TIME-CONTINUOUS PRODUCTION NETWORKS WITH RANDOM BREAKDOWNS

SIMONE GÖTTLICH*, STEPHAN MARTIN*, AND THORSTEN SICKENBERGER†

Abstract. Our main objective is the modelling and simulation of complex production networks originally introduced in [15, 16] with random breakdowns of individual processors. Similar to [10], the breakdowns of processors are exponentially distributed. The resulting network model is of coupled system of partial and ordinary differential equations with Markovian switching and its solution is a stochastic process. We show our model to fit into the framework of piecewise deterministic processes, which allows for a deterministic interpretation of dynamics between a multivariate two-state process. We develop an efficient algorithm with an emphasis on accurately tracing stochastic events. Numerical results are presented for three exemplary networks, including a comparison with the long-chain model proposed in [10].

AMS subject classification: 90B15, 65Cxx, 65Mxx

Key words: production networks, conservation laws, random breakdowns, stochastic differential equation, coupled PDE-ODE systems, Markovian switching, piecewise deterministic processes (PDPs).

1. Introduction. Tackling transportation and dynamic flows on networks by partial differential equations is nowadays discussed in engineering areas like traffic flow [6, 19, 21], gas networks [2, 3], water pipelines [26, 31] or supply chains [1, 5, 15, 20]. Those network problems have gained great attention in the last decade and are recently used for optimization purposes, see [11, 17, 18, 25, 29, 30] for an overview. However, the aforementioned models are normally deterministic and stochastic dynamics have not been studied yet extensively, with a few exceptions such as the work of Degond and Ringhofer [10].

Analysing the dynamical behaviour of manufacturing systems is one important issue in managing or planning supply chains. Generally, goods or parts pass through different stages of production according to customer demand and cost-efficiency. Mathematical models are able to capture system-dependent attributes and predict the time evolution of moving goods. The modelling basically allows for two unrelated ways: either parts are tracked individually through the production process or, alternatively, average quantities are used to calculate the total load. This leads to the established class of discrete event simulations (see [1]) and continuous models, respectively. In order to avoid the computational expensiveness of discrete event simulations, we focus on dynamical models based on ordinary and partial differential equations. The latter are capable to study large-scale production systems and furthermore admit a detailed analytical treatment in the deterministic setting [7, 22].

In this paper, we are interested in the modelling and simulation of time-continuous production networks with random breakdowns of processors. Starting from a purely deterministic model that might incorporate simulation as well as optimisation purposes [15, 16] we develop a stochastic framework describing the situation of unexpected failures. The basic idea is that each processor has a finite maximal capacity which is set to zero in case the processor is shut down. We suppose the total breakdown of an individual processor is a critical point in the system and failure rates are used to describe the frequency at which the processors in the network fail, cf. [10]. In

*Technische Universität Kaiserslautern, Fachbereich Mathematik, 67663 Kaiserslautern, Germany (E-mail:{goettlich,smartin}@mathematik.uni-kl.de)

†Maxwell Institute and Heriot-Watt University, Dept. of Mathematics, Edinburgh, EH14 4AS, Scotland, UK (E-mail: t.sickenberger@hw.ac.uk)

our network model, the random breakdown of a processor is modelled by breaking down its maximal capacity at exponentially distributed time points. We will see that stochastic effects will affect the dynamic behaviour of the system significantly.

This modelling approach leads to a stochastic network model, where the solution is a piecewise deterministic process (PDP). A PDP consists of a mixture of deterministic descriptions and random jumps [8, 9]. The random jumps are given by a multivariate state process, which models the current state of the processors. Hence, the randomness appears only as point events, i.e., there is a sequence of random occurrences at random times when one of the processors is switching from operating to in-operating state or the other way around, but there is no additional component of uncertainty between these times. The underlying deterministic part is a network model described by a coupled systems of deterministic partial differential equation and ordinary differential equations (coupled PDE-ODE system). In contrast to time-discrete network formulations, the queueing strategy is described by an ODE and the physical production process of each processor in the network is modelled by a hyperbolic PDE. Combining all ingredients of the stochastic system we get a coupled PDE-ODE model with Markovian switching. These systems are quite different from stochastic networks that are queueing-type Markov chains and usually discrete in time and space [24, 34].

The simulation of time-continuous network models requires appropriate numerical methods which can handle the different time behaviour of the dynamics as well as the random breakdowns of processors. We therefore introduce a modification of the stochastic simulation algorithm which can be used to perform Monte-Carlo simulations of our network model for general topologies. In this way, we simulate the production network many times and average over the sampled realisations to determine several quantities. We are mainly interested in exception and deviation of processors density, the queue loads and the network outflow versus time.

The structure of this paper is as follows: In Section 2 we start with a description of the deterministic model, then we model the random breakdowns of single processors by a stochastic jump process and add these processes to the network model. In Section 3 we derive a multivariate state process describing the system of processors based on the single jump processes of each processor. Moreover, we point out that the stochastic solution of our model is piecewise deterministic and the multivariate state process is a random constant between switching points. This plays a decisive role, since we can apply a stochastic simulation algorithm to compute sample realisations and scenarios. The algorithm is based on two parts: (i) The sampling of the switching points and the states of the multivariate state process, and (ii) The approximation of the PDE-ODE part by deterministic numerics. Details of the sampling, the time and space discretisation and the numerical methods are given in Section 4. With a numerical simulation tool at hand, we perform Monte-Carlo simulations in Section 5. At first, stochastic default is considered for a chain of processors and results are brought in line with the model of Degond and Ringhofer ([10]). Secondly, control parameters for a small-scale stochastic network are investigated. In the last example, our algorithm is demonstrated on larger cascade networks.

2. Modelling of production networks with random breakdowns. We start with an introduction to the deterministic model. In a second step, we model the random breakdowns of the processors by a set of two-state processes with exponentially distributed switching times between two states. Finally, we combine both approaches and give a stochastic model of the network with random breakdowns of

the processors.

2.1. The deterministic network model. Time-continuous deterministic network models of serial networks have been introduced in [1], where the authors derived a conservation law for the part density of processors. In [15, 16], this model has been reformulated by installing queues in front of processors. This approach can be applied to more complex network topologies and considers densities of the processors as well as loads of the queues. We are now concerned with the latter approach, which is given by the following description.

First, we set up a couple of notation. Let $(\mathcal{V}, \mathcal{A})$ denote a directed graph consisting of a set of arcs $\mathcal{A}$ and a set of vertices $\mathcal{V}$ and define $N := |\mathcal{V}|, M := |\mathcal{A}|$. For any fixed vertex $v \in \mathcal{V}$, the set of ingoing arcs is denoted by δ_v^- and the set of outgoing arcs by δ_v^+. In our network model, each arc represents a processor with its preceding queue and each vertex is a distribution node between different processors. Each processor e $(e = 1, \dots, M)$ is identified with the real interval $[a^e, b^e]$, where $b^e > a^e$, and its length is $\ell^e = b^e - a^e$ accordingly.

A vertex without predecessors act as a network inflow point. We collect all of these vertices in $\mathcal{V}_{\text{in}} := \{v \in \mathcal{V} : |\delta_v^-| = 0\}$. Their influx is externally given as an inflow function of time denoted by $G_{\text{in}}^v(t)$ for all $v \in \mathcal{V}_{\text{in}}$. Likewise, define the vertices of product outflows from the network by $\mathcal{V}_{\text{out}} := \{v \in \mathcal{V} : |\delta_v^+| = 0\}$ and let $\mathrm{s} : \mathcal{A} \to \mathcal{V}$ map any arc onto its vertex of origin.

Assume that we like to model and simulate the network quantities over a time period $[t_0, T]$. Then the time-continuous network model describes the evolution of the density of goods $\rho^e(x,t)$ inside each supplier e and the time evolution of the buffer $q^e(t)$ belonging to supplier e. For $e = 1, \dots, M$ denote

$$\rho^e : [a^e, b^e] \times \mathbb{R}_0^+ \to \mathbb{R}_0^+ : (x,t) \mapsto \rho^e(x,t)$$

the density of products at position x and time t in processor e as well as

$$q^e : \mathbb{R}_0^+ \to \mathbb{R}_0^+ : t \mapsto q^e(t)$$

the load of the queue in front of processor e at time t. We abbreviate $\mathbf{p} := (\rho^1, \dots, \rho^M)$, $\mathbf{q} := (q^1, \dots, q^M)$. Each processor is working with a constant velocity v^e and has a maximal processing rate μ^e that is constant, too. This means that on each arc e the density $\rho^e(x,t)$ is transported with velocity v^e if the flux of goods is less or equal than the maximal processing rate μ^e. Hence, the density ρ^e satisfies the partial differential equations (PDE) of hyperbolic type

$$\partial_t \rho^e(x,t) + \partial_x f^e(\rho^e(x,t)) = 0\,, \quad x \in [a^e, b^e],\ t \in [t_0, T]\,, \tag{2.1a}$$

where the relation between flux and density is given by the flux function

$$f^e(\rho^e) = \min(v^e \cdot \rho^e, \mu^e)\,. \tag{2.1b}$$

REMARK 2.1. *The benefit of representation* (2.1b) *is the easy inclusion of random breakdowns. To model a complete failure of a single processor, the maximal processing rate will be set to zero and hence the processing rate is considered as a time-dependent random variable, see Subsection 2.3.*

For each processor e, we introduce a buffer zone for the incoming but not yet processed goods. This buffering is modelled by the time-dependent function $q^e(t)$ where the dynamics is governed by the difference of all incoming and outgoing fluxes

denoted by the functions g^e_{in} and g^e_{out}, respectively. Hence, the load q^e of a queue fulfils the ordinary differential equation (ODE)

$$\partial_t q^e(t) = g^e_{\text{in}}(t) - g^e_{\text{out}}(t)\,, \quad t \in [t_0, T]. \tag{2.1c}$$

For the ingoing flux $g^e_{\text{in}}(t)$, we have the following model: If the origin of processor e does not belong to the set of network inflow points $\mathcal{V}_{\text{in}}$, the incoming flux is given by the sum of all incoming fluxes multiplied by the distribution rate $A^{v,e}(t)$. For any outgoing arc, we introduce distribution rates $A^{v,e}(t)$, $v \in \mathcal{V}$ such that $0 \le A^{v,e}(t) \le 1$ and $\sum_{e\in\delta^+_v} A^{v,e}(t) = 1\,\forall v$, thereby $A^{v,e} = 1$ if $|\delta^+_v| = 1$. Those rates describe the distribution of incoming parts among the outgoing processors, and represent natural controls of the production network. If the origin of processor e belongs to the set of network inflow points $\mathcal{V}_{\text{in}}$ the incoming flow is the inflow function $G^{s(e)}_{\text{in}}(t)$, where $s(e)$ is the origin of processor e:

$$g^e_{\text{in}}(t) = \begin{cases} A^{s(e),e}(t) \sum_{\bar{e}\in\delta^-_{s(e)}} f^{\bar{e}}(\rho^{\bar{e}}(b^{\bar{e}}_v, t)) & \text{if } s(e) \notin \mathcal{V}_{\text{in}}, \\ G^{s(e)}_{\text{in}}(t) & \text{if } s(e) \in \mathcal{V}_{\text{in}}. \end{cases} \tag{2.1d}$$

For the outgoing flux $g^e_{\text{out}}(t)$ in (2.1c), we consider the following model: If the queue is empty, the outgoing flux is the minimum of the incoming flux and the processing rate μ^e. If the buffer is not empty, the buffer is always reduced with a capacity determined by processing the maximal capacities of the processor:

$$g^e_{\text{out}}(t) = \begin{cases} \min\{g^e_{\text{in}}(t)\,,\, \mu^e\} & \text{if } q^e(t) = 0, \\ \mu^e & \text{if } q^e(t) > 0. \end{cases} \tag{2.1e}$$

REMARK 2.2. *Since we are mainly interested in the simulation of stochastic networks, finding optimal distribution rates $A^{s(e),e}(t)$ in Eq.* (2.1d) *is not the focus of this work. Therefore, one can think of constant rates $\alpha^{s(e)}$, e.g. the naive control*

$$\alpha^{s(e)} := 1/|\delta^+_{s(e)}| \tag{2.1f}$$

in the following.

In order to complete the network formulation, we introduce the following initial values of the ODE as well as of the PDE

$$\rho^e(x, t_0) = \rho^e_0(x),\ q^e(t_0) = q^e_0 \text{ (PDE and ODE initial conditions (i.c.))} \tag{2.1g}$$

and the boundary conditions of the PDE, which are given by the outgoing flow of the queues

$$\rho^e(a^e, t) = g^e_{\text{out}}(q^e(t))/v^e \text{ (PDE boundary conditions (b.c.))}. \tag{2.1h}$$

This boundary condition represents the coupling of the processors at their connection points.

Putting all these equations together, we end up with the following definition of a solution.

DEFINITION 2.3 (The time-continuous network model). *Let $[t_0, T]$ be a fixed real interval. Then, we call $(\mathbf{p}, \mathbf{q})$ a* solution of the time-continuous network model *on $[t_0, T]$ if it fulfils equations* (2.1a) - (2.1h). *The solution depends on the initial values $\mathbf{p}(t_0) = \mathbf{p}_0$ and $\mathbf{q}(t_0) = \mathbf{q}_0$ at time t_0.*

Apart from the solution components, the total outflow of the network will be an interesting performance measure for our numerical experiments, see Section 5. The time-dependent outflow of any outgoing vertex $v \in \mathcal{V}_{\text{out}}$ can be computed by

$$G^v_{\text{out}}(t) = \sum_{e \in \delta^-_v} f^e(\rho^e(v^e, t)).$$

Hence, the time evolution of all produced goods of the network is given by

$$G^{\text{net}}_{\text{out}}(t) = \sum_{v \in \mathcal{V}_{\text{out}}} \int_{t_0}^{t} G^v_{\text{out}}(s)\,\mathrm{d}s\,. \tag{2.2}$$

2.2. Modelling random breakdowns of processors. The key idea in modelling random breakdowns of processors is to switch the individual maximal processing rates between on or off states, i.e. either the processing rate is on and possesses goods with rate μ^e as usual or the processing rates is turned off by setting $\mu^e = 0$. This procedure is a Markov process since we assume the probability to change the state is independent of the time elapsed since the last switch. Naturally, these on or off durations are exponentially distributed since they emanate from a memoryless process, as suggested in [10]. In order to incorporate random breakdowns of processors in a mathematical framework, we define a set of two-state stochastic process

$$r^e : \mathbb{R}^+_0 \times \Omega \to \{0, 1\} : t \times \omega \mapsto r^e(t, \omega)\,,\ e = 1, \ldots, M$$

indicating whether the processor e is operating or in-operating. The state process $r^e(\cdot)$ depends on the time as well as on the random sample $\omega \in \Omega$, whose argument ω are usually dropped. For fixed time $r^e(t, \cdot)$ is a random variable which takes values in $\{0, 1\}$, and for a fixed random sample $r^e(\cdot, \omega)$ is a realisation or scenario of the state process shown in Fig. 2.1.

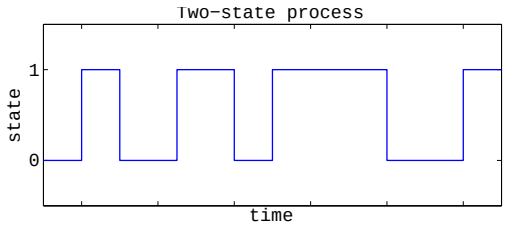

Fig. 2.1: A realisation of the two-state process r^e: Values in $\{0, 1\}$ and exponentially distributed times between switchings.

We assume that each processor e goes through alternating operating and in-operating periods, which are distributed exponentially. If $r^e(t) = 1$ the processor e is operating and the case $r^e(t) = 0$ indicates that the processor is in-operating and under repair. We call a change in the status r^e of a processor *switching*. Transitions from the operating to the in-operating state are called *breakdowns*, while those from the in-operating to the operating state are *repairs*.

The operating and in-operating periods are independent of the states of the other processors and of the number of goods in the queue as well as of the density of goods in the processor. To model the switching between the periods we introduce

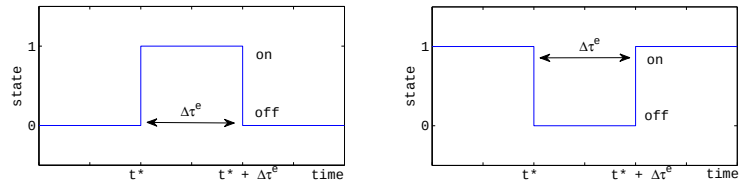

Fig. 2.2: A processor switching from off to on and back (left) and vice versa (right).

two parameters for each processor: Let τ^e_{on} describe the mean time between failures (MTBF) and let τ^e_{off} describe the mean repair time (MRT), respectively.

Assuming that the processor e at time $t^* + \Delta\tau^e$ has just switched back from operating to the in-operating status ($r^e(t^*+\Delta\tau^e) = 0$) or in-operating to the operating status ($r^e(t^* + \Delta\tau^e) = 1$) the length $\Delta\tau^e$ of that period is chosen randomly from the exponential distribution with rate parameter $\lambda(r^e(t^*))$. The rate parameter is given by

$$\lambda(r^e(t^*)) = \begin{cases} 1/\tau^e_{\text{on}}\,, & \text{if } \; r^e(t^*) = 1, \\ 1/\tau^e_{\text{off}}\,, & \text{if } \; r^e(t^*) = 0. \end{cases} \tag{2.3}$$

and the density function $Exp(t;\lambda)$ of the exponential distribution where $\lambda = \lambda(r^e(t^*))$.

Hence, from the mathematical viewpoint the state process $r^e(t)$, $t \geq t_0$ is a right-continuous continuous-time Markov process taking values in $\{0, 1\}$, describing the operating and in-operating time periods. The switching events occur continuously and independently of one another. For each processor the state process $r^e(\cdot)$ is a single Markov process, where almost every sample path is a right-continuous step function with a finite number of simple jumps (switchings) on $[t_0, T]$.

The state of the whole network is described by the multivariate state process

$$\mathbf{r}(t) = (r^1(t), \ldots, r^M(t)) \in \{0, 1\}^M, \quad t \geq t_0\,, \tag{2.4}$$

which consist of the state processes of all processors. If one of the processors breaks down or gets repaired the multivariate state process is switching from one state into one other. In Section 3.1 we show that $\mathbf{r}(\cdot)$ is a Markov process itself, which also has a finite number of jumps on $[t_0, T]$.

2.3. The stochastic network model with random breakdowns. The random breakdowns of the processors are included in the network as a model for the capacities of the processors. We model the capacity of a processor e by

$$\mu^e(t) := \mu^e \cdot r^e(t) \tag{2.5}$$

where μ^e is the maximal processing rate and r^e is the two state process describing its operating and in-operating times. This model of the capacity is now used in the flux function (2.1b) and in the function of the outgoing flux (2.1e).

Let us discuss the impact of the random breakdowns in the network formulation. If a breakdown occurs during the operation of a processor, the capacity of the processor switches to zero and is resumed from the point of interruption after the repair. Incoming goods continue to join the queue during in-operating periods.

Moreover, due to the stochastic processes r^e the quantities in the network formulation are changing. Its solution is a stochastic process depending on the time t, the space x and the random sample ω. Thus, the solution components ρ^e and q^e of each processor are stochastic processes, too. The stochastic process of the densities reads

$$\rho^e : [a^e, b^e] \times \mathbb{R}_0^+ \times \Omega \to \mathbb{R}_0^+ : (x, t, \omega) \mapsto \rho^e(x, t, \omega)$$

and the loads of the queues are given by

$$q^e : \mathbb{R}_0^+ \times \Omega \to \mathbb{R}_0^+ : (t, \omega) \mapsto q^e(t, \omega)\,.$$

In this context the densities of the processors and the loads of the queues at fixed time t and space x, respectively, are random variables $\rho^e(x, t, \cdot) = \rho^e(x, t)$ and $q^e(t, \cdot) = q^e(t)$ whose argument ω are usually dropped. For a fixed sample or realisation ω the solution components $\mathbf{p}(\cdot, \cdot, \omega)$, $\mathbf{q}(\cdot, \omega)$ as well as the multivariate state process $\mathbf{r}(\cdot, \omega)$ is called a realisation or a scenario of the solution.

Including the state-processes r^e in the network model, the stochastic network formulation reads ($t \in [t_0, T]$ and $\omega \in \Omega$)

$$\partial_t \rho^e(x, t, \omega) + \partial_x f^e(\rho^e(x, t, \omega)) = 0\,, \quad x \in [a^e, b^e] \tag{2.6a}$$

$$f^e(\rho^e) = \min(v^e \cdot \rho^e(x, t, \omega), \mu^e \cdot r^e(t, \omega)) \tag{2.6b}$$

$$\partial_t q^e(t, \omega) = g^e_{\text{in}}(t) - g^e_{\text{out}}(t)\,, \tag{2.6c}$$

$$g^e_{\text{in}}(t) = \begin{cases} A^{s(e),e}(t) \sum_{\bar{e} \in \delta^-_{s(e)}} f^{\bar{e}}(\rho^{\bar{e}}(b^{\bar{e}}, t, \omega)) & \text{if } s(e) \notin \mathcal{V}_{\text{in}}, \\ G^{s(e)}_{\text{in}}(t) & \text{if } s(e) \in \mathcal{V}_{\text{in}}. \end{cases} \tag{2.6d}$$

$$g^e_{\text{out}}(t) = \begin{cases} \min\{g^e_{\text{in}}(t)\,,\, \mu^e \cdot r^e(t, \omega)\} & \text{if } q^e(t, \omega) = 0, \\ \mu^e \cdot r^e(t, \omega) & \text{if } q^e(t, \omega) > 0. \end{cases} \tag{2.6e}$$

$$\rho^e(x, t_0, \omega) = \rho^e_0(x), \;\; q^e(0, \omega) = q^e_{t_0} \text{(PDE and ODE initial conditions (i.c.))} \tag{2.6f}$$

$$\rho^e(a^e, t, \omega) = g^e_{\text{out}}(q^e(t, \omega))/v^e \text{ (PDE boundary conditions (b.c.))} \tag{2.6g}$$

$$r^e(0, \omega) = r^e_0 \;\; \text{(initial state of the processors)} \tag{2.6h}$$

Here, we have replaced the function μ^e of each processor by the stochastic process $\mu \cdot r^e$ and we have rewritten the components ρ^e and q^e of the solution into their stochastic processes. This is denoted by the additional argument ω. Though the network system (2.6) formally differs only by the additional status process r^e in (2.5) from the deterministic network model (2.1), its solution is now a stochastic process.

Definition 2.4 (The time-continuous network model with random breakdowns). *Let $[t_0, T]$ be a fixed real interval. Then, the stochastic process $X = (\mathbf{p}, \mathbf{q}, \mathbf{r})$ is called a solution of the* time-continuous network model with random breakdowns *on $[t_0, T]$ if equations* (2.6a) – (2.6h) *hold with probability 1. The solution consists of the densities of processors, the loads of queues and the state process, where $\mathbf{p}(t_0) = \mathbf{p}_0$, $\mathbf{q}(t_0) = \mathbf{q}_0$, and $\mathbf{r}(t_0) = \mathbf{r}_0$ denote the initial values at time t_0.*

3. The coupled PDE-ODE system with Markovian switching. Due to the influence of the state process $\mathbf{r}(t)$, $t \geq t_0$, the randomness in the solution X of the network with random breakdowns appears as point events when one of the processors is breaking down or gets repaired. There is no additional component of uncertainty and the network behaves completely deterministic between these points. Hence, its solution belongs to the class of piecewise deterministic processes (PDPs).

PDPs are originally introduced by Davis [8, 9] for a class of hybrid Markov processes. A PDP consists of a mixture of deterministic time evolution and random jumps. The jump process itself may not necessarily be Markovian, because the jump times of a PDP can depend on the deterministic time evolution. In the setting of PDPs the interaction between both parts define a Markov process.

In our network formulation the state processes r^e represent the jump processes. These processes are only time dependent and do not depend on other network quantities and solution components, respectively. In the following we will see that the multivariate state process $\mathbf{r}$ is a Markov process by itself and our network formulation belongs to a special subclass of PDPs, namely X is the solution of deterministic equations with Markovian switching. More precisely, the time evolution is formulated by a coupled deterministic PDE-ODE system and the multivariate state process $\mathbf{r}$ gives the Markovian switching. We will apply a stochastic simulation algorithm to compute sample realisations of the solution process.

Systems that combine a part of the state that takes values continuously (coupled PDE-ODE system) and another part of the state that takes discrete values (state process) are often called hybrid systems. When the first part is an ordinary differential equation and the latter part is a Markovian process these systems are called ODEs with Markovian switching. Equations of these type have been considered in some practical applications, like electrical power systems [33], solar receiver [32] and economics [23]. A general introduction can be found in [28], and more recently in [27], where the authors consider stochastic differential equations (SDEs) with Markovian switching.

3.1. The multivariate jump process. In Section 2.2 we have considered the two-state processes r^e, which describe if the single processor e is operating or in-operating at time t. We have assumed that these operating and in-operating periods are exponentially distributed and we have modelled these periods by exponential distributed random variables. Hence, each single state process r^e is memoryless and Markovian.

Now, we combine the properties of the single processes r^e and consider the multivariate jump process defined by $\mathbf{r}(t) = (r^1(t), \ldots, r^M(t))$, $t \geq t_0$. We assume that this process just switched at the time t^* from one state into another. We are now interested in the length $\Delta\tau$ up to the next switching. This length can be computed as the minimum over all random variables $\Delta\tau^e$ of processors e until the time point of the next state switch

$$\Delta\tau = \min\{\Delta\tau^1, \ldots, \Delta\tau^M\}\,. \tag{3.1}$$

Here, each time period $\Delta\tau^e$ are independent exponentially distributed random variables with rate parameters $\lambda^e(r^e(t))$ depending on the current state of the processor (cf. Section 2.2). We have the following result.

LEMMA 3.1. *Assume that a switch in the network has occurred at time t^* and that the state of the network is given by $\mathbf{r}(t^*)$. Then the probability that the next*

switch will be the eth processor and will occur after the time period s is given by

$$\mathbb{P}(s, e|\,\mathbf{r}(t^*)) = \frac{\lambda^e(r^e(t^*))}{\lambda^{sum}(\mathbf{r}(t^*))} \cdot \lambda^{sum}(\mathbf{r}(t^*))\ \exp(-\lambda^{sum}(\mathbf{r}(t^*))s), \tag{3.2a}$$

where

$$\lambda^{sum}(\mathbf{r}(t^*)) = \sum_e \lambda^e(r^e(t^*))\,. \tag{3.2b}$$

As for the single switch processes r^e, this probability depends on the current state $\mathbf{r}(t^*)$ of the system, but not on its history.

Proof. Let $\Delta\tau^1, \ldots, \Delta\tau^M$ be independently exponentially distributed random variables with rate parameters λ^e. It is well-known that the minimum

$$\Delta\tau = \min\{\Delta\tau^1, \ldots, \Delta\tau^M\}$$

is an exponentially distributed random variable, too, where the rate parameter is given by the sum λ^{sum} of the individual rate parameters. Hence, the probability that the next switch of the network will occur after a time s reads

$$\mathbb{P}(\min\{\Delta\tau^1, \ldots, \Delta\tau^M\} > s) = \exp\left(-\lambda^{\text{sum}} s\right).$$

and the index of the variable which achieves the minimum is distributed according to the law

$$\mathbb{P}(\Delta\tau^e = \min\{\Delta\tau^1, \ldots, \Delta\tau^M\}) = \frac{\lambda^e}{\lambda^{\text{sum}}}.$$

Together this gives the lemma. ∎

REMARK 3.2. *Formula* (3.2a) *can be interpreted as the joint density function of two independent random variables: (i) length of the time period until next switch, and (ii) index of the next switching processor. This allows us to simulate independently the length of the next time period and the switching processor. The length of the time interval until this next switching is an exponentially distributed random variable with rate parameter* $\lambda^{sum}(\mathbf{r})$*, whereas the next switching index corresponds to a discrete random variable: Choose one of the processors such that the chance of picking the eth processor is proportional to* λ^e/λ^{sum}*. Both can be computed via a uniform* $(0, 1)$ *sample (see Algorithm 4.1 for algorithmic details).*

3.2. A stochastic simulation algorithm. Next we consider the solution process of the stochastic network model. A switching between two states of $\mathbf{r}$ occurs only in one of the processors and the randomness appears only at the switching points.

PROPOSITION 3.3. *Almost every sample path of* $\mathbf{r}(\cdot)$ *is a multi-dimensional step-function with a finite number of simple jumps on* $[t_0, T]$*.*

So, there is a sequence $\mathcal{T}^* = \{t_i^*\}_{i\geq 0}$ of stopping times such that for almost every random sample $\omega \in \Omega$ there is a finite number $S = S(\omega)$ for $t_0 = t_0^* < t_1^* < \cdots < t_S^* = T$, which counts the number of switchings, and $\mathbf{r}(\cdot)$ is a random constant on every interval $[t_i^*, t_{i+1}^*[$, namely $\mathbf{r}(t) = \mathbf{r}(t_i^*)$ on $t_i^* \leq t < t_{i+1}^*$ for all $i \geq 0$. We refer to t_i^* as a switching point and to $\mathcal{T}^*$ as the sequence of switchings.

The network behaves deterministically in the time interval $[t^*, t^* + \Delta\tau]$ and its solution X is piecewise deterministic. The time interval is still random because the

switching points t_i^* are random variables, but the state process $\mathbf{r}(\cdot)$ is a random constant on $[t^*, t^* + \Delta\tau]$.

Whenever $\mathbf{r}$ is a random constant, there are no random effects in the network and the stochastic network formulation reduces to a deterministic one. We can use a deterministic approach to compute a solution of the coupled PDE-ODE system between the switching point. We can compute single realisations or scenarios of the piecewise deterministic solution process based on the multivariate jump process and the coupled PDE-ODE system by the following stochastic simulation algorithm.

The main idea is to sample iteratively a solution or realisations of the stochastic solution process by (i) sample the switching times from the mean up and mean down times τ^e_{down} and τ^e_{up} of the processors, and (ii) use deterministic numerics between these time points to solve the coupled PDE-ODE system. The initial values of one step are given by the final values of the network of the former step.

DEFINITION 3.4 (A modified stochastic simulation algorithm). *Let $[t_0, T]$ be a fixed real interval. Then, a realisation $\overline{X}$ of the solution process X of the* time network model with random breakdowns *on $[t_0, T]$ can be sampled by the following stochastic simulation algorithm for the coupled PDE-ODE system with Markovian switching* (2.6)*: Iterate*

1. *Sample next switching point $\bar{t}^*_{i+1}$ and the next state of the multivariate state process $\bar{\mathbf{r}}(\bar{t}^*_{i+1})$*
2. *Compute the solution over $t \in [\bar{t}^*_i, \bar{t}^*_{i+1}]$ where $\bar{\mathbf{r}}(t) = \bar{\mathbf{r}}(\bar{t}^*_i)$ using deterministic numerics.*

As output we get a sampled realisation or sampled scenario of the solution process X.

Note that the stochastic simulation algorithm samples the switching points $\bar{t}^*_i$, where one of the M processors changes its current state from operating to in-operating (1 to 0) or vice versa. This gives a set of strictly increasing time points denoted by

$$\overline{\mathcal{T}}^* = \{\bar{t}^*_0, \bar{t}^*_1, \dots\}\,.$$

REMARK 3.5. *Our approach is almost similar to the stochastic simulation algorithm (SAA) to compute a single scenario of the Chemical Master Equation (see [12]). The SSA was invented by Dan Gillespie (see the classic early reference [13]). One can show that the SSA reproduces the statistics of the network exactly, however it comes with a high computational cost especially in the case that the network consists of processors with very fast switchings between their operating and in-operating states.*

4. Numerics for coupled PDE-ODE systems with Markovian switching. In this section we introduce a numerical scheme to compute sample realisations of the stochastic supply chain model introduced in Section 2. As we have shown in Section 3, this requires the subsequent iteration of two algorithms: the simulation of switching points of the multivariate state-process $\mathbf{r}$ and the deterministic computation of the coupled PDE-ODE system between these switchings. The computation of single realisations will be used for Monte-Carlo simulations in the next section.

Suppose we are at time $t = \bar{t}^*_i$, and the network has either just seen a breakdown or repair of one of its processors. Following Section 3, we obtain the next switching time and the next processor to default or to be repaired with the following algorithm:

ALGORITHM 4.1 (Sample next switching point $\bar{t}^*_{i+1}$ and processor).

1. collect exponential weights of all processors e from (2.3): $\lambda^e := \lambda(\bar{r}^e(\bar{t}^*_i))$; (4.1a)

2. calculate total exponential weight, cf. (3.2b): $\lambda^{sum} := \sum_e \lambda^e$; (4.1b)

3. sample time period and set next switching: $\overline{\Delta\tau} \sim Exp(\lambda^{sum})\,,\ \bar{t}^*_{i+1} = \bar{t}^*_i + \overline{\Delta\tau}\,;$ (4.1c)

4. define a partition of the unit interval: $h_e := \left[\frac{\sum_{i=1}^{e-1}\lambda_i}{\lambda_{sum}}, \frac{\sum_{i=1}^{e}\lambda_i}{\lambda_{sum}}\right),\ \bigcup_{e\in\mathcal{A}} h_e = [0,1);$ (4.1d)

5. sample from uniform distribution on $[0,1)$: $\bar{u} \sim \mathcal{U}([0,1))\,;$ (4.1e)

6. detect switching processor: $\bar{e} \in \{1,\dots,M\}$ such that $\bar{u} \in h_{\bar{e}}$; (4.1f)

7. compute new state: copy $\bar{\mathbf{r}}(\bar{t}^*_{i+1}) := \bar{\mathbf{r}}(\bar{t}^*_i)$ and change $\bar{r}^{\bar{e}}(\bar{t}^*_{i+1}) := \begin{cases} 1, & \bar{r}^{\bar{e}}(\bar{t}^*_i) = 0 \\ 0, & \bar{r}^{\bar{e}}(\bar{t}^*_i) = 1; \end{cases}$ (4.1g)

Let us now consider the numerical solution the PDE-ODE system (2.6) between two sampled switching times. We start by defining the spatial discretisation of each processors e to consist of $K^e + 1$ equidistant points: $x^e_k = a^e + k \cdot \Delta x^e\,,\quad k = 0,\dots,K^e$, where $\Delta x^e := l_e/K^e$ and $x^e_{K^e} = b^e$. Recall, that propagation in spatial length associated with a processor can be interpreted as e.g. increase of completion within one production step in a supply chain. A fine discretisation in space therefore corresponds to a detailed tracing of the level of completion of items within processors. For the model to be properly defined, we need at least two discretisation points for every processor ($K^e \geq 1\,\forall e$). Note, that this marks a difference between the model introduced here and the long-chain model by Degond and Ringhofer in [10], where each processor is identified with one discretisation point on the line.

Any proper time discretisation of $[t_0, T]$ has to include the sampled switching times $\bar{t}^*_i \in \overline{\mathcal{T}^*}$, since the network configuration changes randomly at any $t^*_i \in \mathcal{T}^*$, whereas the system evolves deterministically in between any two switching times, as it has been argued in the previous section. A Monte-Carlo study of the stochastic evolution of the complete network (see Section 5) requires common points in time discretisation in each simulation. Therefore, a deterministic grid

$$\mathcal{T} := (t_i),\ t_i := t_0 + i \cdot \Delta t \quad \text{with} \quad \Delta t := \min_e \left\{ \frac{\Delta x^e}{v^e} \right\}$$

given by the CFL-condition is defined common to all simulations, and the time discretisation for a particular simulation of (2.6) between $\bar{t}^*_i$ and $\bar{t}^*_{i+1}$ is given by $(t_j) = \{\bar{t}^*_i, \bar{t}^*_{i+1}\} \cup \mathcal{T} | [\bar{t}^*_i, \bar{t}^*_{i+1}]$.

We solve the ODE (2.6c) with the explicit Euler scheme. Equations (2.6a), (2.6b) reduce to a linear advection equation at positive speed between any two switching times. Hence, (2.6a) is solved with the upwind scheme, which gives rise to the CFL-condition. The further equations (2.6d)–(2.6f) and (2.6h) are easily discretised (see also [15]), where a numerical cut-off function prevents negative queue values in the discretisation of (2.6e). Emphasis has to be placed on coupling condition (2.6g): We

stress, that *product transfer from the end of one processor through the vertex into the queue and onto the successive processor is instantaneous in the continuous model.* Diversion, control, and passing through the queue takes up no time. To preserve this key feature in the numerical scheme, we may not introduce an artificial time delay in the transfer from incoming into outgoing processors. This can be ensured with a careful sequencing of computations: We first compute the spatially last step of the upwind schemes in any processors to compute the outflow of product and the total inflow into vertices. With the evaluation of the control functions, inflows into the queue and the resulting outflows are obtained. The decisive step is, that the outflow from the queue is stored as the boundary condition of the density at the *current* time point t_j instead of t_{j+1}, which would cause a numerical delay:

$$\bar{\rho}^e(x_0, t_j) = g^e_{out}(t_j)/v^e$$

After that, the remaining grid points of all processors are iterated and the boundary condition $\bar{\rho}^e(a^e, t_j)$ is transported into the processors (if $\bar{r}^e = 1$). The complete numerical scheme looks as follows:

ALGORITHM 4.2 (Piecewise deterministic approximation of the coupled PDE-ODE system). *Let $\bar{t}^*_i$ and $\bar{t}^*_{i+1}$ be two sampled switching times, such that $\bar{\mathbf{r}}(t)$ is constant on $[\bar{t}^*_i, \bar{t}^*_{i+1})$, and no switching occurs in between. Let the approximation $(\bar{\mathbf{p}}(\bar{t}^*_i), \bar{\mathbf{q}}(\bar{t}^*_i))$ up to the time $\bar{t}^*_i$ be given. Moreover, let the number of spatial discretisation intervals for the network processors be given as K^e, $e \in \mathcal{A}$ and let $\mathcal{T}$ be a global time discretisation of $[t_0, T]$. Then, the numerical approximation $(\bar{\mathbf{p}}, \bar{\mathbf{q}})$ of the solution process $(\mathbf{p}, \mathbf{q})$ of system (2.6) on $[\bar{t}^*_i, \bar{t}^*_{i+1}]$ is given by the following algorithm:*

Initialisation:

$(x_k)_{k=0:K^e} := \left\{a^e, a^e + 1 \cdot l^e/K^e, \ldots, b^e\right\}$ (discretisation of the processors)

$\mathcal{T} = \{t_0, \ldots, t_J\}$ (global time discretisation of $[t_0, T]$)

$(t_j)_j := \{\bar{t}^*_i, \bar{t}^*_{i+1}\} \cup \mathcal{T} | [\bar{t}^*_i, \bar{t}^*_{i+1}] = \left\{\bar{t}^*_i, t_c, t_{c+1}, \ldots, t_{c+C-1}, \bar{t}^*_{i+1}\right\}$ (local time discret.)

Iteration: For $j = 0 : C-1$

- For $e = 1 : M$
  - $\text{outflow}^e = v^e \cdot \bar{r}^e(t_j) \cdot \bar{\rho}^e(x_{K^e}, t_j)$; (single upwind step at processor end)
- For $v = 1 : N$
  - $\text{vertexflow}^v = \begin{cases} \sum_{e \in \delta^-_v} \text{outflow}^e, & v \in \mathcal{V} \backslash \mathcal{V}_{\text{in}} \\ G^v_{\text{in}}(t_j), & v \in \mathcal{V}_{\text{in}} \end{cases}$ (collect flow into vertex)
- For $e = 1 : M$
  - $g^e_{\text{in}} = \alpha^e_{s(e)} \cdot \text{vertexflow}^{s(e)}$ (divert flow into queues)
  - $g^e_{\text{out}} = \begin{cases} \mu^e \cdot \bar{r}^e(t_j), & \bar{q}^e(t_j) > 0 \\ \min(\mu^e \cdot \bar{r}^e(t_j), g^e_{\text{in}}), & \bar{q}^e(t_j) = 0 \end{cases}$ (queue outflow)
  - $\bar{q}^e(t_{j+1}) = \bar{q}^e(t_j) + \Delta t_j \cdot (g^e_{\text{in}} - g^e_{\text{out}})$ (Euler step in queue)
  - $\bar{\rho}^e(x_0, t_j) = g^e_{\text{out}}/v^e$ (set queue outflow as boundary condition at t_j)
  - For $k = 0 : K^e - 1$ (complete upwind scheme)
    - $\bar{\rho}^e(x_{k+1}, t_{j+1}) = \bar{\rho}^e(x_{k+1}, t_j) - \frac{(v^e \cdot \bar{r}^e(t_j))\Delta t_j}{\Delta^e x} (\bar{\rho}^e(x_{k+1}, t_j) - \bar{\rho}^e(x_k, t_j))$

where $\Delta t_j = t_{j+1} - t_j$ and $\Delta^e x = l^e/K^e$.

A sample simulation of the stochastic model (2.6) over $[t_0, T]$ is obtained through the subsequent iteration of Algorithms 4.1 and 4.2.

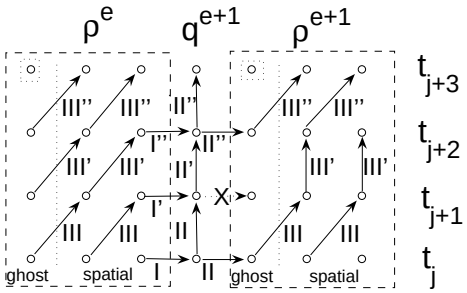

Fig. 4.1: Illustration of Algorithm 4.2 for the case of two successive processors e, $e+1$ and two switching times t_{j+1}, t_{j+2} marking the default and repair of processor $e+1$: The first discretisation point of a processors serves as a ghost cell whose value does not represent mass in the processor, as indicates by dotted lines. Apostrophes distinguish three time iterations ($t_j \rightarrow t_{j+3}$). Each iteration subsequently divides as follows: I) Outflux from the right boundary of e into queue $e+1$ is computed (g_{in}). II) Flux out of the queue into processor $e+1$ is obtained (g_{out}) and queue value is updated accordingly (explicit Euler). The flux into the processors updates the ghost cell at *current* time (no time delay). III) Spatial discretisation points of processors are updated (Upwind scheme). There is no transport in case of default. We can see, that ghost cells are updated only in the successive iteration, as indicated by squared dotted boxes at t_{j+3}.

REMARK 4.3 (Conservation of mass). *The numerical coupling condition* (4) *included in Algorithm 4.2 induces a particular way of measuring the total mass of products stored in the network processors. The first discretisation point $x_0^e = a^e$ serves as a ghost cell, where mass, which is to be transported if the processor is running, is temporarily stored. Contrary, the last discretisation point $x_{K^e}^e = b^e$ is a point containing mass and determines the flux into successive vertices. Therefore, the total numerical mass in a processor e is given by*

$$TM^e(t_j) := \sum_{k=1}^{K^e} \bar{\rho}^e(x_k, t_j)\Delta x^e$$

The total mass in the network is obtained by adding all queue loads to the above quantities. With this definition, it can be seen that the only source for gain or loss in numerical total mass is the continuous conversion between flux and mass in the scheme, which is however in the scale of machine accuracy. Apart from that, the total numerical mass in Algorithm 4.2 is conserved.

REMARK 4.4. *We would like to point out, that numerical simulations of the stochastic network model presented in this work are completely parallelisable. Thanks to the use of at least two discretisation points for all processors and the CFL-condition, all processors can be iterated simultaneously and it is not necessary to mirror the flow of products in the sequential order of processor iterations, which is an ease for actual implementation and enable us to consider networks, which include loops.*

REMARK 4.5. *The subsequent iteration of Algorithms 4.1 and 4.2 is possible due to the independence of the break down and repair process* $\mathbf{r}$ *from the system solution* $(\mathbf{p}, \mathbf{q})$. *Dropping this assumption leads to interesting problems also for the numerical simulation. For example, the modelling and simulation of load-dependent failure rates,*

where the probability of a processor to break down is related to its workload, can be a subject of further research.

5. Simulation experiments. We are mainly interested in average values of network quantities obtained by Monte-Carlo simulations, i.e., from a big number of simulations of independent single scenarios. We use these quantities to analyse the behaviour of three different networks with random breakdowns and to compare their simulation results in the deterministic and stochastic setting.

The advantage of a Monte Carlo approach stems from the fact that it can be used in very complex networks and is straightforward to implement since it simply requires to simulate independently sample scenarios of the network model (2.6). These scenario results are used to compute various estimates of average values of network quantities. In the following examples, we consider the loads of the queues $\mathbf{q}(t)$, the density $\mathbf{p}(x,t)$ of a processor at fixed time t, and the outgoing flow of the network:

$$\rho^e_{AV}(x_k,t_i) = \frac{1}{L}\sum_{\ell=1}^{L} \bar{\rho}^e_\ell(x_k,t_i), \quad t_i \in \mathcal{T},\ k=0,\dots,K^e \tag{5.1}$$

$$q^e_{AV}(t_i) = \frac{1}{L}\sum_{\ell=1}^{L} \bar{q}^e_\ell(t_i), \qquad t_i \in \mathcal{T} \tag{5.2}$$

$$G^{\text{net}}_{\text{outAV}}(t_i) = \frac{1}{L}\sum_{\ell=1}^{L} \bar{G}^{\text{net}}_{\text{out}_\ell}(t_i), \quad t_i \in \mathcal{T} \tag{5.3}$$

where $\mathcal{T}$ is the global time grid, L is the number of sample scenarios, K^e is the number of spacial discretisation points of processor e, $\bar{\mathbf{q}}_\ell = (\bar{q}^1_\ell,\dots,\bar{q}^M_\ell)$ and $\bar{\mathbf{p}}_\ell = (\bar{\rho}^1_\ell,\dots,\bar{\rho}^M_\ell)$ are the sample solution of scenario ℓ, and $G^{\text{net}}_{\text{out}_\ell}$ is the corresponding network outflow as in (2.6e).

To start the simulation, we assume that each processor is ready to work at the beginning of the simulation, i.e., let $\bar{r}^e(t_0) = 0$ for all processors e. Moreover, all queues appear empty $\bar{q}^e(t_0) = 0$ and no goods are stored in the processors, i.e. $\bar{\rho}^e(x,t_0) = 0$ for all x.

5.1. A chain of processors. In our first simulation experiment, we consider a simple line of 5 subsequent processors ($M = 5$), cf. similar results in [15]. Here, each processor e has length $l^e = 0.2$ and is assumed to work with the same processing rate $\mu^e = 2$ and processing velocity $v^e = 1$. The raw material is joining the network in front of processor 1 and transported through the chain. This simple network does not have any splitting or merging of processor line such that all distribution rates are equally one, i.e., we have $A^{s(e),e}(\cdot,t) = \alpha^{s(e)}_e = 1$ for all processors e. Assuming a constant network inflow $G^{s(1)}_{\text{in}}(t) = 1.8$ for $t \geq t_0$ and a time horizon $T = 4$ the deterministic simulation results are given to the left in Figure 5.1. Here, we have not considered random breakdowns of processors. We have connected the densities of all processors into a single graph. The density of processor 1 is given within $x \in [0, 0.2)$, the density of processor 2 is given by $x \in [0.2, 0.4)$ and so on. Hence, this graph gives the time evolution of the density of the whole chain of processors. We can observe that the goods are transported through the chain with constant velocity and no goods are joining the queues. Each processor is working with a processing rate of 1.8 which is lower than their maximal processing rates. After a delay of 1 time unit the first good is fully completed and the network outflow is constant 1.8 from these time point on.

Next, we assume random breakdowns of the processors. We introduce different mean times between failures (MTBF) and mean repair times (MRT) to the processors. These parameters are given in Table 5.1, where processor 2 remains to work constantly and is not assumed to break down randomly.

Processor e	1	2	3	4	5
MTBF τ^e_{on}	0.95	–	0.85	1.90	0.95
MRT τ^e_{off}	0.05	–	0.15	0.10	0.05

Table 5.1: Mean times between failures (MTBF) and mean repair times (MRT) of the processors.

The pictures in the middle and to the right in Figure 5.1 show the simulation result of the stochastic model. Here we have plotted the mean values of the density and the loads over 1000 computed scenarios. The right graph illustrates that only the queue of processor 2 remains empty, because we have assumed that this processor works constantly and all goods can join processor 2 directly. All other queues are busy. Especially queue 3 in front of processor 3 increases, because this processor has only a mean availability of 85%, whereas all the other processors are working with a mean availability of 95% (expect of processor 2, which works constantly). Due to the goods in queue 3, processor 3 is working with its maximum capacity of 2 throughout its working periods and the density of this processor is higher. Moreover,

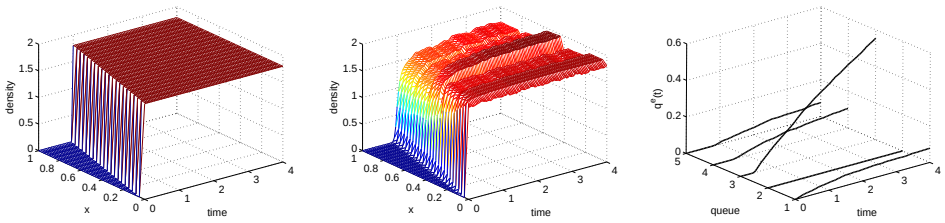

Fig. 5.1: Deterministic simulation results of densities of processors in a linear chain consisting of 5 processors (left). As expected, all queues remain empty due to the fact all possible maximal processing rates are larger than the network inflow $G^{s(1)}_{\text{in}}(t)$, i.e. $\mu^e > G^{s(1)}_{\text{in}}(t)$ for all e. This behaves completely different if random breakdowns of processors are involved. The stochastic simulation results are illustrated by the mean densities of processors (middle) and the loads of queues (right) vs. time over 1000 scenarios.

it can be observed, that the average total network outflow $G^{\text{net}}_{\text{outAV}}(T)$ decreases. In the deterministic setting the chain has produced 5.4 units of goods until 4 time units (since $G^{\text{net}}_{\text{out}}(T) = (T - T_{through}) \cdot G^{s(1)}_{\text{in}}(t)$ where $T_{through}$ denotes the total deterministic throughput time) but when we consider random breakdowns of processors in the network this production rate reduces to $G^{\text{net}}_{\text{outAV}}(T) = 4.57$ goods units - determined by 1000 sample scenarios. This indicates a reduction of approximately 15 per cent.

REMARK 5.1. *We report on numerical similarities and differences to the stochastic supply chain model of Degond and Ringhofer presented in [10]. Therein, the authors propose a PDE model for the density of goods of the form*

$$\partial_t \tilde{\rho}(x,t) + \partial_x \Big(\min\{\mu(x,t), v(x)\tilde{\rho}(x,t)\} \Big) = 0, \quad x \in [0,1] \tag{5.4}$$

where a partition of the stage interval $[0,1]$ *corresponds to* M *processors and* $v(x)$ *is again the processing velocity and* $\mu(x)$ *the maximal processing rate that should be random in time. In each Monte-Carlo simulation iteration random signals* $\mu(x,t)$ *are sampled* for each processor *in a preprocessing step according to the following procedure:*

1. *Processor* e *hast just turned on at time* $\bar{t}^*$
2. *Pick* $\Delta\bar{\tau}_{on}$ *and* $\Delta\bar{\tau}_{off}$ *from the exponential distribution* $\mathbb{P}(\Delta\bar{\tau}_{on}=t)=\frac{1}{\tau_{on}}\exp(\frac{-t}{\tau_{on}})dt$ *and* $\mathbb{P}(\Delta\bar{\tau}_{on}=t)=\frac{1}{\tau_{on}}\exp(\frac{-t}{\tau_{on}})dt$*, respectively.*
3. *Choose* $\mu(t)=1$ *for* $\bar{t}^*<t<\bar{t}^*+\Delta\bar{\tau}_{on}$ *and* $\mu(t)=0$ *for* $\bar{t}^*+\Delta\bar{\tau}_{on}<t<\bar{t}^*+\Delta\bar{\tau}_{on}+\Delta\bar{\tau}_{off}$*.*

In summary, the model defined in this manner is restricted to linear chains and measures only the total average density of processors without showing individual queueloads. The latter constitutes the main difference to the network model (2.1)*. In addition, we are scanning the switching points of each single processor switching exactly which leads to adaptivity in time and more accurate representation of the switching behaviour in the approximate solution of* (2.6)*. However, the model* (5.4) *uses only a fixed time grid. Consequently, to compare the simulation results of the aforementioned model with our approach, we need to reformulate our averaged quantities* ρ^e_{AV} *and* q^e_{AV} *in terms of an averaged density* $\tilde{\rho}^e_{AV}(t_j)$ *allocating both measures onto a single processor.*

$$\tilde{\rho}^e_{AV}(t_j)=\frac{1}{l^e}\Big(\Delta^e x\sum_{k=1}^{K^e}\rho^e_{AV}(x_k,t_j)+q^e_{AV}(t_j)\Big),\quad \forall e. \tag{5.5}$$

The comparable results of the two models are illustrated in Figure 5.2. Obviously, the averaged density $\tilde{\rho}^e_{AV}$ *emanating from* (2.6) *is able to capture the same dynamical behaviour as the averaged density of* (5.4)*. But due to different space-time discretisations and the Monte Carlo average, minor deviations in the values are natural.*

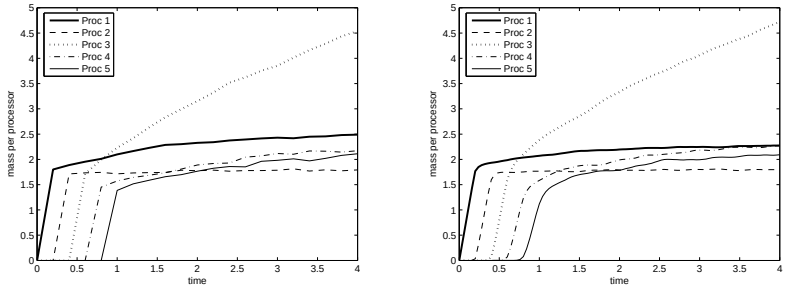

Fig. 5.2: A linear chain with 5 processors and parameter setting as defined above: We perform 1000 Monte Carlo runs of the stochastic model (5.4) and compute the average density for each processor over time (left). The corresponding measure (5.5) for the network model (2.6) can be found to the right.

5.2. A diamond network. We consider a sample production network with two controls given in [16]. The graph of that network is given in Fig. 5.3. It consists of 7 processors with different processing rates μ^e, but constant length $l^e=1$ and processing velocity $v^e=1$. The network has two distribution points with distribution

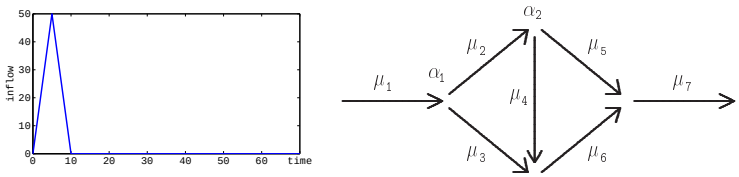

Fig. 5.3: Diamond network: 7 processors and two controls (right) and inflow profile (left).

rates α_1 and α_2. As in [16], we choose the maximal processing rates $\mu_1 = 40$, $\mu_2 = 30$, $\mu_3 = 20$, $\mu_4 = 20$, $\mu_5 = 5$, $\mu_6 = 10$, $\mu_7 = 10$ and a piecewise linear inflow with a total amount of 250 parts, see Figure 5.3. The controls in front of processor 2 and 3 are given by α_1 and $(1 - \alpha_1)$, respectively, whereas the controls in front of processor 4 and 5 are given by α_2 and $(1 - \alpha_2)$. Our simulation time period is $[t_0, T] = [0, 70]$.

In the following, we show both the deterministic as well as the averaged stochastic simulation result. Default and repairs are set to $\tau_{\text{on}}^{2,4} = \tau_{\text{off}}^{2,4} = 0.50$ for processors 2 and 4. All other processors are working with mean times $\tau_{\text{on}}^{e} = 0.95$ and $\tau_{\text{off}}^{e} = 0.05$, $e = 1, 3, 5, 6, 7$. This equates to an average availability of 50% and 95%, respectively.

Figure 5.4 (left) shows the total processing time necessary to produce all goods completely for varying control parameters α_1, α_2. In the deterministic setting the pro-

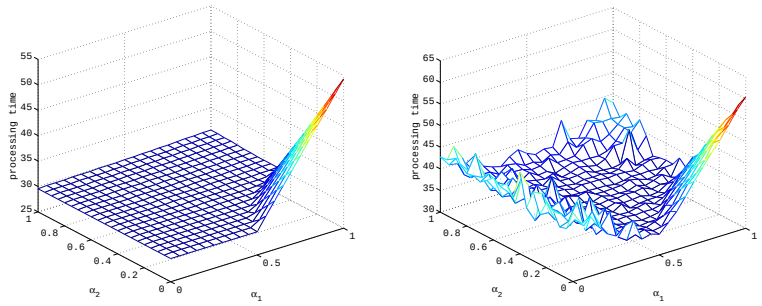

Fig. 5.4: Total production time of 250 units for deterministic (left) and averaged stochastic (right) simulation using different combinations of distribution rates α_1 and α_2..

duction is completed after 29.56 time units, but the inclusion of random breakdowns leads to a best-possible production time of 34.11 time units on average. Thus, we see a considerable effect of default and repair times to the production time profiles. The reason is that in-operating processors are stopping the flow through the network which causes a delay in the production. Let us now look at the total outflow of the network at time $t = T/3 = 23.22$ units. At this point in time, the production is not completed. We show deterministic and averaged stochastic results in Figure 5.5 each with two illustrations: A surface as well as a contour plot.

The best possible network outflow in the deterministic setting is 184 units given by parameter sets $(\alpha_1, \alpha_2) = (0, [0, 1])$ and $(0, [0, 0.5])$, whereas, on average, the best possible outflow in the stochastic setting is 170.23 at $(\alpha_1, \alpha_2) = (0.6, 0.5)$. However, the contour plot shows a whole level set of suitable parameters with averaged outflow

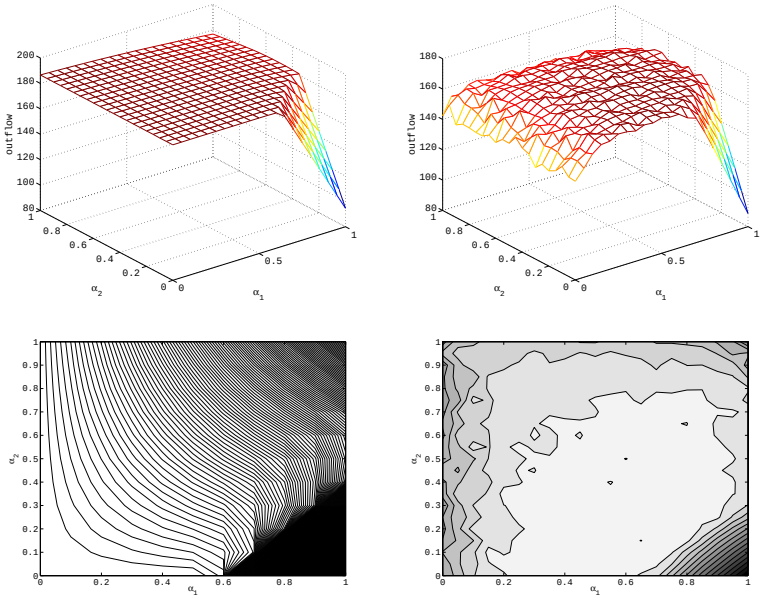

Fig. 5.5: Total product outflow at $t = T/3$ for deterministic (left upper, left lower) and averaged stochastic (right upper, right lower) simulation using different combinations of distribution rates α_1 and α_2.

close to 170 (light gray area). Advisable parameters hence differ for the deterministic and the stochastic setting. This is a motivation to consider optimisation of distribution rates of a production network with random breakdowns in the near future.

5.3. Cascade network. Finally, we consider a network introduced by Battiston et. al in [4]. This network was analysed to study bankruptcy and production failures and is depicted schematically in Figure 5.6.

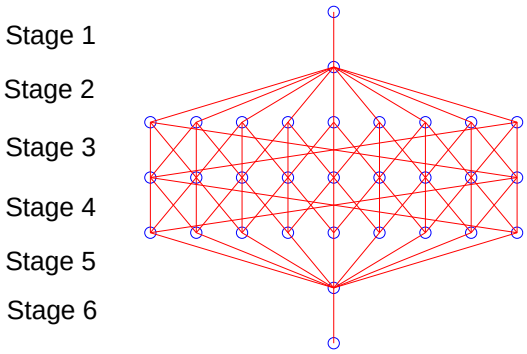

Fig. 5.6: A cascade of three levels at a width of 9 parallel processors (74 in total).

Following [17], it consists of one raw material supplier and one customer. To simplify the presentation of results we consider the following configuration: All suppliers have common length $l^e = 1$ and production velocity $v^e = 1$. The network is initially empty and a constant product inflow of 27 units per unit time is given. The capacities are chosen, such that processors need to run at maximum load in order to prevent queueing in absence of stochasticity. At initial and final stage, we have $\mu^e = 27$, second and fifth stage processors work at $\mu^e = 3$, whereas $\mu^e = 1$ at stages 3 and 4. Hence, we obtain a constant product flow of 27 units out of the network starting after 6 time units in the deterministic setting.

Stochastic default is introduced to small-load processors of stages 3 and 4 with common parameters $\tau_{\text{on}} = 9.5, \tau_{\text{off}} = 0.5$, thus each processor requires 5% repair time on long time average. Processors of stages $1, 2, 5, 6$ never break down, so a distribution of material into small-load stages is ensured. In Table 5.2, we present some computational results for the simulation of cascade networks, including network information, average number of switchings, average CPU time and average long-term network outflow.

	N	M	$\|\mathcal{T}\|$	av. $\|\mathcal{T}^*\|$	av. CPU time	av. $G_{\text{out}}^{\text{end}}(T)$
Cascade-9 (det)	31	74	361	–	3.51 s	27
Cascade-9 (sto)	31	74	361	430.53	9.49 s	25.29
Cascade-10 (sto)	34	82	361	478.65	11.75 s	26.76
Cascade-11 (sto)	37	90	361	525.89	13.91 s	26.92
Cascade-12 (sto)	40	98	361	573.84	16.51 s	26.94
Cascade-20 (sto)	64	162	361	955.15	45.09 s	26.95

Table 5.2: Monte-Carlo simulation of cascade networks (100 simulations) with $9, 10, 11$, and 20 parallel processors in stages 3 and 4 over the time interval $[t_0, T] = [0, 40]$: We show no. of vertices, no. of processors, size of deterministic time grid, average no. of switching events due to random breakdowns, average CPU time for a single simulation and average network outflow at $T = 40$. The processor configuration is chosen as for the Cascade-9 network, so we see, how the installation of additional, identical and parallel processors to stages 3 and 4 can compensate the loss of outflow due to stochastic default.

In particular, we can see that the number of switching events and therefore the average computation time increase with the size of the network. This is due to the switchings of the additional processors in the networks whose switching is traced exactly in Algorithm 4.2. Moreover, in the stochastic setting we need 12 or more processors on stages 3 and 4 to get close to the deterministic network outflow.

6. Conclusions. In this work, a model for stochastic default of processors has been introduced to supply chain networks. The independent exponential failure introduced in [10] for lines has been integrated into general network topologies of 'queue-processor' type [15]. A numerical algorithm for the detailed simulation of the stochastic network has been developed. Based on the PDP-property of the model, it traces all random switchings yet also conserves numerical mass. These stochastic simulation algorithm allows a full parallelisation of iterations, which eases computational effort, and is applicable to networks including loops. The effect and relevance of stochastic failures has been demonstrated in three network examples.

The future work is twofold. On the one hand, we intend to study load-dependent failure rates. This modelling corresponds to the idea that processors running perma-

nently at full capacity are more prone to fail. Difficulties may occur in the numerical analysis and simulation as well since failure rates will depend on the density of the processors. On the other hand, we would like to integrate optimisation issues into the stochastic network model. This can be done by routing the flow of products through the network such that the total averaged output will be maximised. In a first step, heuristics should be created to find suitable distribution rates.

Acknowledgement. This work was supported by the BC/DAAD ARC project "Robust simulation of networks with random switching" (1349/50021880) and the DFG grant HE 5386/6-1. The third author acknowledges support by the Leverhulme Trust funded project "Stability issues for SDEs" (F/00 276/K). Furthermore, we wish to thank stud. math. Sebastian Kühn (TU Kaiserslautern) for his assistance in performing the numerical experiments.

REFERENCES

[1] D. Armbruster, P. Degond, C. Ringhofer: *A model for the dynamics of large queuing networks and supply chains.* SIAM J. Appl. Math. **66**, pp. 896–920 (2004).

[2] M.K. Banda, M. Herty, A. Klar: *Gas flow in pipeline networks.* Networks and Heterogenous Media **1**(1), pp. 41–56 (2006).

[3] M.K. Banda, M. Herty, A. Klar: *Coupling conditions for gas networks governed by the isothermal Euler equations.* Networks and Heterogenous Media **1**(2), pp. 295–314 (2006).

[4] S. Battiston, D. Delli Gatti, M. Gallegati, B. Greenwald, J.E. Stiglitz: *Credit chains and bankruptcy propagation in production networks.* J. Economic Dynamics and Control **31**(6), pp. 2061–2084 (2007).

[5] G. Bretti, C. D'Apice, R. Manzo, B. Piccoli: *A continuum-discrete model for supply chains dynamics.* Networks and Heterogeneous Media **2**, pp. 661–694 (2007).

[6] G. Coclite, M. Garavello, B. Piccoli: *Traffic flow on road networks.* SIAM J. on Mathematical Analysis **36**, pp. 1862–1886 (2005).

[7] C. D'Apice, R. Manzo: *A fluid-dynamic model for supply chain.* Networks and Heterogeneous Media **1**(3), pp. 379–398 (2006).

[8] M.H.A. Davis: *Piecewise-deterministic Markov processes: A general class of non-diffusion stochastic models (with discussion).* J. Royal Statistical Society B **46**, pp. 353–388 (1984).

[9] M.H.A. Davis: *Markov Models and Optimisation.* Monograph on Statistics and Applied Probability 49, Chapmand & Hall, London, 1993.

[10] P. Degond, C. Ringhofer: *Stochastic dynamics of long supply chains with random breakdowns.* SIAM J. Appl. Math. **68**(1), pp. 59–79 (2007).

[11] A. Fügenschuh, M. Herty, A. Klar, A. Martin: *Combinatorial and Continuous Models for the Optimization of Traffic Flows on Networks.* SIAM J. on Optimization **16**(4), pp. 1155–1176 (2006).

[12] C.W. Gardiner: Handbook of Stochastic Methods for Physics, Chemistry and the Natural Sciences. Springer, 3rd ed., 2004.

[13] D.T. Gillespie: *A general method for numerically simulating the stochastic time evolution of coupled chemical reactions.* J. Phys. Chem. A **104**, pp. 403–434 (2000).

[14] D.T. Gillespie: *Approximate accelerated stochastic simulation of chemically reacting systems.* J. Chem. Phys. **115**, pp. 1716–1733 (2001).

[15] S. Göttlich, M. Herty, A. Klar: *Network models for supply chains.* Comm. Math. Sci. **3**(4), pp. 545–559 (2005).

[16] S. Göttlich, M. Herty, A. Klar: *Modelling and optimization of supply chains on complex networks.* Comm. Math. Sci. **4**(2), pp. 315–330 (2006).

[17] S. Göttlich, M. Herty, C. Ringhofer: *Optimization of order policies in supply networks.* European J. of Operational Research **202**(2), pp. 456–465 (2010).

[18] M. Gugat, M. Herty, A. Klar, G. Leugering: *Optimal control for traffic flow networks.* J. of Optimization Theory and Applicationy **126**(3), pp. 589-616 (2005).

[19] D. Helbing: Verkehrsdynamik. Springer Verlag, New York, Berlin, Heidelberg, 1997.

[20] D. Helbing, S. Lämmer, T. Seidel: *Physics, stability and dynamics of supply chains.* Physical Review E **70**, pp. 066116–066120 (2004).

[21] M. Herty, A. Klar: *Modeling, Simulation and Optimization of Traffic Flow Networks.* SIAM J. on Scientific Computing **25**(3), pp. 1066–1087 (2003).

[22] M. Herty, A. Klar, B. Piccoli: *Existence of solutions for supply chain models based on partial differential equations.* SIAM J. on Mathematical Analysis (2007).
[23] T. Kazangey, D.D. Sworder: *Effective federal policies for regulating residential housing.* Proc. Summer Computer Simulation Conf., pp. 1120–1128 (1971).
[24] F.P. Kelly, S. Zachary, I. Ziedins (Eds.): *Stochastic Networks: Theory and Apllications.* Oxford University Press, 2002.
[25] C. Kirchner, M. Herty, S. Göttlich, A. Klar: *Optimal Control for Continuous Supply Network Models.* Networks and Heterogenous Media **1**(4), pp. 675–688 (2006).
[26] G. Leugering, E. Schmidt: *On the modelling and stabilization of flows in networks of open channels.* SIAM J. Control and Optimization **41**(1), pp. 164–180 (2002).
[27] X. Mao, C. Yuan: Stochastic Differential Equations with Markovian Switching. Imperial College Press, 2006.
[28] M. Mariton: Jump Linear Systems in Automatic Control. Marcel Dekker, 1990.
[29] M. Martin, A.M. Möller, S. Moritz: *Mixed Integer Models for the stationary case of gas network optimization.* Math. Programming **105**, pp. 563–582 (2005).
[30] M. Steinbach: *On PDE Solution in Transient Optimization of Gas Networks.* J. Comput. Appl. Math. **203**(2), pp. 345–361 (2007).
[31] G. Steinebach, S. Rademacher, P. Rentrop, M. Schulz: *Mechanisms of coupling in river flow simulation systems.* J. Comput. Appl. Math. **168**, pp. 459-470 (2004).
[32] D.D. Sworder, V.G. Robinson: *Feedbackregulators for jump parameter systems with state and control depend transistion rates.* IEEE Trans. Automat. Control **19**, pp. 355–360 (1773).
[33] A.S. Willsky, B.C. Rogers: *Stochastic stability research for complex power systems.* DOE Contract, LIDS, MIT, Rep., ET-76-C-01-2295.
[34] G.G. Yin, Q. Zhang: Discrete-Time Markov Chains. Springer-Verlag, Berlin, 2005.