Continued fractions and number systems: applications to correctly-rounded implementations of elementary functions and modular arithmetic.

Mourad Gouicem

PEQUAN Team, LIP6/UPMC

Nancy, France
May 28th 2013
1. Continued fraction expansion reminders

2. Application to correctly-rounded implementations of elementary functions

3. Application to modular arithmetic
1. Continued fraction expansion reminders

2. Application to correctly-rounded implementations of elementary functions

3. Application to modular arithmetic
Let a real $0 < \alpha < 1$. There exists a unique integer sequence $(k_i)_{i \in \mathbb{N}}$ with $k_i \in \mathbb{N}^*$ such that

$$\alpha = \frac{1}{k_1 + \frac{1}{k_2 + \frac{1}{\ddots}}} := [0; k_1, k_2, \ldots].$$

This sequence is finite if $\alpha$ is rational, or infinite otherwise.
We denote by:

- \((r_i)_{i \in \mathbb{N}}\) the real sequence of the *tails* of \(\alpha\) such that 
  \[\alpha = [0; k_1, k_2, \ldots, k_i + r_i];\]
- \((p_i/q_i)_{i \in \mathbb{N}}\) the rational sequence of the *convergents* such that 
  \[p_i/q_i = [0; k_1, k_2, \ldots, k_i];\]
- \((\theta_i)_{i \in \mathbb{N}}\) the real sequence of the such that 
  \[\theta_i = |q_i \alpha - p_i|;\]
We denote by:

- \((r_i)_{i \in \mathbb{N}}\) the real sequence of the *tails* of \(\alpha\) such that 
  \(\alpha = [0; k_1, k_2, \ldots, k_i + r_i];\)
- \((p_i/q_i)_{i \in \mathbb{N}}\) the rational sequence of the *convergents* such that 
  \(p_i/q_i = [0; k_1, k_2, \ldots, k_i];\)
- \((\theta_i)_{i \in \mathbb{N}}\) the real sequence of the such that 
  \(\theta_i = |q_i \alpha - p_i|;\)

**Leitmotif of the talk**

Use the fact that \(r_i = \theta_i/\theta_{i-1}\) to do modular arithmetic!
All sequences can be computed recursively:

\[
\begin{align*}
p_{-1} &= 1 & p_0 &= 0 & p_i &= p_{i-2} + k_ip_{i-1}, \\
q_{-1} &= 0 & q_0 &= 1 & q_i &= q_{i-2} + k_iq_{i-1}, \\
\theta_{-1} &= 1 & \theta_0 &= \alpha & \theta_i &= \theta_{i-2} - k_i\theta_{i-1}.
\end{align*}
\]

with \( k_i = \lfloor \theta_{i-2}/\theta_{i-1} \rfloor \).

\( k_i \) can be computed by subtraction (subtraction-based Euclidean algorithm) or by division (division-based Euclidean algorithm).
Continued fraction expansion reminders

Application to correctly-rounded implementations of elementary functions

Application to modular arithmetic
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, ···
⇒ IEEE-754-2008 only recommends correct rounding because of the Table Maker’s Dilemma
Correct rounding

\[ \circ_p(f(x) \pm \varepsilon) = \circ_p(f(x)) \]

Hard-to-round case

Midpoints

Floating-points

\[ [f(x) - \varepsilon, f(x) + \varepsilon] \]
Function Isolate(Exists?, P, D, depth, k)

Input: Exists? (P, D) test the existence of (p, ε') HR-cases of P in D, P an approximation of f in D, depth and k two integers

if Exists? (P, D) then
  if depth = 0 then
    return ExhaustiveSearch (P, D);
  else
    (D₁, ..., Dₖ) := Subdivide (D, k);
    (P₁, ..., Pₖ) := Refine (P, D, k);
    return \bigcup_i Isolate (Exists?, Pᵢ, Dᵢ, depth - 1, k);
else
  return \emptyset;
Function Isolate(Exists?, P, D, depth, k)

Input: Exists? (P, D) test the existence of (p, ϵ') HR-cases of P in D, P an approximation of f in D, depth and k two integers

if Exists? (P, D) then
    if depth = 0 then
        return ExhaustiveSearch (P, D);
    else
        (D₁, ..., Dₖ) := Subdivide (D, k);
        (P₁, ..., Pₖ) := Refine (P, D, k);
        return \bigcup_i Isolate (Exists?, Pᵢ, Dᵢ, depth - 1, k);

else
    return ∅;
Problem

Given $P \in \mathbb{R}[x]$, is there any $x \in [0, \#_p D]$ such that

$$P(x) \mod 1 < \epsilon.$$ 

Solutions (with $p$ the floating-point precision)

- Exhaustive search : $O(2^p)$;
- Lefêvre when $\deg P = 1 : O(2^{2p/3})$ intervals in $O(p^2)$;
- SLZ when $\deg P > 1 : O(2^{p/2})$ intervals in $O(\text{poly}(p, \deg P, \alpha))$.

Example of computation times

- $e^x$ in full domain and $p = 53$ with Lefêvre : 5 years of CPU time;
- $2^x$ in $[1/2, 1[$ and $p = 64$ with SLZ : 8 years of CPU time.
Challenges

- Binary128 is actually out of reach;
- compute the hardest-to-round case for each of the 32 functions recommended by the IEEE std 754-2008 in binary64;
- tackle any function in reasonable time in binary64;
- and certify the results.

Lefèvre HR-case search

- Efficient for binary64 in practice: all known hardest-to-round in binary64 have been generated by Lefèvre;
- Massively parallel;
- Fine-grain parallelism.

\{ Perfect for GPU! \}
Lefèvre HR-case existence test

Problem

Given \( b - ax \in \mathbb{R}[x] \), is there any \( x \in [0, \#_pD] \) such that

\[
b - ax \mod 1 < \epsilon.
\]

Geometrically

Is there any multiple of \( a \) in \( \{ ax \mod 1 | x \leq \#_pD \} \) at a distance less than \( \epsilon \) to the left of \( b \)?
The three distance theorem

Three distance theorem (Slater)
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.

Link with continued fraction expansion
Given \( a = [0; k_1, k_2, k_3, \ldots ] \) and \( \frac{p_i}{q_i} \) the \(i^{th}\) convergent.

- The lengths are the \( \theta_{i-1,t} = \theta_{i-1} - t \cdot \theta_i \), with \(0 \leq t < k_{i+1}\).
  - Their number are the \( q_{i-1,t} = q_{i-1} + t \cdot q_i \), with \(0 \leq t < k_{i+1}\).
- There are \(O(\log N)\) two-length configurations and they verify

\[
q_i \theta_{i-1,t} + q_{i-1,t} \theta_i = 1.
\]

- Interpretation: if we place \( N = q_i + q_{i-1,t} \) multiples of \( a \),
  - there are \(q_i\) intervals of length \(\theta_{i-1,t}\);
  - there are \(q_{i-1,t}\) intervals of length \(\theta_i\).
Example: \( a = \frac{14}{45} \)
Idea

Write $b$ in the basis $(\theta_{i,t})_{i \in \mathbb{N}}$ to get best approximations.

If $i$ is even

$(b - \tilde{b}_{i,t+1}) = (b - \tilde{b}_{i,t}) - \theta_i$ or $(b - \tilde{b}_{i+1,0}) = (b - \tilde{b}_{i,t})$

If $i$ is odd

$(b - \tilde{b}_{i+1,0}) = (b - \tilde{b}_{i,t}) - \theta_{i-1,t+1}$ or $(b - \tilde{b}_{i,t+1}) = (b - \tilde{b}_{i,t})$
Algorithm 1: Lefèvre HR-case existence test.

input : $b - a \cdot x$, $\epsilon''$, $N$

initialisation : $p \leftarrow \{a\}$; $q \leftarrow 1 - \{a\}$; $d \leftarrow \{b\}$; $u \leftarrow 1$; $v \leftarrow 1$;

if $d < \epsilon''$ then return Failure;

while True do
  if $d < p$ then
    $k = \lfloor q/p \rfloor$;
    $q \leftarrow q - k \cdot p$; $u \leftarrow u + k \cdot v$;
    if $u + v \geq N$ then return Success;
    $p \leftarrow p - q$; $v \leftarrow v + u$;
  else
    $d \leftarrow d - p$;
    if $d < \epsilon''$ then return Failure;
    $k = \lfloor p/q \rfloor$;
    $p \leftarrow p - k \cdot q$; $v \leftarrow v + k \cdot u$;
    if $u + v \geq N$ then return Success;
    $q \leftarrow q - p$; $u \leftarrow u + v$;

M. Gouicem

Continued fractions and Number systems applications 17 / 37
An irregular control flow: the SPMD on SIMD model

**SPMD on SIMD**
- Develop a scalar program: a *kernel*
- Launch multiple threads running the same *kernel*
- Group their execution on SIMD units by *warps* of 32 threads

**Control flow regularity**

```
switch (i) {
  i = 0 : ...  
  i = 1 : ...  
  i = 2 : ...  
  i = 3 : ...  
}
```
An irregular control flow: loop divergence

Normalized mean deviation to the maximum (NMDM)

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

Lefèvre existence test \( (e^x, [1, 1 + 2^{-13}], \epsilon = 2^{-32}) \)

No implementation trick works!

- Re-organize data \( \Rightarrow \) no \textit{a priori} information
- Compute several sub-domains per thread without exiting the loop \( \Rightarrow \) too few instructions to issue in the loop to offset the extra cost.
An irregular control flow

Why Lefèvre HR-case existence test is irregular?

It goes from subtraction-based to division-based Euclidean algorithm depending on the position of $b$.

⇒ The number of loop iterations is hence conditioned by:
- the position of $b$ on the unit segment,
- the number of quotient to compute and
- the value of the quotients.

Goal

Break this dependency by considering only $(i, 0)$ configurations.

⇒ Write $b$ in the basis $(\theta_i)_{i \in \mathbb{N}}$ to obtain the same sequence of best approximations.
New reduction rules

If \( i \) is even

\[
\begin{align*}
(b - \tilde{b}_{i+1}) &= (b - \tilde{b}_i) \mod \theta_i
\end{align*}
\]

If \( i \) is odd

\[
\begin{align*}
(b - \tilde{b}_{i+1}) &= (b - \tilde{b}_i) - \theta_{i+1} \mod \theta_i
\end{align*}
\]
Algorithm 2: Regular HR-case existence test.

input \( b - a \cdot x, \epsilon'', N \)

initialisation: \( p \leftarrow \{a\}; \quad q \leftarrow 1; \quad d \leftarrow \{b\}; \quad u \leftarrow 1; \quad v \leftarrow 0; \)

if \( d < \epsilon'' \) then return Failure;

while True do

  if \( p < q \) then
    \( k = \lfloor q/p \rfloor \);
    \( q = q - k \cdot p \); \( u = u + k \cdot v \);
    \( d = d \mod p \);
  else
    \( k = \lfloor p/q \rfloor \);
    \( p = p - k \cdot q \); \( v = v + k \cdot u \);
    if \( d \geq p \) then
      \( d = (d - p) \mod q \);
  
  if \( u + v \geq N \) then return \( d > \epsilon'' \);
A deterministic test

\[ i \text{ is alternatively odd and even.} \]
\[ \Rightarrow \text{We can avoid divergence by unrolling 2 loop iterations.} \]

**Algorithm 3:** Regular HR-case existence test unrolled.

**input:** \( b - ax, \epsilon'', N \)

**initialisation:** \( p \leftarrow \{a\}; \quad q \leftarrow 1; \quad d \leftarrow \{b\}; \)
\[ u \leftarrow 1; \quad v \leftarrow 0; \]

**while** True **do**

\[ k = \lfloor q/p \rfloor; \]
\[ q = q - k \times p; \quad u = u + k \times v; \]
\[ d = d \mod p; \]
\[ \text{if } u + v \geq N \text{ then return } d > \epsilon''; \]
\[ k = \lfloor p/q \rfloor; \]
\[ p = p - k \times q; \quad v = v + k \times u; \]
\[ \text{if } d \geq p \text{ then} \]
\[ \quad d = d - p \mod q; \]
\[ \text{if } u + v \geq N \text{ then return } d > \epsilon''; \]
Normalized mean deviation to the maximum (NMDM)

\[ 1 - \frac{\text{Mean}\left(\{n_i, 0 \leq i < w\}\right)}{\text{Max}\left(\{n_i, 0 \leq i < w\}\right)} \]

Lefèvre Algorithm

New Algorithm
A regular control flow

Why the regular HR-case existence test is regular?

It uses only division-based Euclidean algorithm.
⇒ The number of loop iterations only depend on the number of quotient to compute, which is very stable from one interval to the next.

<table>
<thead>
<tr>
<th>Method</th>
<th>Seq.</th>
<th>MPI</th>
<th>CPU-GPU</th>
<th>Seq. MPI</th>
<th>MPI CPU-GPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pol. approx.</td>
<td>43300.81</td>
<td>5251.53</td>
<td>788.84</td>
<td>8.25</td>
<td>6.66</td>
</tr>
<tr>
<td>Lefèvre</td>
<td>36816.10</td>
<td>5292.67</td>
<td>2446.27</td>
<td>6.96</td>
<td>2.16</td>
</tr>
<tr>
<td>Regular</td>
<td>34039.94</td>
<td>4716.97</td>
<td>711.92</td>
<td>7.22</td>
<td>6.63</td>
</tr>
<tr>
<td>Lef. /Reg.</td>
<td>1.08</td>
<td>1.12</td>
<td>3.44</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table: Performance result on $e^x$ in $[1, 2]$ for binary64 (Intel Xeon X5650 hexa-core, Nvidia C2070). Lefèvre MPI/New GPU: 7.4.
### Perspectives

**Remaining development**
- Argument reduction of periodic functions for large binades
- Implicit vectorization (OpenCL, ispc, …)

**Lefèvre HR-case existence test**
- Consider minimax approximations (libsollya) rather than Taylor to widen domains?
- Generalize Lefèvre test to higher degree polynomial (change of variable + Hensel lifting)?

**SLZ**
- Efficient parallel implementation of LLL
- Use structure of Coppersmith lattices
- Investigate structure in lattices involved in Coppersmith method over translated polynomials
1. Continued fraction expansion reminders

2. Application to correctly-rounded implementations of elementary functions

3. Application to modular arithmetic
**Notations**

- \((\theta_i)_{i \in \mathbb{N}}\) sequence of the \(|q_i \alpha - p_i|\) in CF,
- \((\theta'_i)_{i \in \mathbb{N}}\) sequence of the \(|q_i a - p_i d|\) in \(\text{extgcd}(a, d)\).

**Remark**

- CF is arithmetic modulo 1,
- CF over a rational \(a/d\) and \(\gcd(a, d)\) is identical, only difference is \(\theta'_i = d \cdot \theta_i\).

**Goal**

Use \((\theta_i)_{i \in \mathbb{N}}\) number scale to perform modular operations.
### Ostrowski integer number system

Given \((q_i)_{i \in \mathbb{N}}\) the denominators of the convergents of any irrational \(0 < \alpha < 1\), every positive integer \(b\) can be uniquely written as

\[
b = 1 + \sum_{i=1}^{m} b_i q_{i-1}
\]

where \(0 \leq b_1 \leq k_1 - 1, 0 \leq b_i \leq k_i, \text{ for } i \geq 2,\) \(b_i = 0\) if \(b_{i+1} = k_{i+1}\).

**Associated number scale over real numbers**: \(((−1)^i \theta_i)_{i \in \mathbb{N}}\).

**Decomposition algorithm**: greedy algorithm by default.
Signed Ostrowski integer number system

Given \((q_i)_{i \in \mathbb{N}}\) the denominators of the convergents of any irrational \(0 < \alpha < 1\), every positive integer \(b\) can be uniquely written as

\[
b = 1 + \sum_{i=1}^{m} (-1)^i b_i q_{i-1}
\]

where \(\begin{cases} 0 \leq b_1 \leq k_1 - 1, 0 \leq b_i \leq k_i, & \text{for } i \geq 2, \\ b_{i+1} = 0 & \text{if } b_i = k_i. \end{cases}\)

Associated number scale over real numbers : \((\theta_i)_{i \in \mathbb{N}}\).
Decomposition algorithm : greedy algorithm by excess.
Compute $c = a \cdot b \mod d$.

**Algorithm**

1. Compute the sequences $(\theta'_i)_{i \in \mathbb{N}}$ and $(q_i)_{i \in \mathbb{N}}$ from \(\text{extgcd}(a, d)\)
2. Compute the sequence $(b_i)_{i \in \mathbb{N}}$ such that $b = 1 + \sum_{i=1}^{m} b_i q_{i-1}$
3. Return $a + \sum_{i=1}^{m} b_i (-1)^i \theta'_{i-1}$

**Proof:** Let $\alpha = a/d$ and $b = 1 + \sum_{i=1}^{m} b_i q_{i-1}$.

$$\alpha \cdot b = \alpha + \sum_{i=1}^{m} b_i q_{i-1} \alpha$$

As $(-1)^i \theta_i = q_i \alpha - p_i$,

$$\alpha \cdot b = \alpha + \sum_{i=1}^{m} b_i (-1)^i \theta_{i-1} + \sum_{i=1}^{m} b_i p_{i-1}$$

By the uniqueness of the decomposition,

$$0 \leq \alpha + \sum_{i=1}^{m} b_i (-1)^i \theta_{i-1} < 1.$$

And finally, by multiplying by $d$, we get

$$0 \leq a + \sum_{i=1}^{m} b_i (-1)^i \theta'_{i-1} < d$$
Compute \( c = a^{-1} \cdot b \mod d \).

**Algorithm**

1. Compute the sequences \((\theta'_i)_{i \in \mathbb{N}}\) and \((q_i)_{i \in \mathbb{N}}\) from \(\text{extgcd}(a, d)\).
2. Compute the sequence \((b_i)_{i \in \mathbb{N}}\) such that
   \[
   b = a + \sum_{i=1}^{m} b_i (-1)^{i-1} \theta'_{i-1}
   \]
3. Return \(1 + \sum_{i=1}^{m} b_i q_{i-1}\)

Proof: Similar to modular multiplication.
Complexity considerations

Compute the sequences \((\theta'_i)_{i \in \mathbb{N}}\) and \((q_i)_{i \in \mathbb{N}}\) from \(\text{extgcd}(a, d)\)

\[O(\log(d)^2)\]

Compute the sequence \((b_i)_{i \in \mathbb{N}}\) from \((q_i)_{i \in \mathbb{N}}\) (or \((\theta'_i)_{i \in \mathbb{N}}\))

\[O(\log(d)^2)\]

Evaluate the sequence \((b_i)_{i \in \mathbb{N}}\) in \((\theta'_i)_{i \in \mathbb{N}}\) (or \((q_i)_{i \in \mathbb{N}}\))

\[O(\log(d)^2)\]
### Implementation considerations

#### Both algorithm
- Integrate multiplication and reduction
  $\Rightarrow$ we manipulate only words of size less than $\log d$;
- Quotients $k_i$ and $b_i$ can be computed only with subtractions as they are likely very small
  $\Rightarrow$ mean value is Khinchin constant $\approx 2.69$.

#### Modular Multiplication
- $(q_i)_{i \in \mathbb{N}}$ is an increasing sequence
  $\Rightarrow$ every $q_i \leq b$ need to be stored to decompose $b$
  $\Rightarrow$ the needed part of the sequence is of size $O(\log(b)^2)$.

#### Modular Division
- $(\theta_i)_{i \in \mathbb{N}}$ is an decreasing sequence
  $\Rightarrow$ we can decompose $b$ on-the-fly
  $\Rightarrow$ no extra storage is needed!
Algorithm enhancement

- Use other decompositions from Euclidean algorithm?
  - compute remainders with centered division
  - use decomposition from three distance theorem (irregular control flow? optimal?)
- Find mean optimal decomposition (minimizing $\sum |b_i| + |k_i|$)
- Use binary GCD algorithm to build the sequences $(\theta_i)_{i \in \mathbb{N}}$ and $(q_i)_{i \in \mathbb{N}}$ and avoid divisions?
### Implementation
- We have a 64 bits proof of concept C implementation (speedup of 1.5 – 2.5x over GMP). Now provide multiprecision.

### Searching application...
- Compact hardware implementation of modular arithmetic? (Jérémie?)
- When multiple modular mult/div by the same value a are needed (e.g. Gauss elimination)?
Thank you for your attention!!
Any question, remark, recommendation?


