PEPA

**Prefix** $(\alpha, \, r).P$ — action $\alpha$ happens at rate $r$ before transitioning to component $P$

**Choice** $P + Q$ — allows either $P$ or $Q$ to occur, choice determined by races

**Cooperation** $P \boxtimes^L Q$ — For $\alpha \in L$, if $P$ and $Q$ both enable an $\alpha$-activity, only then can both make the $\alpha$-transition simultaneously at the rate of the slowest

**Hiding** $P/L$ — Action types in set $L$ become the hidden $\tau$ action, which cannot be cooperated over

**Constant** $A \overset{def}{=} P$ — Assigns the label $A$ to component $P$, allows $P \overset{def}{=} (\alpha, \, r).P$ etc.
Prefix \((\alpha, r).P\) — action \(\alpha\) happens at rate \(r\) before transitioning to component \(P\).

Choice \(P + Q\) — allows either \(P\) or \(Q\) to occur, choice determined by races.

Cooperation \(P \otimes^L Q\) — For \(\alpha \in L\), if \(P\) and \(Q\) both enable an \(\alpha\)-activity, only then can both make the \(\alpha\)-transition simultaneously at the rate of the slowest.

Hiding \(P/L\) — Action types in set \(L\) become the hidden \(\tau\) action, which cannot be cooperated over.

Constant \(A \overset{\text{def}}{=} P\) — Assigns the label \(A\) to component \(P\), allows \(P \overset{\text{def}}{=} (\alpha, r).P\) etc.
PEPA

**Prefix** \((\alpha, r).P\) — action \(\alpha\) happens at rate \(r\) before transitioning to component \(P\)

**Choice** \(P + Q\) — allows either \(P\) or \(Q\) to occur, choice determined by races

**Cooperation** \(P \overset{\alpha}{\underset{L}{\otimes}} Q\) — For \(\alpha \in L\), if \(P\) and \(Q\) both enable an \(\alpha\)-activity, only then can both make the \(\alpha\)-transition simultaneously at the rate of the slowest

**Hiding** \(P/L\) — Action types in set \(L\) become the hidden \(\tau\) action, which cannot be cooperated over

**Constant** \(A \overset{\text{def}}{=} P\) — Assigns the label \(A\) to component \(P\), allows \(P \overset{\text{def}}{=} (\alpha, r).P\) etc.
**Prefix** \((\alpha, r).P\) — action \(\alpha\) happens at rate \(r\) before transitioning to component \(P\)

**Choice** \(P + Q\) — allows either \(P\) or \(Q\) to occur, choice determined by races

**Cooperation** \(P \bigotimes^L Q\) — For \(\alpha \in L\), if \(P\) and \(Q\) both enable an \(\alpha\)-activity, only then can both make the \(\alpha\)-transition simultaneously at the rate of the slowest

**Hiding** \(P/L\) — Action types in set \(L\) become the hidden \(\tau\) action, which cannot be cooperated over

**Constant** \(A \overset{\text{def}}{=} P\) — Assigns the label \(A\) to component \(P\), allows \(P \overset{\text{def}}{=} (\alpha, r).P\) etc.
Prefix $(\alpha, r).P$ — action $\alpha$ happens at rate $r$ before transitioning to component $P$

Choice $P + Q$ — allows either $P$ or $Q$ to occur, choice determined by races

Cooperation $P \otimes^L Q$ — For $\alpha \in L$, if $P$ and $Q$ both enable an $\alpha$-activity, only then can both make the $\alpha$-transition simultaneously at the rate of the slowest

Hiding $P/L$ — Action types in set $L$ become the hidden $\tau$ action, which cannot be cooperated over

Constant $A \trianglerighteq P$ — Assigns the label $A$ to component $P$, allows $P \trianglerighteq (\alpha, r).P$ etc.
Prefix \((\alpha, r).P\) — action \(\alpha\) happens at rate \(r\) before transitioning to component \(P\)

Choice \(P + Q\) — allows either \(P\) or \(Q\) to occur, choice determined by races

Cooperation \(P \Join[L] Q\) — For \(\alpha \in L\), if \(P\) and \(Q\) both enable an \(\alpha\)-activity, only then can both make the \(\alpha\)-transition simultaneously at the rate of the slowest

Hiding \(P/L\) — Action types in set \(L\) become the hidden \(\tau\) action, which cannot be cooperated over

Constant \(A \overset{\text{def}}{=} P\) — Assigns the label \(A\) to component \(P\), allows \(P \overset{\text{def}}{=} (\alpha, r).P\) etc.
A simple client/server model in PEPA

\[
C \overset{\text{def}}{=} (\text{request}, r_{\text{req}}).C_{\text{wait}} \\
C_{\text{wait}} \overset{\text{def}}{=} (\text{data}, r_{\text{data}}).C_{\text{think}} \\
C_{\text{think}} \overset{\text{def}}{=} (\text{think}, r_{\text{think}}).C
\]

\[
S \overset{\text{def}}{=} (\text{request}, r_{\text{req}}).S_{\text{get}} \\
+ (\text{break}, r_{\text{break}}).S_{\text{broken}} \\
S_{\text{get}} \overset{\text{def}}{=} (\text{data}, r_{\text{data}}).S \\
+ (\text{break}, r_{\text{break}}).S_{\text{broken}} \\
S_{\text{broken}} \overset{\text{def}}{=} (\text{reset}, r_{\text{reset}}).S
\]

\[
\text{System} \overset{\text{def}}{=} (C \parallel \ldots \parallel C) \quad \begin{array}{c} \oplus \end{array} \quad (S \parallel \ldots \parallel S)
\]

\[
3^N_C + N_S \quad \text{or} \quad \frac{N_C^2 N_S^2}{4} \quad \text{states}
\]

{rh, jb}@doc.ic.ac.uk
A simple client/server model in PEPA

\[
C \overset{\text{def}}{=} (\text{request}, r_{\text{req}}).C_{\text{wait}} \\
C_{\text{wait}} \overset{\text{def}}{=} (\text{data}, r_{\text{data}}).C_{\text{think}} \\
C_{\text{think}} \overset{\text{def}}{=} (\text{think}, r_{\text{think}}).C
\]

\[
S \overset{\text{def}}{=} (\text{request}, r_{\text{req}}).S_{\text{get}} + (\text{break}, r_{\text{break}}).S_{\text{broken}} \\
S_{\text{get}} \overset{\text{def}}{=} (\text{data}, r_{\text{data}}).S + (\text{break}, r_{\text{break}}).S_{\text{broken}} \\
S_{\text{broken}} \overset{\text{def}}{=} (\text{reset}, r_{\text{reset}}).S
\]

System \overset{\text{def}}{=} (C \parallel \ldots \parallel C) \{\text{request, data}\} (S \parallel \ldots \parallel S)_{N_C} \quad \square \quad (S \parallel \ldots \parallel S)_{N_S}

\[3^{N_C} + N_S \text{ or } \approx \frac{N_C^2 N_S^2}{4} \text{ states}\]
A simple client/server model in PEPA

\[
C \overset{\text{def}}{=} (\text{request}, r_{\text{req}}).C_{\text{wait}} \\
C_{\text{wait}} \overset{\text{def}}{=} (\text{data}, r_{\text{data}}).C_{\text{think}} \\
C_{\text{think}} \overset{\text{def}}{=} (\text{think}, r_{\text{think}}).C
\]

\[
S \overset{\text{def}}{=} (\text{request}, r_{\text{req}}).S_{\text{get}} \\
+ (\text{break}, r_{\text{break}}).S_{\text{broken}}
\]

\[
S_{\text{get}} \overset{\text{def}}{=} (\text{data}, r_{\text{data}}).S \\
+ (\text{break}, r_{\text{break}}).S_{\text{broken}}
\]

\[
S_{\text{broken}} \overset{\text{def}}{=} (\text{reset}, r_{\text{reset}}).S
\]

\[
\text{System} \overset{\text{def}}{=} (C \parallel \ldots \parallel C) \overset{\text{∅}}{\begin{array}{c}
\{\text{request}, \text{data}\}
\end{array}} \overset{\text{∅}}{\begin{array}{c}
\{S \parallel \ldots \parallel S\}
\end{array}}
\]

\[3^{N_C}N_S \quad \text{or} \quad \frac{N_C^2N_S^2}{4} \quad \text{states}\]
Fluid analysis via ODEs

\[ C \overset{\text{def}}{=} (\text{request}, r_{\text{req}}) \cdot C_{\text{wait}} \]
\[ C_{\text{wait}} \overset{\text{def}}{=} (\text{data}, r_{\text{data}}) \cdot C_{\text{think}} \]
\[ C_{\text{think}} \overset{\text{def}}{=} (\text{think}, r_{\text{think}}) \cdot C \]

\[ S \overset{\text{def}}{=} (\text{request}, r_{\text{req}}) \cdot S_{\text{get}} \]
\[ + (\text{break}, r_{\text{break}}) \cdot S_{\text{broken}} \]
\[ S_{\text{get}} \overset{\text{def}}{=} (\text{data}, r_{\text{data}}) \cdot S \]
\[ + (\text{break}, r_{\text{break}}) \cdot S_{\text{broken}} \]
\[ S_{\text{broken}} \overset{\text{def}}{=} (\text{reset}, r_{\text{reset}}) \cdot S \]

\[ \text{System} \overset{\text{def}}{=} \left( C \parallel \ldots \parallel C \right) \underbrace{\left( S \parallel \ldots \parallel S \right)}_{N_C} \]

At time \( t \), let \( N_C(t) \) count the number of \( C \) components . . .

- \( C \rightarrow C_{\text{wait}} \) at rate \( r_{\text{req}} \times \min(N_C(t), N_S(t)) \)
- \( C_{\text{think}} \rightarrow C \) at rate \( r_{\text{think}} \times N_{C_{\text{think}}}(t) \)

So approximate \( N(t) \) by \( v(t) \), defined by:

\[ \frac{dv_C(t)}{dt} = -r_{\text{req}} \times \min(v_C(t), v_S(t)) + r_{\text{think}} \times v_{C_{\text{think}}}(t) \]
Fluid analysis via ODEs

\[ C \overset{\text{def}}{=} (\text{request}, r_{\text{req}}).C_{\text{wait}} \]
\[ C_{\text{wait}} \overset{\text{def}}{=} (\text{data}, r_{\text{data}}).C_{\text{think}} \]
\[ C_{\text{think}} \overset{\text{def}}{=} (\text{think}, r_{\text{think}}).C \]

\[ S \overset{\text{def}}{=} (\text{request}, r_{\text{req}}).S_{\text{get}} + (\text{break}, r_{\text{break}}).S_{\text{broken}} \]
\[ S_{\text{get}} \overset{\text{def}}{=} (\text{data}, r_{\text{data}}).S + (\text{break}, r_{\text{break}}).S_{\text{broken}} \]
\[ S_{\text{broken}} \overset{\text{def}}{=} (\text{reset}, r_{\text{reset}}).S \]

\[ \text{System} \overset{\text{def}}{=} (C \parallel \ldots \parallel C) \underbrace{\bigotimes_{\text{request, data}}}_{N_C} (S \parallel \ldots \parallel S) \underbrace{\bigotimes_{\text{reset}}}_{N_S} \]

At time \( t \), let \( N_C(t) \) count the number of \( C \) components . . .

\[ \bullet \quad C \rightarrow C_{\text{wait}} \text{ at rate } r_{\text{req}} \times \min(N_C(t), N_S(t)) \]
\[ \bullet \quad C_{\text{think}} \rightarrow C \text{ at rate } r_{\text{think}} \times N_{C_{\text{think}}}(t) \]

So approximate \( N.(t) \) by \( v.(t) \), defined by:
\[
\frac{dv_C(t)}{dt} = -r_{\text{req}} \times \min(v_C(t), v_S(t)) + r_{\text{think}} \times v_{C_{\text{think}}}(t)
\]

{rh, jb}@doc.ic.ac.uk
Functional law of large numbers

Define $N$ to be total number of derivative states, so here, $N = 6$, then 
\[ \{N^i(t) \in \mathbb{Z}^N_+\}_{i=1}^\infty \], counts components.

Define $S_i$ to be total component population, so here, $S_i = 3i$, then 
\[ v(t) := v_i(t)/S_i \in \mathbb{R}^N_+ \] is rescaled fluid approximation (indep. of $i$).
Functional law of large numbers

\[ \text{System} \overset{\text{def}}{=} \left( C \parallel \ldots \parallel C \right) \{\text{request, data}\} \left( S \parallel \ldots \parallel S \right) \]

\[ \text{System} \overset{\text{def}}{=} \left( C \parallel \ldots \parallel C \right) \{\text{request, data}\} \left( S \parallel \ldots \parallel S \right) \]

\[ \vdots \]

\[ \left\{ \text{System}_i \overset{\text{def}}{=} \left( C \parallel \ldots \parallel C \right) \{\text{request, data}\} \left( S \parallel \ldots \parallel S \right) \right\}_{i=1}^{\infty} \]

- Define \( N \) to be total number of derivative states, so here, \( N = 6 \), then \( \left\{ N_i(t) \in \mathbb{Z}_+^N \right\}_{i=1}^{\infty} \), counts components

- Define \( S_i \) to be total component population, so here, \( S_i = 3i \), then \( v(t) := v_i(t)/S_i \in \mathbb{R}^N_+ \) is rescaled fluid approximation (indep. of \( i \))
Functional law of large numbers

\[\text{System} \overset{\text{def}}{=} (C \parallel \ldots \parallel C) \begin{array}{c} \{ \text{request, data} \} \end{array} (S \parallel \ldots \parallel S) \]

\[\text{System} \overset{\text{def}}{=} (C \parallel \ldots \parallel C) \begin{array}{c} \{ \text{request, data} \} \end{array} (S \parallel \ldots \parallel S) \]

\[\vdots\]

\[\left\{ \text{System}_i \overset{\text{def}}{=} (C \parallel \ldots \parallel C) \begin{array}{c} \{ \text{request, data} \} \end{array} (S \parallel \ldots \parallel S) \right\}_{i=1}^{\infty}\]

- Define \(N\) to be total number of derivative states, so here, \(N = 6\), then \(\{N^i(t) \in \mathbb{Z}_+^N\}_{i=1}^{\infty}\), counts components
- Define \(S_i\) to be total component population, so here, \(S_i = 3i\), then \(v(t) := v^i(t)/S_i \in \mathbb{R}_+^N\) is rescaled fluid approximation (indep. of \(i\))
**Functional law of large numbers**

\[
\begin{align*}
\text{System} & \overset{\text{def}}{=} (C \parallel \ldots \parallel C) \{\text{request, data}\} (S \parallel \ldots \parallel S) \\
& \quad 10 \quad 5 \\
\text{System} & \overset{\text{def}}{=} (C \parallel \ldots \parallel C) \{\text{request, data}\} (S \parallel \ldots \parallel S) \\
& \quad 20 \quad 10 \\
\vdots \\
\left\{ \text{System}_i \overset{\text{def}}{=} (C \parallel \ldots \parallel C) \{\text{request, data}\} (S \parallel \ldots \parallel S) \right\}_{i=1}^\infty \\
& \quad 2i \\
\end{align*}
\]

- Define \( N \) to be total number of derivative states, so here, \( N = 6 \), then \( \{\mathbf{N}_i(t) \in \mathbb{Z}_+^N\}_{i=1}^\infty \), counts components.

- Define \( S_i \) to be total component population, so here, \( S_i = 3i \), then \( \mathbf{v}(t) := \mathbf{v}_i(t)/S_i \in \mathbb{R}_+^N \) is rescaled fluid approximation (indep. of \( i \)).
Functional law of large numbers

\[ \{N_i(t) \in \mathbb{Z}_+^N\}_{i=1}^\infty \] counts components

\[ v(t) := v_i(t)/S_i \in \mathbb{R}_+^N \] is rescaled fluid approximation (indep. of \( i \))

**Theorem**

*For all \( \delta > 0 \) and \( T > 0 \), if \( S_i \to \infty \) as \( i \to \infty \):

\[ P \left\{ \sup_{t \in [0, T]} \left\| N_i(t)/S_i - v(t) \right\| > \delta \right\} \to 0 \text{ as } i \to \infty \]

**Proof.**

Can use arguments of Kurtz for ‘density-dependent’ Markov chains [1].

A *convergence in probability* or *concentration of measure* result. Roughly, with high probability, error less than \( O(\text{component population size}) \).

FLLN — example

Rescaled component counts vs. Time, t

- 10 Client, 5 Server N(t)/S Server trace
- 50 Client, 25 Server N(t)/S Server trace
- 200 Client, 100 Server N(t)/S Server trace
- 400 Client, 200 Server N(t)/S Server trace
Introduction

Functional central limit theorem

Extracting information

Motivation

Result

Examples

Quantifying the difference

FLLN suggests approximation $\mathbf{N}^i(t) \approx S_i \times \mathbf{v}(t)$

- This is pretty coarse — replaces a *stochastic process* with a *completely deterministic quantity*

- Bounds on the rate of convergence require populations of the order of millions to tell you anything (at present)

- On the other hand, ODEs are *extremely tractable*

- We need a middle-ground\(^1\) — we might sacrifice a bit of tractability for some more accuracy

\(^1\)There may be many, here we discuss one!
Second-order refinement

We seek a second-order refinement to the fluid limit:

\[ N^i(t) \approx S_i \times v(t) + \sqrt{S_i} \times E(t) \]

- \( E(t) \) will be a stochastic process
- Root-scaling form suggested by standard CLT and, FCLTs such as that of Donsker

A possible approach is via another of Kurtz’s papers [1].

*Breaks down* due to non-smooth rates (\( \min(\cdot, \cdot) \) functions!)

Second-order refinement

We seek a second-order refinement to the fluid limit:

\[ N'(t) \approx S_i \times v(t) + \sqrt{S_i} \times E(t) \]

- \( E(t) \) will be a stochastic process
- Root-scaling form suggested by standard CLT and, FCLTs such as that of Donsker

A possible approach is via another of Kurtz’s papers [1].

*Breaks down* due to non-smooth rates (\( \min(\cdot, \cdot) \) functions!)


{rh, jb}@doc.ic.ac.uk
FCLT generalisation – setup

Must consider the $K$ transitions in aggregated state space individually.

\[ \{ I^k \in \mathbb{Z}^N \}_{k=1}^K \] are the \textit{jump vectors}

\[ \{ f^k(\cdot) \in \mathbb{R}_+^N \rightarrow \mathbb{R}_+ \}_{k=1}^K \] are the \textit{state-dependent transition rate functions}

So for client/server model, $K = 6$, and:

\[ I^1 = (-1, 1, 0, -1, 1, 0) \quad f^1(x) = \min(x_1, x_4) r_{req} \]
\[ I^2 = (0, -1, 1, 1, -1, 0) \quad f^2(x) = \min(x_2, x_5) r_{data} \]
\[ I^3 = (1, 0, -1, 0, 0, 0) \quad f^3(x) = x_3 r_{think} \]
\[ I^4 = (0, 0, 0, -1, 0, 1) \quad f^4(x) = x_4 r_{break} \]
\[ I^5 = (0, 0, 0, 0, -1, 1) \quad f^5(x) = x_5 r_{break} \]
\[ I^6 = (0, 0, 0, 1, 0, -1) \quad f^6(x) = x_6 r_{reset} \]
FCLT generalisation – setup

Must consider the $K$ transitions in aggregated state space individually.

- $\{l^k \in \mathbb{Z}^N\}_{k=1}^K$ are the *jump vectors*
- $\{f^k(\cdot) \in \mathbb{R}_+^N \to \mathbb{R}_+\}_{k=1}^K$ are the *state-dependent transition rate functions*

So for client/server model, $K = 6$, and:

- $l^1 = (-1, 1, 0, -1, 1, 0)$ \quad $f^1(x) = \min(x_1, x_4)r_{req}$
- $l^2 = (0, -1, 1, 1, -1, 0)$ \quad $f^2(x) = \min(x_2, x_5)r_{data}$
- $l^3 = (1, 0, -1, 0, 0, 0)$ \quad $f^3(x) = x_3r_{think}$
- $l^4 = (0, 0, 0, -1, 0, 1)$ \quad $f^4(x) = x_4r_{break}$
- $l^5 = (0, 0, 0, -1, 1, 0)$ \quad $f^5(x) = x_5r_{break}$
- $l^6 = (0, 0, 0, 1, 0, -1)$ \quad $f^6(x) = x_6r_{reset}$

{rh, jb}@doc.ic.ac.uk
We do require that the fluid limit is sufficiently well-behaved

- Let $T > 0$ and let $\hat{T}$ be the subset of $\{ t \in [0, T) \}$ for which the rate function, say, $f(\cdot)$ is not totally differentiable at the point $v(t)$. A condition of the theorem is that this set has Lebesgue measure zero.

- Then the Jacobian, $Df(v(t))$ exists $t$-almost everywhere.

- On $\hat{T}$, define it arbitrarily.
We do require that the fluid limit is sufficiently well-behaved

- Let $T > 0$ and let $\hat{T}$ be the subset of \{t \in [0, T)\} for which the rate function, say, $f(\cdot)$ is not totally differentiable at the point $v(t)$. A condition of the theorem is that this set has Lebesgue measure zero.

- Then the Jacobian, $Df(v(t))$ exists $t$-almost everywhere.

- On $\hat{T}$, define it arbitrarily.
We do require that the fluid limit is sufficiently well-behaved

- Let $T > 0$ and let $\hat{T}$ be the subset of $\{ t \in [0, \, T) \}$ for which the rate function, say, $f(\cdot)$ is *not* totally differentiable at the point $v(t)$. A condition of the theorem is that this set has *Lebesgue measure zero*.

- Then the Jacobian, $Df(v(t))$ exists $t$-almost everywhere.

- On $\hat{T}$, define it arbitrarily.
FCLT generalisation – pre-condition

We do require that the fluid limit is sufficiently well-behaved

- Let $T > 0$ and let $\hat{T}$ be the subset of $\{ t \in [0, T) \}$ for which the rate function, say, $f(\cdot)$ is not totally differentiable at the point $v(t)$. A condition of the theorem is that this set has Lebesgue measure zero.

- Then the Jacobian, $Df(v(t))$ exists $t$-almost everywhere.

- On $\hat{T}$, define it arbitrarily.
If $S_i \to \infty$ as $i \to \infty$, then:

\[
\frac{N^i(t)}{\sqrt{S_i}} - \sqrt{S_i}v(t) \Rightarrow E(t)
\]

weakly on $D_{\mathbb{R}^N_+}[0, T)$, equipped with the Skorohod $J_1$ topology\textsuperscript{a}. $E(t)$ is defined by:

\[
E(t) := \int_0^t Df(v(s)) \cdot E(s) \, ds + \sum_{k \in K} W^k \left( \int_0^t f^k(v(s)) \, ds \right) l^k
\]

where the $\{W^k(t)\}_{k=1}^K$ are $K$ mutually independent standard Wiener processes (aka Brownian motions)

\textsuperscript{a}Practically, this can be considered ‘convergence in distribution’
FCLT in action

Component counts vs. Time, t

100 Client, 50 Server second-order approximation
100 Client, 50 Server N(t)

{rh, jb}@doc.ic.ac.uk
FCLT in action

Introduction
Functional central limit theorem
Extracting information
Motivation
Result
Examples

-1.5
-1
-0.5
0
0.5
1
1.5

0
2
4
6
8
10

Root-scaled component counts
Time, t

50 Client, 25 Server root-scaled trace of divergence of CTMC from ODE (Clients)
50 Client, 25 Server E(t) trace (Clients)
50 Client, 25 Server root-scaled trace of divergence of CTMC from ODE (Clients)
50 Client, 25 Server E(t) trace (Clients)

-1.5
-1
-0.5
0
0.5
1
1.5

0
2
4
6
8
10

Root-scaled component counts
Time, t

100 Client, 50 Server root-scaled trace of divergence of CTMC from ODE (Clients)
100 Client, 50 Server E(t) trace (Clients)
100 Client, 50 Server root-scaled trace of divergence of CTMC from ODE (Clients)
100 Client, 50 Server E(t) trace (Clients)

-1.5
-1
-0.5
0
0.5
1
1.5

0
2
4
6
8
10

Root-scaled component counts
Time, t

500 Client, 250 Server root-scaled trace of divergence of CTMC from ODE (Clients)
500 Client, 250 Server E(t) trace (Clients)
500 Client, 250 Server root-scaled trace of divergence of CTMC from ODE (Clients)
500 Client, 250 Server E(t) trace (Clients)

{rh, jb}@doc.ic.ac.uk
\( \mathbb{E}(t) \) can also be represented as the solution of an (Itô) SDE:

\[
d\mathbb{E}(t) = \mu(\mathbb{E}(t), t) \, dt + \sigma(t) \, dW(t)
\]

where \( W(t) \) is a \( K \)-dimensional standard Wiener process and:

\[
\mu(x, t) := Df(v(t)) \cdot x
\]
\[
\sigma(t) := \left( l_i^j \times \sqrt{f_j(v(t))} \right)_{ij}
\]

An **Ornstein-Uhlenbeck** process with *time-dependent* coefficients
The probability density, $p(x, t)$ of $E(t)$ evolves as the solution to the following Fokker-Planck PDE:

$$\frac{\partial p}{\partial t} = - \sum_{i=1}^{N} \frac{\partial}{\partial x_i} \left[ g^1_i(x, t)p(x, t) \right] + \sum_{i=1}^{N} \sum_{j=1}^{N} g^2_{ij}(t) \frac{\partial^2}{\partial x_i \partial x_j} [p(x, t)]$$

where:

$$g^1_i(x, t) := \mu_i(x, t)$$

$$g^2_{ij}(t) := \frac{1}{2} \sum_{k=1}^{K} \sigma_{ik}(t)\sigma_{jk}(t)$$
**Moment ODEs**

- $\mathbf{E}(t)$ is Gaussian so marginals determined by first and second moments
- Easy to see $\mathbf{E}(t)$ has mean zero
- Using Itō’s lemma, we can extract ODEs for covariance matrix from SDE representation:

\[
\frac{d \text{Cov}[\mathbf{E}(t), \mathbf{E}(t)]}{dt} = \text{Cov}[\mathbf{E}(t), \mathbf{E}(t)] \cdot (Df(v(t)))^T \\
+ (Df(v(t))) \cdot (\text{Cov}[\mathbf{E}(t), \mathbf{E}(t)])^T \\
+ \sum_{k \in K} f^k(v(t)) I^k \cdot (I^k)^T
\]
Moment ODEs

- $E(t)$ is Gaussian so marginals determined by first and second moments
- Easy to see $E(t)$ has mean zero
- Using Itô’s lemma, we can extract ODEs for covariance matrix from SDE representation:

$$
\frac{d \text{Cov}[E(t), E(t)]}{dt} = \text{Cov}[E(t), E(t)] \cdot (Df(v(t)))^T \\
+ (Df(v(t))) \cdot (\text{Cov}[E(t), E(t)])^T \\
+ \sum_{k \in K} f^k(v(t))l^k \cdot (l^k)^T
$$
Moment ODEs

- \( \mathbf{E}(t) \) is Gaussian so marginals determined by first and second moments
- Easy to see \( \mathbf{E}(t) \) has mean zero
- Using Itô’s lemma, we can extract ODEs for covariance matrix from SDE representation:

\[
\frac{d \text{Cov}[\mathbf{E}(t), \mathbf{E}(t)]}{dt} = \text{Cov}[\mathbf{E}(t), \mathbf{E}(t)] \cdot (Df(v(t)))^T
+ (Df(v(t))) \cdot (\text{Cov}[\mathbf{E}(t), \mathbf{E}(t)])^T
+ \sum_{k \in K} f_k(v(t))I_k \cdot (I_k)^T
\]
Moment ODEs

- $\mathbf{E}(t)$ is Gaussian so marginals determined by first and second moments
- Easy to see $\mathbf{E}(t)$ has mean zero
- Using Itô’s lemma, we can extract ODEs for covariance matrix from SDE representation:

$$
\frac{d \text{Cov} [\mathbf{E}(t), \mathbf{E}(t)]}{dt} = \text{Cov} [\mathbf{E}(t), \mathbf{E}(t)] \cdot (Df(v(t)))^T \\
+ (Df(v(t))) \cdot (\text{Cov} [\mathbf{E}(t), \mathbf{E}(t)])^T \\
+ \sum_{k \in K} f^k(v(t)) I^k \cdot (I^k)^T
$$
Example

Approximating normal CDF
- 10 Client, 5 Server CDF
- 20 Client, 10 Server CDF
- 40 Client, 20 Server CDF
- 100 Client, 50 Server CDF
Conclusions and future work

- Tractable second-order approximation for massively-parallel models
- Extraction of approximate passage-time densities should be straightforward
- Hybrid analysis in this context?
Conclusions and future work

- Tractable second-order approximation for massively-parallel models

- Extraction of approximate passage-time densities should be straightforward

- Hybrid analysis in this context?
Conclusions and future work

- Tractable second-order approximation for massively-parallel models
- Extraction of approximate passage-time densities should be straightforward
- Hybrid analysis in this context?