

Parallel Tensor-Formatted Numerics for the Chemical Master Equation

Simon Etter

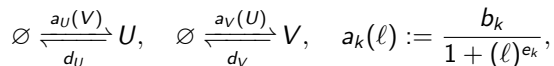
Master's Thesis at the
Seminar of Applied Mathematics, ETH Zurich.
Advised by Vladimir Kazeev, Robert Gantner and Prof. Christoph Schwab.
http://www.sam.math.ethz.ch/teaching/termproj/selected_theses

2 March 2015

Chemical Reaction Networks

Example: The toggle switch reaction network.

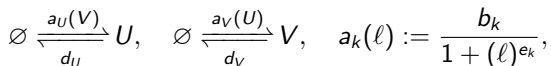
Two chemical species U , V produced and destroyed according to



Chemical Reaction Networks

Example: The toggle switch reaction network.

Two chemical species U , V produced and destroyed according to



Deterministic, Continuous Model:

Let $c(t, U)$, $c(t, V)$ denote concentrations of U , V at time t .

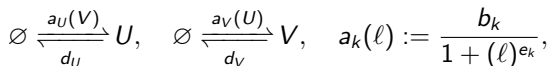
$$\begin{aligned} \frac{dc}{dt}(t, U) &= a_U(c(t, V)) - d_U c(t, U) \\ \frac{dc}{dt}(t, V) &= a_V(c(t, U)) - d_V c(t, V) \end{aligned}$$

- ▶ Two stable steady states $c(U) \gg c(V)$, $c(U) \ll c(V)$.
- ▶ Initial conditions fully determine eventual steady state.
- ▶ A biological bit!

Chemical Reaction Networks

Example: The toggle switch reaction network.

Two chemical species U , V produced and destroyed according to



Stochastic, Discrete Model:

Let $p(t, i_U, i_V)$ denote probability to count i_U, i_V copies of U, V at time t .

$$\begin{aligned} \frac{dp}{dt}(t, i_U, i_V) = & a_U(i_V) p(t, i_U - 1, i_V) - a_U(i_V) p(t, i_U, i_V) + \dots \\ & d_U(i_U + 1) p(t, i_U + 1, i_V) - d_U i_U p(t, i_U, i_V) + \dots \\ & \text{analogous terms in } V \end{aligned}$$

- ▶ Knowing the initial conditions only gives probability of which steady state will be reached.
- ▶ Even once a steady state is reached, flipping is possible.
- ▶ A biological die!

The above ODE is called the *chemical master equations* (CME).

The Curse of Dimensionality

Stochastic description of toggle switch model:

- ▶ In principle, infinitely many unknowns $p(t, i_U, i_V)$ with $i_U, i_V \in \mathbb{N}$.
- ▶ In practice, choose *maximal copy numbers* n_U, n_V and set

$$p(t, i_U, i_V) := 0 \quad \text{if} \quad i_U \geq n_U \text{ or } i_V \geq n_V.$$

- ▶ Still, state space is n^2 compared to 2 for deterministic model.

The Curse of Dimensionality:

General case of d species, uniform maximal copy numbers n :

$$n^d \text{ unknowns!}$$

Although “just” an ODE, CME hardly solvable for nontrivial $n, d!$

Stochastic Simulation Algorithm (SSA):

- ▶ Monte Carlo procedure to generate realisations of stochastic model.
- ▶ Simple & avoids curse of dimensionality.
- ▶ **But:** poor convergence.

Tensor Notation and Definitions

Combined Indices

Let D be a set.

- ▶ The symbol i_D represents the $\#D$ indices $i_k, k \in D$.
- ▶ We write $i_D \times i_{D'}$ to combine $i_D, i_{D'}$ into $i_{D \cup D'}$.

Tensor

A numerical array $a(i_D)$ with $\#D$ indices is called a *tensor*.

Mode Multiplication

Let $x(i_M \times i_K), y(i_K \times i_N)$ be tensors.

The *mode product* $z := xy$ is defined through

$$z(i_M \times i_N) := \sum_{i_K} x(i_M \times i_K) y(i_K \times i_N).$$

Example: The marginal distribution of U is given by $p 1_V$ with $1_V(i_V) := 1$.

Tensor Network Diagrams

Problem: Keeping track of the mode products can become difficult.

Solution: **Tensor Network Diagrams**

- ▶ Each tensor corresponds to a vertex.
- ▶ Each edge corresponds to a mode.
- ▶ Connecting edges implies mode multiplication.

Example:

$$a(i_1 \times i_2 \times i_3 \times i_4) = \begin{array}{c} 2 \quad 1 \\ \diagup \quad \diagdown \\ \bullet \\ \diagdown \quad \diagup \\ 3 \quad 4 \end{array}, \quad b(i_1 \times i_4 \times i_5) = \begin{array}{c} 1 \\ \diagdown \\ \bullet \\ \diagup \\ 4 \quad 5 \end{array}$$

$$ab = \begin{array}{c} 2 \\ \diagup \\ \bullet \\ \diagdown \\ 3 \end{array} \text{---} \begin{array}{c} \bullet \\ \diagup \\ 4 \end{array} \text{---} 5$$

Low-Rank Matrix Representation

Example: Throwing two fair dice.

The probability density function (PDF) is

$$\frac{1}{36} \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} = \frac{1}{36} \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} (1 \ 1 \ 1 \ 1 \ 1 \ 1) .$$

- ▶ Instead of 36 entries, store 12.
- ▶ Exploited property: independence of dice: $p(i_1 \times i_2) = p(i_1)p(i_2)$.
- ▶ Independence of species would reduce storage cost from n^d to $n \cdot d$!

But: independent species are not interesting.

Low-Rank Matrix Representation

Example: Throwing two fair dice until at least one does not show 1.
The probability density function (PDF) is

$$\begin{aligned} \frac{1}{35} \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} &= \frac{1}{35} \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} - \frac{1}{35} \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \\ &= \frac{1}{35} \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} (1 \ 1 \ 1 \ 1 \ 1 \ 1) - \frac{1}{35} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} (1 \ 0 \ 0 \ 0 \ 0 \ 0) \end{aligned}$$

- ▶ We can write PDF as a short sum of independent PDFs!
- ▶ Number of terms is called *rank*.
- ▶ Trade accuracy vs. effort by dropping terms in the sum (*truncation*).

Quantization

Low-rank representation techniques can also be applied to 1D functions!

- ▶ Consider the vector

$$(2 \ 1 \ 3 \mid 0 \ 0 \ 0 \mid 4 \ 2 \ 6) .$$

- ▶ Reshape to matrix, and apply low-rank separation:

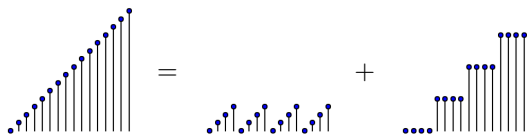
$$\begin{pmatrix} 2 & 1 & 3 \\ 0 & 0 & 0 \\ 4 & 2 & 6 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 2 \end{pmatrix} (2 \ 1 \ 3) .$$

- ▶ Quantization allows to exploit recurring patterns!

Quantization

Most common type of pattern: smoothness.

- ▶ Constant function c : rank 1.
- ▶ Linear function $ax + b$: rank 2.



- ▶ General polynomial of order p : rank $\leq p + 1$.
- ▶ Exponential function $\exp(\omega x)$: rank 1.
- ▶ Trigonometric functions $\sin(\omega x)$, $\cos(\omega x)$: rank 2.

Tensor Network Formats

- ▶ Tensors can be reshaped to matrices
→ matrix separation techniques generalize to tensors.
- ▶ Different matricizations for $d > 2$ → new degree of freedom.

Consider family of tensors $a(\alpha, i_D)$, $D := \{1, \dots, d\}$, parametrized by α .

Tensor Network Formats

- ▶ Tensors can be reshaped to matrices
→ matrix separation techniques generalize to tensors.
- ▶ Different matricizations for $d > 2$ → new degree of freedom.

Consider family of tensors $a(\alpha, i_D)$, $D := \{1, \dots, d\}$, parametrized by α .

Tensor Train (TT) Format

- ▶ Separate the first mode:

$$a(\alpha, i_D) \approx \sum_{\alpha_1=1}^{r_1} u_1(\alpha, i_1, \alpha_1) v(\alpha_1, i_{D \setminus \{1\}}).$$

- ▶ Recurse for v .

Tensor Network Formats

- ▶ Tensors can be reshaped to matrices
→ matrix separation techniques generalize to tensors.
- ▶ Different matricizations for $d > 2$ → new degree of freedom.

Consider family of tensors $a(\alpha, i_D)$, $D := \{1, \dots, d\}$, parametrized by α .

Hierarchical Tucker Representation (HTR)

- ▶ Split D into $L = \{1, \dots, k-1\}$, $R := \{k, \dots, d\}$.
- ▶ Separate R :

$$a(\alpha, i_D) \approx \sum_{\alpha_R=1}^{r_R} v(\alpha, i_L, \alpha_R) u_R(\alpha_R, i_R).$$

- ▶ Separate L :

$$a(\alpha, i_D) \approx \sum_{\alpha_L=1}^{r_L} \sum_{\alpha_R=1}^{r_R} c(\alpha, \alpha_L, \alpha_R) u_L(\alpha_L, i_L) u_R(\alpha_R, i_R).$$

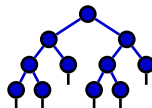
- ▶ Recurse for u_L , u_R .

Tensor Network Formats

Tensor network diagram of resulting representations:



TT format



HTR

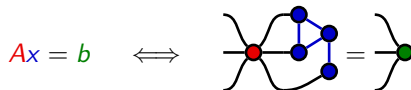
Terminology and notation:

- ▶ Vertex v : a particular position in the network.
- ▶ Vertex tensor x_v : the tensor at position v .
- ▶ Vertex set V .
- ▶ Set of rank modes (connected edges) E .
- ▶ Set of free modes (dangling edges) D .

Tensor-Network Structured Linear Systems

- ▶ We use an implicit time-stepping scheme to solve the CME.
- ▶ Solving LSEs is only feasible if carried out directly in TN format.

Problem:

$$Ax = b \iff \text{TN Diagram} = \text{TN Diagram}$$


An LSE becomes a

- ▶ high-dimensional
- ▶ multi-linear
- ▶ overdetermined

equation in terms of the vertex tensors x_v .

How to solve this?

Tensor-Network Structured Linear Systems

- ▶ Assume A is symmetric and positive definite.
- ▶ Solving $Ax = b$ is equivalent to the optimization problem

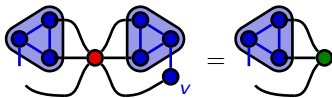
$$\arg \min_x \|x - A^{-1}b\|_A^2 = \arg \min_x x^T A x - 2x^T b.$$

- ▶ Idea: optimize only over one vertex tensor x_v at a time!
 - ▶ Define environment tensor $U_v(x) := \prod_{u \in V \setminus \{v\}} x_u$
 - ▶ The above global problem becomes the *local problem*

$$\arg \min_{x_v} x_v^T U_v^T(x) A U_v(x) x_v - 2x_v^T U_v^T(x) b$$

which yields the *local LSE*

$$U_v^T(x) A U_v(x) x_v = U_v^T(x) b.$$



The Alternating Least Squares (ALS) Algorithm

Algorithm 1 Alternating Least Squares (ALS)

- 1: **repeat**
 - 2: **for** vertex $v \in V$ **do**
 - 3: Solve local LSE at v
 - 4: **end for**
 - 5: **until** convergence
-

Important technicalities:

- ▶ Condition number $\kappa(U_v^T(x) A U_v(x))$ must be reasonably small.
In particular, $U_v(x)$ must have full rank.
- ▶ Assembly of the local matrix and right-hand side

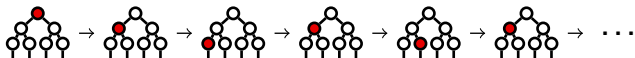
$$U_v^T(x) A U_v(x), \quad U_v^T(x) b$$

must be efficient.

The HTR ALS Algorithm

Both problems can be solved if

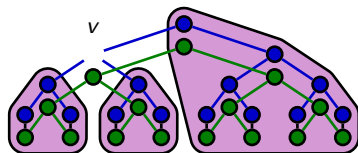
- ▶ x , A and b are represented in TT or HTR.
- ▶ we traverse the network *mole-like*:



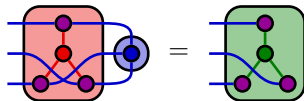
The HTR ALS Algorithm

Key to assembling local LSE: **contracted subtrees**

Example: $U_v^T(x)b$



The local LSE in terms of contracted subtrees:



Obtain contracted subtrees cheaply through recursion and memoization:

$$(x|b)(v) \begin{matrix} \vdots \\ \bullet \end{matrix} = \begin{matrix} \vdots \\ x_v \bullet \end{matrix} \begin{matrix} \vdots \\ \bullet \end{matrix} \begin{matrix} \vdots \\ b_v \end{matrix} \begin{matrix} \vdots \\ \bullet \end{matrix} \\ (x|b)(v_L) \bullet \bullet (x|b)(v_R)$$

ALS-Type Algorithms

- ▶ The ALS algorithm cannot adapt the ranks!
- ▶ Performance and final accuracy strongly depend on quality of the a priori guessed ranks.

There are multiple way to achieve rank-adaptivity.

Algorithm 2 Density Matrix Renormalization Group (DMRG)

```
1: repeat
2:   for edge  $(u, v) \in E$  do
3:     Form supercore  $w := x_u x_v$ .
4:     Solve local LSE  $U_{u,v}^T(x) A U_{u,v}(x) w = U_{u,v}^T(x) b$ .
5:     Split  $x_u x_v := w$ .
6:   end for
7: until convergence
```

- ▶ Can speed up convergence due to increased size of local LSE.
- ▶ Used for > 20 years in computational quantum physics.
- ▶ **But:** only efficient for TT format.
Problem: $\text{numel}(w)$ ranges up to r^4 for HTR, compared to $n^2 r^2$ for TT.

ALS-Type Algorithms

- ▶ The ALS algorithm cannot adapt the ranks!
- ▶ Performance and final accuracy strongly depend on quality of the a priori guessed ranks.

There are multiple way to achieve rank-adaptivity.

Algorithm 3 ALS + Steepest Descent (ALS(SD))

- 1: **repeat**
 - 2: Compute residual approximation $z \approx b - Ax$
 - 3: Update $x := x + z$ // *increases ranks*
 - 4: Run a single ALS iteration
 - 5: Truncate x // *decreases ranks*
 - 6: **until** convergence
-

- ▶ Avoids increasing the local problems, therefore also suitable for HTR.
- ▶ Provable geometric convergence.

Application to the CME

Already serial MATLAB implementation of TT + DMRG ansatz outperformed SSA running on 1500 cores!

But:

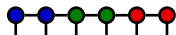
- ▶ SSA is straightforward to parallelize.
- ▶ Only few parallelization attempts for tensor network computations.

Loop structure in our problem:

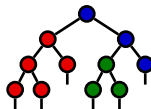
time \rightarrow solver iterations \rightarrow **vertices** \rightarrow local problem

Parallelize over the vertices!

- ▶ Assign each vertex to a process.
- ▶ Let this process store vertex tensors and execute local operations.



TT format



HTR

Parallel Tensor Network Computations

The TT format is not parallelizable over the vertices!

The problem:

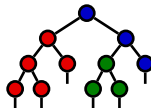
- ▶ Any non-trivial operation requires gathering information from all parts of the network.
E.g. in ALS algorithm, we need to compute the contracted subtrees.
- ▶ Information gathering is only efficient if messages are passed on from one vertex to its neighbour like the baton in a relay race.
- ▶ The longest distance in the TT network is $\mathcal{O}(d)$!

In contrast:

- ▶ Longest distance in HTR: $\mathcal{O}(\log(d))$.
- ▶ All basic HTR algorithms (addition, dot product, truncation) achieve the optimal parallel runtime out of the box.



TT format



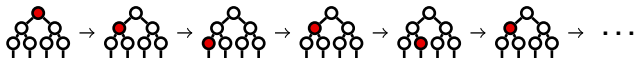
HTR

Parallelizing the HTR ALS Algorithm

The ALS algorithm is not parallelizable over the vertices!

The problem:

- ▶ Local LSE at v depends on all x_u , $u \neq v$, through $U_v(x)$.
- ▶ Updating a vertex tensor invalidates local LSEs at other vertices.



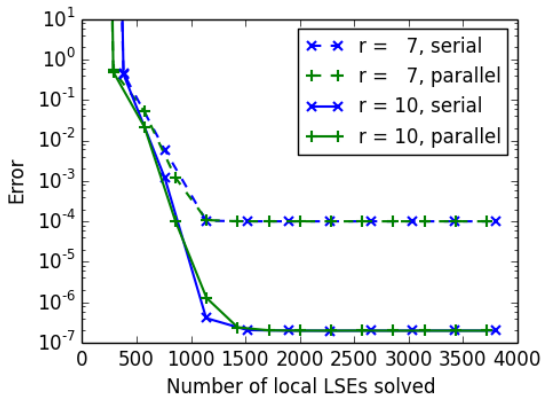
The solution:

- ▶ Just ignore this dependence temporarily.
- ▶ Easy because local LSEs are assembled from cached contracted subtrees.



Case Study: The 16D Poisson Equation

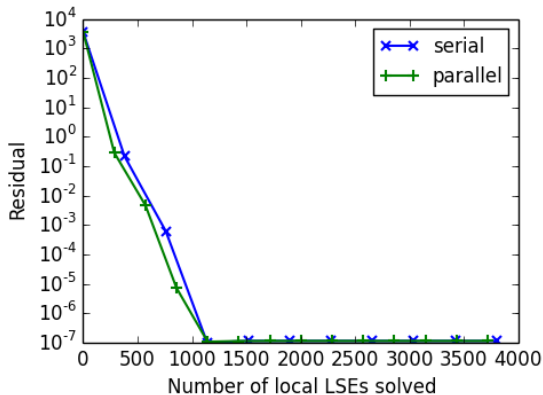
We modified the algorithm. Does this have an impact on convergence?



Convergence of ALS.

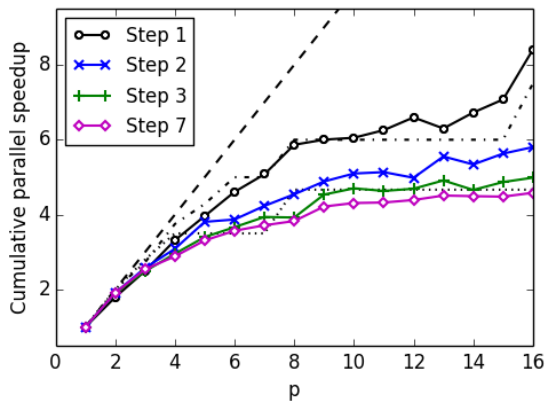
Case Study: The 16D Poisson Equation

We modified the algorithm. Does this have an impact on convergence?



Convergence of ALS(SD).

Case Study: The 16D Poisson Equation



Strong scaling of parallel ALS(SD) solver.

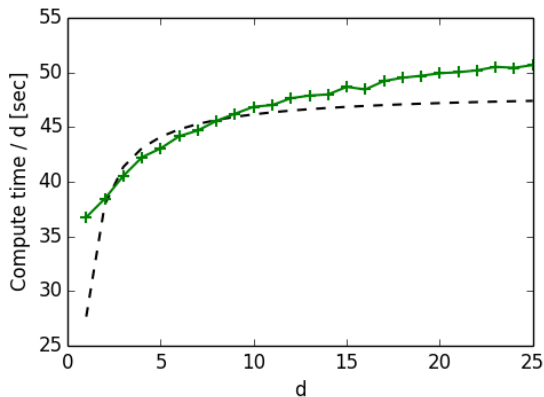
Independent Birth-Death Processes

Species: X_1, \dots, X_d

Reactions: $\emptyset \xrightleftharpoons[d_k]{b_k} X_k$

Maximal copy numbers: $n_k = 4096$ for all $k = 1, \dots, d$

Independent Birth-Death Processes



Compute time per dimension.

We solve a problem in 10^{90} unknowns in 20 minutes!

Estimated number of atoms in universe: 10^{80} .

Vertex Distributions

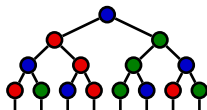
How do we assign vertices to processes?

Round-robin vertex distribution

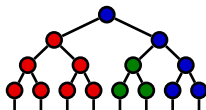
Enumerate vertices in breadth-first order and deal in round-robin manner.

Optimized vertex distribution

Try to assign neighbouring vertices to the same process.

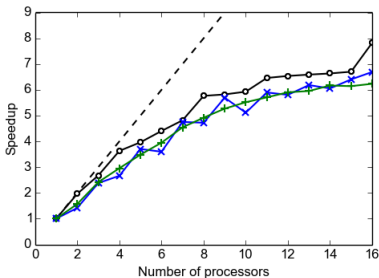


Round-robin

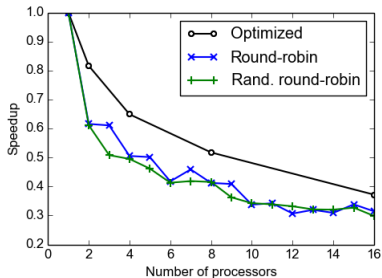


Optimized

Independent Birth-Death Processes

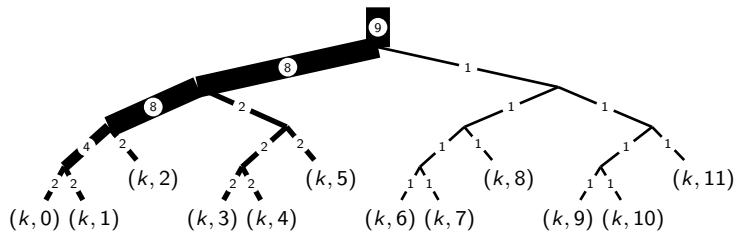


Strong scaling ($d = 16$)



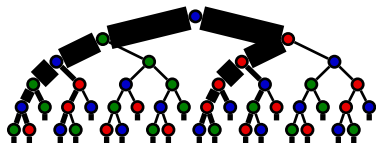
Weak scaling ($d = p$)

Independent Birth-Death Processes

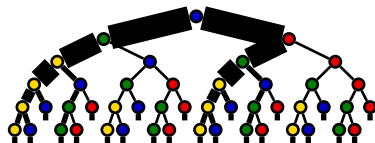


Virtual ranks of a single species.

Independent Birth-Death Processes



$p = 3$



$p = 4$

Round-robin vertex distribution.

Toggle Switch

Species: U, V

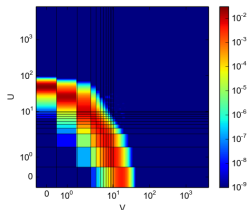
Reactions: $\emptyset \xrightleftharpoons[d_U]{a_U(V)} U, \quad \emptyset \xrightleftharpoons[d_V]{a_V(U)} V,$

$$a_k(\ell) := \frac{b_k}{1 + (\ell)^{e_k}},$$

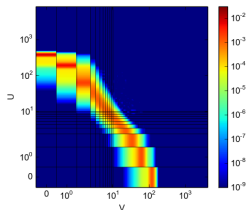
Initial conditions: $p(t = 0, i_U \times i_V) = \delta((i_U = 0) \times (i_V = 0))$

Maximal copy numbers: $n_U = 8192, n_V = 4096$

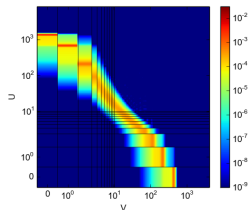
Toggle Switch



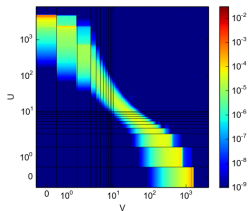
$t = 0.01$



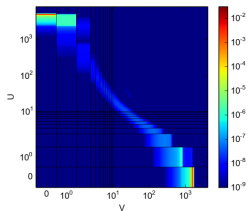
$t = 0.08$



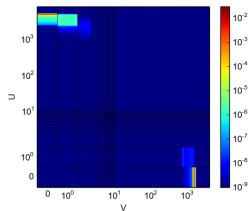
$t = 0.32$



$t = 2.56$



$t = 10.24$



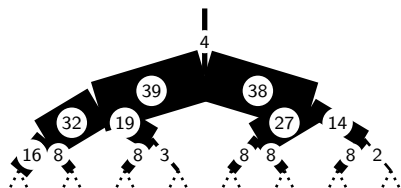
$t = 100$

Toggle Switch

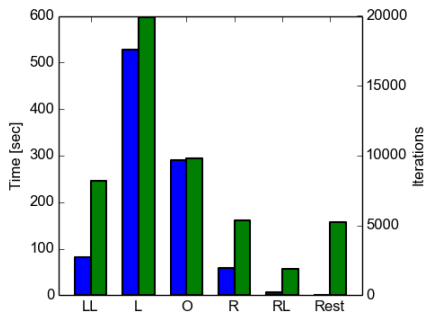
	$p = 1$	$p = 2$	$p = 4$
Default		1.09x	0.89x
Round-robin	982 sec.	1.12x	1.09x

Serial runtime for $p = 1$ and parallel speedup for $p > 1$.

Toggle Switch



Ranks of solution.



Local LSE statistics.

Parallel Tensor-Formatted Numerics for the Chemical Master Equation

Simon Etter

Master's Thesis at the
Seminar of Applied Mathematics, ETH Zurich.
Advised by Vladimir Kazeev, Robert Gantner and Prof. Christoph Schwab.
http://www.sam.math.ethz.ch/teaching/termproj/selected_theses

2 March 2015