Hyperdual Numbers and Forward Differentiation

Filip Smola, Jacques Fleuriot

January 5, 2022

Abstract

Hyperdual numbers are ones with a real component and a number of infinitesimal components, usually written as $a_0 + a_1 \cdot \epsilon_1 + a_2 \cdot \epsilon_2 + a_3 \cdot \epsilon_1\epsilon_2$. They have been proposed by Fike and Alonso [1] in an approach to automatic differentiation.

In this entry we formalise hyperdual numbers and their application to forward differentiation. We show them to be an instance of multiple algebraic structures and then, along with facts about twice-differentiability, we define what we call the hyperdual extensions of functions on real-normed fields. This extension formally represents the proposed way that the first and second derivatives of a function can be automatically calculated. We demonstrate it on the standard logistic function $f(x) = \frac{1}{1+e^{-x}}$ and also reproduce the example analytic function $f(x) = \frac{e^x}{\sqrt{sin(x)^3+cos(x)^3}}$ used for demonstration by Fike and Alonso.

Contents

theory *Hyperdual*
  imports *HOL−Analysis.Analysis*
begin

1 Hyperdual Numbers

Let τ be some type. Second-order hyperdual numbers over τ take the form $a_1 + a_2\varepsilon_1 + a_3\varepsilon_2 + a_4\varepsilon_1\varepsilon_2$ where all $a_i :: \tau$, and ε_1 and ε_2 are non-zero but nilpotent infinitesimals: ${\varepsilon_1}^2 = {\varepsilon_2}^2 = (\varepsilon_1\varepsilon_2)^2 = \mathit{0}$.

We define second-order hyperdual numbers as a coinductive data type with four components: the base component, two first-order hyperdual components and one second-order hyperdual component.

codatatype *'a hyperdual* = *Hyperdual* (*Base*: *'a*) (*Eps1*: *'a*) (*Eps2*: *'a*) (*Eps12*: *'a*)

Two hyperduals are equal iff all their components are equal.

lemma *hyperdual-eq-iff* [*iff*]:
  $x = y \longleftrightarrow ((\mathit{Base}\ x = \mathit{Base}\ y) \wedge (\mathit{Eps1}\ x = \mathit{Eps1}\ y) \wedge (\mathit{Eps2}\ x = \mathit{Eps2}\ y) \wedge (\mathit{Eps12}\ x = \mathit{Eps12}\ y))$
  ⟨*proof*⟩

lemma *hyperdual-eqI*:
  assumes *Base x* = *Base y*
    and *Eps1 x* = *Eps1 y*
    and *Eps2 x* = *Eps2 y*
    and *Eps12 x* = *Eps12 y*
  shows *x* = *y*
  ⟨*proof*⟩

The embedding from the component type to hyperduals requires the component type to have a zero element.

definition *of-comp* :: (*'a* :: *zero*) ⇒ *'a hyperdual*
  where *of-comp a* = *Hyperdual a 0 0 0*

lemma *of-comp-simps* [*simp*]:
  Base (*of-comp a*) = *a*
  Eps1 (*of-comp a*) = *0*
  Eps2 (*of-comp a*) = *0*
  Eps12 (*of-comp a*) = *0*
  ⟨*proof*⟩

1.1 Addition and Subtraction

We define hyperdual addition, subtraction and unary minus pointwise, and zero by embedding.

instantiation *hyperdual* :: (*plus*) *plus*
begin

primcorec *plus-hyperdual*
  where
    Base (*x* + *y*) = *Base x* + *Base y*
  | *Eps1* (*x* + *y*) = *Eps1 x* + *Eps1 y*

| *Eps2* $(x + y)$ = *Eps2* x + *Eps2* y
| *Eps12* $(x + y)$ = *Eps12* x + *Eps12* y

instance ⟨*proof*⟩
end

instantiation *hyperdual* :: (*zero*) *zero*
begin

definition *zero-hyperdual*
where *0* = *of-comp 0*

instance ⟨*proof*⟩
end

lemma *zero-hyperdual-simps* [*simp*]:
Base 0 = *0*
Eps1 0 = *0*
Eps2 0 = *0*
Eps12 0 = *0*
Hyperdual 0 0 0 0 = *0*
⟨*proof*⟩

instantiation *hyperdual* :: (*uminus*) *uminus*
begin

primcorec *uminus-hyperdual*
where
Base $(-x)$ = $-$ *Base* x
| *Eps1* $(-x)$ = $-$ *Eps1* x
| *Eps2* $(-x)$ = $-$ *Eps2* x
| *Eps12* $(-x)$ = $-$ *Eps12* x

instance ⟨*proof*⟩
end

instantiation *hyperdual* :: (*minus*) *minus*
begin

primcorec *minus-hyperdual*
where
Base $(x - y)$ = *Base* x $-$ *Base* y
| *Eps1* $(x - y)$ = *Eps1* x $-$ *Eps1* y
| *Eps2* $(x - y)$ = *Eps2* x $-$ *Eps2* y
| *Eps12* $(x - y)$ = *Eps12* x $-$ *Eps12* y

instance ⟨*proof*⟩
end

If the components form a commutative group under addition then so do the hyperduals.

instance *hyperdual* :: (*semigroup-add*) *semigroup-add*
⟨*proof*⟩

instance *hyperdual* :: (*monoid-add*) *monoid-add*
⟨*proof*⟩

instance *hyperdual* :: (*ab-semigroup-add*) *ab-semigroup-add*
⟨*proof*⟩

instance *hyperdual* :: (*comm-monoid-add*) *comm-monoid-add*
⟨*proof*⟩

instance *hyperdual* :: (*group-add*) *group-add*
⟨*proof*⟩

instance *hyperdual* :: (*ab-group-add*) *ab-group-add*
⟨*proof*⟩

lemma *of-comp-add*:
fixes *a b* :: *'a* :: *monoid-add*
shows *of-comp* (*a* + *b*) = *of-comp a* + *of-comp b*
⟨*proof*⟩

lemma
fixes *a b* :: *'a* :: *group-add*
shows *of-comp-minus*: *of-comp* (− *a*) = − *of-comp a*
and *of-comp-diff*: *of-comp* (*a* − *b*) = *of-comp a* − *of-comp b*
⟨*proof*⟩

1.2 Multiplication and Scaling

Multiplication of hyperduals is defined by distributing the expressions and using the nilpotence of ε_1 and ε_2, resulting in the definition used here. The hyperdual one is again defined by embedding.

instantiation *hyperdual* :: ({*one*, *zero*}) *one*
begin

definition *one-hyperdual*
where *1* = *of-comp 1*

instance ⟨*proof*⟩
end

lemma *one-hyperdual-simps* [*simp*]:
Base 1 = *1*
Eps1 1 = *0*
Eps2 1 = *0*
Eps12 1 = *0*
Hyperdual 1 0 0 0 = *1*
⟨*proof*⟩

instantiation *hyperdual* :: ({*times*, *plus*}) *times*
begin

primcorec *times-hyperdual*
where
Base (*x* ∗ *y*) = *Base x* ∗ *Base y*
| *Eps1* (*x* ∗ *y*) = (*Base x* ∗ *Eps1 y*) + (*Eps1 x* ∗ *Base y*)
| *Eps2* (*x* ∗ *y*) = (*Base x* ∗ *Eps2 y*) + (*Eps2 x* ∗ *Base y*)
| *Eps12* (*x* ∗ *y*) = (*Base x* ∗ *Eps12 y*) + (*Eps1 x* ∗ *Eps2 y*) + (*Eps2 x* ∗ *Eps1 y*) + (*Eps12 x* ∗ *Base y*)

instance ⟨*proof*⟩
end

If the components form a ring then so do the hyperduals.

instance *hyperdual* :: (*semiring*) *semiring*
⟨*proof*⟩

instance *hyperdual* :: ({*monoid-add*, *mult-zero*}) *mult-zero*
⟨*proof*⟩

instance *hyperdual* :: (*ring*) *ring*
⟨*proof*⟩

instance *hyperdual* :: (*comm-ring*) *comm-ring*
⟨*proof*⟩

instance *hyperdual* :: (*ring-1*) *ring-1*
⟨*proof*⟩

instance *hyperdual* :: (*comm-ring-1*) *comm-ring-1*
⟨*proof*⟩

lemma *of-comp-times*:
fixes *a b* :: *'a* :: *semiring-0*
shows *of-comp* (*a* ∗ *b*) = *of-comp a* ∗ *of-comp b*
⟨*proof*⟩

Hyperdual scaling is multiplying each component by a factor from the component type.

primcorec *scaleH* :: ($'a$ :: *times*) $\Rightarrow$ $'a$ *hyperdual* $\Rightarrow$ $'a$ *hyperdual* (**infixr** $*_H$ *75*)
where
Base ($f *_H x$) $= f *$ *Base* x
| *Eps1* ($f *_H x$) $= f *$ *Eps1* x
| *Eps2* ($f *_H x$) $= f *$ *Eps2* x
| *Eps12* ($f *_H x$) $= f *$ *Eps12* x

lemma *scaleH-times*:
fixes f :: $'a$ :: {*monoid-add*, *mult-zero*}
shows $f *_H x =$ *of-comp* $f * x$
⟨*proof*⟩

lemma *scaleH-add*:
fixes a :: $'a$ :: *semiring*
shows $(a + a') *_H b = a *_H b + a' *_H b$
and $a *_H (b + b') = a *_H b + a *_H b'$
⟨*proof*⟩

lemma *scaleH-diff*:
fixes a :: $'a$ :: *ring*
shows $(a - a') *_H b = a *_H b - a' *_H b$
and $a *_H (b - b') = a *_H b - a *_H b'$
⟨*proof*⟩

lemma *scaleH-mult*:
fixes a :: $'a$ :: *semigroup-mult*
shows $(a * a') *_H b = a *_H a' *_H b$
⟨*proof*⟩

lemma *scaleH-one* [*simp*]:
fixes b :: ($'a$ :: *monoid-mult*) *hyperdual*
shows $1 *_H b = b$
⟨*proof*⟩

lemma *scaleH-zero* [*simp*]:
fixes b :: ($'a$ :: {*mult-zero*, *times*}) *hyperdual*
shows $0 *_H b = 0$
⟨*proof*⟩

lemma
fixes b :: ($'a$ :: *ring-1*) *hyperdual*
shows *scaleH-minus* [*simp*]:$- 1 *_H b = - b$
and *scaleH-minus-left*: $- (a *_H b) = - a *_H b$
and *scaleH-minus-right*: $- (a *_H b) = a *_H - b$
⟨*proof*⟩

Induction rule for natural numbers that takes 0 and 1 as base cases.

lemma *nat-induct01Suc*[*case-names 0 1 Suc*]:
assumes *P 0*

and $P\ 1$
and $\bigwedge n.\ n > 0 \Longrightarrow P\ n \Longrightarrow P\ (Suc\ n)$
shows $P\ n$
$\langle proof \rangle$

lemma *hyperdual-power*:
fixes $x :: ('a :: comm\text{-}ring\text{-}1)\ hyperdual$
shows $x \mathbin{\hat{}} n = Hyperdual\ ((Base\ x) \mathbin{\hat{}} n)$
$(Eps1\ x * of\text{-}nat\ n * (Base\ x) \mathbin{\hat{}} (n - 1))$
$(Eps2\ x * of\text{-}nat\ n * (Base\ x) \mathbin{\hat{}} (n - 1))$
$(Eps12\ x * of\text{-}nat\ n * (Base\ x) \mathbin{\hat{}} (n - 1) + Eps1\ x * Eps2\ x * of\text{-}nat\ n * of\text{-}nat\ (n - 1) * (Base\ x) \mathbin{\hat{}} (n - 2))$
$\langle proof \rangle$

lemma *hyperdual-power-simps* [*simp*]:
shows $Base\ ((x :: 'a :: comm\text{-}ring\text{-}1\ hyperdual) \mathbin{\hat{}} n) = Base\ x \mathbin{\hat{}} n$
and $Eps1\ ((x :: 'a :: comm\text{-}ring\text{-}1\ hyperdual) \mathbin{\hat{}} n) = Eps1\ x * of\text{-}nat\ n * (Base\ x) \mathbin{\hat{}} (n - 1)$
and $Eps2\ ((x :: 'a :: comm\text{-}ring\text{-}1\ hyperdual) \mathbin{\hat{}} n) = Eps2\ x * of\text{-}nat\ n * (Base\ x) \mathbin{\hat{}} (n - 1)$
and $Eps12\ ((x :: 'a :: comm\text{-}ring\text{-}1\ hyperdual) \mathbin{\hat{}} n) =$
$(Eps12\ x * of\text{-}nat\ n * (Base\ x) \mathbin{\hat{}} (n - 1) + Eps1\ x * Eps2\ x * of\text{-}nat\ n * of\text{-}nat\ (n - 1) * (Base\ x) \mathbin{\hat{}} (n - 2))$
$\langle proof \rangle$

Squaring the hyperdual one behaves as expected from the reals.

lemma *hyperdual-square-eq-1-iff* [*iff*]:
fixes $x :: ('a :: \{real\text{-}div\text{-}algebra,\ comm\text{-}ring\})\ hyperdual$
shows $x * x = 1 \longleftrightarrow x = 1 \vee x = -\ 1$
$\langle proof \rangle$

1.2.1 Properties of Zero Divisors

Unlike the reals, hyperdual numbers may have non-trivial divisors of zero as we show below.

First, if the components have no non-trivial zero divisors then that behaviour is preserved on the base component.

lemma *divisors-base-zero*:
fixes $a\ b :: ('a :: ring\text{-}no\text{-}zero\text{-}divisors)\ hyperdual$
assumes $Base\ (a * b) = 0$
shows $Base\ a = 0 \vee Base\ b = 0$
$\langle proof \rangle$
lemma *hyp-base-mult-eq-0-iff* [*iff*]:
fixes $a\ b :: ('a :: ring\text{-}no\text{-}zero\text{-}divisors)\ hyperdual$
shows $Base\ (a * b) = 0 \longleftrightarrow Base\ a = 0 \vee Base\ b = 0$
$\langle proof \rangle$

However, the conditions are relaxed on the full hyperdual numbers. This is due to some terms vanishing in the multiplication and thus not constraining the result.

lemma *divisors-hyperdual-zero* [*iff*]:
  fixes $a\ b :: ('a :: \mathit{ring\text{-}no\text{-}zero\text{-}divisors})\ \mathit{hyperdual}$
  shows $a * b = 0 \longleftrightarrow (a = 0 \lor b = 0 \lor (\mathit{Base}\ a = 0 \land \mathit{Base}\ b = 0 \land \mathit{Eps1}\ a * \mathit{Eps2}\ b = -\ \mathit{Eps2}\ a * \mathit{Eps1}\ b))$
⟨*proof*⟩

1.2.2 Multiplication Cancellation

Similarly to zero divisors, multiplication cancellation rules for hyperduals are not exactly the same as those for reals.

First, cancelling a common factor has a relaxed condition compared to reals. It only requires the common factor to have base component zero, instead of requiring the whole number to be zero.

lemma *hyp-mult-left-cancel* [*iff*]:
  fixes $a\ b\ c :: ('a :: \mathit{ring\text{-}no\text{-}zero\text{-}divisors})\ \mathit{hyperdual}$
  assumes *baseC*: $\mathit{Base}\ c \neq 0$
  shows $c * a = c * b \longleftrightarrow a = b$
⟨*proof*⟩

lemma *hyp-mult-right-cancel* [*iff*]:
  fixes $a\ b\ c :: ('a :: \mathit{ring\text{-}no\text{-}zero\text{-}divisors})\ \mathit{hyperdual}$
  assumes *baseC*: $\mathit{Base}\ c \neq 0$
    shows $a * c = b * c \longleftrightarrow a = b$
⟨*proof*⟩

Next, when a factor absorbs another there are again relaxed conditions compared to reals. For reals, either the absorbing factor is zero or the absorbed is the unit. However, with hyperduals there are more possibilities again due to terms vanishing during the multiplication.

lemma *hyp-mult-cancel-right1* [*iff*]:
  fixes $a\ b :: ('a :: \mathit{ring\text{-}1\text{-}no\text{-}zero\text{-}divisors})\ \mathit{hyperdual}$
  shows $a = b * a \longleftrightarrow a = 0 \lor b = 1 \lor (\mathit{Base}\ a = 0 \land \mathit{Base}\ b = 1 \land \mathit{Eps1}\ b * \mathit{Eps2}\ a = -\ \mathit{Eps2}\ b * \mathit{Eps1}\ a)$
⟨*proof*⟩
lemma *hyp-mult-cancel-right2* [*iff*]:
  fixes $a\ b :: ('a :: \mathit{ring\text{-}1\text{-}no\text{-}zero\text{-}divisors})\ \mathit{hyperdual}$
  shows $b * a = a \longleftrightarrow a = 0 \lor b = 1 \lor (\mathit{Base}\ a = 0 \land \mathit{Base}\ b = 1 \land \mathit{Eps1}\ b * \mathit{Eps2}\ a = -\ \mathit{Eps2}\ b * \mathit{Eps1}\ a)$
  ⟨*proof*⟩

lemma *hyp-mult-cancel-left1* [*iff*]:
  fixes $a\ b :: ('a :: \mathit{ring\text{-}1\text{-}no\text{-}zero\text{-}divisors})\ \mathit{hyperdual}$
  shows $a = a * b \longleftrightarrow a = 0 \lor b = 1 \lor (\mathit{Base}\ a = 0 \land \mathit{Base}\ b = 1 \land \mathit{Eps1}\ a * \mathit{Eps2}\ b = -\ \mathit{Eps2}\ a * \mathit{Eps1}\ b)$

⟨*proof*⟩
lemma *hyp-mult-cancel-left2* [*iff*]:
  fixes *a b* :: (*'a* :: *ring-1-no-zero-divisors*) *hyperdual*
  shows $a * b = a \longleftrightarrow a = 0 \vee b = 1 \vee (Base\ a = 0 \wedge Base\ b = 1 \wedge Eps1\ a * Eps2\ b = -\ Eps2\ a * Eps1\ b)$
  ⟨*proof*⟩

1.3 Multiplicative Inverse and Division

If the components form a ring with a multiplicative inverse then so do the hyperduals. The hyperdual inverse of a is defined as the solution to $a * x = (1::'a)$. Hyperdual division is then multiplication by divisor's inverse.

Each component of the inverse has as denominator a power of the base component. Therefore this inverse is only well defined for hyperdual numbers with non-zero base components.

instantiation *hyperdual* :: ({*inverse*, *ring-1*}) *inverse*
begin

primcorec *inverse-hyperdual*
  where
    $Base\ (inverse\ a) = 1\ /\ Base\ a$
  $|\ Eps1\ (inverse\ a) = -\ Eps1\ a\ /\ (Base\ a)\hat{}2$
  $|\ Eps2\ (inverse\ a) = -\ Eps2\ a\ /\ (Base\ a)\hat{}2$
  $|\ Eps12\ (inverse\ a) = 2 * (Eps1\ a * Eps2\ a\ /\ (Base\ a)\hat{}3) - Eps12\ a\ /\ (Base\ a)\hat{}2$

primcorec *divide-hyperdual*
  where
    $Base\ (divide\ a\ b) = Base\ a\ /\ Base\ b$
  $|\ Eps1\ (divide\ a\ b) = (Eps1\ a * Base\ b - Base\ a * Eps1\ b)\ /\ ((Base\ b)\hat{}2)$
  $|\ Eps2\ (divide\ a\ b) = (Eps2\ a * Base\ b - Base\ a * Eps2\ b)\ /\ ((Base\ b)\hat{}2)$
  $|\ Eps12\ (divide\ a\ b) = (2 * Base\ a * Eps1\ b * Eps2\ b -$
        $Base\ a * Base\ b * Eps12\ b -$
        $Eps1\ a * Base\ b * Eps2\ b -$
        $Eps2\ a * Base\ b * Eps1\ b +$
        $Eps12\ a * ((Base\ b)\hat{}2))\ /\ ((Base\ b)\hat{}3)$
instance
  ⟨*proof*⟩
end

Because hyperduals have non-trivial zero divisors, they do not form a division ring and so we can't use the *division-ring* type class to establish properties of hyperdual division. However, if the components form a division ring as well as a commutative ring, we can prove some similar facts about hyperdual division inspired by *division-ring*.

Inverse is multiplicative inverse from both sides.

lemma

fixes *a* :: (*'a* :: {*inverse*, *comm-ring-1*, *division-ring*}) *hyperdual*
assumes *Base a* ≠ *0*
shows *hyp-left-inverse* [*simp*]: *inverse a* ∗ *a* = *1*
and *hyp-right-inverse* [*simp*]: *a* ∗ *inverse a* = *1*
⟨*proof*⟩

Division is multiplication by inverse.

lemma *hyp-divide-inverse*:
fixes *a b* :: (*'a* :: {*inverse*, *comm-ring-1*, *division-ring*}) *hyperdual*
shows *a* / *b* = *a* ∗ *inverse b*
⟨*proof*⟩

Hyperdual inverse is zero when not well defined.

lemma *zero-base-zero-inverse*:
fixes *a* :: (*'a* :: {*inverse*, *comm-ring-1*, *division-ring*}) *hyperdual*
assumes *Base a* = *0*
shows *inverse a* = *0*
⟨*proof*⟩

lemma *zero-inverse-zero-base*:
fixes *a* :: (*'a* :: {*inverse*, *comm-ring-1*, *division-ring*}) *hyperdual*
assumes *inverse a* = *0*
shows *Base a* = *0*
⟨*proof*⟩

lemma *hyp-inverse-zero*:
fixes *a* :: (*'a* :: {*inverse*, *comm-ring-1*, *division-ring*}) *hyperdual*
shows (*inverse a* = *0*) = (*Base a* = *0*)
⟨*proof*⟩

Inverse preserves invertibility.

lemma *hyp-invertible-inverse*:
fixes *a* :: (*'a* :: {*inverse*, *comm-ring-1*, *division-ring*}) *hyperdual*
shows (*Base a* = *0*) = (*Base* (*inverse a*) = *0*)
⟨*proof*⟩

Inverse is the only number that satisfies the defining equation.

lemma *hyp-inverse-unique*:
fixes *a b* :: (*'a* :: {*inverse*, *comm-ring-1*, *division-ring*}) *hyperdual*
assumes *a* ∗ *b* = *1*
shows *b* = *inverse a*
⟨*proof*⟩

Multiplicative inverse commutes with additive inverse.

lemma *hyp-minus-inverse-comm*:
fixes *a* :: (*'a* :: {*inverse*, *comm-ring-1*, *division-ring*}) *hyperdual*
shows *inverse* (− *a*) = − *inverse a*
⟨*proof*⟩

Inverse is an involution (only) where well defined. Counter-example for non-invertible is *Hyperdual* $(0::'a)$ $(0::'a)$ $(0::'a)$ $(0::'a)$ with inverse *Hyperdual* $(0::'a)$ $(0::'a)$ $(0::'a)$ $(0::'a)$ which then inverts to *Hyperdual* $(0::'a)$ $(0::'a)$ $(0::'a)$ $(0::'a)$.

lemma *hyp-inverse-involution*:
  fixes *a* :: (*'a* :: {*inverse*, *comm-ring-1*, *division-ring*}) *hyperdual*
  assumes *Base a* $\neq$ *0*
  shows *inverse* (*inverse a*) = *a*
  ⟨*proof*⟩

lemma *inverse-inverse-neq-Ex*:
  ∃ *a* :: (*'a* :: {*inverse*, *comm-ring-1*, *division-ring*}) *hyperdual* . *inverse* (*inverse a*) $\neq$ *a*
⟨*proof*⟩

Inverses of equal invertible numbers are equal. This includes the other direction by inverse preserving invertibility and being an involution.

From a different point of view, inverse is injective on invertible numbers. The other direction for is again by inverse preserving invertibility and being an involution.

lemma *hyp-inverse-injection*:
  fixes *a b* :: (*'a* :: {*inverse*, *comm-ring-1*, *division-ring*}) *hyperdual*
  assumes *Base a* $\neq$ *0*
    and *Base b* $\neq$ *0*
   shows (*inverse a* = *inverse b*) = (*a* = *b*)
  ⟨*proof*⟩

One is its own inverse.

lemma *hyp-inverse-1* [*simp*]:
  inverse (*1* :: (*'a* :: {*inverse*, *comm-ring-1*, *division-ring*}) *hyperdual*) = *1*
  ⟨*proof*⟩

Inverse distributes over multiplication (even when not well defined).

lemma *hyp-inverse-mult-distrib*:
  fixes *a b* :: (*'a* :: {*inverse*, *comm-ring-1*, *division-ring*}) *hyperdual*
  shows *inverse* (*a* ∗ *b*) = *inverse b* ∗ *inverse a*
⟨*proof*⟩

We derive expressions for addition and subtraction of inverses.

lemma *hyp-inverse-add*:
  fixes *a b* :: (*'a* :: {*inverse*, *comm-ring-1*, *division-ring*}) *hyperdual*
  assumes *Base a* $\neq$ *0*
    and *Base b* $\neq$ *0*
   shows *inverse a* + *inverse b* = *inverse a* ∗ (*a* + *b*) ∗ *inverse b*
  ⟨*proof*⟩

lemma *hyp-inverse-diff*:

fixes $a\ b$ :: ($'a$ :: {*inverse*, *comm-ring-1*, *division-ring*}) *hyperdual*
assumes a: *Base* $a \neq 0$
and b: *Base* $b \neq 0$
shows *inverse* a − *inverse* b = *inverse* $a * (b - a) *$ *inverse* b
⟨*proof*⟩

Division is one only when dividing by self.

lemma *hyp-divide-self*:
fixes $a\ b$ :: ($'a$ :: {*inverse*, *comm-ring-1*, *division-ring*}) *hyperdual*
assumes *Base* $b \neq 0$
shows $a\ /\ b = 1 \longleftrightarrow a = b$
⟨*proof*⟩

Taking inverse is the same as division of one, even when not invertible.

lemma *hyp-inverse-divide-1* [*divide-simps*]:
fixes a :: ($'a$ :: {*inverse*, *comm-ring-1*, *division-ring*}) *hyperdual*
shows *inverse* $a = 1\ /\ a$
⟨*proof*⟩

Division distributes over addition and subtraction.

lemma *hyp-add-divide-distrib*:
fixes $a\ b\ c$ :: ($'a$ :: {*inverse*, *comm-ring-1*, *division-ring*}) *hyperdual*
shows $(a + b)\ /\ c = a/c + b/c$
⟨*proof*⟩

lemma *hyp-diff-divide-distrib*:
fixes $a\ b\ c$ :: ($'a$ :: {*inverse*, *comm-ring-1*, *division-ring*}) *hyperdual*
shows $(a - b)\ /\ c = a\ /\ c - b\ /\ c$
⟨*proof*⟩

Multiplication associates with division.

lemma *hyp-times-divide-assoc* [*simp*]:
fixes $a\ b\ c$ :: ($'a$ :: {*inverse*, *comm-ring-1*, *division-ring*}) *hyperdual*
shows $a * (b\ /\ c) = (a * b)\ /\ c$
⟨*proof*⟩

Additive inverse commutes with division, because it is multiplication by inverse.

lemma *hyp-divide-minus-left* [*simp*]:
fixes $a\ b$ :: ($'a$ :: {*inverse*, *comm-ring-1*, *division-ring*}) *hyperdual*
shows $(-a)\ /\ b = -\ (a\ /\ b)$
⟨*proof*⟩

lemma *hyp-divide-minus-right* [*simp*]:
fixes $a\ b$ :: ($'a$ :: {*inverse*, *comm-ring-1*, *division-ring*}) *hyperdual*
shows $a\ /\ (-b) = -\ (a\ /\ b)$
⟨*proof*⟩

Additive inverses on both sides of division cancel out.

lemma *hyp-minus-divide-minus*:
  fixes $a\ b :: ('a :: \{inverse,\ comm\text{-}ring\text{-}1,\ division\text{-}ring\})\ hyperdual$
  shows $(-a)\ /\ (-b) = a\ /\ b$
  ⟨*proof*⟩

We can multiply both sides of equations by an invertible denominator.

lemma *hyp-denominator-eliminate* [*divide-simps*]:
  fixes $a\ b\ c :: ('a :: \{inverse,\ comm\text{-}ring\text{-}1,\ division\text{-}ring\})\ hyperdual$
  assumes $Base\ c \neq 0$
    shows $a = b\ /\ c \longleftrightarrow a * c = b$
  ⟨*proof*⟩

We can move addition and subtraction to a common denominator in the following ways:

lemma *hyp-add-divide-eq-iff*:
  fixes $x\ y\ z :: ('a :: \{inverse,\ comm\text{-}ring\text{-}1,\ division\text{-}ring\})\ hyperdual$
  assumes $Base\ z \neq 0$
  shows $x + y\ /\ z = (x * z + y)\ /\ z$
  ⟨*proof*⟩

Result of division by non-invertible number is not invertible.

lemma *hyp-divide-base-zero* [*simp*]:
  fixes $a\ b :: ('a :: \{inverse,\ comm\text{-}ring\text{-}1,\ division\text{-}ring\})\ hyperdual$
  assumes $Base\ b = 0$
    shows $Base\ (a\ /\ b) = 0$
  ⟨*proof*⟩

Division of self is 1 when invertible, 0 otherwise.

lemma *hyp-divide-self-if* [*simp*]:
  fixes $a :: ('a :: \{inverse,\ comm\text{-}ring\text{-}1,\ division\text{-}ring\})\ hyperdual$
  shows $a\ /\ a = (if\ Base\ a = 0\ then\ 0\ else\ 1)$
  ⟨*proof*⟩

Repeated division is division by product of the denominators.

lemma *hyp-denominators-merge*:
  fixes $a\ b\ c :: ('a :: \{inverse,\ comm\text{-}ring\text{-}1,\ division\text{-}ring\})\ hyperdual$
  shows $(a\ /\ b)\ /\ c = a\ /\ (c * b)$
  ⟨*proof*⟩

Finally, we derive general simplifications for division with addition and subtraction.

lemma *hyp-add-divide-eq-if-simps* [*divide-simps*]:
  fixes $a\ b\ z :: ('a :: \{inverse,\ comm\text{-}ring\text{-}1,\ division\text{-}ring\})\ hyperdual$
  shows $a + b\ /\ z = (if\ Base\ z = 0\ then\ a\ else\ (a * z + b)\ /\ z)$
    and $a\ /\ z + b = (if\ Base\ z = 0\ then\ b\ else\ (a + b * z)\ /\ z)$
    and $-\ (a\ /\ z) + b = (if\ Base\ z = 0\ then\ b\ else\ (-a + b * z)\ /\ z)$
    and $a - b\ /\ z = (if\ Base\ z = 0\ then\ a\ else\ (a * z - b)\ /\ z)$

and $a\ /\ z - b = (\mathit{if}\ \mathit{Base}\ z = 0\ \mathit{then}\ {-b}\ \mathit{else}\ (a - b * z)\ /\ z)$
and $-\ (a\ /\ z) - b = (\mathit{if}\ \mathit{Base}\ z = 0\ \mathit{then}\ {-b}\ \mathit{else}\ (-\ a - b * z)\ /\ z)$
⟨*proof*⟩

lemma *hyp-divide-eq-eq* [*divide-simps*]:
fixes $a\ b\ c$ :: ($'a$ :: {*inverse*, *comm-ring-1*, *division-ring*}) *hyperdual*
shows $b\ /\ c = a \longleftrightarrow (\mathit{if}\ \mathit{Base}\ c \neq 0\ \mathit{then}\ b = a * c\ \mathit{else}\ a = 0)$
⟨*proof*⟩

lemma *hyp-eq-divide-eq* [*divide-simps*]:
fixes $a\ b\ c$ :: ($'a$ :: {*inverse*, *comm-ring-1*, *division-ring*}) *hyperdual*
shows $a = b\ /\ c \longleftrightarrow (\mathit{if}\ \mathit{Base}\ c \neq 0\ \mathit{then}\ a * c = b\ \mathit{else}\ a = 0)$
⟨*proof*⟩

lemma *hyp-minus-divide-eq-eq* [*divide-simps*]:
fixes $a\ b\ c$ :: ($'a$ :: {*inverse*, *comm-ring-1*, *division-ring*}) *hyperdual*
shows $-\ (b\ /\ c) = a \longleftrightarrow (\mathit{if}\ \mathit{Base}\ c \neq 0\ \mathit{then}\ -\ b = a * c\ \mathit{else}\ a = 0)$
⟨*proof*⟩

lemma *hyp-eq-minus-divide-eq* [*divide-simps*]:
fixes $a\ b\ c$ :: ($'a$ :: {*inverse*, *comm-ring-1*, *division-ring*}) *hyperdual*
shows $a = -\ (b\ /\ c) \longleftrightarrow (\mathit{if}\ \mathit{Base}\ c \neq 0\ \mathit{then}\ a * c = -\ b\ \mathit{else}\ a = 0)$
⟨*proof*⟩

1.4 Real Scaling, Real Vector and Real Algebra

If the components can be scaled by real numbers then so can the hyperduals. We define the scaling pointwise.

instantiation *hyperdual* :: (*scaleR*) *scaleR*
begin

primcorec *scaleR-hyperdual*
where
$\mathit{Base}\ (f *_R x) = f *_R \mathit{Base}\ x$
$|\ \mathit{Eps1}\ (f *_R x) = f *_R \mathit{Eps1}\ x$
$|\ \mathit{Eps2}\ (f *_R x) = f *_R \mathit{Eps2}\ x$
$|\ \mathit{Eps12}\ (f *_R x) = f *_R \mathit{Eps12}\ x$

instance
⟨*proof*⟩
end

If the components form a real vector space then so do the hyperduals.

instance *hyperdual* :: (*real-vector*) *real-vector*
⟨*proof*⟩

If the components form a real algebra then so do the hyperduals

instance *hyperdual* :: (*real-algebra-1*) *real-algebra-1*

$\langle proof \rangle$

If the components are reals then *of-real* matches our embedding *of-comp*, and $(*_R)$ matches our scalar product $(*_H)$.

lemma *of-real* = *of-comp*
$\langle proof \rangle$

lemma *scaleR-eq-scale*:
$(*_R) = (*_H)$
$\langle proof \rangle$

Hyperdual scalar product $(*_H)$ is compatible with $(*_R)$.

lemma *scaleH-scaleR*:
fixes a :: $'a$ :: *real-algebra-1*
and b :: $'a$ *hyperdual*
shows $(f *_R a) *_H b = f *_R a *_H b$
and $a *_H f *_R b = f *_R a *_H b$
$\langle proof \rangle$

1.5 Real Inner Product and Real-Normed Vector Space

We now take a closer look at hyperduals as a real vector space.

If the components form a real inner product space then we can define one on the hyperduals as the sum of componentwise inner products. The norm is then defined as the square root of that inner product. We define signum, distance, uniformity and openness similarly as they are defined for complex numbers.

instantiation *hyperdual* :: (*real-inner*) *real-inner*
begin

definition *inner-hyperdual* :: $'a$ *hyperdual* $\Rightarrow$ $'a$ *hyperdual* $\Rightarrow$ *real*
where $x \cdot y$ = *Base* $x \cdot$ *Base* y + *Eps1* $x \cdot$ *Eps1* y + *Eps2* $x \cdot$ *Eps2* y + *Eps12* $x \cdot$ *Eps12* y

definition *norm-hyperdual* :: $'a$ *hyperdual* $\Rightarrow$ *real*
where *norm-hyperdual* x = *sqrt* $(x \cdot x)$

definition *sgn-hyperdual* :: $'a$ *hyperdual* $\Rightarrow$ $'a$ *hyperdual*
where *sgn-hyperdual* $x = x /_R$ *norm* x

definition *dist-hyperdual* :: $'a$ *hyperdual* $\Rightarrow$ $'a$ *hyperdual* $\Rightarrow$ *real*
where *dist-hyperdual* a b = *norm*$(a - b)$

definition *uniformity-hyperdual* :: ($'a$ *hyperdual* $\times$ $'a$ *hyperdual*) *filter*
where *uniformity-hyperdual* = (*INF* $e \in \{0 <..\}$. *principal* $\{(x, y).$ *dist* x $y < e\}$)

definition *open-hyperdual* :: ($'a$ *hyperdual*) *set* $\Rightarrow$ *bool*

where *open-hyperdual U* $\longleftrightarrow$ ($\forall x{\in}U.$ *eventually* ($\lambda(x', y).\ x' = x \longrightarrow y \in U$) *uniformity*)

instance
⟨*proof*⟩
end

We then show that with this norm hyperduals with components that form a real normed algebra do not themselves form a normed algebra, by counter-example to the assumption that class adds.

lemma *not-normed-algebra*:
shows $\neg(\forall x\ y :: ('a :: \{\textit{real-normed-algebra-1}, \textit{real-inner}\})\ \textit{hyperdual}\ .\ \textit{norm}\ (x * y) \leq \textit{norm}\ x * \textit{norm}\ y)$
⟨*proof*⟩

1.6 Euclidean Space

Next we define a basis for the space, consisting of four elements one for each component with *1* in the relevant component and *0* elsewhere.

definition *ba* :: (*'a* :: *zero-neq-one*) *hyperdual*
where *ba* = *Hyperdual 1 0 0 0*
definition *e1* :: (*'a* :: *zero-neq-one*) *hyperdual*
where *e1* = *Hyperdual 0 1 0 0*
definition *e2* :: (*'a* :: *zero-neq-one*) *hyperdual*
where *e2* = *Hyperdual 0 0 1 0*
definition *e12* :: (*'a* :: *zero-neq-one*) *hyperdual*
where *e12* = *Hyperdual 0 0 0 1*

lemmas *hyperdual-bases* = *ba-def e1-def e2-def e12-def*

Using the constructor *Hyperdual* is equivalent to using the linear combination with coefficients the relevant arguments.

lemma *Hyperdual-eq*:
fixes *a b c d* :: *'a* :: *ring-1*
shows *Hyperdual a b c d* $= a *_H ba + b *_H e1 + c *_H e2 + d *_H e12$
⟨*proof*⟩

Projecting from the combination returns the relevant coefficient:

lemma *hyperdual-comb-sel* [*simp*]:
fixes *a b c d* :: *'a* :: *ring-1*
shows *Base* $(a *_H ba + b *_H e1 + c *_H e2 + d *_H e12) = a$
and *Eps1* $(a *_H ba + b *_H e1 + c *_H e2 + d *_H e12) = b$
and *Eps2* $(a *_H ba + b *_H e1 + c *_H e2 + d *_H e12) = c$
and *Eps12* $(a *_H ba + b *_H e1 + c *_H e2 + d *_H e12) = d$
⟨*proof*⟩

Any hyperdual number is a linear combination of these four basis elements.

lemma *hyperdual-linear-comb*:
  fixes $x :: ('a :: ring\text{-}1)\ hyperdual$
  obtains $a\ b\ c\ d :: 'a$ **where** $x = a *_H ba + b *_H e1 + c *_H e2 + d *_H e12$
  ⟨*proof*⟩

The linear combination expressing any hyperdual number has as coefficients the projections of that number onto the relevant basis element.

lemma *hyperdual-eq*:
  fixes $x :: ('a :: ring\text{-}1)\ hyperdual$
  shows $x = Base\ x *_H ba + Eps1\ x *_H e1 + Eps2\ x *_H e2 + Eps12\ x *_H e12$
  ⟨*proof*⟩

Equality of hyperduals as linear combinations is equality of corresponding components.

lemma *hyperdual-eq-parts-cancel* [*simp*]:
  fixes $a\ b\ c\ d :: 'a :: ring\text{-}1$
  shows $(a *_H ba + b *_H e1 + c *_H e2 + d *_H e12 = a' *_H ba + b' *_H e1 + c' *_H e2 + d' *_H e12) \equiv$
    $(a = a' \wedge b = b' \wedge c = c' \wedge d = d')$
  ⟨*proof*⟩

lemma *scaleH-cancel* [*simp*]:
  fixes $a\ b :: 'a :: ring\text{-}1$
  shows $(a *_H ba = b *_H ba) \equiv (a = b)$
    and $(a *_H e1 = b *_H e1) \equiv (a = b)$
    and $(a *_H e2 = b *_H e2) \equiv (a = b)$
    and $(a *_H e12 = b *_H e12) \equiv (a = b)$
  ⟨*proof*⟩

We can now also show that the multiplication we use indeed has the hyperdual units nilpotent.

lemma *epsilon-squares* [*simp*]:
  $(e1 :: ('a :: ring\text{-}1)\ hyperdual) * e1 = 0$
  $(e2 :: ('a :: ring\text{-}1)\ hyperdual) * e2 = 0$
  $(e12 :: ('a :: ring\text{-}1)\ hyperdual) * e12 = 0$
  ⟨*proof*⟩

However none of the hyperdual units is zero.

lemma *hyperdual-bases-nonzero* [*simp*]:
  $ba \neq 0$
  $e1 \neq 0$
  $e2 \neq 0$
  $e12 \neq 0$
  ⟨*proof*⟩

Hyperdual units are orthogonal.

lemma *hyperdual-bases-ortho* [*simp*]:
  $(ba :: ('a :: \{real\text{-}inner, zero\text{-}neq\text{-}one\})\ hyperdual) \cdot e1 = 0$

$(ba :: ('a :: \{real\text{-}inner,zero\text{-}neq\text{-}one\})\ hyperdual) \cdot e2 = 0$
$(ba :: ('a :: \{real\text{-}inner,zero\text{-}neq\text{-}one\})\ hyperdual) \cdot e12 = 0$
$(e1 :: ('a :: \{real\text{-}inner,zero\text{-}neq\text{-}one\})\ hyperdual) \cdot e2 = 0$
$(e1 :: ('a :: \{real\text{-}inner,zero\text{-}neq\text{-}one\})\ hyperdual) \cdot e12 = 0$
$(e2 :: ('a :: \{real\text{-}inner,zero\text{-}neq\text{-}one\})\ hyperdual) \cdot e12 = 0$
⟨*proof*⟩

Hyperdual units of norm equal to 1.

lemma *hyperdual-bases-norm* [*simp*]:
$(ba :: ('a :: \{real\text{-}inner,real\text{-}normed\text{-}algebra\text{-}1\})\ hyperdual) \cdot ba = 1$
$(e1 :: ('a :: \{real\text{-}inner,real\text{-}normed\text{-}algebra\text{-}1\})\ hyperdual) \cdot e1 = 1$
$(e2 :: ('a :: \{real\text{-}inner,real\text{-}normed\text{-}algebra\text{-}1\})\ hyperdual) \cdot e2 = 1$
$(e12 :: ('a :: \{real\text{-}inner,real\text{-}normed\text{-}algebra\text{-}1\})\ hyperdual) \cdot e12 = 1$
⟨*proof*⟩

We can also express earlier operations in terms of the linear combination.

lemma *add-hyperdual-parts*:
fixes $a\ b\ c\ d :: 'a :: ring\text{-}1$
shows $(a *_H ba + b *_H e1 + c *_H e2 + d *_H e12) + (a' *_H ba + b' *_H e1 + c' *_H e2 + d' *_H e12) =$
$(a + a') *_H ba + (b + b') *_H e1 + (c + c') *_H e2 + (d + d') *_H e12$
⟨*proof*⟩

lemma *times-hyperdual-parts*:
fixes $a\ b\ c\ d :: 'a :: ring\text{-}1$
shows $(a *_H ba + b *_H e1 + c *_H e2 + d *_H e12) * (a' *_H ba + b' *_H e1 + c' *_H e2 + d' *_H e12) =$
$(a * a') *_H ba + (a * b' + b * a') *_H e1 + (a * c' + c * a') *_H e2 + (a * d' + b * c' + c * b' + d * a') *_H e12$
⟨*proof*⟩

lemma *inverse-hyperdual-parts*:
fixes $a\ b\ c\ d :: 'a :: \{inverse,ring\text{-}1\}$
shows $inverse\ (a *_H ba + b *_H e1 + c *_H e2 + d *_H e12) =$
$(1 / a) *_H ba + (- b / a \hat{}\ 2) *_H e1 + (- c / a \hat{}\ 2) *_H e2 + (2 * (b * c / a \hat{}\ 3) - d / a \hat{}\ 2) *_H e12$
⟨*proof*⟩

Next we show that hyperduals form a euclidean space with the help of the basis we defined earlier and the above inner product if the component is an instance of *euclidean-space* and *real-algebra-1*. The basis of this space is each of the basis elements we defined scaled by each of the basis elements of the component type, representing the expansion of the space for each component of the hyperdual numbers.

instantiation *hyperdual* :: ({*euclidean-space*, *real-algebra-1*}) *euclidean-space*
begin

definition *Basis-hyperdual* :: (*'a hyperdual*) *set*

where $Basis = (\bigcup i{\in}\{ba,e1,e2,e12\}.\ (\lambda u.\ u *_H i)\ `\ Basis)$

instance
$\langle proof \rangle$
end

1.7 Bounded Linear Projections

Now we can show that each projection to a basis element is a bounded linear map.

lemma *bounded-linear-Base*: *bounded-linear Base*
$\langle proof \rangle$
lemma *bounded-linear-Eps1*: *bounded-linear Eps1*
$\langle proof \rangle$
lemma *bounded-linear-Eps2*: *bounded-linear Eps2*
$\langle proof \rangle$
lemma *bounded-linear-Eps12*: *bounded-linear Eps12*
$\langle proof \rangle$

This bounded linearity gives us a range of useful theorems about limits, convergence and derivatives of these projections.

lemmas *tendsto-Base* = *bounded-linear.tendsto*[*OF bounded-linear-Base*]
lemmas *tendsto-Eps1* = *bounded-linear.tendsto*[*OF bounded-linear-Eps1*]
lemmas *tendsto-Eps2* = *bounded-linear.tendsto*[*OF bounded-linear-Eps2*]
lemmas *tendsto-Eps12* = *bounded-linear.tendsto*[*OF bounded-linear-Eps12*]

lemmas *has-derivative-Base* = *bounded-linear.has-derivative*[*OF bounded-linear-Base*]
lemmas *has-derivative-Eps1* = *bounded-linear.has-derivative*[*OF bounded-linear-Eps1*]
lemmas *has-derivative-Eps2* = *bounded-linear.has-derivative*[*OF bounded-linear-Eps2*]
lemmas *has-derivative-Eps12* = *bounded-linear.has-derivative*[*OF bounded-linear-Eps12*]

1.8 Convergence

lemma *inner-mult-le-mult-inner*:
fixes $a\ b :: 'a :: \{real\text{-}inner, real\text{-}normed\text{-}algebra\}$
shows $((a * b) \cdot (a * b)) \leq (a \cdot a) * (b \cdot b)$
$\langle proof \rangle$

lemma *bounded-bilinear-scaleH*:
bounded-bilinear $((*_H) :: ('a :: \{real\text{-}normed\text{-}algebra\text{-}1,\ real\text{-}inner\}) \Rightarrow 'a\ hyperdual \Rightarrow 'a\ hyperdual)$
$\langle proof \rangle$

lemmas *tendsto-scaleH* = *bounded-bilinear.tendsto*[*OF bounded-bilinear-scaleH*]

We describe how limits behave for general hyperdual-valued functions.

First we prove that we can go from convergence of the four component functions to the convergence of the hyperdual-valued function whose components they define.

lemma *tendsto-Hyperdual*:
  fixes $f :: 'a \Rightarrow ('b :: \{real\text{-}normed\text{-}algebra\text{-}1,\ real\text{-}inner\})$
  assumes $(f \longrightarrow a)\ F$
    and $(g \longrightarrow b)\ F$
    and $(h \longrightarrow c)\ F$
    and $(i \longrightarrow d)\ F$
   shows $((\lambda x.\ Hyperdual\ (f\ x)\ (g\ x)\ (h\ x)\ (i\ x)) \longrightarrow Hyperdual\ a\ b\ c\ d)\ F$
⟨*proof*⟩

Next we complete the equivalence by proving the other direction, from convergence of a hyperdual-valued function to the convergence of the projected component functions.

lemma *tendsto-hyperdual-iff*:
  fixes $f :: 'a \Rightarrow ('b :: \{real\text{-}normed\text{-}algebra\text{-}1,\ real\text{-}inner\})\ hyperdual$
  shows $(f \longrightarrow x)\ F \longleftrightarrow$
   $((\lambda x.\ Base\ (f\ x)) \longrightarrow Base\ x)\ F\ \wedge$
   $((\lambda x.\ Eps1\ (f\ x)) \longrightarrow Eps1\ x)\ F\ \wedge$
   $((\lambda x.\ Eps2\ (f\ x)) \longrightarrow Eps2\ x)\ F\ \wedge$
   $((\lambda x.\ Eps12\ (f\ x)) \longrightarrow Eps12\ x)\ F$
⟨*proof*⟩

1.9 Derivatives

We describe how derivatives of hyperdual-valued functions behave. Due to hyperdual numbers not forming a normed field, the derivative relation we must use is the general Fréchet derivative (*has-derivative*).

The left to right implication of the following equivalence is easily proved by the known derivative behaviour of the projections. The other direction is more difficult, because we have to construct the two requirements of the (*has-derivative*) relation, the limit and the bounded linearity of the derivative. While the limit is simple to construct from the component functions by previous lemma, the bounded linearity is more involved.

lemma *has-derivative-hyperdual-iff*:
  fixes $f :: ('a :: real\text{-}normed\text{-}vector) \Rightarrow ('b :: \{real\text{-}normed\text{-}algebra\text{-}1,\ real\text{-}inner\})$
hyperdual
  shows $(f\ has\text{-}derivative\ Df)\ F \longleftrightarrow$
   $((\lambda x.\ Base\ (f\ x))\ has\text{-}derivative\ (\lambda x.\ Base\ (Df\ x)))\ F\ \wedge$
   $((\lambda x.\ Eps1\ (f\ x))\ has\text{-}derivative\ (\lambda x.\ Eps1\ (Df\ x)))\ F\ \wedge$
   $((\lambda x.\ Eps2\ (f\ x))\ has\text{-}derivative\ (\lambda x.\ Eps2\ (Df\ x)))\ F\ \wedge$
   $((\lambda x.\ Eps12\ (f\ x))\ has\text{-}derivative\ (\lambda x.\ Eps12\ (Df\ x)))\ F$
⟨*proof*⟩

Stop automatically unfolding hyperduals into components outside this theory:

lemmas [*iff del*] = *hyperdual-eq-iff*

end

2 Twice Field Differentiable

theory *TwiceFieldDifferentiable*
  imports *HOL−Analysis.Analysis*
begin

2.1 Differentiability on a Set

A function is differentiable on a set iff it is differentiable at any point within that set.

definition *field-differentiable-on* :: $('a \Rightarrow 'a$::*real-normed-field*$) \Rightarrow 'a$ *set* $\Rightarrow$ *bool*
  (**infix** *field′-differentiable′-on 50*)
  where *f field-differentiable-on s* $\equiv \forall x \in s.$ *f field-differentiable* (*at x within s*)

This is preserved for subsets.

lemma *field-differentiable-on-subset*:
  assumes *f field-differentiable-on S*
    and $T \subseteq S$
   shows *f field-differentiable-on T*
  ⟨*proof*⟩

2.2 Twice Differentiability

Informally, a function is twice differentiable at x iff it is differentiable on some neighbourhood of x and its derivative is differentiable at x.

definition *twice-field-differentiable-at* :: $['a \Rightarrow 'a$::*real-normed-field*, $'a\ ] \Rightarrow$ *bool*
  (**infixr** (*twice′-field′-differentiable′-at*) *50*)
  where *f twice-field-differentiable-at x* $\equiv$
    $\exists S.$ *f field-differentiable-on S* $\wedge$ $x \in$ *interior S* $\wedge$ (*deriv f*) *field-differentiable* (*at x*)

lemma *once-field-differentiable-at*:
  f twice-field-differentiable-at x $\Longrightarrow$ *f field-differentiable* (*at x*)
  ⟨*proof*⟩

lemma *deriv-field-differentiable-at*:
  f twice-field-differentiable-at x $\Longrightarrow$ *deriv f field-differentiable* (*at x*)
  ⟨*proof*⟩

For a composition of two functions twice differentiable at x, the chain rule eventually holds on some neighbourhood of x.

lemma *eventually-deriv-compose*:
  assumes $\exists S.$ *f field-differentiable-on S* $\wedge$ $x \in$ *interior S*
    and *g twice-field-differentiable-at* (*f x*)

shows $\forall_F$ *x in nhds x. deriv* ($\lambda x.$ *g* (*f x*)) *x* = *deriv g* (*f x*) $*$ *deriv f x*
⟨*proof*⟩

lemma *eventually-deriv-compose′*:
assumes *f twice-field-differentiable-at x*
and *g twice-field-differentiable-at* (*f x*)
shows $\forall_F$ *x in nhds x. deriv* ($\lambda x.$ *g* (*f x*)) *x* = *deriv g* (*f x*) $*$ *deriv f x*
⟨*proof*⟩

Composition of twice differentiable functions is twice differentiable.

lemma *twice-field-differentiable-at-compose*:
assumes *f twice-field-differentiable-at x*
and *g twice-field-differentiable-at* (*f x*)
shows ($\lambda x.$ *g* (*f x*)) *twice-field-differentiable-at x*
⟨*proof*⟩

2.2.1 Constant

lemma *twice-field-differentiable-at-const* [*simp*, *intro*]:
($\lambda x.$ *a*) *twice-field-differentiable-at x*
⟨*proof*⟩

2.2.2 Identity

lemma *twice-field-differentiable-at-ident* [*simp*, *intro*]:
($\lambda x.$ *x*) *twice-field-differentiable-at x*
⟨*proof*⟩

2.2.3 Constant Multiplication

lemma *twice-field-differentiable-at-cmult* [*simp*, *intro*]:
($*$) *k twice-field-differentiable-at x*
⟨*proof*⟩

lemma *twice-field-differentiable-at-uminus* [*simp*, *intro*]:
uminus twice-field-differentiable-at x
⟨*proof*⟩

lemma *twice-field-differentiable-at-uminus-fun* [*intro*]:
assumes *f twice-field-differentiable-at x*
shows ($\lambda x.$ − *f x*) *twice-field-differentiable-at x*
⟨*proof*⟩

2.2.4 Real Scaling

lemma *deriv-scaleR-right-id* [*simp*]:
(*deriv* (($*_R$) *k*)) = ($\lambda z.$ *k* $*_R$ *1*)
⟨*proof*⟩

lemma *deriv-deriv-scaleR-right-id* [*simp*]:

deriv (*deriv* (($*_R$) *k*)) = ($\lambda z.$ *0*)
⟨*proof*⟩

lemma *deriv-scaleR-right*:
f field-differentiable (*at z*) $\Longrightarrow$ *deriv* ($\lambda x.$ $k *_R f\ x$) $z = k *_R$ *deriv f z*
⟨*proof*⟩

lemma *field-differentiable-scaleR-right* [*intro*]:
f field-differentiable F $\Longrightarrow$ ($\lambda x.$ $c *_R f\ x$) *field-differentiable F*
⟨*proof*⟩

lemma *has-field-derivative-scaleR-deriv-right*:
assumes *f twice-field-differentiable-at z*
shows (($\lambda x.$ $k *_R$ *deriv f x*) *has-field-derivative* $k *_R$ *deriv* (*deriv f*) *z*) (*at z*)
⟨*proof*⟩

lemma *deriv-scaleR-deriv-right*:
assumes *f twice-field-differentiable-at z*
shows *deriv* ($\lambda x.$ $k *_R$ *deriv f x*) $z = k *_R$ *deriv* (*deriv f*) *z*
⟨*proof*⟩

lemma *twice-field-differentiable-at-scaleR* [*simp*, *intro*]:
($*_R$) *k twice-field-differentiable-at x*
⟨*proof*⟩

lemma *twice-field-differentiable-at-scaleR-fun* [*simp*, *intro*]:
assumes *f twice-field-differentiable-at x*
shows ($\lambda x.$ $k *_R f\ x$) *twice-field-differentiable-at x*
⟨*proof*⟩

2.2.5 Addition

lemma *eventually-deriv-add*:
assumes *f twice-field-differentiable-at x*
and *g twice-field-differentiable-at x*
shows $\forall_F$ *x in nhds x*. *deriv* ($\lambda x.$ $f\ x + g\ x$) $x =$ *deriv f x* + *deriv g x*
⟨*proof*⟩

lemma *twice-field-differentiable-at-add* [*intro*]:
assumes *f twice-field-differentiable-at x*
and *g twice-field-differentiable-at x*
shows ($\lambda x.$ $f\ x + g\ x$) *twice-field-differentiable-at x*
⟨*proof*⟩

lemma *deriv-add-id-const* [*simp*]:
deriv ($\lambda x.$ $x + a$) = ($\lambda z.$ *1*)
⟨*proof*⟩

lemma *deriv-deriv-add-id-const* [*simp*]:

deriv (*deriv* ($\lambda x.$ $x + a$)) $z = 0$
⟨*proof*⟩

lemma *twice-field-differentiable-at-cadd* [*simp*]:
($\lambda x.$ $x + a$) *twice-field-differentiable-at* x
⟨*proof*⟩

2.2.6 Linear Function

lemma *twice-field-differentiable-at-linear* [*simp*, *intro*]:
($\lambda x.$ $k * x + a$) *twice-field-differentiable-at* x
⟨*proof*⟩

lemma *twice-field-differentiable-at-linearR* [*simp*, *intro*]:
($\lambda x.$ $k *_R x + a$) *twice-field-differentiable-at* x
⟨*proof*⟩

2.2.7 Multiplication

lemma *eventually-deriv-mult*:
assumes f *twice-field-differentiable-at* x
and g *twice-field-differentiable-at* x
shows $\forall_F$ x *in nhds* x. *deriv* ($\lambda x.$ $f\ x * g\ x$) $x = f\ x *$ *deriv* $g\ x +$ *deriv* $f\ x * g\ x$
⟨*proof*⟩

lemma *twice-field-differentiable-at-mult* [*intro*]:
assumes f *twice-field-differentiable-at* x
and g *twice-field-differentiable-at* x
shows ($\lambda x.$ $f\ x * g\ x$) *twice-field-differentiable-at* x
⟨*proof*⟩

2.2.8 Sine and Cosine

lemma *deriv-sin* [*simp*]: *deriv sin* $a =$ *cos* a
⟨*proof*⟩

lemma *deriv-sinf* [*simp*]: *deriv sin* = ($\lambda x.$ *cos* x)
⟨*proof*⟩

lemma *deriv-cos* [*simp*]: *deriv cos* $a = -$ *sin* a
⟨*proof*⟩

lemma *deriv-cosf* [*simp*]: *deriv cos* = ($\lambda x.$ $-$ *sin* x)
⟨*proof*⟩

lemma *deriv-sin-minus* [*simp*]:
deriv ($\lambda x.$ $-$ *sin* x) $a = -$ *deriv* ($\lambda x.$ *sin* x) a
⟨*proof*⟩

lemma *twice-field-differentiable-at-sin* [*simp*, *intro*]:

sin twice-field-differentiable-at x
⟨*proof*⟩

lemma *twice-field-differentiable-at-sin-fun* [*intro*]:
assumes *f twice-field-differentiable-at x*
shows (λx. *sin* (*f x*)) *twice-field-differentiable-at x*
⟨*proof*⟩

lemma *twice-field-differentiable-at-cos* [*simp*, *intro*]:
cos twice-field-differentiable-at x
⟨*proof*⟩

lemma *twice-field-differentiable-at-cos-fun* [*intro*]:
assumes *f twice-field-differentiable-at x*
shows (λx. *cos* (*f x*)) *twice-field-differentiable-at x*
⟨*proof*⟩

2.2.9 Exponential

lemma *deriv-exp* [*simp*]: *deriv exp x* = *exp x*
⟨*proof*⟩

lemma *deriv-expf* [*simp*]: *deriv exp* = *exp*
⟨*proof*⟩

lemma *deriv-deriv-exp* [*simp*]: *deriv* (*deriv exp*) *x* = *exp x*
⟨*proof*⟩

lemma *twice-field-differentiable-at-exp* [*simp*, *intro*]:
exp twice-field-differentiable-at x
⟨*proof*⟩

lemma *twice-field-differentiable-at-exp-fun* [*simp*, *intro*]:
assumes *f twice-field-differentiable-at x*
shows (λx. *exp* (*f x*)) *twice-field-differentiable-at x*
⟨*proof*⟩

2.2.10 Square Root

lemma *deriv-real-sqrt* [*simp*]: $x > 0 \Longrightarrow$ *deriv sqrt x* = *inverse* (*sqrt x*) / *2*
⟨*proof*⟩

lemma *has-real-derivative-inverse-sqrt*:
assumes $x > 0$
shows ((λx. *inverse* (*sqrt x*) / *2*) *has-real-derivative* − (*inverse* (*sqrt x* ^ *3*) / *4*)) (*at x*)
⟨*proof*⟩

lemma *deriv-deriv-real-sqrt'*:
assumes $x > 0$

shows *deriv* ($\lambda x.$ *inverse* (*sqrt* x) / 2) $x = -$ *inverse* ((*sqrt* x)^3)/4
⟨*proof*⟩

lemma *has-real-derivative-deriv-sqrt*:
assumes $x > 0$
shows (*deriv sqrt has-real-derivative* $-$ *inverse* (*sqrt* x ^ 3) / 4) (*at* x)
⟨*proof*⟩

lemma *deriv-deriv-real-sqrt* [*simp*]:
assumes $x > 0$
shows *deriv*(*deriv sqrt*) $x = -$ *inverse* ((*sqrt* x)^3)/4
⟨*proof*⟩

lemma *twice-field-differentiable-at-sqrt* [*simp*, *intro*]:
assumes $x > 0$
shows *sqrt twice-field-differentiable-at* x
⟨*proof*⟩

lemma *twice-field-differentiable-at-sqrt-fun* [*intro*]:
assumes *f twice-field-differentiable-at* x
and $f\ x > 0$
shows ($\lambda x.$ *sqrt* ($f\ x$)) *twice-field-differentiable-at* x
⟨*proof*⟩

2.2.11 Natural Power

lemma *field-differentiable-power* [*simp*]:
($\lambda x.\ x$ ^ n) *field-differentiable at* x
⟨*proof*⟩

lemma *deriv-power-fun* [*simp*]:
assumes *f field-differentiable at* x
shows *deriv* ($\lambda x.\ f\ x$ ^ n) $x =$ *of-nat* $n\ *$ *deriv* $f\ x * f\ x$ ^ $(n - 1)$
⟨*proof*⟩

lemma *deriv-power* [*simp*]:
deriv ($\lambda x.\ x$ ^ n) $x =$ *of-nat* $n * x$ ^ $(n - 1)$
⟨*proof*⟩

lemma *deriv-deriv-power* [*simp*]:
deriv (*deriv* ($\lambda x.\ x$ ^ n)) $x =$ *of-nat* $n\ *$ *of-nat* ($n -$ *Suc* 0) $* x$ ^ $(n - 2)$
⟨*proof*⟩

lemma *twice-field-differentiable-at-power* [*simp*, *intro*]:
($\lambda x.\ x$ ^ n) *twice-field-differentiable-at* x
⟨*proof*⟩

lemma *twice-field-differentiable-at-power-fun* [*intro*]:
assumes *f twice-field-differentiable-at* x

shows $(\lambda x.\ f\ x\ \widehat{}\ n)$ *twice-field-differentiable-at x*
⟨*proof*⟩

2.2.12 Inverse

lemma *eventually-deriv-inverse*:
assumes $x \neq 0$
shows $\forall_F\ x$ *in nhds x. deriv inverse* $x = -\ 1\ /\ (x\ \widehat{}\ 2)$
⟨*proof*⟩

lemma *deriv-deriv-inverse* [*simp*]:
assumes $x \neq 0$
shows *deriv* (*deriv inverse*) $x = 2 * $ *inverse* $(x\ \widehat{}\ 3)$
⟨*proof*⟩

lemma *twice-field-differentiable-at-inverse* [*simp*, *intro*]:
assumes $x \neq 0$
shows *inverse twice-field-differentiable-at x*
⟨*proof*⟩

lemma *twice-field-differentiable-at-inverse-fun* [*simp*, *intro*]:
assumes *f twice-field-differentiable-at x*
$f\ x \neq 0$
shows $(\lambda x.$ *inverse* $(f\ x))$ *twice-field-differentiable-at x*
⟨*proof*⟩

lemma *twice-field-differentiable-at-divide* [*intro*]:
assumes *f twice-field-differentiable-at x*
and *g twice-field-differentiable-at x*
and $g\ x \neq 0$
shows $(\lambda x.\ f\ x\ /\ g\ x)$ *twice-field-differentiable-at x*
⟨*proof*⟩

2.2.13 Polynomial

lemma *twice-field-differentiable-at-polyn* [*simp*, *intro*]:
fixes *coef* :: *nat* ⇒ *'a* :: {*real-normed-field*}
and *n* :: *nat*
shows $(\lambda x.\ \sum i<n.\ coef\ i * x\ \widehat{}\ i)$ *twice-field-differentiable-at x*
⟨*proof*⟩

lemma *twice-field-differentiable-at-polyn-fun* [*simp*]:
fixes *coef* :: *nat* ⇒ *'a* :: {*real-normed-field*}
and *n* :: *nat*
assumes *f twice-field-differentiable-at x*
shows $(\lambda x.\ \sum i<n.\ coef\ i * f\ x\ \widehat{}\ i)$ *twice-field-differentiable-at x*
⟨*proof*⟩

end

3 Hyperdual Extension of Functions

theory *HyperdualFunctionExtension*
imports *Hyperdual TwiceFieldDifferentiable*
begin

The following is an important fact in the derivation of the hyperdual extension.

lemma
fixes $x :: ('a :: comm\text{-}ring\text{-}1)\ hyperdual$ **and** $n :: nat$
assumes $Base\ x = 0$
shows $x\ \hat{}\ (n + 3) = 0$
$\langle proof \rangle$

We define the extension of a function to the hyperdual numbers.

primcorec $hypext :: (('a :: real\text{-}normed\text{-}field) \Rightarrow 'a) \Rightarrow 'a\ hyperdual \Rightarrow 'a\ hyperdual$ $(\langle *h* \text{ -}\rangle\ [80]\ 80)$
where
$Base\ ((*h*\ f)\ x) = f\ (Base\ x)$
$|\ Eps1\ ((*h*\ f)\ x) = Eps1\ x * deriv\ f\ (Base\ x)$
$|\ Eps2\ ((*h*\ f)\ x) = Eps2\ x * deriv\ f\ (Base\ x)$
$|\ Eps12\ ((*h*\ f)\ x) = Eps12\ x * deriv\ f\ (Base\ x) + Eps1\ x * Eps2\ x * deriv\ (deriv\ f)\ (Base\ x)$

This has the expected behaviour when expressed in terms of the units.

lemma *hypext-Hyperdual-eq*:
$(*h*\ f)\ (Hyperdual\ a\ b\ c\ d) =$
$Hyperdual\ (f\ a)\ (b * deriv\ f\ a)\ (c * deriv\ f\ a)\ (d * deriv\ f\ a + b * c * deriv\ (deriv\ f)\ a)$
$\langle proof \rangle$

lemma *hypext-Hyperdual-eq-parts*:
$(*h*\ f)\ (Hyperdual\ a\ b\ c\ d) =$
$f\ a *_H ba + (b * deriv\ f\ a) *_H e1 + (c * deriv\ f\ a) *_H e2 +$
$(d * deriv\ f\ a + b * c * deriv\ (deriv\ f)\ a) *_H e12$
$\langle proof \rangle$

The extension can be used to extract the function value, and first and second derivatives at x when applied to $x *_H re + e1 + e2 + (0::'a) *_H e12$, which we denote by $\beta\ x$.

definition $hyperdualx :: ('a :: real\text{-}normed\text{-}field) \Rightarrow 'a\ hyperdual\ (\beta)$
where $\beta\ x = (Hyperdual\ x\ 1\ 1\ 0)$

lemma *hyperdualx-sel* [*simp*]:
shows $Base\ (\beta\ x) = x$
and $Eps1\ (\beta\ x) = 1$
and $Eps2\ (\beta\ x) = 1$
and $Eps12\ (\beta\ x) = 0$

$\langle proof \rangle$

lemma *hypext-extract-eq*:
$(*h* f)\ (\beta\ x) = f\, x *_H ba + deriv\, f\, x *_H e1 + deriv\, f\, x *_H e2 + deriv\ (deriv\, f)\ x *_H e12$
$\langle proof \rangle$

lemma *Base-hypext*:
$Base\ ((*h* f)\ (\beta\ x)) = f\, x$
$\langle proof \rangle$

lemma *Eps1-hypext*:
$Eps1\ ((*h* f)\ (\beta\ x)) = deriv\, f\, x$
$\langle proof \rangle$

lemma *Eps2-hypext*:
$Eps2\ ((*h* f)\ (\beta\ x)) = deriv\, f\, x$
$\langle proof \rangle$

lemma *Eps12-hypext*:
$Eps12\ ((*h* f)\ (\beta\ x)) = deriv\ (deriv\, f)\ x$
$\langle proof \rangle$

3.0.1 Convenience Interface

Define a datatype to hold the function value, and the first and second derivative values.

datatype $('a :: real\text{-}normed\text{-}field)\ derivs = Derivs\ (Value: 'a)\ (First: 'a)\ (Second: 'a)$

Then we convert a hyperdual number to derivative values by extracting the base component, one of the first-order components, and the second-order component.

fun $hyperdual\text{-}to\text{-}derivs :: ('a :: real\text{-}normed\text{-}field)\ hyperdual \Rightarrow 'a\ derivs$
where $hyperdual\text{-}to\text{-}derivs\ x = Derivs\ (Base\ x)\ (Eps1\ x)\ (Eps12\ x)$

Finally we define way of converting any compatible function into one that yields the value and the derivatives.

fun $autodiff :: ('a :: real\text{-}normed\text{-}field \Rightarrow 'a) \Rightarrow 'a \Rightarrow 'a\ derivs$
where $autodiff\ f = (\lambda x.\ hyperdual\text{-}to\text{-}derivs\ ((*h* f)\ (\beta\ x)))$

lemma *autodiff-sel*:
$Value\ (autodiff\ f\ x) = Base\ ((*h* f)\ (\beta\ x))$
$First\ (autodiff\ f\ x) = Eps1\ ((*h* f)\ (\beta\ x))$
$Second\ (autodiff\ f\ x) = Eps12\ ((*h* f)\ (\beta\ x))$
$\langle proof \rangle$

The result contains the expected values.

lemma *autodiff-extract-value*:
  Value (*autodiff f x*) = *f x*
  ⟨*proof*⟩

lemma *autodiff-extract-first*:
  First (*autodiff f x*) = *deriv f x*
  ⟨*proof*⟩

lemma *autodiff-extract-second*:
  Second (*autodiff f x*) = *deriv* (*deriv f*) *x*
  ⟨*proof*⟩

The derivative components of the result are actual derivatives if the function is sufficiently differentiable on that argument.

lemma *autodiff-first-derivative*:
  assumes *f field-differentiable* (*at x*)
  shows (*f has-field-derivative First* (*autodiff f x*)) (*at x*)
  ⟨*proof*⟩

lemma *autodiff-second-derivative*:
  assumes *f twice-field-differentiable-at x*
  shows ((*deriv f*) *has-field-derivative Second* (*autodiff f x*)) (*at x*)
  ⟨*proof*⟩

3.0.2 Composition

Composition of hyperdual extensions is the hyperdual extension of composition:

lemma *hypext-compose*:
  assumes *f twice-field-differentiable-at* (*Base x*)
    and *g twice-field-differentiable-at* (*f* (*Base x*))
   shows ($*h*$ ($\lambda x.$ *g* (*f x*))) *x* = ($*h*$ *g*) (($*h*$ *f*) *x*)
⟨*proof*⟩

3.1 Concrete Instances

3.1.1 Constant

Component embedding is an extension of the constant function.

lemma *hypext-const* [*simp*]:
  ($*h*$ ($\lambda x.$ *a*)) *x* = *of-comp a*
  ⟨*proof*⟩

lemma *autodiff* ($\lambda x.$ *a*) = ($\lambda x.$ *Derivs a 0 0*)
  ⟨*proof*⟩

3.1.2 Identity

Identity is an extension of the component identity.

lemma *hypext-ident*:
  $(*h* \ (\lambda x. \ x)) \ x = x$
  $\langle proof \rangle$

3.1.3 Component Scalar Multiplication

Component scaling is an extension of component constant multiplication:

lemma *hypext-scaleH*:
  $(*h* \ (\lambda x. \ k * x)) \ x = k *_H \ x$
  $\langle proof \rangle$

lemma *hypext-fun-scaleH*:
  assumes f *twice-field-differentiable-at* $(Base \ x)$
  shows $(*h* \ (\lambda x. \ k * f \ x)) \ x = k *_H \ (*h* \ f) \ x$
  $\langle proof \rangle$

Unary minus is just an instance of constant multiplication:

lemma *hypext-uminus*:
  $(*h* \ uminus) \ x = - \ x$
  $\langle proof \rangle$

3.1.4 Real Scalar Multiplication

Real scaling is an extension of component real scaling:

lemma *hypext-scaleR*:
  $(*h* \ (\lambda x. \ k *_R \ x)) \ x = k *_R \ x$
  $\langle proof \rangle$

lemma *hypext-fun-scaleR*:
  assumes f *twice-field-differentiable-at* $(Base \ x)$
  shows $(*h* \ (\lambda x. \ k *_R \ f \ x)) \ x = k *_R \ (*h* \ f) \ x$
  $\langle proof \rangle$

3.1.5 Addition

Addition of hyperdual extensions is a hyperdual extension of addition of functions.

lemma *hypext-fun-add*:
  assumes f *twice-field-differentiable-at* $(Base \ x)$
    and g *twice-field-differentiable-at* $(Base \ x)$
   shows $(*h* \ (\lambda x. \ f \ x + g \ x)) \ x = (*h* \ f) \ x + (*h* \ g) \ x$
$\langle proof \rangle$

lemma *hypext-cadd* [*simp*]:

$(*h* (\lambda x.\ x + a))\ x = x + \textit{of-comp}\ a$
⟨*proof*⟩

lemma *hypext-fun-cadd*:
assumes f *twice-field-differentiable-at* (*Base* x)
shows $(*h* (\lambda x.\ f\ x + a))\ x = (*h*\ f)\ x + \textit{of-comp}\ a$
⟨*proof*⟩

3.1.6 Component Linear Function

Hyperdual linear function is an extension of the component linear function:

lemma *hypext-linear*:
$(*h* (\lambda x.\ k * x + a))\ x = k *_H x + \textit{of-comp}\ a$
⟨*proof*⟩

lemma *hypext-fun-linear*:
assumes f *twice-field-differentiable-at* (*Base* x)
shows $(*h* (\lambda x.\ k * f\ x + a))\ x = k *_H (*h*\ f)\ x + \textit{of-comp}\ a$
⟨*proof*⟩

3.1.7 Real Linear Function

We have the same for real scaling instead of component multiplication:

lemma *hypext-linearR*:
$(*h* (\lambda x.\ k *_R x + a))\ x = k *_R x + \textit{of-comp}\ a$
⟨*proof*⟩

lemma *hypext-fun-linearR*:
assumes f *twice-field-differentiable-at* (*Base* x)
shows $(*h* (\lambda x.\ k *_R f\ x + a))\ x = k *_R (*h*\ f)\ x + \textit{of-comp}\ a$
⟨*proof*⟩

3.1.8 Multiplication

Extension of multiplication is multiplication of the functions' extensions.

lemma *hypext-fun-mult*:
assumes f *twice-field-differentiable-at* (*Base* x)
and g *twice-field-differentiable-at* (*Base* x)
shows $(*h* (\lambda z.\ f\ z * g\ z))\ x = (*h*\ f)\ x * (*h*\ g)\ x$
⟨*proof*⟩

3.1.9 Sine and Cosine

The extended sin and cos at an arbitrary hyperdual.

lemma *hypext-sin-Hyperdual*:
$(*h*\ sin)\ (\textit{Hyperdual}\ a\ b\ c\ d) = sin\ a *_H ba + (b * cos\ a) *_H e1 + (c * cos\ a) *_H e2 + (d * cos\ a - b * c * sin\ a) *_H e12$

$\langle proof \rangle$

lemma *hypext-cos-Hyperdual*:
$(*h* \ cos) \ (Hyperdual \ a \ b \ c \ d) = cos \ a *_H \ ba - (b * sin \ a) *_H \ e1 - (c * sin \ a) *_H \ e2 - (d * sin \ a + b * c * cos \ a) *_H \ e12$
$\langle proof \rangle$

lemma *Eps1-hypext-sin* [*simp*]:
$Eps1 \ ((*h* \ sin) \ x) = Eps1 \ x * cos \ (Base \ x)$
$\langle proof \rangle$

lemma *Eps2-hypext-sin* [*simp*]:
$Eps2 \ ((*h* \ sin) \ x) = Eps2 \ x * cos \ (Base \ x)$
$\langle proof \rangle$

lemma *Eps12-hypext-sin* [*simp*]:
$Eps12 \ ((*h* \ sin) \ x) = Eps12 \ x * cos \ (Base \ x) - Eps1 \ x * Eps2 \ x * sin \ (Base \ x)$
$\langle proof \rangle$

lemma *hypext-sin-e1* [*simp*]:
$(*h* \ sin) \ (x * e1) = e1 * x$
$\langle proof \rangle$

lemma *hypext-sin-e2* [*simp*]:
$(*h* \ sin) \ (x * e2) = e2 * x$
$\langle proof \rangle$

lemma *hypext-sin-e12* [*simp*]:
$(*h* \ sin) \ (x * e12) = e12 * x$
$\langle proof \rangle$

lemma *hypext-cos-e1* [*simp*]:
$(*h* \ cos) \ (x * e1) = 1$
$\langle proof \rangle$

lemma *hypext-cos-e2* [*simp*]:
$(*h* \ cos) \ (x * e2) = 1$
$\langle proof \rangle$

lemma *hypext-cos-e12* [*simp*]:
$(*h* \ cos) \ (x * e12) = 1$
$\langle proof \rangle$

The extended sin and cos at $\beta \ x$.

lemma *hypext-sin-extract*:
$(*h* \ sin) \ (\beta \ x) = sin \ x *_H \ ba + cos \ x *_H \ e1 + cos \ x *_H \ e2 - sin \ x *_H \ e12$
$\langle proof \rangle$

lemma *hypext-cos-extract*:

$(*h*\ cos)\ (\beta\ x) = cos\ x *_H\ ba - sin\ x *_H\ e1 - sin\ x *_H\ e2 - cos\ x *_H\ e12$
⟨*proof*⟩

Extracting the extended sin components at $\beta\ x$.

lemma *Base-hypext-sin-extract* [*simp*]:
$Base\ ((*h*\ sin)\ (\beta\ x)) = sin\ x$
⟨*proof*⟩

lemma *Eps2-hypext-sin-extract* [*simp*]:
$Eps2\ ((*h*\ sin)\ (\beta\ x)) = cos\ x$
⟨*proof*⟩

lemma *Eps12-hypext-sin-extract* [*simp*]:
$Eps12\ ((*h*\ sin)\ (\beta\ x)) = -\ sin\ x$
⟨*proof*⟩

Extracting the extended cos components at $\beta\ x$.

lemma *Base-hypext-cos-extract* [*simp*]:
$Base\ ((*h*\ cos)\ (\beta\ x)) = cos\ x$
⟨*proof*⟩

lemma *Eps2-hypext-cos-extract* [*simp*]:
$Eps2\ ((*h*\ cos)\ (\beta\ x)) = -\ sin\ x$
⟨*proof*⟩

lemma *Eps12-hypext-cos-extract* [*simp*]:
$Eps12\ ((*h*\ cos)\ (\beta\ x)) = -\ cos\ x$
⟨*proof*⟩

We get one of the key trigonometric properties for the extensions of sin and cos.

lemma $((*h*\ sin)\ x)^2 + ((*h*\ cos)\ x)^2 = 1$
⟨*proof*⟩

lemma $(*h*\ sin)\ x + (*h*\ cos)\ x = (*h*\ (\lambda x.\ sin\ x + cos\ x))\ x$
⟨*proof*⟩

3.1.10 Exponential

The exponential function extension behaves as expected.

lemma *hypext-exp-Hyperdual*:
$(*h*\ exp)\ (Hyperdual\ a\ b\ c\ d) =$
$exp\ a *_H\ ba + (b * exp\ a) *_H\ e1 + (c * exp\ a) *_H\ e2 + (d * exp\ a + b * c * exp\ a) *_H\ e12$
⟨*proof*⟩

lemma *hypext-exp-extract*:

$(*h* \ exp) \ (\beta \ x) = exp \ x *_H \ ba + exp \ x *_H \ e1 + exp \ x *_H \ e2 + exp \ x *_H \ e12$
⟨*proof*⟩

lemma *hypext-exp-e1* [*simp*]:
$(*h* \ exp) \ (x * e1) = 1 + e1 * x$
⟨*proof*⟩

lemma *hypext-exp-e2* [*simp*]:
$(*h* \ exp) \ (x * e2) = 1 + e2 * x$
⟨*proof*⟩

lemma *hypext-exp-e12* [*simp*]:
$(*h* \ exp) \ (x * e12) = 1 + e12 * x$
⟨*proof*⟩

Extracting the parts for the exponential function extension.

lemma *Eps1-hypext-exp-extract* [*simp*]:
$Eps1 \ ((*h* \ exp) \ (\beta \ x)) = exp \ x$
⟨*proof*⟩

lemma *Eps2-hypext-exp-extract* [*simp*]:
$Eps2 \ ((*h* \ exp) \ (\beta \ x)) = exp \ x$
⟨*proof*⟩

lemma *Eps12-hypext-exp-extract* [*simp*]:
$Eps12 \ ((*h* \ exp) \ (\beta \ x)) = exp \ x$
⟨*proof*⟩

3.1.11 Square Root

Square root function extension.

lemma *hypext-sqrt-Hyperdual-Hyperdual*:
assumes $a > 0$
shows $(*h* \ sqrt) \ (Hyperdual \ a \ b \ c \ d) =$
$Hyperdual \ (sqrt \ a) \ (b * inverse \ (sqrt \ a) \ / \ 2) \ (c * inverse \ (sqrt \ a) \ / \ 2)$
$(d * inverse \ (sqrt \ a) \ / \ 2 - b * c * inverse \ (sqrt \ a \ \hat{} \ 3) \ / \ 4)$
⟨*proof*⟩

lemma *hypext-sqrt-Hyperdual*:
$a > 0 \Longrightarrow (*h* \ sqrt) \ (Hyperdual \ a \ b \ c \ d) =$
$sqrt \ a *_H \ ba + (b * inverse \ (sqrt \ a) \ / \ 2) *_H \ e1 + (c * inverse \ (sqrt \ a) \ / \ 2)$
$*_H \ e2 +$
$(d * inverse \ (sqrt \ a) \ / \ 2 - b * c * inverse \ (sqrt \ a \ \hat{} \ 3) \ / \ 4) *_H \ e12$
⟨*proof*⟩

lemma *hypext-sqrt-extract*:
$x > 0 \Longrightarrow (*h* \ sqrt) \ (\beta \ x) = sqrt \ x *_H \ ba + (inverse \ (sqrt \ x) \ / \ 2) *_H \ e1 +$
$(inverse \ (sqrt \ x) \ / \ 2) *_H \ e2 - (inverse \ (sqrt \ x \ \hat{} \ 3) \ / \ 4) *_H \ e12$
⟨*proof*⟩

Extracting the parts for the square root extension.

lemma *Eps1-hypext-sqrt-extract* [*simp*]:
$x > 0 \Longrightarrow Eps1\ ((*h*\ sqrt)\ (\beta\ x)) = inverse\ (sqrt\ x)\ /\ 2$
⟨*proof*⟩

lemma *Eps2-hypext-sqrt-extract* [*simp*]:
$x > 0 \Longrightarrow Eps2\ ((*h*\ sqrt)\ (\beta\ x)) = inverse\ (sqrt\ x)\ /\ 2$
⟨*proof*⟩

lemma *Eps12-hypext-sqrt-extract* [*simp*]:
$x > 0 \Longrightarrow Eps12\ ((*h*\ sqrt)\ (\beta\ x)) = -\ (inverse\ (sqrt\ x\ \hat{}\ 3)\ /\ 4)$
⟨*proof*⟩

lemma *Base* $x > 0 \Longrightarrow (*h*\ sin)\ x + (*h*\ sqrt)\ x = (*h*\ (\lambda x.\ sin\ x + sqrt\ x))\ x$
⟨*proof*⟩

3.1.12 Natural Power

lemma *hypext-power*:
$(*h*\ (\lambda x.\ x\ \hat{}\ n))\ x = x\ \hat{}\ n$
⟨*proof*⟩

lemma *hypext-fun-power*:
assumes *f twice-field-differentiable-at* (*Base x*)
shows $(*h*\ (\lambda x.\ (f\ x)\ \hat{}\ n))\ x = ((*h*\ f)\ x)\ \hat{}\ n$
⟨*proof*⟩

lemma *hypext-power-Hyperdual*:
$(*h*\ (\lambda x.\ x\ \hat{}\ n))\ (Hyperdual\ a\ b\ c\ d) =$
$a\ \hat{}\ n *_H ba + (of\text{-}nat\ n * b * a\ \hat{}(n - 1)) *_H e1 + (of\text{-}nat\ n * c * a\ \hat{}(n - 1)) *_H e2 +$
$(d * (of\text{-}nat\ n * a\ \hat{}(n - 1)) + b * c * (of\text{-}nat\ n * of\text{-}nat\ (n - 1) * a\ \hat{}(n - 2))) *_H e12$
⟨*proof*⟩

lemma *hypext-power-Hyperdual-parts*:
$(*h*\ (\lambda x.\ x\ \hat{}\ n))\ (a *_H ba + b *_H e1 + c *_H e2 + d *_H e12) =$
$a\ \hat{}\ n *_H ba + (of\text{-}nat\ n * b * a\ \hat{}(n - 1)) *_H e1 + (of\text{-}nat\ n * c * a\ \hat{}(n - 1)) *_H e2 +$
$(d * (of\text{-}nat\ n * a\ \hat{}(n - 1)) + b * c * (of\text{-}nat\ n * of\text{-}nat\ (n - 1) * a\ \hat{}(n - 2))) *_H e12$
⟨*proof*⟩

lemma *hypext-power-extract*:
$(*h*\ (\lambda x.\ x\ \hat{}\ n))\ (\beta\ x) =$
$x\ \hat{}\ n *_H ba + (of\text{-}nat\ n * x\ \hat{}(n - 1)) *_H e1 + (of\text{-}nat\ n * x\ \hat{}(n - 1)) *_H e2 +$
$(of\text{-}nat\ n * of\text{-}nat\ (n - 1) * x\ \hat{}(n - 2)) *_H e12$

⟨proof⟩

lemma *Eps1-hypext-power* [*simp*]:
Eps1 ((∗*h*∗ (λ*x*. *x* ^ *n*)) *x*) = *of-nat n* ∗ *Eps1 x* ∗ (*Base x*) ^ (*n* − *1*)
⟨proof⟩

lemma *Eps2-hypext-power* [*simp*]:
Eps2 ((∗*h*∗ (λ*x*. *x* ^ *n*)) *x*) = *of-nat n* ∗ *Eps2 x* ∗ (*Base x*) ^ (*n* − *1*)
⟨proof⟩

lemma *Eps12-hypext-power* [*simp*]:
Eps12 ((∗*h*∗ (λ*x*. *x* ^ *n*)) *x*) =
Eps12 x ∗ (*of-nat n* ∗ *Base x* ^ (*n* − *1*)) + *Eps1 x* ∗ *Eps2 x* ∗ (*of-nat n* ∗ *of-nat* (*n* − *1*) ∗ *Base x* ^ (*n* − *2*))
⟨proof⟩

3.1.13 Inverse

lemma *hypext-inverse*:
assumes *Base x* ≠ *0*
shows (∗*h*∗ *inverse*) *x* = *inverse x*
⟨proof⟩

lemma *hypext-fun-inverse*:
assumes *f twice-field-differentiable-at* (*Base x*)
and *f* (*Base x*) ≠ *0*
shows (∗*h*∗ (λ*x*. *inverse* (*f x*))) *x* = *inverse* ((∗*h*∗ *f*) *x*)
⟨proof⟩

lemma *hypext-inverse-Hyperdual*:
a ≠ *0* ⟹
(∗*h*∗ *inverse*) (*Hyperdual a b c d*) =
Hyperdual (*inverse a*) (− (*b* / a^2)) (− (*c* / a^2)) (*2* ∗ *b* ∗ *c* / (*a* ^ *3*) − *d* / a^2)
⟨proof⟩

lemma *hypext-inverse-Hyperdual-parts*:
a ≠ *0* ⟹
(∗*h*∗ *inverse*) (*a* $*_H$ *ba* + *b* $*_H$ *e1* + *c* $*_H$ *e2* + *d* $*_H$ *e12*) =
inverse a $*_H$ *ba* + − (*b* / a^2) $*_H$ *e1* + − (*c* / a^2) $*_H$ *e2* + (*2* ∗ *b* ∗ *c* / *a* ^ *3* − *d* / a^2) $*_H$ *e12*
⟨proof⟩

lemma *inverse-Hyperdual-parts*:
(*a*::'*a*::*real-normed-field*) ≠ *0* ⟹
inverse (*a* $*_H$ *ba* + *b* $*_H$ *e1* + *c* $*_H$ *e2* + *d* $*_H$ *e12*) =
inverse a $*_H$ *ba* + − (*b* / a^2) $*_H$ *e1* + − (*c* / a^2) $*_H$ *e2* + (*2* ∗ *b* ∗ *c* / *a* ^ *3* − *d* / a^2) $*_H$ *e12*
⟨proof⟩

lemma *hypext-inverse-extract*:
$x \neq 0 \Longrightarrow (*h* \ inverse) \ (\beta \ x) = inverse \ x *_H \ ba - (1 \ / \ x^2) *_H \ e1 - (1 \ / \ x^2) *_H \ e2 + (2 \ / \ x \ \hat{} \ 3) *_H \ e12$
⟨*proof*⟩

lemma *inverse-extract*:
$x \neq 0 \Longrightarrow inverse \ (\beta \ x) = inverse \ x *_H \ ba - (1 \ / \ x^2) *_H \ e1 - (1 \ / \ x^2) *_H \ e2 + (2 \ / \ x \ \hat{} \ 3) *_H \ e12$
⟨*proof*⟩

lemma *Eps1-hypext-inverse* [*simp*]:
$Base \ x \neq 0 \Longrightarrow Eps1 \ ((*h* \ inverse) \ x) = - \ Eps1 \ x * (1 \ / \ (Base \ x)^2)$
⟨*proof*⟩
lemma *Eps1-inverse* [*simp*]:
$Base \ (x{::}'a{::}real\text{-}normed\text{-}field \ hyperdual) \neq 0 \Longrightarrow Eps1 \ (inverse \ x) = - \ Eps1 \ x * (1 \ / \ (Base \ x)^2)$
⟨*proof*⟩

lemma *Eps2-hypext-inverse* [*simp*]:
$Base \ (x{::}'a{::}real\text{-}normed\text{-}field \ hyperdual) \neq 0 \Longrightarrow Eps2 \ (inverse \ x) = - \ Eps2 \ x * (1 \ / \ (Base \ x)^2)$
⟨*proof*⟩

lemma *Eps12-hypext-inverse* [*simp*]:
$Base \ (x{::}'a{::}real\text{-}normed\text{-}field \ hyperdual) \neq 0$
$\Longrightarrow Eps12 \ (inverse \ x) = Eps1 \ x * Eps2 \ x * (2/ \ (Base \ x \ \hat{} \ 3)) - Eps12 \ x \ / \ (Base \ x)^2$
⟨*proof*⟩

3.1.14 Division

lemma *hypext-fun-divide*:
assumes *f twice-field-differentiable-at* (*Base x*)
and *g twice-field-differentiable-at* (*Base x*)
and $g \ (Base \ x) \neq 0$
shows $(*h* \ (\lambda x. \ f \ x \ / \ g \ x)) \ x = (*h* \ f) \ x \ / \ (*h* \ g) \ x$
⟨*proof*⟩

3.1.15 Polynomial

lemma *hypext-polyn*:
fixes $coef :: nat \Rightarrow 'a :: \{real\text{-}normed\text{-}field\}$
and $n :: nat$
shows $(*h* \ (\lambda x. \ \sum i{<}n. \ coef \ i * x\hat{}i)) \ x = (\sum i{<}n. \ (coef \ i) *_H \ (x\hat{}i))$
⟨*proof*⟩

lemma *hypext-fun-polyn*:
fixes $coef :: nat \Rightarrow 'a :: \{real\text{-}normed\text{-}field\}$
and $n :: nat$
assumes *f twice-field-differentiable-at* (*Base x*)

shows $(*h* \ (\lambda x. \sum i{<}n. \ coef\ i * (f\ x)\widehat{\ }i)) \ x = (\sum i{<}n. \ (coef\ i) *_H (((*h* \ f) \ x)\widehat{\ }i))$
⟨*proof*⟩

end

theory *LogisticFunction*
imports *HyperdualFunctionExtension*
begin

3.2 Logistic Function

Define the standard logistic function and its hyperdual variant:

definition *logistic* :: *real* ⇒ *real*
where *logistic x* = *inverse* (*1* + *exp* (−*x*))
definition *hyp-logistic* :: *real hyperdual* ⇒ *real hyperdual*
where *hyp-logistic x* = *inverse* (*1* + (∗*h*∗ *exp*) (−*x*))

Hyperdual extension of the logistic function is its hyperdual variant:

lemma *hypext-logistic*:
(∗*h*∗ *logistic*) *x* = *hyp-logistic x*
⟨*proof*⟩

From properties of autodiff we know it gives us the derivative:

lemma *Eps1* (*hyp-logistic* (*β x*)) = *deriv logistic x*
⟨*proof*⟩

which is equal to the known derivative of the standard logistic function:

lemma *First* (*autodiff logistic x*) = *exp* (− *x*) / (*1* + *exp* (− *x*)) ^ *2*

⟨*proof*⟩

Similarly we can get the second derivative:

lemma *Second* (*autodiff logistic x*) = *deriv* (*deriv logistic*) *x*
⟨*proof*⟩

and derive its value:

lemma *Second* (*autodiff logistic x*) = ((*exp* (− *x*) − *1*) ∗ *exp* (− *x*)) / ((*1* + *exp* (− *x*)) ^ *3*)

⟨*proof*⟩

end

theory *AnalyticTestFunction*
imports *HyperdualFunctionExtension HOL−Decision-Procs.Approximation*
begin

3.3 Analytic Test Function

We investigate the analytic test function used by Fike and Alonso in their 2011 paper [1] as a relatively non-trivial example. The function is defined as: $\lambda x.\ exp\ x\ /\ sqrt\ ((sin\ x)^3 + (cos\ x)^3)$.

definition *fa-test* :: *real* ⇒ *real*
where *fa-test x* = *exp x* / (*sqrt* (*sin x* ^ *3* + *cos x* ^ *3*))

We define the same composition of functions but using the relevant hyperdual versions. Note that we implicitly use the facts that hyperdual extensions of plus, times and inverse are the same operations on hyperduals.

definition *hyp-fa-test* :: *real hyperdual* ⇒ *real hyperdual*
where *hyp-fa-test x* = ((∗*h*∗ *exp*) *x*) / ((∗*h*∗ *sqrt*) (((∗*h*∗ *sin*) *x*) ^ *3* + ((∗*h*∗ *cos*) *x*) ^ *3*))

We prove lemmas useful to show when this function is well defined.

lemma *sin-cube-plus-cos-cube*:
sin x ^ *3* + *cos x* ^ *3* = *1/2* ∗ (*sin x* + *cos x*) ∗ (*2* − *sin* (*2* ∗ *x*))
for *x* :: ′*a*::{*real-normed-field*,*banach*}
⟨*proof*⟩

lemma *sin-cube-plus-cos-cube-gt-zero-iff*:
(*sin x* ^ *3* + *cos x* ^ *3* > *0*) = (*sin x* + *cos x* > *0*)
for *x* ::*real*
⟨*proof*⟩

lemma *sin-plus-cos-eq-45*:
sin x + *cos x* = *sqrt 2* ∗ *sin* (*x* + *pi*/*4*)
⟨*proof*⟩

lemma *sin-cube-plus-cos-cube-gt-zero-iff* ′:
(*sin x* ^ *3* + *cos x* ^ *3* > *0*) = (*sin* (*x* + *pi*/*4*) > *0*)
⟨*proof*⟩

lemma *sin-less-zero-pi*:
⟦− *pi* < *x*; *x* < *0*⟧ ⟹ *sin x* < *0*
⟨*proof*⟩

lemma *sin-45-positive-intervals*:
(*sin* (*x* + *pi*/*4*) > *0*) = (*x* ∈ (⋃ *n*::*int*. {−*pi*/*4* + *2*∗*pi*∗*n* <..< *3*∗*pi*/*4* + *2*∗*pi*∗*n*}))
⟨*proof*⟩

When the function is well defined our hyperdual definition is equal to the hyperdual extension.

lemma *hypext-fa-test*:
assumes *Base x* ∈ (⋃ *n*::*int*. {−*pi*/*4* + *2*∗*pi*∗*n* <..< *3*∗*pi*/*4* + *2*∗*pi*∗*n*})
shows (∗*h*∗ *fa-test*) *x* = *hyp-fa-test x*
⟨*proof*⟩

We can show that our hyperdual extension gives (approximately) the same values as those found by Fike and Alonso when evaluated at $(15::'a)$ / $(10::'a)$.

lemma
  assumes $x = \textit{hyp-fa-test}\ (\beta\ 1.5)$
  shows $|\textit{Base}\ x - 4.4978| \leq 0.00005$
    and $|\textit{Eps1}\ x - 4.0534| \leq 0.00005$
    and $|\textit{Eps12}\ x - 9.4631| \leq 0.00005$
$\langle proof \rangle$

A number of additional lemmas that will be required to prove the derivatives:

lemma *hypext-sqrt-hyperdual-parts*:
  $a > 0 \Longrightarrow (*h*\ sqrt)\ (a *_H ba + b *_H e1 + c *_H e2 + d *_H e12) =$
  $sqrt\ a *_H ba + (b * inverse\ (sqrt\ a)\ /\ 2) *_H e1 + (c * inverse\ (sqrt\ a)\ /\ 2) *_H e2 +$
  $(d * inverse\ (sqrt\ a)\ /\ 2 - b * c * inverse\ (sqrt\ a\ \hat{}\ 3)\ /\ 4) *_H e12$
  $\langle proof \rangle$

lemma *cos-multiple*: $cos\ (n * x) = 2 * cos\ x * cos\ ((n - 1) * x) - cos\ ((n - 2) * x)$
  for $x :: 'a :: \{banach,real\text{-}normed\text{-}field\}$
$\langle proof \rangle$

lemma *sin-multiple*: $sin\ (n * x) = 2 * cos\ x * sin\ ((n - 1) * x) - sin\ ((n - 2) * x)$
  for $x :: 'a :: \{banach,real\text{-}normed\text{-}field\}$
$\langle proof \rangle$

lemma *power5*:
  fixes $z :: 'a :: monoid\text{-}mult$
  shows $z\ \hat{}\ 5 = z * z * z * z * z$
  $\langle proof \rangle$

lemma *power6*:
  fixes $z :: 'a :: monoid\text{-}mult$
  shows $z\ \hat{}\ 6 = z * z * z * z * z * z$
  $\langle proof \rangle$

We find the derivatives of *fa-test* by applying a Wengert list approach, as done by Fike and Alonso, to make the same composition but in hyperduals. We know that this is equal to the hyperdual extension which in turn gives us the derivatives.

lemma *Wengert-autodiff-fa-test*:
  assumes $x \in (\bigcup n::int.\ \{-pi/4 + 2*pi*n <..< 3*pi/4 + 2*pi*n\})$
  shows $First\ (autodiff\ fa\text{-}test\ x) =$
    $(exp\ x * (3 * cos\ x + 5 * cos\ (3 * x) + 9 * sin\ x + sin\ (3 * x)))\ /$
      $(8 * (sqrt\ (sin\ x\ \hat{}\ 3 + cos\ x\ \hat{}\ 3))\ \hat{}\ 3)$
    and $Second\ (autodiff\ fa\text{-}test\ x) =$

$(exp\ x * (130 - 12 * cos\ (2 * x) +$
$30 * cos\ (4 * x) + 12 * cos\ (6 * x) -$
$111 * sin\ (2 * x) +$
$48 * sin\ (4 * x) + 5 * sin\ (6 * x))) \ /$
$(64 * (sqrt\ (sin\ x \hat{}\ 3 + cos\ x \hat{}\ 3)) \hat{}\ 5)$

$\langle proof \rangle$

end

References

[1] J. A. Fike and J. J. Alonso. The development of hyper-dual numbers for exact second-derivative calculations. In *AIAA paper 2011-886, 49th AIAA Aerospace Sciences Meeting*, 2011.