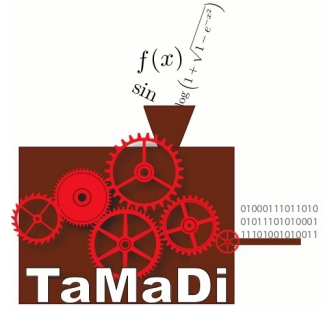
Correctly rounding elementary functions on GPU

Pierre Fortin, Mourad Gouicem, Stef Graillat

PEQUAN Team, LIP6/UPMC

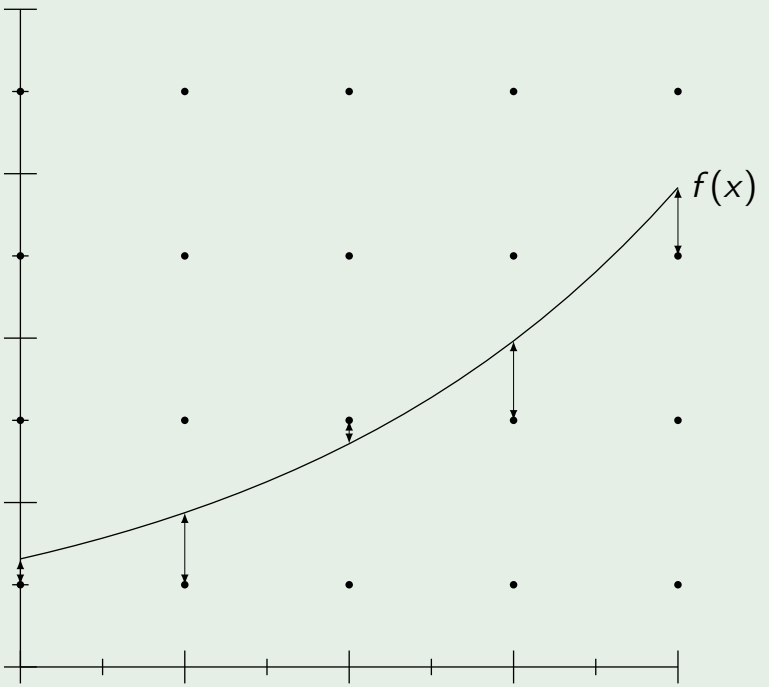
TaMaDi project Meeting, Paris, France
October 19^{th} 2012



The Table Maker's Dilemma

Given a function f defined over I and a rounding mode $\circ_p$, find ϵ such that $\forall x \in I$

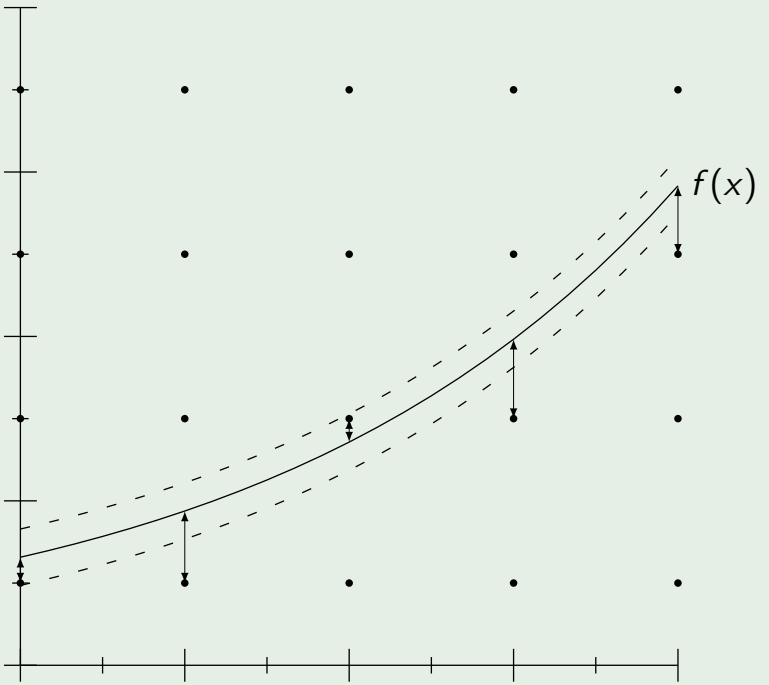
$$\circ_p(f(x) - \epsilon) = \circ_p(f(x) + \epsilon).$$



The Table Maker's Dilemma

Lefèvre general framework for HR-case search

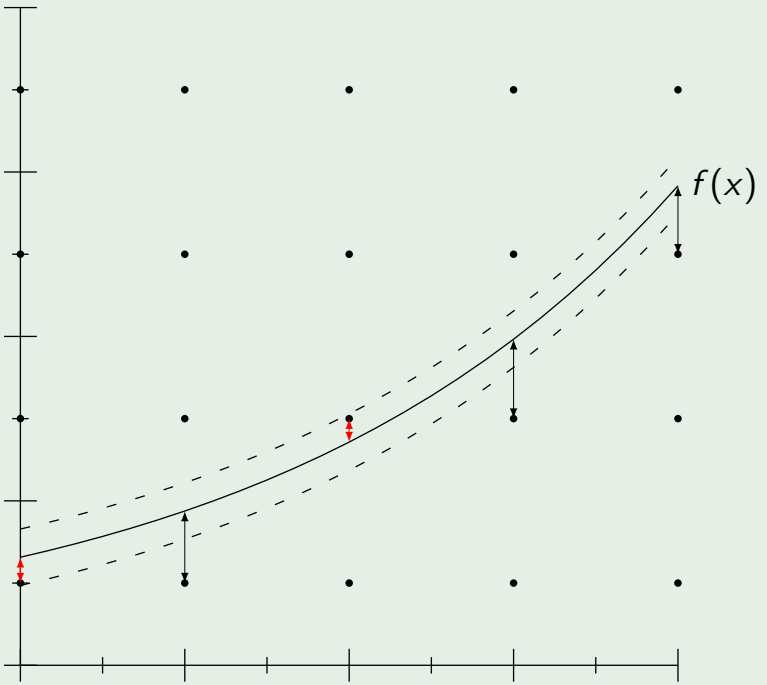
1. Split the domain and approximate the function by a degree 2 polynomial on each sub-domain with error ε.



The Table Maker's Dilemma

Lefèvre general framework for HR-case search

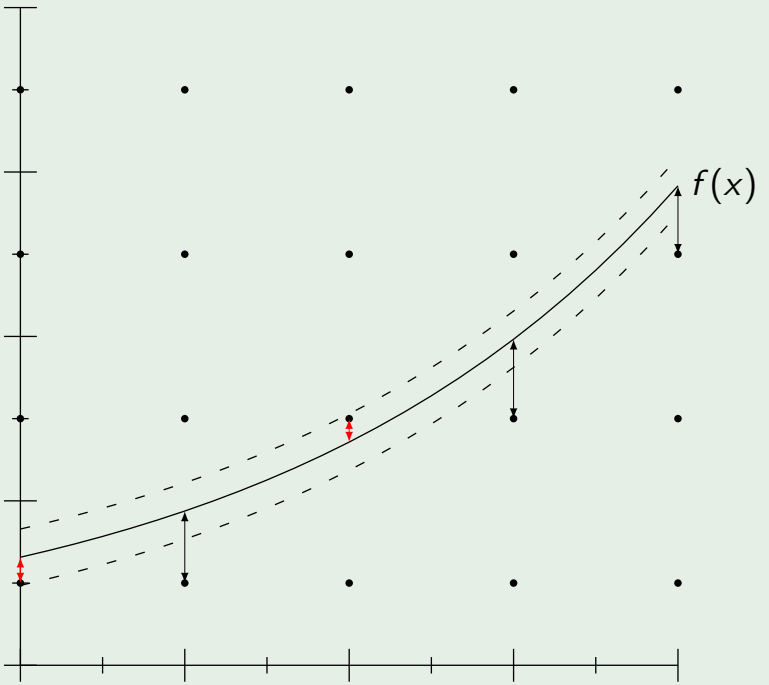
1. Split the domain and approximate the function by a degree 2 polynomial on each sub-domain with error ε.
2. Test efficiently if there exist hard-to-round cases.



The Table Maker's Dilemma

Lefèvre general framework for HR-case search

1. Split the domain and approximate the function by a degree 2 polynomial on each sub-domain with error ε.
2. Test efficiently if there exist hard-to-round cases.
3. If so, search them exhaustively.



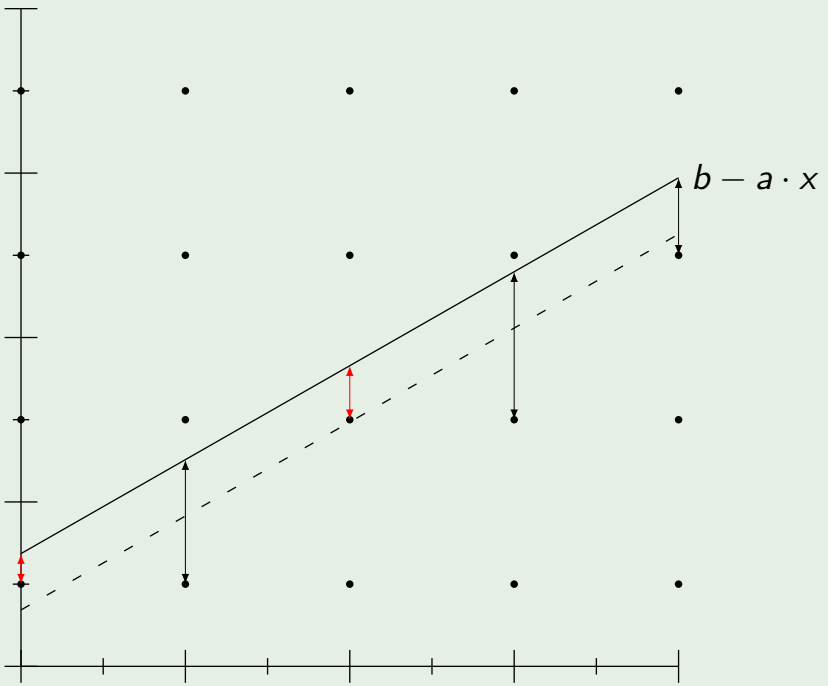
HR-case existence test

Problem

Given $b - a \cdot x$, is there a solution to :

$$\begin{cases} x < N \\ b - a \cdot x \bmod 1 < \varepsilon \end{cases}$$

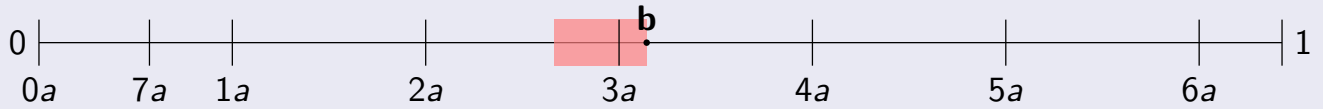
with $a, b, \varepsilon \in \mathbb{Q}$ and $x, N \in \mathbb{N}$?

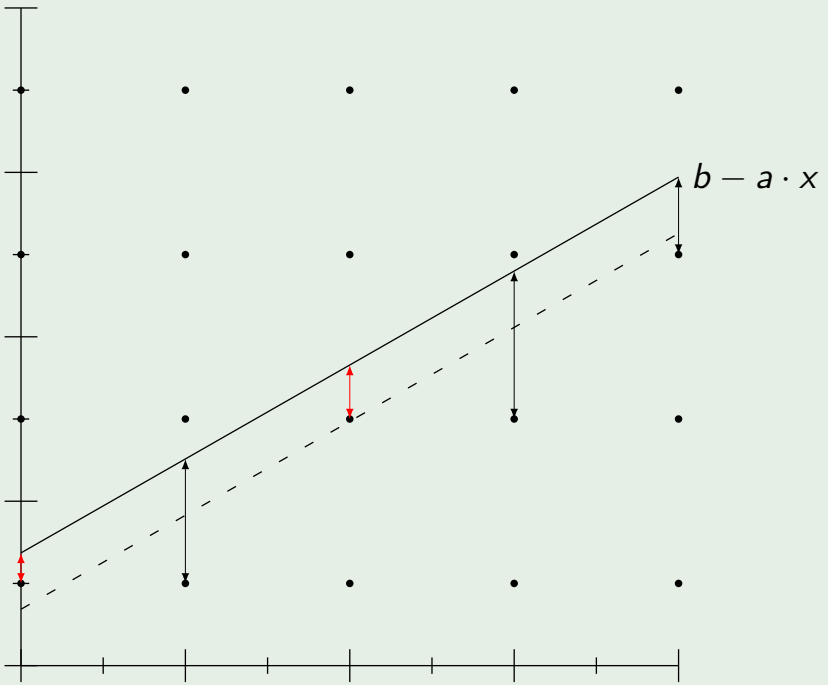


HR-case existence test

Strategy

- Place $a \cdot x$ modulo 1.
- Test if there are points at distance ε at the left of b.



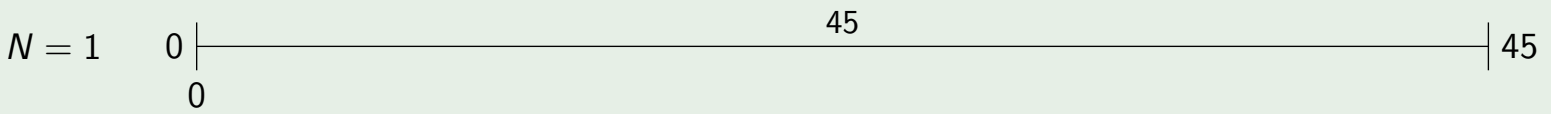


Positions of the $a \cdot x \mod 1$ on $[0, 1[$

Three distance theorem [Sla50]

The points $\{a \cdot x \mod 1 \mid x < N\}$ split the segment $[0, 1[$ into N segments. Their lengths take at most three different values, one being the sum of the two others.

Example : $a = 17/45$

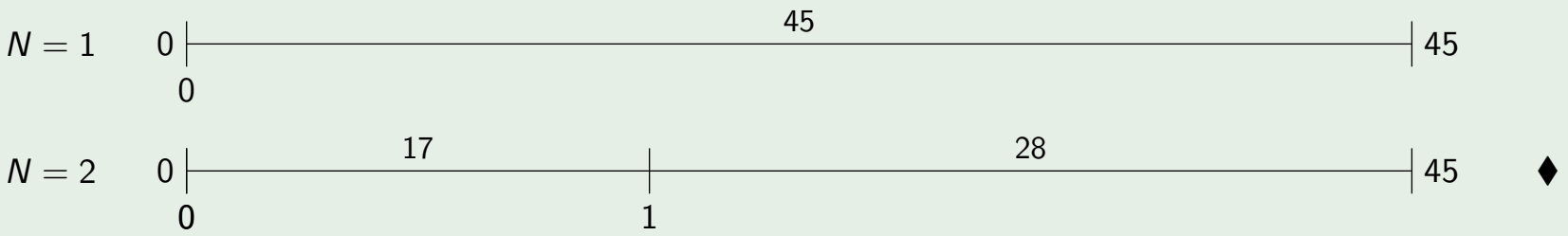


Positions of the $a \cdot x \mod 1$ on $[0, 1[$

Three distance theorem [Sla50]

The points $\{a \cdot x \mod 1 \mid x < N\}$ split the segment $[0, 1[$ into N segments. Their lengths take at most three different values, one being the sum of the two others.

Example : $a = 17/45$



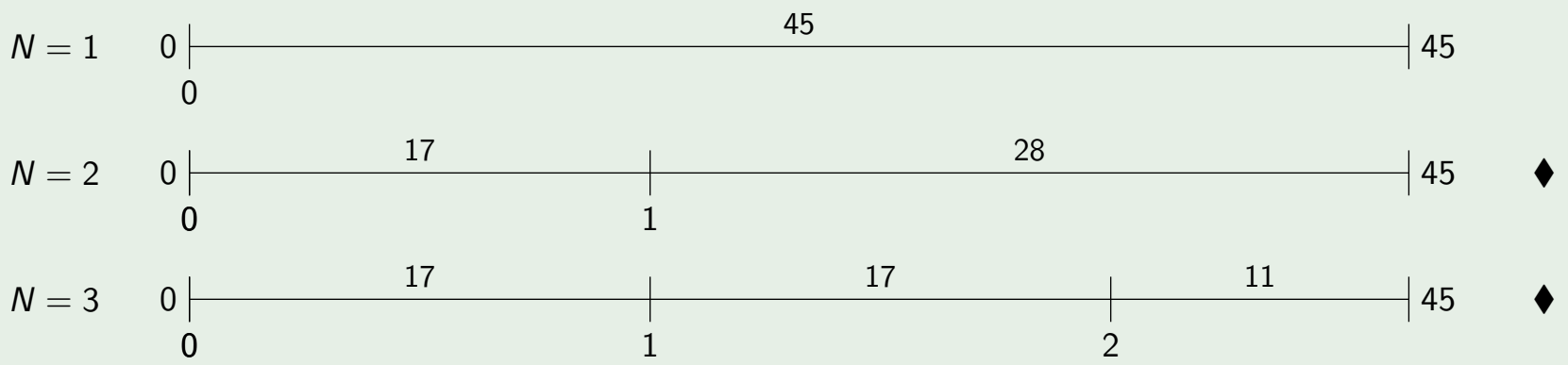
♦ : 2-length configurations

Positions of the $a \cdot x \mod 1$ on $[0, 1[$

Three distance theorem [Sla50]

The points $\{a \cdot x \mod 1 \mid x < N\}$ split the segment $[0, 1[$ into N segments. Their lengths take at most three different values, one being the sum of the two others.

Example : $a = 17/45$



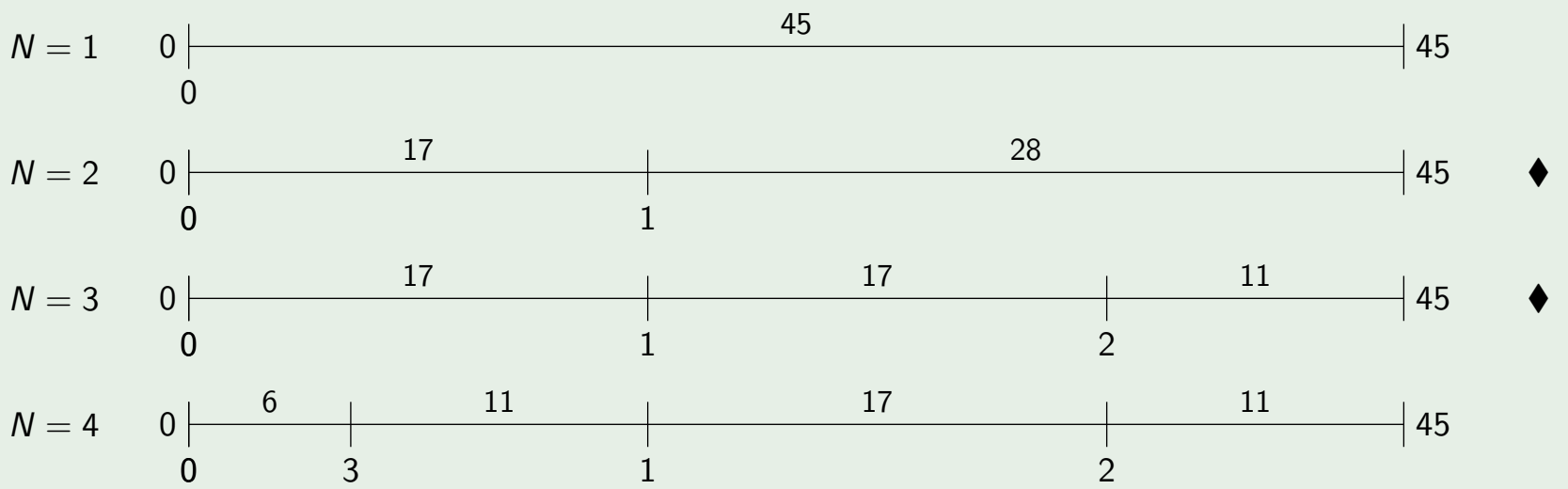
♦ : 2-length configurations

Positions of the $a \cdot x \mod 1$ on $[0, 1[$

Three distance theorem [Sla50]

The points $\{a \cdot x \mod 1 \mid x < N\}$ split the segment $[0, 1[$ into N segments. Their lengths take at most three different values, one being the sum of the two others.

Example : $a = 17/45$



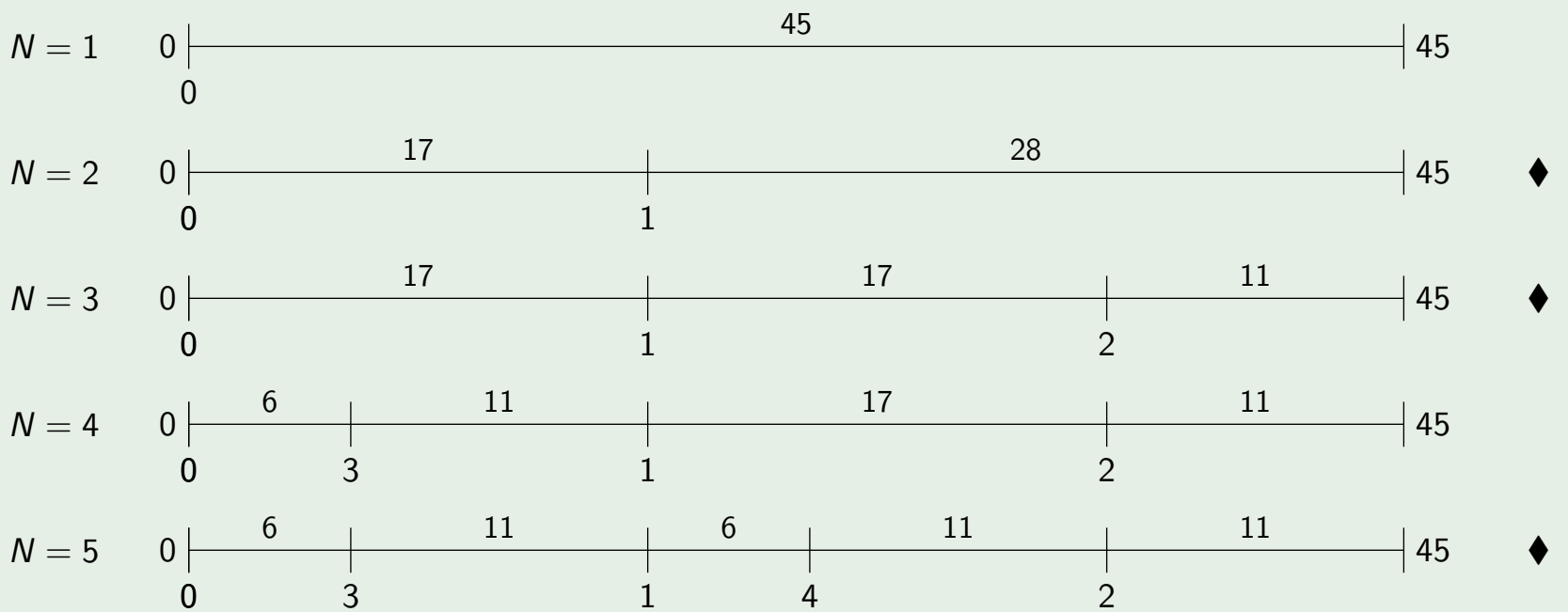
♦ : 2-length configurations

Positions of the $a \cdot x \mod 1$ on $[0, 1[$

Three distance theorem [Sla50]

The points $\{a \cdot x \mod 1 \mid x < N\}$ split the segment $[0, 1[$ into N segments. Their lengths take at most three different values, one being the sum of the two others.

Example : $a = 17/45$



♦ : 2-length configurations

Positions of the $a \cdot x \mod 1$ on $[0, 1[$

Generating 2-length configurations [Sla67]

- Let $a = [0; k_1, k_2, k_3, \cdots]$ and $\frac{p_i}{q_i}$ the i^{th} convergent.

Positions of the $a \cdot x \mod 1$ on $[0, 1[$

Generating 2-length configurations [Sla67]

- Let $a = [0; k_1, k_2, k_3, \cdots]$ and $\frac{p_i}{q_i}$ the i^{th} convergent.
- The distances equal $\theta_i = (-1)^i(q_i a - p_i)$ (> 0) and

$$q_i \theta_{i-1} + q_{i-1} \theta_i = 1 \tag{1}$$

Interpretation : if we place $N = q_i + q_{i-1}$ multiples of a, we have q_i intervals of length θ_{i-1} and q_{i-1} intervals of length θ_i.

Generating 2-length configurations [Sla67]

- Let $a = [0; k_1, k_2, k_3, \cdots]$ and $\frac{p_i}{q_i}$ the i^{th} convergent.
- The distances equal $\theta_i = (-1)^i(q_i a - p_i)$ (> 0) and

$$q_i\theta_{i-1} + q_{i-1}\theta_i = 1 \tag{1}$$

  Interpretation : if we place $N = q_i + q_{i-1}$ multiples of a, we have q_i intervals of length θ_{i-1} and q_{i-1} intervals of length θ_i.
- Recurrence relations :

$$\begin{array}{lll} q_{-1} = 0 & q_0 = 1 & q_i = q_{i-2} + k_i q_{i-1} \\ \theta_{-1} = 1 & \theta_0 = a & \theta_i = \theta_{i-2} - k_i \theta_{i-1} \end{array}$$

Computing $b - a \cdot x \mod 1$

Best left approximation [Ber01]

$$b = \sum_{i=0}^{+\infty} b_i \theta_i$$

where $0 \leq b_i \leq k_i$ and $b_{i+1} = 0$ if $b_i = k_i$.

Lefèvre HR-case search

- Computes iteratively $r_n = b - \sum_{i=0}^{n} b_i \theta_i$
- Stops when $r_n < \varepsilon$ or $q_{n-1} + q_n \geq N$ (number of fp in the tested domain).
- Difference with regular HR-case search : we compute the same b_i but in a different way.

Computing $b - a \cdot x \mod 1$

3 cases

- b is in an interval of length θ_i,
  $\Rightarrow$ Nothing to do : $b_{i+1} = 0$ and $r_{i+1} = r_i$
- b is in an interval of length θ_{i-1} and i is even,
  $\Rightarrow$ Reduction "from the left" : $r_{i+1} = r_i \mod \theta_i$
- b is in an interval of length θ_{i-1} and i is odd.
  $\Rightarrow$ Reduction "from the right" : $r_{i+1} = (r_i - \theta_{i+1}) \mod \theta_i$

Polynomial approximation generation

Using Taylor shifts

- Generate an approximation P of degree δ of the function.
- Translate P such that $P_i(x) = P(x + iN)$ (Taylor shift).

Hierarchical Method [Lef00]

- We write $x = kN + m$,
- We interpolate P in the binomial basis w.r.t the variable m

$$P(kN + m) = \sum_{j=0}^{\delta} a_j(k) \binom{j}{m}$$

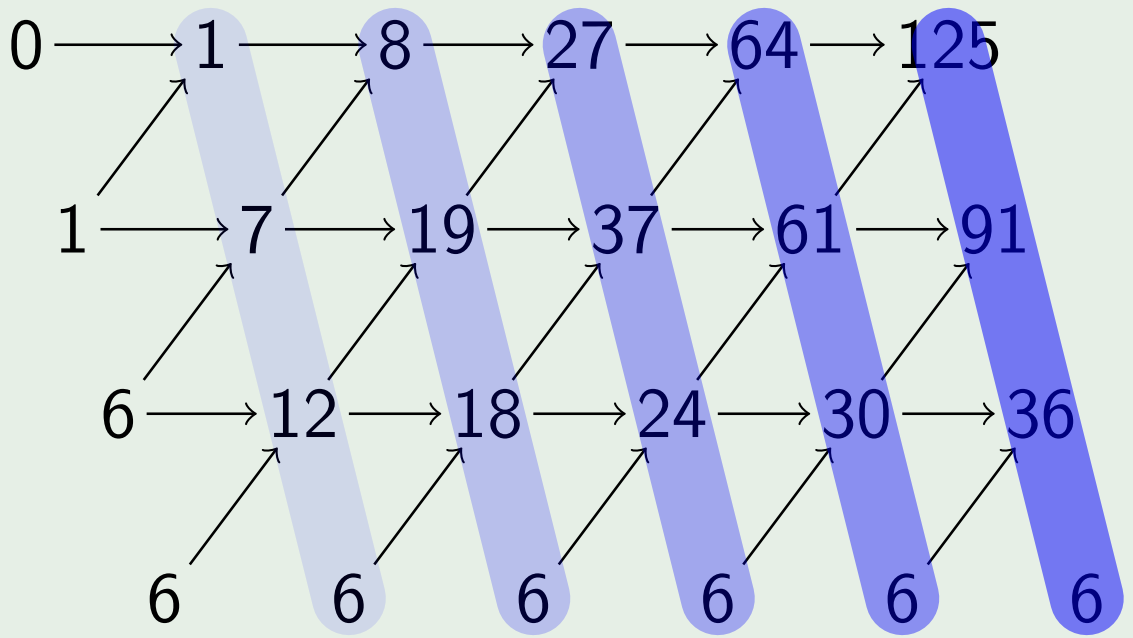
$\Rightarrow$ We can get $P_i(m)$ by evaluating the $a_j(k)$ in i.
$\Rightarrow$ A shift by N becomes $\delta + 1$ independent shifts by 1.

Polynomial approximation generation

Difference table algorithm

- Needs only $deg(a_j)$ additions to shift by 1.
- Is intrinsically sequential.

Example with x^3



Polynomial approximation generation

Straightforward algorithm

- Computes $a_j(k)$ by multiplying its coefficient vector by

$$\begin{pmatrix} \binom{k}{0} & \binom{k}{1} & \cdots & \binom{k}{deg(a_j)} \\ 0 & \binom{k}{0} & \cdots & \binom{k}{deg(a_j)-1} \\ 0 & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \binom{k}{0} \end{pmatrix}$$

- All the evaluations can be done in parallel.
- Is exactly similar as computing k difference table step.
- Needs more operations than difference table algorithm.

Polynomial approximation generation

Hybrid CPU-GPU deployment

- Compute $a_j(tS)$ with $t < T$ on CPU to form T packets of size S.
- Send the $a_j(tS)$ to the GPU memory.
- Run T threads on GPU. Each thread t computes the $a_j(tS + s)$ with $s < S$.

Remarks

- Packets are formed using the straightforward algorithm.
- Threads on GPU compute difference table algorithm.
- CPU and GPU computations are overlapped :
  while the $a_j(k)$ are being computed on GPU, the packets of a_{j+1} are simultaneously computed.

Polynomial approximation generation

Multi-precision on GPU

- Only needed operation : integer addition.
- Using C++ templates to generate multi-precision addition code (with word size fixed at compile time)
  $\Rightarrow$ enables loop unrolling.
- Template core in PTX (Nvidia assembly)
  $\Rightarrow$ enables efficient carry propagation.
- Word chunks are interleaved in GPU global memory
  $\Rightarrow$ enables coalesced memory accesses.

Benchmarks

Test platform

- One hex-core Intel X5650 CPU (with 2-way SMT)
- One Nvidia C2070 GPU

We measure real time : these include all computations (CPU + GPU) and CPU-GPU data transfers.

Three implementations

- Sequential C reference code (Lefèvre implementation)
- MPI implementation (Lefèvre implementation + MPI)
- CPU-GPU deployment

Benchmarks

Timings in seconds for exponential function in extended precision over $[1, 2[$

	Seq.	MPI	CPU-GPU	$\frac{\text{Seq.}}{\text{MPI}}$	$\frac{\text{Seq.}}{\text{CPU-GPU}}$
Pol. approx.	43300.81	5251.53	788.84	8.25	54.89
Lefèvre	36816.10	5292.67	2446.27	6.96	15.05
Regular	34039.94	4716.97	711.92	7.22	47.81
Lef. /Reg.	1.08	1.12	3.44		

Overall

- Regular on GPU over sequential Lefèvre $\Rightarrow$ speedup of 51.71x
- Pol. approx. + regular on CPU-GPU over pol. approx. + sequential Lefèvre $\Rightarrow$ speedup of 53.4x

Benchmarks

Timings in seconds for exponential function in extended precision over $[1, 2[$ and $[128, 256[$.

		Pol. approx.	Lefèvre	Regular
MPI	$[1, 2[$	5336.81	5292.67	–
	$[128, 256[$	11243.26	169911.90	–
CPU-GPU	$[1, 2[$	785.14	2446.78	711.8
	$[128, 256[$	1612.03	55627.92	61559.17
Speedup	$[1, 2[$	6.74x	2.18x	7.50x
	$[128, 256[$	6.97x	3.05x	2.76x

Remarks

- Pol. approx. speedup do not change when changing degree/precision
- Regular is slowed in $[128, 256[$
- Lefèvre is 10% more efficient than regular in $[128, 256[$

Benchmarks on the filtering efficiency

Filtering strategy

- Phase 1 : run HR-case non-existence test
- Phase 2 : if failed, subdivide the domain and run HR-case non-existence test on each subdomain (number of subdomain $\ll$ optimally $\gg$ chosen for each HR-case search)
- Phase 3 : exhaustively search HR-cases

Benchmarks on the filtering efficiency

Timings in seconds for HR-case search on exponential function in extended precision over $[1, 2[$ on GPU

	Lefèvre		Regular	
	Arguments	Time	Arguments	Time
Phase 1	$9.01 \cdot 10^{15}$	2372.60	$9.01 \cdot 10^{15}$	583.97
Phase 2	$3.19 \cdot 10^{13}$	61.31	$1.62 \cdot 10^{14}$	91.41
Phase 3	$7.65 \cdot 10^{10}$	11.02	$5.14 \cdot 10^{11}$	35.17

Remarks

- Most of computation time is spent in Phase 1
- Regular HR-case search speedup comes from Phase 1
- Regular HR-case search filters less than Lefèvre

fully computing quotients in CF $\Rightarrow$ placing more multiples of a on the unit segment

Benchmarks on the filtering efficiency

Timings in seconds for HR-case search on exponential function in extended precision over $[128, 256[$

	Lefèvre		Regular	
	Arguments	Time	Arguments	Time
Phase 1	$9.01 \cdot 10^{15}$	2308.02	$9.01 \cdot 10^{15}$	1634.67
Phase 2	$8.97 \cdot 10^{15}$	32646.50	$9.01 \cdot 10^{15}$	21443.60
Phase 3	$4.19 \cdot 10^{14}$	20673.40	$9.00 \cdot 10^{14}$	38480.90

Remarks

- Exp is not well approximated by deg. 1 in $[128, 256[$
  $\Rightarrow$ truncating to degree 1 strongly affects the filtering strategy
- Requires parallel prefix sums on GPU
- Most of computation time spent in Phases 2 and 3
- Hybrid Lefèvre/regular based on truncation error ?

Perspectives

1. Extend regular to higher degree polynomials
2. Parallelize LLL/SLZ

Extend regular HR-case search to higher degree : example

- $P(x) = ax^2 + bx + c$
- Change of variable : $X = x^2 + \frac{b}{a}x$
- Find Y such that $aY + c < \varepsilon \mod 2^e$ with regular HR-case search
- Use Hensel lemma to find y such that $Y = y^2 + \frac{b}{a}y$?

Perspectives

Parallelize LLL/SLZ

- Given $B = (b_1, \cdots, b_m)$ a basis of a lattice, find two $\ll$ short $\gg$ vectors.
- We need short vectors, not a reduced basis.
- Parallelize one LLL reduction or many LLL reductions ?
  $\Rightarrow$ We think one LLL by SIMD unit.
  $\Rightarrow$ To parallelize efficiently one LLL, we need independent computations.

Option 1

- Compute $r_i = \gcd_j(< b_i, b_j >)$ for each b_i in parallel.
- Randomly choose μ_i until $\sum_{i=0}^{m} \mu_i r_i \frac{b_i}{||b_i||}$ is in the lattice.
- What is the probability to find a vector in the lattice this way ?

Perspectives

Key consideration

We can build an orthogonal basis using Gram-Schmidt **in** the lattice

$$b_i^* = < b_{i-1}^*, b_{i-1}^* > b_i - < b_{i-1}^*, b_i > b_{i-1}^*.$$

Gram-Schmidt in the lattice

$$G = \begin{pmatrix} 1 & 0 & 0 \\ < b_1, b_2 > & < b_1, b_1 > & 0 \\ & < b_1, b_1 >< b_2, b_3 > & < b_1, b_1 >< b_2, b_2 > \end{pmatrix}$$

Remarks

- We can have shorter vectors using gcd.
- $|\mathrm{Det(G)}| \neq 1$: we deform the lattice
- Consider the deformed lattice as a deformation of B or a sub-lattice of B ?

Perspectives

Option 2

- Compute a Gram-Schmidt **in** the lattice to obtain an orthogonal basis of a sub-lattice
- We have m orthogonal plans
- Project the initial basis on each of these plans
- Reduce the vectors on these plans using Gauss algorithm independently on each plan
- Hope this leads to a short vector...

References I

- Valérie Berthé, *Autour du système de numération d'Ostrowski*, Bulletin of the Belgian Mathematical Society **8** (2001), 209–238.
- Vincent Lefèvre, *Moyens arithmétiques pour un calcul fiable*, Ph.D. thesis, École normale supérieure de Lyon, 2000.
- Noel Bryan Slater, *The distribution of the integers n for which* $\{\theta n\} < \phi$, Proceedings of the Cambridge Philosophical Society **46** (1950), 525–534.
- _____, *Gaps and steps for the sequence* $n\theta \mod 1$, Mathematical Proceedings of the Cambridge Philosophical Society (1967), 1115–1123.