Performance Modelling of Computer Systems

Mirco Tribastone

Institut für Informatik
Ludwig-Maximilians-Universität München

Differential Approximation of PEPA Models

Motivation

- State-space explosion for a large number of sequential components.

Example

$$Download \stackrel{def}{=} (transfer, r_d).Think$$

$$Think \stackrel{def}{=} (think, r_t).Download$$

$$Upload \stackrel{def}{=} (transfer, r_u).Log$$

$$Log \stackrel{def}{=} (log, r_l).Upload$$

$$System := Download[N_C] \underset{\{transfer\}}{\bowtie} Upload[N_S]$$

N_C	N_S	$ds(System)$
1	1	4
2	2	16
10	10	1048576
N_C	N_S	$2^{N_C+N_S}$

Deterministic Approximation

- The goal is to find an approximation which is independent from the population counts of the sequential components.
- The approximation is deterministic and is based on a system of ordinary differential equations (ODEs).
- Define

$$\mathbf{N}(t) = \big(N(Dowload, t), N(Think, t), N(Upload, t), N(Log, t)\big).$$

- The ODEs will have the form $d\mathbf{N}(t)/dt = F\big(\mathbf{N}(t)\big)$, in components:

$$\frac{dN(Dowload, t)}{dt} = f_1\big(\mathbf{N}(t)\big), \qquad \frac{dN(Think, t)}{dt} = f_2\big(\mathbf{N}(t)\big),$$
$$\frac{dN(Upload, t)}{dt} = f_3\big(\mathbf{N}(t)\big), \qquad \frac{dN(Log, t)}{dt} = f_4\big(\mathbf{N}(t)\big).$$

- $\mathbf{N}(t)$ will be the solution of an initial value problem with

$$\mathbf{N}(0) = (N_C, 0, N_S, 0).$$

Some Remarks

- These ODEs are different than those obtained from the forward equations of the continuous-time Markov chain which underlies a PEPA model.
- Firstly, because the state descriptor is different.
- Secondly, each of the forward equations gives the time-course evolution of the probability of being in a state of the chain. Therefore the total number of equations equals the state space size (which may get extremely large).
- Instead, in the approximating ODEs the number of equations is independent from the population levels, but it does depend on the number of local states of the sequential components in the system.

Numerical Vector Form

Consider the initial PEPA process $Download[N_C] \underset{\{transfer\}}{\bowtie} Upload[N_S]$.

The following general structure for its derivative set may be inferred:

$$C_1 \parallel C_2 \parallel \ldots \parallel C_{N_C} \underset{\{transfer\}}{\bowtie} S_1 \parallel S_2 \parallel \ldots \parallel S_{N_S},$$

with, for all $1 \leq i \leq N_C$ and $1 \leq j \leq N_S$,

$$C_i \in ds(Download) = \{Download, Think\},$$
$$S_j \in ds(Upload) = \{Upload, Log\}.$$

The numerical vector form (NVF) is an alternative state representation which counts how many sequential components exhibit a local state.

$$\left(\sum_{C_i = Download} 1, \sum_{C_i = Think} 1, \sum_{S_j = Upload} 1, \sum_{S_j = Log} 1 \right)$$

Examples

$$\left(\sum_{C_i = Download} 1, \sum_{C_i = Think} 1, \sum_{S_j = Upload} 1, \sum_{S_j = Log} 1 \right) \equiv (n_d, n_t, n_u, n_l)$$

Process	*NVF*
$Download[N_C] \underset{\{transfer\}}{\bowtie} Upload[N_S]$	$(N_C, 0, N_S, 0)$
$Download \parallel Think \underset{\{transfer\}}{\bowtie} Log \parallel Log$	$(1, 1, 0, 2)$
$Think \parallel Download \underset{\{transfer\}}{\bowtie} Log \parallel Log$	$(1, 1, 0, 2)$

Programme

The ODEs are obtained through a continuous-time Markov chain in which the state descriptor is in the numerical vector form. This CTMC is called the population-based CTMC.

It needs not be explicitly derived, instead a symbolic representation will contain all the information necessary to forming the ODEs.

In this way, the behaviour of the ODE solution can be related to that of the stochastic process, thus justifying the differential approximation introduced.

Population-based CTMC

Consider a PEPA component P such that there exists $P \xrightarrow{(\alpha,r)} P'$, with $r > 0$, and $ds(P) = \{P, P'\}$.

The process $P[2] = P \parallel P$ admits the following two derivations

$$\frac{P \xrightarrow{(\alpha,r)} P'}{P \parallel P \xrightarrow{(\alpha,r)} P' \parallel P} \qquad \text{and} \qquad \frac{P \xrightarrow{(\alpha,r)} P'}{P \parallel P \xrightarrow{(\alpha,r)} P \parallel P'}$$

Similarly, the process $P[3] = P \parallel P \parallel P$ admits three derivations:

$$P \parallel P \parallel P \xrightarrow{(\alpha,r)} P' \parallel P \parallel P,$$
$$P \parallel P \parallel P \xrightarrow{(\alpha,r)} P \parallel P' \parallel P,$$
$$P \parallel P \parallel P \xrightarrow{(\alpha,r)} P \parallel P \parallel P'.$$

Array of Components

In general, the process $P[n_p]$, for any $n_p \in \mathbb{N}$, admits n_p derivations where $n_p - 1$ components do not change state and one component behaves as P':

$$\underbrace{P \parallel P \parallel \ldots \parallel P}_{n_p} \xrightarrow{(\alpha,r)} P' \parallel \underbrace{P \parallel P \parallel \ldots \parallel P}_{n_p-1},$$

$$\underbrace{P \parallel P \parallel \ldots \parallel P}_{n_p} \xrightarrow{(\alpha,r)} \underbrace{P \parallel \ldots \parallel P}_{i} \parallel P' \parallel \underbrace{P \parallel \ldots \parallel P}_{n_p-i-1}, \qquad 1 \leq i \leq n_p - 2$$

$$\underbrace{P \parallel P \parallel \ldots \parallel P}_{n_p} \xrightarrow{(\alpha,r)} \underbrace{P \parallel \ldots \parallel P}_{n_p-1} \parallel P'.$$

In the population-based CTMC, we observe that all target states correspond to the same state $(n_p - 1, 1)$, hence one can write

$$(n_p, 0) \xrightarrow{(\alpha, n_p \times r)} (n_p - 1, 1)$$

and this holds for any process P, and for any n_p, α, and $r > 0$.

Arrays in Parallel

Therefore, from the knowledge of a single process $P \xrightarrow{(\alpha,r)} P'$ we can infer the behaviour of n_p such processes in parallel.

Now, let us consider the process $P[n_p] \parallel Q[n_q]$, where $ds(Q) = \{Q, Q'\}$ and there exists $Q \xrightarrow{(\beta,s)} Q', s > 0$.
According to the semantics of PEPA, we can write n_p derivations of kind,

$$P[n_p] \parallel Q[n_q] \xrightarrow{(\alpha,r)} \underbrace{P \parallel \ldots \parallel P}_{i} \parallel P' \parallel \underbrace{P \parallel \ldots \parallel P}_{n_p-i-1} \parallel Q[n_q], \quad 0 \leq i \leq n_p,$$

and n_q derivations of kind

$$P[n_p] \parallel Q[n_q] \xrightarrow{(\beta,s)} P[n_p] \parallel \underbrace{Q \parallel \ldots \parallel Q}_{j} \parallel Q' \parallel \underbrace{Q \parallel \ldots \parallel Q}_{n_q-j-1}, \quad 0 \leq j \leq n_q.$$

Arrays in Parallel

In the NVF, these two groups of transitions can be represented as follows:

$$(n_p, 0, n_q, 0) \xrightarrow{(\alpha, n_p \times r)} (n_p - 1, 1, n_q, 0)$$
$$(n_p, 0, n_q, 0) \xrightarrow{(\beta, n_q \times s)} (n_p, 0, n_q - 1, 1)$$

In general, for a state $(n_p, n_{p'}, n_q, n_{q'})$, with $n_{p'}, n_{q'} > 0$ the transitions may be written as follows:

$$(n_p, n_{p'}, n_q, n_{q'}) \xrightarrow{(\alpha, n_p \times r)} (n_p - 1, n_{p'} + 1, n_q, n_{q'})$$
$$(n_p, n_{p'}, n_q, n_{q'}) \xrightarrow{(\alpha, n_q \times s)} (n_p, n_{p'}, n_q - 1, n_{q'} + 1)$$

(...and all transitions in which P' and Q' are involved)

Again, the population-based transition may be inferred from the behaviour of the single components

Arrays in Parallel

In, the running example, the initial state $Download[N_C] \underset{\{transfer\}}{\bowtie} Upload[N_S]$ enables us to say that the generic state of the population-based CTMC will be in the form:

$$Download[n_d] \parallel Think[n_t] \underset{\{transfer\}}{\bowtie} Upload[n_u] \parallel Log[n_l]$$

with state descriptor (n_d, n_t, n_u, n_l).

The derivations so far considered allow us to infer transitions for $Download[n_d] \parallel Think[n_t]$.

Next, we wish to infer derivations for $P[n_p] \underset{L}{\bowtie} Q[n_q]$, with $L \neq \emptyset$, from the basic synchronisation behaviour of $P \underset{L}{\bowtie} Q$.

Synchronisation between Arrays

Let $ds(P) = \{P, P'\}$, $P \xrightarrow{(\alpha,r)} P'$ and $ds(Q) = \{Q, Q'\}$, $Q \xrightarrow{(\alpha,s)} Q'$.

$$P \underset{\{\alpha\}}{\bowtie} Q \xrightarrow{\left(\alpha, \frac{r}{r} \times \frac{s}{s} \times \min(r,s)\right)} P' \underset{\{\alpha\}}{\bowtie} Q'$$

Consider now the process $P[2] \underset{\{\alpha\}}{\bowtie} Q[2]$.

$$P[2] \underset{\{\alpha\}}{\bowtie} Q[2] \xrightarrow{\left(\alpha, \frac{r}{2\times r} \times \frac{s}{2\times s} \min(2\times r, 2\times s)\right)} P' \parallel P \underset{\{\alpha\}}{\bowtie} Q' \parallel Q$$

$$P[2] \underset{\{\alpha\}}{\bowtie} Q[2] \xrightarrow{\left(\alpha, \frac{r}{2\times r} \times \frac{s}{2\times s} \min(2\times r, 2\times s)\right)} P' \parallel P \underset{\{\alpha\}}{\bowtie} Q \parallel Q'$$

$$P[2] \underset{\{\alpha\}}{\bowtie} Q[2] \xrightarrow{\left(\alpha, \frac{r}{2\times r} \times \frac{s}{2\times s} \min(2\times r, 2\times s)\right)} P \parallel P' \underset{\{\alpha\}}{\bowtie} Q' \parallel Q$$

$$P[2] \underset{\{\alpha\}}{\bowtie} Q[2] \xrightarrow{\left(\alpha, \frac{r}{2\times r} \times \frac{s}{2\times s} \min(2\times r, 2\times s)\right)} P \parallel P' \underset{\{\alpha\}}{\bowtie} Q \parallel Q'$$

In the NVF $(n_p, n_{p'}, n_q, n_{q'})$ there is a transition

$$(2, 0, 2, 0) \xrightarrow{4 \times \frac{r}{2\times r} \times \frac{s}{2\times s} \min(2\times r, 2\times s) = \min(2\times r, 2\times s)} (1, 1, 1, 1)$$

Apparent Rate Calculation

Apparent rates can be also inferred from the rates of the individual components:

$$
\begin{aligned}
r_\alpha(P[n_p]) &= n_p \times r_\alpha(P) \\
r_\alpha(P[n_p] \parallel Q[n_q]) &= n_p \times r_\alpha(P) + n_q \times r_\alpha(Q) \\
r_\alpha(P[n_p] \underset{L}{\bowtie} Q[n_q]) &= \begin{cases} \min(n_p \times r_\alpha(P), n_q \times r_\alpha(Q)) & \text{if } \alpha \in L \\ n_p \times r_\alpha(P) + n_q \times r_\alpha(Q) & \text{otherwise} \end{cases} \\
& \dots
\end{aligned}
$$

General Procedure

1. Given a PEPA model, the cooperation structure of the process can be found. For example, $\mathit{Download}[N_C] \underset{\{\mathit{transfer}\}}{\bowtie} \mathit{Upload}[N_S]$ gives rise to

$$\mathit{Download}[n_d] \parallel \mathit{Think}[n_t] \underset{\{\mathit{transfer}\}}{\bowtie} \mathit{Upload}[n_u] \parallel \mathit{Log}[n_l],$$

   with NVF $\mathbf{n} = (n_d, n_t, n_u, n_l)$.
2. Interpret the symbols within square brackets (e.g., n_d, n_t, n_u, n_l) as variables. For instance setting $n_d = N_C, n_t = n_l = 0, n_u = N_S$ gives the initial state.
3. Infer transitions for the generic state (parametric in the counter variables) from the behaviour of the single components. . .

Parametric Transitions

$$\frac{P \xrightarrow{(\alpha, r)} P'}{P[n_p] \xrightarrow{(\alpha, n_p \times r)}_* P[n_p - 1] \parallel P'}$$

For example,

$$\frac{Download \xrightarrow{(transfer, r_d)} Think}{Download[n_d] \xrightarrow{(transfer, n_d \times r_d)}_* Download[n_d - 1] \parallel Think}.$$

$$\frac{Upload \xrightarrow{(transfer, r_u)} Log}{Upload[n_u] \xrightarrow{(transfer, n_u \times r_u)}_* Upload[n_u - 1] \parallel Log}.$$

Parametric Transitions

$$\frac{P \xrightarrow{(\alpha, r_p(\mathbf{n}))}_* P'}{P \underset{L}{\bowtie} Q \xrightarrow{(\alpha, r_p(\mathbf{n}))}_* P' \underset{L}{\bowtie} Q}, \alpha \notin L, \qquad \frac{Q \xrightarrow{(\alpha, r_q(\mathbf{n}))}_* Q'}{P \underset{L}{\bowtie} Q \xrightarrow{(\alpha, r_p(\mathbf{n}))}_* P \underset{L}{\bowtie} Q'}, \alpha \notin L$$

For example,

$$\frac{Download[n_d] \xrightarrow{(transfer, n_d \times r_d)}_* Download[n_d - 1] \parallel Think}{Download[n_d] \parallel Think[n_t] \xrightarrow{(transfer, n_d \times r_d)}_* Download[n_d - 1] \parallel Think \parallel Think[n_t]}$$

$$\frac{Upload[n_u] \xrightarrow{(transfer, n_u \times r_u)}_* Upload[n_u - 1] \parallel Log}{Upload[n_u] \parallel Log[n_l] \xrightarrow{(transfer, n_u \times r_u)}_* Upload[n_u - 1] \parallel Log \parallel Log[n_l]}$$

Parametric Transitions

$$\frac{P \xrightarrow{(\alpha, r_p(\mathbf{n}))}_* P' \qquad Q \xrightarrow{(\alpha, r_q(\mathbf{n}))}_* Q'}{P \underset{L}{\bowtie} Q \xrightarrow{(\alpha, R(\mathbf{n}))}_* P' \underset{L}{\bowtie} Q'}, \alpha \in L$$

$$R(\mathbf{n}) = \frac{r_p(\mathbf{n})}{r_\alpha(P)} \frac{r_q(\mathbf{n})}{r_\alpha(Q)} \min(r_\alpha(P), r_\alpha(Q))$$

For example,

$$\frac{\begin{array}{c} D[n_d] \parallel T[n_t] \xrightarrow{(transfer, n_d \times r_d)}_* D[n_d - 1] \parallel T[n_t + 1] \\ U[n_u] \parallel L[n_l] \xrightarrow{(transfer, n_u \times r_u)}_* U[n_u - 1] \parallel L \parallel L[n_l] \end{array}}{\begin{array}{c} D[n_d] \parallel T[n_t] \underset{\{\alpha\}}{\bowtie} U[n_u] \parallel L[n_l] \xrightarrow{(transfer, \min(n_d\, r_d, n_u\, r_u))}_* \\ D[n_d - 1] \parallel T[n_t + 1] \underset{\{\alpha\}}{\bowtie} U[n_u - 1] \parallel L[n_l + 1] \end{array}}$$

$$(n_d, n_t, n_u, n_l) \xrightarrow{(transfer, \min(n_d\, r_d, n_u\, r_u))}_* (n_d - 1, n_t + 1, n_u - 1, n_l + 1)$$

Continuous Approximation

$$(n_d, n_t, n_u, n_l) \xrightarrow{(transfer, \min(n_d\, r_d, n_u\, r_u))}_* (n_d - 1, n_t + 1, n_u - 1, n_l + 1)$$
$$(n_d, n_t, n_u, n_l) \xrightarrow{(think, n_t\, r_t)}_* (n_d + 1, n_t - 1, n_u, n_l)$$
$$(n_d, n_t, n_u, n_l) \xrightarrow{(log, n_l\, r_l)}_* (n_d, n_t, n_u + 1, n_l - 1)$$

- $\mathbf{N}(t) = \big(N(Dowload, t), N(Think, t), N(Upload, t), N(Log, t)\big)$ is a vector of continuous variables.
- After one time unit, the population counts of some components change by an amount equal to the transition rate, e.g.

  $N(Dowload, t{+}1) \approx N(Dowload, t){-}\min(N(Dowload, t)\, r_d, N(Upload, t)\, r_u)$.
- We assume that after Δt time units, components change linearly with this rate, e.g.

$$\begin{aligned} N(Dowload, t + \Delta t) = {} & N(Dowload, t) \\ & - \min(N(Dowload, t)\, r_d, N(Upload, t)\, r_u)\Delta t + o(\Delta t) \end{aligned}$$

Continuous Approximation

$$(n_d, n_t, n_u, n_l) \xrightarrow{(transfer, \min(n_d\, r_d, n_u\, r_u))}_* (n_d - 1, n_t + 1, n_u - 1, n_l + 1)$$

$$(n_d, n_t, n_u, n_l) \xrightarrow{(think, n_t\, r_t)}_* (n_d + 1, n_t - 1, n_u, n_l)$$

$$(n_d, n_t, n_u, n_l) \xrightarrow{(log, n_l\, r_l)}_* (n_d, n_t, n_u + 1, n_l - 1)$$

$$N(Dowload, t + \Delta t) = N(Dowload, t) - \min(N(Dowload, t)\, r_d, N(Upload, t)\, r_u)\Delta t + o(\Delta t)$$

Rearranging and taking the limit $\Delta t \to 0$ yields

$$\frac{dN(Download, t)}{dt} = -\min(N(Dowload, t)\, r_d, N(Upload, t)\, r_u)$$

But this is only a partial view of the overall model!

$$\frac{dN(Download, t)}{dt} = -\min(N(Dowload, t)\, r_d, N(Upload, t)\, r_u) + N(Think, t) r_t$$

ODE Generation

$$(n_d, n_t, n_u, n_l) \xrightarrow{(transfer, \min(n_d\, r_d, n_u\, r_u))}_* (n_d - 1, n_t + 1, n_u - 1, n_l + 1)$$

$$(n_d, n_t, n_u, n_l) \xrightarrow{(think, n_t\, r_t)}_* (n_d + 1, n_t - 1, n_u, n_l)$$

$$(n_d, n_t, n_u, n_l) \xrightarrow{(log, n_l\, r_l)}_* (n_d, n_t, n_u + 1, n_l - 1)$$

In general, for a transition $\mathbf{n} \xrightarrow{(\alpha, f(\mathbf{n}))} \mathbf{n}'$ define $f_\alpha(\mathbf{n}, \mathbf{l}) := f(\mathbf{n})$, with $\mathbf{l} = \mathbf{n}' - \mathbf{n}$. The system of ODEs is defined as

$$\frac{d\mathbf{N}(t)}{dt} = \sum_\alpha \sum_{\mathbf{l}} \mathbf{l} f_\alpha(\mathbf{N}(t), \mathbf{l}).$$

In our example,

$$\frac{d\mathbf{N}(t)}{dt} = (-1, 1, -1, 1) \min(N(Dowload, t)\, r_d, N(Upload, t)\, r_u) + (1, -1, 0, 0) N(Think, t) r_t + (0, 0, 1, -1) N(Log, t) r_l$$

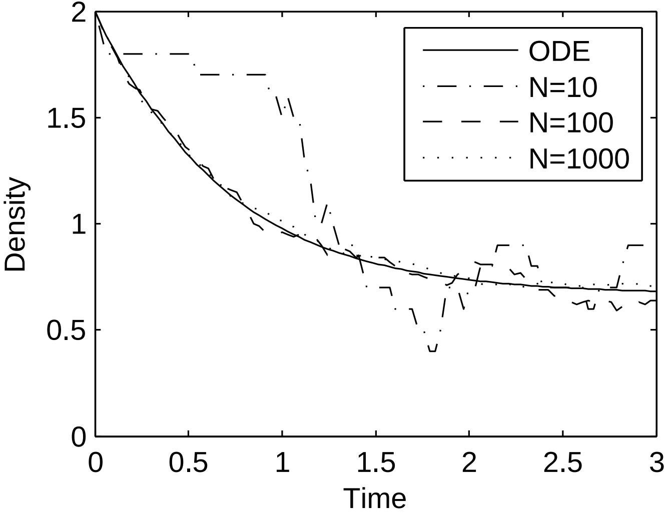
ODE Generation

$$(n_d, n_t, n_u, n_l) \xrightarrow{(transfer, \min(n_d\, r_d, n_u\, r_u))}_* (n_d - 1, n_t + 1, n_u - 1, n_l + 1)$$
$$(n_d, n_t, n_u, n_l) \xrightarrow{(think, n_t\, r_t)}_* (n_d + 1, n_t - 1, n_u, n_l)$$
$$(n_d, n_t, n_u, n_l) \xrightarrow{(log, n_l\, r_l)}_* (n_d, n_t, n_u + 1, n_l - 1)$$

In components:

$$\frac{dN(Download, t)}{dt} = -\min(N(Dowload, t)\, r_d, N(Upload, t)\, r_u) + N(Think, t) r_t$$
$$\frac{dN(Think, t)}{dt} = +\min(N(Dowload, t)\, r_d, N(Upload, t)\, r_u) - N(Think, t) r_t$$
$$\frac{dN(Upload, t)}{dt} = -\min(N(Dowload, t)\, r_d, N(Upload, t)\, r_u) + N(Log, t) r_l$$
$$\frac{dN(Log, t)}{dt} = +\min(N(Dowload, t)\, r_d, N(Upload, t)\, r_u) - N(Log, t) r_l$$

Relationship between ODE and CTMC



Relationship between ODE and CTMC

	$n = 1$			$n = 10$			$n = 50$			$n = 100$		
Component	5%	Avg.	95%	5%	Avg.	95%	5%	Avg.	95%	5%	Avg.	95%
$Cl : Request$	0.09%	19.62%	74.20%	0.01%	5.15%	29.09%	0.01%	1.87%	8.73%	0.01%	1.16%	4.85%
$Cl : Wait$	0.22%	17.09%	59.36%	0.03%	1.97%	7.57%	0.02%	0.76%	2.60%	0.02%	0.55%	1.70%
$Cl : Think$	0.70%	31.13%	87.57%	0.09%	2.96%	9.92%	0.06%	1.71%	6.00%	0.07%	1.62%	5.16%
$Sr : Wait$	0.31%	13.02%	50.49%	0.06%	2.46%	9.66%	0.05%	1.24%	4.56%	0.05%	1.23%	4.14%
$Sr : Fresh$	0.56%	20.21%	60.54%	0.09%	3.74%	12.81%	0.03%	2.09%	7.03%	0.06%	1.82%	5.68%
$Sr : Force$	1.20%	31.02%	85.57%	0.29%	4.39%	11.49%	0.22%	3.63%	9.17%	0.21%	3.27%	7.80%
$Sr : Write$	0.95%	27.68%	80.39%	0.21%	4.14%	12.38%	0.12%	2.91%	9.26%	0.10%	2.64%	8.91%
$Sr : Reply$	0.26%	24.69%	71.60%	0.07%	3.70%	13.10%	0.04%	1.69%	4.70%	0.05%	1.48%	5.44%
$Sr : Repair$	0.16%	13.19%	50.63%	0.01%	2.77%	11.37%	0.01%	1.32%	5.32%	0.02%	0.90%	3.92%
$Db : Wait$	0.01%	3.64%	20.21%	0.01%	0.77%	3.66%	0.01%	0.43%	1.70%	0.01%	0.38%	1.33%
$Db : Update$	0.04%	4.04%	17.08%	0.03%	1.07%	4.33%	0.01%	0.79%	2.93%	0.01%	0.81%	2.76%
$Rb : Gather$	0.05%	4.00%	16.56%	0.02%	1.09%	3.54%	0.02%	0.95%	3.23%	0.02%	0.89%	3.52%
$Rb : Write$	0.03%	2.82%	15.60%	0.02%	1.03%	3.12%	0.02%	0.91%	3.01%	0.01%	0.89%	3.00%

Numerical experimentation over 300 randomly generated models. Percentage absolute error with respect to steady-state stochastic simulation of the PEPA model.