# On least fixed points of systems of positive polynomials

**Javier Esparza**[a] **Andreas Gaiser**[a] **Stefan Kiefer**[a]

[a] Faculty of Computer Science, Technische Universität München, Germany
esparza@in.tum.de,gaiser@model.in.tum.de,kiefer@in.tum.de

We present an algorithm to compute the least nonnegative solution of a *system of probabilistic polynomials (SPrP)*, a system of equations of the form

$$X_1 = f_1(X_1, ..., X_n) \quad ... \quad X_n = f_n(X_1, ..., X_n)$$

where, for every $i \in \{1, ..., n\}$, $f_i$ is a polynomial over $X_1, \ldots, X_n$ with positive rational coefficients that *add up to 1*. The solutions of a SPrP are the fixed points of the function $f : \mathbb{R}^n \to \mathbb{R}^n$ with $f = (f_1, ..., f_n)$. For example, $X_1 = \frac{1}{2}X_1X_2 + \frac{1}{2}, X_2 = \frac{1}{4}X_2X_2 + \frac{1}{4}X_1 + \frac{1}{2}$ is a SPrP with $f(X_1, X_2) = (\frac{1}{2}X_1X_2 + \frac{1}{2}, \frac{1}{4}X_2X_2 + \frac{1}{4}X_1 + \frac{1}{2})$. Obviously, $\overline{1} = (1, ..., 1)$ is a solution for every SPrP. By Kleene's theorem, every SPrP has a least nonnegative solution (called just least solution in what follows).

SPrPs are important in different areas of the theory of stochastic processes and computational models. A fundamental result of the theory of branching processes, with numerous applications in physics and biology (see e.g. [6, 1]), states that extinction probabilities of species are equal to the least solution of a SPrP. The same result has been recently shown for the probability of termination of certain probabilistic recursive programs ([5, 4]). The consistency of stochastic context-free grammars, a problem of interest in statistical natural language processing, also reduces to checking whether the least solution of a SPrP equals $\overline{1}$ (see e.g. [8]).

We fix an SPrP with function $f$ and denote its least solution by $\mu_f$. The following two problems are motivated by the applications above: (1) decide whether $\mu_f = \overline{1}$, and (2) given $\epsilon > 0$ compute $lb, ub$ such that $lb \leq \mu_f \leq ub$ and $ub - lb \leq \overline{\epsilon}$.

Etessami and Yannakakis show in [5] that Problem (1) can be solved in polynomial time using exact Linear Programming; however, it is known that in practice this method is inefficient (over 30 minutes in average for Maple's exact Simplex on random examples with 200 variables). The same experiments show that inexact Linear Programming solvers based on floating point arithmetic easily lead to false results because of severe numerical problems (10 variables and coefficients from the set $\{0.01, 0.5, 0.49\}$ are enough to break lpsolve). Concerning Problem (2), lower bounds for $\mu_f$ can be computed using Newton's method to approximate a root of the function $f(\overline{X}) - \overline{X}$ (the roots obviously correspond to the fixed points of $f$). It is shown in [5] and [3, 7] that if Newton's method starts at a *prefixed point* $p$ of $f$ (a point $p \in [0, 1]^n$ such that $f(p) \geq p$), for example $p = \overline{0}$, and exact arithmetic is used, then the method converges to $\mu_f$ and every approximant is again a prefixed point. Again, experiments show that naively using exact arithmetic is very inefficient. In [9] a tool is presented that uses Newton's method with floating point arithmetic, but for the same 7-variable example as above the tool wrongly indicates that $\mu_f = \overline{1}$. Finally, the computation of *upper* bounds for $\mu_f$ does not seem to have been considered so far.

The poster presents our new algorithm based on Newton's method to simultaneously solve Problems (1) and (2). To the best of our knowledge, it is the first algorithm usable in practice that never returns a wrong result and provides not only a lower but also an upper bound for the solution.

The algorithm, shown in Figure 1, handles *normalized* SPrPs where $f_i$ is of quadratical degree and the Jacobian matrix $f'(\overline{1})$ is an irreducible matrix such that $Id - f'(\overline{1})$ is nonsingular. It is essentially sufficient to consider this class of SPrPs, since we can transform an arbitrary SPrP into a collection of SPrPs of this form, although here some problems still have to be solved. The algorithm is based on the following theorem, which builds on a result of [5] and Perron-Frobenius theory. (By "$\prec$" resp. "$\succ$" we denote smaller resp. greater in every component.)

**Theorem 1.** *If $f$ is normalized and $\mu_f = \overline{1}$, then there is an $i \in \mathbb{N}$ such that $f'(\overline{1})(\overline{1} - \nu^{(i)}) \prec (\overline{1} - \nu^{(i)})$ where $\nu^{(i)}$ is the i-th Newton iterant. If $\mu_f < \overline{1}$, then the same property holds with "$\prec$" replaced by "$\succ$".*

We have proved that, with exact arithmetic, the index $i$ has polynomial size in the size of the SPrP. The proof relies on recent results about the convergence speed of Newton's method [3, 7] and on a gap theorem by Dedieu [2].

To overcome the inefficiency of exact arithmetic, we compute each Newton iterant using inexact arithmetic of variable precision, and then check whether it is a prefixed point. If so, rounding errors do not affect correctness, and the computation can proceed. If not, the computation of the iterant is repeated with increased precision; in our prototype implementation we increment the bitsize of floating point numbers. In order to obtain upper bounds on $\mu_f$, if our method detects that $\mu_f \neq \overline{1}$ , we compute a *postfixed point*, i.e. a $p \in [0,1]^n$ with $p \geq f(p)$, which can be used as a first upper bound for $\mu_f$. We can show that the sequence $p, f(p), f(f(p)), \ldots$ converges to $\mu_f$ from above with linear convergence rate.

$\nu^{(0)} \Leftarrow \overline{0}$
$i \Leftarrow 0$
**while** true **do**
  $\nu^{(i+1)} \Leftarrow$ `NewtonNumeric`$(f, \nu^{(i)})$
  **if** $\nu^{(i+1)}$ no prefixed point or $\nu^{(i+1)} \not\succ \nu^{(i)}$ **then**
    $\nu^{(i+1)} \Leftarrow \nu^{(i)}$
    increase precision of `NewtonNumeric`
  **else if** $f'(\overline{1}) \cdot (\overline{1} - \nu^{(i+1)}) \succ \overline{1} - \nu^{(i+1)}$ **then**
    calculate postfixed point $p$ using $\nu^{(i+1)}$
    compute $\nu^{(i+1+j)}$ and $f^k(p)$ for increasing $j, k$
      until $f^k(p) - \nu^{(i+1+j)} \leq \overline{\epsilon}$
    return $(\nu^{(i+1+j)}, f^k(p))$
  **else if** $f'(\overline{1}) \cdot (\overline{1} - \nu^{(i+1)}) \prec \overline{1} - \nu^{(i+1)}$ **then**
    return $(\overline{1}, \overline{1})$
  **else**
    $i \Leftarrow i + 1$
  **end if**
**end while**

Figure 1: The algorithm returns $(lb, ub)$ with $lb \leq \mu_f \leq ub$.

In order to compute the elements of the sequence we again use inexact arithmetic and increase precision when needed. In this way we obtain a sequence of increasingly better upper bounds for $\mu_f$ which, together with the lower bounds provided by the Newton iterants, yield a solution to Problem (2).

The price to pay for a correctness guarantee with inexact arithmetic is the possibility that the algorithm does not terminate. This can only happen when $\mu_f = \overline{1}$. While in our experiments the algorithm always terminates, we are currently searching for easy conditions that guarantee termination. Our claim that ours is the first algorithm usable in practice that never returns a wrong result is supported by a speed-up factor of 15 on random systems with 200 variables over Maple's exact Simplex. Recall also that Linear Programming only solves problem (1), while our algorithm simultaneously solves (1) and (2).

# References

[1] Krishna B. Athreya and Peter E. Ney. *Branching Processes*. Springer-Verlag, 1972.

[2] Jean-Pierre Dedieu. Estimations for the separation number of a polynomial system. *Journal of Symbolic Computation*, 24(6):683 – 693, 1997.

[3] Javier Esparza, Stefan Kiefer, and Michael Luttenberger. Convergence thresholds of Newton's method for monotone polynomial equations. In *Proceedings of the 25th International Symposium on Theoretical Aspects of Computer Science*, pages 289–300, 2008.

[4] Javier Esparza, Antonín Kučera, and Richard Mayr. Model checking probabilistic pushdown automata. In *LICS 2004*. IEEE Computer Society, 2004.

[5] Kousha Etessami and Mikhalis Yannakakis. Recursive markov chains, stochastic grammars, and monotone systems of nonlinear equations. *Journal of the ACM*, 56(1):1–66, 2009.

[6] Theodore E. Harris. *The theory of branching processes*. Springer-Verlag, Berlin, 1963.

[7] Stefan Kiefer, Michael Luttenberger, and Javier Esparza. On the convergence of Newton's method for monotone systems of polynomial equations. In *Proceedings of the 39th ACM Symposium on Theory of Computing*, pages 217–226. ACM, 2007.

[8] Christopher D. Manning and Hinrich Schuetze. *Foundations of Statistical Natural Language Processing.* The MIT Press, June 1999.

[9] Dominik Wojtczak and Kousha Etessami. Premo : An analyzer for probabilistic recursive models. In *TACAS*, volume 4424 of *Lecture Notes in Computer Science*, pages 66–71. Springer, 2007.