Simultaneous diagonalization of pairwise commuting Hermitian matrices

Mnacho Echenim

March 17, 2025

Abstract

A Hermitian matrix is a square complex matrix A that is equal to its conjugate transpose $A^\dagger$. The (finite-dimensional) spectral theorem states that for any such matrix A, we have the equality $A = U \cdot B \cdot U^\dagger$, where U is a unitary matrix and B is a diagonal matrix containing only real elements. We formalize the generalization of this result, which states that if $\{A_1, \ldots, A_n\}$ are Hermitian and pairwise commuting matrices, then there exists a unitary matrix U such that $A_i = U \cdot B_i \cdot U^\dagger$, for $i = 1, \ldots, n$, and each B_i is diagonal and contains only real elements. Sets of pairwise commuting Hermitian matrices are called *Complete Sets of Commuting Observables* in Quantum Mechanics, where they represent physical quantities that can be simultaneously measured to uniquely distinguish quantum states.

Contents

Acknowledgments This work was partially supported by Agence Nationale de la Recherche, through *Plan France 2030 (ref. ANR-22-PETQ-0007)*.

theory *Spectral-Theory-Complements* **imports** *HOL−Combinatorics.Permutations*
Projective-Measurements.Linear-Algebra-Complements
Projective-Measurements.Projective-Measurements

begin

1 Some preliminary results

1.1 Roots of a polynomial

Results on polynomials, the main one being that the set of roots of a polynomial is uniquely defined.

lemma *root-poly-linear*:
shows *poly* $(\prod a{\leftarrow}L.\ [:-\ a,\ 1:])\ (c::'a::\mathit{field}) = 0 \Longrightarrow c \in \mathit{set}\ L$
⟨*proof*⟩

lemma *poly-root-set-subseteq*:
assumes $(\prod (a::'a::\mathit{field}){\leftarrow}L.\ [:-\ a,\ 1:]) = (\prod a{\leftarrow}M.\ [:-\ a,\ 1:])$
shows $\mathit{set}\ L \subseteq \mathit{set}\ M$
⟨*proof*⟩

lemma *poly-root-set-eq*:
assumes $(\prod (a::'a::\mathit{field}){\leftarrow}L.\ [:-\ a,\ 1:]) = (\prod a{\leftarrow}M.\ [:-\ a,\ 1:])$
shows $\mathit{set}\ L = \mathit{set}\ M$ ⟨*proof*⟩

1.2 Linear algebra preliminaries

lemma *minus-zero-vec-eq*:
fixes $v::'a::\{\mathit{ab\text{-}group\text{-}add}\}$ *Matrix.vec*
assumes $\mathit{dim\text{-}vec}\ v = n$
and $\mathit{dim\text{-}vec}\ w = n$

and $v - w = 0_v\ n$
shows $v = w$
⟨*proof*⟩

lemma *right-minus-zero-mat*:
fixes $A::'a::\{group\text{-}add\}$ *Matrix.mat*
shows $A - 0_m\ (dim\text{-}row\ A)\ (dim\text{-}col\ A) = A$
⟨*proof*⟩

lemma *smult-zero*:
shows $(0::'a::comm\text{-}ring) \cdot_m A = 0_m\ (dim\text{-}row\ A)\ (dim\text{-}col\ A)$ ⟨*proof*⟩

lemma *rank-1-proj-col-carrier*:
assumes $i < dim\text{-}col\ A$
shows *rank-1-proj* $(Matrix.col\ A\ i) \in carrier\text{-}mat\ (dim\text{-}row\ A)\ (dim\text{-}row\ A)$
⟨*proof*⟩

lemma *zero-adjoint*:
shows *Complex-Matrix.adjoint* $(0_m\ n\ m) = ((0_m\ m\ n)::'a::conjugatable\text{-}field$ *Matrix.mat*)
⟨*proof*⟩

lemma *assoc-mat-mult-vec'*:
assumes $A \in carrier\text{-}mat\ n\ n$
and $B \in carrier\text{-}mat\ n\ n$
and $C \in carrier\text{-}mat\ n\ n$
and $v \in carrier\text{-}vec\ n$
shows $A * B * C *_v v = A *_v (B *_v (C *_v v))$ ⟨*proof*⟩

lemma *adjoint-dim'*:
$A \in carrier\text{-}mat\ n\ m \Longrightarrow Complex\text{-}Matrix.adjoint\ A \in carrier\text{-}mat\ m\ n$
⟨*proof*⟩

definition *mat-conj* **where**
$mat\text{-}conj\ U\ V = U * V * (Complex\text{-}Matrix.adjoint\ U)$

lemma *mat-conj-adjoint*:
shows $mat\text{-}conj\ (Complex\text{-}Matrix.adjoint\ U)\ V =$
$Complex\text{-}Matrix.adjoint\ U * V * U$ ⟨*proof*⟩

lemma *map2-mat-conj-exp*:
assumes $length\ A = length\ B$
shows $map2\ (*)\ (map2\ (*)\ A\ B)\ (map\ Complex\text{-}Matrix.adjoint\ A) =$
$map2\ mat\text{-}conj\ A\ B$ ⟨*proof*⟩

lemma *mat-conj-unit-commute*:
assumes *unitary* U
and $U*A = A*U$
and $A \in carrier\text{-}mat\ n\ n$

and $U \in$ *carrier-mat n n*
shows *mat-conj U A* = *A*
⟨*proof*⟩

lemma *hermitian-mat-conj*:
assumes $A \in$ *carrier-mat n n*
and $U \in$ *carrier-mat n n*
and *hermitian A*
shows *hermitian* (*mat-conj U A*)
⟨*proof*⟩

lemma *hermitian-mat-conj'*:
assumes $A \in$ *carrier-mat n n*
and $U \in$ *carrier-mat n n*
and *hermitian A*
shows *hermitian* (*mat-conj* (*Complex-Matrix.adjoint U*) *A*)
⟨*proof*⟩

lemma *mat-conj-uminus-eq*:
assumes $A \in$ *carrier-mat n n*
and $U \in$ *carrier-mat n n*
and $B \in$ *carrier-mat n n*
and *A* = *mat-conj U B*
shows $-A$ = *mat-conj U* $(-B)$ ⟨*proof*⟩

lemma *mat-conj-smult*:
assumes $A \in$ *carrier-mat n n*
and $U \in$ *carrier-mat n n*
and $B \in$ *carrier-mat n n*
and *A* = *U* ∗ *B* ∗ (*Complex-Matrix.adjoint U*)
shows *x* $\cdot_m$ *A* = *U* ∗ (*x* $\cdot_m$ *B*) ∗ (*Complex-Matrix.adjoint U*) ⟨*proof*⟩

lemma *mult-adjoint-hermitian*:
fixes *A*::'*a*::*conjugatable-field Matrix.mat*
assumes $A \in$ *carrier-mat n m*
shows *hermitian* ((*Complex-Matrix.adjoint A*) ∗ *A*) ⟨*proof*⟩

lemma *hermitian-square-hermitian*:
fixes *A*::'*a*::*conjugatable-field Matrix.mat*
assumes *hermitian A*
shows *hermitian* (*A* ∗ *A*)
⟨*proof*⟩

2 Properties of the spectrum of a matrix

2.1 Results on diagonal matrices

lemma *diagonal-mat-uminus*:
fixes *A*::'*a*::{*ring*} *Matrix.mat*

assumes *diagonal-mat A*
shows *diagonal-mat* $(-A)$ ⟨*proof*⟩

lemma *diagonal-mat-smult*:
fixes *A*::'*a*::{*ring*} *Matrix.mat*
assumes *diagonal-mat A*
shows *diagonal-mat* $(x \cdot_m A)$ ⟨*proof*⟩

lemma *diagonal-imp-upper-triangular*:
assumes *diagonal-mat A*
and $A \in$ *carrier-mat n n*
shows *upper-triangular A* ⟨*proof*⟩

lemma *set-diag-mat-uminus*:
assumes *A*∈ *carrier-mat n n*
shows *set* (*diag-mat* $(-A)$) = $\{-a \mid a.\ a \in$ *set* (*diag-mat A*)$\}$ (**is** *?L* = *?R*)
⟨*proof*⟩

lemma *set-diag-mat-smult*:
assumes *A*∈ *carrier-mat n n*
shows *set* (*diag-mat* $(x \cdot_m A)$) = $\{x * a \mid a.\ a \in$ *set* (*diag-mat A*)$\}$ (**is** *?L* = *?R*)
⟨*proof*⟩

lemma *diag-mat-diagonal-eq*:
assumes *diag-mat A* = *diag-mat B*
and *diagonal-mat A*
and *diagonal-mat B*
and *dim-col A* = *dim-col B*
shows *A* = *B*
⟨*proof*⟩

lemma *diag-elems-ne*:
assumes $B \in$ *carrier-mat n n*
and $0 < n$
shows *diag-elems B* ≠ {}
⟨*proof*⟩

lemma *diagonal-mat-mult-vec*:
fixes *B*::'*a*::*conjugatable-field Matrix.mat*
assumes *diagonal-mat B*
and $B \in$ *carrier-mat n n*
and *v*∈ *carrier-vec n*
and $i < n$
shows *vec-index* $(B *_v v)$ *i* = *B* $$ $(i,i) * ($*vec-index v i*$)$
⟨*proof*⟩

lemma *diagonal-mat-mult-index*:
fixes B::$'a$::{*ring*} *Matrix.mat*
assumes *diagonal-mat A*
and $A\in$ *carrier-mat n n*
and $B \in$ *carrier-mat n n*
and $i < n$
and $j < n$
shows $(A * B)$ \$\$ $(i,j) = A\$\$(i,i) * B\$\$(i,j)$ ⟨*proof*⟩

lemma *diagonal-mat-mult-index'*:
fixes A::$'a$::*comm-ring Matrix.mat*
assumes $A \in$ *carrier-mat n n*
and $B\in$ *carrier-mat n n*
and *diagonal-mat B*
and $j < n$
and $i < n$
shows $(A*B)$ \$\$$(i,j) = B\$\$(j,j) *A$ \$\$ (i, j)

⟨*proof*⟩

lemma *diagonal-mat-times-diag*:
assumes $A\in$ *carrier-mat n n*
and $B\in$ *carrier-mat n n*
and *diagonal-mat A*
and *diagonal-mat B*
shows *diagonal-mat* $(A*B)$ ⟨*proof*⟩

lemma *diagonal-mat-commute*:
fixes A::$'a$::{*comm-ring*} *Matrix.mat*
assumes $A\in$ *carrier-mat n n*
and $B\in$ *carrier-mat n n*
and *diagonal-mat A*
and *diagonal-mat B*
shows $A * B = B * A$
⟨*proof*⟩

lemma *diagonal-mat-sq-index*:
fixes B::$'a$::{*ring*} *Matrix.mat*
assumes *diagonal-mat B*
and $B \in$ *carrier-mat n n*
and $i < n$
and $j < n$
shows $(B * B)$ \$\$ $(i,j) = B\$\$(i,i) * B\$\$(j,i)$
⟨*proof*⟩

lemma *diagonal-mat-sq-index'*:
fixes B::$'a$::{*ring*} *Matrix.mat*
assumes *diagonal-mat B*
and $B \in$ *carrier-mat n n*

and $i < n$
and $j < n$
shows $(B * B)$ \$\$ $(i,j) = B\$\$(i,j) * B\$\$(i,j)$
⟨*proof*⟩

lemma *diagonal-mat-sq-diag*:
fixes $B::'a::\{ring\}$ *Matrix.mat*
assumes *diagonal-mat B*
and $B \in$ *carrier-mat n n*
shows *diagonal-mat* $(B * B)$ ⟨*proof*⟩

lemma *real-diagonal-hermitian*:
fixes *B*::*complex Matrix.mat*
assumes *B*∈ *carrier-mat n n*
and *diagonal-mat B*
and $\forall\, i <$ *dim-row B*. *B*\$\$(*i*, *i*) ∈ *Reals*
shows *hermitian B* ⟨*proof*⟩

2.2 Unitary equivalence

definition *unitarily-equiv* **where**
unitarily-equiv A B U ≡ (*unitary U* ∧
similar-mat-wit A B U (*Complex-Matrix.adjoint U*))

lemma *unitarily-equivD*:
assumes *unitarily-equiv A B U*
shows *unitary U*
similar-mat-wit A B U (*Complex-Matrix.adjoint U*) ⟨*proof*⟩

lemma *unitarily-equivI*:
assumes *similar-mat-wit A B U* (*Complex-Matrix.adjoint U*)
and *unitary U*
shows *unitarily-equiv A B U* ⟨*proof*⟩

lemma *unitarily-equivI'*:
assumes *A* = *mat-conj U B*
and *unitary U*
and *A*∈ *carrier-mat n n*
and *B*∈ *carrier-mat n n*
shows *unitarily-equiv A B U* ⟨*proof*⟩

lemma *unitarily-equiv-carrier*:
assumes *A*∈ *carrier-mat n n*
and *unitarily-equiv A B U*
shows *B* ∈ *carrier-mat n n U*∈ *carrier-mat n n*
⟨*proof*⟩

lemma *unitarily-equiv-carrier'*:
assumes *unitarily-equiv A B U*

shows $A \in$ *carrier-mat* (*dim-row* A) (*dim-row* A)
$B \in$ *carrier-mat* (*dim-row* A) (*dim-row* A)
$U \in$ *carrier-mat* (*dim-row* A) (*dim-row* A)
⟨*proof*⟩

lemma *unitarily-equiv-eq*:
assumes *unitarily-equiv* A B U
shows $A = U * B * (\textit{Complex-Matrix.adjoint}\ U)$ ⟨*proof*⟩

lemma *unitarily-equiv-smult*:
assumes $A \in$ *carrier-mat* n n
and *unitarily-equiv* A B U
shows *unitarily-equiv* $(x \cdot_m A)$ $(x \cdot_m B)$ U
⟨*proof*⟩

lemma *unitarily-equiv-uminus*:
assumes $A \in$ *carrier-mat* n n
and *unitarily-equiv* A B U
shows *unitarily-equiv* $(-A)$ $(-B)$ U
⟨*proof*⟩

lemma *unitarily-equiv-adjoint*:
assumes *unitarily-equiv* A B U
shows *unitarily-equiv* B A (*Complex-Matrix.adjoint* U)
⟨*proof*⟩

lemma *unitary-mult-conjugate*:
assumes $A \in$ *carrier-mat* n n
and $V \in$ *carrier-mat* n n
and $U \in$ *carrier-mat* n n
and $B \in$ *carrier-mat* n n
and *unitary* V
and *mat-conj* (*Complex-Matrix.adjoint* V) A = *mat-conj* U B
shows $A = V * U * B *$ *Complex-Matrix.adjoint* $(V * U)$
⟨*proof*⟩

lemma *unitarily-equiv-conjugate*:
assumes $A \in$ *carrier-mat* n n
and $V \in$ *carrier-mat* n n
and $U \in$ *carrier-mat* n n
and $B \in$ *carrier-mat* n n
and *unitarily-equiv* (*mat-conj* (*Complex-Matrix.adjoint* V) A) B U
and *unitary* V
shows *unitarily-equiv* A B $(V * U)$
⟨*proof*⟩

lemma *mat-conj-commute*:
assumes $A \in$ *carrier-mat* n n
and $B \in$ *carrier-mat* n n

and $U \in$ *carrier-mat n n*
and *unitary U*
and $A*B = B*A$
shows (*mat-conj* (*Complex-Matrix.adjoint U*) *A*) $*$
(*mat-conj* (*Complex-Matrix.adjoint U*) *B*) =
(*mat-conj* (*Complex-Matrix.adjoint U*) *B*) $*$
(*mat-conj* (*Complex-Matrix.adjoint U*) *A*) (**is** *?L*?R* = *?R* ?L*)
⟨*proof*⟩

lemma *unitarily-equiv-commute*:
assumes *unitarily-equiv A B U*
and $A*C = C*A$
shows $B * (\textit{Complex-Matrix.adjoint } U * C * U) =$
Complex-Matrix.adjoint U $*$ *C* $*$ *U* $*$ *B*
⟨*proof*⟩

definition *unitary-diag* **where**
unitary-diag A B U $\equiv$ *unitarily-equiv A B U* $\wedge$ *diagonal-mat B*

lemma *unitary-diagI*:
assumes *similar-mat-wit A B U* (*Complex-Matrix.adjoint U*)
and *diagonal-mat B*
and *unitary U*
shows *unitary-diag A B U* ⟨*proof*⟩

lemma *unitary-diagI'*:
assumes $A\in$ *carrier-mat n n*
and $B\in$ *carrier-mat n n*
and *diagonal-mat B*
and *unitary U*
and $A =$ *mat-conj U B*
shows *unitary-diag A B U* ⟨*proof*⟩

lemma *unitary-diagD*:
assumes *unitary-diag A B U*
shows *similar-mat-wit A B U* (*Complex-Matrix.adjoint U*)
diagonal-mat B unitary U ⟨*proof*⟩

lemma *unitary-diag-imp-unitarily-equiv*[*simp*]:
assumes *unitary-diag A B U*
shows *unitarily-equiv A B U* ⟨*proof*⟩

lemma *unitary-diag-diagonal*[*simp*]:
assumes *unitary-diag A B U*
shows *diagonal-mat B* ⟨*proof*⟩

lemma *unitary-diag-carrier*:
assumes $A\in$ *carrier-mat n n*
and *unitary-diag A B U*

shows $B \in$ *carrier-mat n n* $U \in$ *carrier-mat n n*
⟨*proof*⟩

lemma *unitary-mult-square-eq*:
assumes $A \in$ *carrier-mat n n*
and $U \in$ *carrier-mat n n*
and $B \in$ *carrier-mat n n*
and $A =$ *mat-conj U B*
and (*Complex-Matrix.adjoint U*) $*$ $U = 1_m$ *n*
shows $A * A =$ *mat-conj U* $(B*B)$
⟨*proof*⟩

lemma *hermitian-square-similar-mat-wit*:
fixes *A*::*complex Matrix.mat*
assumes *hermitian A*
and $A \in$ *carrier-mat n n*
and *unitary-diag A B U*
shows *similar-mat-wit* $(A * A)$ $(B * B)$ *U* (*Complex-Matrix.adjoint U*)
⟨*proof*⟩

lemma *unitarily-equiv-square*:
assumes $A \in$ *carrier-mat n n*
and *unitarily-equiv A B U*
shows *unitarily-equiv* $(A*A)$ $(B*B)$ *U*
⟨*proof*⟩

lemma *conjugate-eq-unitarily-equiv*:
assumes $A \in$ *carrier-mat n n*
and $V \in$ *carrier-mat n n*
and *unitarily-equiv A B U*
and *unitary V*
and $V * B *$ (*Complex-Matrix.adjoint V*) $= B$
shows *unitarily-equiv A B* $(U*V)$
⟨*proof*⟩

definition *real-diag-decomp* **where**
real-diag-decomp A B U $\equiv$ *unitary-diag A B U* $\wedge$
$(\forall\, i <$ *dim-row B*. *B*$$(i, i) \in$ *Reals*)

lemma *real-diag-decompD*[*simp*]:
assumes *real-diag-decomp A B U*
shows *unitary-diag A B U*
$(\forall\, i <$ *dim-row B*. *B*$$(i, i) \in$ *Reals*) ⟨*proof*⟩

lemma *hermitian-decomp-decomp′*:
fixes *A*::*complex Matrix.mat*
assumes *hermitian-decomp A B U*
shows *real-diag-decomp A B U*

⟨*proof*⟩

lemma *real-diag-decomp-hermitian*:
fixes *A*::*complex Matrix.mat*
assumes *real-diag-decomp A B U*
shows *hermitian A*
⟨*proof*⟩

lemma *unitary-conjugate-real-diag-decomp*:
assumes *A*∈ *carrier-mat n n*
and *Us*∈ *carrier-mat n n*
and *unitary Us*
and *real-diag-decomp* (*mat-conj* (*Complex-Matrix.adjoint Us*) *A*) *B U*
shows *real-diag-decomp A B* (*Us* ∗ *U*) ⟨*proof*⟩

2.3 On the spectrum of a matrix

lemma *similar-spectrum-eq*:
fixes *A*::*complex Matrix.mat*
assumes *A*∈ *carrier-mat n n*
and *similar-mat A B*
and *upper-triangular B*
shows *spectrum A* = *set* (*diag-mat B*)
⟨*proof*⟩

lemma *unitary-diag-spectrum-eq*:
fixes *A*::*complex Matrix.mat*
assumes *A*∈ *carrier-mat n n*
and *unitary-diag A B U*
shows *spectrum A* = *set* (*diag-mat B*)
⟨*proof*⟩

lemma *unitary-diag-spectrum-eq′*:
fixes *A*::*complex Matrix.mat*
assumes *A*∈ *carrier-mat n n*
and *unitary-diag A B U*
shows *spectrum A* = *diag-elems B*
⟨*proof*⟩

lemma *hermitian-real-diag-decomp*:
fixes *A*::*complex Matrix.mat*
assumes *A*∈ *carrier-mat n n*
and *0* < *n*
and *hermitian A*
obtains *B U* **where** *real-diag-decomp A B U*
⟨*proof*⟩

lemma *spectrum-smult*:
fixes *A*::*complex Matrix.mat*

assumes *hermitian A*
and $A \in$ *carrier-mat n n*
and $0 < n$
shows *spectrum* $(x \cdot_m A) = \{x * a | a.\ a \in$ *spectrum* $A\}$
⟨*proof*⟩

lemma *spectrum-uminus*:
fixes A::*complex Matrix.mat*
assumes *hermitian A*
and $A \in$ *carrier-mat n n*
and $0 < n$
shows *spectrum* $(-A) = \{-a | a.\ a \in$ *spectrum* $A\}$
⟨*proof*⟩

3 Properties of the inner product

3.1 Some analysis complements

lemma *add-conj-le*:
shows $z +$ *cnj* $z \leq 2 *$ *cmod* z
⟨*proof*⟩

lemma *abs-real*:
fixes x::*complex*
assumes $x \in$ *Reals*
shows *abs* $x \in$ *Reals* ⟨*proof*⟩

lemma *csqrt-cmod-square*:
shows *csqrt* $((cmod\ z)^2) =$ *cmod* z
⟨*proof*⟩

lemma *cpx-real-le*:
fixes z::*complex*
assumes $0 \leq z$
and $0 \leq u$
and $z^2 \leq u^2$
shows $z \leq u$
⟨*proof*⟩

lemma *mult-conj-real*:
fixes v::*complex*
shows $v *$ (*conjugate* v) $\in$ *Reals*
⟨*proof*⟩

lemma *real-sum-real*:
assumes $\bigwedge i.\ i < n \Longrightarrow ((f\ i)$::*complex*$) \in$ *Reals*
shows $(\sum\ i \in \{0\ ..<\ n\}.\ f\ i) \in$ *Reals*
⟨*proof*⟩

lemma *real-mult-re*:
assumes $a \in$ *Reals* **and** $b \in$ *Reals*
shows *Re* $(a * b) =$ *Re* $a *$ *Re* b ⟨*proof*⟩

lemma *complex-positive-Im*:
fixes b::*complex*
assumes $0 \leq b$
shows *Im* $b = 0$
⟨*proof*⟩

lemma *cmod-pos*:
fixes z::*complex*
assumes $0 \leq z$
shows *cmod* $z = z$
⟨*proof*⟩

lemma *cpx-pos-square-pos*:
fixes z::*complex*
assumes $0 \leq z$
shows $0 \leq z^2$
⟨*proof*⟩

lemma *cmod-mult-pos*:
fixes b:: *complex*
fixes z::*complex*
assumes $0 \leq b$
shows *cmod* $(b * z) =$ *Re* $b *$ *cmod* z ⟨*proof*⟩

lemma *cmod-conjugate-square-eq*:
fixes z::*complex*
shows *cmod* $(z *$ (*conjugate* $z)) = z *$ (*conjugate* $z)$
⟨*proof*⟩

lemma *pos-sum-gt-comp*:
assumes *finite* I
and $\bigwedge i.\ i \in I \Longrightarrow (0$::*real*$) \leq f\ i$
and $j \in I$
and $c < f\ j$
shows $c <$ *sum* $f\ I$
⟨*proof*⟩

lemma *pos-sum-le-comp*:
assumes *finite* I
and $\bigwedge i.\ i \in I \Longrightarrow (0$::*real*$) \leq f\ i$
and *sum* $f\ I \leq c$

shows $\forall\, i \in I.\ f\ i \leq c$
⟨*proof*⟩

lemma *square-pos-mult-le*:
assumes *finite I*
and $\bigwedge i.\ i \in I \Longrightarrow ((0::real) \leq f\ i \wedge f\ i \leq 1)$
shows *sum* $(\lambda x.\ f\ x * f\ x)\ I \leq$ *sum f I* ⟨*proof*⟩

lemma *square-pos-mult-lt*:
assumes *finite I*
and $\bigwedge i.\ i \in I \Longrightarrow ((0::real) \leq f\ i \wedge f\ i \leq 1)$
and $j \in I$
and $f\ j < 1$
and $0 < f\ j$
shows *sum* $(\lambda x.\ f\ x * f\ x)\ I <$ *sum f I* ⟨*proof*⟩

3.2 Inner product results

In particular we prove the triangle inequality, i.e. that for vectors u and v we have $\|u + v\| \leq \|u\| + \|v\|$.

lemma *inner-prod-vec-norm-pow2*:
shows $(\textit{vec-norm}\ v)^2 = v \cdot_c v$ ⟨*proof*⟩

lemma *inner-prod-mult-mat-vec-left*:
assumes $v \in$ *carrier-vec n*
and $w \in$ *carrier-vec n'*
and $A \in$ *carrier-mat m n*
and $B \in$ *carrier-mat m n'*
shows *inner-prod* $(A *_v v)\ (B *_v w) =$
inner-prod $(((\textit{Complex-Matrix.adjoint}\ B) * A) *_v v)\ w$
⟨*proof*⟩

lemma *rank-1-proj-trace-inner*:
fixes $A :: 'a$::*conjugatable-field Matrix.mat* **and** $v :: 'a$ *Matrix.vec*
assumes *A*: $A \in$ *carrier-mat n n*
and *v*: $v \in$ *carrier-vec n*
shows *Complex-Matrix.trace* $(A * (\textit{rank-1-proj}\ v)) =$ *Complex-Matrix.inner-prod* $v\ (A *_v v)$
⟨*proof*⟩

lemma *unitary-inner-prod*:
assumes $v \in$ *carrier-vec n*
and $w \in$ *carrier-vec n*
and $U \in$ *carrier-mat n n*
and *Complex-Matrix.unitary U*
shows *inner-prod* $(U *_v v)\ (U *_v w) =$ *inner-prod v w*

⟨*proof*⟩

lemma *unitary-vec-norm*:
assumes $v \in$ *carrier-vec n*
and $U \in$ *carrier-mat n n*
and *Complex-Matrix.unitary U*
shows *vec-norm* $(U *_v v) =$ *vec-norm v* ⟨*proof*⟩

lemma *unitary-col-norm-square*:
assumes *unitary U*
and $U \in$ *carrier-mat n n*
and $i < n$
shows $\|$*Matrix.col U i*$\|^2 = 1$
⟨*proof*⟩

lemma *unitary-col-norm*:
assumes *unitary U*
and $U \in$ *carrier-mat n n*
and $i < n$
shows $\|$*Matrix.col U i*$\| = 1$ ⟨*proof*⟩

lemma *inner-mult-diag-expand*:
fixes *B*::*complex Matrix.mat*
assumes *diagonal-mat B*
and $B \in$ *carrier-mat n n*
and $v \in$ *carrier-vec n*
shows *inner-prod* $(B *_v v)$ *v* =
$(\sum i \in \{0 ..< n\}.$ (*conjugate* (*B* $$ (*i*,*i*))) $*$ (*vec-index v i* $*$
(*conjugate* (*vec-index v i*))))
⟨*proof*⟩

lemma *inner-mult-diag-expand′*:
fixes *B*::*complex Matrix.mat*
assumes *diagonal-mat B*
and $B \in$ *carrier-mat n n*
and $v \in$ *carrier-vec n*
shows *inner-prod v* $(B *_v v)$ =
$(\sum i \in \{0 ..< n\}.$ *B* $$ (*i*,*i*) $*$ (*vec-index v i* $*$
(*conjugate* (*vec-index v i*))))
⟨*proof*⟩

lemma *self-inner-prod-real*:
fixes *v*::*complex Matrix.vec*
shows *Complex-Matrix.inner-prod v v* $\in$ *Reals*
⟨*proof*⟩

lemma *inner-mult-diag-real*:
fixes *B*::*complex Matrix.mat*
assumes *diagonal-mat B*

and $B \in$ *carrier-mat n n*
and $\forall\, i < n.$ *B$$(i, i)* $\in$ *Reals*
and $v \in$ *carrier-vec n*
shows *inner-prod* $(B *_v v)$ $v \in$ *Reals*
⟨*proof*⟩

lemma *inner-mult-diag-real′*:
fixes *B*::*complex Matrix.mat*
assumes *diagonal-mat B*
and $B \in$ *carrier-mat n n*
and $\forall\, i < n.$ *B$$(i, i)* $\in$ *Reals*
and $v \in$ *carrier-vec n*
shows *inner-prod v* $(B *_v v)$ $\in$ *Reals*
⟨*proof*⟩

lemma *inner-prod-mult-mat-vec-right*:
assumes $v \in$ *carrier-vec n*
and $w \in$ *carrier-vec n′*
and $A \in$ *carrier-mat m n*
and $B \in$ *carrier-mat m n′*
shows *inner-prod* $(A *_v v)$ $(B *_v w) =$
inner-prod v (((*Complex-Matrix.adjoint A*) $*$ *B*) $*_v$ *w*)
⟨*proof*⟩

lemma *Cauchy-Schwarz-complex-vec-norm*:
assumes *dim-vec x* $=$ *dim-vec y*
shows *cmod* (*inner-prod x y*) $\leq$ *vec-norm x* $*$ *vec-norm y*
⟨*proof*⟩

lemma *vec-norm-triangle-sq*:
fixes *u*::*complex Matrix.vec*
assumes *dim-vec u* $=$ *dim-vec v*
shows (*vec-norm* $(u+ v))^2 \leq$ (*vec-norm u* $+$ *vec-norm v*)^2
⟨*proof*⟩

lemma *vec-norm-triangle*:
fixes *u*::*complex Matrix.vec*
assumes *dim-vec u* $=$ *dim-vec v*
shows *vec-norm* $(u+ v) \leq$ *vec-norm u* $+$ *vec-norm v*
⟨*proof*⟩

4 Matrix decomposition

lemma (**in** *cpx-sq-mat*) *sum-decomp-cols*:
fixes *A*::*complex Matrix.mat*
assumes *hermitian A*
and $A \in$ *fc-mats*
and *unitary-diag A B U*
shows *sum-mat* $(\lambda i.$ (*diag-mat B* ! *i*) $\cdot_m$ *rank-1-proj* (*Matrix.col U i*))

$\{..< dimR\} = A$
⟨*proof*⟩

lemma *unitary-col-inner-prod*:
assumes $A \in$ *carrier-mat n n*
and $0 < n$
and *Complex-Matrix.unitary A*
and $j < n$
and $k < n$
shows *Complex-Matrix.inner-prod* (*Matrix.col A j*) (*Matrix.col A k*) =
$(1_m\ n)$ $$ (j,k)
⟨*proof*⟩

lemma (**in** *cpx-sq-mat*) *sum-mat-ortho-proj*:
assumes *finite I*
and $j \in I$
and $A\ j * A\ j = A\ j$
and $\bigwedge i.\ i \in I \Longrightarrow A\ i \in$ *fc-mats*
and $\bigwedge\ i\ .\ i \in I \Longrightarrow i \neq j \Longrightarrow A\ i * (A\ j) = (0_m\ dimR\ dimR)$
shows $(sum\text{-}mat\ A\ I) * (A\ j) = (A\ j)$ ⟨*proof*⟩

lemma (**in** *cpx-sq-mat*) *sum-mat-ortho-one*:
assumes *finite I*
and $j \in I$
and $B \in$ *fc-mats*
and $\bigwedge i.\ i \in I \Longrightarrow A\ i \in$ *fc-mats*
and $\bigwedge\ i\ .\ i \in I \Longrightarrow i \neq j \Longrightarrow A\ i * B = (0_m\ dimR\ dimR)$
shows $(sum\text{-}mat\ A\ I) * B = A\ j * B$ ⟨*proof*⟩

lemma *unitarily-equiv-rank-1-proj-col-carrier*:
assumes $A \in$ *carrier-mat n n*
and *unitarily-equiv A B U*
and $i < n$
shows *rank-1-proj* (*Matrix.col U i*) $\in$ *carrier-mat n n*
⟨*proof*⟩

lemma *decomp-eigenvector*:
fixes *A*::*complex Matrix.mat*
assumes $A \in$ *carrier-mat n n*
and $0 < n$
and *hermitian A*
and *unitary-diag A B U*
and $j < n$
shows *Complex-Matrix.trace* $(A * (rank\text{-}1\text{-}proj\ (Matrix.col\ U\ j))) = B$ $$(j,j)
⟨*proof*⟩

lemma *positive-unitary-diag-pos*:
fixes *A*::*complex Matrix.mat*
assumes $A \in$ *carrier-mat n n*

and *Complex-Matrix.positive A*
and *unitary-diag A B U*
and $j < n$
shows $0 \leq B\ \$\$\ (j, j)$
⟨*proof*⟩

lemma *unitary-diag-trace-mult-sum*:
fixes *A*::*complex Matrix.mat*
assumes $A \in$ *carrier-mat n n*
and $C \in$ *carrier-mat n n*
and *hermitian A*
and *unitary-diag A B U*
and $0 < n$
shows *Complex-Matrix.trace* $(C * A) =$
$(\sum\ i = 0\ ..<\ n.\ B\$\$(i,i) *$
Complex-Matrix.trace $(C *$ *rank-1-proj* $($*Matrix.col U i*$)))$
⟨*proof*⟩

lemma *unitarily-equiv-trace*:
assumes $A \in$ *carrier-mat n n*
and *unitarily-equiv A B U*
shows *Complex-Matrix.trace A* = *Complex-Matrix.trace B*
⟨*proof*⟩

lemma *unitarily-equiv-trace′*:
assumes $A \in$ *carrier-mat n n*
and *unitarily-equiv A B U*
shows *Complex-Matrix.trace A* $= (\sum\ i = 0\ ..<$ *dim-row A*. *B* $\$\$\ (i,i))$
⟨*proof*⟩

lemma *positive-decomp-cmod-le*:
fixes *A*::*complex Matrix.mat*
assumes *A*∈ *carrier-mat n n*
and *C*∈ *carrier-mat n n*
and $0 < n$
and *Complex-Matrix.positive A*
and *unitary-diag A B U*
and $\bigwedge i.\ i < n \Longrightarrow$ *cmod* $($*Complex-Matrix.trace* $(C *$ *rank-1-proj* $($*Matrix.col U i*$))) \leq M$
shows *cmod* $($*Complex-Matrix.trace* $(C * A)) \leq$ *Re* $($*Complex-Matrix.trace A*$) *$ *M*
⟨*proof*⟩
end

theory *Commuting-Hermitian* **imports** *Spectral-Theory-Complements Commuting-Hermitian-Misc*
Projective-Measurements.Linear-Algebra-Complements
Projective-Measurements.Projective-Measurements **begin**

5 Additional results on block decompositions of matrices

5.1 Split block results

lemma *split-block-diag-carrier*:
assumes $D \in$ *carrier-mat n n*
and $a \leq n$
and *split-block D a a* $= (D1, D2, D3, D4)$
shows $D1 \in$ *carrier-mat a a* $D4 \in$ *carrier-mat* $(n-a)$ $(n-a)$
⟨*proof*⟩

lemma *split-block-diagonal*:
assumes *diagonal-mat D*
and $D \in$ *carrier-mat n n*
and $a \leq n$
and *split-block D a a* $= (D1, D2, D3, D4)$
shows *diagonal-mat D1* $\wedge$ *diagonal-mat D4* ⟨*proof*⟩

lemma *split-block-times-diag-index*:
fixes *B*::'*a*::*comm-ring Matrix.mat*
assumes *diagonal-mat D*
and $D \in$ *carrier-mat n n*
and $B \in$ *carrier-mat n n*
and $a \leq n$
and *split-block B a a* $= (B1, B2, B3, B4)$
and *split-block D a a* $= (D1, D2, D3, D4)$
and $i <$ *dim-row* $(D4 * B4)$
and $j <$ *dim-col* $(D4 * B4)$
shows $(B4 * D4)$ \$\$ $(i, j) = (B*D)$ \$\$ $(i+a, j+a)$
$(D4 * B4)$ \$\$ $(i, j) = (D*B)$ \$\$ $(i+a, j+a)$
⟨*proof*⟩

lemma *split-block-commute-subblock*:
fixes *B*::'*a*::*comm-ring Matrix.mat*
assumes *diagonal-mat D*
and $D \in$ *carrier-mat n n*
and $B \in$ *carrier-mat n n*
and $a \leq n$
and *split-block B a a* $= (B1, B2, B3, B4)$
and *split-block D a a* $= (D1, D2, D3, D4)$
and $B * D = D * B$
shows $B4 * D4 = D4 * B4$
⟨*proof*⟩

lemma *commute-diag-mat-zero-comp*:
fixes *D*::'*a*::{*field*} *Matrix.mat*
assumes *diagonal-mat D*
and $D \in$ *carrier-mat n n*

and $B \in$ *carrier-mat n n*
and $B * D = D * B$
and $i < n$
and $j < n$
and $D\$\$(i,i) \neq D\$\$(j,j)$
shows $B\ \$\$(i,j) = 0$
⟨*proof*⟩

lemma *commute-diag-mat-split-block*:
fixes $D::'a::\{field\}$ *Matrix.mat*
assumes *diagonal-mat D*
and $D \in$ *carrier-mat n n*
and $B \in$ *carrier-mat n n*
and $B * D = D * B$
and $k \leq n$
and $\forall\, i\ j.\ (i < k \wedge k \leq j \wedge j < n) \longrightarrow D\$\$(i,i) \neq D\$\$(j,j)$
and $(B1,\ B2,\ B3,\ B4) =$ *split-block B k k*
shows $B2 = 0_m\ k\ (n-k)\ B3 = 0_m\ (n-k)\ k$
⟨*proof*⟩

lemma *split-block-hermitian-1*:
assumes *hermitian A*
and $n \leq$ *dim-row A*
and $(A1,\ A2,\ A3,\ A4) =$ *split-block A n n*
shows *hermitian A1* ⟨*proof*⟩

lemma *split-block-hermitian-4*:
assumes *hermitian A*
and $n \leq$ *dim-row A*
and $(A1,\ A2,\ A3,\ A4) =$ *split-block A n n*
shows *hermitian A4* ⟨*proof*⟩

lemma *diag-block-split-block*:
assumes $B \in$ *carrier-mat n n*
and $k < n$
and $(B1,\ B2,\ B3,\ B4) =$ *split-block B k k*
and $B2 = 0_m\ k\ (n-k)$
and $B3 = 0_m\ (n-k)\ k$
shows $B =$ *diag-block-mat* $[B1,B4]$
⟨*proof*⟩

5.2 Diagonal block matrices

abbreviation *four-block-diag* **where**
four-block-diag B1 B2 ≡
(*four-block-mat B1* (0_m (*dim-row B1*) (*dim-col B2*))
(0_m (*dim-row B2*) (*dim-col B1*)) *B2*)

lemma *four-block-diag-cong-comp*:

assumes *dim-row A1 = dim-row B1*
and *dim-col A1 = dim-col B1*
and *four-block-diag A1 A2 = four-block-diag B1 B2*
shows *A1 = B1*
⟨*proof*⟩

lemma *four-block-diag-cong-comp'*:
assumes *dim-row A1 = dim-row B1*
and *dim-col A1 = dim-col B1*
and *four-block-diag A1 A2 = four-block-diag B1 B2*
shows *A2 = B2*
⟨*proof*⟩

lemma *four-block-mat-real-diag*:
assumes ∀ *i* < *dim-row B1*. *B1*$$(*i*,*i*) ∈ *Reals*
and ∀ *i* < *dim-row B2*. *B2*$$(*i*,*i*) ∈ *Reals*
and *dim-row B1 = dim-col B1*
and *dim-row B2 = dim-col B2*
and *i* < *dim-row* (*four-block-diag B1 B2*)
shows (*four-block-diag B1 B2*) $$ (*i*,*i*) ∈ *Reals*
⟨*proof*⟩

lemma *four-block-diagonal*:
assumes *dim-row B1 = dim-col B1*
and *dim-row B2 = dim-col B2*
and *diagonal-mat B1*
and *diagonal-mat B2*
shows *diagonal-mat* (*four-block-diag B1 B2*) ⟨*proof*⟩

lemma *four-block-diag-zero*:
assumes *B*∈ *carrier-mat 0 0*
shows *four-block-diag A B = A*
⟨*proof*⟩

lemma *four-block-diag-zero'*:
assumes *B*∈ *carrier-mat 0 0*
shows *four-block-diag B A = A*
⟨*proof*⟩

lemma *mult-four-block-diag*:
assumes *A1* ∈ *carrier-mat nr1 n1 D1* ∈ *carrier-mat nr2 n2*
and *A2* ∈ *carrier-mat n1 nc1 D2* ∈ *carrier-mat n2 nc2*
shows *four-block-diag A1 D1* ∗
four-block-diag A2 D2
= *four-block-diag* (*A1* ∗ *A2*) (*D1* ∗ *D2*)
⟨*proof*⟩

lemma *four-block-diag-adjoint*:
shows (*Complex-Matrix.adjoint* (*four-block-diag A1 A2*)) =

(*four-block-diag* (*Complex-Matrix.adjoint A1*)
(*Complex-Matrix.adjoint A2*))
⟨*proof*⟩

lemma *four-block-diag-unitary*:
assumes *unitary U1*
and *unitary U2*
shows *unitary*
(*four-block-diag U1 U2*)
(**is** *unitary ?fU*)
⟨*proof*⟩

lemma *four-block-diag-similar*:
assumes *unitarily-equiv A1 B1 U1*
and *unitarily-equiv A2 B2 U2*
and *dim-row A1* = *dim-col A1*
and *dim-row A2* = *dim-col A2*
shows *similar-mat-wit*
(*four-block-diag A1 A2*)
(*four-block-diag B1 B2*)
(*four-block-diag U1 U2*)
(*Complex-Matrix.adjoint* (*four-block-diag U1 U2*))
⟨*proof*⟩

lemma *four-block-unitarily-equiv*:
assumes *unitarily-equiv A1 B1 U1*
and *unitarily-equiv A2 B2 U2*
and *dim-row A1* = *dim-col A1*
and *dim-row A2* = *dim-col A2*
shows *unitarily-equiv*
(*four-block-diag A1 A2*)
(*four-block-diag B1 B2*)
(*four-block-diag U1 U2*)
(**is** *unitarily-equiv ?fA ?fB ?fU*)
⟨*proof*⟩

lemma *four-block-unitary-diag*:
assumes *unitary-diag A1 B1 U1*
and *unitary-diag A2 B2 U2*
and *dim-row A1* = *dim-col A1*
and *dim-row A2* = *dim-col A2*
shows *unitary-diag*
(*four-block-diag A1 A2*)
(*four-block-diag B1 B2*)
(*four-block-diag U1 U2*)
(**is** *unitary-diag ?fA ?fB ?fU*)
⟨*proof*⟩

lemma *four-block-real-diag-decomp*:

assumes *real-diag-decomp A1 B1 U1*
and *real-diag-decomp A2 B2 U2*
and *dim-row A1 = dim-col A1*
and *dim-row A2 = dim-col A2*
shows *real-diag-decomp*
(*four-block-diag A1 A2*)
(*four-block-diag B1 B2*)
(*four-block-diag U1 U2*)
(**is** *real-diag-decomp ?fA ?fB ?fU*)
⟨*proof*⟩

lemma *diag-block-mat-mult*:
assumes *length Al = length Bl*
and ∀ *i* < *length Al*. *dim-col* (*Al*!*i*) = *dim-row* (*Bl*!*i*)
shows *diag-block-mat Al* ∗ (*diag-block-mat Bl*) =
(*diag-block-mat* (*map2* (∗) *Al Bl*)) ⟨*proof*⟩

lemma *real-diag-decomp-block*:
fixes *Al*::*complex Matrix.mat list*
assumes *Al* ≠ []
and *list-all* (λ*A*. *0* < *dim-row A* ∧ *hermitian A*) *Al*
shows ∃ *Bl Ul*. *length Ul* = *length Al* ∧
(∀ *i* < *length Al*.
Ul!*i* ∈ *carrier-mat* (*dim-row* (*Al*!*i*)) (*dim-col* (*Al*!*i*)) ∧ *unitary* (*Ul*!*i*) ∧
Bl!*i* ∈ *carrier-mat* (*dim-row* (*Al*!*i*)) (*dim-col* (*Al*!*i*))) ∧
real-diag-decomp (*diag-block-mat Al*) (*diag-block-mat Bl*) (*diag-block-mat Ul*)
⟨*proof*⟩

lemma *diag-block-mat-adjoint*:
shows *Complex-Matrix.adjoint* (*diag-block-mat Al*) =
diag-block-mat (*map Complex-Matrix.adjoint Al*)
⟨*proof*⟩

lemma *diag-block-mat-mat-conj*:
assumes *length Al = length Bl*
and ∀ *i* < *length Al*. *dim-col* (*Al*!*i*) = *dim-row* (*Bl*!*i*)
and ∀ *i* < *length Al*. *dim-row* (*Bl*!*i*) = *dim-col* (*Bl*!*i*)
shows *mat-conj* (*diag-block-mat Al*) (*diag-block-mat Bl*) =
diag-block-mat (*map2 mat-conj Al Bl*)
⟨*proof*⟩

lemma *diag-block-mat-commute*:
assumes *length Al = length Bl*
and ∀ *i* < *length Al*. *Al*!*i* ∗ (*Bl*!*i*) = *Bl*!*i* ∗ (*Al*!*i*)
and ∀ *i*<*length Al*. *dim-col* (*Al* ! *i*) = *dim-row* (*Bl* ! *i*)
and ∀ *i*<*length Al*. *dim-col* (*Bl* ! *i*) = *dim-row* (*Al* ! *i*)
shows *diag-block-mat Al* ∗ (*diag-block-mat Bl*) =
diag-block-mat Bl ∗ (*diag-block-mat Al*)
⟨*proof*⟩

lemma *diag-block-mat-length-1*:
assumes *length Al = 1*
shows *diag-block-mat Al = Al!0*
⟨*proof*⟩

lemma *diag-block-mat-cong-hd*:
assumes *0 < length Al*
and *length Al = length Bl*
and *dim-row (hd Al) = dim-row (hd Bl)*
and *dim-col (hd Al) = dim-col (hd Bl)*
and *diag-block-mat Al = diag-block-mat Bl*
shows *hd Al = hd Bl*
⟨*proof*⟩

lemma *diag-block-mat-cong-tl*:
assumes *0 < length Al*
and *length Al = length Bl*
and *dim-row (hd Al) = dim-row (hd Bl)*
and *dim-col (hd Al) = dim-col (hd Bl)*
and *diag-block-mat Al = diag-block-mat Bl*
shows *diag-block-mat (tl Al) = diag-block-mat (tl Bl)*
⟨*proof*⟩

lemma *diag-block-mat-cong-comp*:
assumes *length Al = length Bl*
and ∀ *i*<*length Al*. *dim-row (Al ! i) = dim-row (Bl ! i)*
and ∀ *i*<*length Al*. *dim-col (Al ! i) = dim-col (Bl ! i)*
and *diag-block-mat Al = diag-block-mat Bl*
and *j < length Al*
shows *Al!j = Bl!j* ⟨*proof*⟩

lemma *diag-block-mat-commute-comp*:
assumes *length Al = length Bl*
and ∀ *i*<*length Al*. *dim-row (Al ! i) = dim-col (Al ! i)*
and ∀ *i*<*length Al*. *dim-row (Al ! i) = dim-row (Bl ! i)*
and ∀ *i*<*length Al*. *dim-col (Al ! i) = dim-col (Bl ! i)*
and *diag-block-mat Al ∗ (diag-block-mat Bl) =*
diag-block-mat Bl ∗ (diag-block-mat Al)
and *i < length Al*
shows *Al!i ∗ Bl!i = Bl!i ∗ Al!i*
⟨*proof*⟩

lemma *diag-block-mat-dim-row-cong*:
assumes *length Ul = length Bl*
and ∀ *i < length Bl*. *dim-row (Bl!i) = dim-row (Ul!i)*
shows *dim-row (diag-block-mat Ul) = dim-row (diag-block-mat Bl)*
⟨*proof*⟩

lemma *diag-block-mat-dim-col-cong*:
assumes *length Ul* = *length Bl*
and ∀ *i* < *length Bl*. *dim-col* (*Bl*!*i*) = *dim-col* (*Ul*!*i*)
shows *dim-col* (*diag-block-mat Ul*) = *dim-col* (*diag-block-mat Bl*)
⟨*proof*⟩

lemma *diag-block-mat-dim-row-col-eq*:
assumes ∀ *i* < *length Al*. *dim-row* (*Al*!*i*) = *dim-col* (*Al*!*i*)
shows *dim-row* (*diag-block-mat Al*) = *dim-col* (*diag-block-mat Al*)
⟨*proof*⟩

6 Block matrix decomposition

6.1 Subdiagonal extraction

`extract_subdiags` returns a list of diagonal sub-blocks, the sizes of which are specified by the list of integers provided as parameters.

fun *extract-subdiags* **where**
extract-subdiags B [] = []
| *extract-subdiags B* (*x*#*xs*) =
(*let* (*B1*, *B2*, *B3*, *B4*) = (*split-block B x x*) *in*
B1 # (*extract-subdiags B4 xs*))

lemma *extract-subdiags-not-emp*:
fixes *x*::*nat* **and** *l*::*nat list*
assumes (*B1*, *B2*, *B3*, *B4*) = (*split-block B x x*)
shows *hd* (*extract-subdiags B* (*x*#*l*)) = *B1*
tl (*extract-subdiags B* (*x*#*l*)) = *extract-subdiags B4 l*
⟨*proof*⟩

lemma *extract-subdiags-neq-Nil*:
shows *extract-subdiags B* (*a*#*l*) ≠ []
⟨*proof*⟩

lemma *extract-subdiags-length*:
shows *length* (*extract-subdiags B l*) = *length l*
⟨*proof*⟩

lemma *extract-subdiags-carrier*:
assumes *i* < *length l*
shows (*extract-subdiags B l*)!*i* ∈ *carrier-mat* (*l*!*i*) (*l*!*i*) ⟨*proof*⟩

lemma *extract-subdiags-diagonal*:
assumes *diagonal-mat B*
and *B* ∈ *carrier-mat n n*
and *l* ≠ []
and *sum-list l* ≤ *n*
and *i* < *length l*

shows *diagonal-mat* ((*extract-subdiags B l*)!*i*) ⟨*proof*⟩

lemma *extract-subdiags-diag-elem*:
fixes *B*::*complex Matrix.mat*
assumes *B*∈ *carrier-mat n n*
and *0* < *n*
and *l* ≠ []
and *i* < *length l*
and *j*< *l*!*i*
and *sum-list l* ≤ *n*
and ∀ *j* < *length l*. *0* < *l*!*j*
shows *extract-subdiags B l*!*i* $$ (*j*,*j*) =
diag-mat B!(*n-sum i l* + *j*) ⟨*proof*⟩

lemma *hermitian-extract-subdiags*:
assumes *hermitian A*
and *sum-list l* ≤ *dim-row A*
and *list-all* (λ*a*. *0* < *a*) *l*
shows *list-all* (λ*B*. *0* < *dim-row B* ∧ *hermitian B*) (*extract-subdiags A l*)
⟨*proof*⟩

6.2 Predicates on diagonal block matrices

The predicate `diag_compat` ensures that the provided matrix, when decomposed according to the list of integers provided as an input, is indeed a diagonal block matrix.

fun *diag-compat* **where**
diag-compat B [] = (*dim-row B* = *0* ∧ *dim-col B* = *0*)
| *diag-compat B* (*x*#*xs*) =
(*x* ≤ *dim-row B* ∧
(*let n* = *dim-row B*; (*B1*, *B2*, *B3*, *B4*) = (*split-block B x x*) *in*
B2 = (*0*$_m$ *x* (*n* − *x*)) ∧ *B3* = (*0*$_m$ (*n* − *x*) *x*) ∧ *diag-compat B4 xs*))

When this is the case, the decomposition of a matrix leaves it unchanged.

lemma *diag-compat-extract-subdiag*:
assumes *B*∈ *carrier-mat n n*
and *diag-compat B l*
shows *B* = *diag-block-mat* (*extract-subdiags B l*) ⟨*proof*⟩

Predicate `diag_diff` holds when the decomposition of the considered matrix based on the list of integers provided as a parameter, is such that the diagonal elements of separate components are pairwise distinct.

fun *diag-diff* **where**
diag-diff D [] = (*dim-row D* = *0* ∧ *dim-col D* = *0*)
| *diag-diff D* (*x*#*xs*) =
(*x* ≤ *dim-row D* ∧
(*let* (*D1*, *D2*, *D3*, *D4*) = (*split-block D x x*) *in*
(∀ *i j*. *i* < *dim-row D1* ∧ *j* < *dim-row D4* ⟶ *D1*$$(*i*,*i*) ≠ *D4* $$ (*j*,*j*)) ∧

diag-diff D4 xs))

lemma *diag-diff-hd-diff*:
assumes *diag-diff D* (*a*#*xs*)
and *D*∈ *carrier-mat n n*
and $i < a$
and $a \leq j$
and $j < n$
shows *D*$$(*i*,*i*) ≠ *D* $$ (*j*,*j*)
⟨*proof*⟩

lemma *diag-compat-diagonal*:
assumes *B*∈ *carrier-mat* (*dim-row B*) (*dim-row B*)
and *diagonal-mat B*
and *dim-row B* = *sum-list l*
shows *diag-compat B l* ⟨*proof*⟩

The following lemma provides a sufficient condition for the `diag_compat` predicate to hold.

lemma *commute-diag-compat*:
fixes *D*::'*a*::{*field*} *Matrix.mat*
assumes *diagonal-mat D*
and *D*∈ *carrier-mat n n*
and *B*∈ *carrier-mat n n*
and *B*∗ *D* = *D* ∗ *B*
and *diag-diff D l*
shows *diag-compat B l* ⟨*proof*⟩

6.3 Counting similar neighbours in a list

The function `eq_comps` takes a list as an input and counts the number of adjacent elements that are identical.

fun *eq-comps* **where**
eq-comps [] = []
| *eq-comps* [*x*] = [*1*]
| *eq-comps* (*x*#*y*#*l*) = (*let tmp* = (*eq-comps* (*y*#*l*)) *in*
if x = *y then Suc* (*hd tmp*) # (*tl tmp*)
else 1 # *tmp*)

lemma *eq-comps-not-empty*:
assumes *l*≠ []
shows *eq-comps l* ≠ [] ⟨*proof*⟩

lemma *eq-comps-empty-if*:
assumes *eq-comps l* = []
shows *l* = []
⟨*proof*⟩

lemma *eq-comps-hd-eq-tl*:
assumes $x = y$
shows *tl* (*eq-comps* ($x\#y\#l$)) = *tl* (*eq-comps* ($y\#l$)) ⟨*proof*⟩

lemma *eq-comps-hd-neq-tl*:
assumes $x \neq y$
shows *tl* (*eq-comps* ($x\#y\#l$)) = *eq-comps* ($y\#l$) ⟨*proof*⟩

lemma *eq-comps-drop*:
assumes $x\#xs$ = *eq-comps l*
shows *xs* = *eq-comps* (*drop x l*) ⟨*proof*⟩

lemma *eq-comps-neq-0*:
assumes $a\#m$ = *eq-comps l*
shows $a \neq 0$ ⟨*proof*⟩

lemma *eq-comps-gt-0*:
assumes $l \neq []$
shows *list-all* ($\lambda a.\ 0 < a$) (*eq-comps l*)
⟨*proof*⟩

lemma *eq-comps-elem-le-length*:
assumes $a\#m$ = *eq-comps l*
shows $a \leq$ *length l* ⟨*proof*⟩

lemma *eq-comps-length*:
shows *length* (*eq-comps l*) ≤ *length l*
⟨*proof*⟩

lemma *eq-comps-eq*:
assumes $a\#m$ = *eq-comps l*
and $i < a$
shows *nth l i* = *hd l* ⟨*proof*⟩

lemma *eq-comps-singleton*:
assumes $[a]$ = *eq-comps l*
shows a = *length l* ⟨*proof*⟩

lemma *eq-comps-leq*:
assumes $a\#b\#m$ = *eq-comps l*
and *sorted l*
shows *hd l* < *hd* (*drop a l*) ⟨*proof*⟩

lemma *eq-comps-compare*:
assumes *sorted l*
and $a\#m$ = *eq-comps l*
and $i < a$
and $a \leq j$
and $j <$ *length l*

shows *nth l i* < *nth l j* ⟨*proof*⟩

lemma *eq-comps-singleton-elems*:
assumes *eq-comps l* = [*a*]
shows ∀ *i* < *length l*. *l*!*i* = *l*!*0* ⟨*proof*⟩

lemma *eq-comp-Re*:
assumes ∀ *z* ∈ *set l*. *z* ∈ *Reals*
and *m* = *eq-comps l*
shows *m* = *eq-comps* (*map Re l*) ⟨*proof*⟩

lemma *eq-comps-sum-list*:
shows *sum-list* (*eq-comps l*) = *length l*
⟨*proof*⟩

lemma *eq-comps-elem-lt*:
assumes *1* < *length* (*eq-comps l*)
shows *hd* (*eq-comps l*) < *length l*
⟨*proof*⟩

lemma *eq-comp-sum-diag-mat*:
shows *sum-list* (*eq-comps* (*diag-mat A*)) = *dim-row A*
⟨*proof*⟩

lemma *nsum-Suc-elem*:
assumes *1* < *length* (*eq-comps l*)
shows *l*!(*n-sum* (*Suc i*) (*eq-comps l*)) =
(*drop* (*hd* (*eq-comps l*)) *l*)!(*n-sum i* (*tl* (*eq-comps l*))) ⟨*proof*⟩

lemma *eq-comps-elems-eq*:
assumes *l*≠[]
and *i* < *length* (*eq-comps l*)
and *j* < (*eq-comps l*)!*i*
shows *l*!(*n-sum i* (*eq-comps l*)) = *l*!(*n-sum i* (*eq-comps l*) + *j*) ⟨*proof*⟩

When the diagonal block matrices are extracted using `eq_comp`, each extracted matrix is a multiple of the identity.

lemma *extract-subdiags-eq-comp*:
fixes *A*::*complex Matrix.mat*
assumes *diagonal-mat A*
and *A*∈ *carrier-mat n n*
and *0* < *n*
and *i* < *length* (*eq-comps* (*diag-mat A*))
shows ∃ *k*. (*extract-subdiags A* (*eq-comps* (*diag-mat A*)))!*i* =
k ·$_m$ (*1*$_m$ ((*eq-comps* (*diag-mat A*))!*i*))
⟨*proof*⟩

lemma *extract-subdiags-comp-commute*:
fixes *A*::*complex Matrix.mat*

assumes *diagonal-mat A*
and *A*∈ *carrier-mat n n*
and *0* < *n*
and *i* < *length* (*eq-comps* (*diag-mat A*))
and *B* ∈ *carrier-mat* ((*eq-comps* (*diag-mat A*))!*i*) ((*eq-comps* (*diag-mat A*))!*i*)
shows (*extract-subdiags A* (*eq-comps* (*diag-mat A*)))!*i* ∗ *B* =
B ∗ (*extract-subdiags A* (*eq-comps* (*diag-mat A*)))!*i*
⟨*proof*⟩

In particular, extracting the diagonal sub-blocks of a diagonal matrix leaves it unchanged.

lemma *diagonal-extract-eq*:
assumes *B*∈ *carrier-mat n n*
and *diagonal-mat B*
shows *B* = *diag-block-mat* (*extract-subdiags B* (*eq-comps* (*diag-mat B*)))
⟨*proof*⟩

fun *lst-diff* **where**
lst-diff l [] = (*l* = [])
| *lst-diff l* (*x*#*xs*) = (*x* ≤ *length l* ∧
(∀ *i j*. *i* < *x* ∧ *x* ≤ *j* ∧ *j* < *length l* ⟶ *nth l i* < *nth l j*) ∧
lst-diff (*drop x l*) *xs*)

lemma *sorted-lst-diff*:
assumes *sorted l*
and *m* = *eq-comps l*
shows *lst-diff l m* ⟨*proof*⟩

lemma *lst-diff-imp-diag-diff*:
fixes *D*::*'a*::*preorder Matrix.mat*
assumes *D*∈ *carrier-mat n n*
and *lst-diff* (*diag-mat D*) *m*
shows *diag-diff D m* ⟨*proof*⟩

lemma *sorted-diag-diff*:
fixes *D*::*'a*::*linorder Matrix.mat*
assumes *D*∈ *carrier-mat n n*
and *sorted* (*diag-mat D*)
shows *diag-diff D* (*eq-comps* (*diag-mat D*))
⟨*proof*⟩

lemma *Re-sorted-lst-diff*:
fixes *l*::*complex list*
assumes ∀ *z* ∈ *set l*. *z* ∈ *Reals*
and *sorted* (*map Re l*)
and *m* = *eq-comps l*
shows *lst-diff l m* ⟨*proof*⟩

The following lemma states a sufficient condition for the `diag_diff` predi-

cate to hold.

lemma *cpx-sorted-diag-diff*:
fixes *D*::*complex Matrix.mat*
assumes *D*∈ *carrier-mat n n*
and ∀ *i* < *n*. *D*$$(*i*,*i*) ∈ *Reals*
and *sorted* (*map Re* (*diag-mat D*))
shows *diag-diff D* (*eq-comps* (*diag-mat D*))
⟨*proof*⟩

7 Sorted hermitian decomposition

We prove that any Hermitian matrix A can be decomposed into a product $U^\dagger \cdot B \cdot U$, where U is a unitary matrix and B is a diagonal matrix containing only real components which are ordered along the diagonal.

definition *per-col* **where**
per-col A f = *Matrix.mat* (*dim-row A*) (*dim-col A*) (λ (*i*,*j*). *A* $$(*i*, (*f j*)))

lemma *per-col-carrier*:
assumes *A* ∈ *carrier-mat n m*
shows *per-col A f* ∈ *carrier-mat n m* ⟨*proof*⟩

lemma *per-col-col*:
assumes *A* ∈ *carrier-mat n m*
and *j* < *m*
shows *Matrix.col* (*per-col A f*) *j* = *Matrix.col A* (*f j*)
⟨*proof*⟩

lemma *per-col-adjoint-row*:
assumes *A* ∈ *carrier-mat n n*
and *i* < *n*
and *f i* < *n*
shows *Matrix.row* (*Complex-Matrix.adjoint* (*per-col A f*)) *i* =
Matrix.row (*Complex-Matrix.adjoint A*) (*f i*)
⟨*proof*⟩

lemma *per-col-mult-adjoint*:
assumes *A* ∈ *carrier-mat n n*
and *i* < *n*
and *j* < *n*
and *f i* < *n*
and *f j* < *n*
shows ((*Complex-Matrix.adjoint* (*per-col A f*)) ∗ (*per-col A f*))$$(*i*,*j*) =
((*Complex-Matrix.adjoint A*) ∗ *A*)$$(*f i*, *f j*)
⟨*proof*⟩

lemma *idty-index*:
assumes *bij-betw f* {..< *n*} {..< *n*}

and $i < n$
and $j < n$
shows $(1_m\ n)\$\$(i,j) = (1_m\ n)\$\$(f\ i,\ f\ j)$
⟨*proof*⟩

lemma *per-col-unitary*:
assumes $A \in$ *carrier-mat n n*
and *unitary A*
and *bij-betw f* $\{..< n\}$ $\{..< n\}$
shows *unitary* (*per-col A f*) ⟨*proof*⟩

definition *per-diag* **where**
per-diag A f = *Matrix.mat* (*dim-row A*) (*dim-col A*) (λ (*i*,*j*). *A* \$\$(*f i*, (*f j*)))

lemma *per-diag-carrier*:
shows *per-diag A f* $\in$ *carrier-mat* (*dim-row A*) (*dim-col A*)
⟨*proof*⟩

lemma *per-diag-diagonal*:
assumes $D \in$ *carrier-mat n n*
and *diagonal-mat D*
and *bij-betw f* $\{..< n\}$ $\{..< n\}$
shows *diagonal-mat* (*per-diag D f*) ⟨*proof*⟩

lemma *per-diag-diag-mat*:
assumes $A \in$ *carrier-mat n n*
and $i < n$
and $f\ i < n$
shows *diag-mat* (*per-diag A f*)!i = *diag-mat A* ! (*f i*)
⟨*proof*⟩

lemma *per-diag-diag-mat-Re*:
assumes $A \in$ *carrier-mat n n*
and $i < n$
and $f\ i < n$
shows *map Re* (*diag-mat* (*per-diag A f*))!i = *map Re* (*diag-mat A*) ! (*f i*)
⟨*proof*⟩

lemma *per-diag-real*:
fixes *B*::*complex Matrix.mat*
assumes $B \in$ *carrier-mat n n*
and $\forall\, i< n.\ B\$\$(i,i) \in$ *Reals*
and *bij-betw f* $\{..<n\}$ $\{..<n\}$
shows $\forall\, j<n.$ (*per-diag B f*)\$\$(*j*,*j*) $\in$ *Reals*
⟨*proof*⟩

lemma *per-col-mult-unitary*:
fixes *A*::*complex Matrix.mat*
assumes $A \in$ *carrier-mat n n*

and *unitary A*
and *D∈ carrier-mat n n*
and *diagonal-mat D*
and *0 < n*
and *bij-betw f {..< n} {..< n}*
shows *A * D * (Complex-Matrix.adjoint A) =*
*(per-col A f) * (per-diag D f) * (Complex-Matrix.adjoint (per-col A f))*
*(***is** *?L = ?R)*
⟨*proof*⟩

lemma *sort-permutation*:
assumes *m = sort l*
obtains *f* **where** *bij-betw f {..<length l} {..<length l} ∧*
(∀ i<length l. l ! f i = m ! i)
⟨*proof*⟩

lemma *per-diag-sorted-Re*:
fixes *B*::*complex Matrix.mat*
assumes *B∈ carrier-mat n n*
obtains *f* **where** *bij-betw f {..< n} {..< n} ∧*
map Re (diag-mat (per-diag B f)) = sort (map Re (diag-mat B))
⟨*proof*⟩

lemma *bij-unitary-diag*:
fixes *A*::*complex Matrix.mat*
assumes *unitary-diag A B U*
and *A ∈ carrier-mat n n*
and *bij-betw f {..<n} {..<n}*
and *0 < n*
shows *unitary-diag A (per-diag B f) (per-col U f)*
⟨*proof*⟩

lemma *hermitian-real-diag-sorted*:
assumes *A ∈ carrier-mat n n*
and *0 < n*
and *hermitian A*
obtains *Bs Us* **where** *real-diag-decomp A Bs Us ∧ sorted (map Re (diag-mat Bs))*
⟨*proof*⟩

8 Commuting Hermitian families

This part is devoted to the proof that a finite family of commuting Hermitian matrices is simultaneously diagonalizable.

8.1 Intermediate properties

lemma *real-diag-decomp-mult-dbm-unit*:

assumes $A \in$ *carrier-mat n n*
and *real-diag-decomp A B U*
and $B =$ *diag-block-mat Bl*
and *length Ul* = *length Bl*
and $\forall\, i <$ *length Bl. dim-col* (*Bl*!*i*) = *dim-row* (*Bl*!*i*)
and $\forall\, i <$ *length Bl. dim-row* (*Bl*!*i*) = *dim-row* (*Ul*!*i*)
and $\forall\, i <$ *length Bl. dim-col* (*Bl*!*i*) = *dim-col* (*Ul*!*i*)
and *unitary* (*diag-block-mat Ul*)
and $\forall\, i <$*length Ul. Ul* ! *i* ∗ *Bl* ! *i* = *Bl* ! *i* ∗ *Ul* ! *i*
shows *real-diag-decomp A B* (*U* ∗ (*diag-block-mat Ul*))
⟨*proof*⟩

lemma *real-diag-decomp-block-set*:
assumes *Als* ≠ {}
and *0* < *n*
and ∀ *Al* ∈ *Als. length Al* = *n*
and $\forall\, i < n.\ \forall$ *Al* ∈ *Als. dim-row* (*Al*!*i*) = *dim-col* (*Al*!*i*)
and $\forall\, i < n.\ \exists\, U.\ \forall$ *Al* ∈ *Als.* $\exists\, B.$ *real-diag-decomp* (*Al*!*i*) *B U*
shows $\exists\, Ul.$ (*length Ul* = *n* ∧ ($\forall\, i <$*n*. $\forall\, Al \in Als.$
(*dim-row* (*Ul*!*i*) = *dim-row* (*Al*!*i*) ∧ *dim-col* (*Ul*!*i*) = *dim-col* (*Al*!*i*)))∧
(∀ *Al* ∈ *Als.* $\exists\, Bl.$ (*length Bl* = *n* ∧
real-diag-decomp (*diag-block-mat Al*) (*diag-block-mat Bl*) (*diag-block-mat Ul*))))
⟨*proof*⟩

lemma *real-diag-decomp-eq-comps-props*:
assumes *Ap*∈ *carrier-mat n n*
and *0* < *n*
and *real-diag-decomp Ap Bs Us* ∧ *sorted* (*map Re* (*diag-mat Bs*))
shows *Bs* ∈ *carrier-mat n n diagonal-mat Bs unitary Us*
Us ∈ *carrier-mat n n diag-diff Bs* (*eq-comps* (*diag-mat Bs*))
eq-comps (*diag-mat Bs*) ≠ [] *diag-mat Bs* ≠ []
⟨*proof*⟩

lemma *commuting-conj-mat-set-props*:
fixes *As*::′*a*::*conjugatable-field Matrix.mat set*
and *U*::′*a Matrix.mat*
assumes *finite As*
and *card As* ≤ *i*
and $\forall\, A \in As.$ *hermitian A* ∧ *A*∈ *carrier-mat n n*
and ∀ *A*∈ *As.* ∀ *B* ∈ *As. A*∗*B* = *B*∗*A*
and *unitary U*
and *U* ∈ *carrier-mat n n*
and *CjA* = (λ*A2. mat-conj* (*Complex-Matrix.adjoint U*) *A2*)‘*As*
shows *finite CjA card CjA* ≤ *i*
$\forall\, A \in CjA.\ A \in$ *carrier-mat n n* ∧ *hermitian A*
∀ *C1*∈ *CjA.* ∀ *C2*∈ *CjA. C1*∗*C2* = *C2*∗*C1*
⟨*proof*⟩

lemma *commute-extract-diag-block-eq*:

fixes *Ap*::*complex Matrix.mat*
assumes *Ap*∈ *carrier-mat n n*
and *0 < n*
and *real-diag-decomp Ap Bs Us* ∧ *sorted* (*map Re* (*diag-mat Bs*))
and *finite Afp*
and *card Afp* ≤ *i*
and ∀ *A*∈*Afp*. *hermitian A* ∧ *A* ∈ *carrier-mat n n*
and ∀ *A*∈*Afp*. ∀ *B*∈*Afp*. *A* ∗ *B* = *B* ∗ *A*
and ∀ *A*∈ *Afp*. *Ap* ∗ *A* = *A* ∗ *Ap*
and *CjA* = (λ*A2*. *mat-conj* (*Complex-Matrix.adjoint Us*) *A2*)‘*Afp*
and *eqcl* = *eq-comps* (*diag-mat Bs*)
shows ∀ *C* ∈ *CjA*. *C* = *diag-block-mat* (*extract-subdiags C eqcl*)
⟨*proof*⟩

lemma *extract-dbm-eq-component-commute*:
assumes ∀ *C*∈*Cs*. *C* = *diag-block-mat* (*extract-subdiags C l*)
and ∀ *C1*∈*Cs*. ∀ *C2*∈*Cs*. *C1* ∗ *C2* = *C2* ∗ *C1*
and *ExC* = (λ*A*. *extract-subdiags A l*)‘*Cs*
and *j < length l*
and *Exi* = (λ*A*. (*A*!*j*))‘ *ExC*
and *Al* ∈ *Exi*
and *Bl*∈ *Exi*
shows *Al* ∗ *Bl* = *Bl* ∗ *Al*
⟨*proof*⟩

lemma *extract-comm-real-diag-decomp*:
fixes *CjA*::*complex Matrix.mat set*
assumes ⋀(*Af*::*complex Matrix.mat set*) *n* . *finite Af* ⟹
card Af ≤ *i* ⟹
Af ≠ {} ⟹
(⋀*A*. *A* ∈ *Af* ⟹ *A* ∈ *carrier-mat n n*) ⟹
0 < n ⟹ (⋀*A*. *A* ∈ *Af* ⟹ *hermitian A*) ⟹
(⋀*A B*. *A* ∈ *Af* ⟹ *B* ∈ *Af* ⟹ *A* ∗ *B* = *B* ∗ *A*) ⟹
∃ *U*. ∀ *A*∈*Af*. ∃ *B*. *real-diag-decomp A B U*
and *finite CjA*
and *CjA* ≠ {}
and *card CjA* ≤ *i*
and ∀ *C*∈*CjA*. *C* = *diag-block-mat* (*extract-subdiags C eqcl*)
and ∀ *C1*∈*CjA*. ∀ *C2*∈*CjA*. *C1* ∗ *C2* = *C2* ∗ *C1*
and *Exc* = (λ*A*. *extract-subdiags A eqcl*)‘*CjA*
and ∀ *E*∈ *Exc*. *list-all* (λ*B*. *0 < dim-row B* ∧ *hermitian B*) *E*
and ∀ *i < length eqcl*. *0 < eqcl*!*i*
shows ∀ *i*<*length eqcl*. ∃ *U*. ∀ *Al*∈*Exc*. ∃ *B*. *real-diag-decomp* (*Al* ! *i*) *B U*
⟨*proof*⟩

8.2 The main result

theorem *commuting-hermitian-family-diag*:
fixes *Af*::*complex Matrix.mat set*

assumes *finite Af*
and $Af \neq \{\}$
and $\bigwedge A.\ A \in Af \Longrightarrow A \in$ *carrier-mat n n*
and $0 < n$
and $\bigwedge A.\ A \in Af \Longrightarrow$ *hermitian A*
and $\bigwedge A\ B.\ A \in Af \Longrightarrow B \in Af \Longrightarrow A * B = B * A$
shows $\exists\ U.\ \forall\ A \in Af.\ \exists B.$ *real-diag-decomp A B U* ⟨*proof*⟩

end