Floating-Point Training
Module 2
Math Fundamentals

Paul Zimmermann, Inria Nancy

Plan of the training

- Module 1: The IEEE 754 Standard
- **Module 2: Math Fundamentals**
- Module 3: Core Algorithms
- Module 4: Elementary Function Approximation
- Module 5: Software Tools

Module 2: Math fundamentals

Basic error analysis together with introduction of ulp

Going from relative error from/to ulp error

Higham's notation

The cancellation problem

The double-rounding problem

Testing the radix and precision

Sterbenz's lemma

Absolute splitting

Error-free transformations

Absolute vs relative error

If $\hat{x}$ is a floating-point approximation of a (unknown) real x, the absolute error is bounded by ϵ if:

$$|\hat{x} - x| \leq \epsilon$$

The relative error is bounded by ϵ if:

$$|\hat{x} - x| \leq \epsilon \cdot |x|$$

In practice, it is more convenient to express the relative error in terms of $\hat{x}$:

$$|\hat{x} - x| \leq \epsilon \cdot |\hat{x}|$$

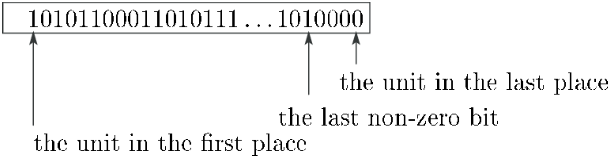
Sources of error

- mathematical error, for example when truncating a Taylor approximation
- rounding error, when evaluating this truncated Taylor approximation in precision p

We will discuss the mathematical error in Module 4.

Unit-in-last-place (ulp)

For a floating-point number, this is the weight of the last significant bit, or the distance between two consecutive floating-point numbers.



Warning: Definition may vary for powers of 2.

Example: in double-precision, $\mathrm{ulp}(0.5) = 2^{-53}$, $\mathrm{ulp}(1) = 2^{-52}$, $\mathrm{ulp}(2) = 2^{-51}$.

Extend to all real numbers x: if $a < x < b$ where a and b are two consecutive floating-point numbers, $\mathrm{ulp}(x) = b - a$.

Basic error analysis (1/2)

With rounding to nearest and any correctly rounded operation $y = \circ(x)$:

$$|y - x| \leq \frac{1}{2}\mathrm{ulp}(x)$$

With other rounding modes:

$$|y - x| < \mathrm{ulp}(x)$$

Basic error analysis (2/2)

Let $u = 2^{-p}$ for precision p.

With rounding to nearest and any correctly rounded operation $y = \circ(x)$:

$$|y - x| \leq u \cdot |x|$$

With other rounding modes:

$$|y - x| < 2u \cdot |x|$$

From relative error to ulp error

If we have a relative error of at most say $\varepsilon = 2^{-50}$ on a 53-bit value x, what is the maximal error in ulps?

Assume $x > 0$. The absolute error is bounded by $\varepsilon \cdot x$.

Since

$$2^{52}\mathrm{ulp}(x) \leq x < 2^{53}\mathrm{ulp}(x),$$

the absolute error is bounded by $2^{53}\varepsilon \cdot \mathrm{ulp}(x)$,

Thus for $\varepsilon = 2^{-50}$, it is bounded by 8 ulps.

From ulp error to relative error

Now if we have an error of at most 8 ulps, what is the maximal relative error?

Recall:

$$2^{52}\mathrm{ulp}(x) \le x < 2^{53}\mathrm{ulp}(x)$$

Since $\mathrm{ulp}(x) \le 2^{-52}x$, the relative error is bounded by $8 \cdot 2^{-52}x$, thus $2^{-49}x$.

Conclusion: when we go from relative error to ulp error and back, we lose a factor 2.

Try if possible to always work with relative error, or always with ulp error.

Ulp calculus

Rule 1. If x is normal:

$$2^{-p}|x| < \mathrm{ulp}(x) \leq 2^{1-p}|x|$$

Rule 2.

$$|a| \leq |b| \Longrightarrow \mathrm{ulp}(a) \leq \mathrm{ulp}(b)$$

Rule 3.

$$\mathrm{ulp}(x) \leq \mathrm{ulp}(\circ(x))$$

Rule 4.

$$y = \mathrm{RN}(x) \Longrightarrow |x - y| \leq \frac{1}{2}\mathrm{ulp}(x) \leq \frac{1}{2}\mathrm{ulp}(y)$$

Ulp calculus

Rule 5. If x and $2^k x$ are normal:

$$\mathrm{ulp}(2^k x) = 2^k \mathrm{ulp}(x)$$

Rule 6. If no underflow/overflow:

$$\frac{1}{2}|x|\mathrm{ulp}(y) < \mathrm{ulp}(xy) < 2|x|\mathrm{ulp}(y)$$

Rule 7. If $y_0 = x_0$, $y_1 = \circ(y_0 x_1)$, ..., $y_k = \circ(y_{k-1} x_k)$, where each rounding is done away from zero, the final error is bounded by $2k$ ulps.

Higham's notation

For any correctly rounded operation $y = \circ(x)$ we can write:

$$y = x(1 + \theta)$$

with $|\theta| \leq u$ for rounding to nearest, and $|\theta| \leq 2u$ for directed roundings.

u is the unit roundoff. In binary64, we have $u = 2^{-53}$.

Example

```
x = a * b
y = x * c
z = y * d
```

$$x = ab(1 + \theta_1)$$
$$y = xc(1 + \theta_2) = abc(1 + \theta_1)(1 + \theta_2)$$
$$z = yd(1 + \theta_3) = abcd(1 + \theta_1)(1 + \theta_2)(1 + \theta_3)$$

Theta-collapse rule 1:

Lemma

If $|\theta_i| \leq u$ *for* $1 \leq i \leq n$*, then*

$$(1 + \theta_1)(1 + \theta_2) \cdots (1 + \theta_n)$$

can be written $(1 + \theta)^n$ *for some* $|\theta| \leq u$*.*

Theta-collapse rule 2:

Lemma

If $nu < 1$ and $|\theta| \leq u$, then $(1+\theta)^n$ can be written $1 + 2n\theta'$ for $|\theta'| \leq u$.

Proof (sketch): it suffices to check for the largest possible values $\theta = \pm u$.

$$(1+\theta)^n = e^{n \log(1+u)}$$

For $\theta = u$, $\log(1+u) < u$, thus $e^{n\log(1+u)} < e^{nu} < 1 + 2nu$, using $e^x < 1 + 2x$ for $x < 1$.

For $\theta = -u$, $\log(1+u) > -2u$ (since $u \leq 1/2$), thus $e^{n\log(1+u)} > 1 - 2nu$ using $e^x \geq 1 + x$.

Going back to our example:

```
x = a * b
y = x * c
z = y * d
```

$$z = abcd(1+\theta_1)(1+\theta_2)(1+\theta_3)$$

Theta-collapse rule 1 gives:

$$z = abcd(1+\theta)^3$$

Theta-collapse rule 2 gives:

$$z = abcd(1+6\theta')$$

Thus the maximal relative error is $6u$.

This kind of analysis works well for multiplication or divisions, but what about additions or subtractions?

For numbers of same sign:

```
x = a + b
y = x + c
```

$$x = (a + b)(1 + \theta_1)$$

$$y = (x + c)(1 + \theta_2) = (a + b)(1 + \theta_1)(1 + \theta_2) + c(1 + \theta_2)$$

The error is bounded by:

$$(a + b)(2u + u^2) + cu \leq (a + b + c)(2u + u^2)$$

Thus we can write:

$$y = (a + b + c)(1 + \theta)^2$$

Cancellation

Unfortunately, this does not work when adding numbers of different signs, due to the cancellation issue.

```
x = a + b
y = x - c
```

The final error (for rounding to nearest) is bounded by $\frac{1}{2}\mathrm{ulp}(x)$ for the error on $a + b$, plus $\frac{1}{2}\mathrm{ulp}(y)$ for the error on $x - c$.

If y is a much smaller magnitude than x (cancellation in $x - c$), then the first error $\frac{1}{2}\mathrm{ulp}(x)$ is amplified.

If the exponent difference between x and y is say 10, then the rounding error on $a + b$ is multiplied by 2^{10}!

Another example

(Credit Claude-Pierre Jeannerod.) Approximate $(a+b)(c+d)$.

$$x = \circ(a+b) = (a+b)(1+\theta_1)$$

$$y = \circ(c+d) = (c+d)(1+\theta_2)$$

$$z = \circ(xy) = (a+b)(c+d)(1+\theta_1)(1+\theta_2)(1+\theta_3) = (a+b)(c+d)(1+\theta)^3$$

Good accuracy for z.

Yet another example

(Credit Claude-Pierre Jeannerod.) Approximate $ab + cd$.

$$x = \circ(ab) = ab \cdot (1 + \theta_1)$$

$$y = \circ(cd) = cd \cdot (1 + \theta_2)$$

$$z = \circ(x + y) = (ab(1 + \theta_1) + cd(1 + \theta_2))(1 + \theta_3)$$

$$z = (ab + cd)(1 + \theta_3) + (ab\theta_1 + cd\theta_2)(1 + \theta_3)$$

Relative error dictated by:

$$\frac{|ab| + |cd|}{|ab + cd|}$$

Reverse rule

$$x = \circ(a + b)$$

Higham's theta rule for $|\theta| \leq u$, where $u = 2^{-53}$ for rounding to nearest, and $u = 2^{-52}$ for directed roundings:

$$x = (a + b)(1 + \theta)$$

Reverse rule:

$$(a + b) = x(1 + \theta)$$

This means that the rounding error can be bounded by $(a + b)\theta$, but also by $x\theta$.

Follows from the fact that the rounding error is bounded (for rounding to nearest) by $\frac{1}{2}\mathrm{ulp}(x) \leq 2^{-53}|x|$.

Reverse rule

As a consequence, for a sequence of multiplications of divisions, the collapse rule can be reversed:

$$x = \circ(\circ(ab)c)$$

$$x = abc(1+\theta)^2$$

$$abc = x(1+\theta')^2$$

Double rounding

Let x be a real number. First round x in precision p:

$$y = \circ_p(x)$$

Then round y in precision $q < p$:

$$z = \circ_q(y)$$

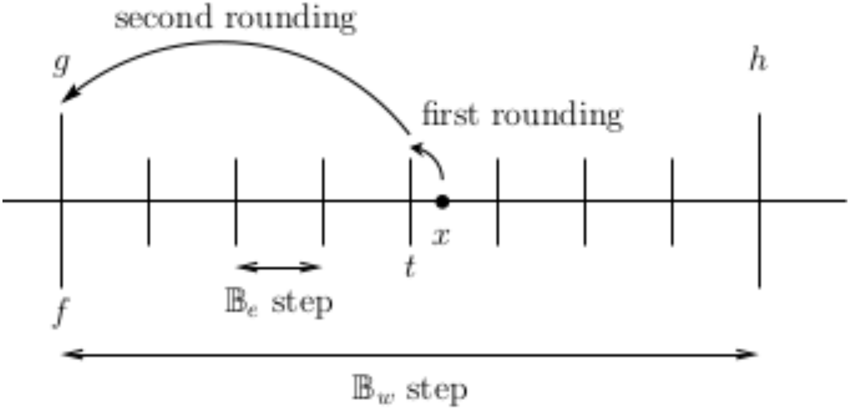
We can have $z \neq \circ_q(x)$.

Example (credit Vincent Lefèvre): $x = 2^{53} + 2$, $y = 1 - 2^{-16}$, with $p = 64$, $q = 53$.

Rounding in double precision: $x + y = 2^{53} + 3 - 2^{-16}$ is rounded to $2^{53} + 2$.

Rounding in extended precision: $x + y$ is first rounded to $2^{53} + 3$, then to $2^{53} + 4$ in double precision.

Double rounding



Credit: Boldo and Melquiond.

Round to nearest, whatever the rule to break ties, exhibits the double rounding issue.

Lemma

The double rounding issue can only happen in rounding to nearest, and when the second rounding has to break ties with the even rule.

Directed rounding: rounding boundaries in precision p include rounding boundaries in precision $q < p - 1$. Let u and u' be the closest rounding boundaries of x in precision p:

$$u \leq x \leq u'$$

Then if v and v' are the closest rounding boundaries of x in precision q, we have:

$$v \leq u \leq x \leq u' \leq v'$$

Thus if x rounds to u in precision p, then u rounds to v in precision q, which is the correct rounding of x in precision q.

Rounding to nearest: First round x in precision p:

$$y = \circ_p(x)$$

Then round y in precision $q < p$:

$$z = \circ_q(y)$$

Assume y is not a rounding boundary (in precision q).

Let v and v' be the two q-bit floating-point numbers enclosing y:

$$v \leq y \leq v'$$

Let $m = (v + v')/2$ be the rounding boundary in precision q, and u, u' the two p-bit numbers enclosing m:

$$v < u < m < u' < v'$$

Since $y \neq m$, necessarily $y \in [v, u]$, or $y \in [u', v']$.

If $y \in [v, u]$, then $v \leq x < (u + m)/2$: both x and y are rounded to v to precision q.

If $y \in [u', v']$, then $(m + u')/2 < x \leq v'$, both x and y are rounded to v' to precision q.

Double-rounding on the x87

Before the advent of SSE, floating-point operations on x86 were performed on the x87 coprocessor.

x87 performs single precision (precision of 24 bits), double (53 bits) and extended double (64 bits, aka `long double` in C).

For double precision computations, if the rounding precision is set to `long double`, computations are done internally in double extended, then stored to double. This was the default under Linux.

If the rounding precision is set to `double`, all computations are done in double precision. This was the default under Windows.

Thus the same program could yield different results under Linux or Windows!

Round-to-odd

A way to avoid the double-rounding issue. Not (yet) in IEEE 754!

Round-to-odd(x):

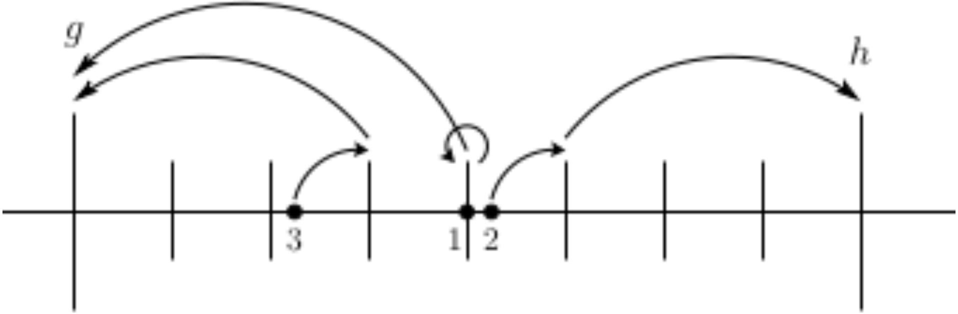
- if x is exactly representable in the target precision, return x;
- otherwise return the closest floating-point number with an odd significand.

	nearest, ties to even	round-to-odd
110100.000	110100	110100
110100.001	110100	110101
110100.100	110100	110101
110100.101	110101	110101
110101.000	110101	110101
110101.001	110101	110101
110101.100	110110	110101
110101.101	110110	110101

Round-to-odd

- $t = \circ_p^{\mathrm{odd}}(x)$
- $f = \circ_q(t)$

Why it works by the example:



Credit Boldo and Melquiond.

Testing the radix

Malcolm, 1972

```
sage: A = 1.0
sage: B = 1.0
sage: while ((A + 1.0) - A) - 1.0 == 0.0:
....:     A = 2.0 * A
sage: while ((A + B) - A) - B != 0.0:
....:     B = B + 1.0
sage: B
2.00000000000000
```

Testing the precision for radix β

```
sage: A = RR(1)
sage: p = 0; beta = 2
sage: while ((A + 1) - A) - 1 == 0:
....:     A = beta * A
....:     p = p + 1
sage: p
53
```

```
sage: A = RealField(17)(1)
sage: p = 0; beta = 2
sage: while ((A + 1) - A) - 1 == 0:
....:     A = beta * A
....:     p = p + 1
sage: p
17
```

Sterbenz's Lemma (1974)

Lemma

If a and b are two floating-point values (of same precision) such that $a \leq b \leq 2a$, then $\circ(b - a)$ is exact.

Proof: $b - a$ is an integer multiple of $\mathrm{ulp}(a)$, and $0 \leq b - a \leq a$.

Important: Sterbenz's Lemma holds whatever the rounding mode!

Sterbenz's Lemma in practice

```
sage: R=RealField(42)
sage: x=R(catalan)
sage: y=R(euler_gamma)
sage: x.hex()
'0xe.a7cb89f408p-4'
sage: y.hex()
'0x9.3c467e37dcp-4'
sage: z=x-y
sage: z.hex()
'0x5.6b850bbc2cp-4'
sage: t = x.exact_rational() - y.exact_rational()
sage: t == z.exact_rational()
True
```

Absolute splitting

How to get the integer part of a floating-point value x?

```
double C = 0x1.8p+53; // magic constant
double y = C + x;
double z = y - C;
```

It works for $|x| \leq 2^{52}$.

$C = 2^{53} + 2^{52}$ thus $2^{52} \leq C + x \leq 2^{53}$.

$\mathrm{ulp}(y) = 1$ thus y is rounded to an integer (according to the current rounding mode), and z is the integer part of x.

We then get the fractional part by $x - z$.

We will see a relative splitting in Module 3.

The drift phenomenon

Assume for round to nearest, we break ties away from zero (to minimize the relative error).

Let $x_0 = 1.2345$ (5-digit number).

Let $y = 0.00005$. At each step we subtract and add y.

$$x_{n+1} = (x_n \ominus y) \oplus y$$

$x_0 \ominus y = \circ(1.23445) = 1.2345$ (away rule)

$1.2345 \oplus y = \circ(1.23455) = 1.2346$ (away rule)

Thus $x_1 = 1.2346$, $x_2 = 1.2347$, $x_3 = 1.2348$, ...

This is the drift phenomenon.

Round-to-nearest-even avoids the drift phenomenon

$x_0 = 1.2345$, $y = 0.00005$

$$x_{n+1} = (x_n \ominus y) \oplus y$$

$x_0 \ominus y = \circ(1.23445) = 1.2344$ (even rule)

$1.2344 \oplus y = \circ(1.23445) = 1.2344$ (even rule)

We then have $x_n = 1.2344$ for all $n \geq 1$.

If $x_0 = 1.2344$ (even), we have $x_n = 1.2344$ for all $n \geq 0$.

Error-Free Transformations (EFTs)

The use of an FMA enables several error-free transformations, or to estimate the error in some operation.

EFT for product: if $h = \circ(xy)$ and $\ell = \circ(\mathrm{fma}(x, y, -h))$, then:

$$x \cdot y = h + \ell$$

whatever the rounding mode.

EFT for addition/subtraction

See algorithms TwoSum and FastTwosum (Module 3).

Square root

Let $h = \circ(\sqrt{x})$, $r = \circ(\mathrm{fma}(h, h, -x))$, and $\ell = \circ(-\frac{r}{2h})$, then

$$h + \ell \approx \sqrt{x}$$

We have $r \approx h^2 - x$ thus $x \approx h^2 - r$.

$$\sqrt{x} \approx h \cdot \sqrt{1 - \frac{r}{h^2}} \approx h \cdot (1 - \frac{r}{2h^2})$$

Division

Let $h = \circ(y/x)$ and $r = \circ(\mathrm{fma}(h, x, -y))$, and $\ell = \circ(r/x)$, then

$$h + \ell \approx \frac{y}{x}$$

We have $r \approx hx - y$ thus $y/x \approx h - r/x$.

Error analysis of loops

Example: Horner's evaluation

$$p(x) = a_0 + a_1 x + \cdots + a_n x^n$$

$p_n \leftarrow a_n$

for i from $n-1$ downto 0

  $p_i \leftarrow \circ(a_i + \circ(x p_{i+1}))$

Let ϵ_i be a bound on the rounding error on p_i where $\hat{p}_i$ is the approximation:

$$|\hat{p}_i - p_i| \leq \epsilon_i \cdot \mathrm{ulp}(\hat{p}_i)$$

1. try to obtain a recurrence on ϵ_i
2. solve the recurrence and deduce a bound for ϵ_0

Detailed example (1/3)

$$-\log(1-x)/x \approx 1 + \frac{x}{2} + \cdots + \frac{x^{n-1}}{n}$$

$p_n \leftarrow 0$

for i from $n-1$ downto 0

  $a_i \leftarrow \circ(1/(i+1))$

  $q_i \leftarrow \circ(xp_{i+1})$

  $p_i \leftarrow \circ(a_i + q_i)$

The error on a_i is bounded by 1 ulp.

The error on q_i is bounded by 1 ulp, plus the induced error on p_{i+1} multiplied by x:

$$\begin{aligned} \mathrm{err}(q_i) &\le \mathrm{ulp}(q_i) + |x|\epsilon_{i+1}\mathrm{ulp}(p_{i+1}) \\ &\le \mathrm{ulp}(q_i) + 2\epsilon_{i+1}\mathrm{ulp}(xp_{i+1}) \quad \text{[Rule 6]} \\ &\le (1+2\epsilon_{i+1})\mathrm{ulp}(q_i) \quad \text{[Rule 2]} \end{aligned}$$

Detailed example (2/3)

$a_i \leftarrow \circ(1/(i+1))$

$q_i \leftarrow \circ(xp_{i+1})$

$p_i \leftarrow \circ(a_i + q_i)$

$$\begin{aligned}\mathrm{err}(p_i) &\leq \mathrm{ulp}(p_i) + \mathrm{err}(a_i) + \mathrm{err}(q_i)\\ &\leq \mathrm{ulp}(p_i) + 2\mathrm{ulp}(p_i) + (1 + 2\epsilon_{i+1})\mathrm{ulp}(q_i)\\ &\leq (4 + 2\epsilon_{i+1})\mathrm{ulp}(p_i)\end{aligned}$$

$$|\frac{x^k}{k+1} + \frac{x^{k+1}}{k+2} + \cdots| \leq \frac{1}{k+1}|x^k + x^{k+1} + \cdots| \leq \frac{1}{2}\frac{|x|^{k-1}}{k} \quad \text{for } |x| \leq 1/3$$

Detailed example (3/3)

$$\epsilon_i \leq 4 + 2\epsilon_{i+1} \quad \text{with } \epsilon_n = 0$$

$$\epsilon_i + 4 \leq 2(\epsilon_{i+1} + 4)$$

Solution:

$$\epsilon_0 < 2^{n+2}$$

In conclusion, we need $n + 2$ guard bits.

Exercises

Exercise 1: In the error bound $|y - x| \leq u \cdot |x|$, with $u = 2^{-p}$. show that u can be replaced by $u/(1+u)$.

Exercise 2: prove that $h = \circ(xy)$, $\ell = \circ(\mathrm{fma}(x, y, -h))$ yields $x \cdot y = h + \ell$ whatever the rounding mode.

Exercise 3: prove that if $y = \circ(x)$, then $\mathrm{ulp}(x) \leq \mathrm{ulp}(y)$ (cf. Rule 4).

Exercise 4: prove Rule 6.

Exercise 5: prove that with $x = \circ(ab)$, $y = \circ(xc)$, $z = \circ(yd)$, the maximal relative error between z and $abcd$ is $4u$ (for u small enough).

Exercise 6: find a simple example in single precision where, due to cancellation, the relative error is huge.

Exercise 7: find an example where the double-rounding issue happens in single precision, with internal computations in double precision.

Exercise 8: if you have an old pocket calculator, test its radix β with Malcolm's algorithm, and test its precision (replacing $A = 2A$ by $A = \beta A$)

Exercise 9: prove that Sterbenz lemma holds for any radix β (but with the factor 2 in $a \leq b \leq 2a$ independent of the radix)

Exercise 10: in single precision, what is the magic constant C to get the integer part of x using absolute splitting?

Takeover message

As long as you use only basic operations, thanks to correct rounding, it is possible to bound the rounding errors.

If all values are of the same sign, after n operations, the error is roughly of n ulps (units in last place), thus using $\log_2 n$ guard bits is enough.

In case of cancellation, the induced error from previous roundings can become huge: whenever possible, try to avoid cancellation.

For loops, try to work out a recurrence for the rounding error.

References

Handbook of Floating-point Arithmetic, Jean-Michel et al., 2nd edition, Birkhäuser, 2018.

What Every Computer Scientist Should Know About Floating-Point Arithmetic, David Goldberg, 1991.

Accuracy and Stability of Numerical Algorithms, Nicholas J. Higham, SIAM books.

When double rounding is odd, Sylvie Boldo and Guillaume Melquiond, `https://hal.archives-ouvertes.fr/inria-00070603`.

The MPFR library: algorithms and proofs, `https://mpfr.org/algorithms.pdf`

Lectures by Jean-Michel Muller and colleagues, `https://ensl-m2info-fparith.gitlabpages.inria.fr/2025/`

Questions, comments ?

`Paul.Zimmermann@inria.fr`