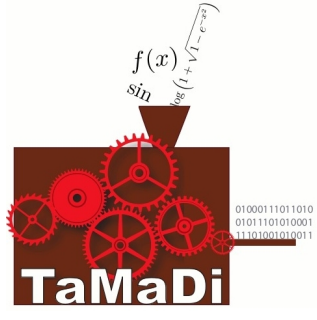
Towards solving the Table Maker's Dilemma on GPU

Deployment of Lefèvre's algorithm on GPU

Pierre Fortin, Mourad Gouicem, Stef Graillat

PEQUAN Team, LIP6/UPMC

ANR TaMaDi Project meeting
Lyon, December 13-14th 2011



The IEEE 754-2008 standard

Aim

Ensure predictable and portable numerical software.

Basic Formats

- single-precision (binary32)
- double-precision (binary64)
- quadruple-precision (binary128)

Rounding Modes

- Rounding to nearest
- Directed rounding (towards 0, $-\infty$ and $+\infty$)

Correctly rounded operations

$+, -, \times, /, \sqrt{}$

And for elementary mathematical functions ?

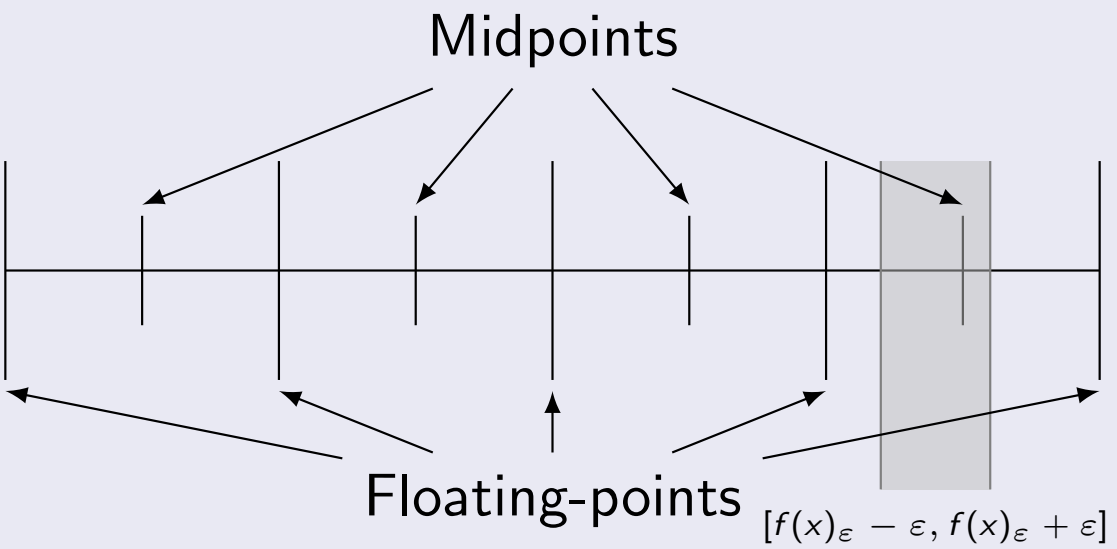
exp, log, sin, cos, tan, $\cdots$

$\Rightarrow$ IEEE-754-2008 only recommends correct rounding because of the Table Maker's Dilemma

Correct rounding

$$\circ_p(f(x)_\varepsilon) = \circ_p(f(x)_0)$$

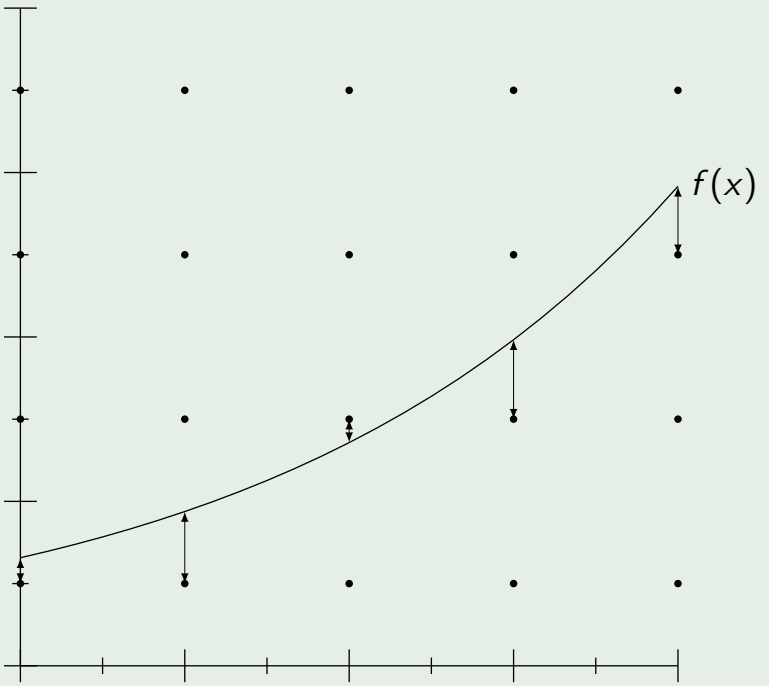
Hard-to-round case



The Table Maker's Dilemma

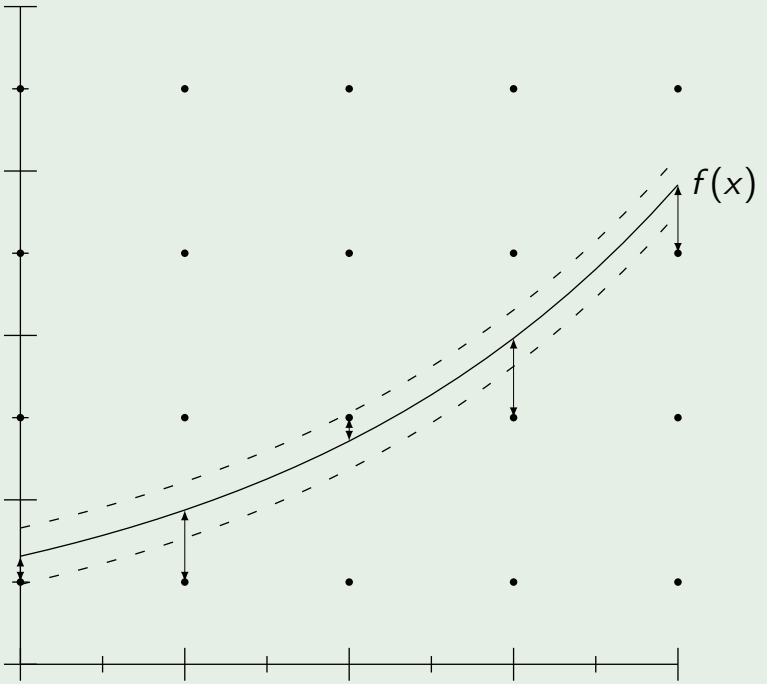
Given a function f defined over I and a rounding mode $\circ_p$, find ϵ such that $\forall x \in I$

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



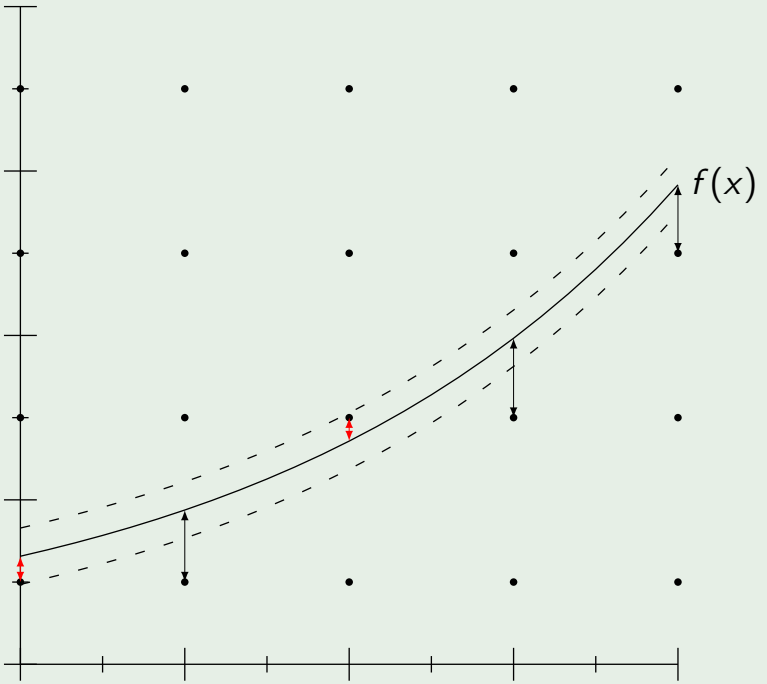
General Framework

1. Fix a suitable ε.



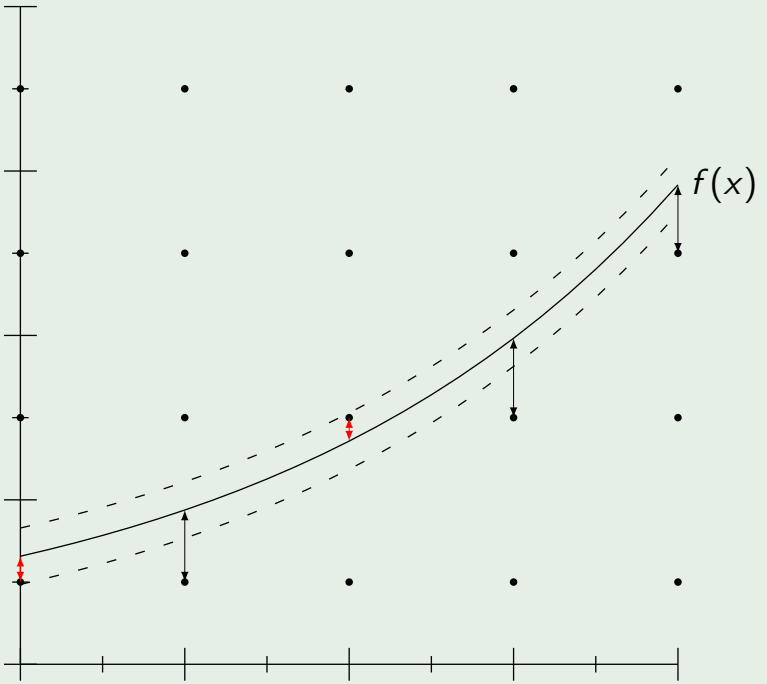
General Framework

1. Fix a suitable ε.
2. Search hard-to-round cases
   (exhaustive search, Lefèvre's algorithm, SLZ algorithm, ...).



General Framework

1. Fix a suitable ε.
2. Search hard-to-round cases
   (exhaustive search, Lefèvre's algorithm, SLZ algorithm, ...).
3. Find the hardness-to-round ϵ of f among the hard-to-round cases.



General Framework

1. Fix a suitable ε.
2. Search hard-to-round cases
   (exhaustive search, Lefèvre's algorithm, SLZ algorithm, ...).
3. Find the hardness-to-round ϵ of f among the hard-to-round cases.

Problem

- Very computationally intensive.
- Complexity exponential in the number of bits of the targeted format.

$\Rightarrow$ We focus on Lefèvre's algorithm on GPU : fine-grained and adapted to double-precision.

Lefèvre's algorithm

General Idea

1. Generation of affine approximations step : approximate $f : I \to \mathbb{R}$ by polynomials of degree 1 over intervals $J \subset I$.
2. HR-case search step : Filter arguments with an efficient test. If the Lower bound on the distance between a line Segment and a Grid is less than ε then Failure else Success (LSG test).

Deployment on GPU

- Generation of affine approximations step on CPU (because of multi-precision needs).
- HR-case search step ported on GPU (CUDA 4.0 + Fermi C2070).

HR-cases search on GPU

Filtering strategy

1. Compute LSG test on intervals J in $O(\log|J|)$,
2. if Failure then split J in 8 sub-intervals K, refine the affine approximation and run LSG test on each K in $O(\log|K|)$,
3. if Failure then exhaustively compute distances in each K in $O(|K|)$.

One kernel deployment

- One interval J per thread.
- Each thread computes the three phases.

Timings for searching HR-cases for *exp* in $[1, 1 + 2^{-13}]$ (2^{40} *doubles*)

Version		Time (s)
Exhaustive	CPU (1 core)	1079.41
	GPU (C2070)	33.23
Lefèvre's algorithm	CPU (1 core)	4.52
	GPU One kernel (C2070)	0.348

Speedups

- Exhausive search : CPU $\rightarrow$ GPU : x32.5
- Lefèvre's : CPU $\rightarrow$ GPU one kernel : x13.0

Problem

- Fewer and fewer intervals from one phase to the next.
  For *exp* in $[1, 1 + 2^{-13}]$ (2^{40} *doubles*, $\varepsilon = 2^{-96}$),

Phase	Number of intervals	Per thousand
1	$2^{25} \approx 33.55 \cdot 10^6$	1000‰
2	109048	3.25‰
3	2182	0.07‰
HR-cases	243	0.007‰

- Different phases $\rightarrow$ different computations.

Three kernel deployment

- Separate each phase in a different kernel.
- Write intervals for next phase consecutively (coalesced memory accesses) $\rightarrow$ atomic operations.
- Select optimal block size for each phase.

Deploy the filtering strategy

Times and register consumptions per thread for searching HR-cases for *exp* in $[1, 1 + 2^{-13}]$ (2^{40} *doubles*)

Version			Registers	Time (s)
Lefèvre's algorithm	X5650 (1 core)		-	4.52
	One kernel (C2070)		31	0.348
	Three kernels (C2070)	Phase 1	30	0.258
		Phase 2	37	0.006
		Phase 3	20	0.001
		Total	-	0.265

Speedups

- CPU $\rightarrow$ GPU one kernel : x13.0
- CPU $\rightarrow$ GPU three kernels : x17.1

Deploy the filtering strategy

Times and register consumptions per thread for searching HR-cases for *exp* in $[1, 1 + 2^{-13}]$ (2^{40} *doubles*)

Version			Registers	Time (s)
Lefèvre's algorithm	X5650 (1 core)		-	4.52
	One kernel (C2070)		31	0.348
	Three kernels (C2070)	Phase 1	30	0.258
		Phase 2	37	0.006
		Phase 3	20	0.001
		Total	-	0.265

Speedups

- CPU $\rightarrow$ GPU one kernel : x13.0
- CPU $\rightarrow$ GPU three kernels : x17.1

Two sources of divergence

Lefèvre's LSG test [Lefèvre2005]

input : $P(x) = ax + b$, ε, N

initialisation : $x \leftarrow \{a\}$; $y \leftarrow 1 - \{a\}$; $z \leftarrow \{b\}$;
$u \leftarrow 1$; $v \leftarrow 1$;

if $z < \varepsilon$ **then return** Failure;
while *True* **do**
  if $z < x$ **then**
    $q \leftarrow \lfloor x/y \rfloor$;
    $y \leftarrow y - q \times x$;
    $u \leftarrow u + q \times v$;
    if $u + v \geq N$ **then return** Success;
    $x \leftarrow x - y$; $v \leftarrow u + v$;
  else
    $z \leftarrow z - x$;
    if $z < \varepsilon$ **then return** Failure;
    $q \leftarrow \lfloor y/x \rfloor$;
    $x \leftarrow x - q \times y$; $v \leftarrow v + q \times u$;
    if $u + v \geq N$ **then return** Success;
    $y \leftarrow y - x$; $u \leftarrow u + v$;

Background on divergence

Execution model

- *Threads* are executed by *warp* of 32.
- Each *warp* is affected to a *Stream multiprocessor*.
- *Stream multiprocessors* are SIMD units.

Divergence

If the treads in a warp do not follow the same execution path (conditionals and loops), they *diverge*.
$\Rightarrow$ Their executions are serialized.

Divergence on the main loop

Lefèvre's LSG test [Lefèvre2005]

input : $P(x) = ax + b$, ε, N

initialisation : $x \leftarrow \{a\}\,;\ y \leftarrow 1 - \{a\}\,;\ z \leftarrow \{b\}\,;$
$u \leftarrow 1\,;\ v \leftarrow 1\,;$

if $z < \varepsilon$ **then return** Failure;
while *True* **do**
  if $z < x$ **then**
    $q \leftarrow \lfloor x/y \rfloor$;
    $y \leftarrow y - q \times x$;
    $u \leftarrow u + q \times v$;
    if $u + v \geq N$ **then return** Success;
    $x \leftarrow x - y\,;\ v \leftarrow u + v$;
  else
    $z \leftarrow z - x$;
    if $z < \varepsilon$ **then return** Failure;
    $q \leftarrow \lfloor y/x \rfloor$;
    $x \leftarrow x - q \times y\,;\ v \leftarrow v + q \times u$;
    if $u + v \geq N$ **then return** Success;
    $y \leftarrow y - x\,;\ u \leftarrow u + v$;

Divergence within the main loop

Lefèvre's LSG test [Lefèvre2005]

input : $P(x) = ax + b$, ε, N

initialisation : $x \leftarrow \{a\}$; $y \leftarrow 1 - \{a\}$; $z \leftarrow \{b\}$;
$u \leftarrow 1$; $v \leftarrow 1$;

if $z < \varepsilon$ **then return** Failure;
while *True* **do**
  if $z < x$ **then**
    $q \leftarrow \lfloor x/y \rfloor$;
    $y \leftarrow y - q \times x$;
    $u \leftarrow u + q \times v$;
    if $u + v \geq N$ **then return** Success;
    $x \leftarrow x - y$; $v \leftarrow u + v$;
  else
    $z \leftarrow z - x$;
    if $z < \varepsilon$ **then return** Failure;
    $q \leftarrow \lfloor y/x \rfloor$;
    $x \leftarrow x - q \times y$; $v \leftarrow v + q \times u$;
    if $u + v \geq N$ **then return** Success;
    $y \leftarrow y - x$; $u \leftarrow u + v$;

Divergence on the main loop

Loop divergence metric

Normalized mean deviation to the maximum (MDM)

$$1 - \frac{Mean(\{n_i, 0 \leq i < w\})}{Max(\{n_i, 0 \leq i < w\})}$$

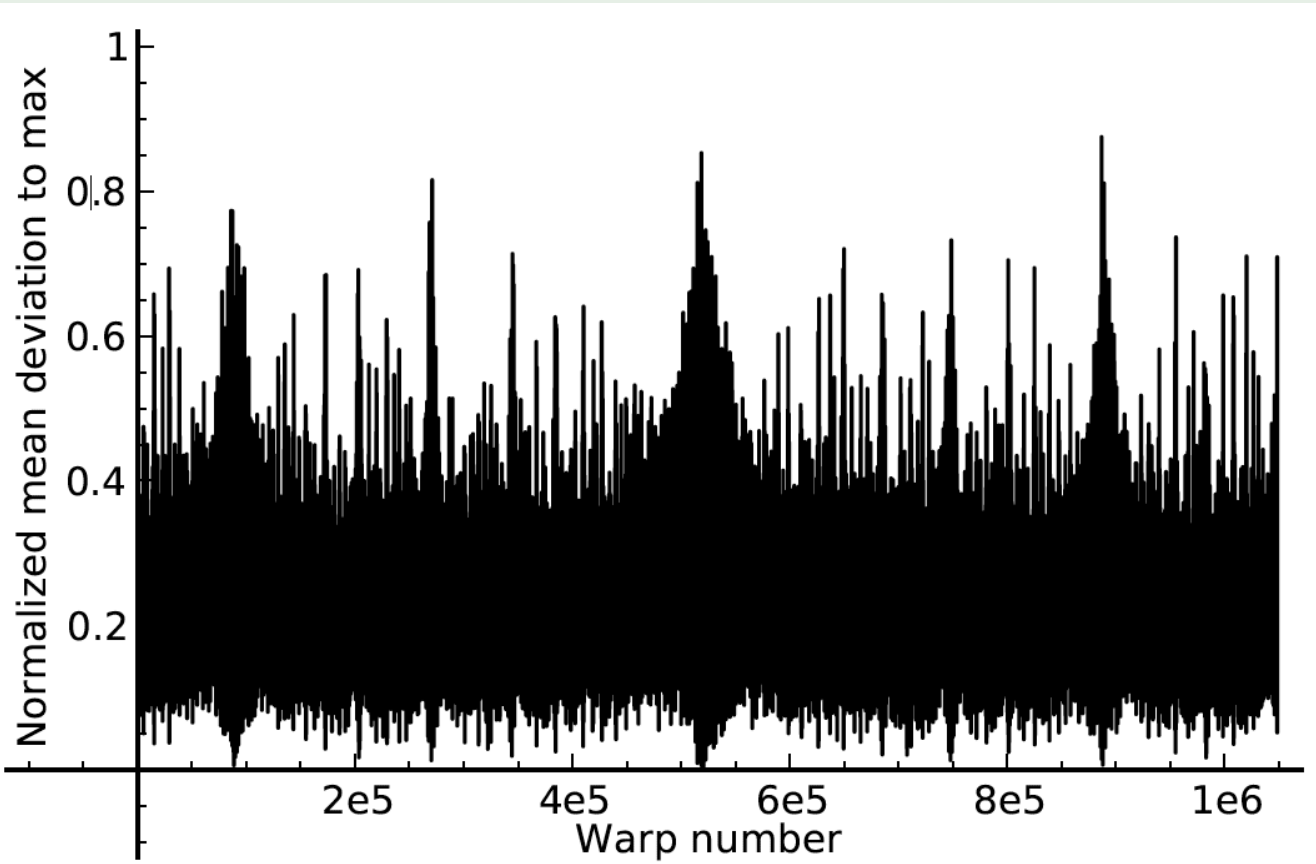
with :

- $w = 32$ the size of the warp,
- n_i the number of loop iteration of the ith thread of the warp.

Percentage of loop iterations a thread is idle within its warp.

Divergence on the main loop

Normalized MDM for Lefèvre's LSG and *exp* in $[1, 1 + 2^{-13}]$ (2^{40} *doubles*, 2^{25} LSG tests, 2^{20} warps)



Divergence on the main loop

Software Solution

- Re-organize data ⇒ No *a priori* information.
- Put reconvergence in the loop (not after)
  ⇒ compute LSG for several intervals per thread without exiting the loop.

Bottlenecks

- Uncoalesces the global memory accesses.
  Can this be offset by the Fermi C2070 cache ?
- Adds divergence within the loop.

Performance test : *exp* in $[1, 1 + 2^{-13}]$ (2^{40} *doubles*)

- 1.50 seconds : only 3.2 times faster than CPU ...

Divergence within the main loop

Lefèvre's LSG test [Lefèvre2005]

Initialisation :
$x \leftarrow a$; $y \leftarrow 1 - \{a\}$; $z \leftarrow b$;
$u \leftarrow 1$; $v \leftarrow 1$;

if $z < \varepsilon$ **then return** Failure;
while *True* **do**
  if $z < x$ **then**
    $q \leftarrow \lfloor x/y \rfloor$;
    $y \leftarrow y - q \times x$;
    $u \leftarrow u + q \times v$;
    if $u + v \geq N$ **then return** Success;
    $x \leftarrow x - y$; $v \leftarrow u + v$;
  else
    $z \leftarrow z - x$;
    if $z < \varepsilon$ **then return** Failure;
    $q \leftarrow \lfloor y/x \rfloor$;
    $x \leftarrow x - q \times y$; $v \leftarrow v + q \times u$;
    if $u + v \geq N$ **then return** Success;
    $y \leftarrow y - x$; $u \leftarrow u + v$;

Divergence within the main loop

Lefèvre's LSG test [Lefèvre2005]

Initialisation :
$x \leftarrow a$; $y \leftarrow 1 - \{a\}$; $z \leftarrow b$;
$u \leftarrow 1$; $v \leftarrow 1$;

if $z < \varepsilon$ **then return** Failure;
while *True* **do**
  if $z < x$ **then**
    $q \leftarrow \lfloor x/y \rfloor$;
    $y \leftarrow y - q \times x$;
    $u \leftarrow u + q \times v$;
    if $u + v \geq N$ **then return** Success;
    $x \leftarrow x - y$; $v \leftarrow u + v$;
  else
    $z \leftarrow z - x$;
    if $z < \varepsilon$ **then return** Failure;
    $q \leftarrow \lfloor y/x \rfloor$;
    $x \leftarrow x - q \times y$; $v \leftarrow v + q \times u$;
    if $u + v \geq N$ **then return** Success;
    $y \leftarrow y - x$; $u \leftarrow u + v$;

Divergence within the main loop

Lefèvre's LSG test [Lefèvre2005]

Initialisation :
$x \leftarrow a$; $y \leftarrow 1 - \{a\}$; $z \leftarrow b$;
$u \leftarrow 1$; $v \leftarrow 1$;

if $z < \varepsilon$ **then return** Failure;
while *True* **do**
  if $z < x$ **then**
    $q \leftarrow \lfloor x/y \rfloor$;
    $y \leftarrow y - q \times x$;
    $u \leftarrow u + q \times v$;
    if $u + v \geq N$ **then return** Success;
    $x \leftarrow x - y$; $v \leftarrow u + v$;
  else
    $z \leftarrow z - x$;
    if $z < \varepsilon$ **then return** Failure;
    $q \leftarrow \lfloor y/x \rfloor$;
    $x \leftarrow x - q \times y$; $v \leftarrow v + q \times u$;
    if $u + v \geq N$ **then return** Success;
    $y \leftarrow y - x$; $u \leftarrow u + v$;

SWAP function

input : x, y, u, v, *are_swapped*

$tmp \leftarrow x$;
$x \leftarrow y$;
$y \leftarrow tmp$;

$tmp \leftarrow u$;
$u \leftarrow v$;
$v \leftarrow tmp$;

$are\_swapped \leftarrow not(are\_swapped)$;

Divergence within the main loop

Lefèvre's LSG test [Lefèvre2005]

Initialisation :
$x \leftarrow a$; $y \leftarrow 1 - \{a\}$; $z \leftarrow b$;
$u \leftarrow 1$; $v \leftarrow 1$;

if $z < \varepsilon$ **then return** Failure;
while *True* **do**
  if $z < x$ **then**
    $q \leftarrow \lfloor x/y \rfloor$;
    $y \leftarrow y - q \times x$;
    $u \leftarrow u + q \times v$;
    if $u + v \geq N$ **then return** Success;
    $x \leftarrow x - y$; $v \leftarrow u + v$;
  else
    $z \leftarrow z - x$;
    if $z < \varepsilon$ **then return** Failure;
    $q \leftarrow \lfloor y/x \rfloor$;
    $x \leftarrow x - q \times y$; $v \leftarrow v + q \times u$;
    if $u + v \geq N$ **then return** Success;
    $y \leftarrow y - x$; $u \leftarrow u + v$;

Swapped version of LSG

Initialisation :
$x \leftarrow a$; $y \leftarrow 1 - \{a\}$; $z \leftarrow b$;
$u \leftarrow 1$; $v \leftarrow 1$; *are_swapped* $\leftarrow$ *false*;

if $z < \varepsilon$ **then return** Failure;
if $(z \geq x)$ **then**
  SWAP(x, y, u, v, *are_swapped*) ;
while *True* **do**
  if *are_swapped* **then**
    $z \leftarrow z - x$;
    if $z < \varepsilon$ **then return** Failure;
  $q \leftarrow \lfloor x/y \rfloor$;
  $y \leftarrow y - q \times x$;
  $u \leftarrow u + q \times v$;
  if $u + v \geq N$ **then return** Success;
  $x \leftarrow x - y$; $v \leftarrow u + v$;
  if *are_swapped xor* $(z \geq x)$ **then**
    SWAP(x, y, u, v, *are_swapped*) ;

Divergence within the main loop

Lefèvre's LSG test [Lefèvre2005]

Initialisation :
$x \leftarrow a$; $y \leftarrow 1 - \{a\}$; $z \leftarrow b$;
$u \leftarrow 1$; $v \leftarrow 1$;

if $z < \varepsilon$ **then return** Failure;
while *True* **do**
  if $z < x$ **then**
    $q \leftarrow \lfloor x/y \rfloor$;
    $y \leftarrow y - q \times x$;
    $u \leftarrow u + q \times v$;
    if $u + v \geq N$ **then return** Success;
    $x \leftarrow x - y$; $v \leftarrow u + v$;
  else
    $z \leftarrow z - x$;
    if $z < \varepsilon$ **then return** Failure;
    $q \leftarrow \lfloor y/x \rfloor$;
    $x \leftarrow x - q \times y$; $v \leftarrow v + q \times u$;
    if $u + v \geq N$ **then return** Success;
    $y \leftarrow y - x$; $u \leftarrow u + v$;

Swapped version of LSG

Initialisation :
$x \leftarrow a$; $y \leftarrow 1 - \{a\}$; $z \leftarrow b$;
$u \leftarrow 1$; $v \leftarrow 1$; *are_swapped* $\leftarrow$ *false*;

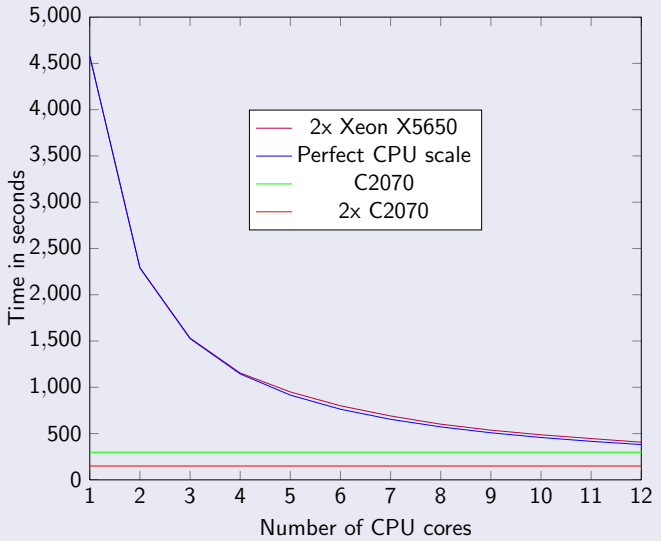
if $z < \varepsilon$ **then return** Failure;
if $(z \geq x)$ **then**
  SWAP(x, y, u, v, *are_swapped*) ;
while *True* **do**
  if *are_swapped* **then**
    $z \leftarrow z - x$;
    if $z < \varepsilon$ **then return** Failure;
  $q \leftarrow \lfloor x/y \rfloor$;
  $y \leftarrow y - q \times x$;
  $u \leftarrow u + q \times v$;
  if $u + v \geq N$ **then return** Success;
  $x \leftarrow x - y$; $v \leftarrow u + v$;
  if *are_swapped xor* $(z \geq x)$ **then**
    SWAP(x, y, u, v, *are_swapped*) ;

Divergence within the main loop

Performance test : *exp* in $[1, 1 + 2^{-13}]$ (2^{40} *doubles*, $\varepsilon = 2^{-96}$)

- CPU $\rightarrow$ GPU one kernel : x13.0
- CPU $\rightarrow$ GPU three kernels : x17.1
- CPU $\rightarrow$ GPU three kernels with swap : x18.2 (0.249 seconds)

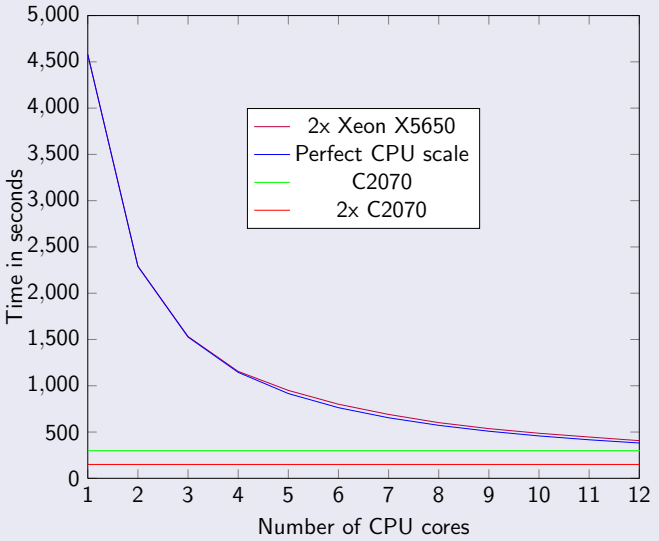
Comparison with multi-core CPU



Strong scaling tests

- Strong scaling over the 1024 intervals J in $[1; 1 + 2^{-3}]$ (2^{50} *doubles*).
- Cyclic decomposition with MPI.

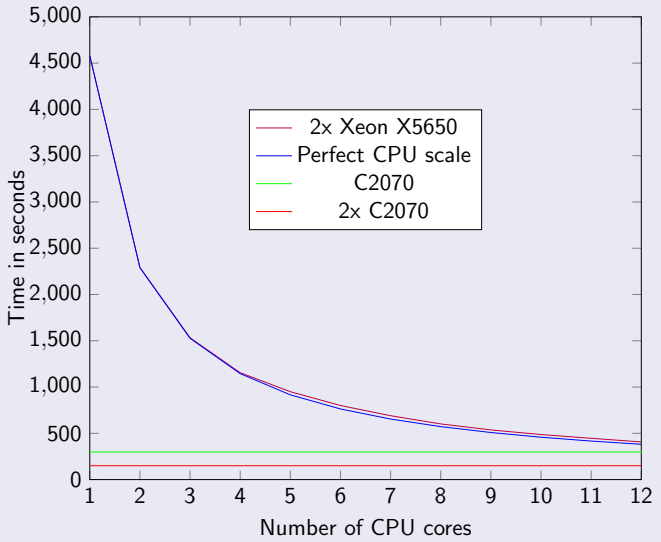
Comparison with multi-core CPU



Strong scaling tests

- On 12 cores CPU, very good speedup of 11.3
- On 2 GPU C2070, perfect speedup of 2.0

Comparison with multi-core CPU



Speedups

- 1xC2070 vs 1 X5650 core : x15.4
  $\Rightarrow$ Swap speedup depends on the intervals J.
- 1xC2070 vs 1xX5650 (6 cores) : x2.7 .
- 2xC2070 vs 2xX5650 (12 cores) : x2.7 .

Conclusion

Conclusion

- Software solutions to minimize :
  - divergence due to filtering strategy,
  - loop divergence,
  - conditional divergence.
- Speedup of 15.4 for Lefèvre's algorithm on GPU w.r.t. 1 CPU core.

Conclusion

Future works

- Generation of affine approximations on GPU.
  - Needs of multi-precision library.
  - Needs sharper bound on the extension of precision needed.
  - Best method known is intrinsically sequential.
- New LSG test algorithm which minimizes divergence on SIMD architectures.
- Deployment on GPU of the SLZ algorithm (based on the LLL algorithm).