The Functional Calculus of Hermitian Matrices and Product Formulas

Dustin Bryant

November 18, 2026

Abstract

For a Hermitian matrix A and a function $g : \mathbb{R} \to \mathbb{C}$ we define $g(A)$ as the matrix that multiplies every eigenvector of A with eigenvalue λ by $g(\lambda)$ (it exists and is unique), and derive the algebra of this functional calculus: it is linear in g, $g(A)h(A) = (gh)(A)$, $g(A)^* = \bar{g}(A)$, $g(A)^k = (g^k)(A)$, $g(f(A)) = (g \circ f)(A)$ for real f, $g(A)$ is unitary when $|g| = 1$ on the spectrum; special cases are the time evolution e^{itA} and the inverse. On top of it we verify the Lie–Trotter product formula $\|(e^{itH_1/r} \cdots e^{itH_m/r})^r - e^{it(H_1+\cdots+H_m)}\| \leq 2(mLt)^2/r$ and Suzuki's higher-order formulas: Suzuki's recursion S_{2k+2} agrees with e^{itH} up to degree $2k+2$ of the Taylor expansion, and r steps of it are within $r(2|F|+1)(\Lambda|t|/r)^{2k+3}e^{\Lambda|t|/r}/(2k+3)!$ of e^{itH}. The matrices are those of the AFP entry *Jordan Normal Form* [6]; the spectral theorem comes from the Schur decomposition of the AFP entry *Quantum Hoare Logic* [3], the adjoint, unitarity and the vector norm from *Isabelle Marries Dirac* [2]. The product formulas are due to Trotter [7] and Suzuki [5]; they are the basis of quantum simulation [4, 1].

Contents

1 Overview **3**

2 Using the entry **3**
2.1 Examples 4

3 Functional calculus for Hermitian matrices **4**
3.1 Adjoints 5
3.2 Vector norms 5
3.3 Spectral decompositions 7
3.4 The definition 8
3.5 Algebra 10
3.6 Time evolution 12
3.7 The inverse 13

4 Finite sums of matrices **14**

5 The Lie–Trotter product formula **14**
5.1 Scalar bounds 14
5.2 Bounds from the spectrum 15
5.3 Closeness of operators 15
5.4 First-order bounds 16
5.5 Products of evolutions 17

6 Order conditions for product formulas **19**

7 Higher-order product formulas **26**

1 Overview

The entry has five parts.

- `Hermitian_Functional_Calculus`: $g(A)$ for Hermitian A (*mat-fun-eigen*, *mat-fun-decomp*), its algebra ($g(A)h(A) = (gh)(A)$, $g(A)^* = \bar{g}(A)$, linearity, powers, composition), unitarity, time evolution, and the inverse (*herm-inv*).
- `Matrix_Sums`: finite sums of matrices.
- `Trotter_Product_Formula`: the Lie–Trotter formula with error bound (*trotter*).
- `Suzuki_Order`: Suzuki's recursion and its order conditions (*suzuki-order*).
- `Suzuki_Error`: the error of a product formula (*formula-product-approx*, *suzuki-error*).

2 Using the entry

For a Hermitian $A \in M_n$ ($A \in$ *carrier-mat n n* and *hermitian A*) and a function $g : \mathbb{R} \to \mathbb{C}$, *mat-fun g A* is $g(A)$.

- Computing $g(A)$: on eigenvectors (*mat-fun-eigen*), from a spectral decomposition (*spectral-decomp-exists*, *mat-fun-decomp*), or entrywise for real diagonal matrices (*mat-fun-diag*). It exists and is unique (*mat-fun-unique*); to show that a given matrix is $g(A)$, use *mat-fun-eqI*.
- Algebra: *mat-fun-add*, *mat-fun-smult*, *mat-fun-mult*, *mat-fun-pow*, *mat-fun-comp*, *mat-fun-dagger*, *mat-fun-hermitian*, *mat-fun-unitary*, and *mat-fun-cong* (only the values on the spectrum matter).
- *evolution A t* is e^{itA} and *herm-inv A* is A^{-1}.
- $U \approx_\eta V$ says $\|Uv - Vv\| \leq \eta\|v\|$ for all v; the product formulas (*trotter*, *suzuki-error*) are stated with it.

Matrices are those of Jordan_Normal_Form, with the adjoint $M^\dagger$ and *unitary* of Isabelle_Marries_Dirac and *hermitian* of QHLProver. The theories hide Isabelle_Marries_Dirac's gate constants *X*, *Y*, *Z*, *H*, *S*, *T* (still available as *Quantum.X* etc.), so that these letters can name matrices.

2.1 Examples

The evolution under $Z = \mathrm{diag}(1, -1)$ is $e^{itZ} = \mathrm{diag}(e^{it}, e^{-it})$:

lemma *example-evolution-Z*: *evolution* (*diag-matrix* [*1*, *−1*]) *t* = *diag-matrix* [*cis t*, *cis* (*− t*)]
⟨*proof*⟩

The inverse of $\mathrm{diag}(2, 4)$ is $\mathrm{diag}(\frac{1}{2}, \frac{1}{4})$:

lemma *example-inverse*: *herm-inv* (*diag-matrix* [*2*, *4*]) = *diag-matrix* [*1/2*, *1/4*]
⟨*proof*⟩

3 Functional calculus for Hermitian matrices

For Hermitian A and $g : \mathbb{R} \to \mathbb{C}$, $g(A)$ is the matrix that acts on every eigenvector v of A with eigenvalue λ as multiplication by $g(\lambda)$. The eigenvectors span $\mathbb{C}^n$, so this determines $g(A)$; for a spectral decomposition $A = U\,\mathrm{diag}(D)\,U^*$ it is $U\,\mathrm{diag}(g \circ D)\,U^*$.

$\mathrm{diag}(x_1, \ldots, x_n)$, the diagonal matrix with the entries of a list.

definition *diag-matrix* :: *complex list* ⇒ *complex Matrix.mat* **where**
diag-matrix xs = *mat-diag* (*length xs*) (*λi. xs* ! *i*)

lemma *diag-matrix-carrier* [*simp*]: *diag-matrix xs* ∈ *carrier-mat* (*length xs*) (*length xs*)
⟨*proof*⟩

lemma *dim-diag-matrix* [*simp*]: *dim-row* (*diag-matrix xs*) = *length xs dim-col* (*diag-matrix xs*) = *length xs*
⟨*proof*⟩

$$\mathrm{diag}(x_1, \ldots, x_n)\, w = (x_1 w_1, \ldots, x_n w_n)$$

lemma *diag-matrix-mult-vec*:
assumes *dim-vec w* = *length xs*
shows *diag-matrix xs* ∗$_v$ *w* = *Matrix.vec* (*length xs*) (*λi. xs* ! *i* ∗ *w* $ *i*)
⟨*proof*⟩

$$\left(M\ \mathrm{diag}(x)\right)_{*j} = x_j\, M_{*j}$$

lemma *col-mult-diag*:
assumes *M*: *M* ∈ *carrier-mat n* (*length xs*) **and** *j*: *j* < *length xs*
shows *Matrix.col* (*M* ∗ *diag-matrix xs*) *j* = *xs* ! *j* ·$_v$ *Matrix.col M j*
⟨*proof*⟩

$$\mathrm{diag}(x)^* = \mathrm{diag}(\overline{x})$$

lemma *diag-matrix-dagger*: $(\mathit{diag\text{-}matrix}\ xs)^\dagger = \mathit{diag\text{-}matrix}\ (\mathit{map}\ \mathit{cnj}\ xs)$
⟨*proof*⟩

3.1 Adjoints

QHLProver's *hermitian* and Schur decomposition, with IMD's adjoint $M^\dagger$, *Quantum.unitary* and norm $\|v\|$.

lemma *dagger-adjoint*: $M^\dagger = \mathit{Complex\text{-}Matrix.adjoint}\ M$
⟨*proof*⟩

lemma *hermitian-dagger*: $\mathit{hermitian}\ A \longleftrightarrow A^\dagger = A$
⟨*proof*⟩

lemma *dagger-carrier*: $M \in \mathit{carrier\text{-}mat}\ m\ n \Longrightarrow M^\dagger \in \mathit{carrier\text{-}mat}\ n\ m$
⟨*proof*⟩

lemma *unitary-dagger*: $\mathit{unitary}\ A \Longrightarrow \mathit{unitary}\ (A^\dagger)$
⟨*proof*⟩

lemma *unitary-from-complex-unitary*:
assumes $P \in \mathit{carrier\text{-}mat}\ n\ n\ \mathit{Complex\text{-}Matrix.unitary}\ P$
shows $\mathit{unitary}\ P$
⟨*proof*⟩

lemma *one-smult-mat*: $(1{::}\mathit{complex}) \cdot_m M = M$
⟨*proof*⟩

lemma *diag-conj-carrier*:
assumes *U*: $U \in \mathit{carrier\text{-}mat}\ n\ n$ **and** *xs*: $\mathit{length}\ xs = n$
shows $U * \mathit{diag\text{-}matrix}\ xs * (U^\dagger) \in \mathit{carrier\text{-}mat}\ n\ n$
⟨*proof*⟩

3.2 Vector norms

lemma *cpx-vec-length-smult*: $\|a \cdot_v v\| = \mathit{cmod}\ a * \|v\|$
⟨*proof*⟩

The Euclidean norm $\|v\| = \left(\sum_i |v_i|^2\right)^{1/2}$, with $\|v\|^2 = \mathrm{Re}\langle v|v\rangle$, is positive for $v \neq 0$.

lemma *vnorm-L2*: $\|v\| = \mathit{L2\text{-}set}\ (\lambda i.\ \mathit{cmod}\ (v\ \$\ i))\ \{..{<}\mathit{dim\text{-}vec}\ v\}$
⟨*proof*⟩

lemma *vnorm-ge0*: $0 \leq \|v\|$
⟨*proof*⟩

lemma *vnorm-pos*:
assumes *v*: *dim-vec v* = *n v* ≠ 0_v *n*
shows *0* < ‖*v*‖
⟨*proof*⟩

lemma *vnorm-sq*: ‖*v*‖² = (∑ *i*<*dim-vec v*. (*cmod* (*v* $ *i*))²)
⟨*proof*⟩

lemma *vnorm-Re-inner*: *Re* ⟨*v*|*v*⟩ = ‖*v*‖²
⟨*proof*⟩

The triangle inequality, in the forms used later, and its reverse $|\|a\| - \|b\|| \leq \|a - b\|$:

lemma *vnorm-add-le*:
assumes *dim-vec a* = *dim-vec b*
shows ‖*a* + *b*‖ ≤ ‖*a*‖ + ‖*b*‖
⟨*proof*⟩

lemma *vnorm-minus-commute*:
assumes *dim-vec a* = *dim-vec b*
shows ‖*a* − *b*‖ = ‖*b* − *a*‖
⟨*proof*⟩

lemma *vnorm-diff-le*:
assumes *a*: *dim-vec a* = *n* **and** *b*: *dim-vec b* = *n* **and** *c*: *dim-vec c* = *n*
shows ‖*a* − *c*‖ ≤ ‖*a* − *b*‖ + ‖*b* − *c*‖
⟨*proof*⟩

lemma *vnorm-le-diff-add*:
assumes *dim-vec a* = *dim-vec b*
shows ‖*a*‖ ≤ ‖*a* − *b*‖ + ‖*b*‖
⟨*proof*⟩

lemma *vnorm-reverse*:
assumes *dim-vec a* = *dim-vec b*
shows |‖*a*‖ − ‖*b*‖| ≤ ‖*a* − *b*‖
⟨*proof*⟩

lemma *vnorm-zero-vec*: ‖0_v *m*‖ = *0*
⟨*proof*⟩

A unitary is square: comparing $\mathrm{Tr}(U^*U) = \mathrm{Tr}(I_n) = n$ with $\mathrm{Tr}(UU^*) = \mathrm{Tr}(I_m) = m$, both being $\sum_{i,j} |u_{ij}|^2$.

lemma *unitary-square-dim*:
assumes *unitary A*
shows *dim-row A* = *dim-col A*
⟨*proof*⟩

Unitaries preserve the norm, $\|Wy\| = \|y\|$; only the number of columns of

W has to match y.

lemma *unitary-vnorm*:
assumes W: $W \in$ *carrier-mat m n unitary W* **and** y: *dim-vec y* $= n$
shows $\|W *_v y\| = \|y\|$
⟨*proof*⟩

Products of unitaries are unitary.

lemma *unitary-mult*:
assumes *unitary A unitary B A* $\in$ *carrier-mat n n B* $\in$ *carrier-mat n n*
shows *unitary* $(A * B)$
⟨*proof*⟩

Powers of a unitary are unitary.

lemma *unitary-pow*:
assumes U: $U \in$ *carrier-mat m m* **and** u: *unitary U*
shows *unitary* $(U \mathbin{\hat{}}_m k)$
⟨*proof*⟩

3.3 Spectral decompositions

spectral-decomp A U D: U unitary and $U^*AU = \mathrm{diag}(D)$.

definition *spectral-decomp* :: *complex Matrix.mat* ⇒ *complex Matrix.mat* ⇒ *real list* ⇒ *bool* **where**
spectral-decomp A U D ⟷ $U \in$ *carrier-mat* (*dim-row A*) (*dim-row A*) ∧ *unitary U* ∧
length D = *dim-row A* ∧ $U^\dagger * A * U =$ *diag-matrix* (*map complex-of-real D*)

The spectral theorem, from QHLProver's Schur decomposition of Hermitian matrices.

lemma *spectral-decomp-exists*:
assumes A: $A \in$ *carrier-mat n n hermitian A*
shows ∃ *U D*. *spectral-decomp A U D*
⟨*proof*⟩

Unpacking a spectral decomposition: $UU^* = U^*U = I$ and $A = U\ \mathrm{diag}(D)\, U^*$.

lemma *spectral-decompD*:
assumes A: $A \in$ *carrier-mat n n* **and** *sd*: *spectral-decomp A U D*
shows *U* ∈ *carrier-mat n n unitary U length D* = *n* $U^\dagger * U = 1_m\ n\ U * (U^\dagger) = 1_m\ n$
A = *U* ∗ *diag-matrix* (*map complex-of-real D*) ∗ $(U^\dagger)$ *A* ∗ *U* = *U* ∗ *diag-matrix* (*map complex-of-real D*)
⟨*proof*⟩

The columns of U are eigenvectors: $Au_j = D_j u_j$.

lemma *spectral-decomp-col*:
assumes A: $A \in$ *carrier-mat n n* **and** *sd*: *spectral-decomp A U D* **and** j: $j < n$
shows $A *_v$ *Matrix.col U j* = *complex-of-real* $(D\ !\ j) \cdot_v$ *Matrix.col U j*
⟨*proof*⟩

3.4 The definition

B acts as g on the eigenvectors of A, written B *acts-as* g *on-eigenvectors-of* A, if $Av = \lambda v$ implies $Bv = g(\lambda)\, v$. The matrix $g(A)$ is the $n \times n$ matrix with this property; it exists and is unique (below).

definition *acts-on-eigenvectors* :: *complex Matrix.mat* ⇒ (*real* ⇒ *complex*) ⇒ *complex Matrix.mat* ⇒ *bool* **where**
acts-on-eigenvectors A g B ⟷ (∀ *v l. dim-vec v* = *dim-row A* ⟶
A ∗$_v$ *v* = *complex-of-real l* ·$_v$ *v* ⟶ *B* ∗$_v$ *v* = *g l* ·$_v$ *v*)

definition *mat-fun* :: (*real* ⇒ *complex*) ⇒ *complex Matrix.mat* ⇒ *complex Matrix.mat* **where**
mat-fun g A = (*SOME B. B* ∈ *carrier-mat* (*dim-row A*) (*dim-row A*) ∧ *B acts-as g on-eigenvectors-of A*)

The spectrum: the real eigenvalues (JNF's *eigenvalue*).

definition *herm-spectrum* :: *complex Matrix.mat* ⇒ *real set* **where**
herm-spectrum A = {*l. eigenvalue A* (*complex-of-real l*)}

lemma *herm-spectrum-iff*:
l ∈ *herm-spectrum A* ⟷ (∃ *v. dim-vec v* = *dim-row A* ∧ *v* ≠ 0$_v$ (*dim-row A*) ∧ *A* ∗$_v$ *v* = *complex-of-real l* ·$_v$ *v*)
⟨*proof*⟩

Each D_j of a spectral decomposition is an eigenvalue of A.

lemma *spectral-decomp-eig-in-spectrum*:
assumes *A*: *A* ∈ *carrier-mat n n* **and** *sd*: *spectral-decomp A U D* **and** *j*: *j* < *n*
shows *D* ! *j* ∈ *herm-spectrum A*
⟨*proof*⟩

lemma *acts-on-eigenvectorsD*:
B acts-as g on-eigenvectors-of A ⟹ *dim-vec v* = *dim-row A* ⟹ *A* ∗$_v$ *v* = *complex-of-real l* ·$_v$ *v* ⟹
B ∗$_v$ *v* = *g l* ·$_v$ *v*
⟨*proof*⟩

$U\,\mathrm{diag}(g \circ D)\,U^*$ acts correctly on eigenvectors: if $Av = \lambda v$ then $w = U^*v$ has $D_i w_i = \lambda w_i$, so $g(D_i)w_i = g(\lambda)w_i$.

lemma *spectral-decomp-acts-on-eigenvectors*:
assumes *A*: *A* ∈ *carrier-mat n n* **and** *sd*: *spectral-decomp A U D*
shows *U* ∗ *diag-matrix* (*map g D*) ∗ ($U^\dagger$) *acts-as g on-eigenvectors-of A*
⟨*proof*⟩

Two matrices that agree on every eigenvector of a Hermitian A agree on its eigenbasis U, hence are equal.

lemma *eigvec-mat-eqI*:
assumes *A*: *A* ∈ *carrier-mat n n hermitian A*

and *B1*: *B1* ∈ *carrier-mat K n* **and** *B2*: *B2* ∈ *carrier-mat K n*
and *eq*: ⋀*v l*. *dim-vec v* = *n* ⟹ *A* ∗$_v$ *v* = *complex-of-real l* ·$_v$ *v* ⟹ *B1* ∗$_v$ *v* = *B2* ∗$_v$ *v*
shows *B1* = *B2*
⟨*proof*⟩

Hence $g(A)$ is well defined: an $n \times n$ matrix acting as g on the eigenvectors.

lemma
assumes *A*: *A* ∈ *carrier-mat n n hermitian A*
shows *mat-fun-carrier* [*simp*]: *mat-fun g A* ∈ *carrier-mat n n*
and *mat-fun-acts-on-eigenvectors*: *mat-fun g A acts-as g on-eigenvectors-of A*
⟨*proof*⟩

In particular $g(A)$ has the dimensions of A.

lemma *mat-fun-dims* [*simp*]:
assumes *A* ∈ *carrier-mat n n hermitian A*
shows *dim-row* (*mat-fun g A*) = *n dim-col* (*mat-fun g A*) = *n*
⟨*proof*⟩

$$Av = \lambda v \quad \Longrightarrow \quad g(A)\, v = g(\lambda)\, v.$$

theorem *mat-fun-eigen*:
assumes *A*: *A* ∈ *carrier-mat n n hermitian A*
and *v*: *dim-vec v* = *n* **and** *ev*: *A* ∗$_v$ *v* = *complex-of-real l* ·$_v$ *v*
shows *mat-fun g A* ∗$_v$ *v* = *g l* ·$_v$ *v*
⟨*proof*⟩

Uniqueness: every $n \times n$ matrix acting as g on the eigenvectors is $g(A)$.

lemma *mat-fun-eqI*:
assumes *A*: *A* ∈ *carrier-mat n n hermitian A*
and *B*: *B* ∈ *carrier-mat n n B acts-as g on-eigenvectors-of A*
shows *mat-fun g A* = *B*
⟨*proof*⟩

So $g(A)$ is the unique $n \times n$ matrix that acts as g on the eigenvectors of A.

corollary *mat-fun-unique*:
assumes *A*: *A* ∈ *carrier-mat n n hermitian A*
shows ∃!*B*. *B* ∈ *carrier-mat n n* ∧ *B acts-as g on-eigenvectors-of A*
⟨*proof*⟩

So for every spectral decomposition of A,

$$g(A) = U\ \mathrm{diag}\big(g(D_1), \ldots, g(D_n)\big)\, U^*.$$

theorem *mat-fun-decomp*:
assumes *A*: *A* ∈ *carrier-mat n n hermitian A* **and** *sd*: *spectral-decomp A U D*
shows *mat-fun g A* = *U* ∗ *diag-matrix* (*map g D*) ∗ (*U*†)

⟨*proof*⟩

Only the values of g on the spectrum matter.

lemma *mat-fun-cong*:
  assumes *A*: $A \in$ *carrier-mat n n hermitian A*
    and *gh*: $\bigwedge l.\ l \in$ *herm-spectrum* $A \Longrightarrow g\ l = h\ l$
  shows *mat-fun g A* = *mat-fun h A*
⟨*proof*⟩

3.5 Algebra

The identity function gives A, a constant c gives cI:

lemma *mat-fun-id*: $A \in$ *carrier-mat n n* $\Longrightarrow$ *hermitian A* $\Longrightarrow$ *mat-fun complex-of-real A* = *A*
  ⟨*proof*⟩

lemma *mat-fun-const*:
  assumes *A*: $A \in$ *carrier-mat n n hermitian A*
  shows *mat-fun* (λ-. *c*) *A* = *c* $\cdot_m$ 1_m *n*
⟨*proof*⟩

lemma *mat-fun-one*: $A \in$ *carrier-mat n n* $\Longrightarrow$ *hermitian A* $\Longrightarrow$ *mat-fun* (λ-. *1*) *A* = 1_m *n*
  ⟨*proof*⟩

The calculus is multiplicative and respects adjoints:

$$g(A)\,h(A) = (gh)(A), \qquad g(A)^* = \overline{g}(A).$$

theorem *mat-fun-mult*:
  assumes *A*: $A \in$ *carrier-mat n n hermitian A*
  shows *mat-fun g A* $*$ *mat-fun h A* = *mat-fun* ($\lambda x.$ *g x* $*$ *h x*) *A*
⟨*proof*⟩

theorem *mat-fun-dagger*:
  assumes *A*: $A \in$ *carrier-mat n n hermitian A*
  shows (*mat-fun g A*)† = *mat-fun* ($\lambda x.$ *cnj* (*g x*)) *A*
⟨*proof*⟩

Hence $f(A)$ is Hermitian for real f, and $g(A)$ is unitary when $|g| = 1$ on the spectrum, since $g(A)^* g(A) = |g|^2(A) = I$.

lemma *mat-fun-hermitian*:
  assumes *A*: $A \in$ *carrier-mat n n hermitian A*
  shows *hermitian* (*mat-fun* ($\lambda x.$ *complex-of-real* (*f x*)) *A*)
  ⟨*proof*⟩

theorem *mat-fun-unitary*:
  assumes *A*: $A \in$ *carrier-mat n n hermitian A*

and *g*: $\bigwedge l.\ l \in$ *herm-spectrum A* $\Longrightarrow$ *cmod* (*g l*) = *1*
shows *unitary* (*mat-fun g A*)
⟨*proof*⟩

The calculus is additive in the function: $(g + h)(A) = g(A) + h(A)$.

lemma *mat-fun-add*:
assumes *A*: *A* ∈ *carrier-mat n n hermitian A*
shows *mat-fun* (λ*x*. *g x* + *h x*) *A* = *mat-fun g A* + *mat-fun h A*
⟨*proof*⟩

It is homogeneous: $(cg)(A) = c\,g(A)$.

lemma *mat-fun-smult*:
assumes *A*: *A* ∈ *carrier-mat n n hermitian A*
shows *mat-fun* (λ*x*. *c* ∗ *g x*) *A* = *c* ·$_m$ *mat-fun g A*
⟨*proof*⟩

Any two functions of A commute: $g(A)\,h(A) = h(A)\,g(A)$.

lemma *mat-fun-commute*:
assumes *A*: *A* ∈ *carrier-mat n n hermitian A*
shows *mat-fun g A* ∗ *mat-fun h A* = *mat-fun h A* ∗ *mat-fun g A*
⟨*proof*⟩

Powers: $g(A)^k = (g^k)(A)$.

lemma *mat-fun-pow*:
assumes *A*: *A* ∈ *carrier-mat n n hermitian A*
shows *mat-fun g A* ^$_m$ *k* = *mat-fun* (λ*x*. *g x* ^ *k*) *A*
⟨*proof*⟩

Composition: for real f, $g(f(A)) = (g \circ f)(A)$, since an eigenvector of A with eigenvalue λ is one of $f(A)$ with eigenvalue $f(\lambda)$.

lemma *mat-fun-comp*:
assumes *A*: *A* ∈ *carrier-mat n n hermitian A*
shows *mat-fun g* (*mat-fun* (λ*x*. *complex-of-real* (*f x*)) *A*) = *mat-fun* (λ*x*. *g* (*f x*)) *A*
⟨*proof*⟩

For a real diagonal matrix the calculus acts entrywise:

$$g(\mathrm{diag}(D_1, \ldots, D_n)) = \mathrm{diag}(g(D_1), \ldots, g(D_n)).$$

lemma *mat-fun-diag*:
mat-fun g (*diag-matrix* (*map complex-of-real D*)) = *diag-matrix* (*map g D*)
⟨*proof*⟩

Powers: $G^{i+j} = G^i G^j$, and $Uu = zu$ implies $U^k u = z^k u$.

lemma *pow-mat-add*: *G* ∈ *carrier-mat m m* $\Longrightarrow$ *G* ^$_m$ (*i* + *j*) = *G* ^$_m$ *i* ∗ *G* ^$_m$ *j*
⟨*proof*⟩

lemma *pow-eigen*:
fixes U :: *complex Matrix.mat*
assumes U: $U \in$ *carrier-mat m m* **and** u: *dim-vec u* $= m$ **and** *ev*: $U *_v u = z \cdot_v u$
shows $(U \hat{}_m k) *_v u = z \hat{} k \cdot_v u$
⟨*proof*⟩

3.6 Time evolution

e^{itA}, the evolution under the Hamiltonian A.

definition *evolution* :: *complex Matrix.mat* ⇒ *real* ⇒ *complex Matrix.mat* **where**
evolution A t = *mat-fun* (λ*x*. *cis* (*t* ∗ *x*)) *A*

context
fixes A :: *complex Matrix.mat* **and** n :: *nat*
assumes A: $A \in$ *carrier-mat n n hermitian A*
begin

e^{itA} is unitary, acts on an eigenvector by $e^{itA}v = e^{it\lambda}v$, and is a one-parameter group:

$$e^{i0A} = I, \qquad e^{i(s+t)A} = e^{isA}e^{itA}, \qquad \left(e^{itA}\right)^k = e^{iktA}.$$

lemma *evolution-carrier* [*simp*]: *evolution A t* ∈ *carrier-mat n n*
⟨*proof*⟩

lemma *evolution-dims* [*simp*]: *dim-row* (*evolution A t*) = *n dim-col* (*evolution A t*) = *n*
⟨*proof*⟩

lemma *evolution-eigen*:
dim-vec v = *n* ⟹ *A* ∗$_v$ *v* = *complex-of-real l* ·$_v$ *v* ⟹ *evolution A t* ∗$_v$ *v* = *cis* (*t* ∗ *l*) ·$_v$ *v*
⟨*proof*⟩

lemma *evolution-unitary*: *unitary* (*evolution A t*)
⟨*proof*⟩

lemma *evolution-zero*: *evolution A 0* = *1*$_m$ *n*
⟨*proof*⟩

lemma *evolution-add*: *evolution A* (*s* + *t*) = *evolution A s* ∗ *evolution A t*
⟨*proof*⟩

lemma *evolution-pow*: *evolution A t* $\hat{}_m$ *k* = *evolution A* (*real k* ∗ *t*)
⟨*proof*⟩

end

3.7 The inverse

$A^{-1} = g(A)$ for $g(x) = 1/x$; since $1/0 = 0$ in HOL this is the pseudo-inverse in general.

definition *herm-inv* :: *complex Matrix.mat* ⇒ *complex Matrix.mat* **where**
herm-inv A = *mat-fun* (λ*x*. *complex-of-real* (*inverse x*)) *A*

A^{-1} is an $n \times n$ matrix.

lemma *herm-inv-carrier* [*simp*]: *A* ∈ *carrier-mat n n* ⟹ *hermitian A* ⟹ *herm-inv A* ∈ *carrier-mat n n*
⟨*proof*⟩

lemma *herm-inv-dims* [*simp*]:
assumes *A* ∈ *carrier-mat n n hermitian A*
shows *dim-row* (*herm-inv A*) = *n dim-col* (*herm-inv A*) = *n*
⟨*proof*⟩

If 0 is not an eigenvalue, $g(x) = 1/x$ gives the inverse: $A^{-1}A = AA^{-1} = I$.

theorem *herm-inv*:
assumes *A*: *A* ∈ *carrier-mat n n hermitian A* **and** *z*: *0* ∉ *herm-spectrum A*
shows *herm-inv A* ∗ *A* = *1*$_m$ *n A* ∗ *herm-inv A* = *1*$_m$ *n*
⟨*proof*⟩

So $x = A^{-1}b$ solves $Ax = b$; A^{-1} acts on eigenvectors by λ^{-1} and is Hermitian.

lemma *herm-inv-solves*:
assumes *A*: *A* ∈ *carrier-mat n n hermitian A* **and** *z*: *0* ∉ *herm-spectrum A* **and** *b*: *dim-vec b* = *n*
shows *A* ∗$_v$ (*herm-inv A* ∗$_v$ *b*) = *b*
⟨*proof*⟩

lemma *herm-inv-eigen*:
assumes *A*: *A* ∈ *carrier-mat n n hermitian A*
and *v*: *dim-vec v* = *n* **and** *ev*: *A* ∗$_v$ *v* = *complex-of-real l* ·$_v$ *v*
shows *herm-inv A* ∗$_v$ *v* = *complex-of-real* (*inverse l*) ·$_v$ *v*
⟨*proof*⟩

lemma *herm-inv-hermitian*: *A* ∈ *carrier-mat n n* ⟹ *hermitian A* ⟹ *hermitian* (*herm-inv A*)
⟨*proof*⟩

lemma *mult-zero-vec-right*: *B* ∈ *carrier-mat K n* ⟹ *B* ∗$_v$ *0*$_v$ *n* = *0*$_v$ *K*
⟨*proof*⟩

For a unit vector b and invertible A the solution $A^{-1}b$ is not zero.

lemma *herm-inv-sol-pos*:
assumes *A*: *A* ∈ *carrier-mat n n hermitian A* **and** *z*: *0* ∉ *herm-spectrum A* **and** *b*: *dim-vec b* = *n* ‖*b*‖ = *1*

shows $0 < \|\mathit{herm\text{-}inv}\ A *_v b\|$
⟨*proof*⟩

4 Finite sums of matrices

For QHLProver's sums $\sum_{k<n} F_k$ of $K \times K$ matrices: entries are sums of entries, and a sum of multiples of one matrix is a multiple of it,

$$\Big(\sum_k F_k\Big)_{ij} = \sum_k (F_k)_{ij}, \qquad \sum_k f_k\, B = \Big(\sum_k f_k\Big) B.$$

lemma *matrix-sum-index*:
assumes *f*: $\bigwedge k.\ k < n \Longrightarrow f\ k \in \mathit{carrier\text{-}mat}\ K\ K$ **and** *ij*: $i < K\ j < K$
shows *matrix-sum K f n* $$ (*i*, *j*) = ($\sum k{<}n.$ *f k* $$ (*i*, *j*))
⟨*proof*⟩

lemma *matrix-sum-scalar*:
assumes *B*: $B \in \mathit{carrier\text{-}mat}\ d\ d$
shows *matrix-sum d* $(\lambda k.\ f\ k \cdot_m B)\ m = (\sum k{<}m.\ f\ k) \cdot_m B$
⟨*proof*⟩

5 The Lie–Trotter product formula

For Hermitian $H_1, \ldots, H_m$ with $\|H_j v\| \le L\|v\|$, r rounds of $e^{itH_1/r} \cdots e^{itH_m/r}$ are within $2(mLt)^2/r$ of $e^{it(H_1+\cdots+H_m)}$ (*trotter*). Each factor is $I + i\delta H_j$ up to $(\delta L)^2$, from $|e^{ix} - 1 - ix| \le x^2$ and the functional calculus; the errors add along products and powers of unitaries.

5.1 Scalar bounds

Taylor remainders of first order:

$$|\sin x - x| \le \tfrac{x^2}{2}, \qquad |\cos x - 1| \le \tfrac{x^2}{2}, \qquad |e^{ix} - 1 - ix| \le x^2, \qquad |e^{ix} - 1| \le |x|.$$

lemma *sin-minus-x-bound*: $|\mathit{sin}\ (x{::}\mathit{real}) - x| \le x^2\ /\ 2$
⟨*proof*⟩

lemma *cos-minus-one-bound*: $|\mathit{cos}\ (x{::}\mathit{real}) - 1| \le x^2\ /\ 2$
⟨*proof*⟩

lemma *cis-remainder*: $\mathit{cmod}\ (\mathit{cis}\ (x{::}\mathit{real}) - 1 - \mathrm{i} * \mathit{complex\text{-}of\text{-}real}\ x) \le x^2$
⟨*proof*⟩

lemma *cis-minus-one*: $\mathit{cmod}\ (\mathit{cis}\ (x{::}\mathit{real}) - 1) \le |x|$
⟨*proof*⟩

5.2 Bounds from the spectrum

lemma *vnorm-minus-le*:
assumes *dim-vec a = dim-vec b*
shows $\|a - b\| \leq \|a\| + \|b\|$
⟨*proof*⟩

If $\|Av\| \leq L\|v\|$ for all v, every eigenvalue satisfies $|\lambda| \leq L$; and if $|g| \leq c$ on the spectrum, $\|g(A)v\| \leq c\|v\|$.

lemma *herm-spectrum-bound*:
assumes *A*: $A \in$ *carrier-mat n n* **and** *b*: $\bigwedge v.$ *dim-vec v = n* $\Longrightarrow \|A *_v v\| \leq L * \|v\|$
and *l*: $l \in$ *herm-spectrum A*
shows $|l| \leq L$
⟨*proof*⟩

lemma *mat-fun-bound*:
assumes *A*: $A \in$ *carrier-mat n n hermitian A* **and** *c*: $0 \leq c$
and *g*: $\bigwedge l.$ $l \in$ *herm-spectrum A* $\Longrightarrow$ *cmod* (*g l*) $\leq c$ **and** *v*: *dim-vec v = n*
shows $\|$*mat-fun g A* $*_v$ *v*$\| \leq c * \|v\|$
⟨*proof*⟩

The functions $e^{idx} - 1 - idx$ and $e^{idx} - 1$ applied to A give $e^{idA} - I - idA$ and $e^{idA} - I$.

lemma *mat-fun-evolution-rem*:
assumes *A*: $A \in$ *carrier-mat n n hermitian A*
shows *mat-fun* (λ*x*. *cis* (*d* ∗ *x*) − *1* − i ∗ *complex-of-real* (*d* ∗ *x*)) *A* =
evolution A d − 1_m *n* − (i ∗ *complex-of-real d*) $\cdot_m$ *A* (**is** - = *?B*)
⟨*proof*⟩

lemma *mat-fun-evolution-one*:
assumes *A*: $A \in$ *carrier-mat n n hermitian A*
shows *mat-fun* (λ*x*. *cis* (*d* ∗ *x*) − *1*) *A* = *evolution A d* − 1_m *n* (**is** - = *?B*)
⟨*proof*⟩

5.3 Closeness of operators

U and V are η-close, written $U \approx_\eta V$, if $\|Uv - Vv\| \leq \eta\|v\|$ for every v: the operator norm of $U - V$ is at most η.

definition *op-close* :: *complex Matrix.mat* ⇒ *real* ⇒ *complex Matrix.mat* ⇒ *bool* ((- ≈-/ -) [*51*, *0*, *51*] *50*) **where**
U ≈$_\eta$ *V* ⟷ (∀ *v*. *dim-vec v = dim-col U* ⟶ $\|U *_v v - V *_v v\| \leq \eta * \|v\|$)

For $U \in M_{m,n}$ closeness is a bound on the vectors of length n.

lemma *op-close-iff*:
assumes *U* ∈ *carrier-mat m n*
shows *U* ≈$_\eta$ *V* ⟷ (∀ *v*. *dim-vec v = n* ⟶ $\|U *_v v - V *_v v\| \leq \eta * \|v\|$)
⟨*proof*⟩

lemma *op-closeI*:
assumes $U \in$ *carrier-mat m n* **and** $\bigwedge v$. *dim-vec* $v = n \Longrightarrow \|U *_v v - V *_v v\| \leq \eta * \|v\|$
shows $U \approx_\eta V$
⟨*proof*⟩

lemma *op-closeD*:
$U \approx_\eta V \Longrightarrow U \in$ *carrier-mat m n* $\Longrightarrow$ *dim-vec* $v = n \Longrightarrow \|U *_v v - V *_v v\| \leq \eta * \|v\|$
⟨*proof*⟩

Closeness is reflexive and symmetric, and passes to adjoints of unitaries, since $U^* - V^* = U^*(V - U)V^*$ and unitaries preserve norms.

lemma *op-close-refl*:
assumes *e*: $0 \leq \eta$ **and** *U*: $U \in$ *carrier-mat n n*
shows $U \approx_\eta U$
⟨*proof*⟩

lemma *op-close-sym*:
assumes $U \approx_\eta V$ $U \in$ *carrier-mat n n* $V \in$ *carrier-mat n n*
shows $V \approx_\eta U$
⟨*proof*⟩

lemma *op-close-dagger*:
assumes *U*: $U \in$ *carrier-mat n n unitary U* **and** *V*: $V \in$ *carrier-mat n n unitary V*
and *c*: $U \approx_\eta V$
shows $U^\dagger \approx_\eta V^\dagger$
⟨*proof*⟩

5.4 First-order bounds

context
fixes *A* :: *complex Matrix.mat* **and** *n* :: *nat* **and** *L* :: *real*
assumes *A*: $A \in$ *carrier-mat n n hermitian A*
and *Ab*: $\bigwedge v$. *dim-vec* $v = n \Longrightarrow \|A *_v v\| \leq L * \|v\|$
begin

For $\|Av\| \leq L\|v\|$, the bounds above applied on the spectrum give

$$\|e^{idA}v - v - id\,Av\| \leq (dL)^2\|v\|, \qquad \|e^{idA}v - v\| \leq |d|L\|v\|.$$

lemma *evolution-first-order*:
assumes *v*: *dim-vec* $v = n$
shows $\|$*evolution A d* $*_v$ $v - v -$ (i $*$ *complex-of-real d*) $\cdot_v$ ($A *_v v$)$\| \leq (d * L)^2 * \|v\|$
⟨*proof*⟩

lemma *evolution-step*:
  assumes *v*: *dim-vec v* = *n*
  shows ‖*evolution A d* ∗$_v$ *v* − *v*‖ ≤ |*d*| ∗ *L* ∗ ‖*v*‖
⟨*proof*⟩

end

5.5 Products of evolutions

evolution-product n d Hs is $e^{idH_1} \cdots e^{idH_m}$ and *hamiltonian-sum n Hs* is $H_1 + \cdots + H_m$.

fun *evolution-product* :: *nat* ⇒ *real* ⇒ *complex Matrix.mat list* ⇒ *complex Matrix.mat* **where**
  evolution-product n d [] = *1*$_m$ *n*
| *evolution-product n d* (*H* # *Hs*) = *evolution H d* ∗ *evolution-product n d Hs*

fun *hamiltonian-sum* :: *nat* ⇒ *complex Matrix.mat list* ⇒ *complex Matrix.mat* **where**
  hamiltonian-sum n [] = *0*$_m$ *n n*
| *hamiltonian-sum n* (*H* # *Hs*) = *H* + *hamiltonian-sum n Hs*

herm-bounded n L H: H is a Hermitian $n \times n$ matrix with $\|Hv\| \leq L\|v\|$.

definition *herm-bounded* :: *nat* ⇒ *real* ⇒ *complex Matrix.mat* ⇒ *bool* **where**
  herm-bounded n L H ⟷ *H* ∈ *carrier-mat n n* ∧ *hermitian H* ∧
    (∀ *v*. *dim-vec v* = *n* ⟶ ‖*H* ∗$_v$ *v*‖ ≤ *L* ∗ ‖*v*‖)

lemma *herm-boundedD*:
  assumes *herm-bounded n L H*
  shows *H* ∈ *carrier-mat n n hermitian H* ⋀*v*. *dim-vec v* = *n* ⟹ ‖*H* ∗$_v$ *v*‖ ≤ *L* ∗ ‖*v*‖
  ⟨*proof*⟩

lemma *herm-bounded-herm*: *herm-bounded n L H* ⟹ *H* ∈ *carrier-mat n n* ∧ *hermitian H*
  ⟨*proof*⟩

lemma *evolution-product-carrier*:
  (⋀*H*. *H* ∈ *set Hs* ⟹ *H* ∈ *carrier-mat n n* ∧ *hermitian H*) ⟹ *evolution-product n d Hs* ∈ *carrier-mat n n*
⟨*proof*⟩

lemma *hamiltonian-sum-carrier*: (⋀*H*. *H* ∈ *set Hs* ⟹ *H* ∈ *carrier-mat n n*) ⟹ *hamiltonian-sum n Hs* ∈ *carrier-mat n n*
  ⟨*proof*⟩

context
  fixes *n* :: *nat* **and** *L* :: *real*
begin

For Hamiltonians bounded by L: the product of their evolutions is unitary, their sum is Hermitian with $\|\sum_j H_j v\| \leq mL\|v\|$, and

$$\|e^{idH_1} \cdots e^{idH_m} v - v\| \leq m\,|d|L\,\|v\|.$$

lemma *evolution-product-unitary*:
(⋀*H*. *H* ∈ *set Hs* ⟹ *herm-bounded n L H*) ⟹ *unitary* (*evolution-product n d Hs*)
⟨*proof*⟩

lemma *hamiltonian-sum-hermitian*:
(⋀*H*. *H* ∈ *set Hs* ⟹ *herm-bounded n L H*) ⟹ *hermitian* (*hamiltonian-sum n Hs*)
⟨*proof*⟩

lemma *hamiltonian-sum-bound*:
(⋀*H*. *H* ∈ *set Hs* ⟹ *herm-bounded n L H*) ⟹ *dim-vec v* = *n* ⟹
‖*hamiltonian-sum n Hs* ∗$_v$ *v*‖ ≤ *real* (*length Hs*) ∗ *L* ∗ ‖*v*‖
⟨*proof*⟩

lemma *evolution-product-norm*:
(⋀*H*. *H* ∈ *set Hs* ⟹ *herm-bounded n L H*) ⟹ *dim-vec v* = *n* ⟹ ‖*evolution-product n d Hs* ∗$_v$ *v*‖ = ‖*v*‖
⟨*proof*⟩

lemma *evolution-product-step*:
(⋀*H*. *H* ∈ *set Hs* ⟹ *herm-bounded n L H*) ⟹ *dim-vec v* = *n* ⟹
‖*evolution-product n d Hs* ∗$_v$ *v* − *v*‖ ≤ *real* (*length Hs*) ∗ (|*d*| ∗ *L*) ∗ ‖*v*‖
⟨*proof*⟩

end

To first order the product of the evolutions is the evolution of the sum:

$$\left\| e^{idH_1} \cdots e^{idH_m} v - v - id \sum_j H_j v \right\| \leq m^2 (dL)^2 \|v\|.$$

lemma *evolution-product-first-order*:
fixes *n* :: *nat* **and** *L* :: *real*
shows (⋀*H*. *H* ∈ *set Hs* ⟹ *herm-bounded n L H*) ⟹ *dim-vec v* = *n* ⟹
‖*evolution-product n d Hs* ∗$_v$ *v* − *v* − (i ∗ *complex-of-real d*) ·$_v$ (*hamiltonian-sum n Hs* ∗$_v$ *v*)‖
≤ (*d* ∗ *L*)^2 ∗ (*real* (*length Hs*))^2 ∗ ‖*v*‖
⟨*proof*⟩

One Trotter step: $\|e^{idH_1} \cdots e^{idH_m} - e^{id(H_1+\cdots+H_m)}\| \leq 2m^2(dL)^2$.

lemma *trotter-step*:
assumes *Hs*: ⋀*H*. *H* ∈ *set Hs* ⟹ *herm-bounded n L H*

shows *evolution-product n d Hs* $\approx_{2 * (d * L)^2 * (real\ (length\ Hs))^2}$ *evolution* (*hamiltonian-sum n Hs*) *d*
⟨*proof*⟩

Errors of unitaries add up along powers.

lemma *op-close-pow*:
assumes *U*: *U* ∈ *carrier-mat n n unitary U* **and** *V*: *V* ∈ *carrier-mat n n unitary V*
and *c*: *U* $\approx_\eta$ *V*
shows *U* $\widehat{}_m$ *k* $\approx_{real\ k * \eta}$ *V* $\widehat{}_m$ *k*
⟨*proof*⟩

The Lie–Trotter product formula with explicit error: $\|(e^{itH_1/r} \cdots e^{itH_m/r})^r - e^{it(H_1+\cdots+H_m)}\| \le 2(mLt)^2/r$.

theorem *trotter*:
assumes *Hs*: ⋀*H*. *H* ∈ *set Hs* ⟹ *herm-bounded n L H* **and** *r*: *0* < *r*
shows *evolution-product n* (*t* / *real r*) *Hs* $\widehat{}_m$ *r* $\approx_{2 * (real\ (length\ Hs) * L * t)^2\ /\ real\ r}$ *evolution* (*hamiltonian-sum n Hs*) *t*
⟨*proof*⟩

6 Order conditions for product formulas

Taylor coefficients of products $e^{tY_1} \cdots e^{tY_m}$ as convolutions of matrix sequences. A palindromic product has odd error order (*sym-order*), and Suzuki's five-fold recursion raises the order by two (*scaled-copies*), so S_{2k+2} agrees with e^{itH} up to degree $2k+2$ (*suzuki-order*).

A sequence $f_0, f_1, \ldots$ of $n \times n$ matrices stands for the power series $\sum_p f_p t^p$, and the product of two series has the coefficients of the Cauchy product

$$(f * g)_p = \sum_{q \le p} f_q\, g_{p-q}.$$

definition *cauchy-prod* :: *nat* ⇒ (*nat* ⇒ *complex Matrix.mat*) ⇒ (*nat* ⇒ *complex Matrix.mat*) ⇒ *nat* ⇒ *complex Matrix.mat* **where**
cauchy-prod n f g p = *matrix-sum n* (λ*q*. *f q* ∗ *g* (*p* − *q*)) (*Suc p*)

All terms of the sequence are $n \times n$.

definition *seq-carrier* :: *nat* ⇒ (*nat* ⇒ *complex Matrix.mat*) ⇒ *bool* **where**
seq-carrier n f ⟷ (∀ *p*. *f p* ∈ *carrier-mat n n*)

lemma *seq-carrier-in* [*simp*]: *seq-carrier n f* ⟹ *f q* ∈ *carrier-mat n n*
⟨*proof*⟩

lemma *seq-carrier-row* [*simp*]: *seq-carrier n f* ⟹ *dim-row* (*f q*) = *n*

⟨*proof*⟩

lemma *seq-carrier-col* [*simp*]: *seq-carrier n f* ⟹ *dim-col* (*f q*) = *n*
⟨*proof*⟩

lemma *seq-carrier-mult* [*simp*]: *seq-carrier n f* ⟹ *seq-carrier n g* ⟹ *f q* ∗ *g r* ∈ *carrier-mat n n*
⟨*proof*⟩

lemma *cauchy-prod-carrier*: *seq-carrier n f* ⟹ *seq-carrier n g* ⟹ *seq-carrier n* (*cauchy-prod n f g*)
⟨*proof*⟩

lemma *cauchy-prod-index*:
assumes *seq-carrier n f seq-carrier n g i* < *n j* < *n*
shows *cauchy-prod n f g p* $$ (*i*, *j*) = (∑ *q*≤*p*. ∑ *c*<*n*. *f q* $$ (*i*, *c*) ∗ *g* (*p* − *q*) $$ (*c*, *j*))
⟨*proof*⟩

lemma *triple-reindex*:
fixes *F* :: *nat* ⇒ *nat* ⇒ *nat* ⇒ *'a*::*comm-monoid-add*
shows (∑ *q*≤*p*. ∑ *r*≤*q*. *F r* (*q* − *r*) (*p* − *q*)) = (∑ *r*≤*p*. ∑ *s*≤*p* − *r*. *F r s* (*p* − *r* − *s*))
⟨*proof*⟩

The Cauchy product is associative:

lemma *cauchy-prod-assoc*:
assumes *f*: *seq-carrier n f* **and** *g*: *seq-carrier n g* **and** *h*: *seq-carrier n h*
shows *cauchy-prod n* (*cauchy-prod n f g*) *h p* = *cauchy-prod n f* (*cauchy-prod n g h*) *p*
⟨*proof*⟩

The unit series $1, 0, 0, \ldots$; scaling $f_p \mapsto u^p f_p$, which substitutes ut for t; and the coefficients $X^p/p!$ of e^{tX}.

definition *seq-one* :: *nat* ⇒ *nat* ⇒ *complex Matrix.mat* **where**
seq-one n p = (*if p* = *0 then 1*$_m$ *n else 0*$_m$ *n n*)

definition *seq-scale* :: *complex* ⇒ (*nat* ⇒ *complex Matrix.mat*) ⇒ *nat* ⇒ *complex Matrix.mat* **where**
seq-scale u f p = (*u* ^ *p*) ·$_m$ *f p*

definition *exp-coeff* :: *complex Matrix.mat* ⇒ *nat* ⇒ *complex Matrix.mat* **where**
exp-coeff X p = *complex-of-real* (*1* / *fact p*) ·$_m$ *X* ^$_m$ *p*

lemma *seq-one-carrier* [*simp*]: *seq-carrier n* (*seq-one n*)
⟨*proof*⟩

lemma *seq-scale-carrier* [*simp*]: *seq-carrier n f* ⟹ *seq-carrier n* (*seq-scale u f*)
⟨*proof*⟩

lemma *exp-coeff-carrier* [*simp*]: $X \in$ *carrier-mat n n* $\Longrightarrow$ *seq-carrier n* (*exp-coeff X*)
⟨*proof*⟩

The unit is neutral and scaling commutes with products:

lemma *cauchy-prod-unit-left*: *seq-carrier n f* $\Longrightarrow$ *cauchy-prod n* (*seq-one n*) *f p* = *f p*
⟨*proof*⟩

lemma *cauchy-prod-unit-right*: *seq-carrier n f* $\Longrightarrow$ *cauchy-prod n f* (*seq-one n*) *p* = *f p*
⟨*proof*⟩

lemma *seq-scale-cauchy-prod*: *seq-carrier n f* $\Longrightarrow$ *seq-carrier n g* $\Longrightarrow$ *cauchy-prod n* (*seq-scale u f*) (*seq-scale u g*) *p* = *seq-scale u* (*cauchy-prod n f g*) *p*
⟨*proof*⟩

lemma *smult-pow-mat*: $X \in$ *carrier-mat n n* $\Longrightarrow$ $(u \cdot_m X) \mathbin{\hat{}}_m p = (u \mathbin{\hat{}} p) \cdot_m (X \mathbin{\hat{}}_m p)$
for *X* :: *complex Matrix.mat*
⟨*proof*⟩

lemma *smult-mult-smult*:
fixes *A B* :: *complex Matrix.mat*
assumes $A \in$ *carrier-mat n n* $B \in$ *carrier-mat n n*
shows $(c \cdot_m A) * (d \cdot_m B) = (c * d) \cdot_m (A * B)$
⟨*proof*⟩

lemma *exp-coeff-smult*: $X \in$ *carrier-mat n n* $\Longrightarrow$ *exp-coeff* $(u \cdot_m X)$ *p* = *seq-scale u* (*exp-coeff X*) *p*
⟨*proof*⟩

lemma *exp-coeff-zero*: *exp-coeff* $(0_m\ n\ n)$ *p* = *seq-one n p*
⟨*proof*⟩

$e^{taX}e^{tbX} = e^{t(a+b)X}$, coefficient by coefficient:

$$\sum_{q \le p} \frac{(aX)^q}{q!} \frac{(bX)^{p-q}}{(p-q)!} = \frac{((a+b)X)^p}{p!}.$$

lemma *exp-coeff-add*:
assumes *X*: $X \in$ *carrier-mat n n*
shows *cauchy-prod n* (*exp-coeff* $(a \cdot_m X)$) (*exp-coeff* $(b \cdot_m X)$) *p* = *exp-coeff* $((a + b) \cdot_m X)$ *p*
⟨*proof*⟩

The coefficients of $e^{tY_1} \cdots e^{tY_m}$.

fun *prod-coeff* :: *nat* ⇒ *complex Matrix.mat list* ⇒ *nat* ⇒ *complex Matrix.mat* **where**
  prod-coeff n [] = *seq-one n*
| *prod-coeff n* (*Y* # *Ys*) = *cauchy-prod n* (*exp-coeff Y*) (*prod-coeff n Ys*)

lemma *prod-coeff-carrier*: ∀ *Y* ∈ *set Ys*. *Y* ∈ *carrier-mat n n* ⟹ *seq-carrier n* (*prod-coeff n Ys*)
  ⟨*proof*⟩

Concatenating factor lists multiplies the series, scaling all factors scales the series, and $e^{-tY_m} \cdots e^{-tY_1}$ inverts $e^{tY_1} \cdots e^{tY_m}$:

lemma *prod-coeff-append*:
  assumes ∀ *Y* ∈ *set Ys*. *Y* ∈ *carrier-mat n n* ∀ *Y* ∈ *set Zs*. *Y* ∈ *carrier-mat n n*
  shows *prod-coeff n* (*Ys* @ *Zs*) = *cauchy-prod n* (*prod-coeff n Ys*) (*prod-coeff n Zs*)
  ⟨*proof*⟩

lemma *seq-scale-seq-one*: *seq-scale u* (*seq-one n*) = *seq-one n*
  ⟨*proof*⟩

lemma *prod-coeff-scale*: ∀ *Y* ∈ *set Ys*. *Y* ∈ *carrier-mat n n* ⟹ *prod-coeff n* (*map* (($\cdot_m$) *u*) *Ys*) = *seq-scale u* (*prod-coeff n Ys*)
⟨*proof*⟩

lemma *zero-smult-mat*: *A* ∈ *carrier-mat n n* ⟹ (*0*::*complex*) $\cdot_m$ *A* = 0_m *n n*
  ⟨*proof*⟩

lemma *exp-coeff-inverse*:
  assumes *Y*: *Y* ∈ *carrier-mat n n*
  shows *cauchy-prod n* (*exp-coeff* ((−*1*) $\cdot_m$ *Y*)) (*exp-coeff Y*) = *seq-one n*
⟨*proof*⟩

lemma *cauchy-prod-assoc-fun*: *seq-carrier n f* ⟹ *seq-carrier n g* ⟹ *seq-carrier n h* ⟹ *cauchy-prod n* (*cauchy-prod n f g*) *h* = *cauchy-prod n f* (*cauchy-prod n g h*)
  ⟨*proof*⟩

lemma *cauchy-prod-unit-right-fun*: *seq-carrier n f* ⟹ *cauchy-prod n f* (*seq-one n*) = *f*
  ⟨*proof*⟩

lemma *cauchy-prod-unit-left-fun*: *seq-carrier n f* ⟹ *cauchy-prod n* (*seq-one n*) *f* = *f*
  ⟨*proof*⟩

lemma *prod-coeff-inverse*:
  ∀ *Y* ∈ *set Ys*. *Y* ∈ *carrier-mat n n* ⟹ *cauchy-prod n* (*prod-coeff n* (*rev* (*map* (($\cdot_m$) (−*1*)) *Ys*))) (*prod-coeff n Ys*) = *seq-one n*
⟨*proof*⟩

f vanishes below degree q: $f_p = 0$ for $p < q$.

definition *vanishes-below* :: *nat* ⇒ *nat* ⇒ (*nat* ⇒ *complex Matrix.mat*) ⇒ *bool* **where**
vanishes-below n q f ⟷ (∀ *p*<*q*. *f p* = 0_m *n n*)

lemma *cauchy-prod-add-left*:
seq-carrier n f ⟹ *seq-carrier n g* ⟹ *seq-carrier n h* ⟹ *cauchy-prod n* (λ*p*. *f p* + *g p*) *h p* = *cauchy-prod n f h p* + *cauchy-prod n g h p*
⟨*proof*⟩

lemma *cauchy-prod-add-right*:
seq-carrier n f ⟹ *seq-carrier n g* ⟹ *seq-carrier n h* ⟹ *cauchy-prod n h* (λ*p*. *f p* + *g p*) *p* = *cauchy-prod n h f p* + *cauchy-prod n h g p*
⟨*proof*⟩

lemma *matrix-sum-zero*: *matrix-sum n* (λ*r*. 0_m *n n*) *m* = 0_m *n n*
⟨*proof*⟩

lemma *matrix-sum-single*:
assumes ⋀*q*. *q* < *m* ⟹ *F q* ∈ *carrier-mat n n* **and** ⋀*q*. *q* < *m* ⟹ *q* ≠ *k* ⟹ *F q* = 0_m *n n* **and** *k* < *m*
shows *matrix-sum n F m* = *F k*
⟨*proof*⟩

If d vanishes below q, then $(d * g)_q = d_q g_0$, $(g * d)_q = g_0 d_q$, both products vanish below q, and the product of two such sequences vanishes at $q > 0$ too:

lemma *cauchy-prod-low-left*:
assumes *d*: *seq-carrier n d vanishes-below n q d* **and** *g*: *seq-carrier n g*
shows *cauchy-prod n d g q* = *d q* ∗ *g 0*
⟨*proof*⟩

lemma *cauchy-prod-low-right*:
assumes *d*: *seq-carrier n d vanishes-below n q d* **and** *g*: *seq-carrier n g*
shows *cauchy-prod n g d q* = *g 0* ∗ *d q*
⟨*proof*⟩

lemma *cauchy-prod-low-below*:
assumes *d*: *seq-carrier n d vanishes-below n q d* **and** *g*: *seq-carrier n g* **and** *p*: *p* < *q*
shows *cauchy-prod n d g p* = 0_m *n n cauchy-prod n g d p* = 0_m *n n*
⟨*proof*⟩

lemma *cauchy-prod-low-both*:
assumes *seq-carrier n d vanishes-below n q d seq-carrier n e vanishes-below n q e 0* < *q*
shows *cauchy-prod n d e q* = 0_m *n n*
⟨*proof*⟩

A palindromic formula has odd error order: agreement with e^{tX} below an even degree q extends to degree q.

lemma *sym-order*:
assumes *Ys*: $\forall\, Y \in$ *set Ys*. $Y \in$ *carrier-mat n n* **and** *X*: $X \in$ *carrier-mat n n* **and** *pal*: *rev Ys* $=$ *Ys* **and** *ord*: $\forall\, p{<}q$. *prod-coeff n Ys p* $=$ *exp-coeff X p* **and** *q*: *even q 0* $<$ *q*
shows *prod-coeff n Ys q* $=$ *exp-coeff X q*
⟨*proof*⟩

Perturbations of degree q add: if $A_0 = B_0 = I$ and D_a, D_b vanish below q, then up to degree q

$$(A + D_a) * (B + D_b) = A * B + D_a + D_b.$$

lemma *pert2*:
assumes *A*: *seq-carrier n A A 0* $= 1_m$ *n* **and** *B*: *seq-carrier n B B 0* $= 1_m$ *n* **and** *Da*: *seq-carrier n Da vanishes-below n q Da* **and** *Db*: *seq-carrier n Db vanishes-below n q Db* **and** *q*: *0* $<$ *q* **and** *p*: $p \le q$
shows *cauchy-prod n* (λ*p*. *A p* + *Da p*) (λ*p*. *B p* + *Db p*) *p* $=$ *cauchy-prod n A B p* + *Da p* + *Db p*
⟨*proof*⟩

lemma *cauchy-prod-congr*:
assumes $\forall\, p{\le}q$. *f p* $=$ *f′ p* $\forall\, p{\le}q$. *g p* $=$ *g′ p p* $\le$ *q*
shows *cauchy-prod n f g p* $=$ *cauchy-prod n f′ g′ p*
⟨*proof*⟩

The key step of Suzuki's recursion. If the formula Φ (with factors Y) agrees with e^{tX} below degree q, then the formula made of copies of Φ with the factors scaled by $c_1, \ldots, c_k$ satisfies, up to degree q,

$$\mathrm{coeff}_p = \frac{\big((\sum_j c_j)X\big)^p}{p!} + \Big(\sum_j c_j^p\Big)\big(\mathrm{coeff}_p(\Phi) - X^p/p!\big),$$

so the error at degree q vanishes when $\sum_j c_j^q = 0$.

lemma *scaled-copies*:
assumes *Ys*: $\forall\, Y \in$ *set Ys*. $Y \in$ *carrier-mat n n* **and** *X*: $X \in$ *carrier-mat n n* **and** *ord*: $\forall\, p{<}q$. *prod-coeff n Ys p* $=$ *exp-coeff X p* **and** *q*: *0* $<$ *q*
shows $\forall\, p{\le}q$. *prod-coeff n* (*concat* (*map* (λ*c*. *map* ((·$_m$) *c*) *Ys*) *cs*)) *p*
$=$ *exp-coeff* (*sum-list cs* ·$_m$ *X*) *p* + (∑ *c*←*cs*. *c* ^ *p*) ·$_m$ (*prod-coeff n Ys p* − *exp-coeff X p*)
⟨*proof*⟩

The constant term of a product is I and its linear term the sum of the factors; the sum of the factors is additive, symmetric under reversal, and linear under scaling.

lemma *prod-coeff-zero*: $\forall\, Y \in$ *set Ys*. $Y \in$ *carrier-mat n n* $\Longrightarrow$ *prod-coeff n Ys 0* $= 1_m$ *n*

⟨*proof*⟩

lemma *prod-coeff-one*: ∀ *Y* ∈ *set Ys*. *Y* ∈ *carrier-mat n n* ⟹ *prod-coeff n Ys 1* = *hamiltonian-sum n Ys*
⟨*proof*⟩

lemma *hamiltonian-sum-append*: ∀ *Y* ∈ *set Ys*. *Y* ∈ *carrier-mat n n* ⟹ ∀ *Y* ∈ *set Zs*. *Y* ∈ *carrier-mat n n* ⟹ *hamiltonian-sum n* (*Ys* @ *Zs*) = *hamiltonian-sum n Ys* + *hamiltonian-sum n Zs*
⟨*proof*⟩

lemma *hamiltonian-sum-rev*: ∀ *Y* ∈ *set Ys*. *Y* ∈ *carrier-mat n n* ⟹ *hamiltonian-sum n* (*rev Ys*) = *hamiltonian-sum n Ys*
⟨*proof*⟩

lemma *hamiltonian-sum-scale*: ∀ *Y* ∈ *set Ys*. *Y* ∈ *carrier-mat n n* ⟹ *hamiltonian-sum n* (*map* ((·$_m$) *c*) *Ys*) = *c* ·$_m$ *hamiltonian-sum n Ys*
⟨*proof*⟩

A product formula is a list of factors (a, H) standing for e^{iatH}; its coefficients are those of the matrices *factor-gen* $(a,\ H) = iaH$. The symmetric Strang formula $e^{itH_1/2}\cdots e^{itH_m/2}e^{itH_m/2}\cdots e^{itH_1/2}$ and Suzuki's recursion $S_{2k+2} = S_{2k}(u)^2\, S_{2k}(1-4u)\, S_{2k}(u)^2$ with $4u^q + (1-4u)^q = 0$, $q = 2k+3$.

definition *factor-gen* :: *real* × *complex Matrix.mat* ⇒ *complex Matrix.mat* **where**
factor-gen x = (i ∗ *complex-of-real* (*fst x*)) ·$_m$ *snd x*

definition *strang* :: *complex Matrix.mat list* ⇒ (*real* × *complex Matrix.mat*) *list* **where**
strang Hs = *map* (λ*H*. (*1* / *2*, *H*)) *Hs* @ *rev* (*map* (λ*H*. (*1* / *2*, *H*)) *Hs*)

definition *suzuki-weight* :: *nat* ⇒ *real* **where**
suzuki-weight q = *1* / (*4* − *root q 4*)

definition *suzuki-weights* :: *nat* ⇒ *real list* **where**
suzuki-weights q = (*let u* = *suzuki-weight q in* [*u*, *u*, *1* − *4* ∗ *u*, *u*, *u*])

fun *suzuki* :: *nat* ⇒ *complex Matrix.mat list* ⇒ (*real* × *complex Matrix.mat*) *list* **where**
suzuki 0 Hs = *strang Hs*
| *suzuki* (*Suc k*) *Hs* = *concat* (*map* (λ*c*. *map* (λ*x*. (*c* ∗ *fst x*, *snd x*)) (*suzuki k Hs*)) (*suzuki-weights* (*2* ∗ *k* + *3*)))

Suzuki's weights: for odd $q \geq 3$, $u = 1/(4-4^{1/q})$ satisfies $4u^q + (1-4u)^q = 0$, and the weights $u, u, 1-4u, u, u$ add up to 1.

lemma *suzuki-weight-cond*:
assumes *q*: *odd q 3* ≤ *q*
shows *4* ∗ *suzuki-weight q* ^ *q* + (*1* − *4* ∗ *suzuki-weight q*) ^ *q* = *0*
⟨*proof*⟩

Scaling a factor scales its generator; the formulas consist of factors of the given Hamiltonians, are $n \times n$, and are palindromes:

lemma *factor-gen-scale*: *factor-gen* $(c * a, H)$ = *complex-of-real* $c \cdot_m$ *factor-gen* (a, H)
⟨*proof*⟩

lemma *suzuki-snd*: $x \in$ *set* (*suzuki k Hs*) $\Longrightarrow$ *snd* $x \in$ *set Hs*
⟨*proof*⟩

lemma *suzuki-carrier*: $\forall H \in$ *set Hs*. $H \in$ *carrier-mat n n* $\Longrightarrow \forall Y \in$ *set* (*map factor-gen* (*suzuki k Hs*)). $Y \in$ *carrier-mat n n*
⟨*proof*⟩

lemma *suzuki-palindrome*: *rev* (*suzuki k Hs*) = *suzuki k Hs*
⟨*proof*⟩

The Strang formula agrees with e^{itH}, $H = \sum_j H_j$, up to degree 2:

lemma *strang-order*:
assumes *Hs*: $\forall H \in$ *set Hs*. $H \in$ *carrier-mat n n*
shows $\forall p{\leq}2$. *prod-coeff n* (*map factor-gen* (*strang Hs*)) *p* = *exp-coeff* (i $\cdot_m$ *hamiltonian-sum n Hs*) *p*
⟨*proof*⟩

By induction on k, using the palindrome property (odd order) and *scaled-copies* with $\sum_j c_j = 1$ and $\sum_j c_j^{2k+3} = 0$:

$$\mathrm{coeff}_p(S_{2k+2}) = \frac{(iH)^p}{p!} \qquad (p \le 2k+2).$$

theorem *suzuki-order*:
assumes *Hs*: $\forall H \in$ *set Hs*. $H \in$ *carrier-mat n n*
shows $\forall p{\leq}2 * k + 2$. *prod-coeff n* (*map factor-gen* (*suzuki k Hs*)) *p* = *exp-coeff* (i $\cdot_m$ *hamiltonian-sum n Hs*) *p*
⟨*proof*⟩

7 Higher-order product formulas

Suzuki's formula of order $2k{+}2$ over r steps is within $r(2|F|{+}1)(\Lambda|t|/r)^{2k+3}e^{\Lambda|t|/r}/(2k{+}3)!$ of e^{itH} (*suzuki-error*), so $r \sim |t|^{1+1/(2k+2)}$ steps suffice. The proof compares both sides with their Taylor polynomials of degree $2k+2$, which agree by *suzuki-order*.

Two facts about factorials and the exponential series: $i!\,k! \le (i+k)!$, and

$$\Big|e^z - \sum_{q \le P} \frac{z^q}{q!}\Big| \le \frac{|z|^{P+1}e^{|z|}}{(P+1)!}.$$

lemma *fact-mult-le-fact-add*: *fact i * fact k ≤ (fact (i + k) :: real)*
⟨*proof*⟩

lemma *exp-tail*:
fixes *z* :: *complex*
shows *cmod (exp z − (∑ q≤P. z ^ q / fact q)) ≤ cmod z ^ Suc P * exp (cmod z) / fact (Suc P)*
⟨*proof*⟩

A sum of matrices that act on v by scalars acts by the sum of the scalars.

lemma *matrix-sum-eigen*:
fixes *F :: nat ⇒ complex Matrix.mat*
assumes *F*: *⋀q. q < m ⟹ F q ∈ carrier-mat n n* **and** *v*: *dim-vec v = n*
and *ev*: *⋀q. q < m ⟹ F q *v v = c q ·v v*
shows *matrix-sum n F m *v v = (∑ q<m. c q) ·v v*
⟨*proof*⟩

The Taylor polynomial $\sum_{q \le P} (isH)^q/q!$ of e^{isH}. On an eigenvector it is the Taylor polynomial of $e^{is\lambda}$, so it is within $(|s|L)^{P+1}e^{|s|L}/(P+1)!$ of e^{isH} when $\|Hv\| \le L\|v\|$ (*evolution-taylor*).

definition *exp-taylor :: nat ⇒ complex Matrix.mat ⇒ real ⇒ nat ⇒ complex Matrix.mat* **where**
*exp-taylor n H s P = matrix-sum n (λq. exp-coeff ((i * complex-of-real s) ·m H) q) (Suc P)*

lemma *exp-taylor-carrier*: *H ∈ carrier-mat n n ⟹ exp-taylor n H s P ∈ carrier-mat n n*
⟨*proof*⟩

lemma *exp-taylor-eigen*:
assumes *H*: *H ∈ carrier-mat n n* **and** *v*: *dim-vec v = n* **and** *ev*: *H *v v = complex-of-real l ·v v*
shows *exp-taylor n H s P *v v = (∑ q≤P. (i * complex-of-real (s * l)) ^ q / fact q) ·v v*
⟨*proof*⟩

lemma *evolution-taylor*:
assumes *H*: *herm-bounded n L H* **and** *L*: *0 ≤ L*
shows *evolution H s ≈(|s| * L) ^ Suc P * exp (|s| * L) / fact (Suc P) exp-taylor n H s P*
⟨*proof*⟩

lemma *square-reindex*:
fixes *G :: nat ⇒ nat ⇒ 'a::comm-monoid-add*
shows *(∑ q≤P. ∑ r≤Q. G q r) = (∑ p≤P + Q. ∑ q≤p. if q ≤ P ∧ p − q ≤ Q then G q (p − q) else 0)*
⟨*proof*⟩

Evaluating a series up to degree P, $\sum_{p \le P} t^p f_p$, and truncating it at degree P.

Evaluation turns Cauchy products of truncations into products of matrices:

$$\Big(\sum_{p\le P} t^p f_p\Big)\Big(\sum_{q\le Q} t^q g_q\Big) = \sum_{p\le P+Q} t^p\,(f^{\le P} * g^{\le Q})_p.$$

definition *poly-eval* :: *nat* ⇒ (*nat* ⇒ *complex Matrix.mat*) ⇒ *nat* ⇒ *real* ⇒ *complex Matrix.mat* **where**
poly-eval n f P t = *matrix-sum n* (λ*p*. (*complex-of-real t* ^ *p*) ·$_m$ *f p*) (*Suc P*)

definition *truncate-seq* :: *nat* ⇒ *nat* ⇒ (*nat* ⇒ *complex Matrix.mat*) ⇒ *nat* ⇒ *complex Matrix.mat* **where**
truncate-seq n P f p = (*if p* ≤ *P then f p else* 0$_m$ *n n*)

lemma *truncate-seq-carrier* [*simp*]: *seq-carrier n f* ⟹ *seq-carrier n* (*truncate-seq n P f*)
⟨*proof*⟩

lemma *poly-eval-carrier*: *seq-carrier n f* ⟹ *poly-eval n f P t* ∈ *carrier-mat n n*
⟨*proof*⟩

lemma *poly-eval-index*:
assumes *seq-carrier n f i* < *n j* < *n*
shows *poly-eval n f P t* $$ (*i*, *j*) = (∑ *p*≤*P*. *complex-of-real t* ^ *p* ∗ *f p* $$ (*i*, *j*))
⟨*proof*⟩

lemma *poly-eval-mult*:
assumes *f*: *seq-carrier n f* **and** *g*: *seq-carrier n g*
shows *poly-eval n f P t* ∗ *poly-eval n g Q t* = *poly-eval n* (*cauchy-prod n* (*truncate-seq n P f*) (*truncate-seq n Q g*)) (*P* + *Q*) *t*
⟨*proof*⟩

op-bound n c M: $\|Mv\| \le c\|v\|$ for all v. Bounds multiply over products, scale, add over sums, and give $\|X^p/p!\| \le a^p/p!$ and $\|(f*g)_p\| \le \sum_{q\le p} c_q\, c'_{p-q}$.

definition *op-bound* :: *nat* ⇒ *real* ⇒ *complex Matrix.mat* ⇒ *bool* **where**
op-bound n c M ⟷ (∀ *v*. *dim-vec v* = *n* ⟶ ‖*M* ∗$_v$ *v*‖ ≤ *c* ∗ ‖*v*‖)

lemma *op-boundD*: *op-bound n c M* ⟹ *dim-vec v* = *n* ⟹ ‖*M* ∗$_v$ *v*‖ ≤ *c* ∗ ‖*v*‖
⟨*proof*⟩

lemma *op-bound-zero*: *op-bound n* 0 (0$_m$ *n n*)
⟨*proof*⟩

lemma *op-bound-mult*:
assumes *A*: *A* ∈ *carrier-mat n n op-bound n a A* **and** *B*: *B* ∈ *carrier-mat n n op-bound n b B* **and** *a*: 0 ≤ *a*
shows *op-bound n* (*a* ∗ *b*) (*A* ∗ *B*)
⟨*proof*⟩

lemma *op-bound-smult*: $A \in$ *carrier-mat n n* $\Longrightarrow$ *op-bound n a A* $\Longrightarrow$ *op-bound n* (*cmod c* $*$ *a*) (*c* $\cdot_m$ *A*)
⟨*proof*⟩

lemma *op-bound-matrix-sum*:
assumes $\bigwedge q.\ q < m \Longrightarrow F\ q \in$ *carrier-mat n n* $\bigwedge q.\ q < m \Longrightarrow$ *op-bound n* (*c q*) (*F q*)
shows *op-bound n* ($\sum q{<}m.\ c\ q$) (*matrix-sum n F m*)
⟨*proof*⟩

lemma *op-bound-pow*:
assumes *A*: $A \in$ *carrier-mat n n op-bound n a A* **and** *a*: $0 \le a$
shows *op-bound n* (*a* ^ *q*) (*A* ^$_m$ *q*)
⟨*proof*⟩

lemma *exp-coeff-op-bound*:
assumes *Y*: $Y \in$ *carrier-mat n n op-bound n a Y* **and** *a*: $0 \le a$
shows *op-bound n* (*a* ^ *q* / *fact q*) (*exp-coeff Y q*)
⟨*proof*⟩

$$\sum_{q \le p} \frac{a^q\, b^{p-q}}{q!\,(p-q)!} = \frac{(a+b)^p}{p!}.$$

lemma *binomial-exp-coef*:
fixes *a b* :: *'a*::*field-char-0*
shows ($\sum q{\le}p.\ a$ ^ $q * b$ ^ $(p - q)$ / $(fact\ q * fact\ (p - q))) = (a + b)$ ^ p / *fact p*
⟨*proof*⟩

lemma *op-bound-mono*: *op-bound n c M* $\Longrightarrow$ $c \le c' \Longrightarrow$ *op-bound n c' M*
⟨*proof*⟩

lemma *cauchy-prod-op-bound*:
assumes *f*: *seq-carrier n f* **and** *g*: *seq-carrier n g* **and** *bf*: $\bigwedge q.$ *op-bound n* (*cf q*) (*f q*) **and** *bg*: $\bigwedge q.$ *op-bound n* (*cg q*) (*g q*)
and *cf*: $\bigwedge q.\ 0 \le cf\ q$
shows *op-bound n* ($\sum q{\le}p.\ cf\ q * cg\ (p - q)$) (*cauchy-prod n f g p*)
⟨*proof*⟩

The total weight $\Lambda = L \sum_{(a,H)\in F} |a|$ of a formula. The p-th coefficient of the formula is bounded by $\Lambda^p/p!$ (*prod-coeff-op-bound*).

definition *total-weight* :: *real* ⇒ (*real* × *complex Matrix.mat*) *list* ⇒ *real* **where**
total-weight L F = ($\sum x{\leftarrow}F.\ |fst\ x|$) $* L$

lemma *total-weight-nonneg*: $0 \le L \Longrightarrow 0 \le$ *total-weight L F*
⟨*proof*⟩

lemma *factor-gen-carrier*: *herm-bounded n L* (*snd x*) $\Longrightarrow$ *factor-gen x* $\in$ *carrier-mat n n*
⟨*proof*⟩

lemma *factor-gen-op-bound*:
assumes *herm-bounded n L* (*snd x*)
shows *op-bound n* (|*fst x*| $*$ *L*) (*factor-gen x*)
⟨*proof*⟩

lemma *prod-coeff-op-bound*:
assumes *F*: $\forall x \in$ *set F*. *herm-bounded n L* (*snd x*) **and** *L*: *0* $\leq$ *L*
shows *op-bound n* (*total-weight L F* ^ *p* / *fact p*) (*prod-coeff n* (*map factor-gen F*) *p*)
⟨*proof*⟩

The operator of a formula, $\prod_{(a,H)\in F} e^{iatH}$; it is unitary.

fun *formula-product* :: *nat* $\Rightarrow$ *real* $\Rightarrow$ (*real* $\times$ *complex Matrix.mat*) *list* $\Rightarrow$ *complex Matrix.mat* **where**
formula-product n t [] = 1_m *n*
| *formula-product n t* (*x* # *F*) = *evolution* (*snd x*) (*fst x* $*$ *t*) $*$ *formula-product n t F*

lemma *formula-product-carrier-unitary*:
assumes $\forall x \in$ *set F*. *herm-bounded n L* (*snd x*)
shows *formula-product n t F* $\in$ *carrier-mat n n* $\wedge$ *Quantum.unitary* (*formula-product n t F*)
⟨*proof*⟩

lemma *exp-taylor-poly-eval*:
assumes *H*: *H* $\in$ *carrier-mat n n*
shows *exp-taylor n H* (*a* $*$ *t*) *P* = *poly-eval n* (*exp-coeff* (*factor-gen* (*a*, *H*))) *P t*
⟨*proof*⟩

lemma *poly-eval-cong*: ($\bigwedge p$. *p* $\leq$ *P* $\Longrightarrow$ *f p* = *g p*) $\Longrightarrow$ *poly-eval n f P t* = *poly-eval n g P t*
⟨*proof*⟩

Splitting a polynomial at degree P, and the tail bound $\sum_{p<Q} z^{K+p}/(K+p)! \leq z^K e^z/K!$ for $z \geq 0$.

lemma *poly-eval-split*:
assumes *f*: *seq-carrier n f*
shows *poly-eval n f* (*P* + *Q*) *t* = *poly-eval n f P t* + *matrix-sum n* (λp. (*complex-of-real t* ^ (*Suc P* + *p*)) $\cdot_m$ *f* (*Suc P* + *p*)) *Q*
⟨*proof*⟩

lemma *partial-tail*:
fixes *z* :: *real*
assumes *z*: *0* $\leq$ *z*
shows ($\sum p$<*Q*. *z* ^ (*K* + *p*) / *fact* (*K* + *p*)) $\leq$ *z* ^ *K* $*$ *exp z* / *fact K*

⟨*proof*⟩

lemma *poly-eval-op-bound*:
assumes *f*: *seq-carrier n f* **and** *b*: ⋀*p*. *op-bound n* (*c p*) (*f p*)
shows *op-bound n* (∑ *p*≤*P*. |*t*| ^ *p* ∗ *c p*) (*poly-eval n f P t*)
⟨*proof*⟩

lemma *poly-eval-seq-one*: *poly-eval n* (*seq-one n*) *P t* = 1_m *n*
⟨*proof*⟩

A formula is close to its own Taylor polynomial: with $|F|$ factors and weight Λ,

$$\Big\| \prod_{(a,H)\in F} e^{iatH} - \sum_{p\le P} t^p \operatorname{coeff}_p \Big\| \le 2|F| \, \frac{(\Lambda|t|)^{P+1} e^{\Lambda|t|}}{(P+1)!}.$$

theorem *formula-product-approx*:
assumes *F*: ∀ *x* ∈ *set F*. *herm-bounded n L* (*snd x*) **and** *L*: *0* ≤ *L*
shows *formula-product n t F* ≈$_{2 * real\ (length\ F) * (total\text{-}weight\ L\ F * |t|)\ \hat{}\ Suc\ P * exp\ (total\text{-}weight\ L\ F * |t|)}$ *poly-eval n* (*prod-coeff n* (*map factor-gen F*)) *P t*
⟨*proof*⟩

Closeness is transitive, the errors adding up, and monotone in the error.

lemma *op-close-trans*:
assumes *U* ≈$_a$ *V V* ≈$_b$ *W U* ∈ *carrier-mat n n V* ∈ *carrier-mat n n W* ∈ *carrier-mat n n*
shows *U* ≈$_{a + b}$ *W*
⟨*proof*⟩

lemma *op-close-mono*:
assumes *U* ≈$_a$ *V a* ≤ *b*
shows *U* ≈$_b$ *V*
⟨*proof*⟩

For Suzuki's formula: the sum of m Hamiltonians bounded by L is bounded by mL, the formula has $2\cdot 5^k m$ factors, its weights add up to m, and $\Lambda \ge mL$.

lemma *hamiltonian-sum-herm-bounded*:
∀ *H* ∈ *set Hs*. *herm-bounded n L H* ⟹ *herm-bounded n* (*real* (*length Hs*) ∗ *L*) (*hamiltonian-sum n Hs*)
⟨*proof*⟩

lemma *length-suzuki*: *length* (*suzuki k Hs*) = *2* ∗ *5* ^ *k* ∗ *length Hs*
⟨*proof*⟩

lemma *sum-suzuki*: (∑ *x*←*suzuki k Hs*. *fst x*) = *real* (*length Hs*)
⟨*proof*⟩

lemma *total-weight-suzuki*: *0* ≤ *L* ⟹ *real* (*length Hs*) ∗ *L* ≤ *total-weight L* (*suzuki k Hs*)

⟨*proof*⟩

One Suzuki step: $\|S_{2k+2}(\delta) - e^{i\delta H}\| \leq (2|F|+1)(\Lambda|\delta|)^{2k+3} e^{\Lambda|\delta|}/(2k+3)!$.

lemma *suzuki-step*:
fixes k :: *nat* **and** d :: *real*
assumes *Hs*: $\forall H \in$ *set Hs*. *herm-bounded n L H* **and** L: $0 \leq L$
defines $F \equiv$ *suzuki k Hs* **and** $q \equiv 2 * k + 3$
shows *formula-product n d F* $\approx_{(2 * real\ (length\ F) + 1) * (total\text{-}weight\ L\ F * |d|)\ \hat{}\ q * exp\ (total\text{-}weight\ L\ F *}$ |
evolution (*hamiltonian-sum n Hs*) *d*
⟨*proof*⟩

Suzuki's formula of order $2k+2$ over r steps: $\|S_{2k+2}(t/r)^r - e^{itH}\| \leq r(2|F|+1)(\Lambda|t|/r)^{2k+3} e^{\Lambda|t|/r}/(2k+3)!$, with $|F| = 2 \cdot 5^k m$ factors and $\Lambda \geq mL$.

theorem *suzuki-error*:
fixes k :: *nat* **and** t :: *real*
assumes *Hs*: $\forall H \in$ *set Hs*. *herm-bounded n L H* **and** L: $0 \leq L$ **and** r: $0 < r$
defines $F \equiv$ *suzuki k Hs* **and** $q \equiv 2 * k + 3$
shows *formula-product n* (*t* / *real r*) $F\ \hat{}_m\ r \approx_{real\ r * ((2 * real\ (length\ F) + 1) * (total\text{-}weight\ L\ F * |t| / real\ r}$
evolution (*hamiltonian-sum n Hs*) *t*
⟨*proof*⟩

References

[1] D. W. Berry, G. Ahokas, R. Cleve, and B. C. Sanders. Efficient quantum algorithms for simulating sparse Hamiltonians. *Communications in Mathematical Physics*, 270(2):359–371, 2007.

[2] A. Bordg, H. Lachnitt, and Y. He. Isabelle marries Dirac: a library for quantum computation and quantum information. Archive of Formal Proofs, https://isa-afp.org/entries/Isabelle_Marries_Dirac.html, Nov. 2020.

[3] J. Liu, B. Zhan, S. Wang, S. Ying, T. Liu, Y. Li, M. Ying, and N. Zhan. Quantum Hoare logic. Archive of Formal Proofs, https://isa-afp.org/entries/QHLProver.html, Mar. 2019.

[4] S. Lloyd. Universal quantum simulators. *Science*, 273(5278):1073–1078, 1996.

[5] M. Suzuki. General theory of fractal path integrals with applications to many-body theories and statistical physics. *Journal of Mathematical Physics*, 32(2):400–407, 1991.

[6] R. Thiemann and A. Yamada. Matrices, Jordan normal forms, and spectral radius theory. Archive of Formal Proofs, https://isa-afp.org/entries/Jordan_Normal_Form.html, Aug. 2015.

[7] H. F. Trotter. On the product of semi-groups of operators. *Proceedings of the American Mathematical Society*, 10(4):545–551, 1959.