Complete Elliptic Integrals and the Arithmetic–Geometric Mean

Manuel Eberl

July 8, 2026

Abstract

This entry builds up two theories and connects them. The first consists of the *complete elliptic integrals* of the first and second kind

$$K(x) = \int_0^{\frac{\pi}{2}} (1 - x\sin(t)^2)^{-\frac{1}{2}}\,\mathrm{d}t \quad E(x) = \int_0^{\frac{\pi}{2}} (1 - x\sin(t)^2)^{\frac{1}{2}}\,\mathrm{d}t$$

for real or complex x.

The second one is the *arithmetic-geometric mean* function $\mathrm{agm}(x, y)$, which is defined as the limit of the sequence obtained by replacing the pair (x, y) with the pair consisting of the arithmetic and geometric means of x and y, i.e. $(x, y) \mapsto (\frac{1}{2}(x + y), \sqrt{xy})$.

The two theories are then connected by proving (among other things) that:

$$\mathrm{agm}(a, b) = \frac{\pi a}{2K((a^2 - b^2)/a^2)}$$

Various other important properties are shown, e.g.:

- Continuity, derivatives, antiderivatives of K and E as well as their relation to the hypergeometric function ${}_2F_1$
- Legendre's identity

$$K(x)E(1 - x) + E(x)K(1 - x) - K(x)K(1 - x) = \frac{\pi}{2}$$

- The convergence of the AGM iterations, including uniform convergence and an estimate of the speed of convergence
- Upward and downward identities for K and E, e.g.

$$K(x^2) = \frac{K\left(\frac{4x}{(1+x)^2}\right)}{1 + x}$$

- The relationship of the AGM to the Jacobi theta functions
- The Brent–Salamin algorithm to approximate π via AGM iterations (abstractly, without rounding error analysis)

Contents

1 Auxiliary material

theory *AGM_Lemma_Bucket*
  imports *"HOL-Complex_Analysis.Complex_Analysis"*
begin

lemma *asymp_equiv_nhds_iff: "f ∼[nhds x] g ⟷ f ∼[at x] g ∧ f x = g x"*
  ⟨*proof*⟩

lemma *real_nonpos_Reals_eq [simp]: "$\mathbb{R}_{\leq 0}$ = {x::real. x ≤ 0}"*
  ⟨*proof*⟩

lemma *nonpos_Reals_one_over_iff: "1 / (x :: 'a :: real_div_algebra) ∈ $\mathbb{R}_{\leq 0}$ ⟷ x ∈ $\mathbb{R}_{\leq 0}$"*
  ⟨*proof*⟩

lemma *gbinomial_of_real:*
  "(of_real x gchoose k :: 'a :: {real_div_algebra, field_char_0}) = of_real (x gchoose k)"
  ⟨*proof*⟩

lemma *conv_radius_of_real [simp]:*
  "conv_radius (λn. of_real (f n) :: 'a :: {banach, real_normed_algebra_1}) = conv_radius f"
  ⟨*proof*⟩

1.1 Uniform limits and uniform continuity

lemma *uniformly_continuous_on_fst [continuous_intros]:*
  "uniformly_continuous_on (A :: (real × real) set) fst"
  ⟨*proof*⟩

lemma *uniformly_continuous_on_snd [continuous_intros]:*
  "uniformly_continuous_on (A :: (real × real) set) snd"
  ⟨*proof*⟩

lemma *uniform_limit_compose':*
  assumes *"uniform_limit A f g F"* **and** *"h ‘ B ⊆ A"*
  shows *"uniform_limit B (λn x. f n (h x)) (λx. g (h x)) F"*
  ⟨*proof*⟩

1.2 Integrals

lemma *has_absolute_integral_reflect_real:*
fixes *f :: "real ⇒ 'a :: euclidean_space"*
assumes *"uminus ‘ A ⊆ B" "uminus ‘ B ⊆ A" "A ∈ sets lebesgue"*
shows *"(λx. f (-x)) absolutely_integrable_on A ∧ integral A (λx. f (-x)) = b ⟷*
f absolutely_integrable_on B ∧ integral B f = b"
⟨*proof*⟩

lemma *has_integral_spike_set_eq':*
fixes *f :: "'n::euclidean_space ⇒ 'a::banach"*
assumes *"negligible U"*
assumes *"⋀x. x ∈ S - T - U ⟹ f x = 0"*
assumes *"⋀x. x ∈ T - S - U ⟹ f x = 0"*
shows *"(f has_integral y) S ⟷ (f has_integral y) T"*
⟨*proof*⟩

lemma *integral_spike_set':*
fixes *f :: "'n::euclidean_space ⇒ 'a::banach"*
assumes *"negligible U"*
assumes *"⋀x. x ∈ S - T - U ⟹ f x = 0"*
assumes *"⋀x. x ∈ T - S - U ⟹ f x = 0"*
shows *"integral S f = integral T f"*
⟨*proof*⟩

lemma *integrable_spike_set_eq':*
fixes *f :: "'n::euclidean_space ⇒ 'a::banach"*
assumes *"negligible U"*
assumes *"⋀x. x ∈ S - T - U ⟹ f x = 0"*
assumes *"⋀x. x ∈ T - S - U ⟹ f x = 0"*
shows *"f integrable_on S ⟷ f integrable_on T"*
⟨*proof*⟩

lemma *absolutely_integrable_spike_set_eq':*
fixes *f :: "'a::euclidean_space ⇒ 'b::euclidean_space"*
assumes *"negligible U"*
assumes *"⋀x. x ∈ S - T - U ⟹ f x = 0"*
assumes *"⋀x. x ∈ T - S - U ⟹ f x = 0"*
shows *"f absolutely_integrable_on S ⟷ f absolutely_integrable_on T"*
⟨*proof*⟩

lemma *has_integral_of_real:*
assumes *"(f has_integral I) A"*

shows *"((λx. of_real (f x)) has_integral (of_real I)) A"*
⟨*proof*⟩

1.3 Real analysis

lemma *filterlim_abs_real_at_bot:*
"filterlim (abs :: real ⇒ real) at_top at_bot"
⟨*proof*⟩

lemmas *filterlim_abs_real' [tendsto_intros] =*
filterlim_abs_real [THEN filterlim_compose]

lemmas *filterlim_abs_real_at_bot' [tendsto_intros] =*
filterlim_abs_real_at_bot [THEN filterlim_compose]

lemma *filterlim_abs_real_at_infinity:*
assumes *"filterlim f at_infinity F"*
shows *"filterlim (λx. abs (f x :: real)) at_top F"*
⟨*proof*⟩

lemma *has_field_derivative_abs:*
assumes *"(x::real) ≠ 0"*
shows *"(abs has_field_derivative sgn x) (at x within A)"*
⟨*proof*⟩

lemmas *has_field_derivative_abs' [derivative_intros] =*
has_field_derivative_abs [THEN DERIV_chain2]

lemma *abs_has_real_derivative_1:*
assumes *"x > 0 ∨ A ⊆ {0..}"*
shows *"(abs has_real_derivative 1) (at x within A)"*
⟨*proof*⟩

lemma *abs_has_real_derivative_neg1:*
assumes *"x < 0 ∨ A ⊆ {..0}"*
shows *"(abs has_real_derivative (-1)) (at x within A)"*
⟨*proof*⟩

lemma *tendsto_MInfty_eq_at_top:*
"((λz. ereal (f z)) ⟶ -∞) F ⟷ filterlim f at_bot F"
⟨*proof*⟩

lemma *tendsto_ereal_asymp_equiv_transfer:*
fixes *f g :: "'a ⇒ real"*
assumes *"f ∼[F] g"*
shows *"((λx. ereal (f x)) ⟶ c) F ⟷ ((λx. ereal (g x)) ⟶ c) F"*
⟨*proof*⟩

If $f(n)$ and $g(n)$ are increasing and decreasing sequences, respectively, such that the difference $g(n) - f(n)$ is non-negative and vanishes for $n \to \infty$, then $f(n)$ and $g(n)$ tend to a common limit from below and above, respectively.

lemma *incseq_decseq_tendsto_sandwich:*
fixes *f g :: "nat ⇒ real"*
assumes *"incseq f" "decseq g" "⋀n. f n ≤ g n"*
assumes *"(λn. g n - f n) ⟶ 0"*
obtains *L* **where** *"f ⟶ L" "g ⟶ L" "⋀n. f n ≤ L" "⋀n. g n ≥ L"*
⟨*proof*⟩

Suppose we have two sequences of numbers $f(n)$ and $g(n)$ and let I_n denote the interval $[\min(f(n), g(n)), \max(f(n), g(n))]$.

Assume that $I_{n+1} \subseteq I_n$ for all n and that $|g(n) - f(n)| \to 0$ as $n \to \infty$. Then f and g tend to a common limit, and that limit is contained in each of the intervals I_n.

lemma *nested_intervals_tendsto_sandwich:*
fixes *f g :: "nat ⇒ real"*
assumes *f: "⋀n. f (Suc n) ∈ closed_segment (f n) (g n)"*
assumes *g: "⋀n. g (Suc n) ∈ closed_segment (f n) (g n)"*
assumes *fg_vanishes: "(λn. g n - f n) ⟶ 0"*
obtains *L* **where** *"f ⟶ L" "g ⟶ L" "⋀n. L ∈ closed_segment (f n) (g n)"*
⟨*proof*⟩

1.4 Complex numbers in general

lemma *linear_cmult_complex_left: "linear f ⟹ linear (λx. c * f x :: complex)"*
⟨*proof*⟩

lemma *linear_cmult_complex_right: "linear f ⟹ linear (λx. f x * c :: complex)"*
⟨*proof*⟩

lemma *cmod_rcis_add_squared:*
*"norm (rcis r1 a1 + rcis r2 a2) ^ 2 = r1 ^ 2 + r2 ^ 2 + 2 * r1 * r2 * cos (a1 - a2)"*
⟨*proof*⟩

lemma *Im_one_over: "Im (1 / z) = -Im z / norm z ^ 2"*
⟨*proof*⟩

lemma *Re_one_over: "Re (1 / z) = Re z / norm z ^ 2"*
⟨*proof*⟩

lemma *Arg_times':*
assumes *"w ≠ 0" "z ≠ 0"*
defines *"x ≡ Arg w + Arg z"*
shows *"Arg (w * z) = x + (if x ∈ {-pi<..pi} then 0 else if x > pi then -2*pi else 2*pi)"*
⟨*proof*⟩

lemma *Arg_divide':*
assumes *[simp]: "z ≠ 0" "w ≠ 0"*
shows *"Arg (z / w) = Arg z - Arg w +*
*(if Arg z - Arg w > pi then -2 * pi else if Arg z - Arg w ≤ -pi then 2 * pi else 0)"*
(is *"_ = ?rhs")*
⟨*proof*⟩

lemma *Ln_not_in_nonpos_Reals:*
assumes *"Im x ≠ 0 ∨ Re x > 1"*
shows *"Ln x ∉ $\mathbb{R}_{\leq 0}$"*
⟨*proof*⟩

lemma *Ln_one_over: "z ∉ $\mathbb{R}_{\leq 0}$ ⟹ Ln (1 / z) = -Ln z"*
⟨*proof*⟩

lemma *csqrt_conv_rcis: "csqrt x = rcis (sqrt (norm x)) (Arg x / 2)"*
⟨*proof*⟩

lemma *csqrt_rcis:*
assumes *"r ≥ 0" "a ∈ {-pi<..pi}"*
shows *"csqrt (rcis r a) = rcis (sqrt r) (a / 2)"*
⟨*proof*⟩

1.5 Analyticity, holomorphicity, continuity, limit of some complex functions

lemma *continuous_on_csqrt [continuous_intros]:*
assumes *"continuous_on A f" "⋀x. x ∈ A ⟹ f x ∉ $\mathbb{R}_{\leq 0}$"*
shows *"continuous_on A (λx. csqrt (f x))"*
⟨*proof*⟩

lemma *continuous_csqrt [continuous_intros]:*
assumes *"continuous (at x within A) f" "f x ∉ $\mathbb{R}_{\leq 0}$"*
shows *"continuous (at x within A) (λx. csqrt (f x))"*
⟨*proof*⟩

lemma *tendsto_csqrt [tendsto_intros]:*
assumes *"(f ⟶ z) F" "z ∉ $\mathbb{R}_{\leq 0}$ - {0}"*
shows *"((λx. csqrt (f x)) ⟶ csqrt z) F"*
⟨*proof*⟩

lemma *has_field_derivative_csqrt' [derivative_intros]:*
assumes *"(f has_field_derivative f') (at z within A)" "f z ∉ $\mathbb{R}_{\leq 0}$"*
shows *"((λx. csqrt (f x)) has_field_derivative (f' / (2 * csqrt (f z)))) (at z within A)"*
⟨*proof*⟩

lemma *analytic_on_Ln [analytic_intros]:*
assumes *"S ∩ $\mathbb{R}_{\leq 0}$ = {}"*
shows *"Ln analytic_on S"*
⟨*proof*⟩

lemma *analytic_on_Ln' [analytic_intros]:*
"(⋀z. z ∈ A ⟹ f z ∉ $\mathbb{R}_{\leq 0}$) ⟹ f analytic_on A ⟹ (λz. Ln (f z)) analytic_on A"
⟨*proof*⟩

lemma *continuous_Ln [continuous_intros]:*
assumes *"continuous (at x within A) f" "f x ∉ $\mathbb{R}_{\leq 0}$"*
shows *"continuous (at x within A) (λx. Ln (f x))"*
⟨*proof*⟩

lemma *contour_integral_primitive':*
assumes *"⋀x. x ∈ s ⟹ (f has_field_derivative f' x) (at x within s)"*
and *"valid_path g" "path_image g ⊆ s" "pathfinish g = b" "pathstart g = a"*
shows *"(f' has_contour_integral (f b - f a)) g"*
⟨*proof*⟩

1.6 Products over lists

lemma *prod_list_conv_prod_nth: "prod_list xs = (∏i<length xs. xs ! i)"*

⟨*proof*⟩

lemma *prod_list_conv_prod_nth': "prod_list (map f xs) = (∏i<length xs. f (xs ! i))"*
⟨*proof*⟩

lemma *prod_list_const [simp]: "(∏x←xs. c) = c ^ length xs"*
⟨*proof*⟩

lemma *prod_list_pos:*
"(⋀x. x ∈ set xs ⟹ x > 0) ⟹ prod_list xs > (0 :: 'a :: {linordered_semiring_strict, linordered_semiring_1})"
⟨*proof*⟩

lemma *prod_list_nonneg':*
"(⋀x. x ∈ set xs ⟹ f x ≥ 0) ⟹ (∏x←xs. f x) ≥ (0 :: 'a :: linordered_semiring_1)"
⟨*proof*⟩

lemma *prod_list_pos':*
"(⋀x. x ∈ set xs ⟹ f x > 0) ⟹ (∏x←xs. f x) > (0 :: 'a :: {linordered_semiring_stri linordered_semiring_1})"
⟨*proof*⟩

lemma *prod_list_mono:*
fixes *xs ys :: "'a :: linordered_semiring_1 list"*
assumes *"⋀x. x ∈ set xs ⟹ x ≥ 0" "list_all2 (≤) xs ys"*
shows *"prod_list xs ≤ prod_list ys"*
⟨*proof*⟩

lemma *prod_list_mono':*
fixes *f g :: "'a ⇒ 'b :: linordered_semiring_1"*
assumes *"⋀x. x ∈ set xs ⟹ f x ≥ 0" "⋀x. x ∈ set xs ⟹ f x ≤ g x"*
shows *"(∏x←xs. f x) ≤ (∏x←xs. g x)"*
⟨*proof*⟩

lemma *continuous_on_prod_list [continuous_intros]:*
fixes *f :: "'a ⇒ 'b :: topological_space ⇒ 'c :: real_normed_algebra_1"*
assumes *"⋀x. x ∈ set xs ⟹ continuous_on X (f x)"*
shows *"continuous_on X (λy. ∏x←xs. f x y)"*
⟨*proof*⟩

lemma *holomorphic_on_prod_list [holomorphic_intros]:*
assumes *"⋀x. x ∈ set xs ⟹ f x holomorphic_on X"*
shows *"(λy. ∏x←xs. f x y) holomorphic_on X"*
⟨*proof*⟩

lemma *asymp_equiv_prod_list:*
assumes *"list_all2 (λy z. f y ∼[F] g z) ys zs"*
shows *"(λx. (∏y←ys. f y x)) ∼[F] (λx. (∏z←zs. g z x))"*
⟨*proof*⟩

lemma *asymp_equiv_prod_list' [asymp_equiv_intros]:*
assumes *"⋀y. y ∈ set ys ⟹ f y ∼[F] g y"*
shows *"(λx. (∏y←ys. f y x)) ∼[F] (λx. (∏y←ys. g y x))"*
⟨*proof*⟩

lemma *norm_prod_list:*
fixes *xs :: "'a :: real_normed_div_algebra list"*
shows *"norm (prod_list xs) = prod_list (map norm xs)"*
⟨*proof*⟩

1.7 Formal power series and Laurent series

lemma *fls_deriv_divide:*
fixes *f g :: "'a :: field fls"*
shows *"fls_deriv (f / g) = (g * fls_deriv f - f * fls_deriv g) / g ^ 2"*
⟨*proof*⟩

lemma *fls_deriv_divide_const:*
fixes *f g :: "'a :: field fls"*
assumes *"fls_deriv g = 0"*
shows *"fls_deriv (f / g) = fls_deriv f / g"*
⟨*proof*⟩

lemma *fls_X_neq_1 [simp]: "fls_X ≠ (1 :: 'a :: zero_neq_one fls)"*
⟨*proof*⟩

lemma *has_fps_expansion_imp_asymp_equiv_0:*
assumes *"f has_fps_expansion F"*
shows *"f ∼[at 0] (λx. fps_nth F (subdegree F) * x ^ subdegree F)"*
⟨*proof*⟩

lemma *has_fps_expansion_imp_tendsto_0:*
fixes *f :: "'a ⇒ 'a :: {real_normed_field, banach}"*
assumes *"f has_fps_expansion F"*
shows *"(f ⟶ fps_nth F 0) (nhds 0)"*
⟨*proof*⟩

lemma *has_laurent_expansion_0_analytic_continuation':*
assumes *"f has_laurent_expansion 0" "f holomorphic_on A"*
assumes *"open A" "connected A" "0 ∈ A" "x ∈ A"*
shows *"f x = 0"*
⟨*proof*⟩

1.8 Normalisation of angles

The following operation normalises an angle φ, i.e. returns the unique ψ in the range $(-\pi, \pi]$ such that $\varphi \equiv \psi \pmod{2\pi}$. This is the same convention used by the *Arg* function.

definition *normalize_angle :: "real ⇒ real"* **where**
*"normalize_angle x = x - ⌈x / (2 * pi) - 1 / 2⌉ * (2 * pi)"*

lemma *normalize_angle_id [simp]:*
assumes *"x ∈ {-pi<..pi}"*
shows *"normalize_angle x = x"*
⟨*proof*⟩

lemma *normalize_angle_normalized: "normalize_angle x ∈ {-pi<..pi}"*
⟨*proof*⟩

lemma *cis_normalize_angle [simp]: "cis (normalize_angle x) = cis x"*
⟨*proof*⟩

lemma *rcis_normalize_angle [simp]: "rcis r (normalize_angle x) = rcis r x"*
⟨*proof*⟩

lemma *normalize_angle_lbound [intro]: "normalize_angle x > -pi"*
and *normalize_angle_ubound [intro]: "normalize_angle x ≤ pi"*
⟨*proof*⟩

lemma *normalize_angle_idem [simp]: "normalize_angle (normalize_angle x) = normalize_angle x"*
⟨*proof*⟩

lemma *Arg_rcis: "r > 0 ⟹ Arg (rcis r φ) = normalize_angle φ"*
⟨*proof*⟩

1.9 Convexity of circular sectors in the complex plane

In this section we will show that if we have two non-zero points w and z in the complex plane whose arguments differ by less than π, then the argument of any point on the line connecting w and z lies between the arguments of w and z. Moreover, the norm of any such point is no greater than the norms of w and z.

Geometrically, this means that if we have a sector around the origin with a central angle less than π (minus the origin itself) then that region is convex.

lemma *Arg_closed_segment_aux1:*
assumes *"x ≠ 0" "y ≠ 0" "Re x > 0" "Re x = Re y"*
assumes *"z ∈ closed_segment x y"*
shows *"Arg z ∈ closed_segment (Arg x) (Arg y)"*
⟨*proof*⟩

lemma *Arg_closed_segment_aux1':*
fixes *r1 r2 φ1 φ2 :: real*
defines *"x ≡ rcis r1 φ1"* **and** *"y ≡ rcis r2 φ2"*
assumes *"z ∈ closed_segment x y"*
assumes *"r1 > 0" "r2 > 0" "Re x = Re y" "Re x ≥ 0" "Re x = 0 ⟶ Im x * Im y > 0"*
assumes *"dist φ1 φ2 < pi"*
obtains *r φ* **where** *"r ∈ {0<..max r1 r2}" "φ ∈ closed_segment φ1 φ2" "z = rcis r φ"*
⟨*proof*⟩

lemma *Arg_closed_segment':*
fixes *r1 r2 φ1 φ2 :: real*
defines *"x ≡ rcis r1 φ1"* **and** *"y ≡ rcis r2 φ2"*
assumes *"r1 > 0" "r2 > 0" "dist φ1 φ2 < pi" "z ∈ closed_segment x y"*
obtains *r φ* **where** *"r ∈ {0<..max r1 r2}" "φ ∈ closed_segment φ1 φ2" "z = rcis r φ"*
⟨*proof*⟩

lemma *Arg_closed_segment:*
assumes *"x ≠ 0" "y ≠ 0" "dist (Arg x) (Arg y) < pi" "z ∈ closed_segment x y"*
shows *"Arg z ∈ closed_segment (Arg x) (Arg y)"*
⟨*proof*⟩

1.10 Complex cones

We introduce the following notation to describe the set of all complex numbers of the form ce^{ix} where $c \geq 0$ and $x \in [a, b]$.

definition *complex_cone :: "real ⇒ real ⇒ complex set"* **where**
"complex_cone a b = (λ(r,a). rcis r a) ‘ ({0..} × closed_segment a

b)"

lemma *in_complex_cone_iff: "z ∈ complex_cone a b ⟷ (∃x∈closed_segment a b. z = rcis (norm z) x)"*
⟨*proof*⟩

lemma *zero_in_complex_cone [simp, intro]: "0 ∈ complex_cone a b"*
⟨*proof*⟩

lemma *in_complex_cone_iff_Arg:*
assumes *"a ∈ {-pi<..pi}" "b ∈ {-pi<..pi}"*
shows *"z ∈ complex_cone a b ⟷ z = 0 ∨ Arg z ∈ closed_segment a b"*
⟨*proof*⟩

lemma *complex_cone_rotate: "complex_cone (c + a) (c + b) = (*) (cis c) ‘ complex_cone a b"*
⟨*proof*⟩

lemma *complex_cone_subset:*
"a ∈ closed_segment a' b' ⟹ b ∈ closed_segment a' b' ⟹ complex_cone a b ⊆ complex_cone a' b'"
⟨*proof*⟩

lemma *complex_cone_commute: "complex_cone a b = complex_cone b a"*
⟨*proof*⟩

lemma *complex_cone_eq_UNIV:*
assumes *"dist a b ≥ 2*pi"*
shows *"complex_cone a b = UNIV"*
⟨*proof*⟩

lemma *continuous_on_Arg: "continuous_on (-ℝ≤0) Arg"*
⟨*proof*⟩

lemma *continuous_on_Arg' [continuous_intros]:*
assumes *"continuous_on A f" "⋀z. z ∈ A ⟹ f z ∉ ℝ≤0"*
shows *"continuous_on A (λx. Arg (f x))"*
⟨*proof*⟩

A surprisingly tough argument: cones in the complex plane are closed.

lemma *closed_complex_cone [continuous_intros, intro]: "closed (complex_cone a b)"*
⟨*proof*⟩

end

2 The arithmetic mean of two numbers

theory *Binary_Arithmetic_Mean*
  imports *"HOL-Complex_Analysis.Complex_Analysis"*
begin

definition *amean :: "'a :: real_vector ⇒ 'a ⇒ 'a"*
  where *"amean x y = (1/2) *$_R$ (x + y)"*

lemma *amean_commute: "amean x y = amean y x"*
  ⟨*proof*⟩

lemma *amean_self [simp]: "amean x x = x"*
  ⟨*proof*⟩

lemma *amean_of_real [simp]:*
  "amean (of_real x) (of_real y :: 'a :: real_field) = of_real (amean x y)"
  ⟨*proof*⟩

lemma *amean_nonneg: "(x::real) ≥ 0 ⟹ y ≥ 0 ⟹ amean x y ≥ 0"*
  ⟨*proof*⟩

lemma *amean_pos: "(x::real) > 0 ⟹ y > 0 ⟹ amean x y > 0"*
  ⟨*proof*⟩

lemma *amean_scaleR: "amean (a *$_R$ x) (a *$_R$ y) = a *$_R$ amean x y"*
  ⟨*proof*⟩

lemma *amean_mult_left: "amean (a * x) (a * y :: 'a :: real_algebra) = a * amean x y"*
  ⟨*proof*⟩

lemma *amean_mult_right: "amean (x * a) (y * a :: 'a :: real_algebra) = amean x y * a"*
  ⟨*proof*⟩

lemma *amean_minus: "amean (-x) (-y) = -amean x y"*
  ⟨*proof*⟩

lemma *amean_real_mono: "(x :: real) ≤ x' ⟹ y ≤ y' ⟹ amean x y ≤ amean x' y'"*
  ⟨*proof*⟩

lemma *amean_eq_midpoint: "amean x y = midpoint x y"*
  ⟨*proof*⟩

lemma *amean_in_closed_segment: "amean x y ∈ closed_segment x y"*
  ⟨*proof*⟩

lemma *amean_real_between: "amean x (y :: real) ∈ {min x y..max x y}"*
⟨*proof*⟩

lemma *norm_amean_le: "norm (amean x y) ≤ max (norm x) (norm y)"*
⟨*proof*⟩

lemma *amean_le_real: "amean x y ≤ max x (y :: real)"*
⟨*proof*⟩

lemma *amean_ge_real: "amean x y ≥ min x (y :: real)"*
⟨*proof*⟩

lemma *Arg_amean:*
assumes *"x ≠ 0" "y ≠ 0" "dist (Arg x) (Arg y) < pi"*
shows *"Arg (amean x y) ∈ closed_segment (Arg x) (Arg y)"*
⟨*proof*⟩

lemma *has_derivative_amean [derivative_intros]:*
assumes *"(f has_derivative f') (at x within A)"*
assumes *"(g has_derivative g') (at x within A)"*
shows *"((λx. amean (f x) (g x)) has_derivative (λx. amean (f' x) (g' x))) (at x within A)"*
⟨*proof*⟩

lemma *has_vector_derivative_amean [derivative_intros]:*
assumes *"(f has_vector_derivative f') (at x within A)"*
assumes *"(g has_vector_derivative g') (at x within A)"*
shows *"((λx. amean (f x) (g x)) has_vector_derivative (amean f' g')) (at x within A)"*
⟨*proof*⟩

lemma *has_field_derivative_amean [derivative_intros]:*
assumes *"(f has_field_derivative f') (at x within A)"*
assumes *"(g has_field_derivative g') (at x within A)"*
shows *"((λx. amean (f x) (g x)) has_field_derivative (amean f' g')) (at x within A)"*
⟨*proof*⟩

lemma *continuous_on_amean [continuous_intros]:*
fixes *f g :: "'a :: topological_space ⇒ 'b :: real_normed_vector"*
assumes *"continuous_on A f" "continuous_on A g"*
shows *"continuous_on A (λz. amean (f z) (g z))"*
⟨*proof*⟩

lemma *continuous_amean [continuous_intros]:*
fixes *f g :: "'a :: t2_space ⇒ 'b :: real_normed_vector"*
assumes *"continuous (at x within A) f" "continuous (at x within A) g"*

shows *"continuous (at x within A) (λz. amean (f z) (g z))"*
⟨*proof*⟩

lemma *tendsto_amean [tendsto_intros]:*
fixes *f g :: "'a :: t2_space ⇒ 'b :: real_normed_vector"*
assumes *"(f ⟶ lf) F" "(g ⟶ lg) F"*
shows *"((λz. amean (f z) (g z)) ⟶ amean lf lg) F"*
⟨*proof*⟩

lemma *holomorphic_on_amean [holomorphic_intros]:*
assumes *"f holomorphic_on A" "g holomorphic_on A"*
shows *"(λz. amean (f z) (g z)) holomorphic_on A"*
⟨*proof*⟩

lemma *analytic_on_amean [analytic_intros]:*
assumes *"f analytic_on A" "g analytic_on A"*
shows *"(λz. amean (f z) (g z)) analytic_on A"*
⟨*proof*⟩

end

3 The geometric mean of two numbers

theory *Binary_Geometric_Mean*
imports *"HOL-Complex_Analysis.Complex_Analysis" Binary_Arithmetic_Mean AGM_Lemma_Bucket*
begin

We say that two numbers are *opposite* in the complex plane if they are non-zero and their quotient is a negative real.

Another simple geometric characterisation is that the origin lies on the open line segment connecting the two points.

definition *opposite_complex :: "complex ⇒ complex ⇒ bool"* **where**
"opposite_complex w z ⟷ w ≠ 0 ∧ z ≠ 0 ∧ sgn w = -sgn z"

lemma *opposite_complex_sym: "opposite_complex w z ⟷ opposite_complex z w"*
⟨*proof*⟩

lemma *not_opposite_complex_0_left [simp]: "¬opposite_complex 0 z"*
and *not_opposite_complex_0_right [simp]: "¬opposite_complex w 0"*
⟨*proof*⟩

lemma *opposite_complex_minus_self_iff: "opposite_complex w (-w) ⟷ w ≠ 0"*
and *opposite_complex_minus_self_iff': "opposite_complex (-w) w ⟷ w ≠ 0"*
⟨*proof*⟩

lemma *opposite_complex_altdef1:*
"opposite_complex w z ⟷ w ≠ 0 ∧ z ≠ 0 ∧ w / z ∈ ℝ≤0"
⟨*proof*⟩

lemma *opposite_complex_altdef1':*
*"opposite_complex w z ⟷ w ≠ 0 ∧ z ≠ 0 ∧ (∃c>0. w = -of_real c * z)"*
⟨*proof*⟩

lemma *opposite_complex_real_right_iff:*
assumes *"w ∈ ℝ"*
shows *"opposite_complex z w ⟷ z ≠ 0 ∧ Im z = 0 ∧ sgn (Re z) = - sgn (Re w)"*
⟨*proof*⟩

lemma *opposite_complex_real_left_iff:*
assumes *"w ∈ ℝ"*
shows *"opposite_complex w z ⟷ z ≠ 0 ∧ Im z = 0 ∧ sgn (Re z) = - sgn (Re w)"*
⟨*proof*⟩

lemma *opposite_complex_1_right_iff: "opposite_complex w 1 ⟷ w ≠ 0 ∧ w ∈ ℝ≤0"*
⟨*proof*⟩

lemma *opposite_complex_1_left_iff: "opposite_complex 1 w ⟷ w ≠ 0 ∧ w ∈ ℝ≤0"*
⟨*proof*⟩

lemma *opposite_complex_altdef2:*
"opposite_complex w z ⟷ w ≠ 0 ∧ z ≠ 0 ∧ dist (Arg w) (Arg z) = pi"
⟨*proof*⟩

lemma *opposite_complex_altdef3: "opposite_complex a b ⟷ 0 ∈ open_segment a b"*
⟨*proof*⟩

lemma *opposite_complex_mult_left: "opposite_complex a b ⟹ c ≠ 0 ⟹ opposite_complex (c * a) (c * b)"*
⟨*proof*⟩

lemma *opposite_complex_mult_left_iff:*
*"c ≠ 0 ⟹ opposite_complex (c * a) (c * b) ⟷ opposite_complex a b"*
⟨*proof*⟩

lemma *opposite_complex_mult_right_iff:*

*"c ≠ 0 ⟹ opposite_complex (a * c) (b * c) ⟷ opposite_complex a b"*
⟨*proof*⟩

We introduce the geometric mean via a locale in order to make things uniform for both the reals and the complex numbers.

class *gmean = real_normed_field +*
fixes *gmean :: "'a ⇒ 'a ⇒ 'a"*
assumes *gmean_commute: "gmean x y = gmean y x"*
and *gmean_0_left [simp]: "gmean 0 y = 0"*
and *norm_gmean_aux: "norm (gmean x y) = sqrt (norm x * norm y)"*
begin

lemma *gmean_0_right [simp]: "gmean x 0 = 0"*
⟨*proof*⟩

lemma *gmean_0_iff: "gmean x y = 0 ⟷ x = 0 ∨ y = 0"*
⟨*proof*⟩

lemma *norm_gmean_ge: "norm (gmean x y) ≥ min (norm x) (norm y)"*
⟨*proof*⟩

lemma *norm_gmean_le: "norm (gmean x y) ≤ max (norm x) (norm y)"*
⟨*proof*⟩

end

instantiation *real :: gmean*
begin

definition *gmean_real :: "real ⇒ real ⇒ real"*
where *"gmean x y = abs (sqrt (x * y))"*

instance
⟨*proof*⟩

end

lemma *(in gmean) norm_gmean: "norm (gmean x y) = gmean (norm x) (norm y)"*
⟨*proof*⟩

lemma *gmean_real_pos: "(x :: real) > 0 ⟹ y > 0 ⟹ gmean x y > 0"*
⟨*proof*⟩

instantiation *complex :: gmean*
begin

definition *gmean_complex :: "complex ⇒ complex ⇒ complex"*
where *"gmean x y = (if dist (Arg x) (Arg y) ≤ pi then csqrt x * csqrt y else -csqrt x * csqrt y)"*

instance ⟨*proof*⟩

end

lemma *gmean_complex_square: "gmean x y ^ 2 = x * (y :: complex)"*
⟨*proof*⟩

lemma *gmean_real_self [simp]: "x ≥ 0 ⟹ gmean x x = abs (x :: real)"*
⟨*proof*⟩

lemma *gmean_real_nonneg: "gmean x y ≥ (0 :: real)"*
⟨*proof*⟩

lemma *gmean_le_amean_real:*
assumes *"x ≥ 0" "y ≥ (0 :: real)"*
shows *"gmean x y ≤ amean x y"*
⟨*proof*⟩

lemma *gmean_real_between:*
assumes *"x ≥ 0" "y ≥ 0"*
shows *"gmean x (y :: real) ∈ {min x y..max x y}"*
⟨*proof*⟩

lemma *gmean_real_strictly_between:*
assumes *"0 < x" "x < (y::real)"*
shows *"gmean x y ∈ {x<..<y}"*
⟨*proof*⟩

lemma *gmean_mult_real:*
assumes *"a ≥ 0" "b ≥ 0" "x ≥ 0" "y ≥ 0"*
shows *"gmean (a * x :: real) (b * y) = gmean a b * gmean x y"*
⟨*proof*⟩

lemma *gmean_real_mono:*
"(x :: real) ≤ x' ⟹ y ≤ y' ⟹ 0 ≤ x ⟹ 0 ≤ y ⟹ gmean x y ≤ gmean x' y'"
⟨*proof*⟩

lemma *gmean_in_closed_segment_real:*
"(x::real) > 0 ⟹ y > 0 ⟹ gmean x y ∈ closed_segment x y"
⟨*proof*⟩

lemma *gmean_in_open_segment_real:*
"(x::real) > 0 ⟹ y > 0 ⟹ x ≠ y ⟹ gmean x y ∈ open_segment x

```
y"
```
⟨*proof*⟩

lemma *Arg_csqrt_mult_csqrt:*
assumes *"x ≠ 0" "y ≠ 0"*
shows *"Arg (csqrt x * csqrt y) = amean (Arg x) (Arg y)"*
⟨*proof*⟩

lemma *Arg_gmean:*
assumes *"x ≠ 0" "y ≠ 0"*
shows *"Arg (gmean x y) = amean (Arg x) (Arg y) +*
(if dist (Arg x) (Arg y) ≤ pi then 0 else if Arg x + Arg
y > 0 then -pi else pi)"
⟨*proof*⟩

lemma *gmean_complex_of_real [simp]:*
assumes *"x ≥ 0" "y ≥ 0"*
shows *"gmean (complex_of_real x) (complex_of_real y) = of_real (gmean*
x y)"
⟨*proof*⟩

lemma *dist_Arg_amean_gmean_le:*
assumes *"w ≠ 0" "z ≠ 0" "dist (Arg w) (Arg z) < pi"*
shows *"dist (Arg (amean w z)) (Arg (gmean w z)) ≤ dist (Arg w) (Arg*
z) / 2"
⟨*proof*⟩

lemma *norm_amean_over_gmean:*
fixes *w z :: complex*
assumes *"w ≠ 0" "z ≠ 0"*
shows *"norm (amean w z / gmean w z) =*
*sqrt (norm (w / z) + 2 * cos (Arg w - Arg z) + norm (z /*
w)) / 2"
⟨*proof*⟩

lemma *norm_amean_over_gmean_ge:*
assumes *"w ≠ 0" "z ≠ 0"*
shows *"norm (amean w z / gmean w z) ≥ |cos ((Arg w - Arg z) / 2)|"*
⟨*proof*⟩

lemma *dist_le_norm_add_complex_strong:*
assumes *"r1 ≥ 0" "r2 ≥ 0" "|x2 - x1| ≤ pi / 2"*
defines *"w ≡ rcis r1 x1"* **and** *"z ≡ rcis r2 x2"*
shows *"dist w z ≤ norm (w + z)"*
⟨*proof*⟩

lemma *dist_le_norm_add_complex:*
assumes *"dist (Arg w) (Arg z) ≤ pi / 2"*
shows *"dist w z ≤ norm (w + z)"*

⟨*proof*⟩

lemma *csqrt_minus': "csqrt (-z) = (if Arg z ≤ 0 then* i *else* -i*) * csqrt z"*
⟨*proof*⟩

lemma *gmean_opposite:*
fixes *w z :: complex*
assumes *c: "c > 0"* **and** *w: "w = -of_real c * z"*
shows *"gmean w z = (if Arg z ≤ 0 then* i *else* -i*) * (of_real (sqrt c) * z)"*
⟨*proof*⟩

lemma *gmean_mult_left_complex:*
fixes *u w z :: complex*
assumes *"¬opposite_complex w z"*
shows *"gmean (u * w) (u * z) = u * gmean w z"*
⟨*proof*⟩

lemma *gmean_minus_complex:*
assumes *"¬opposite_complex w z"*
shows *"gmean (-w) (-z) = -gmean w z"*
⟨*proof*⟩

lemma *gmean_complex_1_left [simp]: "gmean 1 z = csqrt z"*
and *gmean_complex_1_right [simp]: "gmean z 1 = csqrt z"*
⟨*proof*⟩

lemma *gmean_same_real [simp]: "x ≥ 0 ⟹ gmean x x = (x::real)"*
⟨*proof*⟩

lemma *gmean_same_complex [simp]: "gmean z z = (z::complex)"*
⟨*proof*⟩

lemma *dist_amean_gmean_le_complex_aux:*
assumes *"w ≠ 0" "z ≠ 0" "dist (Arg w) (Arg z) < pi"*
shows *"dist (amean w z) (gmean w z) ≤ dist w z / 2"*
⟨*proof*⟩

lemma *dist_amean_gmean_le_complex:*
fixes *w z :: complex*
shows *"dist (amean w z) (gmean w z) ≤ dist w z / 2"*
⟨*proof*⟩

hide_fact *dist_amean_gmean_le_complex_aux*

lemma *holomorphic_on_gmean [holomorphic_intros]:*

assumes *"f holomorphic_on A" "g holomorphic_on A"*
assumes *"⋀z. z ∈ A ⟹ f z / g z ∉ ℝ≤0"*
shows *"(λz. gmean (f z) (g z)) holomorphic_on A"*
⟨*proof*⟩

lemma *analytic_on_gmean [analytic_intros]:*
assumes *"f analytic_on A" "g analytic_on A"*
assumes *"⋀z. z ∈ A ⟹ f z / g z ∉ ℝ≤0"*
shows *"(λz. gmean (f z) (g z)) analytic_on A"*
⟨*proof*⟩

lemma *has_field_derivative_gmean_real [derivative_intros]:*
assumes *"(f has_field_derivative f') (at x within A)"*
assumes *"(g has_field_derivative g') (at x within A)"*
assumes *"f x > 0" "g x > 0"*
shows *"((λx. gmean (f x :: real) (g x)) has_field_derivative*
*((f' * g x + g' * f x) / (2 * gmean (f x) (g x)))) (at x within A)"*
⟨*proof*⟩

lemma *continuous_on_gmean_real [continuous_intros]:*
fixes *f g :: "'a :: topological_space ⇒ real"*
assumes *"continuous_on A f" "continuous_on A g"*
shows *"continuous_on A (λz. gmean (f z) (g z))"*
⟨*proof*⟩

lemma *continuous_on_gmean_complex [continuous_intros]:*
fixes *f g :: "'a :: topological_space ⇒ complex"*
assumes *"continuous_on A f" "continuous_on A g"*
assumes *"⋀z. z ∈ A ⟹ f z / g z ∉ ℝ≤0"*
shows *"continuous_on A (λz. gmean (f z) (g z))"*
⟨*proof*⟩

lemma *tendsto_gmean_real [tendsto_intros]:*
fixes *f g :: "'a :: t2_space ⇒ real"*
assumes *"(f ⟶ lf) F" "(g ⟶ lg) F"*
shows *"((λz. gmean (f z) (g z)) ⟶ gmean lf lg) F"*
⟨*proof*⟩

lemma *tendsto_gmean_complex [tendsto_intros]:*
fixes *f g :: "'a :: t2_space ⇒ complex"*
assumes *"(f ⟶ lf) F" "(g ⟶ lg) F"*
assumes *"lf / lg ∉ ℝ≤0"*
shows *"((λz. gmean (f z) (g z)) ⟶ gmean lf lg) F"*
⟨*proof*⟩

lemma *continuous_gmean_real [continuous_intros]:*

fixes *f g :: "'a :: t2_space ⇒ real"*
assumes *"continuous (at x within A) f" "continuous (at x within A) g"*
shows *"continuous (at x within A) (λz. gmean (f z) (g z))"*
⟨*proof*⟩

lemma *continuous_gmean_complex [continuous_intros]:*
fixes *f g :: "'a :: t2_space ⇒ complex"*
assumes *"continuous (at x within A) f" "continuous (at x within A) g"*
assumes *"f x / g x ∉ $\mathbb{R}_{\leq 0}$"*
shows *"continuous (at x within A) (λz. gmean (f z) (g z))"*
⟨*proof*⟩

end

4 The arithmetic-geometric mean

theory *Arithmetic_Geometric_Mean*
imports
AGM_Lemma_Bucket
Binary_Arithmetic_Mean
Binary_Geometric_Mean
begin

4.1 Definition and basic properties

definition *agm_seq :: "'a :: gmean ⇒ 'a ⇒ nat ⇒ 'a × 'a"* **where**
"agm_seq x y n = ((λ(a,g). (amean a g, gmean a g)) ^^ n) (x, y)"

lemma *agm_seq_0 [simp]: "agm_seq x y 0 = (x, y)"*
and *agm_seq_rec: "agm_seq x y (Suc n) = agm_seq (amean x y) (gmean x y) n"*
⟨*proof*⟩

lemma *agm_seq_rec': "agm_seq x y (Suc n) = (let (a, g) = agm_seq x y n in (amean a g, gmean a g))"*
⟨*proof*⟩

lemma *agm_seq_commute:*
assumes *"n > 0"*
shows *"agm_seq x y n = agm_seq y x n"*
⟨*proof*⟩

lemma *agm_seq_same_real [simp]: "x ≥ 0 ⟹ agm_seq x x n = (x, x :: real)"*
⟨*proof*⟩

lemma *agm_seq_same_complex [simp]: "agm_seq x x n = (x, x :: complex)"*

⟨*proof*⟩

lemma *agm_seq_real_ge:*
assumes *"0 ≤ y" "y ≤ (x :: real)"*
shows *"fst (agm_seq x y n) ≥ snd (agm_seq x y n)"*
⟨*proof*⟩

lemma *agm_seq_real_ge':*
assumes *"0 ≤ x" "0 ≤ (y::real)" "n > 0"*
shows *"fst (agm_seq x y n) ≥ snd (agm_seq x y n)"*
⟨*proof*⟩

lemma *agm_seq_real_mono:*
assumes *"0 ≤ y" "y ≤ (x :: real)" "m ≤ n"*
shows *"fst (agm_seq x y m) ≥ fst (agm_seq x y n)"* (**is** *?th1*)
and *"snd (agm_seq x y m) ≤ snd (agm_seq x y n)"* (**is** *?th2*)
⟨*proof*⟩

lemma *agm_seq_real_nonneg:*
assumes *"x ≥ 0" "y ≥ (0 :: real)"*
shows *"fst (agm_seq y x n) ≥ 0" "snd (agm_seq y x n) ≥ 0"*
⟨*proof*⟩

lemma *agm_seq_real_pos:*
assumes *"x > 0" "y > (0 :: real)"*
shows *"fst (agm_seq y x n) > 0" "snd (agm_seq y x n) > 0"*
⟨*proof*⟩

lemma *agm_seq_0_right: "agm_seq x 0 n = (x / 2 ^ n, 0)"*
⟨*proof*⟩

lemma *agm_seq_0_left: "n > 0 ⟹ agm_seq 0 x n = (x / 2 ^ n, 0)"*
⟨*proof*⟩

lemma *agm_seq_fst_conv_snd_real:*
assumes *"x > 0" "y > (0 :: real)"*
shows *"fst (agm_seq x y n) = snd (agm_seq x y (Suc n)) ^ 2 / snd (agm_seq x y n)"*
⟨*proof*⟩

lemma *agm_seq_fst_times_snd_complex:*
fixes *x y :: complex*
shows *"fst (agm_seq x y n) * snd (agm_seq x y n) = snd (agm_seq x y (Suc n)) ^ 2"*
⟨*proof*⟩

lemma *agm_seq_mult_real:*
assumes *"a ≥ 0" "x ≥ 0" "y ≥ (0 :: real)"*

shows *"agm_seq (a * x) (a * y) n = map_prod ((*) a) ((*) a) (agm_seq x y n)"*
⟨*proof*⟩

lemma *fst_agm_seq_real_ge_min:*
"(x :: real) ≥ 0 ⟹ y ≥ 0 ⟹ fst (agm_seq x y n) ≥ min x y"
⟨*proof*⟩

lemma *snd_agm_seq_real_ge_min:*
"(x :: real) ≥ 0 ⟹ y ≥ 0 ⟹ snd (agm_seq x y n) ≥ min x y"
⟨*proof*⟩

lemma *snd_agm_seq_le_max:*
"(x :: real) ≥ 0 ⟹ y ≥ 0 ⟹ snd (agm_seq x y n) ≤ max x y"
⟨*proof*⟩

lemma *fst_agm_seq_le_max:*
"(x :: real) ≥ 0 ⟹ y ≥ 0 ⟹ fst (agm_seq x y n) ≤ max x y"
⟨*proof*⟩

4.2 Convergence of the real AGM

The AGM *always* converges at least linearly:

lemma *dist_agm_seq_le_real:*
assumes *"x ≥ 0" "y ≥ (0 :: real)"*
defines *"a ≡ fst ∘ agm_seq x y"* **and** *"b ≡ snd ∘ agm_seq x y"*
shows *"dist (a n) (b n) ≤ dist x y / 2 ^ n"*
⟨*proof*⟩

definition *agm :: "'a :: gmean ⇒ 'a ⇒ 'a"* **where**
"agm x y = lim (fst ∘ agm_seq x y)"

lemma *agm_commute: "agm x y = agm y x"*
⟨*proof*⟩

lemma *agm_same_real [simp]: "x ≥ 0 ⟹ agm x x = (x :: real)"*
⟨*proof*⟩

lemma *agm_same_complex [simp]: "agm z z = (z :: complex)"*
⟨*proof*⟩

lemma
fixes *x y :: real*
assumes *"x ≥ 0" "y ≥ 0"*
shows *tendsto_agm1_real: "(fst ∘ agm_seq x y) ⟶ agm x y"* (**is** *?th1*)
and *tendsto_agm2_real: "(snd ∘ agm_seq x y) ⟶ agm x y"* (**is**

?th2)
and *agm_seq_le_agm_real: "n > 0 ⟹ snd (agm_seq x y n) ≤ agm x y"* **(is** *"_ ⟹ ?th3")*
and *agm_seq_ge_agm_real: "n > 0 ⟹ fst (agm_seq x y n) ≥ agm x y"* **(is** *"_ ⟹ ?th4")*
⟨*proof*⟩

lemma *tendsto_agm_seq [tendsto_intros]:*
fixes *x y :: real*
assumes *"filterlim f at_top F" "x ≥ 0" "y ≥ 0"*
shows *"filterlim (λt. agm_seq x y (f t)) (nhds (agm x y, agm x y)) F"*
⟨*proof*⟩

lemma *agm_0_right_real [simp]: "agm x 0 = (0 :: real)"*
⟨*proof*⟩

lemma *agm_0_left_real [simp]: "agm 0 (x :: real) = 0"*
⟨*proof*⟩

lemma *agm_real_between_gmean_amean:*
assumes *"x ≥ 0" "y ≥ (0 :: real)"*
shows *"agm x y ∈ {gmean x y..amean x y}"*
⟨*proof*⟩

lemma *agm_real_between:*
assumes *"x ≥ 0" "y ≥ (0 :: real)"*
shows *"agm x y ∈ {min x y..max x y}"*
⟨*proof*⟩

lemma *agm_real_upper_bound:*
"x ∈ {0..c::real} ⟹ y ∈ {0..c::real} ⟹ agm x y ≤ c"
⟨*proof*⟩

lemma *agm_real_lower_bound:*
"x ≥ c ⟹ y ≥ c ⟹ c ≥ (0::real) ⟹ agm x y ≥ c"
⟨*proof*⟩

lemma *agm_strictly_between_real:*
assumes *"x ≠ y" "0 < x" "x < (y :: real)"*
shows *"agm x y ∈ {x<..<y}"*
⟨*proof*⟩

lemma *agm_strictly_between_real':*
assumes *"x ≠ y" "x > 0" "y > 0" "x ≠ (y :: real)"*
shows *"agm x y ∈ open_segment x y"*
⟨*proof*⟩

lemma *agm_amean_gmean_real:*

assumes *"x ≥ 0" "y ≥ (0 :: real)"*
shows *"agm (amean x y) (gmean x y) = agm x y"*
⟨*proof*⟩

lemma *agm_mult_real:*
assumes *"a ≥ (0 :: real)" "x ≥ 0" "y ≥ 0"*
shows *"agm (a * x) (a * y) = a * agm x y"*
⟨*proof*⟩

lemma *agm_mono_real:*
fixes *x y :: real*
assumes *"x ≤ x'" "y ≤ y'" "0 ≤ x" "0 ≤ y"*
shows *"agm x y ≤ agm x' y'"*
⟨*proof*⟩

lemma *agm_pos_real:*
assumes *"a > 0" "b > (0 :: real)"*
shows *"agm a b > 0"*
⟨*proof*⟩

lemma *agm_nonneg_real:*
assumes *"a ≥ 0" "b ≥ (0 :: real)"*
shows *"agm a b ≥ 0"*
⟨*proof*⟩

lemma *uniform_limit_fst_agm_seq_real:*
fixes *X :: "(real × real) set"*
assumes *"bounded X" "X ⊆ {0..} × {0..}"*
shows *"uniform_limit X (λn (x,y). fst (agm_seq x y n)) (λ(x,y). agm x y) sequentially"*
⟨*proof*⟩

lemma *uniform_limit_snd_agm_seq:*
fixes *X :: "(real × real) set"*
assumes *"bounded X" "X ⊆ {0..} × {0..}"*
shows *"uniform_limit X (λn (x,y). snd (agm_seq x y n)) (λ(x,y). agm x y) sequentially"*
⟨*proof*⟩

lemma *continuous_on_agm_seq_real [continuous_intros]:*
assumes *[continuous_intros]: "continuous_on A f" "continuous_on A g"*
assumes *"⋀x. x ∈ A ⟹ f x ≥ 0" "⋀x. x ∈ A ⟹ g x ≥ 0"*
shows *"continuous_on A (λx. agm_seq (f x) (g x) n :: real × real)"*
⟨*proof*⟩

lemma *continuous_on_agm_real [continuous_intros]:*
assumes *"continuous_on A f" "continuous_on A g"*
assumes *"⋀x. x ∈ A ⟹ f x ≥ 0" "⋀x. x ∈ A ⟹ g x ≥ 0"*
shows *"continuous_on A (λx. agm (f x) (g x) :: real)"*

⟨*proof*⟩

lemma *tendsto_agm_real [tendsto_intros]:*
  assumes *"(f ⟶ a) F" "(g ⟶ b) F"*
  assumes *"a ≥ 0" "b ≥ 0"*
  assumes *"a = 0 ⟹ eventually (λx. f x ≥ 0) F"*
  assumes *"b = 0 ⟹ eventually (λx. g x ≥ 0) F"*
  shows *"((λx. agm (f x) (g x) :: real) ⟶ agm a b) F"*
⟨*proof*⟩

lemma *continuous_agm_real [continuous_intros]:*
  assumes *"continuous F f" "continuous F g"*
  assumes *"f (netlimit F) ≥ 0" "g (netlimit F) ≥ 0"*
  assumes *"f (netlimit F) = 0 ⟹ eventually (λx. f x ≥ 0) F"*
  assumes *"g (netlimit F) = 0 ⟹ eventually (λx. g x ≥ 0) F"*
  shows *"continuous F (λx. agm (f x) (g x) :: real)"*
  ⟨*proof*⟩

4.3 Eventual quadratic convergence

We define the sequence $a_n = \sqrt{|a_n^2 - b_n^2|} = \frac{1}{2}|a_{n-1} - b_{n-1}|$.

definition *agm_diff :: "real ⇒ real ⇒ nat ⇒ real"* **where**
  "agm_diff x y n = sqrt |fst (agm_seq x y n) ^ 2 - snd (agm_seq x y n) ^ 2|"

context
  fixes *x y :: real* **and** *a b c :: "nat ⇒ real"*
  assumes *xy: "x ≥ 0" "y ≥ 0"*
  defines *"a ≡ (λn. fst (agm_seq x y n))"* **and** *"b ≡ (λn. snd (agm_seq x y n))"* **and** *"c ≡ agm_diff x y"*
begin

lemma *agm_diff_Suc_eq: "c (Suc n) = dist (a n) (b n) / 2"*
⟨*proof*⟩

We have the recurrence $c_{n+1} = \frac{c_n^2}{4a_{n+1}}$. This already shows that if c_n is less than 1, it converges quadratically to 0, which means that a_n and b_n converge quadratically to the AGM.

lemma *agm_diff_rec_aux: "c (Suc n) * 4 * a (Suc n) = c n ^ 2"*
⟨*proof*⟩

lemma *agm_diff_rec: "c (Suc n) = c n ^ 2 / (4 * a (Suc n))"*
⟨*proof*⟩

end

The following makes the quadratic convergence clearer by removing the dependency on a_n.

lemma *agm_diff_Suc_le:*
assumes *xy: "x > 0" "y > 0"*
shows *"agm_diff x y (Suc n) ≤ agm_diff x y n ^ 2 / (4 * agm x y)"*
⟨*proof*⟩

lemma *agm_diff_le:*
fixes *x y :: real*
assumes *xy: "0 < y" "0 < x"*
defines *"c ≡ (λn. agm_diff x y n)"*
defines *"A ≡ 4 * agm x y"*
shows *"c n ≤ A * (c 0 / A) ^ (2 ^ n)"*
⟨*proof*⟩

context
fixes *a b c :: "real ⇒ real ⇒ nat ⇒ real"*
defines *"a ≡ (λx y n. fst (agm_seq x y n))"*
defines *"b ≡ (λx y n. snd (agm_seq x y n))"*
defines *"c ≡ (λx y n. agm_diff x y n)"*
begin

lemma *agm_seq_shift: "agm_seq x y (n + m) = agm_seq (a x y n) (b x y n) m"*
⟨*proof*⟩

lemma *agm_diff_shift: "c x y (n + m) = c (a x y n) (b x y n) m"*
⟨*proof*⟩

lemma *agm_agm_seq_eq:*
assumes *"x ≥ 0" "y ≥ (0 :: real)"*
shows *"agm (fst (agm_seq x y n)) (snd (agm_seq x y n)) = agm x y"*
⟨*proof*⟩

lemma *agm_diff_le':*
fixes *x y :: real*
assumes *xy: "0 < x" "0 < y"*
defines *"A ≡ 4 * agm x y"*
assumes *"n ≥ n0"*
shows *"c x y n ≤ 4 * agm x y * (c x y n0 / (4 * agm x y)) ^ (2 ^ (n - n0))"*
⟨*proof*⟩

lemma *agm_diff_tendsto_0:*
assumes *"x ≥ 0" "y ≥ (0 :: real)"*
shows *"c x y ⟶ 0"*
⟨*proof*⟩

Finally, we explicitly prove quadratic convergence of c_n.

theorem *agm_diff_bigo:*
fixes *x y :: real*

assumes *xy: "0 < x" "0 < y"*
obtains *C* **where** *"C ∈ {0<..<1}" "c x y ∈ O(λn. C ^ (2 ^ n))"*
⟨*proof*⟩

end

4.4 The AGM iteration in the complex plane

lemma *agm_seq_complex_of_real:*
assumes *"x ≥ 0" "y ≥ 0"*
shows *"agm_seq (complex_of_real x) (of_real y) n = map_prod of_real of_real (agm_seq x y n)"*
⟨*proof*⟩

Under mild preconditions, the arguments of x and y after an AGM iteration lie between the arguments of x and y, i.e. the values we get lie in a cone that shrinks with every iteration.

lemma
assumes *"x ≠ 0" "y ≠ 0" "dist (Arg x) (Arg y) < pi"*
shows *Arg_agm_seq:*
"Arg (fst (agm_seq x y n)) ∈ closed_segment (Arg x) (Arg y)" **(is** *?th1)*
"Arg (snd (agm_seq x y n)) ∈ closed_segment (Arg x) (Arg y)" **(is** *?th2)*
"dist (Arg (fst (agm_seq x y n))) (Arg (snd (agm_seq x y n))) < pi"
and *agm_seq_complex_nonzero:*
"fst (agm_seq x y n) ≠ 0" **(is** *?th3)* *"snd (agm_seq x y n) ≠ 0"* **(is** *?th4)*
⟨*proof*⟩

Similarly, the norm of the values also never grows beyond the norm of the bigger one.

lemma *norm_agm_seq_complex_le:*
fixes *x y :: complex*
assumes *"x ≠ 0" "y ≠ 0" "dist (Arg x) (Arg y) < pi"*
shows *"norm (fst (agm_seq x y n)) ≤ max (norm x) (norm y)"* **(is** *?th1)*
and *"norm (snd (agm_seq x y n)) ≤ max (norm x) (norm y)"* **(is** *?th2)*
⟨*proof*⟩

The distance between a_n and b_n is bounded by $|a_0 - b_0|/2^n$ and therefore vanishes.

lemma *dist_agm_seq_complex_le:*
fixes *w z :: complex*
shows *"dist (fst (agm_seq w z n)) (snd (agm_seq w z n)) ≤ dist w z / 2 ^ n"*
⟨*proof*⟩

After an AGM iteration, the values are never opposite in the complex plane (even if they were before).

lemma *not_opposite_complex_amean_gmean:*
  fixes *w z :: complex*
  shows *"¬opposite_complex (amean w z) (gmean w z)"*
⟨*proof*⟩

lemma *not_opposite_complex_agm_seq:*
  assumes *"n > 0 ∨ ¬opposite_complex w z"*
  shows *"¬opposite_complex (fst (agm_seq w z n)) (snd (agm_seq w z n))"*
⟨*proof*⟩

The result of an AGM iteration is zero if and only if one of the input numbers or their sum was zero.

lemma *agm_seq_complex_nonzero':*
  assumes *"w + z ≠ 0" "w ≠ 0" "z ≠ (0::complex)"*
  shows *"fst (agm_seq w z n) ≠ 0" "snd (agm_seq w z n) ≠ 0"*
⟨*proof*⟩

lemma *agm_seq_mult_left_complex:*
  assumes *"n = 0 ∨ ¬opposite_complex w z"*
  shows *"fst (agm_seq (u * w) (u * z) n) = u * fst (agm_seq w z n)"* (**is** *?th1*)
    and *"snd (agm_seq (u * w) (u * z) n) = u * snd (agm_seq w z n)"* (**is** *?th2*)
⟨*proof*⟩

lemma *agm_seq_minus_complex:*
  assumes *"n = 0 ∨ ¬opposite_complex w z"*
  shows *"fst (agm_seq (-w) (-z) n) = -fst (agm_seq w z n)"* (**is** *?th1*)
    and *"snd (agm_seq (-w) (-z) n) = -snd (agm_seq w z n)"* (**is** *?th2*)
  ⟨*proof*⟩

We now analyse the phase difference between a_n and b_n.

definition *agm_phase_diff :: "complex ⇒ complex ⇒ nat ⇒ real"* **where**
  "agm_phase_diff w z n =
     *(let φ = dist (Arg (fst (agm_seq w z n))) (Arg (snd (agm_seq w z n))) in min φ (2 * pi - φ))"*

The phase difference is at least cut in half by every iteration.

lemma *abs_agm_seq_phase_diff_Suc_le_aux:*
  assumes *"w ≠ 0" "z ≠ 0" "dist (Arg w) (Arg z) < pi"*
    shows *"agm_phase_diff w z (Suc n) ≤ agm_phase_diff w z n / 2"*
⟨*proof*⟩

lemma *agm_phase_diff_minus:*
  assumes *"w ≠ 0" "z ≠ 0" "¬opposite_complex w z"*

shows *"agm_phase_diff (-w) (-z) n = agm_phase_diff w z n"*
⟨*proof*⟩

lemma *agm_phase_diff_commute: "agm_phase_diff w z n = agm_phase_diff z w n"*
⟨*proof*⟩

lemma *agm_phase_diff_bounded: "agm_phase_diff w z n ∈ {0..pi}"*
⟨*proof*⟩

lemma *agm_phase_diff_rec: "agm_phase_diff w z (Suc n) = agm_phase_diff (amean w z) (gmean w z) n"*
⟨*proof*⟩

lemma *abs_agm_seq_phase_diff_Suc_le:*
assumes *"w ≠ 0" "z ≠ 0" "w + z ≠ 0"*
shows *"agm_phase_diff w z (Suc n) ≤ agm_phase_diff w z n / 2"*
⟨*proof*⟩

lemma *abs_agm_seq_phase_diff_le:*
assumes *"w ≠ 0" "z ≠ 0" "w + z ≠ 0"*
shows *"agm_phase_diff w z n ≤ agm_phase_diff w z 0 / 2 ^ n"*
⟨*proof*⟩

lemma *abs_agm_seq_phase_diff_le':*
assumes *"w ≠ 0" "z ≠ 0" "w + z ≠ 0"*
shows *"agm_phase_diff w z n ≤ pi / 2 ^ n"*
⟨*proof*⟩

lemma *continuous_on_agm_seq_complex_aux:*
assumes *[continuous_intros]: "continuous_on A f" "continuous_on A g"*
assumes *not_op: "⋀z. z ∈ A ⟹ f z ≠ 0 ∧ g z ≠ 0 ∧ ¬opposite_complex (f z) (g z)"*
shows *"continuous_on A (λz. fst (agm_seq (f z) (g z) n))"* (**is** *?th1*)
"continuous_on A (λz. snd (agm_seq (f z) (g z) n))" (**is** *?th2*)
⟨*proof*⟩

lemma *continuous_on_agm_seq_complex [continuous_intros]:*
assumes *"continuous_on A f" "continuous_on A g"*
assumes *not_op: "⋀z. z ∈ A ⟹ f z ≠ 0 ∧ g z ≠ 0 ∧ ¬opposite_complex (f z) (g z)"*
shows *"continuous_on A (λz. agm_seq (f z) (g z) n)"*
⟨*proof*⟩

lemma *holomorphic_agm_seq [holomorphic_intros]:*
assumes *[holomorphic_intros]: "f holomorphic_on A" "g holomorphic_on A"*
assumes *not_op: "⋀z. z ∈ A ⟹ f z ≠ 0 ∧ g z ≠ 0 ∧ ¬opposite_complex (f z) (g z)"*

shows *"(λz. fst (agm_seq (f z) (g z) n)) holomorphic_on A"* (**is** *?th1*)
"(λz. snd (agm_seq (f z) (g z) n)) holomorphic_on A" (**is** *?th2*)
⟨*proof*⟩

lemma *analytic_agm_seq:*
assumes *[analytic_intros]: "f analytic_on A" "g analytic_on A"*
assumes *not_op: "⋀z. z ∈ A ⟹ f z ≠ 0 ∧ g z ≠ 0 ∧ ¬opposite_complex (f z) (g z)"*
shows *"(λz. fst (agm_seq (f z) (g z) n)) analytic_on A"* (**is** *?th1*)
"(λz. snd (agm_seq (f z) (g z) n)) analytic_on A" (**is** *?th2*)
⟨*proof*⟩

4.5 Convergence of the complex AGM

We now have all the ingredients to show that the complex AGM always converges. Note however that the square root present in the definition of the geometric mean necessarily introduces a branch cut when the input paramers lie opposite of each other.

definition *agm_cball :: "complex ⇒ complex ⇒ complex set"* **where**
"agm_cball w z = cball (amean w z) (dist w z / 2)"

lemma *point_in_agm_cball [simp, intro]: "w ∈ agm_cball w z" "z ∈ agm_cball w z"*
⟨*proof*⟩

lemma *amean_gmean_in_cball:*
fixes *w z :: complex*
defines *"w' ≡ amean w z"* **and** *"z' ≡ gmean w z"*
shows *"w' ∈ agm_cball w z" "z' ∈ agm_cball w z" "agm_cball w' z' ⊆ agm_cball w z"*
⟨*proof*⟩

lemma *agm_cball_minus: "agm_cball (-w) (-z) = uminus ‘ agm_cball w z"*
⟨*proof*⟩

lemma *cos_gt_zero':*
assumes *"x ∈ {-pi/2<..<pi/2}"*
shows *"cos x > 0"*
⟨*proof*⟩

lemma *zero_in_agm_cball_iff:*
"0 ∈ agm_cball w z ⟷
*w = 0 ∨ z = 0 ∨ min (dist (Arg w) (Arg z)) (2 * pi - dist (Arg w) (Arg z)) ≥ pi / 2"*
⟨*proof*⟩

definition *agm_seq_cball :: "complex ⇒ complex ⇒ nat ⇒ complex set"*

where
"agm_seq_cball w z n = agm_cball (fst (agm_seq w z n)) (snd (agm_seq w z n))"

lemma *agm_seq_cball_0 [simp]: "agm_seq_cball w z 0 = agm_cball w z"*
⟨*proof*⟩

lemma *decseq_agm_seq_cball: "decseq (agm_seq_cball w z)"*
⟨*proof*⟩

lemma *agm_seq_in_cball:*
assumes *"m ≤ n"*
shows *"fst (agm_seq w z n) ∈ agm_seq_cball w z m"* (**is** *?th1*)
and *"snd (agm_seq w z n) ∈ agm_seq_cball w z m"* (**is** *?th2*)
⟨*proof*⟩

theorem
fixes *w z :: complex*
shows *tendsto_agm1_complex: "(fst ∘ agm_seq w z) ⟶ agm w z"*
and *tendsto_agm2_complex: "(snd ∘ agm_seq w z) ⟶ agm w z"*
⟨*proof*⟩

lemmas *tendsto_agm1_complex' = tendsto_agm1_complex[unfolded o_def]*
lemmas *tendsto_agm2_complex' = tendsto_agm2_complex[unfolded o_def]*

lemma *agm_complex_of_real:*
assumes *"x ≥ 0" "y ≥ 0"*
shows *"agm (complex_of_real x) (complex_of_real y) = complex_of_real (agm x y)"*
⟨*proof*⟩

lemma *agm_amean_gmean_complex:*
"agm (amean w z) (gmean w z :: complex) = agm w z"
⟨*proof*⟩

lemma *agm_in_cball:*
fixes *w z :: complex*
shows *"agm w z ∈ agm_seq_cball w z n"*
⟨*proof*⟩

lemma *dist_agm_agm_seq_complex_le:*
fixes *w z :: complex*
shows *"dist (agm w z) (fst (agm_seq w z n)) ≤ dist w z / 2 ^ n"*
and *"dist (agm w z) (snd (agm_seq w z n)) ≤ dist w z / 2 ^ n"*
⟨*proof*⟩

lemma *agm_eq_0_complex_iff: "agm w z = 0 ⟷ w = 0 ∨ z = (0::complex)*

```
∨ w + z = 0"
```
⟨*proof*⟩

lemma *agm_0_left_complex [simp]: "agm 0 z = (0::complex)"*
⟨*proof*⟩

lemma *agm_0_right_complex [simp]: "agm z 0 = (0::complex)"*
⟨*proof*⟩

lemma *Arg_agm_bounds:*
assumes *"w ≠ 0" "z ≠ 0" "dist (Arg w) (Arg z) < pi"*
shows *"Arg (agm w z) ∈ closed_segment (Arg w) (Arg z)"*
⟨*proof*⟩

lemma *uniform_limit_fst_agm_seq_complex:*
fixes *X :: "(complex × complex) set"*
assumes *"bounded X"*
shows *"uniform_limit X (λn (w,z). fst (agm_seq w z n)) (λ(w,z). agm w z) sequentially"*
⟨*proof*⟩

lemma *uniform_limit_snd_agm_seq_complex:*
fixes *X :: "(complex × complex) set"*
assumes *"bounded X"*
shows *"uniform_limit X (λn (w,z). snd (agm_seq w z n)) (λ(w,z). agm w z) sequentially"*
⟨*proof*⟩

lemma *agm_mult_complex:*
assumes *"¬opposite_complex w z"*
shows *"agm (u * w) (u * z) = u * agm w z"*
⟨*proof*⟩

lemma *agm_1_rec_complex: "agm 1 (z :: complex) = amean 1 z * agm 1 ((2 * csqrt z) / (1 + z))"*
⟨*proof*⟩

lemma *agm_1_rec_complex': "agm 1 (z :: complex) = gmean 1 z * agm 1 ((1 + z) / (2 * csqrt z))"*
⟨*proof*⟩

The following may be interesting: $\mathrm{agm}(1, z) \in \mathbb{R}$ only if the input is real.

lemma *agm_1_in_Reals_imp_Real:*
fixes *z :: complex*
assumes *"agm 1 z ∈ ℝ" "z ∉ ℝ≤0"*
shows *"Im z = 0 ∧ Re z > 0"*
⟨*proof*⟩

lemma *agm_holomorphic [holomorphic_intros]:*
assumes *"f holomorphic_on A" "g holomorphic_on A"*
assumes *"⋀z. z ∈ A ⟹ g z / f z ∉ ℝ≤0"*
shows *"(λz. agm (f z) (g z)) holomorphic_on A"*
⟨*proof*⟩

lemma *agm_analytic [analytic_intros]:*
assumes *"f analytic_on A" "g analytic_on A"*
assumes *"⋀z. z ∈ A ⟹ g z / f z ∉ ℝ≤0"*
shows *"(λz. agm (f z) (g z)) analytic_on A"*
⟨*proof*⟩

lemma *continuous_on_agm_complex [continuous_intros]:*
assumes *"continuous_on A f" "continuous_on A g"*
assumes *"⋀z. z ∈ A ⟹ g z / f z ∉ ℝ≤0"*
shows *"continuous_on A (λz. agm (f z) (g z) :: complex)"*
⟨*proof*⟩

lemma *tendsto_agm_complex [tendsto_intros]:*
fixes *w z :: complex*
assumes *"(f ⟶ w) F" "(g ⟶ z) F" "z / w ∉ ℝ≤0"*
shows *"((λx. agm (f x) (g x)) ⟶ agm w z) F"*
⟨*proof*⟩

lemma *continuous_agm_complex [continuous_intros]:*
assumes *"continuous F f" "continuous F g" "g (netlimit F) / f (netlimit F) ∉ ℝ≤0"*
shows *"continuous F ((λx. agm (f x) (g x) :: complex))"*
⟨*proof*⟩

end

5 Complete Elliptic Integrals

theory *Complete_Elliptic_Integrals*
imports
"Generalized_Hypergeometric_Series.Generalized_Hypergeometric_Series"
"Incomplete_Gamma.More_Beta"
AGM_Lemma_Bucket
begin

In this section, we will introduce the complete elliptic integrals of the first and second kind, written as $K(m)$ and $E(m)$, respectively.

There is a big caveat concerning the argument of K and E: the m that we use is simply called the *parameter* in the literature. Many books write K in terms of the *modulus* $k = m^2$ instead. The *modular angle* α with $\sin\alpha = k$ can also be used.

Since it is fairly easy to convert any of the other two to m whereas the

reverse direction may require dealing with branch cuts, we chose m as the argument. This is also the choice made e.g. in Mathematica.

lemma *of_real_real: "of_real x = x"*
⟨*proof*⟩

lemma *elliptic_integral_wf_aux2:*
assumes *"x < (1 :: real)" "y ∈ {0..1}"*
shows *"x * y < 1"*
⟨*proof*⟩

lemma *elliptic_integral_wf_aux:*
assumes *"x < (1 :: real)"*
shows *"x * sin y ^ 2 < 1"*
⟨*proof*⟩

5.1 Complete elliptic integrals of the first and second kind

5.1.1 Generic theorems about both functions

Instead of proving everything for the elliptic integrals of K and E separately, we generalise to an arbitrary exponent e and later instantiate it with $e = -\frac{1}{2}$ and $e = \frac{1}{2}$, respectively.

locale *complete_elliptic_integral_gen =*
fixes *ellipticr :: "real ⇒ real"* **and** *ellipticc :: "complex ⇒ complex"* **and** *e :: real*
assumes *real_def:*
*"⋀m. m < 1 ⟹ ellipticr m = integral {0..pi/2} (λt. (1 - m * sin t ^ 2) powr e)"*
assumes *complex_def:*
"⋀m. Im m ≠ 0 ∨ Re m < 1 ⟹
*ellipticc m = integral {0..pi/2} (λt. (1 - m * of_real (sin t ^ 2)) powr of_real e)"*
begin

lemma *at_0_complex [simp]: "ellipticc 0 = of_real pi / 2"*
⟨*proof*⟩

lemma *at_0_real [simp]: "ellipticr 0 = pi / 2"*
⟨*proof*⟩

lemma *pos_real:*
assumes *"m < (1 :: real)"*
shows *"ellipticr m > 0"*
⟨*proof*⟩

lemma *nonzero_real [simp]:*
assumes *"m < (1 :: real)"*
shows *"ellipticr m ≠ 0"*

⟨*proof*⟩

lemma *absolutely_integrable_real:*
assumes *"m < (1 :: real)"*
shows *"(λt::real. (1 - m * sin t ^ 2) powr e) absolutely_integrable_on {0..pi/2}"*
⟨*proof*⟩

lemma *absolutely_integrable_complex:*
assumes *"Im m ≠ 0 ∨ Re m < 1"*
shows *"(λt::real. (1 - m * of_real (sin t ^ 2)) powr e) absolutely_integrable_on {0..pi/2}"*
⟨*proof*⟩

lemma *has_integral_real:*
assumes *"m < (1 :: real)"*
shows *"((λt. (1 - m * sin t ^ 2) powr e) has_integral ellipticr m) {0..pi/2}"*
⟨*proof*⟩

lemma *has_integral_complex:*
fixes *m :: complex*
assumes *"Im m ≠ 0 ∨ Re m < 1"*
shows *"((λt. (1 - m * of_real (sin t ^ 2)) powr e) has_integral ellipticc m) {0..pi/2}"*
⟨*proof*⟩

lemma *has_integral_real2:*
assumes *m: "m < (1 :: real)"*
shows *"(λx. (1 - m * x ^ 2) powr e / sqrt (1 - x ^ 2)) absolutely_integrable_on {0..1}"* **(is** *?th1)*
and *"((λx. (1 - m * x ^ 2) powr e / sqrt (1 - x ^ 2)) has_integral ellipticr m) {0..1}"* **(is** *?th2)*
⟨*proof*⟩

lemma *has_integral_complex2:*
assumes *"Im m ≠ 0 ∨ Re m < 1"*
shows *"(λx. (1 - m * of_real x ^ 2) powr e / sqrt (1 - x ^ 2)) absolutely_integrable_on {0..1}"* **(is** *?th1)*
and *"((λx. (1 - m * of_real x ^ 2) powr e / sqrt (1 - x ^ 2)) has_integral ellipticc m) {0..1}"* **(is** *?th2)*
⟨*proof*⟩

lemma *has_field_derivative_complex:*
fixes *m :: complex*
assumes *m: "Im m ≠ 0 ∨ Re m < 1"*
defines *"f' ≡ (λm t. -of_real e * (1 - m * complex_of_real (sin t) ^ 2) powr (of_real e - 1) * (of_real (sin t) ^ 2))"*

shows *"(ellipticc has_field_derivative integral {0..pi/2} (λt. f' m t)) (at m within A)"*
⟨*proof*⟩

lemma *holomorphic:*
assumes *"⋀z. z ∈ A ⟹ Im z ≠ 0 ∨ Re z < 1"*
shows *"ellipticc holomorphic_on A"*
⟨*proof*⟩

lemma *holomorphic' [holomorphic_intros]:*
assumes *"f holomorphic_on A" "⋀z. z ∈ A ⟹ Im (f z) ≠ 0 ∨ Re (f z) < 1"*
shows *"(λz. ellipticc (f z)) holomorphic_on A"*
⟨*proof*⟩

lemma *analytic [analytic_intros]:*
assumes *"f analytic_on A" "⋀z. z ∈ A ⟹ Im (f z) ≠ 0 ∨ Re (f z) < 1"*
shows *"(λz. ellipticc (f z)) analytic_on A"*
⟨*proof*⟩

lemma *continuous_on_complex:*
assumes *"continuous_on A f" "⋀z. z ∈ A ⟹ Im (f z) ≠ 0 ∨ Re (f z) < 1"*
shows *"continuous_on A (λz. ellipticc (f z))"*
⟨*proof*⟩

lemma *continuous_on_real:*
assumes *"continuous_on A f" "⋀x. x ∈ A ⟹ f x < 1"*
shows *"continuous_on A (λx. ellipticr (f x :: real))"*
⟨*proof*⟩

lemma *continuous_complex:*
assumes *"continuous (at x within A) f" "Im (f x) ≠ 0 ∨ Re (f x) < 1"*
shows *"continuous (at x within A) (λz. ellipticc (f z))"*
⟨*proof*⟩

lemma *continuous_real:*
assumes *"continuous (at x within A) f" "f x < (1 :: real)"*
shows *"continuous (at x within A) (λz. ellipticr (f z))"*
⟨*proof*⟩

lemma *tendsto_complex:*
assumes *"(f ⟶ x) F" "Im x ≠ 0 ∨ Re x < 1"*
shows *"((λz. ellipticc (f z)) ⟶ ellipticc x) F"*
⟨*proof*⟩

lemma *tendsto_real:*

assumes *"(f ⟶ x) F" "x < 1"*
shows *"((λz. ellipticr (f z)) ⟶ ellipticr x) F"*
⟨*proof*⟩

lemma *cnj:*
assumes *"Im x ≠ 0 ∨ Re x < 1"*
shows *"ellipticc (cnj x) = cnj (ellipticc x)"*
⟨*proof*⟩

lemma *of_real:*
assumes *"x < 1"*
shows *"ellipticc (of_real x) = of_real (ellipticr x)"*
⟨*proof*⟩

lemma *of_real':*
assumes *"Im x = 0" "Re x < 1"*
shows *"ellipticc x = of_real (ellipticr (Re x))"*
⟨*proof*⟩

lemma *power_series_complex:*
assumes *z: "norm z < 1"*
defines *"h ≡ (λn. pi/2 * pochhammer (-e) n * pochhammer (1/2) n / fact n ^ 2)"*
shows *"(λn. of_real (h n) * z ^ n) sums ellipticc z"*
⟨*proof*⟩

lemma *power_series_real:*
assumes *x: "|x| < (1 :: real)"*
defines *"h ≡ (λn. pi/2 * pochhammer (-e) n * pochhammer (1/2) n / fact n ^ 2)"*
shows *"(λn. h n * x ^ n) sums ellipticr x"*
⟨*proof*⟩

lemma *conv_hypergeo_F_complex:*
assumes *z: "norm (z :: complex) < 1"*
shows *"ellipticc z = of_real (pi / 2) * hypergeo_F [-of_real e, 1/2] [1] z"*
⟨*proof*⟩

lemma *conv_hypergeo_F_real:*
assumes *x: "|x| < 1"*
shows *"ellipticr x = pi / 2 * hypergeo_F [-e, 1/2] [1] x"*
⟨*proof*⟩

lemma *has_fps_expansion_complex [fps_expansion_intros]:*
*"ellipticc has_fps_expansion (fps_const (of_real pi / 2) * fps_hypergeo [-of_real e, 1/2] [1] 1)"*
⟨*proof*⟩

lemma *has_fps_expansion_real [fps_expansion_intros]:*
*"ellipticr has_fps_expansion (fps_const (pi/2) * fps_hypergeo [-e, 1/2] [1] 1)"*
⟨*proof*⟩

lemmas *has_laurent_expansion_complex [laurent_expansion_intros] = has_laurent_expansion_fps [OF has_fps_expansion_complex]*

end

locale *complete_elliptic_integral_gen' =*
complete_elliptic_integral_gen +
fixes *rpowr :: "real ⇒ real"* **and** *cpowr :: "complex ⇒ complex"*
assumes *rpowr_eq: "⋀x. x ≥ 0 ⟹ x powr e = rpowr x"*
assumes *cpowr_eq: "⋀x. x ≠ 0 ⟹ x powr e = cpowr x"*
begin

lemma *real_def':*
assumes *"m < 1"*
shows *"ellipticr m = integral {0..pi / 2} (λt. rpowr (1 - m * sin t ^ 2))"*
⟨*proof*⟩

lemma *complex_def':*
assumes *"Im m ≠ 0 ∨ Re m < 1"*
shows *"ellipticc m = integral {0..pi / 2} (λt. cpowr (1 - m * of_real (sin t ^ 2)))"*
⟨*proof*⟩

lemma *has_integral_real':*
assumes *"m < (1 :: real)"*
shows *"((λt. rpowr (1 - m * sin t ^ 2)) has_integral ellipticr m) {0..pi/2}"*
⟨*proof*⟩

lemma *absolutely_integrable_real':*
assumes *"m < (1 :: real)"*
shows *"(λt::real. rpowr (1 - m * sin t ^ 2)) absolutely_integrable_on {0..pi/2}"*
⟨*proof*⟩

lemma *has_integral_complex':*
assumes *"Im m ≠ 0 ∨ Re m < 1"*
shows *"((λt. cpowr (1 - m * of_real (sin t ^ 2))) has_integral ellipticc m) {0..pi/2}"*
⟨*proof*⟩

lemma *has_integral_real2':*

assumes *m: "m < (1 :: real)"*
shows *"(λx. rpowr (1 - m * x ^ 2) / sqrt (1 - x ^ 2)) absolutely_integrable_on {0..1}"* (**is** *?th1*)
and *"((λx. rpowr (1 - m * x ^ 2) / sqrt (1 - x ^ 2)) has_integral ellipticr m) {0..1}"* (**is** *?th2*)
⟨*proof*⟩

lemma *has_integral_complex2':*
assumes *"Im m ≠ 0 ∨ Re m < 1"*
shows *"(λx. cpowr (1 - m * of_real x ^ 2) / sqrt (1 - x ^ 2)) absolutely_integrable_on {0..1}"* (**is** *?th1*)
and *"((λx. cpowr (1 - m * of_real x ^ 2) / sqrt (1 - x ^ 2)) has_integral ellipticc m) {0..1}"* (**is** *?th2*)
⟨*proof*⟩

end

We now instantiate this generic locale concretely for the elliptic integrals of the first and second kind, denoted by K and E, respectively.

definition *elliptic_K :: "'a :: {real_normed_field, ln} ⇒ 'a"* **where**
*"elliptic_K m = integral {0..pi/2} (λt. (1 - m * of_real (sin t ^ 2)) powr (-of_real (1/2)))"*

interpretation *elliptic_K:*
complete_elliptic_integral_gen' elliptic_K elliptic_K "-1/2" "λx. 1 / sqrt x" "λx. 1 / csqrt x"
rewrites *"x / y / z = x / (y * z :: 'a :: field)"* **and**
*"pi / 2 * pochhammer (1 / 2) n * pochhammer (1 / 2) n / (fact n)² =*
*pi / 2 * pochhammer (1 / 2) n ^ 2 / fact n ^ 2"* **and**
"- (- 1 / 2) = 1 / (2 :: real)" **and**
"- (- 1 / 2) = 1 / (2 :: complex)" **and**
"complex_of_real (- 1 / 2) = - 1 / 2"
⟨*proof*⟩

lemmas *[continuous_intros] =*
elliptic_K.continuous_on_complex elliptic_K.continuous_on_real
elliptic_K.continuous_complex elliptic_K.continuous_real

lemmas *[tendsto_intros] =*
elliptic_K.tendsto_complex elliptic_K.tendsto_real

thm *elliptic_K.power_series_real*
thm *elliptic_K.power_series_complex*
thm *elliptic_K.conv_hypergeo_F_real*
thm *elliptic_K.conv_hypergeo_F_complex*

definition *elliptic_E :: "'a :: {real_normed_field, ln} ⇒ 'a"* **where**
*"elliptic_E m = integral {0..pi/2} (λt. (1 - m * of_real (sin t ^ 2)) powr (of_real (1/2)))"*

interpretation *elliptic_E:*
complete_elliptic_integral_gen' elliptic_E elliptic_E "1/2" "λx. sqrt x" "λx. csqrt x"
rewrites *"complex_of_real (1 / 2) = 1 / 2"* **and** *"- (1 / 2 :: complex) = - 1 / 2"*
and *"- (1 / 2 :: real) = - 1 / 2"*
⟨*proof*⟩

thm *elliptic_E.power_series_real*
thm *elliptic_E.power_series_complex*
thm *elliptic_E.conv_hypergeo_F_real*
thm *elliptic_E.conv_hypergeo_F_complex*

lemmas *[continuous_intros] =*
elliptic_E.continuous_on_complex elliptic_E.continuous_complex

lemmas *[tendsto_intros] = elliptic_E.tendsto_complex*

lemma *elliptic_E_1_complex [simp]: "elliptic_E (1 :: complex) = 1"*
⟨*proof*⟩

lemma *elliptic_E_1_real [simp]: "elliptic_E (1 :: real) = 1"*
⟨*proof*⟩

lemma *elliptic_E_continuous_on_real [continuous_intros]:*
assumes *"continuous_on A f"* *"⋀x. x ∈ A ⟹ f x ≤ (1::real)"*
shows *"continuous_on A (λx. elliptic_E (f x))"*
⟨*proof*⟩

lemma *elliptic_E_tendsto_real [tendsto_intros]:*
assumes *"(f ⟶ (x :: real)) F"*
assumes *"x < 1 ∨ x = 1 ∧ eventually (λy. f y ≤ 1) F"*
shows *"((λx. elliptic_E (f x)) ⟶ elliptic_E x) F"*
⟨*proof*⟩

lemma *elliptic_E_continuous_real [continuous_intros]:*
assumes *"continuous (at x within A) f"*
assumes *"f x < 1 ∨ f x = (1::real) ∧ eventually (λy. f y ≤ 1) (at x within A)"*
shows *"continuous (at x within A) (λx. elliptic_E (f x))"*
⟨*proof*⟩

5.1.2 Complete elliptic integral of the third kind

definition *elliptic_Pi :: "'a :: {real_normed_field, ln} ⇒ 'a ⇒ 'a"* **where**
"elliptic_Pi n m = integral {0..pi/2}
*(λt. (1 - m * of_real (sin t ^ 2)) powr (-1/2) / (1 - n * of_real*
(sin t ^ 2)))"

lemma *elliptic_Pi_0_left [simp]: "elliptic_Pi 0 m = elliptic_K m"*
⟨*proof*⟩

lemma *elliptic_Pi_has_integral_complex:*
fixes *m n :: complex*
assumes *"Im n ≠ 0 ∨ Re n < 1" "Im m ≠ 0 ∨ Re m < 1"*
shows *"((λt. (1 - m * of_real (sin t ^ 2)) powr (-1/2) / (1 - n * of_real*
(sin t ^ 2)))
has_integral elliptic_Pi n m) {0..pi/2}"
⟨*proof*⟩

lemma *elliptic_Pi_has_integral_real:*
fixes *m n :: real*
assumes *"n < 1" "m < 1"*
shows *"((λt. (1 - m * sin t ^ 2) powr (-1/2) / (1 - n * sin t ^ 2))*
has_integral elliptic_Pi n m) {0..pi/2}"
⟨*proof*⟩

lemma *continuous_on_elliptic_Pi_real [continuous_intros]:*
fixes *f g :: "'a :: topological_space ⇒ real"*
assumes *"continuous_on A f" "continuous_on A g" "⋀x. x ∈ A ⟹ f x*
< 1" "⋀x. x ∈ A ⟹ g x < 1"
shows *"continuous_on A (λx. elliptic_Pi (f x) (g x))"*
⟨*proof*⟩

lemma *continuous_on_elliptic_Pi_complex [continuous_intros]:*
fixes *f g :: "'a :: topological_space ⇒ complex"*
assumes *"continuous_on A f" "continuous_on A g"*
assumes *"⋀x. x ∈ A ⟹ Im (f x) ≠ 0 ∨ Re (f x) < 1"*
assumes *"⋀x. x ∈ A ⟹ Im (g x) ≠ 0 ∨ Re (g x) < 1"*
shows *"continuous_on A (λx. elliptic_Pi (f x) (g x))"*
⟨*proof*⟩

5.1.3 Derivatives and antiderivatives

lemma *fls_deriv_elliptic_K:*
assumes *"SORT_CONSTRAINT('a :: field_char_0)"*
defines *"E ≡ fps_to_fls (fps_hypergeo [-(1/2), 1/2] [1 :: 'a] 1)"*
defines *"K ≡ fps_to_fls (fps_hypergeo [1/2, 1/2] [1 :: 'a] 1)"*
shows *"fls_deriv K = (E - (1 - fls_X) * K) / (2 * fls_X * (1 - fls_X))"*
⟨*proof*⟩

lemma

assumes *z: "Im z ≠ 0 ∨ Re z < 1" "z ≠ 0"*
shows *deriv_elliptic_E: "deriv elliptic_E z = (elliptic_E z - elliptic_K z) / (2 * z)"*
and *deriv_elliptic_K: "deriv elliptic_K z = (elliptic_E z - (1 - z) * elliptic_K z) / (2 * z * (1 - z))"*
⟨*proof*⟩

lemma *has_field_derivative_elliptic_E_complex [derivative_intros]:*
assumes *"(f has_field_derivative f') (at z within A)"*
assumes *z: "Im (f z) ≠ 0 ∨ Re (f z) < 1"*
defines *"d ≡ (if f z = 0 then -(of_real pi / 8) else (elliptic_E (f z) - elliptic_K (f z)) / (2 * f z))"*
shows *"((λz. elliptic_E (f z)) has_field_derivative d * f') (at z within A)"*
⟨*proof*⟩

lemma *has_field_derivative_elliptic_K_complex [derivative_intros]:*
assumes *"(f has_field_derivative f') (at z within A)"*
assumes *z: "Im (f z) ≠ 0 ∨ Re (f z) < 1"*
defines *"d ≡ (if f z = 0 then (of_real pi / 8) else (elliptic_E (f z) - (1 - f z) * elliptic_K (f z)) / (2 * f z * (1 - f z)))"*
shows *"((λz. elliptic_K (f z)) has_field_derivative d * f') (at z within A)"*
⟨*proof*⟩

lemma *has_field_derivative_elliptic_E_real [derivative_intros]:*
assumes *"(f has_field_derivative f') (at z within A)"*
assumes *z: "f z < (1 :: real)"*
defines *"d ≡ (if f z = 0 then -(pi / 8) else (elliptic_E (f z) - elliptic_K (f z)) / (2 * f z))"*
shows *"((λz. elliptic_E (f z)) has_field_derivative d * f') (at z within A)"*
⟨*proof*⟩

lemma *has_field_derivative_elliptic_K_real [derivative_intros]:*
assumes *"(f has_field_derivative f') (at z within A)"*
assumes *z: "f z < (1 :: real)"*
defines *"d ≡ (if f z = 0 then (of_real pi / 8) else (elliptic_E (f z) - (1 - f z) * elliptic_K (f z)) / (2 * f z * (1 - f z)))"*
shows *"((λz. elliptic_K (f z)) has_field_derivative d * f') (at z within A)"*
⟨*proof*⟩

lemma *antiderivative_elliptic_K_complex:*
assumes *"Im m ≠ 0 ∨ Re m < 1"*
shows *"((λm. 2 * (m - 1) * elliptic_K m + 2 * elliptic_E m) has_field_derivative elliptic_K m) (at m within A)"*

⟨*proof*⟩

lemma *antiderivative_elliptic_E_complex:*
assumes *"Im m ≠ 0 ∨ Re m < 1"*
shows *"((λm. 2 / 3 * (m - 1) * elliptic_K m + 2 / 3 * (m + 1) * elliptic_E m)*
has_field_derivative elliptic_E m) (at m within A)"
⟨*proof*⟩

5.2 Legendre's relation

Legendre's relation states that

$$K(m)E(1-m) + E(m)K(1-m) - K(m)K(1-m) = \tfrac{\pi}{2}\ .$$

We first show that it holds for reals (necessarily constrained to $m \in (0,1)$) and then lift it to complex values of m by analytic continuation.

Basic arithmetic shows that the derivative of the left-hand side is identically 0, so it suffices to prove the identity for any arbitrary value of m. We do it by showing that the left-hand side tends to $\frac{\pi}{2}$ as $x \to 0^+$.

theorem *elliptic_KE_legendre_real:*
assumes *"x ∈ {0<..<1}"*
defines *"K ≡ (elliptic_K :: real ⇒ _)"* **and** *"E ≡ (elliptic_E :: real ⇒ _)"*
shows *"K x * E (1 - x) + E x * K (1 - x) - K x * K (1 - x) = pi / 2"*
⟨*proof*⟩

corollary *elliptic_KE_legendre_complex:*
assumes *"Im z ≠ 0 ∨ Re z ∈ {0<..<1}"*
defines *"K ≡ (elliptic_K :: complex ⇒ _)"* **and** *"E ≡ (elliptic_E :: complex ⇒ _)"*
shows *"K z * E (1 - z) + E z * K (1 - z) - K z * K (1 - z) = of_real pi / 2"*
⟨*proof*⟩

corollary *elliptic_KE_legendre_lemniscatic_real:*
defines *"K ≡ (elliptic_K :: real ⇒ _)"* **and** *"E ≡ (elliptic_E :: real ⇒ _)"*
shows *"K (1 / 2) * (2 * E (1 / 2) - K (1 / 2)) = pi / 2"*
⟨*proof*⟩

end

5.3 Relation to complete elliptic integrals

theory *Arithmetic_Geometric_Mean_Integral*
imports *Arithmetic_Geometric_Mean Complete_Elliptic_Integrals*

begin

In this section, we make the connection between the AGM and the complete elliptic integrals.

5.3.1 Complementary moduli

We first define the complementary modulus.

definition *ell_compl :: "'a :: {real_normed_field, ln} ⇒ 'a"* **where**
"ell_compl x = (1 - x ^ 2) powr (1/2)"

lemma *ell_compl_0 [simp]: "ell_compl 0 = 1"*
⟨*proof*⟩

lemma *ell_compl_1 [simp]: "ell_compl 1 = 0"*
⟨*proof*⟩

lemma *ell_compl_sqrt2_half [simp]: "ell_compl (sqrt 2 / 2) = sqrt 2 / 2"*
⟨*proof*⟩

lemma *ell_compl_one_over_sqrt2 [simp]: "ell_compl (1 / sqrt 2) = 1 / sqrt 2"*
⟨*proof*⟩

lemma *ell_compl_real_def: "x ∈ {0..1} ⟹ ell_compl x = sqrt (1 - x ^ 2)"*
⟨*proof*⟩

lemma *ell_compl_complex_def: "ell_compl z = csqrt (1 - z ^ 2)"*
⟨*proof*⟩

lemma *ell_compl_squared_real: "(x :: real) ∈ {0..1} ⟹ ell_compl x ^ 2 = 1 - x ^ 2"*
⟨*proof*⟩

lemma *ell_compl_nonneg: "x ≤ (1 :: real) ⟹ ell_compl x ≥ 0"*
⟨*proof*⟩

lemma *ell_compl_pos: "x ∈ {0..<1::real} ⟹ ell_compl x > 0"*
⟨*proof*⟩

lemma *ell_compl_le_1: "x ∈ {0..1::real} ⟹ ell_compl x ≤ 1"*
⟨*proof*⟩

lemma *ell_compl_less_1: "x ∈ {0<..1::real} ⟹ ell_compl x < 1"*
⟨*proof*⟩

lemma *ell_compl_ell_compl [simp]: "x ∈ {0..1::real} ⟹ ell_compl (ell_compl x) = x"*
⟨*proof*⟩

5.3.2 The AGM integrals

Next, we derive a simple auxiliary integral we will need later, namely those of the form $\int_0^\infty \frac{1}{a^2+x^2}\,\mathrm{d}x$. For $a = 1$, the indefinite integral becomes the arctangent function; hence the name.

lemma *arctan_type_integral_0_infinity:*
fixes *a :: real*
assumes *a: "a > 0"*
shows *"(λx::real. 1 / (a² + x²)) absolutely_integrable_on {0..}"*
*"((λx. 1 / (a² + x²)) has_integral (pi / (2 * a))) {0..}"*
⟨*proof*⟩

lemma *arctan_type_integral_UNIV:*
fixes *a :: real*
assumes *a: "a > 0"*
shows *"(λx::real. 1 / (a² + x²)) absolutely_integrable_on UNIV"*
"((λx. 1 / (a² + x²)) has_integral (pi / a)) UNIV"
⟨*proof*⟩

Next, we look at the integrals

$$\begin{aligned} I(a,b) &= \int_0^{\frac{\pi}{2}} (a^2\cos u^2 + b^2 \sin u^2)^{-\frac{1}{2}}\,\mathrm{d}u \\ &= \int_0^{\infty} ((x^2+a^2)(x^2+b^2))^{-\frac{1}{2}}\,\mathrm{d}u \\ J(a,b) &= \int_0^{\frac{\pi}{2}} (a^2\cos u^2 + b^2 \sin u^2)^{\frac{1}{2}}\,\mathrm{d}u \end{aligned}$$

It is easy to see that $I(a,b) = K(1-(b/a)^2)/a$ and $J(a,b) = aE(1-(b/a)^2)$. Also, we have $I(x,x) = \frac{\pi}{2x}$ and $J(x,x) = \frac{x\pi}{2}$.

lemma *agm_abs_integrable:*
assumes *"a > 0" "b > 0"* **and** *[measurable]: "A ∈ sets borel"*
shows *"(λu. 1 / sqrt ((u ^ 2 + a ^ 2) * (u ^ 2 + b ^ 2))) absolutely_integrable_on A"*
⟨*proof*⟩

lemma *agm_integrable:*
assumes *"a > 0" "b > 0" "A ∈ sets borel"*
shows *"(λu. 1 / sqrt ((u ^ 2 + a ^ 2) * (u ^ 2 + b ^ 2))) integrable_on A"*
⟨*proof*⟩

definition *agm_integral :: "real ⇒ real ⇒ real"* **where**
"agm_integral a b = elliptic_K (1 - (b / a) ^ 2) / a"

definition *agm_integral' :: "real ⇒ real ⇒ real"* **where**
*"agm_integral' a b = a * elliptic_E (1 - (b / a) ^ 2)"*

context
fixes *I J :: "real ⇒ real ⇒ real"*
defines *"I ≡ agm_integral"*
defines *"J ≡ agm_integral'"*
begin

lemma *agm_integral_same_real: "I x x = pi / (2 * x)"*
⟨*proof*⟩

lemma *agm_integral'_same_real: "J x x = x * pi / 2"*
⟨*proof*⟩

lemma *has_integral_agm_integral1:*
assumes *ab: "a > 0" "b > 0"*
shows *"(λu. 1 / sqrt (a ^ 2 * cos u ^ 2 + b ^ 2 * sin u ^ 2)) absolutely_integrable_on {0..pi/2}"*
and *"((λu. 1 / sqrt (a ^ 2 * cos u ^ 2 + b ^ 2 * sin u ^ 2)) has_integral I a b) {0..pi/2}"*
⟨*proof*⟩

lemma *has_integral_agm_integral2:*
assumes *ab: "a > 0" "b > 0"*
shows *"(λx. 1 / (sqrt ((x^2 + a^2) * (x^2 + b^2)))) absolutely_integrable_on {0<..}"*
and *"((λx. 1 / (sqrt ((x^2 + a^2) * (x^2 + b^2)))) has_integral I a b) {0<..}"*
⟨*proof*⟩

lemma *agm_integral_commute_real: "a > 0 ⟹ b > 0 ⟹ I a b = I b a"*
⟨*proof*⟩

lemma *has_integral_agm_integral':*
assumes *ab: "a > 0" "b > 0"*
shows *"(λu. sqrt (a ^ 2 * cos u ^ 2 + b ^ 2 * sin u ^ 2)) absolutely_integrable_on {0..pi/2}"*
and *"((λu. sqrt (a ^ 2 * cos u ^ 2 + b ^ 2 * sin u ^ 2)) has_integral J a b) {0..pi/2}"*
⟨*proof*⟩

The key property of $I(a, b)$ is that it is invariant under a single step of the

AGM iteration, and therefore

$$I(a,b) = I(\mathrm{agm}(a,b), \mathrm{agm}(a,b)) = \frac{\pi}{2\mathrm{agm}(a,b)} \ .$$

For simplicity, we show the real case first and then use analytic continuation.

proposition *agm_integral_preserve_real:*
assumes *ab: "a > 0" "b > 0"*
shows *"I (amean a b) (gmean a b) = I a b"*
⟨*proof*⟩

lemma *agm_integral'_commute:*
assumes *"a > 0" "b > 0"*
shows *"J a b = J b a"*
⟨*proof*⟩

lemma *agm_integral_preserve_real':*
assumes *ab: "a > 0" "b > 0"*
shows *"I (fst (agm_seq a b n)) (snd (agm_seq a b n)) = I a b"*
⟨*proof*⟩

lemma *agm_integral_conv_agm_real:*
assumes *"a > 0" "b > 0"*
shows *"I a b = pi / (2 * agm a b)"*
⟨*proof*⟩

Finally, we get the main result of this section: The identity that expresses $\mathrm{agm}(a,b)$ in terms of K.

theorem *agm_conv_elliptic_K_real:*
assumes *"a > 0" "b > 0"*
shows *"agm a b = pi * a / (2 * elliptic_K ((a ^ 2 - b ^ 2) / a ^ 2))"*
⟨*proof*⟩

corollary *agm_conv_elliptic_K'_real:*
assumes *"a > 0" "b > 0"*
shows *"agm a b = pi * (a + b) / (4 * elliptic_K (((a - b) / (a + b))*
^ 2))"
⟨*proof*⟩

corollary *elliptic_K_conv_agm_real:*
assumes *"m < (1 :: real)"*
shows *"elliptic_K m = pi / (2 * agm 1 (sqrt (1 - m)))"*
⟨*proof*⟩

corollary *elliptic_K_conv_agm_complex:*
assumes *"Im m ≠ 0 ∨ Re m < 1"*
shows *"elliptic_K m = of_real pi / (2 * agm 1 (csqrt (1 - m)))"*
⟨*proof*⟩

corollary *elliptic_K_complex_nonzero:*
assumes *"Im m ≠ 0 ∨ Re m < 1"*
shows *"elliptic_K m ≠ 0"*
⟨*proof*⟩

theorem *agm_conv_elliptic_K_complex:*
assumes *"Re (z / w) > 0"*
shows *"agm w z = of_real pi * w / (2 * elliptic_K ((w ^ 2 - z ^ 2) / w ^ 2))"*
⟨*proof*⟩

corollary *agm_conv_elliptic_K'_complex:*
assumes *"Re (z / w) > 0"*
shows *"agm w z = of_real pi * (w + z) / (4 * elliptic_K (((w - z) / (w + z)) ^ 2))"*
⟨*proof*⟩

5.3.3 Upward and downward identities for complete elliptic integrals

One straightfoward consequence of the above relationship between the AGM and K is the following identity, which allows

corollary *elliptic_K_downward'_real:*
assumes *"m < (1 :: real)"*
defines *"m' ≡ sqrt (1 - m)"*
shows *"elliptic_K m = 2 / (1 + m') * elliptic_K (((1 - m') / (1 + m')) ^ 2)"*
⟨*proof*⟩

The corresponding identity in the complex plane follows by analytic continuation:

corollary *elliptic_K_downward'_complex:*
assumes *"Im m ≠ 0 ∨ Re m < 1"*
defines *"m' ≡ csqrt (1 - m)"*
shows *"elliptic_K m = 2 / (1 + m') * elliptic_K (((1 - m') / (1 + m')) ^ 2)"*
⟨*proof*⟩

corollary *elliptic_K_downward_real:*
assumes *"k ∈ {0<..<1::real}"*
defines *"k' ≡ ell_compl k"*
shows *"elliptic_K (k ^ 2) =*
*2 / (1 + k') * elliptic_K (((1 - k') / (1 + k')) ^ 2)"*
⟨*proof*⟩

corollary *elliptic_K_downward_complex:*
assumes *"Im k ≠ 0 ∨ Re k ∈ {0<..<1}"*
defines *"k' ≡ ell_compl k"*

shows *"elliptic_K (k ^ 2) = 2 / (1 + k') * elliptic_K (((1 - k') / (1 + k')) ^ 2)"*
⟨*proof*⟩

corollary *elliptic_K_upward_real:*
assumes *k: "(k::real) ∈ {0<..<1}"*
shows *"elliptic_K (k ^ 2) = elliptic_K (4 * k / (1 + k)^2) / (1 + k)"*
⟨*proof*⟩

lemma *elliptic_E_upward_real:*
fixes *x :: real*
assumes *x: "x ∈ {0<..<1}"*
defines *"g ≡ (λx. 4 * x / (1 + x) ^ 2)"*
shows *"elliptic_E (x^2) = elliptic_E (g x) * ((1 + x) / 2) + elliptic_K (g x) * ((1 - x) / 2)"*
⟨*proof*⟩

lemma *elliptic_E_downward_real:*
fixes *k :: real*
assumes *k: "k ∈ {0<..<1}"*
defines *"k' ≡ ell_compl k"*
defines *"g ≡ (λx. 4 * x / (1 + x) ^ 2)"*
shows *"elliptic_E (k ^ 2) =*
*(1 + k') * elliptic_E (((1 - k') / (1 + k')) ^ 2) - k' * elliptic_K (k ^ 2)"*
⟨*proof*⟩

lemma *agm_integral'_amean_gmean:*
assumes *"0 < a" "0 < b"*
shows *"2 * J (amean a b) (gmean a b) = J a b + a * b * I a b"*
⟨*proof*⟩

5.3.4 Relating E to K and the AGM

definition *agm_integral_aux :: "real ⇒ real ⇒ real"* **where**
*"agm_integral_aux x y = 2 * (x^2 - J x y / I x y)"*

lemma *agm_integral_aux_altdef:*
*"agm_integral_aux x y = 2 * x^2 * (1 - elliptic_E (1 - (y / x) ^ 2) / elliptic_K (1 - (y / x) ^ 2))"*
⟨*proof*⟩

definition *agm_integral_aux_psum :: "real ⇒ real ⇒ nat ⇒ real"* **where**
*"agm_integral_aux_psum x y n = (∑ i≤n. 2 ^ i * agm_diff x y i ^ 2)"*

lemma *incseq_agm_integral_aux_psum:*
"incseq (agm_integral_aux_psum x y)"
⟨*proof*⟩

We can relate the two integrals $I(a,b)$ and $J(a,b)$ by defining $f(a,b) = 2(a^2 - J(a,b)/I(a,b))$. Then if we write a_n and b_n for the AGM sequence starting with $a_0 = a$ and $b_0 = b$ and $c_n = \sqrt{a_n^2 + b_n^2}$ we have

$$f(a,b) = \sum_{n=0}^{\infty} 2^n c_n^2 \ .$$

Furthermore, the error made by truncating this sum after N terms is between 0 and $2^N c_N^2$. Since eventually $c_n < 1$, this means that the sum converges quadratically (i.e. eventually the number of correct digits at least doubles with each additional term).

Note also that the c_n are a free sideproduct of running the AGM iteration. This means that the AGM can be used to compute $I(x,y)$ and $J(x,y)$ – or, equivalently, $K(m)$ and $E(m)$ – simultaneously.

theorem *agm_diff_sums_agm_integral:*
  fixes *x y :: real* **and** *N :: nat*
  assumes *xy: "0 < y" "y ≤ x"*
  defines *"a ≡ (λn. fst (agm_seq x y n))"*
  defines *"b ≡ (λn. snd (agm_seq x y n))"*
  defines *"c ≡ (λn. agm_diff x y n)"*
  defines *"err ≡ agm_integral_aux x y - agm_integral_aux_psum x y N"*
  shows *"(λn. 2 ^ n * c n ^ 2) sums agm_integral_aux x y"*
    and *"err ∈ {0..2 ^ N * c N ^ 2}"*
⟨*proof*⟩

lemma *summable_agm_diff:*
  assumes *"0 < y" "y ≤ (x :: real)"*
  shows *"summable (λn. 2 ^ n * agm_diff x y n ^ 2)"*
  ⟨*proof*⟩

lemma *agm_integral'_conv_agm_integral:*
  fixes *x y :: real*
  assumes *xy: "0 < y" "y ≤ x"*
  defines *"c ≡ (λn. agm_diff x y n)"*
  shows *"J x y = I x y * (x² - agm_integral_aux x y / 2)"*
⟨*proof*⟩

lemma *elliptic_E_conv_elliptic_K:*
  fixes *k :: real*
  assumes *k: "k ∈ {0<..<1}"*
  defines *"x ≡ sqrt (1 - k)"*
  shows *"elliptic_E k = elliptic_K k * (1 - agm_integral_aux 1 x / 2)"*
⟨*proof*⟩

lemma *pi_conv_elliptic_K:*
  defines *"c ≡ agm_diff 1 (1 / sqrt 2)"*
  shows *"pi = 2 * elliptic_K (1 / 2) ^ 2 * (1 - agm_integral_aux 1 (1 / sqrt 2))"*

⟨*proof*⟩

We now also easily obtain a way to express π using the AGM:

theorem *pi_conv_agm:*
```
  "pi = 2 * agm 1 (1 / sqrt 2) ^ 2 / (1 - agm_integral_aux 1 (1 / sqrt
2))"
```
⟨*proof*⟩

end

end

6 Application: Computing π via the AGM

theory *Pi_Approx_AGM*
  imports *Arithmetic_Geometric_Mean_Integral*
begin

As we just saw, π can be expressed in terms of the AGM and the quantity *agm_integral_aux*, which is an infinite sum over the numbers c_n that is produced by the AGM interation.

We will now analyse the error made by truncating the AGM iterations at some point and show that it decreases exponentially, i.e. every additional AGM iteration doubles the number of correct bits.

Note that this is still an abstract mathematical statement that assumes that the AGM iterations are performed with infinite precision. A more detailed analysis for doing the iteration with finite-precision floating or fixed point numbers would be interesting but much more complicated.

definition *pi_agm_err1 :: "nat ⇒ real"* **where**
```
  "pi_agm_err1 n = fst (agm_seq 1 (1 / sqrt 2) n) - agm 1 (1 / sqrt 2)"
```

definition *pi_agm_err2 :: "nat ⇒ real"* **where**
```
  "pi_agm_err2 n = agm_integral_aux 1 (1 / sqrt 2) - agm_integral_aux_psum
1 (1 / sqrt 2) n"
```

definition *pi_agm :: "nat ⇒ real"* **where**
```
  "pi_agm n = 2 * fst (agm_seq 1 (1 / sqrt 2) (Suc n)) ^ 2 /
               (1 - agm_integral_aux_psum 1 (1 / sqrt 2) n)"
```

lemma *power_int_eq_powerI: "int n = m ⟹ x powi m = x ^ n"*
  ⟨*proof*⟩

lemma *pi_agm_err1_bound_strong: "pi_agm_err1 n ∈ {0..8 / 18 ^ (2 ^ n)}"*
  and *pi_agm_err1_bound:* *"pi_agm_err1 n ∈ {0..2 powi (3 - 2 ^
(n + 2))}"*
  and *pi_agm_err2_bound:* *"n > 0 ⟹ pi_agm_err2 n ∈ {0..2 powi
(int n + 5 - 2 ^ (n + 2))}"*

⟨*proof*⟩

lemma *abs_diff_le_max_real:*
  assumes *"a ≥ 0" "b ≥ (0::real)"*
  shows *"|a - b| ≤ max a b"*
  ⟨*proof*⟩

With some tedious arithmetic, we obtain the following concrete bound for the AGM approximation of π. The number of correct bits after the decimal point is at least $2^{n+2} - n - 10$, i.e. roughly 2^{n+2} for large n.

theorem *dist_pi_agm_bound: "dist pi (pi_agm n) ≤ 2 powi (int n + 10 - 2^(n+2))"*
⟨*proof*⟩

end

7 Relating the complete elliptic integral to the Jacobi theta functions

theory *AGM_Theta*
  imports *"Theta_Functions.Theta_Nullwert" Arithmetic_Geometric_Mean_Integral*
begin

The Jacobi theta nullwert functions have the property that $(\vartheta_3(q)^2, \vartheta_4(q)^2)$ is transformed by a single step of the AGM iteration into $(\vartheta_3(q^2)^2, \vartheta_4(q^2)^2)$. Clearly, for $n \to \infty$, this converges to $(\vartheta_3(0)^2, \vartheta_4(0)^2) = (1, 1)$.

lemma *agm_seq_jacobi_theta_00_01_square_real:*
  fixes *q :: real*
  assumes *"q ∈ {-1<..<1}"*
  shows *"agm_seq (ϑ₃ q ^ 2) (ϑ₄ q ^ 2) n = (ϑ₃ (q ^ (2 ^ n)) ^ 2, ϑ₄ (q ^ (2 ^ n)) ^ 2)"*
  ⟨*proof*⟩

lemma *agm_jacobi_theta_00_01_square_real:*
  fixes *q :: real*
  assumes *"q ∈ {-1<..<1}"*
  shows *"agm (ϑ₃ q ^ 2) (ϑ₄ q ^ 2) = 1"*
⟨*proof*⟩

By recasting the above in terms of the complete elliptic integral k, we get the following identity that relates K to the Jacobi theta functions.

We only show the identity for real q with $0 \leq q < 1$. The version for the complex z-plane is a bit more intricate: there the identity fails to hold at any point within a disc of radius $\frac{1}{2}$ around any point of the form $\frac{1}{2} + \mathbb{Z}$. This is due to the branch cut of K.

theorem *elliptic_K_jacobi_theta_real:*

fixes `q :: real`
assumes `q: "q ∈ {0..<1}"`
shows `"elliptic_K (ϑ₂ q ^ 4 / ϑ₃ q ^ 4) = pi / 2 * ϑ₃ q ^ 2"`
⟨*proof*⟩

end

References

[1] J. M. Borwein and P. B. Borwein. *Pi and the AGM: a study in the analytic number theory and computational complexity.* Wiley-Interscience, USA, 1987.

[2] R. P. Brent. The Borwein brothers, pi and the AGM. In D. H. Bailey, N. S. Borwein, R. P. Brent, R. S. Burachik, J.-a. H. Osborn, B. Sims, and Q. J. Zhu, editors, *From Analysis to Visualization*, pages 323–347, Cham, 2020. Springer International Publishing.