Computing Least Fixed Points of Probabilistic Systems of Polynomials

STACS 2010 / GPMFV 2010

Javier Esparza[1], Andreas Gaiser[1], Stefan Kiefer[2]

[1]Technische Universität München

[2]Oxford University

September 28, 2010

Outline

1. What is a Probabilistic System of Polynomials (PSP)?
2. Why studying PSPs?
3. Algorithms
   - An Exact Algorithm for Consistency
   - An Exact Algorithm for Lower and Upper Bounds of LFP(f)
4. Case study: PSPs in Physics

Definition of a PSP

- We investigate polynomial equation systems

$$\begin{aligned} X_1 &= f_1(X_1, \ldots, X_n) \\ &\ldots \\ X_n &= f_n(X_1, \ldots, X_n) \end{aligned}$$

  where the f_i are polynomials over $X_1, \ldots, X_n$.

- Important restriction: The coefficients of each f_i are nonnegative and sum up to 1.
- The vector $f := (f_1, \ldots, f_n)^\top$ is called a probabilistic system of polynomials (PSP).

An Example

2-dimensional PSP

$$\begin{aligned} X_1 &= \frac{4}{5}X_1X_2 + \frac{1}{5} \\ X_2 &= \frac{2}{5}X_1X_1 + \frac{1}{10}X_2 + \frac{1}{2} \end{aligned}$$

leads to the PSP $f : \mathbb{R}^2 \to \mathbb{R}^2$ with

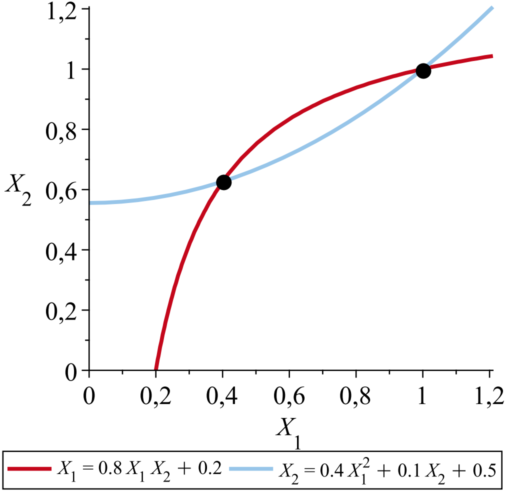
$$\begin{aligned} f_1(X_1, X_2) &= \frac{4}{5}X_1X_2 + \frac{1}{5} \\ f_2(X_1, X_2) &= \frac{2}{5}X_1X_1 + \frac{1}{10}X_2 + \frac{1}{2}. \end{aligned}$$

Fixed Points

- For every PSP, $\overline{1} = (1, \ldots, 1)$ is a fixed point.
- We are interested in the least nonnegative fixed point (LFP) of f, where we mean "least" with respect to the order "$\leq$" defined componentwise.

An Example

$$
\begin{aligned}
X_1 &= \frac{4}{5} X_1 X_2 + \frac{1}{5} \\
X_2 &= \frac{2}{5} X_1 X_1 + \frac{1}{10} X_2 + \frac{1}{2}
\end{aligned}
$$



Problems

Problem 1 (Consistency problem)

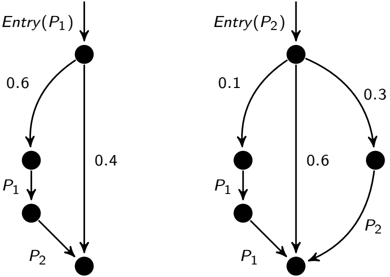
Given a PSP f, decide whether $\mathrm{LFP}(f) = \overline{1}$.

Problem 2 (Computing Lower and Upper bounds)

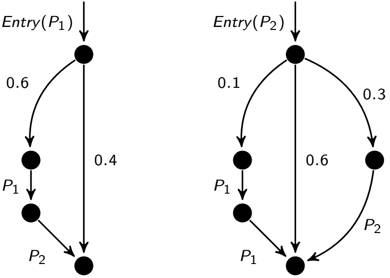
Given a PSP f, for a given $\epsilon > 0$, compute $\mathbf{lb}, \mathbf{ub}$ such that $\mathbf{lb} \leq \mathrm{LFP}(f) \leq \mathbf{ub}$ with $\mathbf{ub} - \mathbf{lb} \leq \overline{\epsilon}$.

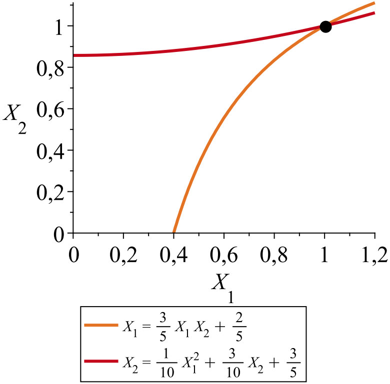
- If $\mathrm{LFP}(f) = \overline{1}$ then f is consistent, otherwise inconsistent.
- Why are those problems interesting?

Termination probability of probabilistic recursive programs



- Probabilistic flow graphs of two simple procedures P_1 and P_2.
- Termination probability for P_i = Probability that a call $P_i()$ eventually terminates.





Corresponding equation system

$$X_1 = 0.6 X_1 X_2 + 0.4$$
$$X_2 = 0.1 X_1 X_1 + 0.3 X_2 + 0.6$$

- Termination probabilities = LFP of the corresponding PSP.
- Here: $\mathrm{LFP}(f) = (1, 1)$
  $\Rightarrow$ Termination with probability 1.
- Termination with prob. 1 depends not only on the program structure.

Applications of PSPs: Multi-type branching processes

$$X_1 \xhookrightarrow{0.6} \{X_1, X_2\} \qquad X_2 \xhookrightarrow{0.1} \{X_1, X_1\}$$

$$X_1 \xhookrightarrow{0.4} \{\} \qquad X_2 \xhookrightarrow{0.3} \{X_2\}$$

$$X_2 \xhookrightarrow{0.6} \{\}$$

Applications in various areas:

- Verification of probabilistic programs: Termination probability of Probabilistic Pushdown Systems and Recursive Markov Chains
- Biology: Reproduction and extinction of species
- Natural Language processing: Stochastic context-free grammars
- Physics: See case study at the end

1. What is a Probabilistic System of Polynomials (PSP)?
2. Why studying PSPs?
3. Algorithms
   - An Exact Algorithm for Consistency
   - An Exact Algorithm for Lower and Upper Bounds of LFP(f)
4. Case study: PSPs in Physics

Consistency can be decided in weakly polynomial time.

Problem 1 (Consistency problem)

Given a PSP f, decide whether $\mathrm{LFP}(f) = \overline{1}$.

- This can be decided [Etessami/Yannakakis, 2009] in (weakly) polynomial time by checking whether the following LP problem has a solution:

$$f'(\overline{1})\mathbf{x} \geq (1 + 2^{-c_f})\mathbf{x} \text{ with } \mathbf{x} \geq \overline{0} \text{ and } \sum_{i=1}^{n} \mathbf{x}_i = 1.$$

- Problem: c_f, although polynomial in f, can be very large...

An almost consistent family of PSPs

- a family of inconsistent (but "almost consistent") PSPs:

$$
\begin{aligned}
X_1 &= 0.5X_1^2 + 0.1X_n^2 + 0.4 \\
X_2 &= 0.01X_1^2 + 0.5X_2 + 0.49 \\
& \ldots \\
X_n &= 0.01X_{n-1}^2 + 0.5X_n + 0.49.
\end{aligned}
$$

- Inexact LP-solvers cannot handle the instances with $n > 10$.
- Experiments with Maple's exact Simplex package

	$n = 100$	$n = 200$	$n = 400$	$n = 600$	$n = 1000$
Exact LP	2 sec	8 sec	67 sec	208 sec	> 2h

Our new consistency-check algorithm

- Algorithm for consistency of strongly connected PSPs:
  1. Solve the system $(Id - f'(\overline{1}))\mathbf{v} = \overline{0}$.
  2. If a solution $\mathbf{v} \neq \overline{0}$ exists, return true iff $\mathbf{v} \succ \overline{0}$ or $\mathbf{v} \prec \overline{0}$.
  3. Else find the unique solution of the system $(Id - f'(\overline{1}))\mathbf{v} = \overline{1}$.
  4. If $\mathbf{v} \geq \overline{1}$ and $f'(\overline{1})\mathbf{v} < \mathbf{v}$ return true, else return false.

$\Rightarrow$ It suffices to solve two linear equation systems.

- We can generalize the algorithm easily to arbitrary PSPs.

Assessment

- No need for invoking Linear Programming
- The algorithm is strongly polynomial and very easy to implement.
- Comparison on the "almost consistent" family:

	$n = 100$	$n = 200$	$n = 400$	$n = 600$	$n = 1000$
Exact LP	2 sec	8 sec	67 sec	208 sec	> 2h
New alg.	1 sec	1 sec	4 sec	10 sec	29 sec

1. What is a Probabilistic System of Polynomials (PSP)?
2. Why studying PSPs?
3. Algorithms
   - An Exact Algorithm for Consistency
   - An Exact Algorithm for Lower and Upper Bounds of LFP(f)
4. Case study: PSPs in Physics

Problem 2 (Computing Lower and Upper Bounds)

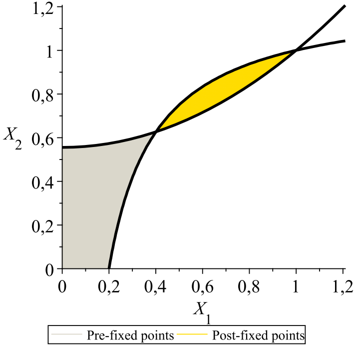
Given a PSP f, for a given $\epsilon > 0$, compute $\mathbf{lb}, \mathbf{ub}$ such that

$$\mathbf{lb} \leq \mathrm{LFP}(f) \leq \mathbf{ub} \text{ with } \mathbf{ub} - \mathbf{lb} \leq \overline{\epsilon}.$$

Pre-fixed and Post-fixed Points

- $\mathbf{x} \in \mathbb{R}^n$ is a pre-fixed (post-fixed) point if $f(\mathbf{x}) \geq \mathbf{x}$ ($f(\mathbf{x}) \leq \mathbf{x}$).
- If $\mathbf{x}$ is strictly greater than $\mathbf{y}$ in all components we write $\mathbf{x} \succ \mathbf{y}$.
- $\mathbf{x} \in \mathbb{R}^n$ is a strict pre-fixed (post-fixed) point if $f(\mathbf{x}) \succ \mathbf{x}$ ($f(\mathbf{x}) \prec \mathbf{x}$).
- Pre-fixed points are lower bounds,

  post-fixed points are upper bounds for $\mathrm{LFP}(f)$.



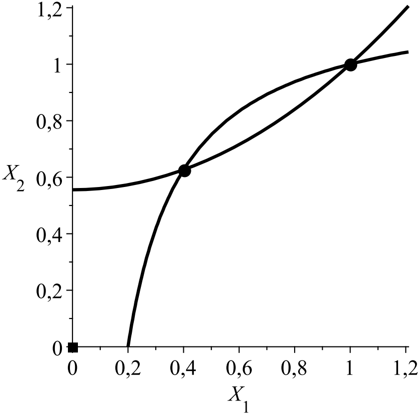
Obtaining lower bounds: Newton's method

- $\overline{0}, f(\overline{0}), f^2(\overline{0}), \ldots$ are all pre-fixed points.
  The sequence converges (in general slowly) to $\mathrm{LFP}(f)$.

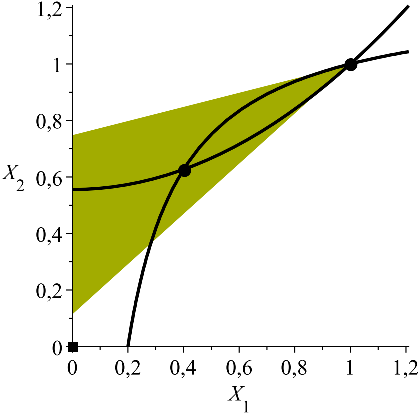
$\Rightarrow$ Apply Newton's method for finding zeros of the map $f(\mathbf{X}) - \mathbf{X}$

- Applying Newton to an approximation $\mathbf{x}$
  gives a better approximation $\mathcal{N}_f(\mathbf{x})$.
- $\overline{0}, \mathcal{N}_f(\overline{0}), \mathcal{N}_f(\mathcal{N}_f(\overline{0})), \ldots$ converges linearly to $\mathrm{LFP}(f)$ from below [Esparza, K., Luttenberger, 2010]
- But what about upper bounds? (Newton cannot be used for that)

Upper bounds

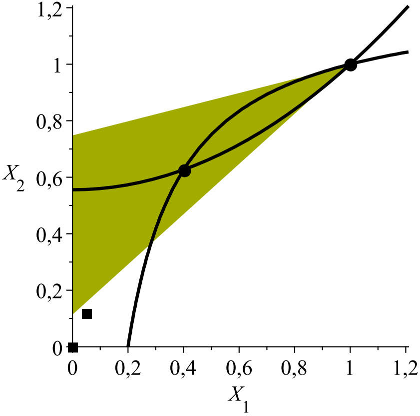


Upper bounds



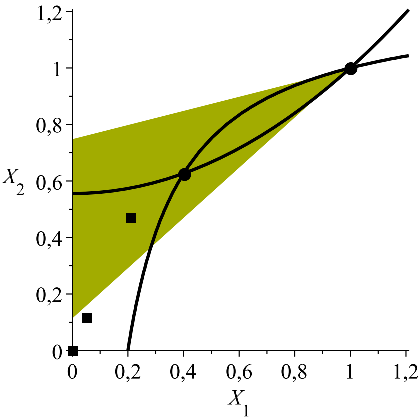
- $\mathrm{LFP}(f) \neq \overline{1} \Rightarrow$ there exists a green area of points $\mathbf{x}$ with $f'(\overline{1})(\overline{1} - \mathbf{x}) \succ (\overline{1} - \mathbf{x})$.

Upper bounds



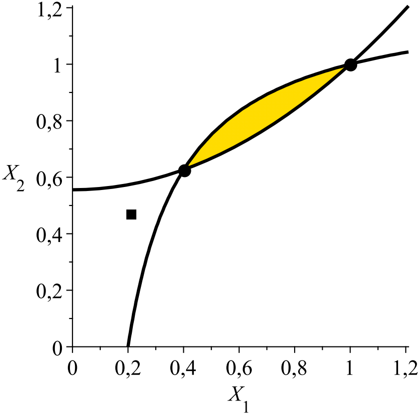
- $\mathrm{LFP}(f) \neq \overline{1} \Rightarrow$ there exists a green area of points $\mathbf{x}$ with $f'(\overline{1})(\overline{1} - \mathbf{x}) \succ (\overline{1} - \mathbf{x})$.

Upper bounds



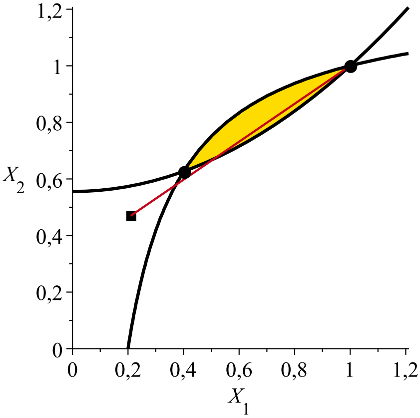
- $\mathrm{LFP}(f) \neq \overline{1} \Rightarrow$ there exists a green area of points $\mathbf{x}$ with $f'(\overline{1})(\overline{1} - \mathbf{x}) \succ (\overline{1} - \mathbf{x})$.
- Any sequence $\mathbf{y}^{(1)}, \mathbf{y}^{(2)}, \ldots$ of points converging to $\mathrm{LFP}(f)$ (e.g. Newton iterates) enters the green area.

Upper bounds



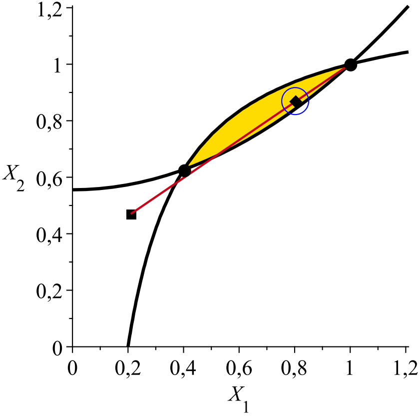
- $\mathrm{LFP}(f) \neq \overline{1} \Rightarrow$ there exists a green area of points $\mathbf{x}$ with $f'(\overline{1})(\overline{1} - \mathbf{x}) \succ (\overline{1} - \mathbf{x})$.
- Any sequence $\mathbf{y}^{(1)}, \mathbf{y}^{(2)}, \ldots$ of points converging to $\mathrm{LFP}(f)$ (e.g. Newton iterates) enters the green area.
- Using such a point we compute a strict post-fixed point $\mathbf{p}$.

Upper bounds



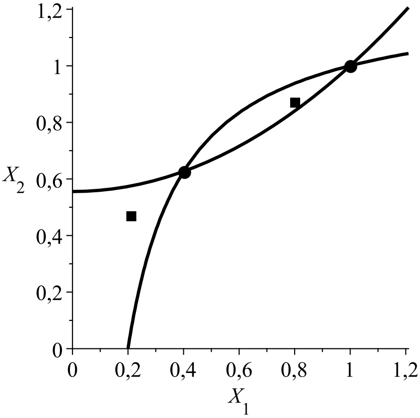
- $\mathrm{LFP}(f) \neq \overline{1} \Rightarrow$ there exists a green area of points $\mathbf{x}$ with $f'(\overline{1})(\overline{1} - \mathbf{x}) \succ (\overline{1} - \mathbf{x})$.
- Any sequence $\mathbf{y}^{(1)}, \mathbf{y}^{(2)}, \ldots$ of points converging to $\mathrm{LFP}(f)$ (e.g. Newton iterates) enters the green area.
- Using such a point we compute a strict post-fixed point $\mathbf{p}$.

Upper bounds



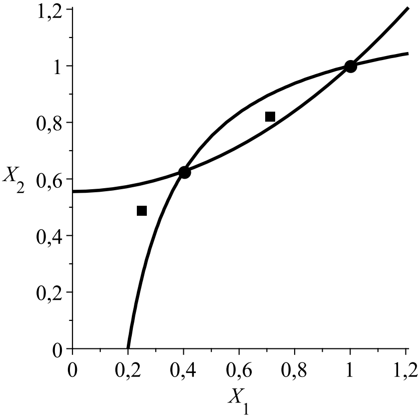
- $\mathrm{LFP}(f) \neq \overline{1} \Rightarrow$ there exists a green area of points $\mathbf{x}$ with $f'(\overline{1})(\overline{1} - \mathbf{x}) \succ (\overline{1} - \mathbf{x})$.
- Any sequence $\mathbf{y}^{(1)}, \mathbf{y}^{(2)}, \ldots$ of points converging to $\mathrm{LFP}(f)$ (e.g. Newton iterates) enters the green area.
- Using such a point we compute a strict post-fixed point $\mathbf{p}$.

Upper bounds



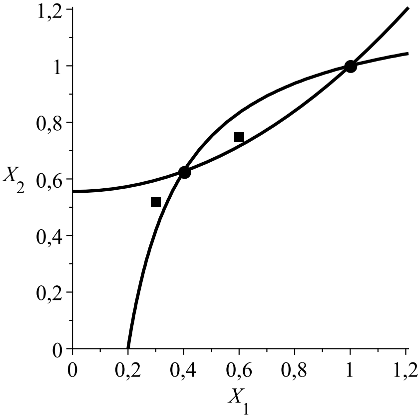
- $\mathrm{LFP}(f) \neq \overline{1} \Rightarrow$ there exists a green area of points $\mathbf{x}$ with $f'(\overline{1})(\overline{1} - \mathbf{x}) \succ (\overline{1} - \mathbf{x})$.
- Any sequence $\mathbf{y}^{(1)}, \mathbf{y}^{(2)}, \ldots$ of points converging to $\mathrm{LFP}(f)$ (e.g. Newton iterates) enters the green area.
- Using such a point we compute a strict post-fixed point $\mathbf{p}$.
- $\mathbf{p}, f(\mathbf{p}), f(f(\mathbf{p})), \ldots$ converges linearly to $\mathrm{LFP}(f)$ from above.

Upper bounds



- $\mathrm{LFP}(f) \neq \overline{1} \Rightarrow$ there exists a green area of points $\mathbf{x}$ with $f'(\overline{1})(\overline{1} - \mathbf{x}) \succ (\overline{1} - \mathbf{x})$.
- Any sequence $\mathbf{y}^{(1)}, \mathbf{y}^{(2)}, \ldots$ of points converging to $\mathrm{LFP}(f)$ (e.g. Newton iterates) enters the green area.
- Using such a point we compute a strict post-fixed point $\mathbf{p}$.
- $\mathbf{p}, f(\mathbf{p}), f(f(\mathbf{p})), \ldots$ converges linearly to $\mathrm{LFP}(f)$ from above.

Upper bounds



- $\mathrm{LFP}(f) \neq \overline{1} \Rightarrow$ there exists a green area of points $\mathbf{x}$ with $f'(\overline{1})(\overline{1} - \mathbf{x}) \succ (\overline{1} - \mathbf{x})$.
- Any sequence $\mathbf{y}^{(1)}, \mathbf{y}^{(2)}, \ldots$ of points converging to $\mathrm{LFP}(f)$ (e.g. Newton iterates) enters the green area.
- Using such a point we compute a strict post-fixed point $\mathbf{p}$.
- $\mathbf{p}, f(\mathbf{p}), f(f(\mathbf{p})), \ldots$ converges linearly to $\mathrm{LFP}(f)$ from above.

The algorithm so far

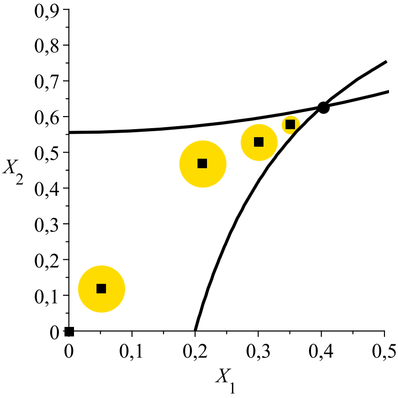
1. Set $\mathbf{lb} := \overline{0}, \mathbf{ub} := \overline{1}$.
2. Set $\mathbf{lb} := \mathcal{N}_f(\mathbf{lb})$.
3. If $f'(\overline{1})(\overline{1} - \mathbf{lb}) \succ (\overline{1} - \mathbf{lb})$ and $\mathbf{ub} = \overline{1}$,
   compute strict post-fixed point $\mathbf{p}$ and set $\mathbf{ub} := \mathbf{p}$.
4. If $\mathbf{ub} \neq \overline{1}$, set $\mathbf{ub} := f(\mathbf{ub})$.
5. If $\mathbf{ub} - \mathbf{lb} \not\leq \overline{\epsilon}$ go to (2).

Problems with exact computations

- For computing a Newton iterate $\mathcal{N}_f(\mathbf{x})$ we have to solve a linear equation system.
- For reliable results: Exact (rational) arithmetic
- The number of bits needed to represent the exact iterates grows exponentially with the number of iterations.
- Similar problem with exact upper bounds.
- We want to use "inexact" arithmetic operations with finite precision, e.g. floating-point arithmetic, in a "controlled" and "local" fashion...
- ... Especially: Detection and correction of round-off errors

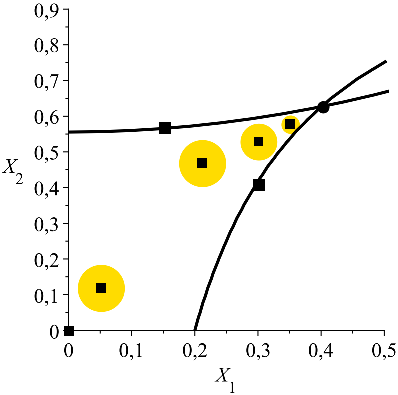
Avoiding exact computations

- Each Newton iterate $\mathcal{N}_f(\mathbf{x})$ is surrounded by a ϵ-ball $C_\mathbf{x}$ of points $\mathbf{y}$ with $\overline{1} \succ f(\mathbf{y}) \succ \mathbf{y} \succ f(\mathbf{x})$.



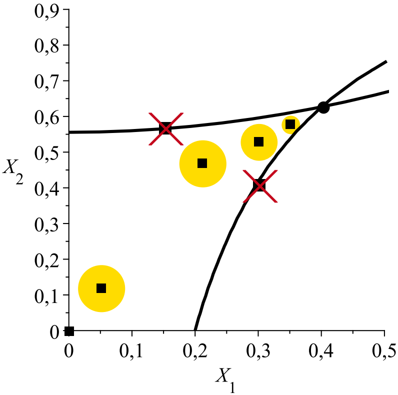
Avoiding exact computations

- Each Newton iterate $\mathcal{N}_f(\mathbf{x})$ is surrounded by a ϵ-ball $C_{\mathbf{x}}$ of points $\mathbf{y}$ with $\overline{1} \succ f(\mathbf{y}) \succ \mathbf{y} \succ f(\mathbf{x})$.



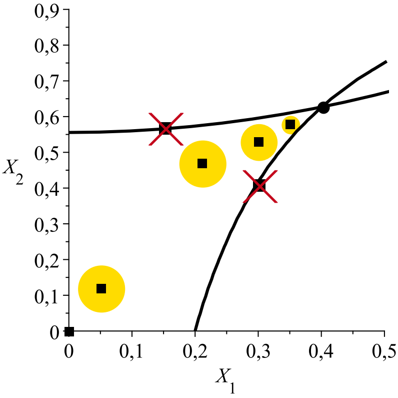
Avoiding exact computations

- Each Newton iterate $\mathcal{N}_f(\mathbf{x})$ is surrounded by a ϵ-ball $C_{\mathbf{x}}$ of points $\mathbf{y}$ with $\overline{1} \succ f(\mathbf{y}) \succ \mathbf{y} \succ f(\mathbf{x})$.



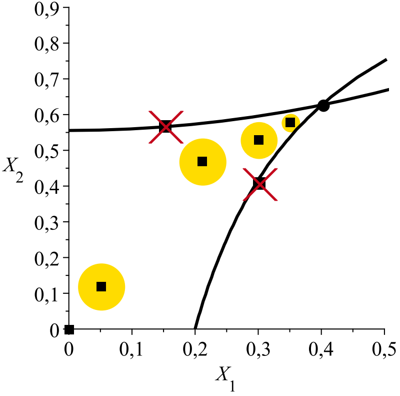
Avoiding exact computations

- Each Newton iterate $\mathcal{N}_f(\mathbf{x})$ is surrounded by a ϵ-ball $C_{\mathbf{x}}$ of points $\mathbf{y}$ with $\overline{1} \succ f(\mathbf{y}) \succ \mathbf{y} \succ f(\mathbf{x})$.
- Idea: Instead of $\mathcal{N}_f(\mathbf{x})$ compute any $\mathbf{y} \in C_{\mathbf{x}}$: Still converges!



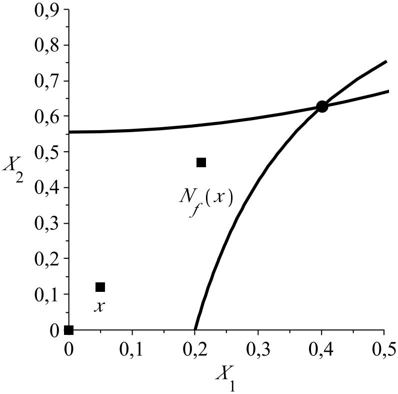
Avoiding exact computations

- Each Newton iterate $\mathcal{N}_f(\mathbf{x})$ is surrounded by a ϵ-ball $C_{\mathbf{x}}$ of points $\mathbf{y}$ with $\overline{1} \succ f(\mathbf{y}) \succ \mathbf{y} \succ f(\mathbf{x})$.
- Idea: Instead of $\mathcal{N}_f(\mathbf{x})$ compute any $\mathbf{y} \in C_{\mathbf{x}}$: Still converges!
- Advantage: fewer bits to store and work with.



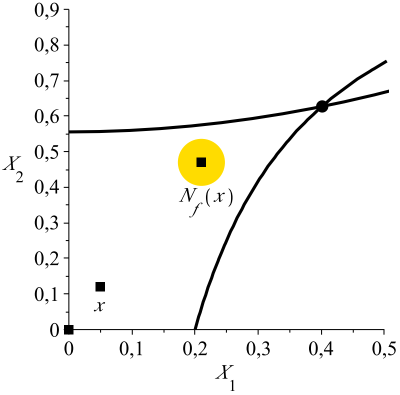
Avoiding exact computations

- Each Newton iterate $\mathcal{N}_f(\mathbf{x})$ is surrounded by a ϵ-ball $C_{\mathbf{x}}$ of points $\mathbf{y}$ with $\overline{1} \succ f(\mathbf{y}) \succ \mathbf{y} \succ f(\mathbf{x})$.
- Idea: Instead of $\mathcal{N}_f(\mathbf{x})$ compute any $\mathbf{y} \in C_{\mathbf{x}}$: Still converges!
- Advantage: fewer bits to store and work with.



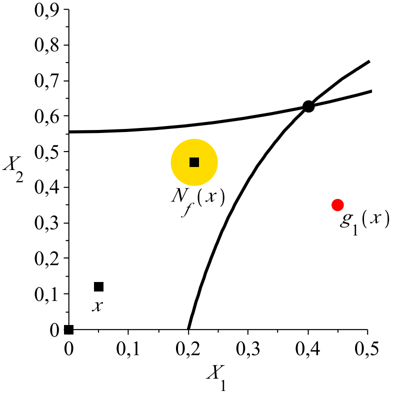
Avoiding exact computations

- Each Newton iterate $\mathcal{N}_f(\mathbf{x})$ is surrounded by a ϵ-ball $C_\mathbf{x}$ of points $\mathbf{y}$ with $\overline{1} \succ f(\mathbf{y}) \succ \mathbf{y} \succ f(\mathbf{x})$.
- Idea: Instead of $\mathcal{N}_f(\mathbf{x})$ compute any $\mathbf{y} \in C_\mathbf{x}$: Still converges!
- Advantage: fewer bits to store and work with.



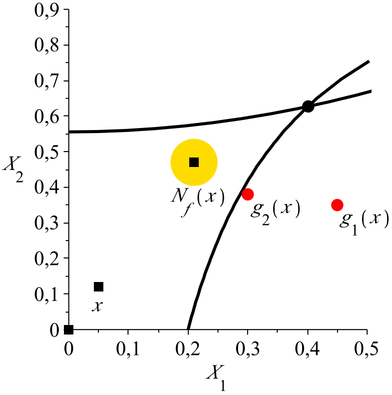
Avoiding exact computations

- Each Newton iterate $\mathcal{N}_f(\mathbf{x})$ is surrounded by a ϵ-ball $C_\mathbf{x}$ of points $\mathbf{y}$ with $\overline{1} \succ f(\mathbf{y}) \succ \mathbf{y} \succ f(\mathbf{x})$.
- Idea: Instead of $\mathcal{N}_f(\mathbf{x})$ compute any $\mathbf{y} \in C_\mathbf{x}$: Still converges!
- Advantage: fewer bits to store and work with.



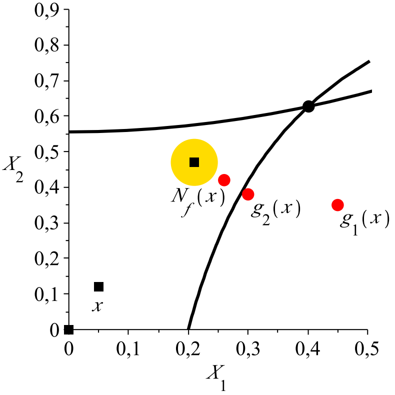
Avoiding exact computations

- Each Newton iterate $\mathcal{N}_f(\mathbf{x})$ is surrounded by a ϵ-ball $C_{\mathbf{x}}$ of points $\mathbf{y}$ with $\overline{1} \succ f(\mathbf{y}) \succ \mathbf{y} \succ f(\mathbf{x})$.
- Idea: Instead of $\mathcal{N}_f(\mathbf{x})$ compute any $\mathbf{y} \in C_{\mathbf{x}}$: Still converges!
- Advantage: fewer bits to store and work with.



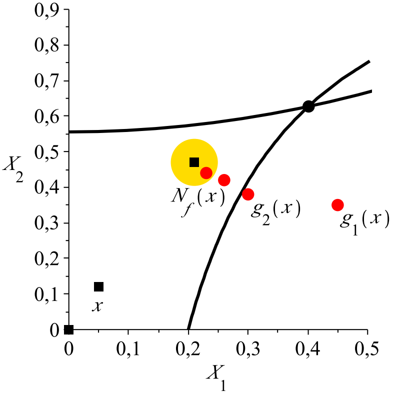
Avoiding exact computations

- Each Newton iterate $\mathcal{N}_f(\mathbf{x})$ is surrounded by a ϵ-ball $C_\mathbf{x}$ of points $\mathbf{y}$ with $\overline{1} \succ f(\mathbf{y}) \succ \mathbf{y} \succ f(\mathbf{x})$.
- Idea: Instead of $\mathcal{N}_f(\mathbf{x})$ compute any $\mathbf{y} \in C_\mathbf{x}$: Still converges!
- Advantage: fewer bits to store and work with.



Avoiding exact computations

- Each Newton iterate $\mathcal{N}_f(\mathbf{x})$ is surrounded by a ϵ-ball $C_\mathbf{x}$ of points $\mathbf{y}$ with $\overline{1} \succ f(\mathbf{y}) \succ \mathbf{y} \succ f(\mathbf{x})$.
- Idea: Instead of $\mathcal{N}_f(\mathbf{x})$ compute any $\mathbf{y} \in C_\mathbf{x}$: Still converges!
- Advantage: fewer bits to store and work with.



The Floating Assignment

- We write

$$\mathbf{y} \leftsquigarrow \mathcal{N}_f(\mathbf{x}) \textbf{ such that } \boxed{\overline{1} \succ f(\mathbf{y}) \succ \mathbf{y} \succ f(\mathbf{x})}$$

  as syntactic sugar for

  1. Set $i := 0$.
  2. Compute $\mathbf{y} := g^{(i)}(\mathbf{x})$.
  3. If not $\overline{1} \succ f(\mathbf{y}) \succ \mathbf{y} \succ f(\mathbf{x})$ set $i := i + 1$ and go to (2).

- Computation of $g^{(i)}$: E.g. convert $\mathbf{x}$ to a floating-point number, perform the operation, convert the result back.
- $g^{(0)}(\mathbf{x}), g^{(1)}(\mathbf{x}), g^{(2)}(\mathbf{x}), \ldots \to \mathcal{N}_f(\mathbf{x})$
- Precision of the numbers/operations increases with i.
  (Maple, GNU Multi-Precision Library)
- We replace the computation of the iterates by floating assignments.

The Algorithm

1. Set $\mathbf{lb} := \overline{0}, \mathbf{ub} := \overline{1}$.
2. Set $\boxed{\mathbf{y} \leftsquigarrow \mathcal{N}_f(\mathbf{x}) \textbf{ such that } \overline{1} \succ f(\mathbf{y}) \succ \mathbf{y} \succ f(\mathbf{x})}$.
3. If $f'(\overline{1})(\overline{1} - \mathbf{lb}) \succ (\overline{1} - \mathbf{lb})$ and $\mathbf{ub} = \overline{1}$,
   compute strict post-fixed point $\mathbf{p}$ and set $\mathbf{ub} := \mathbf{p}$.
4. If $\mathbf{ub} \neq \overline{1}$, set $\mathbf{ub} := f(\mathbf{ub})$.
5. If $\mathbf{ub} - \mathbf{lb} \not\preceq \overline{\epsilon}$ go to (2).

- Floating-Assignments also for upper bounds possible.

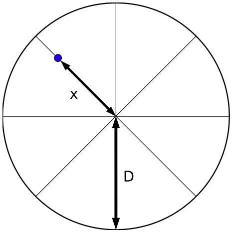
Summary

- The algorithm
  - computes reliable lower and upper bounds for LFP(f),
    which are arbitrarily close.
  - uses inexact arithmetic for costly computations
    $\Rightarrow$ In practice, the precision needs to be increased only rarely.
    $\Rightarrow$ We observe a significant speed-up.

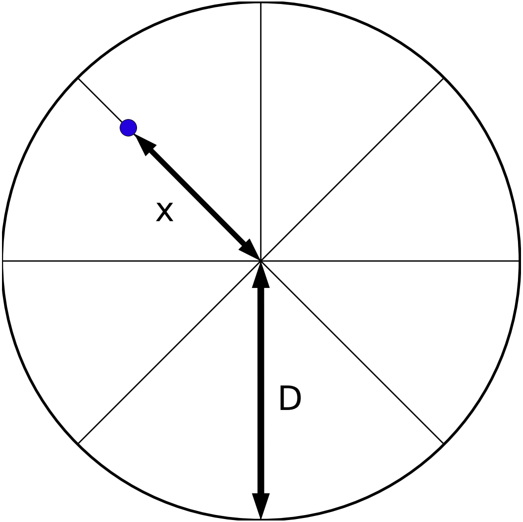

Explosion risk

Nuclear Fission

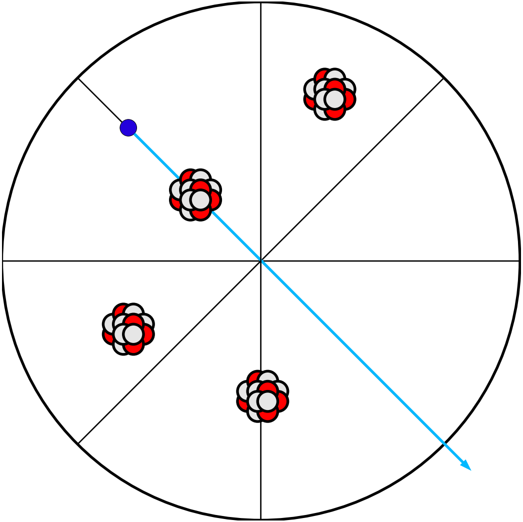
- Example from [Harris, 1963].
- Single neutron with distance x to the centre of a ball of radius D of radioactive material.
- Before exiting the ball, the neutron might collide with a nucleus.
- Other free neutrons may emerge from the collision, which may trigger more collisions $\Rightarrow$ Danger (or chance?) of chain reaction!



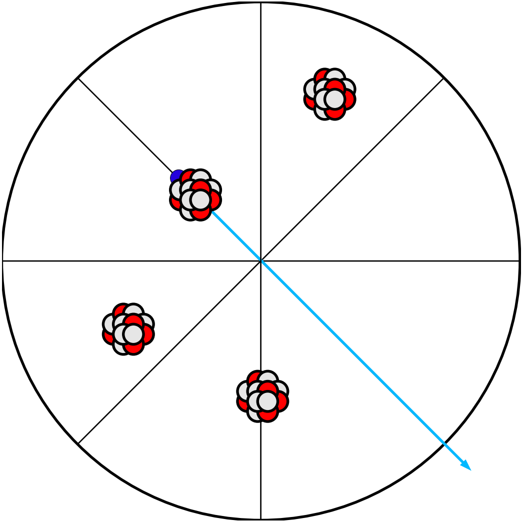
- In case of a collision, with probability ...
  1. 0.025, the neutron will be absorbed into the nucleus.
  2. 0.830, the neutron will be deflected by the nucleus.
  3. 0.070, the nucleus will break up and two neutrons emerge.
  4. 0.050, nucleus breaks and 3 new neutrons.
  5. 0.025, nucleus breaks and 4 new neutrons.



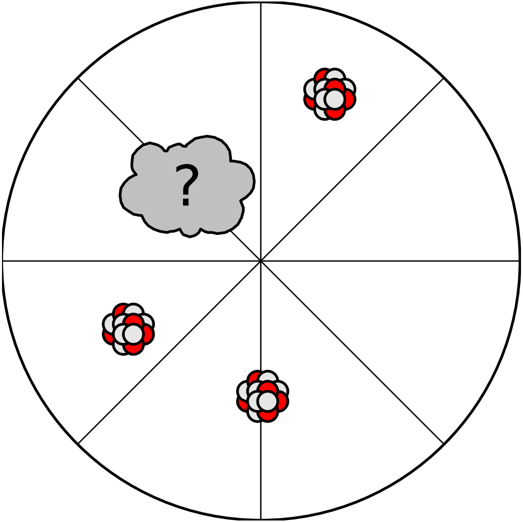
- In case of a collision, with probability ...
  1. 0.025, the neutron will be absorbed into the nucleus.
  2. 0.830, the neutron will be deflected by the nucleus.
  3. 0.070, the nucleus will break up and two neutrons emerge.
  4. 0.050, nucleus breaks and 3 new neutrons.
  5. 0.025, nucleus breaks and 4 new neutrons.



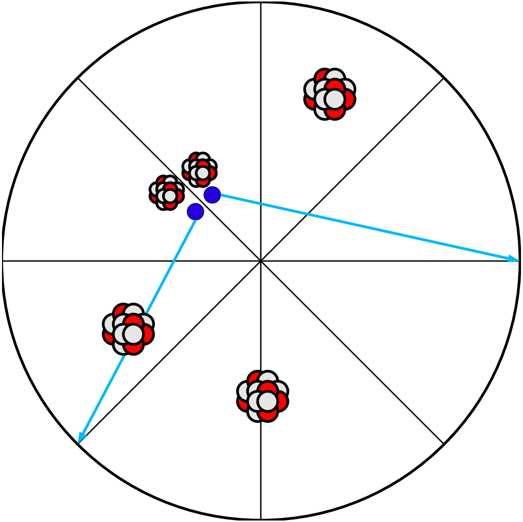
- In case of a collision, with probability ...
  1. 0.025, the neutron will be absorbed into the nucleus.
  2. 0.830, the neutron will be deflected by the nucleus.
  3. 0.070, the nucleus will break up and two neutrons emerge.
  4. 0.050, nucleus breaks and 3 new neutrons.
  5. 0.025, nucleus breaks and 4 new neutrons.



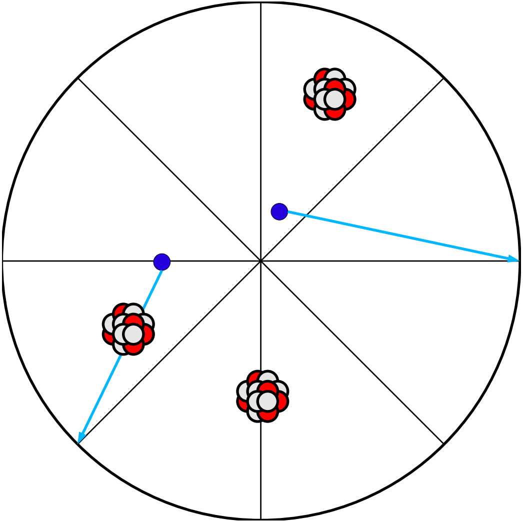
- In case of a collision, with probability ...
  1. 0.025, the neutron will be absorbed into the nucleus.
  2. 0.830, the neutron will be deflected by the nucleus.
  3. 0.070, the nucleus will break up and two neutrons emerge.
  4. 0.050, nucleus breaks and 3 new neutrons.
  5. 0.025, nucleus breaks and 4 new neutrons.



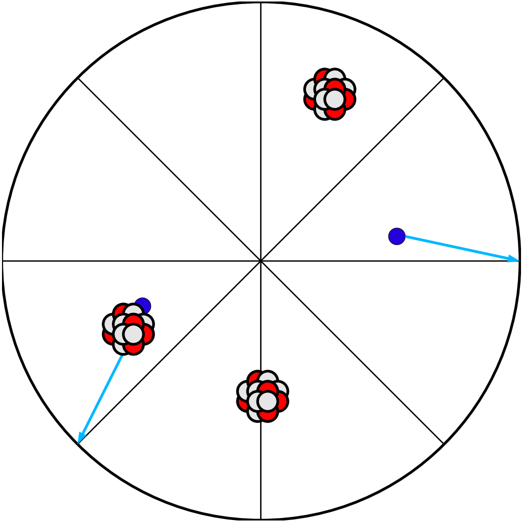
- In case of a collision, with probability ...
  1. 0.025, the neutron will be absorbed into the nucleus.
  2. 0.830, the neutron will be deflected by the nucleus.
  3. 0.070, the nucleus will break up and two neutrons emerge.
  4. 0.050, nucleus breaks and 3 new neutrons.
  5. 0.025, nucleus breaks and 4 new neutrons.



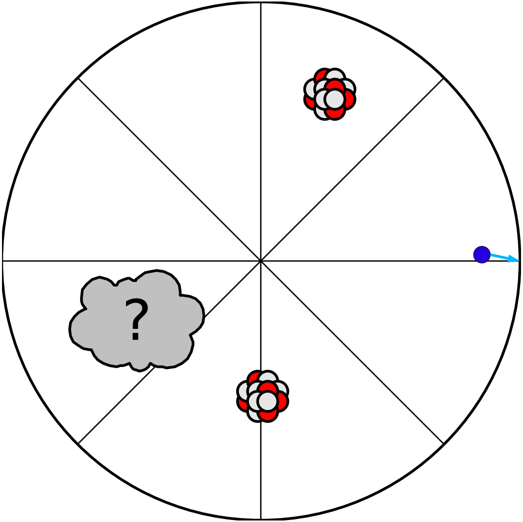
- In case of a collision, with probability ...
  1. 0.025, the neutron will be absorbed into the nucleus.
  2. 0.830, the neutron will be deflected by the nucleus.
  3. 0.070, the nucleus will break up and two neutrons emerge.
  4. 0.050, nucleus breaks and 3 new neutrons.
  5. 0.025, nucleus breaks and 4 new neutrons.



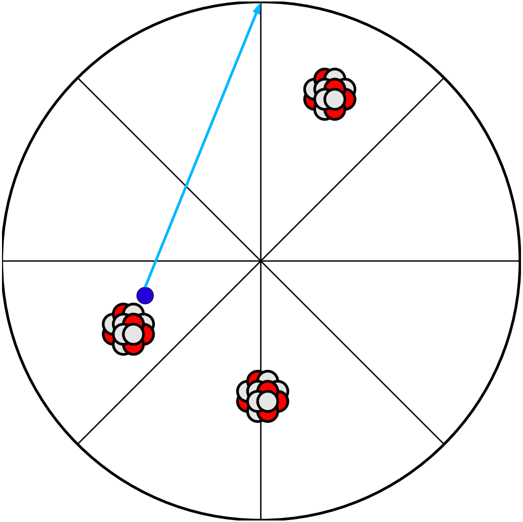
- In case of a collision, with probability ...
  1. 0.025, the neutron will be absorbed into the nucleus.
  2. 0.830, the neutron will be deflected by the nucleus.
  3. 0.070, the nucleus will break up and two neutrons emerge.
  4. 0.050, nucleus breaks and 3 new neutrons.
  5. 0.025, nucleus breaks and 4 new neutrons.



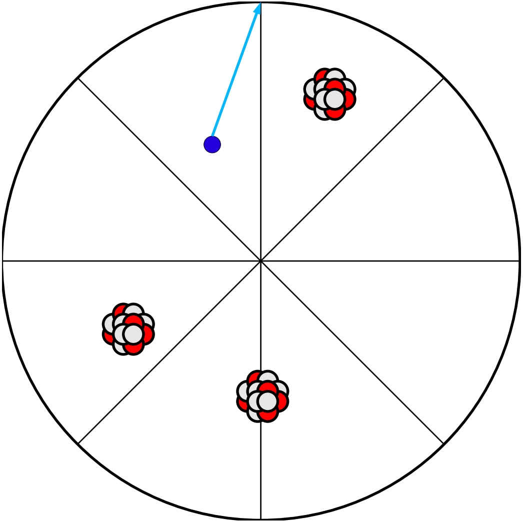
- In case of a collision, with probability ...
  1. 0.025, the neutron will be absorbed into the nucleus.
  2. 0.830, the neutron will be deflected by the nucleus.
  3. 0.070, the nucleus will break up and two neutrons emerge.
  4. 0.050, nucleus breaks and 3 new neutrons.
  5. 0.025, nucleus breaks and 4 new neutrons.



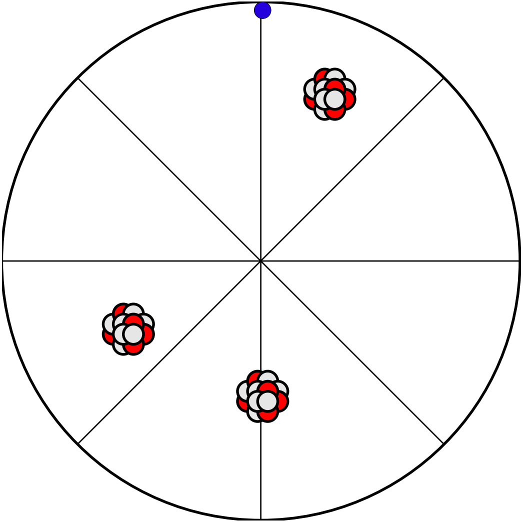
- In case of a collision, with probability ...
  1. 0.025, the neutron will be absorbed into the nucleus.
  2. 0.830, the neutron will be deflected by the nucleus.
  3. 0.070, the nucleus will break up and two neutrons emerge.
  4. 0.050, nucleus breaks and 3 new neutrons.
  5. 0.025, nucleus breaks and 4 new neutrons.



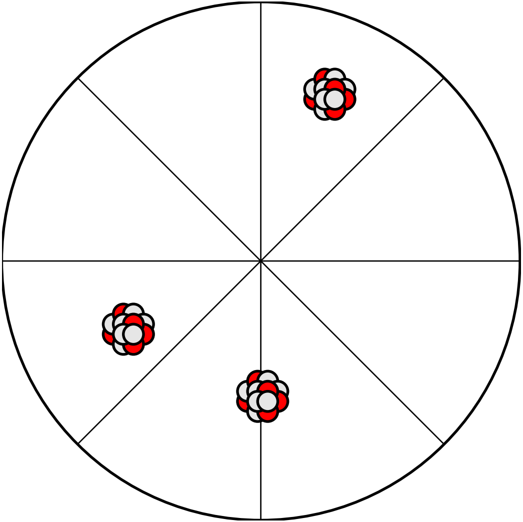
- In case of a collision, with probability ...
  1. 0.025, the neutron will be absorbed into the nucleus.
  2. 0.830, the neutron will be deflected by the nucleus.
  3. 0.070, the nucleus will break up and two neutrons emerge.
  4. 0.050, nucleus breaks and 3 new neutrons.
  5. 0.025, nucleus breaks and 4 new neutrons.



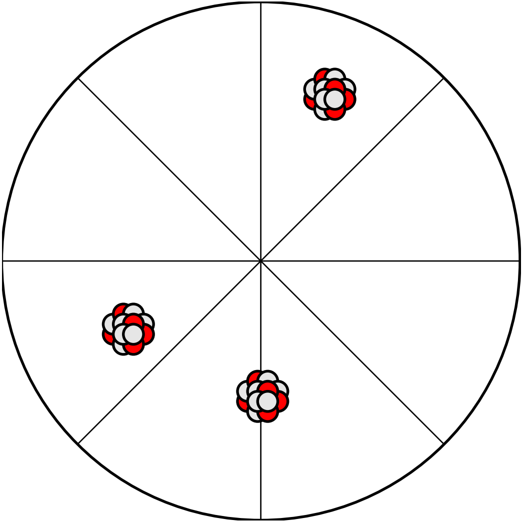
- In case of a collision, with probability ...
  1. 0.025, the neutron will be absorbed into the nucleus.
  2. 0.830, the neutron will be deflected by the nucleus.
  3. 0.070, the nucleus will break up and two neutrons emerge.
  4. 0.050, nucleus breaks and 3 new neutrons.
  5. 0.025, nucleus breaks and 4 new neutrons.



- In case of a collision, with probability ...
  1. 0.025, the neutron will be absorbed into the nucleus.
  2. 0.830, the neutron will be deflected by the nucleus.
  3. 0.070, the nucleus will break up and two neutrons emerge.
  4. 0.050, nucleus breaks and 3 new neutrons.
  5. 0.025, nucleus breaks and 4 new neutrons.



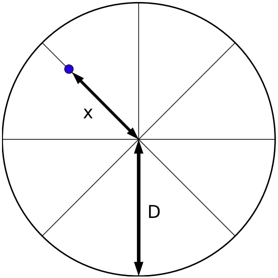
- In case of a collision, with probability ...
  1. 0.025, the neutron will be absorbed into the nucleus.
  2. 0.830, the neutron will be deflected by the nucleus.
  3. 0.070, the nucleus will break up and two neutrons emerge.
  4. 0.050, nucleus breaks and 3 new neutrons.
  5. 0.025, nucleus breaks and 4 new neutrons.



- Given is $\ell(x)$: the probability that a neutron starting at x leaves the ball without collision.
- Given is $R(x, y)$: the probability that a neutron starting at x collides with a nucleus at y.

Wanted Values

- Wanted: $Q_D(x)$, the probability that from a single neutron, starting at x, only finitely many free neutrons emerge $=$ Probability of NO EXPLOSION
- Wanted: Critical radius, i.e., the largest D with $Q_D(0) = 1$



Discretization gives a PSP

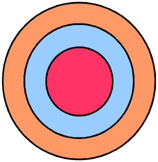
- $Q_D(x)$ satisfies the functional equation

$$Q_D(x) = \ell(x) + \int_0^D R(x,y) f(Q_D(y))\, dy.$$

with $f(z) = 0.025 + 0.83z + 0.07z^2 + 0.05z^3 + 0.025z^4$.

- We discretize the interval $[0, D]$ into
  n shells with thickness D/n.

$\Rightarrow$ PSP with n variables $X_1, \ldots, X_n$.



Discretization gives a PSP

- Resulting equation system
(constants $\ell_1, r_{i,j}$ can be numerically computed):

$$
\begin{aligned}
X_1 &= \ell_1 + \sum_{i=1}^{n} r_{1,i} \cdot (0.025 + 0.83X_i + 0.07X_i^2 + 0.05X_i^3 + 0.025X_i^4) \\
X_2 &= \ell_2 + \sum_{i=1}^{n} r_{2,i} \cdot (0.025 + 0.83X_i + 0.07X_i^2 + 0.05X_i^3 + 0.025X_i^4) \\
& \ldots \\
X_n &= \ell_n + \sum_{i=1}^{n} r_{n,i} \cdot (0.025 + 0.83X_i + 0.07X_i^2 + 0.05X_i^3 + 0.025X_i^4)
\end{aligned}
$$

Experiments

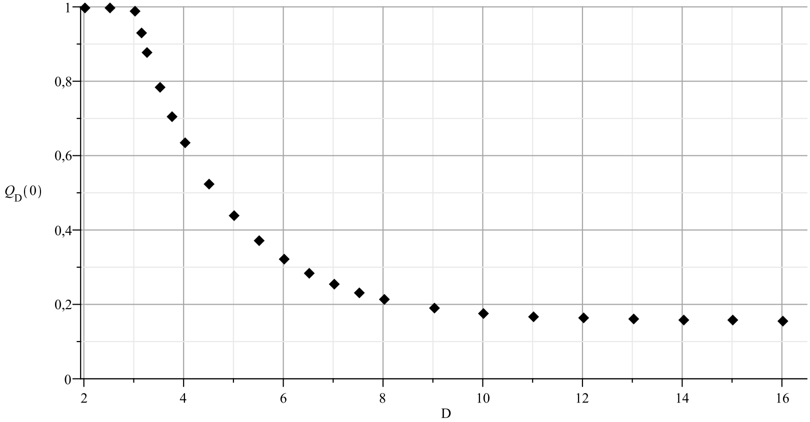
- Computation for $n = 100$, different radii D

D	2	3	6	10
Still safe?	✓	☠	☠	☠
Cons. check (our algorithm)	2s	2s	2s	2s
Cons. check (exact LP)	258s	124s	168s	222s
Approx. Q_D ($\epsilon = 0.001$)	4s	32s	21s	17s

- Numerical results are similar to [Harris, 1963].
- We observed at most two increases of precision per computation.

Experiments

- Values of $Q_D(0)$ for different radii D
- Binary search using consistency algorithm: Critical radius lies in $[2.981, 2.991]$ (Harris: ca. 2.9).



Thank you!

Theodore E. Harris
The theory of branching processes
Springer-Verlag, 1963

Kousha Etessami and Mihalis Yannakakis
Recursive Markov chains, stochastic grammars, and monotone systems of nonlinear equations
Journal of the ACM, 56(1):1-66, 2009

Javier Esparza, Andreas Gaiser, Stefan Kiefer
Computing least fixed points of probabilistic systems of polynomials
STACS 2010

Javier Esparza, Stefan Kiefer, Michael Luttenberger
Computing the least fixed point of positive polynomial systems
SIAM Journal on Computing, 39(6):2282-2335, 2010