Arithmetic-Geometric Means for π
A formal study

Yves Bertot

February 2014

Objectives

- Already studied computations of *pi* through roots of cos, arctan, Archimedes
- Follow an exam from the French national selection exams for math teachers (5 hour exam, graduate level)
- Going further into mathematics (calculus, mainly),
- New aspects: more difficulties around derivatives and integration
- Use of square roots in computation
- Test the capabilities of Coq as a programming language

The context

- The arithmetic geometric mean algorithm
  - Take arbitrary a and b positive real numbers,
  - Set $a_0 = a,\ b_0 = b$,
  - Set $a_{n+1} = \frac{a_n + b_n}{2}$ and $b_{n+1} = \sqrt{a_n b_n}$,
- Properties:
  - $0 < n \Rightarrow b_n < a_n$
  - a_n and b_n converge fast towards a value $M(a, b)$

	a	b
0	1	.5
1	0.75	070...
2	0.7285	0.7282
3	0.72839552	0.72839550
4	0.728395515523453 4	0.728395515523453 3

Derivatives of arithmetic geometric mean

- Consider the case $a_0 = 1\ b_0 = x$,
- Specialize to $f_n(x) = a_n(1, x)$,
- f_n converges uniformly towards $f(x) = M(1, x)$,
- the function f is derivable with the property:

$$\pi = 2\sqrt{2}\frac{f^3(\frac{1}{\sqrt{2}})}{f'(\frac{1}{\sqrt{2}})} = 2\sqrt{2}\lim_{n\rightarrow\infty}\frac{b_n^2\left(1,\frac{1}{\sqrt{2}}\right)a_n\left(1,\frac{1}{\sqrt{2}}\right)}{a_n'\left(1,\frac{1}{\sqrt{2}}\right)}$$

- This property is established by studying *elliptic curves*

Algorithm for π (one variant)

- $y_n(x) = \dfrac{a_n(x)}{b_n(x)} \qquad z_n(x) = \dfrac{b_n'(x)}{a_n'(x)}$
- $y_{n+1}(x) = \dfrac{1 + y_n(x)}{2\sqrt{y_n(x)}} \qquad z_{n+1}(x) = \dfrac{1 + y_n(x) z_n(x)}{(1 + z_n(x))\sqrt{y_n(x)}}$
- $y_0(\frac{1}{\sqrt{2}}) = \sqrt{2} \qquad z_1(\frac{1}{\sqrt{2}}) = \sqrt{\sqrt{2}}$
- $\pi_0 = 2 + \sqrt{2} \qquad \pi_{n+1} = \pi_n \dfrac{1 + y_{n+1}(\dfrac{1}{\sqrt{2}})}{1 + z_{n+1}(\dfrac{1}{\sqrt{2}})}$
- for $1 \leq n$ $0 \leq \pi_{n+1} - \pi \leq \dfrac{4\pi_0}{500^{2^n}}$

Formalization context

- Started with Coq's standard library
- Important tactic: psatzl (F. Besson)
  - Not using the `sos` extension: trouble installing `csdp`
- Switch to Coquelicot (Boldo, Lelay, Melquiond)
  - Drawback: unstable library
  - Drawback: result more complex to distribute
  - Advantage: using others' work
  - Advantage: More regular collections of theorems

General purpose contributions

- A variable change theorem for Riemann integrals
- Improper integrals in the style of Coq's standard library
  - A function is `up_infinite_integrable` if $\int_a^b f(x)dx$ has a limit when b grows to ∞
  - `upInt f a h` is the value (`h` is the proof)
  - the same for `down_infinite_integrable` and `infinite_integrable`
  - General theorems: *Chasles*, linearity, improper integral of x^{-k}, bounds, extensionality
- A study of `arcsinh`
- **Dini** : every increasing sequence of continuous functions converging pointwise to a continous function converges uniformly,

Example lemma: variable change in integrals

```
Lemma RiemannInt_variable_change : (* standard *)
  forall f g g' a b (ab : a <= b)
   (h : Riemann_integrable
         (fun x => g' x * f (g x)) a b)
   (h' : Riemann_integrable f (g a) (g b)),
   (forall x, a <= x <= b -> is_derive g x (g' x)) ->
   (forall x, a <= x <= b -> g a <= g x <= g b) ->
   (forall x, g a <= x <= g b -> continuity_pt f x) ->
   (forall x, a <= x <= b -> continuity_pt g' x) ->
   RiemannInt h = RiemannInt h'.
```

Note that the type of `h` and `h'` is used to know what is computed in `RiemannInt`

Definition of arithmetic geometric mean sequence

- a simple recursive function returning a pair of values

```
Fixpoint ag (a b : R) (n : nat) :=
  match n with
    0%nat => (a, b)
  | S p => let (a_p, b_p) := ag a b p in
             ((a_p + b_p)/2, sqrt(a_p * b_p))
  end.
```

- Proofs that the two sequences are monotonous and converge are easy.

Link to elliptic integrals

- Elliptic integrals come in the form of

$$I(a,b) = \int_0^{\infty} \frac{\mathrm{d}t}{\sqrt{(t^2+a^2)(t^2+b^2)}}$$

- Proved equality :

$$I(a,b) = I(\frac{a+b}{2}, \sqrt{ab})$$

- and also

$$I(a,b) = \int_0^{\frac{\pi}{2}} \frac{\mathrm{d}x}{\sqrt{a^2\cos^2 x + b^2 \sin^2 x}}$$

- proofs using ε reasoning: show that the difference is smaller than any positive ε
- Both proofs rely on variable changes, need 300 and 200 lines of proof.

Commuting derivation and integrals

- Important contribution from the `coquelicot` library

$$\frac{\mathrm{d}\int_u^v f(w,t)\mathrm{d}t}{\mathrm{d}w}(x) = \int_u^v \frac{\mathrm{d}f(w,t)}{\mathrm{d}w}(x)\mathrm{d}t$$

(under the right conditions for f around x)

- Used to show that $I(a,b) = \frac{\pi}{2M(a,b)}$
- This is a formula for computing elliptic integrals when π is known (attributed to Gauß)

Behavior for (1, b) and b close to 0

- Through another variable change we have:

$$\int_0^\infty \frac{\mathrm{d}t}{\sqrt{(t^2+1)(t^2+b^2)}} = 2\int_0^{\sqrt{b}} \frac{\mathrm{d}t}{\sqrt{(t^2+1)(t^2+b^2)}}$$

$$\int_0^{\sqrt{b}} \frac{\mathrm{d}t}{\sqrt{(t^2+1)(t^2+b^2)}} \sim arcsinh(\sqrt{(}b)) \qquad \text{for } b \to 0^+$$

- by direct reasoning on agm we have:

$$2^n f\left(\frac{\sqrt{a_n(1,x)^2 - b_n(1,x)^2}}{a_n(1,x)}\right) = \frac{f(\sqrt{1-x^2})}{f(x)}$$

Link to derivatives

$$f(x) \sim \frac{-\pi}{2ln(x)} \qquad \text{for } x \to 0^+$$

- With equivalences and equalities from the previous slides

$$\lim_{n\to\infty} 2^{-n} ln\left(\frac{a_n(1,x)}{\sqrt{a_n(1,x)^2 - b_n(1,x)^2}}\right) = \frac{\pi}{2}\frac{f(x)}{f(\sqrt{1-x^2})}$$

- Studying separately the derivatives of the left hand side and deriving directly the right-hand-side

$$\frac{f^2(x)}{x(1-x^2)} = \frac{\pi}{2}\frac{f'(x)f(\sqrt{1-x^2}) - \frac{-x}{\sqrt{1-x^2}}f(x)f'(\sqrt{(}1-x^2))}{f^2(\sqrt{1-x^2})}$$

Main derivative formula

- at $x = \frac{1}{\sqrt{2}}$ the last formula simplifies greatly into:

$$\pi = 2\sqrt{2}\frac{f^3(\frac{1}{\sqrt{2}})}{f'(\frac{1}{\sqrt{2}})}$$

- To compute the ratio, we can compute approximations of f and f' as approximated by $a_n(1,x)$ and $b_n(1,x)$ and their derivatives
- More efficient to work with $y_n = \frac{a_n}{b_n}$ and $z_n = \frac{b'_n}{a'_n}$
- Use the Dini theorem to express that a_n converges uniformly towards f.

Abstract description

```
Fixpoint agmpi n :=
  match n with
    0%nat => (2 + sqrt 2)
  | S p => agmpi p * (1 + y_ n (/sqrt 2))
               / (1 + z_ n (/sqrt 2))
  end.
```

Concretely computing a large number of decimals

- Computing with large integers
  - To compute at precision $\frac{1}{p}$, multiply all values by p
- Théry et al. provide numbers as binary trees whose leaves are 31bit words,
  - Still not comparable to GMP : no arrays, more memory consumption
- Fast square roots re-implemented by Théry from previous work by Zimmermann, Magaud, & B.
- Fast execution provided by Dénès' native computation.
  - just-in-time compilation and execution directly inside Coq
- Work yet to be completed: formal proofs about error composition

Fixed precision computation

```
Definition hp1 :=
  (*some large integer*) (2 ^ precision)%bigZ.
Definition invhp x := (hp1 * hp1 / x)%bigZ.
Definition sqrthp x := BigZ.sqrt (x * hp1).
Definition mulhp x y := ((x * y) / hp1)%bigZ.
Definition addhp x y := (x + y)%bigZ.

Notation "x + y" := (addhp x y) : hp_scope.
Notation "x * y" := (mulhp x y) : hp_scope.
Notation "x / y" := (mulhp x (invhp y)) : hp_scope.

Delimit Scope hp_scope with H.
```

Concrete implementation of algorithm

```
Fixpoint agmpi n :=
  match n with
    0%nat => ((hp2 + (sqrthp hp2))%H, y1, z1)
  | S p =>
    let '(pip, yn, zn) := agmpi p in
    let sy := sqrthp yn in
    let zn1 := (hp1 + zn)%H in
      ((pip * ((hp1 + yn)%H / zn1)%H)%H,
       ((hp1 + yn)%H / (hp2 * sy)%H)%H,
       ((hp1 + (yn * zn)%H)%H / (zn1 * sy)%H)%H)
end.
```

Error analysis

- $$0 \leq \pi_n - \pi \leq \frac{4\pi_0}{500^{2^n}}$$
- Square root and division computations on integers
  - rounding by default (towards 0)

$$\sqrt{x} - e < \boxed{\mathrm{sqrt}(x)} \leq \sqrt{x}$$

$$\frac{x}{y} - e < \boxed{\frac{x}{y}} \leq \frac{x}{y}$$

- A theorem to control error propagation

$$e < \frac{e'}{3} \quad \wedge \quad |\boxed{y_n} - y_n| < e' \quad \Rightarrow \quad |\boxed{\frac{1 + \boxed{y_n}}{2\boxed{\mathrm{sqrt}(\boxed{y_n})}}} - \frac{1 + y_n}{2\sqrt{y_n}}| < e'$$

Several attempt for error estimation in y_n

- First attempts looking like naive interval arithmetics
  - with bisection
- Last attempt using derivative and *mean value theorem*
  - error cancellation improves as y_n gets closer to 1
  - derivative of $\dfrac{1+y}{2\sqrt{y}}$ is $\dfrac{y-1}{4y\sqrt{y}}$
- So if error on y_n is within a few `ulp`, the error on y_{n+1} will stay the same

To be continued

- Need the same kind of error control for z_n
- Need to propagate the errors trough the repeated multiplications
- Maybe replace square root with one rounding up

Running the program

- Write (in Coq) a program enumerating the digits
- Write (in Ocaml) the printing of the digits

Packaging the function call

```
Inductive bin := L (x : Z) | N (t1 t2 : bin).

Fixpoint ntb (x:bigZ) (n : nat) (b : bigZ) :=
match n with
  O => L (BigZ.to_Z x)
| S p => let (y, z) := BigZ.div_eucl x (10 ^ b) in
  N (ntb y p (BigZ.div b 2)) (ntb z p (BigZ.div b 2))
end.

Definition digits rank :=
let hp1 := (10 ^ (2 ^ bigZ_of_nat rank))%bigZ in
let hp2 := (hp1 + hp1)%bigZ in
ntb (pi hp1 hp2 rank) rank
   (2 ^ bigZ_of_nat (pred rank))%bigZ.

Extraction "pi.ml" digits.
```

Lessons learned

- Corpus of known facts is really growing
- Navigating libraries, `ssrflect`, Coq standard library, `coquelicot`, *Need for streamlining*
- Reflexions on more advanced tactics
  - Automatic proofs of positivity for simple expressions
  - Automatic proofs of derivability, continuity
- Extracted code may be less efficient than code inside Coq
- Hope to continue the efforts towards an imperative implementation