rounding_errors

May 20, 2020

[1]:
```
restart
```

```
--loading configuration for package "FourTiTwo" from file
/home/hegland/.Macaulay2/init-FourTiTwo.m2
--loading configuration for package "Topcom" from file
/home/hegland/.Macaulay2/init-Topcom.m2
```

0.1 Rounding

One of the challenges of rational arithmetic is that the numerators and denominators can grow strongly during computations. One way to control this growth uses approximate computations and a rounding function. The rounding function is a real function and its range is the set $\mathbb{F}_t$ of floating point numbers, a subset of the ring of dyadic fractions (aka dyadic rationals).

Dyadic fractions are the extension $\mathbb{Z}[1/2]$ of the ring of integers by the fraction $\frac{1}{2}$. Any dyadic fraction can be (non-uniquely) represented by a mantissa $m \in \mathbb{Z}$ and an exponent $e \in \mathbb{N}$ as

$$x = m\, 2^{-e}.$$

The subset of dyadic fractions with $e = 0$ is the set of integers. One thus has

$$\mathbb{Z} \subset \mathbb{Z}[1/2] \subset \mathbb{Q}.$$

Like the rationals the dyadic fractions are dense in the set of real numbers. While they are not a field, they require around half the storage required for a rational number as the exponent typically is much smaller and thus has much less decimal digits than the denominator of a fraction.

The set of floating point numbers $\mathbb{F}_t$ is obtained by limiting the range of the mantissa:

$$\mathbb{F}_t = \{0\} \cup \{m\, 2^e \mid 2^{t-1} \leq |m| < 2^t,\ e, m \in \mathbb{Z}\}.$$

This representation of the floating point numbers is called "normalised". It is unique. The set of floating point numbers is not a ring and has one accumulation point 0. The set of floating point numbers contains a subset of the set of integers (not all of them) and is a subset of the set of dyadic fractions.

- The set $\mathbb{F}_t$ contains all integers between -2^t and 2^t.

  Proof: Choosing $e = 0$ on sees that it contains the integers between 2^{t-1} and 2^t and $e = -1 ... -(t-1)$ revealls that all the positive integers less than 2^{t-1} are also contained. Finally the set is symmetric and contains 0.

- Let $\mathbb{Z}_{2^t+1} = \{-2^{t-1}, \ldots, 2^{t-1}\}$. Then

$$\mathbb{F}_t = \{0\} \cup \bigcup_{e \in \mathbb{Z}} 2^e \mathbb{Z}_{2^{t+1}+1}.$$

  This leads to representations of the floating point numbers which are not unique.

- Consequently $2\mathbb{F}_t = \mathbb{F}_t$.

The rounding function $\rho_t : \mathbb{R} \to \mathbb{F}_t$ is based on a rounding function $\rho_0 : \mathbb{R} \to \mathbb{Z}$ which is provided by Macaulay. This function returns the closest integer to any given real argument x. In particular, this function satisfies $\rho_0(x) = 0$ if $x \in [0, 1/2)$ and $\rho_0(x) = 1$ if $x \in [1/2, 1)$.

Now let $x = y\,2^e$ for positive $x \in \mathbb{R}$ be such that $y \in [2^{t-1}, 2^t)$. (Like the normalised floating point representation, this representation of a positive real number is unique.) Then we define $\rho_t(x)$ to be

$$\rho_t(x) = \rho_0(y)\,2^e.$$

This representation is not normalised if $\rho_0(y) = 2^t$. Finally we set $\rho_t(0) = 0$ and $\rho_t(-x) = -\rho_t(x)$.

Then ρ_t has the following properties:

- $\rho_t(2^s x) = 2^s \rho_t(x)$ for any $s \in \mathbb{Z}$.

  For example one has $\rho_3(7.1) = 7$ as $7.1 \in [4, 8)$ thus $7.1 = 7.1 * 2^0$. Then $3.55 = 7.1 * 2^{-1}$ and thus $\rho_3(7.1/2) = 7/2$. The rounding function ρ_0 does not satisfy this property in general. One has $\rho_0(7.1) = 7 \neq 2 * \rho_0(3.55) = 8$.

  The property follows directly from the uniqueness of the representation of real positive numbers as $x = z\,2^e$ with $z \in [2^{t-1}, 2^t)$.

- For all $x \neq 0$ one has

$$\frac{|\rho_t(x) - x|}{|x|} \leq 2^{-t}.$$

  Proof: As in the definition let $x = y2^e$ where $2^{t-1} \leq |y| < 2^t$. Substituting the definition of $\rho_t(x)$ one then has

$$\frac{|\rho_t(x) - x|}{|x|} = \frac{|\rho_0(y) - y|}{|y|}$$

  As the rounding error of ρ_0 satisfies

$$|\rho_0(y) - y| \leq \frac{1}{2}$$

  and as $|y| \geq 2^{t-1}$ one gets the claimed inequality. QED

  Thus one has a bound for the relative error (the "number of digits") in contrast to ρ_0 which has a bounded absolute error of $1/2$.

We can now rewrite the error bound as

$$\rho_t(x) - x = \delta x$$

or

$$\rho_t(x) = (1 + \delta)x$$

for some real number δ with $|\delta| \leq 2^{-t}$.

0.1.1 An algebraic model for uncertainty

- the following code is for illustrative purposes and might contain (coding) errors

[2]:
```
-- choose the ring for your computations
R = QQ -- rational numbers
Rd = R[delta_1..delta_4]  -- delta_i parameters with uncertain value
```

```
o2 = Rd

o2 : PolynomialRing
```

[6]:
```
-- the ring for computations with uncertainty
-- here we consider affine polynomials of degree one
nu = numgens Rd
Lu = flatten(for i from 0 to nu-1 list
    (for j from i to nu-1 list Rd_i*Rd_j))
Iu = ideal(Lu)
Ru = Rd/Iu
```

```
o6 = Ru

o6 : QuotientRing
```

[7]:
```
-- a random uncertain expression
x = random(R) + sum(for i from 0 to nu-1 list Rd_i*random(R))
```

```
     3                  10         4
o7 = -delta  + delta  + --delta  + -delta  + 10
     2     1        2    7     3   9     4

o7 : Rd
```

[140]:
```
-- the expected value
mu  = coefficient(1_Rd,x)
err = x - mu
<< "expected value = " << promote(mu,RR) << endl;

-- the variance (uncertain variables are N(0,eps^2))
var = sum(for i from 0 to nu-1 list (coefficient(Rd_i,x))^2)
<< "variance factor = " << promote(var,RR) << endl;

-- the maximal error
--errmax = max(for i from 0 to nu-1 list abs(C_0_i))
errmax = max(for i from 0 to nu-1 list abs(coefficient(Rd_i,x)))
```

```
<< "max error factor = " << promote(errmax,RR) << endl;
```

```
expected value = 10

variance factor = 5.48835

max error factor = 1.5
```

[122]:
```
-- the rounding function which creates uncertainty

rho = (x,t,DF) -> (
    if x == 0 then return 0_DF
    else if x > 0 then (
        m = x; f=1_DF;
        while m < 2^(t-1) do (m=2*m; f=h*f);
        while m >= 2^t  do (m=m/2; f = 2*f);
        return round(m)*f)
    else return -rho(-x,t,h))

FR(rho(1/3,53,DF))-1/3 -- same as for floating point RR
```

```
o122 = -1.85037170770859e-17

o122 : RR (of precision 53)
```

lifting functions defined on $\mathbb{D}$ to $\mathbb{F}$

- simple formula for unary functions

$$f_{\mathbb{F}}(x) = \rho(E(f_{\mathbb{D}}(x))$$

- this is an approximation
- the same for binary functions

$$f_{\mathbb{F}}(x,y) = \rho(E(f_{\mathbb{D}}(x,y)))$$

- here E is the embedding of $\mathbb{D}$ into $\mathbb{Q}$, i.e., the function FQ from above
- one could also implement the rounding function on $\mathbb{D}$ instead of $\mathbb{Q}$

[131]:
```
t = 4
UF = f -> (x -> rho(FQ(f(x)),t,DF)) -- unary functions f
BF = f -> ((x,y) -> rho(FQ(f(x,y)),t,DF))-- binary functions f

x = rho(1/3, t, DF)  -- rho incurs an error
y = rho(1/4, t, DF)  -- rho incurse an error
errplus = FR((BF(plus))(x,y) - (x+y)) -- BF(plus) incurs an error
<< "error in addition of 1/3+1/4 = "<< errplus << endl;
<< "sum of errors in approx of 1/3 and 1/4 ="<< FR(x+y)-(1/3+1/4) <<endl;
```

```
error in addition of 1/3+1/4 = .03125

sum of errors in approx of 1/3 and 1/4 =.0104167
```