Matrices for ODEs

Jonathan Julián Huerta y Munive

March 17, 2025

Abstract

Our theories formalise various matrix properties that serve to establish existence, uniqueness and characterisation of the solution to affine systems of ordinary differential equations (ODEs). In particular, we formalise the operator and maximum norm of matrices. Then we use them to prove that square matrices form a Banach space, and in this setting, we show an instance of Picard-Lindelöf's theorem for affine systems of ODEs. Finally, we apply this formalisation by verifying three simple hybrid programs.

Contents

1 Introductory Remarks

Affine systems of ordinary differential equations (ODEs) are those whose associated vector fields are linear transformations. That is, if there is a matrix-valued function $A : \mathbb{R} \to M_{n\times n}(\mathbb{R})$ and vector function $B : \mathbb{R} \to \mathbb{R}^n$ such that the system of ODEs $x'\,t = f\,(t, x\,t)$ can be rewritten as $x'\,t = A\cdot(x\,t)+B\,t$, then the system is affine. Similarly, the associated linear system of ODEs is $x'\,t = A \cdot (x\,t)$ for matrix-vector multiplication $\cdot$. Our theories formalise affine (hence linear) systems of ordinary differential equations. For this purpose, we extend the ODE libraries of [6] and linear algebra in HOL-Analysis. We add to them various results about invertibility of matrices, their diagonalisation, their operator and maximum norms, and properties relating them with vectors. We also define a new type of square matrices and prove that this is a Banach space. Then we obtain results about derivatives of matrix-vector multiplication and use them to prove Picard-Lindelöf's theorem as formalised in [3]. The Banach space instance allows us to characterise the general solution to affine systems of ODEs in terms of the matrix-exponential. Finally, we use the components of [3] to do three simple verification examples in the style of differential dynamic logic [7] as showcased in [1, 2, 5]. The paper [4] has a detailed overview of the various contributions that this formalisation adds to the verification components.

2 Mathematical Preliminaries

This section adds useful syntax, abbreviations and theorems to the Isabelle distribution.

theory *MTX-Preliminaries*

imports *Hybrid-Systems-VCs.HS-Preliminaries*

begin

2.1 Syntax

abbreviation e $k \equiv$ *axis k 1*

syntax
-ivl-integral :: *real* ⇒ *real* ⇒ *'a* ⇒ *pttrn* ⇒ *bool* (‹(3∫ -- (-)∂/-)› [0, 0, 10] 10)

syntax-consts
-ivl-integral ⇌ *ivl-integral*

translations
$\int_a{}^b f \; \partial x$ ⇌ *CONST ivl-integral a b* ($\lambda x.\ f$)

notation *matrix-inv* (‹-$^{-1}$› [90])

abbreviation *entries* (*A*::*'a*^*'n*^*'m*) ≡ {*A* $ *i* $ *j* | *i j*. *i* ∈ *UNIV* ∧ *j* ∈ *UNIV*}

2.2 Topology and sets

lemmas *compact-imp-bdd-above* = *compact-imp-bounded*[*THEN bounded-imp-bdd-above*]

lemma *comp-cont-image-spec*: *continuous-on T f* ⟹ *compact T* ⟹ *compact* {*f t* |*t*. *t* ∈ *T*}
⟨*proof*⟩

lemmas *bdd-above-cont-comp-spec* = *compact-imp-bdd-above*[*OF comp-cont-image-spec*]

lemmas *bdd-above-norm-cont-comp* = *continuous-on-norm*[*THEN bdd-above-cont-comp-spec*]

lemma *open-cballE*: $t_0 \in T$ ⟹ *open T* ⟹ ∃ *e*>*0*. *cball* t_0 *e* ⊆ *T*
⟨*proof*⟩

lemma *open-ballE*: $t_0 \in T$ ⟹ *open T* ⟹ ∃ *e*>*0*. *ball* t_0 *e* ⊆ *T*
⟨*proof*⟩

lemma *funcset-UNIV*: *f* ∈ *A* → *UNIV*
⟨*proof*⟩

lemma *finite-image-of-finite*[*simp*]:
fixes *f*::*'a*::*finite* ⇒ *'b*
shows *finite* {*x*. ∃ *i*. *x* = *f i*}
⟨*proof*⟩

lemma *finite-image-of-finite2*:
fixes *f* :: *'a*::*finite* ⇒ *'b*::*finite* ⇒ *'c*
shows *finite* {*f x y* |*x y*. *P x y*}

⟨*proof*⟩

2.3 Functions

lemma *finite-sum-univ-singleton*: $(sum\ g\ UNIV) = sum\ g\ \{i{::}'a{::}finite\} + sum\ g\ (UNIV - \{i\})$
  ⟨*proof*⟩

lemma *suminfI*:
  fixes $f :: nat \Rightarrow 'a{::}\{t2\text{-}space,comm\text{-}monoid\text{-}add\}$
  shows $f\ sums\ k \Longrightarrow suminf\ f = k$
  ⟨*proof*⟩

lemma *suminf-eq-sum*:
  fixes $f :: nat \Rightarrow ('a{::}real\text{-}normed\text{-}vector)$
  assumes $\bigwedge n.\ n > m \Longrightarrow f\ n = 0$
  shows $(\sum n.\ f\ n) = (\sum n \leq m.\ f\ n)$
  ⟨*proof*⟩

lemma *suminf-multr*: $summable\ f \Longrightarrow (\sum n.\ f\ n * c) = (\sum n.\ f\ n) * c$ **for** $c{::}'a{::}real\text{-}normed\text{-}algebra$
  ⟨*proof*⟩

lemma *sum-if-then-else-simps*[*simp*]:
  fixes $q :: ('a{::}semiring\text{-}0)$ **and** $i :: 'n{::}finite$
  shows $(\sum j{\in}UNIV.\ f\ j * (if\ j = i\ then\ q\ else\ 0)) = f\ i * q$
    and $(\sum j{\in}UNIV.\ f\ j * (if\ i = j\ then\ q\ else\ 0)) = f\ i * q$
    and $(\sum j{\in}UNIV.\ (if\ i = j\ then\ q\ else\ 0) * f\ j) = q * f\ i$
    and $(\sum j{\in}UNIV.\ (if\ j = i\ then\ q\ else\ 0) * f\ j) = q * f\ i$
  ⟨*proof*⟩

2.4 Suprema

lemma *le-max-image-of-finite*[*simp*]:
  fixes $f{::}'a{::}finite \Rightarrow 'b{::}linorder$
  shows $(f\ i) \leq Max\ \{x.\ \exists\, i.\ x = f\ i\}$
  ⟨*proof*⟩

lemma *cSup-eq*:
  fixes $c{::}'a{::}conditionally\text{-}complete\text{-}lattice$
  assumes $\forall x \in X.\ x \leq c$ **and** $\exists x \in X.\ c \leq x$
  shows $Sup\ X = c$
  ⟨*proof*⟩

lemma *cSup-mem-eq*:
  $c \in X \Longrightarrow \forall x \in X.\ x \leq c \Longrightarrow Sup\ X = c$ **for** $c{::}'a{::}conditionally\text{-}complete\text{-}lattice$
  ⟨*proof*⟩

lemma *cSup-finite-ex*:

finite $X \Longrightarrow X \neq \{\} \Longrightarrow \exists\, x{\in}X.\ \mathit{Sup}\ X = x$ **for** $X{::}'a{::}\mathit{conditionally\text{-}complete\text{-}linorder}$ *set*
⟨*proof*⟩

lemma *cMax-finite-ex*:
finite $X \Longrightarrow X \neq \{\} \Longrightarrow \exists\, x{\in}X.\ \mathit{Max}\ X = x$ **for** $X{::}'a{::}\mathit{conditionally\text{-}complete\text{-}linorder}$ *set*
⟨*proof*⟩

lemma *finite-nat-minimal-witness*:
fixes $P :: ('a{::}\mathit{finite}) \Rightarrow \mathit{nat} \Rightarrow \mathit{bool}$
assumes $\forall\, i.\ \exists\, N{::}\mathit{nat}.\ \forall\, n \geq N.\ P\ i\ n$
shows $\exists\, N.\ \forall\, i.\ \forall\, n \geq N.\ P\ i\ n$
⟨*proof*⟩

2.5 Real numbers

named-theorems *field-power-simps simplification rules for powers to the nth*

declare *semiring-normalization-rules*(*18*) [*field-power-simps*]
and *semiring-normalization-rules*(*26*) [*field-power-simps*]
and *semiring-normalization-rules*(*27*) [*field-power-simps*]
and *semiring-normalization-rules*(*28*) [*field-power-simps*]
and *semiring-normalization-rules*(*29*) [*field-power-simps*]

WARNING: Adding $?x * ?x^{?q} = ?x^{\mathit{Suc}\ ?q}$ to our tactic makes its combination with simp to loop infinitely in some proofs.

lemma *sq-le-cancel*:
shows $(a{::}\mathit{real}) \geq 0 \Longrightarrow b \geq 0 \Longrightarrow a\hat{\ }2 \leq b * a \Longrightarrow a \leq b$
and $(a{::}\mathit{real}) \geq 0 \Longrightarrow b \geq 0 \Longrightarrow a\hat{\ }2 \leq a * b \Longrightarrow a \leq b$
⟨*proof*⟩

lemma *frac-diff-eq1*: $a \neq b \Longrightarrow a\ /\ (a - b) - b\ /\ (a - b) = 1$ **for** $a{::}\mathit{real}$
⟨*proof*⟩

lemma *exp-add*: $x * y - y * x = 0 \Longrightarrow \mathit{exp}\ (x + y) = \mathit{exp}\ x * \mathit{exp}\ y$
⟨*proof*⟩

lemmas *mult-exp-exp* = *exp-add*[*symmetric*]

2.6 Vectors and matrices

lemma *sum-axis*[*simp*]:
fixes $q :: ('a{::}\mathit{semiring\text{-}0})$
shows $(\sum j{\in}\mathit{UNIV}.\ f\ j * \mathit{axis}\ i\ q\ \$\ j) = f\ i * q$
and $(\sum j{\in}\mathit{UNIV}.\ \mathit{axis}\ i\ q\ \$\ j * f\ j) = q * f\ i$
⟨*proof*⟩

lemma *sum-scalar-nth-axis*: $\mathit{sum}\ (\lambda i.\ (x\ \$\ i) *s\ \mathrm{e}\ i)\ \mathit{UNIV} = x$ **for** $x :: ('a{::}\mathit{semiring\text{-}1})\hat{\ }'n$

⟨*proof*⟩

lemma *scalar-eq-scaleR*[*simp*]: $c *s\ x = c *_R x$
⟨*proof*⟩

lemma *matrix-add-rdistrib*: $((B + C) ** A) = (B ** A) + (C ** A)$
⟨*proof*⟩

lemma *vec-mult-inner*: $(A *v\ v) \cdot w = v \cdot (transpose\ A *v\ w)$ **for** $A :: real\ \hat{}'n\hat{}'n$
⟨*proof*⟩

lemma *uminus-axis-eq*[*simp*]: $-\ axis\ i\ k = axis\ i\ (-k)$ **for** $k :: 'a::ring$
⟨*proof*⟩

lemma *norm-axis-eq*[*simp*]: $\|axis\ i\ k\| = \|k\|$
⟨*proof*⟩

lemma *matrix-axis-0*:
fixes $A :: ('a::idom)\hat{}'n\hat{}'m$
assumes $k \neq 0$ **and** $h{:}\forall\, i.\ (A *v\ (axis\ i\ k)) = 0$
shows $A = 0$
⟨*proof*⟩

lemma *scaleR-norm-sgn-eq*: $(\|x\|) *_R sgn\ x = x$
⟨*proof*⟩

lemma *vector-scaleR-commute*: $A *v\ c *_R x = c *_R (A *v\ x)$ **for** $x :: ('a::real\text{-}normed\text{-}algebra\text{-}1)\hat{}'n$
⟨*proof*⟩

lemma *scaleR-vector-assoc*: $c *_R (A *v\ x) = (c *_R A) *v\ x$ **for** $x :: ('a::real\text{-}normed\text{-}algebra\text{-}1)\hat{}'n$
⟨*proof*⟩

lemma *mult-norm-matrix-sgn-eq*:
fixes $x :: ('a::real\text{-}normed\text{-}algebra\text{-}1)\hat{}'n$
shows $(\|A *v\ sgn\ x\|) * (\|x\|) = \|A *v\ x\|$
⟨*proof*⟩

2.7 Diagonalization

lemma *invertibleI*: $A ** B = mat\ 1 \Longrightarrow B ** A = mat\ 1 \Longrightarrow invertible\ A$
⟨*proof*⟩

lemma *invertibleD*[*simp*]:
assumes *invertible A*
shows $A^{-1} ** A = mat\ 1$ **and** $A ** A^{-1} = mat\ 1$
⟨*proof*⟩

lemma *matrix-inv-unique*:
assumes $A ** B = mat\ 1$ **and** $B ** A = mat\ 1$

shows $A^{-1} = B$
⟨*proof*⟩

lemma *invertible-matrix-inv*: *invertible* $A \Longrightarrow$ *invertible* (A^{-1})
⟨*proof*⟩

lemma *matrix-inv-idempotent*[*simp*]: *invertible* $A \Longrightarrow A^{-1-1} = A$
⟨*proof*⟩

lemma *matrix-inv-matrix-mul*:
assumes *invertible* A **and** *invertible* B
shows $(A ** B)^{-1} = B^{-1} ** A^{-1}$
⟨*proof*⟩

lemma *mat-inverse-simps*[*simp*]:
fixes c :: *'a::division-ring*
assumes $c \neq 0$
shows *mat* (*inverse* c) $**$ *mat* c = *mat 1*
and *mat* c $**$ *mat* (*inverse* c) = *mat 1*
⟨*proof*⟩

lemma *matrix-inv-mat*[*simp*]: $c \neq 0 \Longrightarrow$ $(mat\ c)^{-1}$ = *mat* (*inverse* c) **for** c :: *'a::division-ring*
⟨*proof*⟩

lemma *invertible-mat*[*simp*]: $c \neq 0 \Longrightarrow$ *invertible* (*mat* c) **for** c :: *'a::division-ring*
⟨*proof*⟩

lemma *matrix-inv-mat-1*: $(mat\ (1::'a::division\text{-}ring))^{-1}$ = *mat 1*
⟨*proof*⟩

lemma *invertible-mat-1*: *invertible* (*mat* (*1::'a::division-ring*))
⟨*proof*⟩

definition *similar-matrix* :: $('a::semiring\text{-}1)^\frown{}'m^\frown{}'m \Rightarrow ('a::semiring\text{-}1)^\frown{}'n^\frown{}'n \Rightarrow$ *bool* (**infixr** ‹∼› *25*)
where *similar-matrix* A $B \longleftrightarrow (\exists\ P.\ invertible\ P \wedge A = P^{-1} ** B ** P)$

lemma *similar-matrix-refl*[*simp*]: $A \sim A$ **for** A :: $'a::division\text{-}ring^\frown{}'n^\frown{}'n$
⟨*proof*⟩

lemma *similar-matrix-simm*: $A \sim B \Longrightarrow B \sim A$ **for** A B :: $('a::semiring\text{-}1)^\frown{}'n^\frown{}'n$
⟨*proof*⟩

lemma *similar-matrix-trans*: $A \sim B \Longrightarrow B \sim C \Longrightarrow A \sim C$ **for** A B C :: $('a::semiring\text{-}1)^\frown{}'n^\frown{}'n$
⟨*proof*⟩

lemma *mat-vec-nth-simps*[*simp*]:

$i = j \Longrightarrow mat\ c\ \$\ i\ \$\ j = c$
$i \neq j \Longrightarrow mat\ c\ \$\ i\ \$\ j = 0$
⟨*proof*⟩

definition $diag\text{-}mat\ f = (\chi\ i\ j.\ if\ i = j\ then\ f\ i\ else\ 0)$

lemma *diag-mat-vec-nth-simps*[*simp*]:
$i = j \Longrightarrow diag\text{-}mat\ f\ \$\ i\ \$\ j = f\ i$
$i \neq j \Longrightarrow diag\text{-}mat\ f\ \$\ i\ \$\ j = 0$
⟨*proof*⟩

lemma *diag-mat-const-eq*[*simp*]: $diag\text{-}mat\ (\lambda i.\ c) = mat\ c$
⟨*proof*⟩

lemma *matrix-vector-mul-diag-mat*: $diag\text{-}mat\ f\ *v\ s = (\chi\ i.\ f\ i * s\$i)$
⟨*proof*⟩

lemma *matrix-vector-mul-diag-axis*[*simp*]: $diag\text{-}mat\ f\ *v\ (axis\ i\ k) = axis\ i\ (f\ i * k)$
⟨*proof*⟩

lemma *matrix-mul-diag-matl*: $diag\text{-}mat\ f ** A = (\chi\ i\ j.\ f\ i * A\$i\$j)$
⟨*proof*⟩

lemma *matrix-matrix-mul-diag-matr*: $A ** diag\text{-}mat\ f = (\chi\ i\ j.\ A\$i\$j * f\ j)$
⟨*proof*⟩

lemma *matrix-mul-diag-diag*: $diag\text{-}mat\ f ** diag\text{-}mat\ g = diag\text{-}mat\ (\lambda i.\ f\ i * g\ i)$
⟨*proof*⟩

lemma *compow-matrix-mul-diag-mat-eq*: $((**)\ (diag\text{-}mat\ f)\ \hat{}\hat{}\ n)\ (mat\ 1) = diag\text{-}mat\ (\lambda i.\ f\ i\hat{}n)$
⟨*proof*⟩

lemma *compow-similar-diag-mat-eq*:
assumes $invertible\ P$
and $A = P^{-1} ** (diag\text{-}mat\ f) ** P$
shows $((**)\ A\ \hat{}\hat{}\ n)\ (mat\ 1) = P^{-1} ** (diag\text{-}mat\ (\lambda i.\ f\ i\hat{}n)) ** P$
⟨*proof*⟩

lemma *compow-similar-diag-mat*:
assumes $A \sim (diag\text{-}mat\ f)$
shows $((**)\ A\ \hat{}\hat{}\ n)\ (mat\ 1) \sim diag\text{-}mat\ (\lambda i.\ f\ i\hat{}n)$
⟨*proof*⟩

no-notation *matrix-inv* (‹-$^{-1}$› [*90*])
and *similar-matrix* (**infixr** ‹∼› *25*)

end

3 Matrix norms

Here, we explore some properties about the operator and the maximum norms for matrices.

theory *MTX-Norms*
imports *MTX-Preliminaries*

begin

3.1 Matrix operator norm

abbreviation *op-norm* :: $('a::real\text{-}normed\text{-}algebra\text{-}1)^{\frown}'n^{\frown}'m \Rightarrow real$ (‹(1‖-‖$_{op}$)› [65] 61)
where $\|A\|_{op} \equiv onorm\ (\lambda x.\ A *v\ x)$

lemma *norm-matrix-bound*:
fixes $A :: ('a::real\text{-}normed\text{-}algebra\text{-}1)^{\frown}'n^{\frown}'m$
shows $\|x\| = 1 \Longrightarrow \|A *v\ x\| \leq \|(\chi\ i\ j.\ \|A\ \$\ i\ \$\ j\|) *v\ 1\|$
⟨*proof*⟩

lemma *onorm-set-proptys*:
fixes $A :: ('a::real\text{-}normed\text{-}algebra\text{-}1)^{\frown}'n^{\frown}'m$
shows *bounded* $(range\ (\lambda x.\ (\|A *v\ x\|) / (\|x\|)))$
and *bdd-above* $(range\ (\lambda x.\ (\|A *v\ x\|) / (\|x\|)))$
and $(range\ (\lambda x.\ (\|A *v\ x\|) / (\|x\|))) \neq \{\}$
⟨*proof*⟩

lemma *op-norm-set-proptys*:
fixes $A :: ('a::real\text{-}normed\text{-}algebra\text{-}1)^{\frown}'n^{\frown}'m$
shows *bounded* $\{\|A *v\ x\| \mid x.\ \|x\| = 1\}$
and *bdd-above* $\{\|A *v\ x\| \mid x.\ \|x\| = 1\}$
and $\{\|A *v\ x\| \mid x.\ \|x\| = 1\} \neq \{\}$
⟨*proof*⟩

lemma *op-norm-def*: $\|A\|_{op} = Sup\ \{\|A *v\ x\| \mid x.\ \|x\| = 1\}$
⟨*proof*⟩

lemma *norm-matrix-le-op-norm*: $\|x\| = 1 \Longrightarrow \|A *v\ x\| \leq \|A\|_{op}$
⟨*proof*⟩

lemma *op-norm-ge-0*: $0 \leq \|A\|_{op}$
⟨*proof*⟩

lemma *norm-sgn-le-op-norm*: $\|A *v\ sgn\ x\| \leq \|A\|_{op}$
⟨*proof*⟩

lemma *norm-matrix-le-mult-op-norm*: $\|A *v\ x\| \leq (\|A\|_{op}) * (\|x\|)$
⟨*proof*⟩

lemma *blin-matrix-vector-mult*: *bounded-linear* $((*v)\ A)$ **for** A :: $('a::real\text{-}normed\text{-}algebra\text{-}1)^\frown'n^\frown'm$
⟨*proof*⟩

lemma *op-norm-eq-0*: $(\|A\|_{op} = 0) = (A = 0)$ **for** A :: $('a::real\text{-}normed\text{-}field)^\frown'n^\frown'm$
⟨*proof*⟩

lemma *op-norm0*: $\|(0::('a::real\text{-}normed\text{-}field)^\frown'n^\frown'm)\|_{op} = 0$
⟨*proof*⟩

lemma *op-norm-triangle*: $\|A + B\|_{op} \leq (\|A\|_{op}) + (\|B\|_{op})$
⟨*proof*⟩

lemma *op-norm-scaleR*: $\|c *_R A\|_{op} = |c| * (\|A\|_{op})$
⟨*proof*⟩

lemma *op-norm-matrix-matrix-mult-le*: $\|A ** B\|_{op} \leq (\|A\|_{op}) * (\|B\|_{op})$
⟨*proof*⟩

lemma *norm-matrix-vec-mult-le-transpose*:
$\|x\| = 1 \Longrightarrow (\|A *v\ x\|) \leq sqrt\ (\|transpose\ A ** A\|_{op}) * (\|x\|)$ **for** A :: $real^\frown'n^\frown'n$
⟨*proof*⟩

lemma *op-norm-le-sum-column*: $\|A\|_{op} \leq (\sum i{\in}UNIV.\ \|column\ i\ A\|)$ **for** A :: $real^\frown'n^\frown'm$
⟨*proof*⟩

lemma *op-norm-le-transpose*: $\|A\|_{op} \leq \|transpose\ A\|_{op}$ **for** A :: $real^\frown'n^\frown'n$
⟨*proof*⟩

3.2 Matrix maximum norm

abbreviation *max-norm* :: $real^\frown'n^\frown'm \Rightarrow real$ (‹$(1\|\text{-}\|_{max})$› [65] 61)
where $\|A\|_{max} \equiv Max\ (abs\ `\ (entries\ A))$

lemma *max-norm-def*: $\|A\|_{max} = Max\ \{|A\ \$\ i\ \$\ j| |i\ j.\ i{\in}UNIV \wedge j{\in}UNIV\}$
⟨*proof*⟩

lemma *max-norm-set-proptys*: *finite* $\{|A\ \$\ i\ \$\ j|\ |i\ j.\ i \in UNIV \wedge j \in UNIV\}$ (**is** *finite ?X*)
⟨*proof*⟩

lemma *max-norm-ge-0*: $0 \leq \|A\|_{max}$
⟨*proof*⟩

lemma *op-norm-le-max-norm*:
fixes A :: $real^\frown('n::finite)^\frown('m::finite)$

shows $\|A\|_{op} \leq$ *real CARD('m)* $*$ *real CARD('n)* $*$ $(\|A\|_{max})$
⟨*proof*⟩

lemma *sqrt-Sup-power2-eq-Sup-abs*:
finite A $\Longrightarrow A \neq \{\} \Longrightarrow$ *sqrt* (*Sup* $\{(f\ i)^2\ |i.\ i \in A\}$) = *Sup* $\{|f\ i|\ |i.\ i \in A\}$
⟨*proof*⟩

lemma *sqrt-Max-power2-eq-max-abs*:
finite A $\Longrightarrow A \neq \{\} \Longrightarrow$ *sqrt* (*Max* $\{(f\ i)^2|i.\ i \in A\}$) = *Max* $\{|f\ i|\ |i.\ i \in A\}$
⟨*proof*⟩

lemma *op-norm-diag-mat-eq*: $\|$*diag-mat f*$\|_{op}$ = *Max* $\{|f\ i|\ |i.\ i \in UNIV\}$ (**is** - = *Max ?A*)
⟨*proof*⟩

lemma *op-max-norms-eq-at-diag*: $\|$*diag-mat f*$\|_{op}$ = $\|$*diag-mat f*$\|_{max}$
⟨*proof*⟩

end

4 Square Matrices

The general solution for affine systems of ODEs involves the exponential function. Unfortunately, this operation is only available in Isabelle for the type class "banach". Hence, we define a type of square matrices and prove that it is an instance of this class.

theory *SQ-MTX*
imports *MTX-Norms*

begin

4.1 Definition

typedef *'m sq-mtx* = *UNIV*::(*real*^*'m*^*'m*) *set*
morphisms *to-vec to-mtx* ⟨*proof*⟩

declare *to-mtx-inverse* [*simp*]
and *to-vec-inverse* [*simp*]

setup-lifting *type-definition-sq-mtx*

lift-definition *sq-mtx-ith* :: *'m sq-mtx* ⇒ *'m* ⇒ (*real*^*'m*) (**infixl** ‹$$› *90*) **is** ($)
⟨*proof*⟩

lift-definition *sq-mtx-vec-mult* :: *'m sq-mtx* ⇒ (*real*^*'m*) ⇒ (*real*^*'m*) (**infixl** ‹$*_V$› *90*) **is** ($*v$) ⟨*proof*⟩

lift-definition *vec-sq-mtx-prod* :: $(real^\frown{}'m) \Rightarrow {}'m$ *sq-mtx* $\Rightarrow (real^\frown{}'m)$ **is** $(v*)$
⟨*proof*⟩

lift-definition *sq-mtx-diag* :: $(('m::finite) \Rightarrow real) \Rightarrow ('m::finite)$ *sq-mtx* (**binder** ‹diag › *10*)
  is *diag-mat* ⟨*proof*⟩

lift-definition *sq-mtx-transpose* :: $('m::finite)$ *sq-mtx* $\Rightarrow {}'m$ *sq-mtx* (‹-†›) **is** *transpose* ⟨*proof*⟩

lift-definition *sq-mtx-inv* :: $('m::finite)$ *sq-mtx* $\Rightarrow {}'m$ *sq-mtx* (‹-$^{-1}$› [*90*]) **is** *matrix-inv* ⟨*proof*⟩

lift-definition *sq-mtx-row* :: $'m \Rightarrow ('m::finite)$ *sq-mtx* $\Rightarrow real^\frown{}'m$ (‹row›) **is** *row*
⟨*proof*⟩

lift-definition *sq-mtx-col* :: $'m \Rightarrow ('m::finite)$ *sq-mtx* $\Rightarrow real^\frown{}'m$ (‹col›) **is** *column*
⟨*proof*⟩

lemma *to-vec-eq-ith*: (*to-vec A*) \$ i = A \$\$ i
  ⟨*proof*⟩

lemma *to-mtx-ith*[*simp*]:
  (*to-mtx A*) \$\$ *i1* = *A* \$ *i1*
  (*to-mtx A*) \$\$ *i1* \$ *i2* = *A* \$ *i1* \$ *i2*
  ⟨*proof*⟩

lemma *to-mtx-vec-lambda-ith*[*simp*]: *to-mtx* (χ *i j. x i j*) \$\$ *i1* \$ *i2* = *x i1 i2*
  ⟨*proof*⟩

lemma *sq-mtx-eq-iff*:
  shows $A = B = (\forall\, i\, j.$ *A* \$\$ *i* \$ *j* = *B* \$\$ *i* \$ *j*)
    and $A = B = (\forall\, i.$ *A* \$\$ *i* = *B* \$\$ *i*)
  ⟨*proof*⟩

lemma *sq-mtx-diag-simps*[*simp*]:
  $i = j \Longrightarrow$ *sq-mtx-diag f* \$\$ *i* \$ *j* = *f i*
  $i \neq j \Longrightarrow$ *sq-mtx-diag f* \$\$ *i* \$ *j* = *0*
  sq-mtx-diag f \$\$ *i* = *axis i* (*f i*)
  ⟨*proof*⟩

lemma *sq-mtx-diag-vec-mult*: (diag *i. f i*) $*_V$ *s* = (χ *i. f i* $*$ *s*\$*i*)
  ⟨*proof*⟩

lemma *sq-mtx-vec-mult-diag-axis*: (diag *i. f i*) $*_V$ (*axis i k*) = *axis i* (*f i* $*$ *k*)
  ⟨*proof*⟩

lemma *sq-mtx-vec-mult-eq*: *m* $*_V$ *x* = (χ *i. sum* (λ*j.* (*m* \$\$ *i* \$ *j*) $*$ (*x* \$ *j*)) *UNIV*)
  ⟨*proof*⟩

lemma *sq-mtx-transpose-transpose*[*simp*]: $(A^\dagger)^\dagger = A$
⟨*proof*⟩

lemma *transpose-mult-vec-canon-row*[*simp*]: $(A^\dagger) *_V$ (e i) = row i A
⟨*proof*⟩

lemma *row-ith*[*simp*]: row i A = A $$ i
⟨*proof*⟩

lemma *mtx-vec-mult-canon*: $A *_V$ (e i) = col i A
⟨*proof*⟩

4.2 Ring of square matrices

instantiation *sq-mtx* :: (*finite*) *ring*
begin

lift-definition *plus-sq-mtx* :: *'a sq-mtx* ⇒ *'a sq-mtx* ⇒ *'a sq-mtx* **is** (+) ⟨*proof*⟩

lift-definition *zero-sq-mtx* :: *'a sq-mtx* **is** *0* ⟨*proof*⟩

lift-definition *uminus-sq-mtx* :: *'a sq-mtx* ⇒ *'a sq-mtx* **is** *uminus* ⟨*proof*⟩

lift-definition *minus-sq-mtx* :: *'a sq-mtx* ⇒ *'a sq-mtx* ⇒ *'a sq-mtx* **is** (−) ⟨*proof*⟩

lift-definition *times-sq-mtx* :: *'a sq-mtx* ⇒ *'a sq-mtx* ⇒ *'a sq-mtx* **is** (∗∗) ⟨*proof*⟩

declare *plus-sq-mtx.rep-eq* [*simp*]
  and *minus-sq-mtx.rep-eq* [*simp*]

instance ⟨*proof*⟩

end

lemma *sq-mtx-zero-ith*[*simp*]: *0* $$ i = *0*
⟨*proof*⟩

lemma *sq-mtx-zero-nth*[*simp*]: *0* $$ i $ j = *0*
⟨*proof*⟩

lemma *sq-mtx-plus-eq*: $A + B$ = *to-mtx* (χ i j. $A$$$i$$j$ + $B$$$i$$j$)
⟨*proof*⟩

lemma *sq-mtx-plus-ith*[*simp*]:($A + B$) $$ i = A $$ i + B $$ i
⟨*proof*⟩

lemma *sq-mtx-uminus-eq*: − A = *to-mtx* (χ i j. − $A$$$i$$j$)
⟨*proof*⟩

lemma *sq-mtx-minus-eq*: $A - B = \textit{to-mtx}\ (\chi\ i\ j.\ A\$\$i\$j - B\$\$i\$j)$
⟨*proof*⟩

lemma *sq-mtx-minus-ith*[*simp*]:$(A - B)\ \$\$\ i = A\ \$\$\ i - B\ \$\$\ i$
⟨*proof*⟩

lemma *sq-mtx-times-eq*: $A * B = \textit{to-mtx}\ (\chi\ i\ j.\ \textit{sum}\ (\lambda k.\ A\$\$i\$k * B\$\$k\$j)\ \textit{UNIV})$
⟨*proof*⟩

lemma *sq-mtx-plus-diag-diag*[*simp*]: $\textit{sq-mtx-diag}\ f + \textit{sq-mtx-diag}\ g = (\text{diag}\ i.\ f\ i + g\ i)$
⟨*proof*⟩

lemma *sq-mtx-minus-diag-diag*[*simp*]: $\textit{sq-mtx-diag}\ f - \textit{sq-mtx-diag}\ g = (\text{diag}\ i.\ f\ i - g\ i)$
⟨*proof*⟩

lemma *sum-sq-mtx-diag*[*simp*]: $(\sum n{<}m.\ \textit{sq-mtx-diag}\ (g\ n)) = (\text{diag}\ i.\ \sum n{<}m.\ (g\ n\ i))$ **for** $m{::}\textit{nat}$
⟨*proof*⟩

lemma *sq-mtx-mult-diag-diag*[*simp*]: $\textit{sq-mtx-diag}\ f * \textit{sq-mtx-diag}\ g = (\text{diag}\ i.\ f\ i * g\ i)$
⟨*proof*⟩

lemma *sq-mtx-mult-diagl*: $(\text{diag}\ i.\ f\ i) * A = \textit{to-mtx}\ (\chi\ i\ j.\ f\ i * A\ \$\$\ i\ \$\ j)$
⟨*proof*⟩

lemma *sq-mtx-mult-diagr*: $A * (\text{diag}\ i.\ f\ i) = \textit{to-mtx}\ (\chi\ i\ j.\ A\ \$\$\ i\ \$\ j * f\ j)$
⟨*proof*⟩

lemma *mtx-vec-mult-0l*[*simp*]: $\mathit{0} *_V x = \mathit{0}$
⟨*proof*⟩

lemma *mtx-vec-mult-0r*[*simp*]: $A *_V \mathit{0} = \mathit{0}$
⟨*proof*⟩

lemma *mtx-vec-mult-add-rdistr*: $(A + B) *_V x = A *_V x + B *_V x$
⟨*proof*⟩

lemma *mtx-vec-mult-add-rdistl*: $A *_V (x + y) = A *_V x + A *_V y$
⟨*proof*⟩

lemma *mtx-vec-mult-minus-rdistrib*: $(A - B) *_V x = A *_V x - B *_V x$
⟨*proof*⟩

lemma *mtx-vec-mult-minus-ldistrib*: $A *_V (x - y) = A *_V x - A *_V y$
⟨*proof*⟩

lemma *sq-mtx-times-vec-assoc*: $(A * B) *_V x = A *_V (B *_V x)$
⟨*proof*⟩

lemma *sq-mtx-vec-mult-sum-cols*: $A *_V x = sum\ (\lambda i.\ x \$ i *_R \text{col } i\ A)\ UNIV$
⟨*proof*⟩

4.3 Real normed vector space of square matrices

instantiation *sq-mtx* :: (*finite*) *real-normed-vector*
begin

definition *norm-sq-mtx* :: *'a sq-mtx* ⇒ *real* **where** $\|A\| = \|to\text{-}vec\ A\|_{op}$

lift-definition *scaleR-sq-mtx* :: *real* ⇒ *'a sq-mtx* ⇒ *'a sq-mtx* **is** *scaleR* ⟨*proof*⟩

definition *sgn-sq-mtx* :: *'a sq-mtx* ⇒ *'a sq-mtx*
where *sgn-sq-mtx A* = $(inverse\ (\|A\|)) *_R A$

definition *dist-sq-mtx* :: *'a sq-mtx* ⇒ *'a sq-mtx* ⇒ *real*
where *dist-sq-mtx A B* = $\|A - B\|$

definition *uniformity-sq-mtx* :: (*'a sq-mtx* × *'a sq-mtx*) *filter*
where *uniformity-sq-mtx* = (*INF e*∈{*0*<..}. *principal* {(*x*, *y*). *dist x y* < *e*})

definition *open-sq-mtx* :: *'a sq-mtx set* ⇒ *bool*
where *open-sq-mtx U* = $(\forall x{\in}U.\ \forall_F\ (x', y)\ in\ uniformity.\ x' = x \longrightarrow y \in U)$

instance ⟨*proof*⟩

end

lemma *sq-mtx-scaleR-eq*: $c *_R A = to\text{-}mtx\ (\chi\ i\ j.\ c *_R A \$\$ i \$ j)$
⟨*proof*⟩

lemma *scaleR-to-mtx-ith*[*simp*]: $c *_R (to\text{-}mtx\ A) \$\$ i1 \$ i2 = c * A \$ i1 \$ i2$
⟨*proof*⟩

lemma *sq-mtx-scaleR-ith*[*simp*]: $(c *_R A) \$\$ i = (c\ *_R (A \$\$ i))$
⟨*proof*⟩

lemma *scaleR-sq-mtx-diag*: $c *_R sq\text{-}mtx\text{-}diag\ f = (\text{diag } i.\ c * f\ i)$
⟨*proof*⟩

lemma *scaleR-mtx-vec-assoc*: $(c *_R A) *_V x = c *_R (A *_V x)$
⟨*proof*⟩

lemma *mtx-vec-scaleR-commute*: $A *_V (c *_R x) = c *_R (A *_V x)$
⟨*proof*⟩

lemma *mtx-times-scaleR-commute*: $A * (c *_R B) = c *_R (A * B)$ **for** A::('*n*::*finite*) *sq-mtx*
⟨*proof*⟩

lemma *le-mtx-norm*: $m \in \{\|A *_V x\| \mid x. \|x\| = 1\} \Longrightarrow m \leq \|A\|$
⟨*proof*⟩

lemma *norm-vec-mult-le*: $\|A *_V x\| \leq (\|A\|) * (\|x\|)$
⟨*proof*⟩

lemma *bounded-bilinear-sq-mtx-vec-mult*: *bounded-bilinear* $(\lambda A\ s.\ A *_V s)$
⟨*proof*⟩

lemma *norm-sq-mtx-def2*: $\|A\| = Sup\ \{\|A *_V x\| \mid x. \|x\| = 1\}$
⟨*proof*⟩

lemma *norm-sq-mtx-def3*: $\|A\| = (SUP\ x. (\|A *_V x\|) / (\|x\|))$
⟨*proof*⟩

lemma *norm-sq-mtx-diag*: $\|sq\text{-}mtx\text{-}diag\ f\| = Max\ \{|f\ i| \mid i.\ i \in UNIV\}$
⟨*proof*⟩

lemma *sq-mtx-norm-le-sum-col*: $\|A\| \leq (\sum i \in UNIV.\ \|\mathrm{col}\ i\ A\|)$
⟨*proof*⟩

lemma *norm-le-transpose*: $\|A\| \leq \|A^\dagger\|$
⟨*proof*⟩

lemma *norm-eq-norm-transpose*[*simp*]: $\|A^\dagger\| = \|A\|$
⟨*proof*⟩

lemma *norm-column-le-norm*: $\|A\ \$\$\ i\| \leq \|A\|$
⟨*proof*⟩

4.4 Real normed algebra of square matrices

instantiation *sq-mtx* :: (*finite*) *real-normed-algebra-1*
begin

lift-definition *one-sq-mtx* :: '*a sq-mtx* **is** *to-mtx* (*mat 1*) ⟨*proof*⟩

lemma *sq-mtx-one-idty*: $1 * A = A\ \ A * 1 = A$ **for** A :: '*a sq-mtx*
⟨*proof*⟩

lemma *sq-mtx-norm-1*: $\|(1$::'*a sq-mtx*$)\| = 1$
⟨*proof*⟩

lemma *sq-mtx-norm-times*: $\|A * B\| \leq (\|A\|) * (\|B\|)$ **for** A :: '*a sq-mtx*

⟨*proof*⟩

instance
⟨*proof*⟩

end

lemma *sq-mtx-one-ith-simps*[*simp*]: $1 \$\$ i \$ i = 1$ $i \neq j \Longrightarrow 1 \$\$ i \$ j = 0$
⟨*proof*⟩

lemma *of-nat-eq-sq-mtx-diag*[*simp*]: $\textit{of-nat}\ m = (\mathrm{diag}\ i.\ m)$
⟨*proof*⟩

lemma *mtx-vec-mult-1*[*simp*]: $1 *_V s = s$
⟨*proof*⟩

lemma *sq-mtx-diag-one*[*simp*]: $(\mathrm{diag}\ i.\ 1) = 1$
⟨*proof*⟩

abbreviation $\textit{mtx-invertible}\ A \equiv \textit{invertible}\ (\textit{to-vec}\ A)$

lemma *mtx-invertible-def*: $\textit{mtx-invertible}\ A \longleftrightarrow (\exists A'.\ A' * A = 1 \wedge A * A' = 1)$
⟨*proof*⟩

lemma *mtx-invertibleI*:
assumes $A * B = 1$ **and** $B * A = 1$
shows *mtx-invertible* A
⟨*proof*⟩

lemma *mtx-invertibleD*[*simp*]:
assumes *mtx-invertible* A
shows $A^{-1} * A = 1$ **and** $A * A^{-1} = 1$
⟨*proof*⟩

lemma *mtx-invertible-inv*[*simp*]: $\textit{mtx-invertible}\ A \Longrightarrow \textit{mtx-invertible}\ (A^{-1})$
⟨*proof*⟩

lemma *mtx-invertible-one*[*simp*]: *mtx-invertible* 1
⟨*proof*⟩

lemma *sq-mtx-inv-unique*:
assumes $A * B = 1$ **and** $B * A = 1$
shows $A^{-1} = B$
⟨*proof*⟩

lemma *sq-mtx-inv-idempotent*[*simp*]: $\textit{mtx-invertible}\ A \Longrightarrow A^{-1-1} = A$
⟨*proof*⟩

lemma *sq-mtx-inv-mult*:

assumes *mtx-invertible* A **and** *mtx-invertible* B
shows $(A * B)^{-1} = B^{-1} * A^{-1}$
⟨*proof*⟩

lemma *sq-mtx-inv-one*[*simp*]: $1^{-1} = 1$
⟨*proof*⟩

definition *similar-sq-mtx* :: ('*n*::*finite*) *sq-mtx* ⇒ '*n* *sq-mtx* ⇒ *bool* (**infixr** ‹∼› *25*)
where $(A \sim B) \longleftrightarrow (\exists\ P.\ \textit{mtx-invertible}\ P \wedge A = P^{-1} * B * P)$

lemma *similar-sq-mtx-matrix*: $(A \sim B) =$ *similar-matrix* (*to-vec* A) (*to-vec* B)
⟨*proof*⟩

lemma *similar-sq-mtx-refl*[*simp*]: $A \sim A$
⟨*proof*⟩

lemma *similar-sq-mtx-simm*: $A \sim B \Longrightarrow B \sim A$
⟨*proof*⟩

lemma *similar-sq-mtx-trans*: $A \sim B \Longrightarrow B \sim C \Longrightarrow A \sim C$
⟨*proof*⟩

lemma *power-sq-mtx-diag*: (*sq-mtx-diag* f)$\hat{\ }n$ = (diag i. $f\ i\hat{\ }n$)
⟨*proof*⟩

lemma *power-similiar-sq-mtx-diag-eq*:
assumes *mtx-invertible* P
and $A = P^{-1} *$ (*sq-mtx-diag* f) $* P$
shows $A\hat{\ }n = P^{-1} *$ (diag i. $f\ i\hat{\ }n$) $* P$
⟨*proof*⟩

lemma *power-similar-sq-mtx-diag*:
assumes $A \sim$ (*sq-mtx-diag* f)
shows $A\hat{\ }n \sim$ (diag i. $f\ i\hat{\ }n$)
⟨*proof*⟩

4.5 Banach space of square matrices

lemma *Cauchy-cols*:
fixes X :: *nat* ⇒ ('*a*::*finite*) *sq-mtx*
assumes *Cauchy* X
shows *Cauchy* (λn. col i ($X\ n$))
⟨*proof*⟩

lemma *col-convergence*:
assumes $\forall\, i$. (λn. col i ($X\ n$)) ⟶ L \$ i
shows X ⟶ *to-mtx* (*transpose* L)
⟨*proof*⟩

instance *sq-mtx* :: (*finite*) *banach*
⟨*proof*⟩

lemma *exp-similiar-sq-mtx-diag-eq*:
assumes *mtx-invertible* P
and $A = P^{-1} * (\text{diag } i.\ f\ i) * P$
shows $exp\ A = P^{-1} * exp\ (\text{diag } i.\ f\ i) * P$
⟨*proof*⟩

lemma *exp-similiar-sq-mtx-diag*:
assumes $A \sim$ *sq-mtx-diag f*
shows *exp* $A \sim$ *exp* (*sq-mtx-diag f*)
⟨*proof*⟩

lemma *suminf-sq-mtx-diag*:
assumes $\forall\, i.\ (\lambda n.\ f\ n\ i)$ *sums* (*suminf* $(\lambda n.\ f\ n\ i)$)
shows $(\sum n.\ (\text{diag } i.\ f\ n\ i)) = (\text{diag } i.\ \sum n.\ f\ n\ i)$
⟨*proof*⟩

lemma *exp-sq-mtx-diag*: *exp* (*sq-mtx-diag f*) = (diag i. *exp* ($f\ i$))
⟨*proof*⟩

lemma *exp-scaleR-diagonal1*:
assumes *mtx-invertible* P **and** $A = P^{-1} * (\text{diag } i.\ f\ i) * P$
shows $exp\ (t *_R A) = P^{-1} * (\text{diag } i.\ exp\ (t * f\ i)) * P$
⟨*proof*⟩

lemma *exp-scaleR-diagonal2*:
assumes *mtx-invertible* P **and** $A = P * (\text{diag } i.\ f\ i) * P^{-1}$
shows $exp\ (t *_R A) = P * (\text{diag } i.\ exp\ (t * f\ i)) * P^{-1}$
⟨*proof*⟩

4.6 Examples

definition *mtx A* = *to-mtx* (*vector* (*map vector A*))

lemma *vector-nth-eq*: (*vector A*) \$ i = *foldr* $(\lambda x\ f\ n.\ (f\ (n + 1))(n := x))$ A $(\lambda n\ x.\ 0)$ *1* i
⟨*proof*⟩

lemma *mtx-ith-eq*[*simp*]: *mtx A* \$\$ i \$ j = *foldr* $(\lambda x\ f\ n.\ (f\ (n + 1))(n := x))$
(*map* (λl. *vec-lambda* (*foldr* $(\lambda x\ f\ n.\ (f\ (n + 1))(n := x))$ l $(\lambda n\ x.\ 0)$ *1*)) A) $(\lambda n\ x.\ 0)$ *1* i \$ j
⟨*proof*⟩

4.6.1 2x2 matrices

lemma *mtx2-eq-iff*: (*mtx*
([*a1*, *b1*] #

[*c1*, *d1*] # []) :: *2 sq-mtx*) = *mtx*
([*a2*, *b2*] #
[*c2*, *d2*] # []) ⟷ *a1* = *a2* ∧ *b1* = *b2* ∧ *c1* = *c2* ∧ *d1* = *d2*
⟨*proof*⟩

lemma *mtx2-to-mtx*: *mtx*
([*a*, *b*] #
[*c*, *d*] # []) =
to-mtx (χ *i j*::*2*. *if i*=*1* ∧ *j*=*1 then a*
else (*if i*=*1* ∧ *j*=*2 then b*
else (*if i*=*2* ∧ *j*=*1 then c*
else d)))
⟨*proof*⟩

abbreviation *diag2* :: *real* ⇒ *real* ⇒ *2 sq-mtx*
where *diag2* ι_1 ι_2 ≡ *mtx*
([ι_1, *0*] #
[*0*, ι_2] # [])

lemma *diag2-eq*: *diag2* (ι *1*) (ι *2*) = (diag *i*. ι *i*)
⟨*proof*⟩

lemma *one-mtx2*: (*1*::*2 sq-mtx*) = *diag2 1 1*
⟨*proof*⟩

lemma *zero-mtx2*: (*0*::*2 sq-mtx*) = *diag2 0 0*
⟨*proof*⟩

lemma *scaleR-mtx2*: *k* $*_R$ *mtx*
([*a*, *b*] #
[*c*, *d*] # []) = *mtx*
([*k*∗*a*, *k*∗*b*] #
[*k*∗*c*, *k*∗*d*] # [])
⟨*proof*⟩

lemma *uminus-mtx2*: −*mtx*
([*a*, *b*] #
[*c*, *d*] # []) = (*mtx*
([−*a*, −*b*] #
[−*c*, −*d*] # [])::*2 sq-mtx*)
⟨*proof*⟩

lemma *plus-mtx2*: *mtx*
([*a1*, *b1*] #
[*c1*, *d1*] # []) + *mtx*
([*a2*, *b2*] #
[*c2*, *d2*] # []) = ((*mtx*
([*a1*+*a2*, *b1*+*b2*] #
[*c1*+*c2*, *d1*+*d2*] # []))::*2 sq-mtx*)

⟨*proof*⟩

lemma *minus-mtx2*: *mtx*
([*a1*, *b1*] #
[*c1*, *d1*] # []) − *mtx*
([*a2*, *b2*] #
[*c2*, *d2*] # []) = ((*mtx*
([*a1*−*a2*, *b1*−*b2*] #
[*c1*−*c2*, *d1*−*d2*] # []))::*2 sq-mtx*)
⟨*proof*⟩

lemma *times-mtx2*: *mtx*
([*a1*, *b1*] #
[*c1*, *d1*] # []) ∗ *mtx*
([*a2*, *b2*] #
[*c2*, *d2*] # []) = ((*mtx*
([*a1*∗*a2*+*b1*∗*c2*, *a1*∗*b2*+*b1*∗*d2*] #
[*c1*∗*a2*+*d1*∗*c2*, *c1*∗*b2*+*d1*∗*d2*] # []))::*2 sq-mtx*)
⟨*proof*⟩

4.6.2 3x3 matrices

lemma *mtx3-to-mtx*: *mtx*
([a_{11}, a_{12}, a_{13}] #
[a_{21}, a_{22}, a_{23}] #
[a_{31}, a_{32}, a_{33}] # []) =
to-mtx (χ *i j*::*3*. *if i*=*1* ∧ *j*=*1 then* a_{11}
else (*if i*=*1* ∧ *j*=*2 then* a_{12}
else (*if i*=*1* ∧ *j*=*3 then* a_{13}
else (*if i*=*2* ∧ *j*=*1 then* a_{21}
else (*if i*=*2* ∧ *j*=*2 then* a_{22}
else (*if i*=*2* ∧ *j*=*3 then* a_{23}
else (*if i*=*3* ∧ *j*=*1 then* a_{31}
else (*if i*=*3* ∧ *j*=*2 then* a_{32}
else a_{33}))))))))
⟨*proof*⟩

abbreviation *diag3* :: *real* ⇒ *real* ⇒ *real* ⇒ *3 sq-mtx*
where *diag3* ι_1 ι_2 ι_3 ≡ *mtx*
([ι_1, *0*, *0*] #
[*0*, ι_2, *0*] #
[*0*, *0*, ι_3] # [])

lemma *diag3-eq*: *diag3* (ι *1*) (ι *2*) (ι *3*) = (diag *i*. ι *i*)
⟨*proof*⟩

lemma *one-mtx3*: (*1*::*3 sq-mtx*) = *diag3 1 1 1*
⟨*proof*⟩

lemma *zero-mtx3*: $(0::3\ sq\text{-}mtx) = diag3\ 0\ 0\ 0$
⟨*proof*⟩

lemma *scaleR-mtx3*: $k *_R mtx$
$([a_{11}, a_{12}, a_{13}]$ #
$[a_{21}, a_{22}, a_{23}]$ #
$[a_{31}, a_{32}, a_{33}]$ # []) = *mtx*
$([k*a_{11}, k*a_{12}, k*a_{13}]$ #
$[k*a_{21}, k*a_{22}, k*a_{23}]$ #
$[k*a_{31}, k*a_{32}, k*a_{33}]$ # [])
⟨*proof*⟩

lemma *plus-mtx3*: *mtx*
$([a_{11}, a_{12}, a_{13}]$ #
$[a_{21}, a_{22}, a_{23}]$ #
$[a_{31}, a_{32}, a_{33}]$ # []) + *mtx*
$([b_{11}, b_{12}, b_{13}]$ #
$[b_{21}, b_{22}, b_{23}]$ #
$[b_{31}, b_{32}, b_{33}]$ # []) = (*mtx*
$([a_{11}+b_{11}, a_{12}+b_{12}, a_{13}+b_{13}]$ #
$[a_{21}+b_{21}, a_{22}+b_{22}, a_{23}+b_{23}]$ #
$[a_{31}+b_{31}, a_{32}+b_{32}, a_{33}+b_{33}]$ # [])::*3 sq-mtx*)
⟨*proof*⟩

lemma *minus-mtx3*: *mtx*
$([a_{11}, a_{12}, a_{13}]$ #
$[a_{21}, a_{22}, a_{23}]$ #
$[a_{31}, a_{32}, a_{33}]$ # []) − *mtx*
$([b_{11}, b_{12}, b_{13}]$ #
$[b_{21}, b_{22}, b_{23}]$ #
$[b_{31}, b_{32}, b_{33}]$ # []) = (*mtx*
$([a_{11}-b_{11}, a_{12}-b_{12}, a_{13}-b_{13}]$ #
$[a_{21}-b_{21}, a_{22}-b_{22}, a_{23}-b_{23}]$ #
$[a_{31}-b_{31}, a_{32}-b_{32}, a_{33}-b_{33}]$ # [])::*3 sq-mtx*)
⟨*proof*⟩

lemma *times-mtx3*: *mtx*
$([a_{11}, a_{12}, a_{13}]$ #
$[a_{21}, a_{22}, a_{23}]$ #
$[a_{31}, a_{32}, a_{33}]$ # []) ∗ *mtx*
$([b_{11}, b_{12}, b_{13}]$ #
$[b_{21}, b_{22}, b_{23}]$ #
$[b_{31}, b_{32}, b_{33}]$ # []) = (*mtx*
$([a_{11}*b_{11}+a_{12}*b_{21}+a_{13}*b_{31}, a_{11}*b_{12}+a_{12}*b_{22}+a_{13}*b_{32}, a_{11}*b_{13}+a_{12}*b_{23}+a_{13}*b_{33}]$
#
$[a_{21}*b_{11}+a_{22}*b_{21}+a_{23}*b_{31}, a_{21}*b_{12}+a_{22}*b_{22}+a_{23}*b_{32}, a_{21}*b_{13}+a_{22}*b_{23}+a_{23}*b_{33}]$
#
$[a_{31}*b_{11}+a_{32}*b_{21}+a_{33}*b_{31}, a_{31}*b_{12}+a_{32}*b_{22}+a_{33}*b_{32}, a_{31}*b_{13}+a_{32}*b_{23}+a_{33}*b_{33}]$
[])::*3 sq-mtx*)

⟨*proof*⟩

end

5 Affine systems of ODEs

Affine systems of ordinary differential equations (ODEs) are those whose vector fields are linear operators. Broadly speaking, if there are functions A and B such that the system of ODEs $X'\,t = f\,(X\,t)$ turns into $X'\,t = (A\,t) \cdot (X\,t) + (B\,t)$, then it is affine. The end goal of this section is to prove that every affine system of ODEs has a unique solution, and to obtain a characterization of said solution.

theory *MTX-Flows*
  imports
    SQ-MTX
    Hybrid-Systems-VCs.HS-ODEs

begin

5.1 Existence and uniqueness for affine systems

definition *matrix-continuous-on* :: *real set* ⇒ (*real* ⇒ (*'a*::*real-normed-algebra-1*)^*'n*^*'m*) ⇒ *bool*
  where *matrix-continuous-on* T A = ($\forall t \in T.\ \forall \varepsilon > 0.\ \exists\ \delta > 0.\ \forall \tau{\in}T.\ |\tau - t| < \delta \longrightarrow \|A\ \tau - A\ t\|_{op} \leq \varepsilon$)

lemma *continuous-on-matrix-vector-multl*:
  assumes *matrix-continuous-on* T A
  shows *continuous-on* T ($\lambda t.\ A\ t *v\ s$)
⟨*proof*⟩

lemma *lipschitz-cond-affine*:
  fixes A :: *real* ⇒ *'a*::*real-normed-algebra-1*^*'n*^*'m* **and** T::*real set*
  defines $L \equiv Sup\ \{\|A\ t\|_{op}\ |t.\ t \in T\}$
  assumes $t \in T$ **and** *bdd-above* $\{\|A\ t\|_{op}\ |t.\ t \in T\}$
  shows $\|A\ t *v\ x - A\ t *v\ y\| \leq L * (\|x - y\|)$
⟨*proof*⟩

lemma *local-lipschitz-affine*:
  fixes A :: *real* ⇒ *'a*::*real-normed-algebra-1*^*'n*^*'m*
  assumes *open* T **and** *open* S
    and *Ahyp*: $\bigwedge\tau\ \varepsilon.\ \varepsilon > 0 \Longrightarrow \tau \in T \Longrightarrow$ *cball* $\tau\ \varepsilon \subseteq T \Longrightarrow$ *bdd-above* $\{\|A\ t\|_{op}\ |t.\ t \in$ *cball* $\tau\ \varepsilon\}$
  shows *local-lipschitz* T S ($\lambda t\ s.\ A\ t *v\ s + B\ t$)
⟨*proof*⟩

lemma *picard-lindeloef-affine*:
  fixes A :: *real* ⇒ *'a*::{*banach*,*real-normed-algebra-1*,*heine-borel*}^*'n*^*'n*

assumes *Ahyp*: *matrix-continuous-on T A*
and $\bigwedge\tau\ \varepsilon.\ \tau \in T \Longrightarrow \varepsilon > 0 \Longrightarrow$ *bdd-above* $\{\|A\ t\|_{op}\ |t.\ dist\ \tau\ t \leq \varepsilon\}$
and *Bhyp*: *continuous-on T B* **and** *open S*
and $t_0 \in T$ **and** *Thyp*: *open T is-interval T*
shows *picard-lindeloef* $(\lambda\ t\ s.\ A\ t *v\ s + B\ t)\ T\ S\ t_0$
⟨*proof*⟩

lemma *picard-lindeloef-autonomous-affine*:
fixes A :: $'a$::{*banach*,*real-normed-field*,*heine-borel*}^'n^'n
shows *picard-lindeloef* $(\lambda\ t\ s.\ A *v\ s + B)$ *UNIV UNIV* t_0
⟨*proof*⟩

lemma *picard-lindeloef-autonomous-linear*:
fixes A :: $'a$::{*banach*,*real-normed-field*,*heine-borel*}^'n^'n
shows *picard-lindeloef* $(\lambda\ t.\ (*v)\ A)$ *UNIV UNIV* t_0
⟨*proof*⟩

lemmas *unique-sol-autonomous-affine* = *picard-lindeloef.ivp-unique-solution*[*OF picard-lindeloef-autonomous-affine UNIV-I - subset-UNIV*]

lemmas *unique-sol-autonomous-linear* = *picard-lindeloef.ivp-unique-solution*[*OF picard-lindeloef-autonomous-linear UNIV-I - subset-UNIV*]

5.2 Flow for affine systems

5.2.1 Derivative rules for square matrices

declare *has-derivative-component* [*simp del*]

lemma *has-derivative-exp-scaleRl*[*derivative-intros*]:
fixes *f*::*real* ⇒ *real*
assumes $D\ f \mapsto f'$ *at t within T*
shows $D\ (\lambda t.\ exp\ (f\ t *_R A)) \mapsto (\lambda h.\ f'\ h *_R (exp\ (f\ t *_R A) * A))$ *at t within T*
⟨*proof*⟩

lemma *vderiv-on-exp-scaleRlI*[*poly-derivatives*]:
assumes $D\ f = f'$ *on T* **and** $g' = (\lambda x.\ f'\ x *_R exp\ (f\ x *_R A) * A)$
shows $D\ (\lambda x.\ exp\ (f\ x *_R A)) = g'$ *on T*
⟨*proof*⟩

lemma *has-derivative-mtx-ith*[*derivative-intros*]:
fixes *t*::*real* **and** *T* :: *real set*
defines $t_0 \equiv$ *netlimit* (*at t within T*)
assumes $D\ A \mapsto (\lambda h.\ h *_R A'\ t)$ *at t within T*
shows $D\ (\lambda t.\ A\ t\ \$\$\ i) \mapsto (\lambda h.\ h *_R A'\ t\ \$\$\ i)$ *at t within T*
⟨*proof*⟩

lemmas *has-derivative-mtx-vec-mult*[*derivative-intros*] =
bounded-bilinear.FDERIV[*OF bounded-bilinear-sq-mtx-vec-mult*]

lemma *vderiv-on-mtx-vec-multI*[*poly-derivatives*]:
assumes $D\ u = u'\ on\ T$ **and** $D\ A = A'\ on\ T$
and $g = (\lambda t.\ A\ t *_V\ u'\ t + A'\ t *_V\ u\ t\ )$
shows $D\ (\lambda t.\ A\ t *_V\ u\ t) = g\ on\ T$
⟨*proof*⟩

lemmas *has-vderiv-on-ivl-integral* = *ivl-integral-has-vderiv-on*[*OF vderiv-on-continuous-on*]

declare *has-vderiv-on-ivl-integral* [*poly-derivatives*]

lemma *has-derivative-mtx-vec-multl*[*derivative-intros*]:
assumes $\bigwedge\ i\ j.\ D\ (\lambda t.\ (A\ t)\ \$\$\ i\ \$\ j) \mapsto (\lambda\tau.\ \tau *_R\ (A'\ t)\ \$\$\ i\ \$\ j)\ (at\ t\ within$
$T)$
shows $D\ (\lambda t.\ A\ t *_V\ x) \mapsto (\lambda\tau.\ \tau *_R\ (A'\ t) *_V\ x)\ at\ t\ within\ T$
⟨*proof*⟩

declare *has-derivative-component* [*simp*]

lemma *continuous-on-mtx-vec-multr*: *continuous-on* $S\ ((*_V)\ A)$
⟨*proof*⟩

Isabelle automatically generates derivative rules from this subsubsection

thm *derivative-eq-intros*(*140*−)

5.2.2 Existence and uniqueness with square matrices

Finally, we can use the *exp* operation to characterize the general solutions for affine systems of ODEs. We show that they satisfy the *local-flow* locale.

lemma *continuous-on-sq-mtx-vec-multl*:
fixes $A :: real \Rightarrow ('n::finite)\ sq\text{-}mtx$
assumes *continuous-on* $T\ A$
shows *continuous-on* $T\ (\lambda t.\ A\ t *_V\ s)$
⟨*proof*⟩

lemmas *continuous-on-affine* = *continuous-on-add*[*OF continuous-on-sq-mtx-vec-multl*]

lemma *local-lipschitz-sq-mtx-affine*:
fixes $A :: real \Rightarrow ('n::finite)\ sq\text{-}mtx$
assumes *continuous-on* $T\ A$ *open* T *open* S
shows *local-lipschitz* $T\ S\ (\lambda t\ s.\ A\ t *_V\ s + B\ t)$
⟨*proof*⟩

lemma *picard-lindeloef-sq-mtx-affine*:
assumes *continuous-on* $T\ A$ **and** *continuous-on* $T\ B$
and $t_0 \in T$ *is-interval* T *open* T **and** *open* S
shows *picard-lindeloef* $(\lambda t\ s.\ A\ t *_V\ s + B\ t)\ T\ S\ t_0$
⟨*proof*⟩

lemmas *sq-mtx-unique-sol-autonomous-affine* = *picard-lindeloef.ivp-unique-solution*[*OF*

picard-lindeloef-sq-mtx-affine[*OF*
continuous-on-const
continuous-on-const
UNIV-I is-interval-univ
open-UNIV open-UNIV]
UNIV-I - subset-UNIV]

lemma *has-vderiv-on-sq-mtx-linear*:
$D\ (\lambda t.\ exp\ ((t - t_0) *_R A) *_V s) = (\lambda t.\ A *_V (exp\ ((t - t_0) *_R A) *_V s))\ on\ \{t_0{-}{-}t\}$
⟨*proof*⟩

lemma *has-vderiv-on-sq-mtx-affine*:
fixes t_0::*real* **and** A :: ('*a*::*finite*) *sq-mtx*
defines $lSol\ c\ t \equiv exp\ ((c * (t - t_0)) *_R A)$
shows $D\ (\lambda t.\ lSol\ 1\ t *_V s + lSol\ 1\ t *_V (\int_{t0}{}^{t} (lSol\ (-1)\ \tau *_V B)\ \partial\tau)) = (\lambda t.\ A *_V (lSol\ 1\ t *_V s + lSol\ 1\ t *_V (\int_{t0}{}^{t} (lSol\ (-1)\ \tau *_V B)\ \partial\tau)) + B)\ on\ \{t_0{-}{-}t\}$
⟨*proof*⟩

lemma *autonomous-linear-sol-is-exp*:
assumes $D\ X = (\lambda t.\ A *_V X\ t)\ on\ \{t_0{-}{-}t\}$ **and** $X\ t_0 = s$
shows $X\ t = exp\ ((t - t_0) *_R A) *_V s$
⟨*proof*⟩

lemma *autonomous-affine-sol-is-exp-plus-int*:
assumes $D\ X = (\lambda t.\ A *_V X\ t + B)\ on\ \{t_0{-}{-}t\}$ **and** $X\ t_0 = s$
shows $X\ t = exp\ ((t - t_0) *_R A) *_V s + exp\ ((t - t_0) *_R A) *_V (\int_{t0}{}^{t}(exp\ (-(\tau - t_0) *_R A) *_V B)\partial\tau)$
⟨*proof*⟩

lemma *local-flow-sq-mtx-linear*: $local\text{-}flow\ ((*_V)\ A)\ UNIV\ UNIV\ (\lambda t\ s.\ exp\ (t *_R A) *_V s)$
⟨*proof*⟩

lemma *local-flow-sq-mtx-affine*: $local\text{-}flow\ (\lambda s.\ A *_V s + B)\ UNIV\ UNIV$
$(\lambda t\ s.\ exp\ (t *_R A) *_V s + exp\ (t *_R A) *_V (\int_0{}^{t}(exp\ (-\tau *_R A) *_V B)\partial\tau))$
⟨*proof*⟩

end

6 Verification examples

theory *MTX-Examples*
imports
MTX-Flows

Hybrid-Systems-VCs.HS-VC-Spartan

begin

6.1 Examples

abbreviation *hoareT* :: ($'a \Rightarrow bool$) $\Rightarrow$ ($'a \Rightarrow 'a\ set$) $\Rightarrow$ ($'a \Rightarrow bool$) $\Rightarrow$ *bool*
(‹*PRE- HP - POST -*› [*85*,*85*]*85*) **where** *PRE P HP X POST Q* $\equiv$ ($P \leq |X] Q$)

6.1.1 Verification by uniqueness.

abbreviation *mtx-circ* :: *2 sq-mtx* (‹*A*›)
where $A \equiv$ *mtx*
([*0*, *1*] #
[-1, *0*] # [])

abbreviation *mtx-circ-flow* :: *real* $\Rightarrow$ *real^2* $\Rightarrow$ *real^2* (‹φ›)
where $\varphi\ t\ s \equiv (\chi\ i.\ if\ i = 1\ then\ s\$1 * cos\ t + s\$2 * sin\ t\ else - s\$1 * sin\ t + s\$2 * cos\ t)$

lemma *mtx-circ-flow-eq*: $exp\ (t *_R A) *_V s = \varphi\ t\ s$
⟨*proof*⟩

lemma *mtx-circ*:
$PRE(\lambda s.\ r^2 = (s\ \$\ 1)^2 + (s\ \$\ 2)^2)$
$HP\ x' {=} (*_V)\ A\ \&\ G$
$POST\ (\lambda s.\ r^2 = (s\ \$\ 1)^2 + (s\ \$\ 2)^2)$
⟨*proof*⟩

no-notation *mtx-circ* (‹*A*›)
and *mtx-circ-flow* (‹φ›)

6.1.2 Flow of diagonalisable matrix.

abbreviation *mtx-hOsc* :: *real* $\Rightarrow$ *real* $\Rightarrow$ *2 sq-mtx* (‹*A*›)
where *A a b* $\equiv$ *mtx*
([*0*, *1*] #
[*a*, *b*] # [])

abbreviation *mtx-chB-hOsc* :: *real* $\Rightarrow$ *real* $\Rightarrow$ *2 sq-mtx* (‹*P*›)
where *P a b* $\equiv$ *mtx*
([*a*, *b*] #
[*1*, *1*] # [])

lemma *inv-mtx-chB-hOsc*:
$a \neq b \Longrightarrow (P\ a\ b)^{-1} = (1/(a - b)) *_R mtx$
([*1*, $-b$] #
[-1, *a*] # [])
⟨*proof*⟩

lemma *invertible-mtx-chB-hOsc*: $a \neq b \Longrightarrow$ *mtx-invertible* $(P\ a\ b)$
⟨*proof*⟩

lemma *mtx-hOsc-diagonalizable*:
fixes $a\ b :: real$
defines $\iota_1 \equiv (b - sqrt\ (b\hat{}2+4*a))/2$ **and** $\iota_2 \equiv (b + sqrt\ (b\hat{}2+4*a))/2$
assumes $b^2 + a * 4 > 0$ **and** $a \neq 0$
shows $A\ a\ b = P\ (-\iota_2/a)\ (-\iota_1/a) * (\text{diag}\ i.\ \textit{if}\ i = 1\ \textit{then}\ \iota_1\ \textit{else}\ \iota_2) * (P\ (-\iota_2/a)\ (-\iota_1/a))^{-1}$
⟨*proof*⟩

lemma *mtx-hOsc-solution-eq*:
fixes $a\ b :: real$
defines $\iota_1 \equiv (b - sqrt\ (b^2+4*a))/2$ **and** $\iota_2 \equiv (b + sqrt\ (b^2+4*a))/2$
defines $\Phi\ t \equiv mtx$ (
$[\iota_2*exp(t*\iota_1) - \iota_1*exp(t*\iota_2), \quad exp(t*\iota_2)-exp(t*\iota_1)]\#$
$[a*exp(t*\iota_2) - a*exp(t*\iota_1), \iota_2*exp(t*\iota_2)-\iota_1*exp(t*\iota_1)]\#[])$
assumes $b^2 + a * 4 > 0$ **and** $a \neq 0$
shows $P\ (-\iota_2/a)\ (-\iota_1/a) * (\text{diag}\ i.\ exp\ (t * (\textit{if}\ i{=}1\ \textit{then}\ \iota_1\ \textit{else}\ \iota_2))) * (P\ (-\iota_2/a)\ (-\iota_1/a))^{-1}$
$= (1/sqrt\ (b^2 + a * 4)) *_R (\Phi\ t)$
⟨*proof*⟩

lemma *local-flow-mtx-hOsc*:
fixes $a\ b$
defines $\iota_1 \equiv (b - sqrt\ (b\hat{}2+4*a))/2$ **and** $\iota_2 \equiv (b + sqrt\ (b\hat{}2+4*a))/2$
defines $\Phi\ t \equiv mtx$ (
$[\iota_2*exp(t*\iota_1) - \iota_1*exp(t*\iota_2), \quad exp(t*\iota_2)-exp(t*\iota_1)]\#$
$[a*exp(t*\iota_2) - a*exp(t*\iota_1), \iota_2*exp(t*\iota_2)-\iota_1*exp(t*\iota_1)]\#[])$
assumes $b^2 + a * 4 > 0$ **and** $a \neq 0$
shows *local-flow* $((*_V)\ (A\ a\ b))$ *UNIV UNIV* $(\lambda t.\ (*_V)\ ((1/sqrt\ (b^2 + a * 4)) *_R \Phi\ t))$
⟨*proof*⟩

lemma *overdamped-door-arith*:
assumes $b^2 + a * 4 > 0$ **and** $a < 0$ **and** $b \leq 0$ **and** $t \geq 0$ **and** $s1 > 0$
shows $0 \leq ((b + sqrt\ (b^2 + 4 * a)) * exp\ (t * (b - sqrt\ (b^2 + 4 * a)) / 2) / 2 -$
$(b - sqrt\ (b^2 + 4 * a)) * exp\ (t * (b + sqrt\ (b^2 + 4 * a)) / 2) / 2) * s1 / sqrt\ (b^2 + a * 4)$
⟨*proof*⟩

abbreviation *open-door* $s \equiv \{s.\ s\$1 > 0 \wedge s\$2 = 0\}$

lemma *overdamped-door*:
assumes $b^2 + a * 4 > 0$ **and** $a < 0$ **and** $b \leq 0$
shows *PRE* $(\lambda s.\ s\$1 = 0)$
HP (*LOOP open-door*; $(x\acute{}=((*_V)\ (A\ a\ b))\ \&\ G)$ *INV* $(\lambda s.\ 0 \leq s\$1))$
POST $(\lambda s.\ 0 \leq s\ \$\ 1)$

⟨*proof*⟩

no-notation *mtx-hOsc* (‹*A*›)
and *mtx-chB-hOsc* (‹*P*›)

6.1.3 Flow of non-diagonalisable matrix.

abbreviation *mtx-cnst-acc* :: *3 sq-mtx* (‹*K*›)
where $K \equiv mtx$ (
[*0*,*1*,*0*] #
[*0*,*0*,*1*] #
[*0*,*0*,*0*] # [])

lemma *pow2-scaleR-mtx-cnst-acc*: $(t *_R K)^2 = mtx$ (
[*0*,*0*,t^2] #
[*0*,*0*,*0*] #
[*0*,*0*,*0*] # [])
⟨*proof*⟩

lemma *powN-scaleR-mtx-cnst-acc*: $n > 2 \Longrightarrow (t *_R K)\hat{}n = 0$
⟨*proof*⟩

lemma *exp-mtx-cnst-acc*: $exp\ (t *_R K) = ((t *_R K)^2/_R\ 2) + (t *_R K) + 1$
⟨*proof*⟩

lemma *exp-mtx-cnst-acc-simps*:
$exp\ (t *_R K)$ $$ *1* $ *1* = *1* $exp\ (t *_R K)$ $$ *1* $ *2* = *t* $exp\ (t *_R K)$ $$ *1* $ *3* = $t\hat{}2/2$
$exp\ (t *_R K)$ $$ *2* $ *1* = *0* $exp\ (t *_R K)$ $$ *2* $ *2* = *1* $exp\ (t *_R K)$ $$ *2* $ *3* = *t*
$exp\ (t *_R K)$ $$ *3* $ *1* = *0* $exp\ (t *_R K)$ $$ *3* $ *2* = *0* $exp\ (t *_R K)$ $$ *3* $ *3* = *1*
⟨*proof*⟩

lemma *exp-mtx-cnst-acc-vec-mult-eq*: $exp\ (t *_R K) *_V s =$
vector [*s*$*3* ∗ $t\hat{}2/2$ + *s*$*2* ∗ *t* + *s*$*1*, *s*$*3* ∗ *t* + *s*$*2*, *s*$*3*]
⟨*proof*⟩

lemma *local-flow-mtx-cnst-acc*:
local-flow $((*_V)\ K)$ *UNIV UNIV* $(\lambda t\ s.\ ((t *_R K)^2/_R\ 2 + (t *_R K) + 1) *_V s)$
⟨*proof*⟩

lemma *docking-station-arith*:
assumes (*d*::*real*) > *x* **and** *v* > *0*
shows $(v = v^2 * t\ /\ (2 * d - 2 * x)) \longleftrightarrow (v * t - v^2 * t^2\ /\ (4 * d - 4 * x) + x = d)$
⟨*proof*⟩

lemma *docking-station*:
assumes $d > x_0$ **and** $v_0 > 0$
shows *PRE* $(\lambda s.\ s\$1 = x_0 \wedge s\$2 = v_0)$
HP $((3 ::= (\lambda s.\ -(v_0\hat{\ }2/(2*(d-x_0))))); x' {=} (*_V)\ K\ \&\ G)$
POST $(\lambda s.\ s\$2 = 0 \longleftrightarrow s\$1 = d)$
⟨*proof*⟩

no-notation *mtx-cnst-acc* (‹*K*›)

end

References

[1] A. Armstrong, V. B. F. Gomes, and G. Struth. Building program construction and verification tools from algebraic principles. *Formal Aspects of Computing*, 28(2):265–293, 2016.

[2] S. Foster, J. J. H. y Munive, and G. Struth. Differential Hoare logics and refinement calculi for hybrid systems with Isabelle/HOL. arXiv:1909.05618[cs.LO], 2019.

[3] J. J. Huerta y Munive. Verification components for hybrid systems. *Archive of Formal Proofs*, 2019.

[4] J. J. Huerta y Munive. Affine systems of ODEs in Isabelle/HOL for hybrid-program verification. In *SEFM 2020*, volume 12310 of *LNCS*, pages 77–92. Springer, 2020.

[5] J. J. Huerta y Munive and G. Struth. Predicate transformer semantics for hybrid systems: Verification components for Isabelle/HOL. arXiv:1909.05618 [cs.LO], 2019.

[6] F. Immler and J. Hölzl. Ordinary differential equations. *Archive of Formal Proofs*, 2012.

[7] A. Platzer. *Logical Analysis of Hybrid Systems*. Springer, 2010.