Two Theorems on Hermitian Matrices

Sage Binder and Zilin Jiang

June 7, 2026

Abstract

We formalize two results on Hermitian matrices. First, Sylvester's criterion: a Hermitian matrix is positive definite if and only if all its leading principal submatrices have positive determinant. Second, Cauchy's eigenvalue interlacing theorem: given a principal submatrix B of a Hermitian matrix A, the eigenvalues of B interlace those of A.

Our approach to Sylvester's criterion is fairly standard, and required us to formalize Schur's block matrix determinant formula, which gives a formula for the determinant of a block matrix (A, B, C, D) when A is invertible.

Our approach to Cauchy's eigenvalue interlacing theorem follows a proof given in a set of lecture notes by Dr. David Bindel [1]. This approach involved formalizing the Courant-Fischer minimax theorem (a theorem about the Rayleigh quotient, which we define in this entry). In our statement of the Courant-Fischer minimax theorem, we refer to the infimum and supremum instead of the minimum and maximum, as this simplifies the proof and is sufficient to prove Cauchy's eigenvalue interlacing theorem.

Contents

1 Polynomial Factorization **4**

2 Vector Normalization **4**

3 Determinant, Invertability, and Eigenvalues **4**

4 Quadratic Form **6**

5 Leading Principal Submatrix **6**

6 Hermitian Matrix **7**
6.1 Locale for Complex Hermitian Matrices 8

7 Matrix Conjugate **9**

8 Block Matrix **11**
8.1 Schur's Formula . . . 11

9 Positive Definite Matrix **12**

10 Matrix-Vector Multiplication **12**

11 Module Span **13**

12 Module Homomorphism, Linear Combination, and Span **13**

13 Vector Summation **14**

14 Vector Space **15**

15 Linear Map **16**

16 Matrix as Linear Map **17**

17 Matrix Isometry **17**

18 Compression **17**

19 Submatrix **18**
19.1 Submatrix of Injective Function on Indices, as Compression . 18
19.2 Submatrix of *pick* Function, as Compression . . . 18

20 Schur Decomposition **20**

21 Rayleigh Quotient **20**

22 Courant-Fischer Minimax Theorem **22**
22.1 Max-Min Statement . . . 25
22.2 Min-Max Statement . . . 25
22.3 Theorem Statement . . . 25

23 Cauchy Eigenvalue Interlacing Theorem **25**
23.1 Theorem Statement and Proof . . . 26
23.2 Principal Submatrix Corollaries (Using *pick* Function) 26
23.3 Principal Submatrix Corollary (as Injective Function on Indices) 27

24 Theorem Statements (Outside Locale) **27**

25 Sylvester's Criterion Setup **29**

26 Sylvester's Criterion **29**
26.1 Forward Implication 29
26.2 Reverse Implication 29
26.3 Theorem Statement 30
26.4 Theorem Statement (Outside Locale) 30

27 Code **30**

theory *Misc-Matrix-Results*
imports *Commuting-Hermitian.Commuting-Hermitian*
BenOr-Kozen-Reif.More-Matrix
Jordan-Normal-Form.Spectral-Radius
Jordan-Normal-Form.DL-Rank-Submatrix
Jordan-Normal-Form.Jordan-Normal-Form-Uniqueness
Jordan-Normal-Form.VS-Connect
QHLProver.Complex-Matrix
Fishers-Inequality.Matrix-Vector-Extras
Complex-Bounded-Operators.Extra-Jordan-Normal-Form
Hermite-Lindemann.Misc-HLW
begin

hide-type (**open**)
Finite-Cartesian-Product.vec
Matrix-Legacy.vec
Matrix-Legacy.mat

hide-fact (**open**)
Matrix-Legacy.mat-def
Finite-Cartesian-Product.mat-def
Finite-Cartesian-Product.row-def
Matrix-Legacy.row-def
Matrix-Legacy.row-def
Matrix-Legacy.col-def
Determinants.det-def
Determinants.det-def
Finite-Cartesian-Product.vec-def
Matrix-Legacy.vec-def
Linear-Algebra.adjoint-def

hide-const (**open**)
Matrix-Legacy.mat
Finite-Cartesian-Product.mat
Finite-Cartesian-Product.row
Matrix-Legacy.row
Matrix-Legacy.row
Matrix-Legacy.col
Determinants.det
Determinants.det
Finite-Cartesian-Product.vec

Matrix-Legacy.vec
Linear-Algebra.adjoint

no-notation *fps-nth* (**infixl** $ *75*)

unbundle *no inner-syntax*
unbundle *no vec-syntax*

hide-const (**open**) *Spectral-Radius.spectrum*
hide-fact (**open**) *Spectral-Radius.spectrum-def*

1 Polynomial Factorization

lemma *idom-poly-factor-unique-aux*:
fixes *rs rs′* :: *′a::idom list*
assumes *eq*: ($\prod$ *a*←*rs*. [:*a*, *1*:]) = ($\prod$ *a*←*rs′*. [:*a*, *1*:])
shows *mset rs* ⊆# *mset rs′*
⟨*proof*⟩

lemma *idom-poly-factor-unique*:
fixes *rs rs′* :: *′a::idom list*
assumes *eq*: ($\prod$ *a*←*rs*. [:*a*, *1*:]) = ($\prod$ *a*←*rs′*. [:*a*, *1*:])
shows *mset rs* = *mset rs′*
⟨*proof*⟩

lemma *idom-poly-factor-unique′*:
fixes *rs rs′* :: *′a::idom list*
assumes *eq*: ($\prod$ *a*←*rs*. [:− *a*, *1*:]) = ($\prod$ *a*←*rs′*. [:− *a*, *1*:])
shows *mset rs* = *mset rs′*
⟨*proof*⟩

2 Vector Normalization

lemma *vec-normalize-norm*: *v* ∈ *carrier-vec n* ⟹ *v* ≠ 0_v *n* ⟹ *vec-norm* (*vec-normalize v*) = *1*
⟨*proof*⟩

3 Determinant, Invertability, and Eigenvalues

definition *eigvals-of* [*simp*]:
eigvals-of M es ⟷ *char-poly M* = ($\prod$ *a*←*es*. [:− *a*, *1*:]) ∧ *length es* = *dim-row M*

lemma *eigvals-of-mset-eq*:
eigvals-of (*A*::*′a::idom mat*) *es* ⟹ *eigvals-of A es′* ⟹ *mset es* = *mset es′*
⟨*proof*⟩

lemma *det-is-prod-of-eigenvalues*:
fixes A :: *complex mat*
assumes *square-mat A*
shows *Determinant.det* $A = (\prod e \leftarrow (eigvals\ A).\ e)$
⟨*proof*⟩

lemma *eigvals-of-spectrum*:
$(A::(complex\ mat)) \in carrier\text{-}mat\ n\ n \Longrightarrow eigvals\text{-}of\ A\ \alpha \Longrightarrow Spectral\text{-}Radius.spectrum$ $A = set\ \alpha$
⟨*proof*⟩

lemma *spectrum-connect*:
$(A::complex\ mat) \in carrier\text{-}mat\ n\ n \Longrightarrow Spectral\text{-}Radius.spectrum\ A = spectrum$ A
⟨*proof*⟩

lemma *spectrum-shift*:
fixes A :: *complex mat*
assumes *dim*: $A \in carrier\text{-}mat\ n\ n$
shows $spectrum\ (A - r \cdot_m 1_m\ n) = \{\mu - r \mid \mu.\ \mu \in spectrum\ A\}$ (**is** *?lhs* = *?rhs*)
⟨*proof*⟩

lemma *trivial-kernel-imp-nonzero-eigenvalues*:
fixes M :: *'a::{idom,ring-1-no-zero-divisors} mat*
assumes *square-mat M*
assumes $mat\text{-}kernel\ M \subseteq \{0_v\ (dim\text{-}row\ M)\}$
assumes *eigenvalue M e*
shows $e \neq 0$
⟨*proof*⟩

lemma *trivial-kernel-imp-invertible*:
fixes M :: *complex mat*
assumes *square-mat M*
assumes $mat\text{-}kernel\ M \subseteq \{0_v\ (dim\text{-}row\ M)\}$
shows *invertible-mat M*
⟨*proof*⟩

lemma *trivial-kernel-imp-det-nz*:
fixes M :: *complex mat*
assumes *square-mat M*
assumes $mat\text{-}kernel\ M \subseteq \{0_v\ (dim\text{-}row\ M)\}$
shows *Determinant.det* $M \neq 0$
⟨*proof*⟩

lemma *similar-mats-eigvals*:
assumes $A \in carrier\text{-}mat\ n\ n$
assumes $B \in carrier\text{-}mat\ n\ n$
assumes *similar-mat A B*

assumes *eigvals-of A es*
shows *eigvals-of B es*
⟨*proof*⟩

lemma *scale-eigvals*:
fixes A :: *'a::conjugatable-ordered-field mat*
assumes *square-mat A*
assumes $B = c \cdot_m A$
assumes *eigvals-of A es*
shows *eigvals-of B* (*map* ($\lambda x.\ c * x$) *es*)
⟨*proof*⟩

lemma *neg-mat-eigvals*:
fixes A :: *complex mat*
assumes *square-mat A*
assumes *eigvals-of A es*
shows *eigvals-of* ($-A$) (*rev* (*map* ($\lambda x.\ -x$) *es*))
⟨*proof*⟩

4 Quadratic Form

definition *quadratic-form* :: *'a mat* ⇒ *'a vec* ⇒ *'a::{conjugatable-ring}* **where**
quadratic-form M x ≡ *inner-prod x* ($M *_v x$)

abbreviation *QF* ≡ *quadratic-form*

declare
quadratic-form-def[*simp*]

5 Leading Principal Submatrix

definition *leading-principal-submatrix* :: *'a mat* ⇒ *nat* ⇒ *'a mat* **where**
[*simp*]: *leading-principal-submatrix A k* = *submatrix A* {..<*k*} {..<*k*}

abbreviation *lps* ≡ *leading-principal-submatrix*

lemma *leading-principal-submatrix-carrier*:
$m \geq n \Longrightarrow A \in$ *carrier-mat m m* $\Longrightarrow$ *lps A n* $\in$ *carrier-mat n n*
⟨*proof*⟩

lemma *pick-n*:
assumes $i \leq n$
shows *pick* {..*n*} *i* = *i*
⟨*proof*⟩

lemma *pick-n-le*:
assumes $i < n$
shows *pick* {..<*n*} *i* = *i*

⟨*proof*⟩

lemma *leading-principal-submatrix-index*:
assumes $A \in$ *carrier-mat n n*
assumes $k \leq n$
assumes $i < k$
assumes $j < k$
shows (*lps A k*)$$(*i*,*j*) = *A*$$(*i*,*j*)
⟨*proof*⟩

lemma *nested-leading-principle-submatrices*:
assumes $A \in$ *carrier-mat n n*
assumes $k_1 \leq k_2$
assumes $k_2 \leq n$
shows *lps A* k_1 = *lps* (*lps A* k_2) k_1 (**is** *?lhs* = *?rhs*)
⟨*proof*⟩

6 Hermitian Matrix

lemma *hermitian-real-diag-decomp-eigvals*:
fixes *A* :: *complex mat*
assumes $A \in$ *carrier-mat n n*
assumes *hermitian A*
assumes *eigvals-of A es*
obtains *B U* **where**
real-diag-decomp A B U
diag-mat B = *es*
set es ⊆ *Reals*
$B \in$ *carrier-mat n n*
$U \in$ *carrier-mat n n*
⟨*proof*⟩

lemma *hermitian-eigvals-of-real*: *hermitian* (*A*::*complex mat*) ⟹ *eigvals-of A es* ⟹ *set es* ⊆ ℝ
⟨*proof*⟩

lemma *hermitian-is-square*: *hermitian A* ⟹ *square-mat A*
⟨*proof*⟩

lemma *hermitian-ij-ji-iff*:
hermitian A
⟷ *square-mat A* ∧ (∀ *i j*. *i* < *dim-row A* ∧ *j* < *dim-row A* ⟶ *A*$$(*i*,*j*) = *conjugate* (*A*$$(*j*,*i*)))
⟨*proof*⟩

lemma *adjoint-is-conjugate-transpose*: A^H = *adjoint A*
⟨*proof*⟩

6.1 Locale for Complex Hermitian Matrices

locale *cmplx-herm-mat* = *complex-vec-space n* **for** *n* +
  fixes *A* :: *complex mat*
  assumes *A-dim*: $A \in$ *carrier-mat n n*
  assumes *A-herm*: *hermitian A*
begin

lemma *hermitian-quadratic-form-real*:
  fixes *v* :: *complex vec*
  assumes *v*: $v \in$ *carrier-vec n*
  shows *QF A v* $\in$ *Reals*
⟨*proof*⟩

definition $eigenvalues_0$ :: *complex list* **where**
  $eigenvalues_0 \equiv$ *eigvals A*

lemma $eigenvalues_0$*-exist*: $\exists$ *es. eigvals-of A es* ⟨*proof*⟩

lemma $eigenvalues_0$: *eigvals-of A* $eigenvalues_0$
  ⟨*proof*⟩

lemma $eigenvalues_0$*-real*: *set* $eigenvalues_0 \subseteq \mathbb{R}$
  ⟨*proof*⟩

definition *eigenvalues* :: *complex list* **where**
  eigenvalues $\equiv$ *rev* (*sort-key Re* $eigenvalues_0$)

lemma *eigenvalues*: *eigvals-of A eigenvalues*
⟨*proof*⟩

lemma *eigenvalues-sorted*: *sorted-wrt* ($\geq$) *eigenvalues*
⟨*proof*⟩

lemma *eigenvalues-unique*:
  assumes *es*: *eigvals-of A es'*
  assumes *es-sorted*: *sorted-wrt* ($\geq$) *es'*
  shows *es'* = *eigenvalues*
⟨*proof*⟩

lemma *eigenvalues-real*: *set eigenvalues* $\subseteq \mathbb{R}$
  ⟨*proof*⟩

lemma *eigenvalues-set-eq*: *set* $eigenvalues_0$ = *set eigenvalues*
  ⟨*proof*⟩

lemma *hermitian-eigenvalues-real*:
  assumes *e*: *eigenvalue A e*
  shows $e \in$ *Reals*
  ⟨*proof*⟩

lemma *hermitian-spectrum-real*: *spectrum* $A \subseteq$ *Reals*
⟨*proof*⟩

lemma *leading-principal-submatrix-hermitian*:
assumes *k*: $k \leq n$
shows *hermitian* (*lps* A k) (**is** *hermitian* ?A')
⟨*proof*⟩

lemma *hermitian-mat-inv*:
assumes A'-*dim*: $A' \in$ *carrier-mat* n n
assumes *inv*: *inverts-mat* A A'
shows *hermitian* A'
⟨*proof*⟩

lemma *negative-hermitian*: *hermitian* $(-A)$
⟨*proof*⟩

lemma *principal-submatrix-hermitian*:
assumes *I*: $I \subseteq \{..<n\}$
shows *hermitian* (*submatrix* A I I) (**is** *hermitian* ?B)
⟨*proof*⟩

lemma *hermitian-row-col*:
assumes $i < n$
shows *row* A i = *conjugate* (*col* A i)
⟨*proof*⟩

lemma *inner-prod-swap*:
assumes *v*: $v \in$ *carrier-vec* n
assumes *w*: $w \in$ *carrier-vec* n
shows $(A *_v v) \cdot c\ w = v \cdot c\ (A *_v w)$
⟨*proof*⟩

end

7 Matrix Conjugate

lemma *conjugate-mat-dist*:
fixes A B :: *'a*::*conjugatable-ring mat*
assumes $A \in$ *carrier-mat* m n
assumes $B \in$ *carrier-mat* n p
shows (*conjugate* A) $*$ (*conjugate* B) = *conjugate* $(A * B)$
⟨*proof*⟩

lemma *conjugate-mat-inv*:
fixes A :: *'a*::{*conjugatable-ring*,*semiring-1*} *mat*
assumes $A \in$ *carrier-mat* n n
assumes $A' \in$ *carrier-mat* n n

assumes *inverts-mat A A'*
shows *inverts-mat* (*conjugate A*) (*conjugate A'*)
⟨*proof*⟩

lemma *conjugate-dist-mult-mat*:
fixes *A* :: *'a::conjugatable-ring mat*
assumes $A \in$ *carrier-mat m n* $B \in$ *carrier-mat n p*
shows *conjugate* $(A * B) =$ *conjugate A* $*$ *conjugate B*
(**is** *?lhs = ?rhs*)
⟨*proof*⟩

lemma *conjugate-dist-add-mat*:
fixes *A* :: *'a::conjugatable-ring mat*
assumes $A \in$ *carrier-mat m n* $B \in$ *carrier-mat m n*
shows *conjugate* $(A + B) =$ *conjugate A* $+$ *conjugate B*
(**is** *?lhs = ?rhs*)
⟨*proof*⟩

lemma *mat-row-conj*:
assumes $A \in$ *carrier-mat m n*
assumes $i < m$
shows *conjugate* (*row A i*) = *row* (*conjugate A*) *i*
⟨*proof*⟩

lemma *conj-mat-vec-mult*:
fixes *A* :: *'a::{conjugate,conjugatable-ring} mat*
fixes *v* :: *'a vec*
assumes $A \in$ *carrier-mat n n*
assumes $v \in$ *carrier-vec n*
shows *conjugate* $(A *_v v) =$ (*conjugate A*) $*_v$ (*conjugate v*)
(**is** *?lhs = ?rhs*)
⟨*proof*⟩

lemma *conjugate-vec-first*:
assumes $v \in$ *carrier-vec n*
assumes $i \leq n$
shows *conjugate* (*vec-first v i*) = *vec-first* (*conjugate v*) *i*
⟨*proof*⟩

lemma *conjugate-vec-last*: $i \leq$ *dim-vec v* $\Longrightarrow$ *conjugate* (*vec-last v i*) = *vec-last* (*conjugate v*) *i*
⟨*proof*⟩

lemma *cscalar-prod-symm-conj*:
dim-vec (*x::('a::{comm-semiring-0,conjugatable-ring} vec)*) = *dim-vec* (*y::'a vec*)
$\Longrightarrow$ *x* $\cdot$*c y* = *conjugate* (*y* $\cdot$*c x*)
⟨*proof*⟩

8 Block Matrix

lemma *block-mat-vec-mult*:
fixes x
assumes $A \in$ *carrier-mat nr1 nc1*
assumes $B \in$ *carrier-mat nr1 nc2*
assumes $C \in$ *carrier-mat nr2 nc1*
assumes $D \in$ *carrier-mat nr2 nc2*
assumes $M =$ *four-block-mat A B C D*
assumes $x \in$ *carrier-vec* (*nc1* + *nc2*)
defines $x_1 \equiv$ *vec-first x nc1*
defines $x_2 \equiv$ *vec-last x nc2*
shows $M *_v x = (A *_v x_1 + B *_v x_2) @_v (C *_v x_1 + D *_v x_2)$
⟨*proof*⟩

lemma *mat-vec-prod-leading-principal-submatrix*:
fixes A :: ('*a* :: *comm-ring*) *mat*
assumes $A \in$ *carrier-mat* (*Suc n*) (*Suc n*)
assumes $x \in$ *carrier-vec* (*Suc n*)
defines $A_n \equiv$ *lps A n*
defines $v_n \equiv$ *vec-first* (*col A n*) *n*
defines $w_n \equiv$ *vec-first* (*row A n*) *n*
defines $a \equiv A$ $$ (n, n)
defines $x_n \equiv$ *vec-first x n*
defines $b \equiv x$\$$n$
shows $A *_v x = (A_n *_v x_n + b \cdot_v v_n) @_v$ (*vec 1* ($\lambda i.$ $(w_n \cdot x_n) + a * b$)) (**is** *?lhs* = *?rhs*)
⟨*proof*⟩

lemma *vec-first-index*: $n \leq$ *dim-vec v* $\Longrightarrow i < n \Longrightarrow v$\$$i$ = (*vec-first v n*)\$$i$
⟨*proof*⟩

lemma *vec-last-index*:
$n \leq$ *dim-vec v* $\Longrightarrow i \in$ {*dim-vec v* − *m*..<*m*} $\Longrightarrow v$\$$i$ = (*vec-last v m*)\$(i − (*dim-vec v* − *m*))
⟨*proof*⟩

lemma *inner-prod-append*:
assumes $x \in$ *carrier-vec* (*dim-vec* (u $@_v$ v))
shows x ·*c* (u $@_v$ v) = (*vec-first x* (*dim-vec u*)) ·*c* u + (*vec-last x* (*dim-vec v*)) ·*c* v
(u $@_v$ v) ·*c* x = u ·*c* (*vec-first x* (*dim-vec u*)) + v ·*c* (*vec-last x* (*dim-vec v*))
⟨*proof*⟩

8.1 Schur's Formula

proposition *schur-formula*:
fixes M :: '*a*::*field mat*
assumes $(A,B,C,D) =$ *split-block M r c*
assumes $r <$ *dim-row M*

assumes $c < \mathit{dim\text{-}col}\ M$
assumes *square-mat M*
assumes *square-mat A*
assumes *inverts-mat A′ A*
assumes *A′-dim*: $A' \in \mathit{carrier\text{-}mat}\ r\ r$
shows $\mathit{Determinant.det}\ M = \mathit{Determinant.det}\ A * \mathit{Determinant.det}\ (D - C * A' * B)$
⟨*proof*⟩

9 Positive Definite Matrix

definition *positive-definite* :: $'a::\{\mathit{ord,conjugatable\text{-}field}\}\ \mathit{mat} \Rightarrow \mathit{bool}$ **where**
$\mathit{positive\text{-}definite}\ M \longleftrightarrow \mathit{hermitian}\ M$
$\wedge\ (\forall x \in \mathit{carrier\text{-}vec}\ (\mathit{dim\text{-}col}\ M).\ x \neq 0_v\ (\mathit{dim\text{-}col}\ M) \longrightarrow \mathit{QF}\ M\ x > 0)$

lemma *leading-principal-submatrix-positive-definite*:
fixes A :: $'a::\{\mathit{conjugatable\text{-}field,ord}\}\ \mathit{mat}$
assumes $A \in \mathit{carrier\text{-}mat}\ n\ n$
assumes *positive-definite A*
assumes $k \leq n$
shows *positive-definite* (*lps A k*)
⟨*proof*⟩

lemma *positive-definite-invertible*:
fixes *M* :: *complex mat*
assumes *positive-definite M*
shows *invertible-mat M*
⟨*proof*⟩

lemma *positive-definite-det-nz*:
fixes *A* :: *complex mat*
assumes *positive-definite A*
shows $\mathit{Determinant.det}\ A \neq 0$
⟨*proof*⟩

10 Matrix-Vector Multiplication

lemma *diagonal-unit-vec′*:
assumes $B \in \mathit{carrier\text{-}mat}\ n\ n$
assumes *diagonal-mat* $(B::'a::\{\mathit{ring,zero\text{-}neq\text{-}one}\}\ \mathit{Matrix.mat})$
shows $B *_v (\mathit{unit\text{-}vec}\ n\ i) = B\ \$\$\ (i,i)\ \cdot_v (\mathit{unit\text{-}vec}\ n\ i)$
⟨*proof*⟩

lemma *mat-vec-idx-disjunct-eq-zero*:
assumes *A*: $A \in \mathit{carrier\text{-}mat}\ m\ n$
assumes *x*: $x \in \mathit{carrier\text{-}vec}\ n$
assumes *some-zero*: $\forall j < n.\ (\mathit{row}\ A\ i)\$j = 0 \vee x\$j = 0$
assumes *i*: $i < m$

shows $(A *_v x)\$i = 0$
⟨*proof*⟩

11 Module Span

context *module*
begin

lemma *mk-coeffs-of-list*:
assumes $\alpha \in (set\ A \to carrier\ R)$
shows $\exists\, c \in \{0..<length\ A\} \to carrier\ R.\ \forall\, v \in set\ A.\ mk\text{-}coeff\ A\ c\ v = \alpha\ v$
⟨*proof*⟩

lemma *span-list-span*:
assumes $set\ A \subseteq carrier\ M$
shows $span\text{-}list\ A = span\ (set\ A)$
⟨*proof*⟩

end

12 Module Homomorphism, Linear Combination, and Span

context *mod-hom*
begin

lemma *lincomb-list-distrib*:
assumes $set\ S \subseteq carrier\ M$
assumes $\alpha \in \{..<length\ S\} \to carrier\ R$
shows $f\ (M.lincomb\text{-}list\ \alpha\ S) = N.lincomb\text{-}list\ \alpha\ (map\ f\ S)$
⟨*proof*⟩

lemma *lincomb-distrib*:
assumes *inj-on f S*
assumes $S \subseteq carrier\ M$
assumes $\alpha \in S \to carrier\ R$
assumes $\forall\, v \in S.\ \alpha\ v = \beta\ (f\ v)$
assumes *finite S*
shows $f\ (M.lincomb\ \alpha\ S) = N.lincomb\ \beta\ (f`S)$
⟨*proof*⟩

lemma *lincomb-distrib-obtain*:
assumes *inj-on f S*
assumes $S \subseteq carrier\ M$
assumes $\alpha \in S \to carrier\ R$
assumes $\forall\, v \in S.\ \alpha\ v = \beta\ (f\ v)$
assumes *finite S*

obtains β **where** $(\forall\, v \in S.\ \alpha\ v = \beta\ (f\ v)) \wedge f\ (M.lincomb\ \alpha\ S) = N.lincomb\ \beta$ $(f`S)$
$\langle proof \rangle$

lemma *image-span-list*:
assumes $set\ vs \subseteq carrier\ M$
shows $f`(M.span\text{-}list\ vs) = N.span\text{-}list\ (map\ f\ vs)$ (**is** $?lhs = ?rhs$)
$\langle proof \rangle$

lemma *image-span*:
assumes *finite vs*
assumes $vs \subseteq carrier\ M$
shows $f`(M.span\ vs) = N.span\ (f`vs)$
$\langle proof \rangle$

lemma *submodule-image*:
assumes $submodule\ R\ M'\ M$
shows $submodule\ R\ (f`M')\ N$
$\langle proof \rangle$

lemma *submodule-restrict*:
assumes $submodule\ R\ M'\ M$
shows $mod\text{-}hom\ R\ (M.md\ M')\ (N.md\ (f`M'))\ f$
$\langle proof \rangle$

end

13 Vector Summation

lemma *complex-vec-norm-sum*:
fixes $x :: complex\ vec$
assumes $x \in carrier\text{-}vec\ n$
shows $vec\text{-}norm\ x = csqrt\ ((\sum i \in \{..<n\}.\ (cmod\ (x\$i))\hat{}2))$
$\langle proof \rangle$

lemma *inner-prod-vec-sum*:
assumes $v \in carrier\text{-}vec\ n$
assumes $w \in carrier\text{-}vec\ n$
assumes $B \subseteq carrier\text{-}vec\ n$
assumes *finite B*
assumes $v = finsum\text{-}vec\ TYPE('a::conjugatable\text{-}ring)\ n\ (\lambda b.\ cs\ b \cdot_v b)\ B$
shows $inner\text{-}prod\ w\ v = (\sum b \in B.\ cs\ b * inner\text{-}prod\ w\ b)$
$\langle proof \rangle$

lemma *sprod-vec-sum*:
assumes $v \in carrier\text{-}vec\ n$
assumes $w \in carrier\text{-}vec\ n$
assumes $B \subseteq carrier\text{-}vec\ n$
assumes *finite B*

assumes $v = finsum\text{-}vec\ TYPE('a::\{comm\text{-}ring\})\ n\ (\lambda b.\ cs\ b \cdot_v b)\ B$
shows $w \cdot v = (\sum b \in B.\ cs\ b * (w \cdot b))$
⟨*proof*⟩

lemma *mat-vec-mult-sum*:
assumes $v \in carrier\text{-}vec\ n$
assumes $A \in carrier\text{-}mat\ n\ n$
assumes $B \subseteq carrier\text{-}vec\ n$
assumes *finite B*
assumes $v = finsum\text{-}vec\ TYPE('a::comm\text{-}ring)\ n\ (\lambda b.\ cs\ b \cdot_v b)\ B$
shows $A *_v v = finsum\text{-}vec\ TYPE('a::comm\text{-}ring)\ n\ (\lambda b.\ cs\ b \cdot_v (A *_v b))\ B$
(**is** *?lhs = ?rhs*)
⟨*proof*⟩

14 Vector Space

context *vectorspace*
begin

lemma *basis-subset-carrier*: $basis\ B \Longrightarrow B \subseteq carrier\ V$ ⟨*proof*⟩

lemma *dim-of-lin-indpt-span*:
assumes *li*: *lin-indpt S*
assumes *carrier*: $S \subseteq carrier\ V$
assumes *fin*: *finite S*
shows $vectorspace.dim\ K\ (vs\ (span\ S)) = card\ S$
⟨*proof*⟩

lemma *subspace-fin-dim*:
assumes *W*: *subspace K W V*
assumes *fin*: *fin-dim*
shows *vectorspace.fin-dim K (vs W)*
⟨*proof*⟩

lemma *nontriv-obtain*:
assumes $dim \neq 0$ *fin-dim*
obtains *v* **where** $v \in carrier\ V$ $v \neq \mathbf{0}_V$
⟨*proof*⟩

lemma *nontriv-subspace-obtain*:
assumes *subspace K W V*
assumes $vectorspace.dim\ K\ (vs\ W) \neq 0$
assumes *fin-dim*
obtains *v* **where** $v \in W$ $v \neq \mathbf{0}_V$
⟨*proof*⟩

lemma *nontriv-exists*:
assumes $dim \neq 0$ *fin-dim*
shows $\exists v \in carrier\ V.\ v \neq \mathbf{0}_V$

⟨*proof*⟩

lemma *nontriv-subspace-exists*:
assumes *subspace K W V*
assumes *vectorspace.dim K (vs W)* $\neq 0$
assumes *fin-dim*
shows $\exists\, v \in W.\ v \neq \mathbf{0}_V$
⟨*proof*⟩

lemma *dim-sum-nontriv-int*:
assumes W_1: *subspace K* W_1 *V*
assumes W_2: *subspace K* W_2 *V*
assumes *dim*: *dim* < *vectorspace.dim K (vs* W_1*)* + *vectorspace.dim K (vs* W_2*)*
assumes *fin*: *fin-dim*
shows $\exists\, v \in W_1 \cap W_2.\ v \neq \mathbf{0}_V$
⟨*proof*⟩

end

15 Linear Map

context *linear-map*
begin

lemma *inj-image-lin-indpt*:
assumes *inj-on T (carrier V)*
assumes *S* ⊆ *carrier V*
assumes *V.module.lin-indpt S*
assumes *finite S*
shows *W.module.lin-indpt (T‘S)*
⟨*proof*⟩

lemma *subspace-restrict*:
assumes *subspace K V′ V*
shows *linear-map K (V.vs V′) (W.vs (T‘V′)) T*
⟨*proof*⟩

lemma *subspace-image*:
assumes *subspace K V′ V*
shows *subspace K (T‘V′) W*
⟨*proof*⟩

lemma *inj-subspace-image*:
assumes *inj*: *inj-on T (carrier V)*
assumes *subspace*: *subspace K V′ V*
assumes *fin*: *V.fin-dim*
shows *vectorspace.dim K (W.vs (T‘V′))* = *vectorspace.dim K (V.vs V′)*
⟨*proof*⟩

end

lemma *linear-map-mat*:
assumes $A \in$ *carrier-mat n m*
shows *linear-map class-ring* (*module-vec TYPE*('*a*::{*field*,*ring-1*}) *m*) (*module-vec TYPE*('*a*) *n*) (($*_v$) *A*)
(**is** *linear-map ?K ?V ?W ?T*)
⟨*proof*⟩

16 Matrix as Linear Map

locale *mat-as-linear-map* =
fixes *A* :: '*a*::*field mat*
fixes *m n*
assumes *carrier*: $A \in$ *carrier-mat m n*
begin

abbreviation $V \equiv$ *module-vec TYPE*('*a*) *n*

abbreviation $W \equiv$ *module-vec TYPE*('*a*) *m*

sublocale *linear-map class-ring V W* ($*_v$) *A*
⟨*proof*⟩

abbreviation $f \equiv$ ($*_v$) *A*

end

17 Matrix Isometry

locale *isometry-mat* = *mat-as-linear-map A m n* **for** *A* :: *complex mat* **and** *m n*
+
assumes *isom*: $A^H * A = 1_m\ n$
begin

lemma *preserves-norm*: $v \in$ *carrier-vec n* $\Longrightarrow$ *vec-norm v* = *vec-norm* (*f v*)
⟨*proof*⟩

lemma *is-inj*: *inj-on f* (*carrier V*)
⟨*proof*⟩

end

18 Compression

lemma *compression-is-hermitian*:
assumes *B*: $B \in$ *carrier-mat n m*
assumes *A*: $A \in$ *carrier-mat n n hermitian A*

shows *hermitian* (B^H ∗ *A* ∗ *B*) B^H ∗ *A* ∗ *B* ∈ *carrier-mat m m*
⟨*proof*⟩

19 Submatrix

19.1 Submatrix of Injective Function on Indices, as Compression

Note, with this definition of submatrix, reordering the indices is possible.

definition *submatrix-of-inj* :: *'a mat* ⇒ (*nat* ⇒ *nat*) ⇒ (*nat* ⇒ *nat*) ⇒ *nat* ⇒ *nat* ⇒ *'a mat* **where**
submatrix-of-inj A f g m n ≡ *mat m n* (λ(*i*,*j*). *A*$$(*f i*, *g j*))

lemma *submatrix-of-inj-as-compression*:
fixes *A* :: *complex mat*
fixes *f* :: *nat* ⇒ *nat*
assumes *A*: *A* ∈ *carrier-mat n n*
assumes *f*: *f* : {..<*m*} → {..<*n*}
assumes *inj*: *inj-on f* {..<*m*}
defines *B* ≡ *submatrix-of-inj A f f m m*
defines (*Q*::*complex mat*) ≡ *mat n m* (λ(*i*,*j*). *if i* = *f j then 1 else 0*)
shows *B* = Q^H ∗ *A* ∗ *Q isometry-mat Q n m Q* ∈ *carrier-mat n m*
⟨*proof*⟩

lemma *submatrix-of-inj-as-compression-obt*:
fixes *A* :: *complex mat*
fixes *f* :: *nat* ⇒ *nat*
assumes *A*: *A* ∈ *carrier-mat n n*
assumes *f*: *f* : {..<*m*} → {..<*n*}
assumes *inj*: *inj-on f* {..<*m*}
defines *B* ≡ *submatrix-of-inj A f f m m*
obtains *Q* **where** *B* = Q^H ∗ *A* ∗ *Q isometry-mat Q n m Q* ∈ *carrier-mat n m*
⟨*proof*⟩

19.2 Submatrix of *pick* Function, as Compression

lemma *submatrix-inj*:
assumes *A*: *A* ∈ *carrier-mat m n*
assumes *I*: *I* ⊆ {..<*m*}
assumes *J*: *J* ⊆ {..<*n*}
defines *m'* ≡ *card I*
defines *n'* ≡ *card J*
defines *B* ≡ *submatrix A I J*
defines *f* ≡ λ*i*. *pick I i*
defines *g* ≡ λ*j*. *pick J j*
shows *f* : {..<*m'*} → {..<*m*} *g* : {..<*n'*} → {..<*n*}
inj-on f {..<*m'*} *inj-on g* {..<*n'*}
B = *submatrix-of-inj A f g m' n'*

⟨*proof*⟩

lemma *submatrix-inj-square*:
assumes A: $A \in$ *carrier-mat n n*
assumes I: $I \subseteq \{..<n\}$
defines $m \equiv$ *card I*
defines $B \equiv$ *submatrix A I I*
defines $f \equiv \lambda i.$ *pick I i*
shows $f : \{..<m\} \rightarrow \{..<n\}$ *inj-on f* $\{..<m\}$ $B =$ *submatrix-of-inj A f f m m*
⟨*proof*⟩

lemma *submatrix-inj-obt*:
assumes A: $A \in$ *carrier-mat m n*
assumes I: $I \subseteq \{..<m\}$
assumes J: $J \subseteq \{..<n\}$
defines $m' \equiv$ *card I*
defines $n' \equiv$ *card J*
defines $B \equiv$ *submatrix A I J*
obtains *f g* **where** $f : \{..<m'\} \rightarrow \{..<m\}$ $g : \{..<n'\} \rightarrow \{..<n\}$
inj-on f $\{..<m'\}$ *inj-on g* $\{..<n'\}$
$B =$ *submatrix-of-inj A f g* m' n'
⟨*proof*⟩

lemma *submatrix-inj-square-obt*:
assumes A: $A \in$ *carrier-mat n n*
assumes I: $I \subseteq \{..<n\}$
defines $m \equiv$ *card I*
defines $B \equiv$ *submatrix A I I*
obtains *f* **where** $f : \{..<m\} \rightarrow \{..<n\}$ *inj-on f* $\{..<m\}$ $B =$ *submatrix-of-inj A*
f f m m
⟨*proof*⟩

lemma *submatrix-as-compression*:
fixes A :: *complex mat*
fixes f :: *nat* $\Rightarrow$ *nat*
assumes A: $A \in$ *carrier-mat n n*
assumes I: $I \subseteq \{..<n\}$
defines $m \equiv$ *card I*
defines $B \equiv$ *submatrix A I I*
defines (Q::*complex mat*) $\equiv$ *mat n m* ($\lambda(i,j)$. *if i = pick I j then 1 else 0*)
shows $B = Q^H * A * Q$ *isometry-mat Q n m* $Q \in$ *carrier-mat n m*
⟨*proof*⟩

lemma *submatrix-as-compression-obt*:
fixes A :: *complex mat*
assumes $A \in$ *carrier-mat n n*
assumes $I \subseteq \{..<n\}$
defines $m \equiv$ *card I*
defines $B \equiv$ *submatrix A I I*

obtains Q **where** $B = Q^H * A * Q$ *isometry-mat* Q n m
⟨*proof*⟩

20 Schur Decomposition

context
fixes A Λ U :: $'a$::{*conjugatable-field*,*real-algebra-1*,*ord*} *mat*
fixes n
assumes $*$: *real-diag-decomp* A Λ U
defines $n \equiv$ *dim-row* A
begin

lemma *diag-decomp-inverse*: *inverts-mat* U (*adjoint* U)
⟨*proof*⟩

lemma *diag-decomp-invertible*: *invertible-mat* U
⟨*proof*⟩

lemma *diag-decomp-cols-distinct*: *distinct* (*cols* U)
⟨*proof*⟩

lemma *diag-decomp-ortho*: *corthogonal-mat* U
⟨*proof*⟩ **lemma** *carriers*: $A \in$ *carrier-mat* n n $U \in$ *carrier-mat* n n $U^H \in$ *carrier-mat* n n $\Lambda \in$ *carrier-mat* n n
⟨*proof*⟩

lemma *diag-decomp-cols-lin-indpt*: *module.lin-indpt class-ring* (*module-vec* *TYPE*($'a$) n) (*set* (*cols* U))
⟨*proof*⟩

lemma *real-diag-decomp-eq*: $A = U * \Lambda * U^H$
⟨*proof*⟩ **lemma** *U-unitary*: $U^H * U = 1_m\ n \wedge\ U * U^H = 1_m\ n$
⟨*proof*⟩ **lemma** Λ*-diag*: *diagonal-mat* Λ
⟨*proof*⟩

lemma *diag-decomp-cols-are-eigenvectors*:
assumes $i < n$
shows *eigenvector* A (*col* U i) ((*diag-mat* Λ)!i) (**is** *eigenvector* A *?v* *?μ*)
⟨*proof*⟩

end

21 Rayleigh Quotient

definition *rayleigh-quotient-complex* :: *complex mat* $\Rightarrow$ *complex vec* $\Rightarrow$ *complex* (ϱ_c) **where**
ϱ_c M x = (*QF* M x) / (x $\cdot c$ x)

definition *rayleigh-quotient* :: *complex mat* ⇒ *complex vec* ⇒ *real* (ϱ) **where**
ϱ *M x* = *Re* (ϱ_c *M x*)

lemma *rayleigh-quotient-negative*: *A* ∈ *carrier-mat n n* ⟹ *x* ∈ *carrier-vec n* ⟹ ϱ *A x* = − ϱ (− *A*) *x*
⟨*proof*⟩

lemma *rayleigh-quotient-complex-scale*:
fixes *k* :: *real*
assumes *A* ∈ *carrier-mat n n*
assumes *v* ∈ *carrier-vec n*
assumes *k* ≠ *0*
shows ϱ_c *A v* = ϱ_c *A* (*k* $\cdot_v$ *v*)
⟨*proof*⟩

lemma *rayleigh-quotient-scale-complex*:
assumes *A* ∈ *carrier-mat n n*
assumes *v* ∈ *carrier-vec n*
assumes *k* ≠ *0*
shows ϱ *A* (*k* $\cdot_v$ *v*) = ϱ *A v*
⟨*proof*⟩

lemma *rayleigh-quotient-scale-real*:
fixes *k* :: *real*
assumes *A* ∈ *carrier-mat n n*
assumes *v* ∈ *carrier-vec n*
assumes *k* ≠ *0*
shows ϱ *A v* = ϱ *A* (*k* $\cdot_v$ *v*)
⟨*proof*⟩

context *cmplx-herm-mat*
begin

lemma *hermitian-rayleigh-quotient-real*:
assumes *v* ∈ *carrier-vec n*
assumes *v* ≠ 0_v *n*
shows ϱ_c *A v* ∈ *Reals*
⟨*proof*⟩

lemma *rayleigh-quotient-QF*:
assumes *x*: *x* ≠ 0_v *n x* ∈ *carrier-vec n*
shows ϱ *A x* = (*QF A x*) / (*vec-norm x*)^2
⟨*proof*⟩

end

lemma *rayleigh-eigenvector*:
assumes *eigenvector A v e*
shows ϱ *A v* = *e*

⟨*proof*⟩

end
theory *Cauchy-Eigenvalue-Interlacing*
imports *Misc-Matrix-Results*

begin

22 Courant-Fischer Minimax Theorem

We took inspiration from the proof given in this set of lecture notes by Dr. David Bindel: https://www.cs.cornell.edu/courses/cs6210/2019fa/lec/2019-11-04.pdf. In particular, the approach presented there explicitly obtains the eigenvector basis via diagonal decomposition, which is conducive to the theorems we have available.

definition *sup-defined* :: $'a$::*preorder set* $\Rightarrow$ *bool* **where**
sup-defined S $\longleftrightarrow$ $S \neq \{\} \wedge$ *bdd-above S*

definition *inf-defined* :: $'a$::*preorder set* $\Rightarrow$ *bool* **where**
inf-defined S $\longleftrightarrow$ $S \neq \{\} \wedge$ *bdd-below S*

locale *cf-setup* = *cmplx-herm-mat n* **for** *n*
begin

Although the *cf-setup* locale adds no more assumptions beyond those from the *cmplx-herm-mat* locale, we separate the Courant Fischer proof into a separate locale so that the terms involved in the proof (such as Λ and U) don't pollute the namespace.

definition *dimensional* :: *complex vec set* $\Rightarrow$ *nat* $\Rightarrow$ *bool* **where**
dimensional $\mathcal{V}$ *k* $\longleftrightarrow$ *subspace class-ring* $\mathcal{V}$ *V* $\wedge$ *vectorspace.dim class-ring* (*vs* $\mathcal{V}$) = *k*

lemma *dimensional-n*: *dimensional* $\mathcal{V}$ *k* $\Longrightarrow$ $\mathcal{V} \subseteq$ *carrier-vec n*
⟨*proof*⟩

lemma *dimensional-n-vec*: $\bigwedge v.\ v \in \mathcal{V} \Longrightarrow$ *dimensional* $\mathcal{V}$ *k* $\Longrightarrow v \in$ *carrier-vec n*
⟨*proof*⟩

Note here that we refer to the Inf and Sup rather than the Min and Max.

definition *rayleigh-min* **where**
rayleigh-min $\mathcal{V}$ = *Inf* $\{\varrho\ A\ v \mid v.\ v \neq 0_v\ n \wedge v \in \mathcal{V}\}$

definition *rayleigh-max* **where**
rayleigh-max $\mathcal{V}$ = *Sup* $\{\varrho\ A\ v \mid v.\ v \neq 0_v\ n \wedge v \in \mathcal{V}\}$

definition *maximin* :: *nat* $\Rightarrow$ *real* **where**
maximin k = *Sup* {*rayleigh-min* $\mathcal{V}$ | $\mathcal{V}$. *dimensional* $\mathcal{V}$ *k*}

definition *minimax* :: *nat* ⇒ *real* **where**
minimax k = *Inf* {*rayleigh-max* $\mathcal{V}$ | $\mathcal{V}$. *dimensional* $\mathcal{V}$ (*n* − *k* + *1*)}

definition *maximin-defined* **where**
maximin-defined k ⟷ *sup-defined* {*rayleigh-min* $\mathcal{V}$ | $\mathcal{V}$. *dimensional* $\mathcal{V}$ *k*}

definition *minimax-defined* **where**
minimax-defined k ⟷ *inf-defined* {*rayleigh-max* $\mathcal{V}$ | $\mathcal{V}$. *dimensional* $\mathcal{V}$ (*n* − *k* + *1*)}

abbreviation *es* :: *complex list* **where** *es* ≡ *eigenvalues*

definition Λ-*U* :: *complex mat* × *complex mat* **where**
Λ-*U* ≡ *SOME* (Λ, *U*). *real-diag-decomp A* Λ *U* ∧ *es* = *diag-mat* Λ

definition Λ :: *complex mat* **where**
Λ ≡ *fst* Λ-*U*

definition *U* :: *complex mat* **where**
U ≡ *snd* Λ-*U*

lemma *A-decomp*: *real-diag-decomp A* Λ *U* **and** *es*: *es* = *diag-mat* Λ
⟨*proof*⟩

lemma *U-carrier*: *U* ∈ *carrier-mat n n* **and** Λ-*carrier*: Λ ∈ *carrier-mat n n*
⟨*proof*⟩

lemma *UH-carrier*: U^H ∈ *carrier-mat n n* ⟨*proof*⟩

lemma *UH-unitary*: *unitary* U^H
⟨*proof*⟩

lemma *len-U-cols*: *length* (*cols U*) = *n* ⟨*proof*⟩

lemma *dim*: *local.dim* = *n*
⟨*proof*⟩

lemma *fin-dim*: *fin-dim* ⟨*proof*⟩

lemma *U-cols-orthogonal-span*:
assumes *I*: *I* ⊆ {*0*..<*n*}
assumes *v*: *v* ∈ *span* {*col U i* | *i*. *i* ∈ *I*}
assumes *j*: *j* ∈ {*0*..<*n*} − *I*
shows (U^H $*_v$ *v*)$*j* = *0*
⟨*proof*⟩

lemma *U-cols-basis*:
assumes *I*: *I* ⊆ {*0*..<*n*}

defines [*simp*]: $\mathcal{B} \equiv \{col\ U\ i \mid i.\ i \in I\}$
shows $\mathcal{B} \subseteq carrier\text{-}vec\ n$
$lin\text{-}indpt\ \mathcal{B}$
$card\ \mathcal{B} = card\ I$
$vectorspace.dim\ class\text{-}ring\ (vs\ (span\ \mathcal{B})) = card\ I$
⟨*proof*⟩

end

context *cmplx-herm-mat*
begin

interpretation *A?*: *cf-setup A n* ⟨*proof*⟩

In the local context, we interpret the *cf-setup* locale, giving us access to the abbreviation *es* as well as other constants like U and Λ which form the decomposition of A. Once the local context is closed, this interpretation is no longer accessible, so these names do not pollute the *cmplx-herm-mat* namespace.

lemma *len-eigenvalues*: $length\ es = n$
⟨*proof*⟩

lemma *unit-vec-rayleigh-formula*:
assumes *unit-v*: $vec\text{-}norm\ v = 1$
assumes *v-dim*: $v \in carrier\text{-}vec\ n$
shows $\varrho\ A\ v = (\sum j \in \{..<n\}.\ es!j * (cmod\ ((U^H *_v\ v)\$j))\hat{\ }2)$
⟨*proof*⟩

lemma *rayleigh-bdd-below'*: $\forall v \in carrier\text{-}vec\ n.\ v \neq 0_v\ n \longrightarrow \varrho\ A\ v \geq Min\ (Re$ ‘ $set\ es)$
⟨*proof*⟩

lemma *rayleigh-bdd-below*:
assumes $dimensional\ \mathcal{V}\ k$
shows $\exists m.\ \forall v \in \mathcal{V}.\ v \neq 0_v\ n \longrightarrow \varrho\ A\ v \geq m$
⟨*proof*⟩

lemma *rayleigh-min-exists*:
assumes $dimensional\ \mathcal{V}\ k$
shows $\forall x \in \{\varrho\ A\ v \mid v.\ v \neq 0_v\ n \wedge v \in \mathcal{V}\}.\ rayleigh\text{-}min\ \mathcal{V} \leq x$
⟨*proof*⟩

lemma *courant-fischer-unit-rayleigh-helper1*:
assumes $dimensional\ \mathcal{V}\ (k + 1)$
shows $\exists v.\ vec\text{-}norm\ v = 1 \wedge v \in \mathcal{V} \wedge v \neq 0_v\ n \wedge \varrho\ A\ v \leq es!k$
⟨*proof*⟩

lemma *courant-fischer-unit-rayleigh-helper2*:
assumes *k*: $k < n$

defines *es-R* ≡ *map Re es*
shows $\exists \mathcal{V}.$ *dimensional* $\mathcal{V}$ $(k + 1) \wedge (\forall v.\ v \neq 0_v\ n \wedge v \in \mathcal{V} \longrightarrow$ *es-R* $!\ k \leq \varrho$
$A\ v)$
⟨*proof*⟩

22.1 Max-Min Statement

proposition *courant-fischer-maximin*:
assumes *k*: $k < n$
shows *es*!k = *maximin* $(k + 1)$
maximin-defined $(k + 1)$
⟨*proof*⟩

22.2 Min-Max Statement

interpretation *neg*: *cf-setup* $-A\ n$
⟨*proof*⟩

lemma *neg-es*: *neg.es* = *rev* (*map* ($\lambda x.\ -x$) *es*)
⟨*proof*⟩

lemma *maximin-minimax*:
assumes *k*: $k < n$
shows *neg.maximin* $(n - k) = -$ *minimax* $(k + 1)$
neg.maximin-defined $(n - k) \Longrightarrow$ *minimax-defined* $(k + 1)$
⟨*proof*⟩

lemma *courant-fischer-minimax*:
assumes *k*: $k < n$
shows *es*!k = *minimax* $(k + 1)$ *minimax-defined* $(k + 1)$
⟨*proof*⟩

22.3 Theorem Statement

theorem *courant-fischer*:
assumes $k < n$
shows *es*!k = *minimax* $(k + 1)$
es!k = *maximin* $(k + 1)$
minimax-defined $(k + 1)$
maximin-defined $(k + 1)$
⟨*proof*⟩

23 Cauchy Eigenvalue Interlacing Theorem

We follow the proof given in this set of lecture notes by Dr. David Bindel: https://www.cs.cornell.edu/courses/cs6210/2019fa/lec/2019-11-04.pdf

23.1 Theorem Statement and Proof

lemma *cauchy-eigval-interlacing-aux*:
fixes $W\ B :: complex\ mat$
fixes $\alpha\ \beta :: complex\ list$
fixes $j\ m :: nat$

defines $B \equiv W^H * A * W$
defines $\alpha \equiv eigenvalues$
defines $\beta \equiv cmplx\text{-}herm\text{-}mat.eigenvalues\ B$

assumes *j*: $j < n\ j < m$
assumes *m*: $0 < m\ m \leq n$

assumes *W-dim*: $W \in carrier\text{-}mat\ n\ m$
assumes *W-isom*: $isometry\text{-}mat\ W\ n\ m$

shows $\beta!j \leq \alpha!j\ set\ \alpha \subseteq \mathbb{R}\ set\ \beta \subseteq \mathbb{R}\ length\ \alpha = n\ length\ \beta = m$
⟨*proof*⟩

theorem *cauchy-eigval-interlacing*:
fixes $W\ B :: complex\ mat$
fixes $\alpha\ \beta :: complex\ list$
fixes $j\ m :: nat$

defines $B \equiv W^H * A * W$
defines $\alpha \equiv eigenvalues$
defines $\beta \equiv cmplx\text{-}herm\text{-}mat.eigenvalues\ B$

assumes *j*: $j < n\ j < m$
assumes *m*: $0 < m\ m \leq n$
assumes *W-dim*: $W \in carrier\text{-}mat\ n\ m$
assumes *W-isom*: $isometry\text{-}mat\ W\ n\ m$

shows $\alpha!(n - m + j) \leq \beta!j\ \beta!j \leq \alpha!j\ set\ \alpha \subseteq \mathbb{R}\ set\ \beta \subseteq \mathbb{R}\ length\ \alpha = n\ length\ \beta = m$
⟨*proof*⟩

23.2 Principal Submatrix Corollaries (Using *pick* Function)

corollary *submatrix-eigval-interlacing*:
fixes $I :: nat\ set$

defines $B \equiv submatrix\ A\ I\ I$
defines $m \equiv card\ I$
defines $\alpha \equiv eigenvalues$
defines $\beta \equiv cmplx\text{-}herm\text{-}mat.eigenvalues\ B$

assumes *j*: $j < m$
assumes *I*: $I \subseteq \{..<n\}\ I \neq \{\}$

shows $\alpha!(n - m + j) \leq \beta!j$ $\beta!j \leq \alpha!j$ *set* $\alpha \subseteq \mathbb{R}$ *set* $\beta \subseteq \mathbb{R}$ *length* $\alpha = n$ *length* $\beta = m$
⟨*proof*⟩

23.3 Principal Submatrix Corollary (as Injective Function on Indices)

corollary *submatrix-eigval-interlacing′*:
fixes B :: *complex mat*
fixes m :: *nat*
fixes f :: *nat* ⇒ *nat*

defines $B \equiv$ *submatrix-of-inj A f f m m*
defines $\alpha \equiv$ *eigenvalues*
defines $\beta \equiv$ *cmplx-herm-mat.eigenvalues B*

assumes *f*: $f : \{..<m\} \to \{..<n\}$
assumes *inj*: *inj-on f* $\{..<m\}$
assumes *j*: $j < m$

shows $\alpha!(n - m + j) \leq \beta!j$ $\beta!j \leq \alpha!j$ *set* $\alpha \subseteq \mathbb{R}$ *set* $\beta \subseteq \mathbb{R}$ *length* $\alpha = n$ *length* $\beta = m$
⟨*proof*⟩

end

24 Theorem Statements (Outside Locale)

theorem *courant-fischer*:
fixes A :: *complex mat*

defines *es* ≡ *cmplx-herm-mat.eigenvalues A*

assumes *carrier*: $A \in$ *carrier-mat n n*
assumes *herm*: *hermitian A*
assumes *k*: $k < n$

shows *es*!k = *cf-setup.minimax A n* $(k + 1)$
es!k = *cf-setup.maximin A n* $(k + 1)$
cf-setup.minimax-defined A n $(k + 1)$
cf-setup.maximin-defined A n $(k + 1)$
⟨*proof*⟩

theorem *cauchy-eigval-interlacing*:
fixes A W B :: *complex mat*
fixes α β :: *complex list*
fixes j m n :: *nat*

defines $B \equiv W^H * A * W$
defines $\alpha \equiv$ *cmplx-herm-mat.eigenvalues A*
defines $\beta \equiv$ *cmplx-herm-mat.eigenvalues B*

assumes *A*: $A \in$ *carrier-mat n n hermitian A*
assumes *j*: $j < n$ $j < m$
assumes *m*: $0 < m$ $m \leq n$
assumes *W-dim*: $W \in$ *carrier-mat n m*
assumes *W-isom*: *isometry-mat W n m*

shows $\alpha!(n - m + j) \leq \beta!j$ $\beta!j \leq \alpha!j$ *set* $\alpha \subseteq \mathbb{R}$ *set* $\beta \subseteq \mathbb{R}$ *length* $\alpha = n$ *length* $\beta = m$
⟨*proof*⟩

corollary *submatrix-eigval-interlacing*:
fixes *A* :: *complex mat*
fixes *I* :: *nat set*

defines $B \equiv$ *submatrix A I I*
defines $m \equiv$ *card I*
defines $\alpha \equiv$ *cmplx-herm-mat.eigenvalues A*
defines $\beta \equiv$ *cmplx-herm-mat.eigenvalues B*

assumes *A*: $A \in$ *carrier-mat n n hermitian A*
assumes *j*: $j < m$
assumes *I*: $I \subseteq \{..<n\}$ $I \neq \{\}$

shows $\alpha!(n - m + j) \leq \beta!j$ $\beta!j \leq \alpha!j$ *set* $\alpha \subseteq \mathbb{R}$ *set* $\beta \subseteq \mathbb{R}$ *length* $\alpha = n$ *length* $\beta = m$
⟨*proof*⟩

corollary *submatrix-eigval-interlacing'*:
fixes *A B* :: *complex mat*
fixes *m* :: *nat*
fixes *f* :: *nat* ⇒ *nat*

defines $B \equiv$ *submatrix-of-inj A f f m m*
defines $\alpha \equiv$ *cmplx-herm-mat.eigenvalues A*
defines $\beta \equiv$ *cmplx-herm-mat.eigenvalues B*

assumes *A*: $A \in$ *carrier-mat n n hermitian A*
assumes *f*: $f : \{..<m\} \to \{..<n\}$
assumes *inj*: *inj-on f* $\{..<m\}$
assumes *j*: $j < m$

shows $\alpha!(n - m + j) \leq \beta!j$ $\beta!j \leq \alpha!j$ *set* $\alpha \subseteq \mathbb{R}$ *set* $\beta \subseteq \mathbb{R}$ *length* $\alpha = n$ *length* $\beta = m$
⟨*proof*⟩

end
theory *Sylvester-Criterion*
imports
Cauchy-Eigenvalue-Interlacing
Jordan-Normal-Form.Determinant-Impl

begin

25 Sylvester's Criterion Setup

definition *sylvester-criterion* :: $('a::\{comm\text{-}ring\text{-}1, ord\})$ *mat* ⇒ *bool* **where**
sylvester-criterion A ⟷ $(\forall\, k \le$ *dim-row A*. *Determinant.det* (*lps A k*) $> 0)$

lemma *leading-principle-submatrix-sylvester*:
assumes $A \in$ *carrier-mat n n*
assumes $m \le n$
assumes *sylvester-criterion A*
shows *sylvester-criterion* (*lps A m*)
⟨*proof*⟩

lemma *sylvester-criterion-positive-det*:
assumes $A \in$ *carrier-mat n n*
assumes *sylvester-criterion A*
shows *Determinant.det A* > 0
⟨*proof*⟩

26 Sylvester's Criterion

26.1 Forward Implication

lemma (**in** *cmplx-herm-mat*) *sylvester-criterion-forward*:
assumes $x \in$ *carrier-vec n*
assumes *sylvester-criterion A*
assumes $x \neq 0_v\ n$
shows *Re* (*QF A x*) > 0
⟨*proof*⟩

26.2 Reverse Implication

lemma *prod-list-gz*:
fixes *l* :: *real list*
assumes $\forall\, x \in$ *set l*. $x > 0$
shows *prod-list l* > 0
⟨*proof*⟩

context *cmplx-herm-mat*
begin

lemma *positive-definite-imp-positive-eigenvalues*:

assumes *pd*: *positive-definite A*
shows $\forall\, \mu \in$ *spectrum A*. $\mu > 0$
⟨*proof*⟩

Technically, the following direction is not needed for the final result, but is useful later.

lemma *positive-eigenvalues-imp-positive-definite*:
assumes *pos-spectrum*: $\forall\, \mu \in$ *spectrum A*. $\mu > 0$
shows *positive-definite A*
⟨*proof*⟩

lemma *positive-definite-imp-positive-eigenvalues*$_0$*-Re*:
assumes *pd*: *positive-definite A*
shows $\forall\, \mu \in$ *set* (*map Re eigenvalues*). $\mu > 0$
⟨*proof*⟩

lemma *sylvester-criterion-reverse*:
assumes *pd*: *positive-definite A*
shows *sylvester-criterion A*
⟨*proof*⟩

26.3 Theorem Statement

theorem *sylvester-criterion*:
shows *sylvester-criterion A* ⟷ *positive-definite A*
⟨*proof*⟩

end

26.4 Theorem Statement (Outside Locale)

theorem *sylvester-criterion*:
fixes *A* :: *complex mat*
assumes *hermitian A*
shows *sylvester-criterion A* ⟷ *positive-definite A*
⟨*proof*⟩

27 Code

definition *rat-of-nat-mat* :: *nat mat* ⇒ *rat mat* **where**
rat-of-nat-mat A = *mat* (*dim-row A*) (*dim-col A*) (λ(*i*,*j*). *rat-of-nat* (*A*$$(*i*,*j*)))

definition *rat-of-int-mat* :: *int mat* ⇒ *rat mat* **where**
rat-of-int-mat A = *mat* (*dim-row A*) (*dim-col A*) (λ(*i*,*j*). *rat-of-int* (*A*$$(*i*,*j*)))

context *comm-ring-hom*
begin

lemma *mat-hom-submatrix*:

assumes *I*: $I \subseteq \{..<\mathit{dim\text{-}row}\ A\}$
assumes *J*: $J \subseteq \{..<\mathit{dim\text{-}col}\ A\}$
shows $\mathit{submatrix}\ (\mathit{mat}_h\ A)\ I\ J = \mathit{mat}_h\ (\mathit{submatrix}\ A\ I\ J)$
⟨*proof*⟩

lemma *mat-hom-carrier*: $A \in \mathit{carrier\text{-}mat}\ m\ n \Longrightarrow \mathit{mat}_h\ A \in \mathit{carrier\text{-}mat}\ m\ n$
⟨*proof*⟩

end

lemma *of-rat-less-complex*: $x < y \longleftrightarrow \mathit{complex\text{-}of\text{-}rat}\ x < \mathit{complex\text{-}of\text{-}rat}\ y$
⟨*proof*⟩

lemma *sylvester-criterion-rat*:
assumes *square*: $A \in \mathit{carrier\text{-}mat}\ n\ n$
defines [*simp*]: $(\mathit{hom}_H :: \mathit{rat\ mat} \Rightarrow \mathit{complex\ mat}) \equiv \mathit{of\text{-}rat\text{-}hom.mat\text{-}hom}$
shows $\mathit{sylvester\text{-}criterion}\ A \longleftrightarrow \mathit{sylvester\text{-}criterion}\ (\mathit{hom}_H\ A)$
⟨*proof*⟩

lemma *rat-subset-real*: $\mathbb{Q} \subseteq (\mathbb{R}{::}\mathit{complex\ set})$
⟨*proof*⟩

lemma *conjugate-of-rat*: $\mathit{conjugate}\ (\mathit{of\text{-}rat}\ x) = ((\mathit{of\text{-}rat}\ x){::}\mathit{complex})$
⟨*proof*⟩

lemma *hermitian-rat*:
fixes $A :: \mathit{rat\ mat}$
shows $\mathit{hermitian}\ A \longleftrightarrow \mathit{hermitian}\ ((\mathit{of\text{-}rat\text{-}hom.mat\text{-}hom}\ A){::}\mathit{complex\ mat})$
⟨*proof*⟩

definition *compute-lps* :: $'a\ \mathit{mat} \Rightarrow \mathit{nat} \Rightarrow 'a\ \mathit{mat}$ **where**
[*code*]: $\mathit{compute\text{-}lps}\ A\ k = \mathit{mat}\ k\ k\ (\lambda(i,j).\ A\$\$(i,j))$

definition *compute-sylvester-criterion* :: $\mathit{rat\ mat} \Rightarrow \mathit{bool}$ **where**
[*code*]: $\mathit{compute\text{-}sylvester\text{-}criterion}\ A \longleftrightarrow (\forall k \in \{0..\mathit{dim\text{-}row}\ A\}.\ \mathit{det\text{-}field}\ (\mathit{compute\text{-}lps}\ A\ k) > 0)$

lemma *compute-lps-correct*:
assumes *A*: $A \in \mathit{carrier\text{-}mat}\ n\ n$
assumes *k*: $k \leq n$
shows $\mathit{compute\text{-}lps}\ A\ k = \mathit{lps}\ A\ k$
⟨*proof*⟩

lemma *compute-sylvester-criterion-correct*:
fixes $A :: \mathit{rat\ mat}$
assumes *A*: *hermitian A*
shows $\mathit{compute\text{-}sylvester\text{-}criterion}\ A \longleftrightarrow \mathit{sylvester\text{-}criterion}\ A$
⟨*proof*⟩

lemma *compute-sylvester-criterion-correct'*:
  fixes A :: *rat mat*
  assumes A: *hermitian* A
  assumes *square*: $A \in$ *carrier-mat* n n
  shows *compute-sylvester-criterion* $A \longleftrightarrow$ *positive-definite* ((*of-rat-hom.mat-hom* A)::*complex mat*)
  ⟨*proof*⟩

definition *smallest-eigval-gr* :: *rat mat* ⇒ *rat* ⇒ *bool* **where**
  [*code*]: *smallest-eigval-gr* A r = (*compute-sylvester-criterion* (A − (r $\cdot_m$ 1_m (*dim-row* A))))

The function *smallest-eigval-gr* checks if the smallest eigenvalue of A is bigger than r.

lemma *hermitian-minus-I*:
  fixes A :: *rat mat*
  assumes A: *hermitian* A
  shows *hermitian* (A − r $\cdot_m$ 1_m (*dim-row* A))
  ⟨*proof*⟩

context *comm-ring-hom*
begin

lemma *mat-hom-add*:
  assumes $A \in$ *carrier-mat* m n
  assumes $B \in$ *carrier-mat* m n
  shows mat_h $(A + B)$ = mat_h A + mat_h B
  ⟨*proof*⟩

lemma *mat-hom-uminus*:
  shows mat_h $(-A)$ = − mat_h A
  ⟨*proof*⟩

lemma *mat-hom-sub*:
  assumes $A \in$ *carrier-mat* m n
  assumes $B \in$ *carrier-mat* m n
  shows mat_h $(A - B)$ = mat_h A − mat_h B
  ⟨*proof*⟩

end

lemma *smallest-eigval-gr-correct-aux*:
  fixes A :: *rat mat*
  assumes A: *hermitian* A
  shows *smallest-eigval-gr* A r $\longleftrightarrow$ *positive-definite* ((*of-rat-hom.mat-hom* (A − r $\cdot_m$ 1_m (*dim-row* A)))::*complex mat*)
  ⟨*proof*⟩

lemma *smallest-eigval-gr-correct*:
fixes *A* :: *rat mat*
assumes *A*: *hermitian A*
shows *smallest-eigval-gr A r* $\longleftrightarrow$ ($\forall$ μ::*complex* $\in$ *spectrum* (*of-rat-hom.mat-hom A*). $r < \mu$)
⟨*proof*⟩

end

References

[1] D. Bindel. Lecture notes. https://www.cs.cornell.edu/courses/cs6210/2019fa/lec/2019-11-04.pdf, 2019. CS6210 at Cornell University.