathcal{S} \subset \mathbb{R}^2$ represents the spatial location in a space.

To define the intensity function, we consider an infinitesimal grid over both space and time:

$$\lambda(\mathbf{s}, t \mid H_t) := \lim_{\Delta_s \to 0, \Delta_t \to 0} \frac{P(\text{Event in } B(\mathbf{s}, \Delta_s) \times [t, t + \Delta_t) \mid H_t)}{|B(\mathbf{s}, \Delta_s)|\, \Delta t},$$

where $|B(\mathbf{s}, \Delta_s)|$ is the Lebesgue measure (i.e., the area) of $B(s, \Delta_s)$, a 2D ball (i.e., disk) centered at $\mathbf{s}$ with radius $\Delta_s$.

Although we define the infinitesimal area around a point using a disk, in practice, we use grids with a uniform size of $\Delta$ in all three dimensions.

Next, we discuss how to generalize the three methods for sampling from the counterfactual distributions to the spatio-temporal case. In most cases, the generalization is straightforward, with Method II being the only one where we encounter some conceptual choices. For ease of notation, we define the marginalized intensity over time as $\lambda(t \mid H_t) := \int_{s \in \mathcal{S}} \lambda(\mathbf{s}, t \mid H_t)$.

We can also establish new desirable properties for the model. For example, when the intensity function changes only within a specific time slice, meaning $\lambda(t \mid H_t) = \lambda^{\text{cf}}(t \mid H_t)$, we would expect that in the counterfactual sequence, only the spatial coordinates of the events change, while the timestamps remain unaffected.

## Method I

We begin by discretizing the space in all three dimensions using a grid of size $\Delta$ (using the same grid size in all dimensions w.l.o.g.). Next, we iterate over the grids sequentially from left to right, corresponding to increasing time, and determine whether an event occurs within each grid. Note that there is no natural order among the grids within a specific time interval. Therefore, it is essential to make the grid size sufficiently small to avoid having multiple events at the same time point, which would introduce ambiguity.

To decide whether an event occurs within a grid, one can multiply the event rate by the grid volume. Specifically, the probability of an event occurring in a grid located at $(\mathbf{s}, t)$ is given by:

$$P^{\mathbf{s},t}_{\text{Event}} = \lambda(\mathbf{s}, t \mid H_t) \times \Delta^3.$$

After computing this probability, the noise posterior $u_{\mathbf{s},t}$ can be calculated for each grid. This posterior can then be used to determine the counterfactual sequence.

## Method II

The rationale behind Method II is that we fix the integral of the intensity function between to consecutive events: $\int_{t_i}^{t_{i+1}} \lambda(t) = \int_{t_i^{\text{cf}}}^{t_{i+1}^{\text{cf}}} \lambda^{\text{cf}}(t)$ and that the size of the integral is determined by an exogenous noise variable. That notion works in space but has no direct translation to time.

However it is straight-forward to move the points in space in order for this condition to be satisfied. Therefore, use the marginalized $\lambda(t \mid H_t)$. This allows us to move the timepoints of all events to be consistent with a counterfactual intensity function.

Moving the spatial coordinates is less straightforward and may not necessarily yield a unique solution. Formally, the challenge arises because we have two intensity maps, $f(\mathbf{s}) := \lambda(\mathbf{s}, t_i \mid H_t)$ and $f^{\text{cf}}(\mathbf{s}) := \lambda^{\text{cf}}(\mathbf{s}, t_i^{\text{cf}} \mid H_{t_i^{\text{cf}}})$, and we need a method to move the event coordinate from $\mathbf{s} = (x_1, x_2)$ to $\mathbf{s}^{\text{cf}} = (x_1^{\text{cf}}, x_2^{\text{cf}})$.

One possible approach is to factorize the spatial dimensions, first moving the $s_1$ coordinate and then the $s_2$ coordinate. Although this method yields a unique mapping, the choice of which dimension to transform first remains somewhat arbitrary. An alternative approach is to normalize the two functions $f(\cdot)$ and $f^{\text{cf}}(\cdot)$, and then compute a transport map based on the Wasserstein distance. This transport map can subsequently be applied to $\mathbf{s}$ to obtain the transformed coordinates $\mathbf{s}^{\text{cf}}$.

## Method III

Fortunately, it is straightforward to generalize Method III. The first step remains the same: we sample from the shadow process and record the posterior noise responsible for determining whether an event was rejected or not. Note that we therefore sample events in time following an exponential distribution and in space following a uniform distribution. Next, we consider the marginalized $\lambda_{\max}(t \mid H_t)$ and adjust the event times (of both rejected and accepted) by shifting them either to the left or to the right, depending on the counterfactual $\lambda_{\max}$. The event coordinates remain the same. Finally, we stochastically swap the rejected and accepted labels of the events based on the counterfactual intensity and the posterior noise obtained in the first step.

- The first step is equivalent: you sample from the shadow process.
- The second step is equivalent: you move the points to the left or right in time.
- The third step is similar to the TPP version: you swap events between the original and the shadow process based on the new intensity.

# B Counterfactual Illustrative Example

Consider a model for tossing a biased coin (cf. Figure 7). We have a noise variable $U$ that follows a uniform distribution on the interval $[0, 1]$, and a variable $V \in \{\text{HEADS}, \text{TAILS}\}$ that determines which side of the coin is facing up. We can define $V$ as a function of $U$:

$$V = f(U) = \begin{cases} \text{HEADS} & \text{if } U < 0.4, \\ \text{TAILS} & \text{if } U \geq 0.4. \end{cases}$$

Now, suppose we observe TAILS. The information, we now have on the noise variable is that it is uniformly distributed in $[0.4, 1]$. This is expressed in the posterior noise distribution. Suppose, we are interested in finding the counterfactual distribution under the assumption that the coin was fair. We can achieve this by modifying the SCM and defining $V = f^{\text{cf}}(U)$ as:

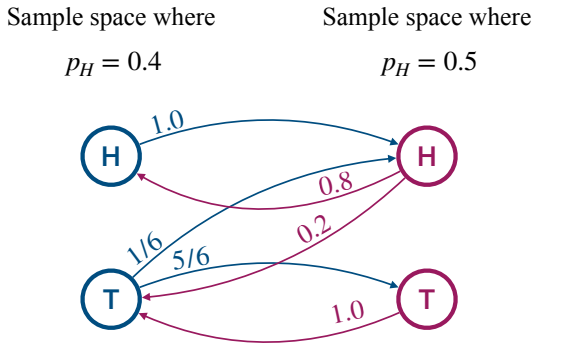


**Figure 7: Counterfactual coin flip: probability of heads being 40% (left) or 50% (right).**

$$V = f^{\text{cf}}(U) = \begin{cases} \text{HEADS} & \text{if } U < 0.5, \\ \text{TAILS} & \text{if } U \geq 0.5. \end{cases}$$

Sampling from the posterior distribution of the noise variable $U$ in this modified model would result in HEADS with probability $\frac{1}{6}$ and TAILS with probability $\frac{5}{6}$. This setting satisfies monotonicity because we increase the probability of HEADS, and all scenarios that lead to HEADS in the original setting will also lead to HEADS in the counterfactual setting. However, we could also define the counterfactual SCM as:

$$V = f^{\text{cf}}(U) = \begin{cases} \text{TAILS} & \text{if } U < 0.5, \\ \text{HEADS} & \text{if } U \geq 0.5. \end{cases}$$

This would not be a monotonic model because, even though the general probability of HEADS increases, all the samples that would have been HEADS in the original setting would switch to being TAILS.